# فهرست مطالب

## فصل اول : مفاهیم شبکه‌های کامپیوتری

## فصل دوم : لایهٔ واسط شبکه

## فصل سوم : لایهٔ IP در شبکهٔ اینترنت



## فصل چهارم : مسیریابی در شبکهٔ اینترنت

## فصل پنجم : لایهٔ انتقال در شبکهٔ اینترنت

## فصل ششم : سرویس‌دهنده‌های نام حوزه و اصول مدیریت شبکه

## فصل هفتم : برنامه‌نویسی تحت شبکهٔ اینترنت (Socket Programming)

## فصل هشتم : پروتکل TelNet و پروتکلهای انتقال فایل

## فصل نهم : سیستم پست الکترونیکی در شبکهٔ اینترنت



## فصل دهم : تور جهان‌گستر و پروتکل HTTP

## فصل یازدهم : امنیت اطلاعات در شبکه

## ۱) مقدمه

شبکه‌های کامپیوتری از دیدگاه اجتماعی یک پدیدهٔ فرهنگی و از دید مهندسی کامپیوتر یک تخصّص و علم تلقی می‌شود. امروزه پیشرفت و توسعهٔ مرزهای دانش به گسترش شبکه‌های کامپیوتری وابسته شده است. هدف اصلی در "فناوری اطلاعات"[۱] –IT– ، گردآوری ، سازماندهی و فرآوری داده‌ها و دانش پراکنده در سطح دنیا است به گونه‌ای که بتوان از این دانش گردآوری‌شده ، معرفت و دانش جدید تولید کرد. بالطبع مؤثرترین ابزار برای جمع‌آوری ، سازماندهی و پردازش داده‌های پراکنده ، شبکه‌های کامپیوتری است.

شبکهٔ کامپیوتری ، مجموعه‌ای از کامپیوترهای "مستقل" است که به نحوی با یکدیگر اطلاعات و داده مبادله می‌نمایند. در این تعریف کوتاه مفاهیم گرانباری نهفته است :

- **"استقلال"** کامپیوترها در یک شبکه بدان معنا است که هر ماشین می‌تواند حتی بدون حضور در شَبکه کار کرده و از شبکه فقط برای تبادل داده‌ها استفاده کند. با استناد به این تعریف ، ماهیت شبکه‌های کامپیوتری از "سیستمهای توزیع‌شده"[۲] یا "سیستمهای کامپیوتری چندکاربرهٔ بزرگ"[۳] کاملاً تفکیک می‌شود.

در یک سیستم توزیع شده ، هر ایستگاه اگرچه کامپیوتری با سخت‌افزار کامل محسوب می‌شود ولی مستقل نیست. در این سیستمها یک "وظیفهٔ ویژه"[۴] بین ایستگاهها تقسیم‌شده و هر ایستگاه قسمتی از آن را انجام می‌دهد. مجموعهٔ پردازشی که کل ایستگاهها انجام می‌دهند ، فرآیند کار را تکمیل می‌نماید. عدم عملکرد صحیح یکی از ایستگاهها منجر به ناقص ماندن کل کار شده و نتیجهٔ پردازش بقیهٔ ایستگاهها را بی‌تاثیر خواهد کرد. از دیدگاه یک کاربر بیرونی کل یک سیستم توزیع‌شده ، بصورت واحد و یکپارچه دیده می‌شود؛ سیستم عامل موظف است یک کار را بین ایستگاهها تقسیم کند و سپس نتایج را جمع‌آوری کرده و آن کار را تکمیل نماید. از این نوع سیستمها در راه‌اندازی خطوط تولید کارخانه‌ها ، کنترل روباتهای صنعتی ، فیلم‌سازی و مواردی از این قبیل استفاده می‌شود.

- **"تبادل داده"** بدین معناست که کامپیوترها در شبکه بتوانند با یکدیگر داده رد وبدل کنند بدون آنکه نوع کانال اهمیت داشته باشد. یعنی هیچ محدودیتی بر روی کانال فیزیکی انتقال ، وجود ندارد؛ کامپیوترها چه از طریق خط تلفن مرتبط شوند و چه از طریق کانالهای فیبر نوری ، در هر حال شبکه تلقی می‌شوند.

---

[۱] Information Technology
[۲] Distributed Systems
[۳] Multi-user Mainframe Systems
[۴] Special Task

تا چند سال قبل ، شبکه‌ها بیشتر موجودیت سازمانی و مستقل داشتند ، بدین معنا که یک شبکه برای رفع نیاز یک گروه یا تشکیلات ، پیاده و نصب می‌شد و تکنولوژی سخت‌افزار ارتباطی ، متناسب با نوع نیاز آن سازمان انتخاب می‌گردید. با وارد شدن شبکه‌های کامپیوتری به سازمانها و تشکیلات اداری و اقتصادی ، نقش پر ارزش آنها در کاهش "گردش کاغذ" و نامه‌نگاریهای بیهوده آشکار شد. صرفه‌جویی در نیروی انسانی و بکارگیری دانش انسان بجای قدرت بدنی ، افزایش سرعت انجام کارها ، دقت بی‌نظیر و کاهش در هزینه‌های جاری ، هدیهٔ علم شبکه‌های کامپیوتری به سیستم مدیریت سنتی بود. به فاصلهٔ زمانی اندک ، اتوماسیون اداری جزو ساختار مدیریت جدید شد و نقش کاغذ در مدیریت سازمانها آرام‌آرام رو به کاهش گذاشت.

با رشد تکنولوژی ، سخت‌افزار لازم برای ارتباطات بین‌شبکه‌ای فراهم و ارزان شد. از طرفی دولتهای غربی سرمایه‌گذاری لازم را برای ایجاد یک زیرساخت ارتباطی در شبکهٔ داده ، انجام دادند. این دو عامل شرایط را برای اتصال شبکه‌های مستقل به یکدیگر و رشد هماهنگ آنها فراهم کرد و به تدریج "شبکهٔ اینترنت" بدون آنکه یک متولی انحصاری داشته باشد ، شکل گرفت. این شبکه که کل جهان را تحت تاثیر قرار داده ، حاصل رشد سریع ، متعادل و خودجوش شبکه‌های مستقلی است که توسط مجموعه‌ها و نهادهای مختلف نصب و راه‌اندازی شده‌اند.

اینترنت مجموعه‌ای از شبکه‌های مستقل و مرتبط با یکدیگر است که ارتباطات همگانی را میسر کرده است. "تکنولوژی اینترنت" نیز مجموعه‌ای از مکانیزمها و استانداردهای ارتباطی است که در خلال این سالها سعی بر آن داشته تا روشی را عرضه کند که شبکه‌های مختلف بتوانند فارغ از جزییات سخت‌افزاری و نرم‌افزاری ، با یکدیگر مبادلهٔ داده داشته باشند. اینترنت در یک رشد هماهنگ با علوم مخابرات و کامپیوتر به ناگاه تبدیل به "شبکه‌ای از کل شبکه‌های جهان" شد. به گونه‌ای که تخمین زده می‌شود در هر روز بیش از ۳۷ میلیون نفر در ۱۱۰ کشور دنیا حداقل یکبار از اینترنت استفاده می‌کنند. با توجه بدانکه رشد اینترنت به دستگاه یا دولت خاصی وابسته نیست ، هیچ شاخص معینی برای رشد اینترنت وجود ندارد. یک فرد حقیقی نیز می‌تواند به نوبهٔ خود اینترنت را توسعه بدهد ، مثلاً یک هنرمند یا بازرگان می‌تواند با سرمایه‌گذاری و نصب شبکهٔ محلی و ارائهٔ آگاهی و دانش ، به سهم خود به گسترش اینترنت کمک کند و هیچ محدودیتی برای سرمایه‌گذاری او وجود ندارد.

تنها "دانش" در اینترنت عرضه نمی‌شود بلکه چیزی که در اینترنت وجود دارد در یک مفهوم عام "دانایی و آگاهی" است؛ به عنوان مثال آگاهی از نرخ سهام یا نتیجهٔ یک مسابقه یا

اخبار حوادث را شاید نتوان در حوزهٔ دانش و علوم طبقه‌بندی کرد ولی نوعاً "آگاهی" محسوب می‌شود.

مجتمع شدن صدا و تصویر در کنار داده‌ها ، امکان ارتباط دوطرفه و انتخاب موضوع ، اینترنت را به مکانی تبدیل کرد که میلیونها دوست و غریبه در کنار هم گرد آمده و اهداف خود را در ورای محدودیتهای جغرافیایی دنبال کنند. شاید در دهه‌های آینده بحثی به نام "جامعه‌شناسی دهکدهٔ اینترنت" مطرح شود.

سال ۱۹۹۲ یکی از پرفروغترین سالهای گسترش اینترنت بود. تا قبل از این تاریخ ، بیشتر افراد به تناسب نوع نیاز خود از شبکه استفاده می‌کردند. با عرضهٔ محیط تور جهانگستر یا **"وب"**[۱] در این سال توجه مردم عادی نیز به اینترنت جلب شد و بدینگونه از اینترنت یک رسانه ، پدید آمد. وب روشی برای سازماندهی اطلاعات است به‌گونه‌ای که غیر از کنار هم قرار دادن متن ، صدا ، تصویر ، فیلم و گرافیک ، بزرگترین حسن آن سادگی دسترسی به اطلاعات پراکنده در دنیا ، از طریق مفهومی به نام "ابرپیوند"[۲] است. در فصول آینده این مفهوم را باز خواهیم کرد.

توسعهٔ وب بر روی اینترنت ، سازمانها را بر آن داشت که سیستم شبکه‌های داخلی خود را متحول کنند. **"اینترانت"**[۳] زادهٔ همین تفکر بود. اینترانت شبکه‌ای داخلی ( با تملک سازمانی و یا خصوصی ) است که از پروتکلهای مرتبط با اینترنت و مخصوصاً تکنولوژی وب برای سازماندهی شبکه استفاده می‌کنند. هیچ لزومی ندارد که اینترانت حتماً به اینترنت متصل باشد. مثلاً ادارهٔ پست می‌تواند برای خود شبکه‌ای داخلی و مستقل از شبکهٔ اینترنت تدارک ببیند ولی در پیاده‌سازی این شبکه از پروتکلهای حاکم بر اینترنت (TCP/IP/HTTP/WWW) و پایگاه دادهٔ مبتنی بر وب استفاده کند. در حقیقت اینترانت مقیاس بسیار کوچکی از کل اینترنت است ولی خصوصی است؛ بر خلاف اینترنت که هیچکس مالک آن نیست.

اگر اجزای یک شبکه ، کامپیوترها باشند ، اجزای اینترنت همین شبکه‌ها هستند که به هم متصل شده اند. برای شناخت فنّی اینترنت ابتدا باید با اجزاء آن آشنا شده و سپس اصول و قواعد حاکم بر کل آنرا بررسی کرد. در این فصل بطور مختصر اصول کلی شبکه‌های کامپیوتری را مورد بحث قرار داده‌ایم؛ برای احاطهٔ کلی بر مقولات شبکه باید به مراجع غنی و کامل مراجعه کنید.

---

[۱] World Wide Web –WWW-
[۲] Hyperlink
[۳] **I**ntr**a**net

## ۲) کاربردهای شبکه‌های کامپیوتری

بطور عام کاربرد شبکه‌های کامپیوتری را می‌توان در موارد زیر خلاصه کرد:

➢ **اشتراک منابع**[۱] : به معنای فراهم آوردن و به اشتراک گذاشتن سخت‌افزار ، نرم‌افزار و داده‌های مورد نیاز در شبکه است ، بگونه‌ای که کاربران بتوانند براحتی از این منابع استفاده کنند. به عنوان مثال یک چاپگر در یک شبکه می‌تواند در اختیار کل کاربران باشد و همه بجای جابجایی فیزیکی چاپگر ، از آن استفاده کنند. یا یک نرم افزار گرانقیمت بر روی یک ماشین خاص به نام "سرویس‌دهندهٔ فایل"[۲] نصب شود و همه بطور مشترک از آن استفاده کنند.
با توسعهٔ صنعت نرم‌افزار و سخت‌افزار و سقوط قیمتها ، امروزه این کاربردها کمرنگ‌تر شده‌اند. پرارزشترین کاربرد شبکه به اشتراک گذاشتن **"داده‌ها"** است. گاهی داده‌ها ، هزاران هزار برابر سخت‌افزار و نرم‌افزار ارزش دارند زیرا با صرف هزینه شاید بتوان سخت‌افزار و نرم‌افزار را خریداری کرد ولی چگونه می‌توان فقط بخشی از داده‌های پرارزش و بیکرانی که در یک شبکهٔ اطلاع‌رسانی مثل اینترنت وجود دارد را تهیه کرد؟

➢ **حذف محدودیتهای جغرافیایی در تبادل داده‌ها** : دهکدهٔ جهانی اینترنت ، فواصل جغرافیایی را بی‌معنا کرده است. با استفاده از شبکه‌های کامپیوتری شما می‌توانید در کسری از ثانیه به منابع اطلاعاتی موجود در فواصل هزاران کیلومتری خود دسترسی داشته باشید و یا با کاربران حاضر در شبکه ، مبادلهٔ پیام و اطلاعات بنمایید.

➢ **کاهش هزینه‌ها** : بکارگیری شبکه‌های کامپیوتری نه تنها در وقت صرفه‌جویی می‌کند بلکه هزینه‌ها را نیز در تمام جوانب کاهش می‌دهد. امروزه استفاده از پست الکترونیکی گذشته از سرعت بسیار زیاد ، عملاً رایگان است؛ یا جابجایی پول و اعتبار و خرید و فروش الکترونیکی هزینه‌ای در حد صفر دارد. استفادهٔ مجاز از نتایج تحقیقات دیگران هزینهٔ تکرار آن تجارب را حذف خواهد کرد. در شبکه‌های کوچک و سازمانی به اشتراک گذاشتن سخت‌افزار و نرم‌افزار مطمئناً هزینه‌های سازمان را کاهش خواهد داد. ( یک چاپگر یا نرم‌افزار برای ۲۰ نفر بجای ۲۰ چاپگر و نرم‌افزار برای ۲۰ نفر )

---

۱ Resources Sharing
۲ File Server

➢ **بالا رفتن قابلیت اعتماد[۱] سیستمها** : شبکه‌های کامپیوتری بگونه‌ای طراحی می‌شوند که وقتی یکی از آنها مختل شود ، بقیهٔ شبکه از هم نمی‌پاشد. گسترده بودن کانالهای انتقال در زیرساخت ارتباطی شبکه ، باعث شده که قطع یکی از کانالها منجر به از دست رفتن کل شبکه نشود. ذخیره‌سازی فایلها و بانکهای اطلاعاتی موجود در شبکهٔ یک سازمان ( مثلاً بانک ) ، بر روی چند ماشین مستقل بعنوان سیستمهای پشتیبان ، این فایده را دارد که در صورت خرابی یکی از آنها ، دیگری جایگزین آن شود ، بدون آنکه وقفه‌ای در کار کل سیستم ایجاد شود یا داده‌ای خراب شود.
کاربرد سیستمهای پشتیبان مبتنی بر شبکه یا توزیع داده‌ها بر روی ماشینهای مختلف در شبکه ، قابلیت اعتماد سیستمهایی همانند کنترل خطوط هوایی ، سیستمهای بانکی ، سیستمهای نظامی و امنیتی و کنترل رآکتورهای هسته‌ای را چندین برابر می‌کند.

➢ **افزایش کارآیی سیستم** : بهره‌گیری از شبکه می‌تواند کارآیی سیستم را افزایش بدهد بدین نحو که توزیع وظائف سازمانی یک مجموعه همانند بانک ، به ماشینهای متفاوت در شبکه (با حفظ استقلال آن ماشین) ، ضمن بالا بردن قابلیت اعتماد ، کارآیی سیستم را از لحاظ سرعت دسترسی به اطلاعات ، سرعت پردازش و ذخیره و بازیابی اطلاعات افزایش خواهد داد.

### ۱-۲) خدمات معمول در شبکه

خدمات شبکه‌های کامپیوتری بقدری فراگیر شده که تمام شئونات زندگی را تحت تاثیر قرار داده است. با توسعهٔ اینترنت یا شبکه‌های ملی ، محیط زیست می‌تواند نجات یابد؛ رفت و آمدهای بیهوده به منظور پیگیری کارهای اداری و اقتصادی از زندگی حذف شود. (کاهش آلودگی هوا). قطع درختان و تبدیل آن به کاغذ و اتلاف وقت در گرداندن این کاغذ بین ادارات و سازمانها در دنیای امروز به تدریج در حال پایان است. بسیاری از مجلات و روزنامه‌ها امروزه فقط بر روی شبکه ارائه می‌شوند و دیگر بر روی کاغذ چاپ نمی‌شوند. نقش اسکناس و اوراق بهادار نیز می‌تواند با جابجایی "اعتبار" به جای پول کاهش یابد. بانکها از طریق یک کارت اعتباری ، تمام خدماتی را که سالیان سال با مبادلهٔ اسناد و اوراق بهادار انجام می‌شد ، در کسری از ثانیه ارائه می‌کنند.

---

[۱] Reliability

خدماتی که شبکه‌ها ارائه می‌کنند بسیار وسیع و خارج از شمارش هستند ولی عمومی‌ترین نوع شبکه‌ها را می‌توان در موارد زیر عنوان کرد:

- دسترسی به بانکهای اطلاعاتی راه دور Remote Access
- پست الکترونیکی E-Mail
- خدمات انتقال فایل File Transfer
- ورود به سیستم از راه دور Remote Login
- گروههای خبری News Groups
- جستجوی اطلاعات مورد نیاز Information Seek
- تبلیغات Advertisement
- تجارت الکترونیکی E-Commerce
- بانکداری الکترونیکی E-Banking
- سرگرمی و محاوره
- مجلات و روزنامه‌های الکترونیکی
- محاورهٔ مستقیم و چهره به چهره از راه دور Face to Face Conversation
- کنفرانس از راه دور Teleconferece
- یافتن اشخاص مورد نظر در جهان People Finding
- تلفن و دورنگار از طریق شبکه
- رادیو از طریق شبکه
- آموزش از راه دور
- ارائهٔ مدون اطلاعات فنی و علمی
- اخبار مربوط به هنر ، ورزش ، سیاست ، تجارت ، بهداشت و ...
- کاریابی و اشتغال
- درمان از راه دور
- خرید و فروش روزمره با استفاده از کارت اعتباری ، شرکت در حراج
- انجمن‌های خیریه
- مشاوره از راه دور

در شکل (۱-۱) نمونه‌ای از هزاران خدمات ارائه شده در شبکهٔ اینترنت نشان داده شده است.


شکل (۱-۱) نمونه‌ای از خدمات ارائه شده در اینترنت

## ۳) سخت‌افزار شبکه

اگر بخواهیم سخت‌افزار شبکه‌های کامپیوتری را دسته‌بندی و تفکیک نماییم از دو دیدگاه می‌توان به شبکه‌ها نگاه کرد.
دیدگاه اول "تکنولوژی انتقال" در شبکه‌های کامپیوتری است؛ بدین معنا که شبکه از چه نوع کانالی بعنوان واسط انتقال استفاده می‌کند.
دیدگاه دوم در تفکیک شبکه‌ها ، مقیاس شبکه و ناحیهٔ تحت پوشش آن است. بدین معنا که شبکه چه مسافت جغرافیایی را پوشش می‌دهد و حداکثر چند ایستگاه می‌تواند در شبکه وجود داشته باشد. ( مقصود از ایستگاه ، یک ماشین همانند کامپیوتر یا چاپگر است که به عنوان یک موجودیت مستقل می‌تواند با بقیهٔ ماشینها تبادل اطلاعات نماید. )

### ۱-۳) دسته بندی شبکه‌ها از دیدگاه تکنولوژی انتقال

از دیدگاه تکنولوژی انتقال دو نوع شبکه قابل تعریف است:

- شبکه‌های پخش فراگیر یا Broadcast
- شبکه‌های نقطه به نقطه یا Point to Point

در شبکه‌های پخش فراگیر ، انتقال اطلاعات از طریق یک کانال فیزیکی که بین تمام ایستگاههای شبکه مشترک است ، انجام می‌شود. همهٔ ایستگاهها موظفند بطور دایم به خط گوش بدهند؛ برای ارسال نیز مجبورند اطلاعات را بر روی همین کانال منتقل نمایند. بنابراین در چنین شبکه‌هایی هر ایستگاه باید یک آدرس یکتا داشته باشد تا گیرندهٔ پیام بتواند از بین پیامهایی که روی شبکه مبادله می‌شود ، پیام مربوط به خودش را تشخیص داده و برای پردازشهای بعدی از روی کانال به حافظهٔ اصلی منتقل نماید.

در این نوع شبکه امکان ارسال پیامهای فراگیر وجود دارد. پیامهای فراگیر پیامهایی هستند که از مبدأ یک ایستگاه برای تمام یا یک گروه خاص از ایستگاهها ارسال شود. استفاده از کانالهای مشترک برای انتقال اطلاعات بین ایستگاههای شبکه از برخی جهات مشکل آفرین است :

- مدیریت پیچیدهٔ کانال : در این شبکه ، مدیریت کانال به نحوی که تمام ایستگاهها بتوانند در یک روال قانونمند و عادلانه ، از کانال استفاده کنند ، پیچیده است؛ زیرا در شبکه‌ها هر ایستگاه عنصری مستقل محسوب می‌شود و هیچگونه حاکمیت بیرونی بر آنها وجود ندارد لذا رعایت قانون و نوبت در استفاده از کانال برعهدهٔ خود ایستگاهها است. دقت کنید که در این نوع شبکه‌ها یک ایستگاه مجاز نیست به محض آنکه داده‌ای برای ارسال داشت ، آن را روی

کانال بفرستد بلکه باید بر طبق قواعدی که به نام "پروتکل نظارت بر واسط انتقال[۱]" مشهور است ، خودش را نوبت‌بندی کرده و سپس اقدام به ارسال نماید. ارسال همزمان دو ایستگاه در این نوع شبکه ، منجر به "تصادم"[۲] شده و در نتیجه داده‌های آنها خراب و فاقد اعتبار خواهد شد.

♦ امنیت کم : با توجه به آنکه تمام ایستگاهها موظف به گوش دادن به خط هستند بنابراین اطلاعات روی کانال مشترک توسط تمام ایستگاهها شنیده می‌شود. کافی است کسی بخواهد به اطلاعات دیگران دسترسی داشته باشد ، در این حالت ایستگاه براحتی با انتقال تمام یا بخشی از اطلاعاتِ در حال تبادل روی کانال به درون حافظهٔ اصلی خود ، آنرا در اختیار شخص بیگانه قرار می‌دهد. بهمین دلیل استفاده از شبکه‌های کانال مشترک ، برای ارسال اطلاعات محرمانه زمانی عقلانی خواهد بود که این اطلاعات قبل از ارسال رمزنگاری شده باشد.

♦ کارآیی پایین : با توجه به آنکه تمام ایستگاهها فقط یک کانال در اختیار دارند ، لذا فقط سهم کوچکی از پهنای باند کانال در اختیار یک ایستگاه قرار می‌گیرد. اگر داده‌ها در اثر بروز تصادم یا نویز ، دچار خرابی شوند وضع به مراتب بدتر خواهد شد.

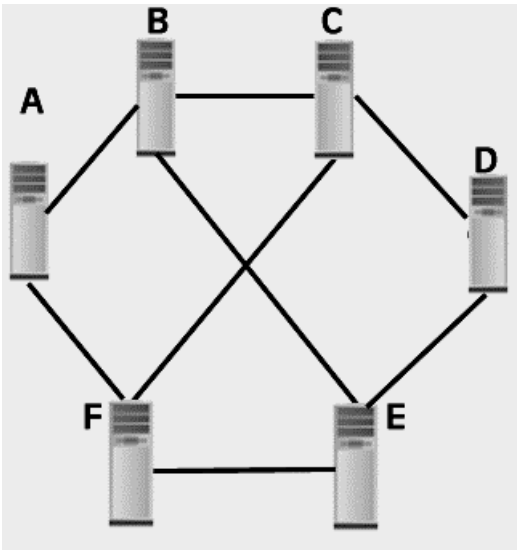
با تمام این تفاصیل استفاده از کانالهای مشترک به عنوان تکنولوژی انتقال بسیار مقرون به صرفه است و به صورت گسترده از آن استفاده می‌شود. شبکه‌های ماهواره‌ای ، شبکه‌های رادیویی ، شبکهٔ محلی BUS و شبکهٔ محلی نوع حلقه ، همگی شبکه‌های نوع "پخش فراگیر" محسوب می‌شوند.

در **شبکه‌های نقطه به نقطه** بین دو ماشین در شبکه ، یک کانال فیزیکی و مستقیم وجود دارد و هیچ ماشین دیگری به آن کانال متصل نخواهد بود. بعبارت ساده‌تر به یک کانال فیزیکی فقط و فقط دو ماشین متصل است. در شکل (۲-۱) نمونه‌ای از یک شبکهٔ نقطه به نقطه به تصویر کشیده شده است. در این مثال بین ( A و B ) و ( A و F ) ارتباط مستقیم و اختصاصی وجود دارد ولی بین ( A و D ) کانال اختصاصی دیده نمی‌شود. در این ساختار ماشین A قادر است از طریق ماشینهای واسطهٔ B و C داده‌های خود را به D برساند.

---

[۱] Medium Access Control Protocol

[۲] Collision




شکل (۲-۱) نمونه‌ای از یک شبکهٔ نقطه به نقطه

بر خلاف شبکه‌های با کانال مشترک که مسیر ارتباطی بین ایستگاهها یکتا است ، در شبکه‌های نقطه به نقطه مسیرهای گوناگونی بین دو ماشین برقرار خواهد بود لذا بحث انتخاب بهترین مسیر از بین این مسیرها مطرح خواهد شد که در فصلی مجزا به آن خواهیم پرداخت.

### ۲-۳) دسته بندی شبکه‌ها از دیدگاه مقیاس بزرگی

دیدگاه دوم در دسته‌بندی و تفکیک شبکه‌ها ، مقیاس شبکه و ناحیهٔ تحت پوشش آن می‌باشد. از این دیدگاه سه نوع شبکه تعریف می‌شود:

- شبکه‌های محلی LAN[۱]
- شبکه‌های بین شهری MAN[۲]
- شبکه‌های گسترده WAN[۳]

---

[۱] Local Area Network
[۲] Metropolitan Area Network
[۳] Wide Area Network

**۱-۲-۳) شبکه‌های محلی -LAN-**

این نوع از شبکه معمولاً در فواصل جغرافیایی محدود (حداکثر تا چند کیلومتر) و تحت تملک سازمانهای کوچک ، ادارات ، نهادها ، محیطهای آموزشی و کارخانه‌های کوچک نصب و راه‌اندازی می‌شود. کوچک بودن این نوع شبکه از لحاظ طول فیزیکی کانال انتقال و کم بودن تعداد ایستگاهها ، محاسن فراوانی را برای این شبکه به ارمغان آورده است :

- با توجه به کوتاه بودن طول کانال ، اولاً افت سیگنال کم و طبعاً نرخ خطا بسیار پایین است. ثانیاً نرخ ارسال می‌تواند بسیار بالا باشد. ( از چند مگابیت در ثانیه تا چند گیگا بیت بر ثانیه ) ثالثاً تاخیر انتشار[۱] بسیار ناچیز خواهد بود. مجموعهٔ این عوامل باعث خواهد شد تا سرعت مبادلهٔ اطلاعات در این نوع شبکه بسیار بالا باشد.
- در این نوع شبکه با توجه به محدود بودن تعداد ایستگاهها ، مدیریت شبکه آسانتر از بقیهٔ شبکه‌ها است.
- هزینهٔ نصب و راه‌اندازی این نوع شبکه چندان بالا نیست.

شبکه‌های نوع LAN با ساختار مختلفی طراحی و پیاده‌سازی شده‌اند. یک شبکهٔ LAN واحد از یک نوع سخت‌افزار و یک نوع کانال فیزیکی انتقال استفاده می‌کند. چگونگی اتصال ایستگاهها از طریق کانال فیزیکی به یکدیگر ، "توپولوژی"[۲] آن شبکه را تعیین می‌نماید. چندین نوع توپولوژی برای شبکه‌های محلی وجود دارد:

**الف ) توپولوژی خطی -Bus-** : در این نوع توپولوژی تمام ایستگاهها از طریق یک کانال فیزیکی مشترک به همدیگر متصل شده‌اند و هرگونه تبادل اطلاعات از طریق این کانال انجام خواهد شد. در شکل (۳-۱) ساختار شبکهٔ باس به تصویر کشیده شده است. این توپولوژی بدلیل سادگی در نصب و راه‌اندازی و ارزان بودن یکی از شبکه‌های پررونق دنیا محسوب می‌شد ولی امروزه به تدریج جای خود را به انواع دیگر می‌دهد. بدلیل اهمیت ، در بخشهای آتی چگونگی طراحی و مدیریت کانال اشتراکی این شبکه را بررسی خواهیم کرد.

**ب) توپولوژی حلقه -Ring-** : در توپولوژی حلقه ، ایستگاهها در یک ساختار بستهٔ حلقوی به یکدیگر متصل می‌شوند. جهت جریان اطلاعات یکی از دو حالت ساعتگرد یا پادساعتگرد

---

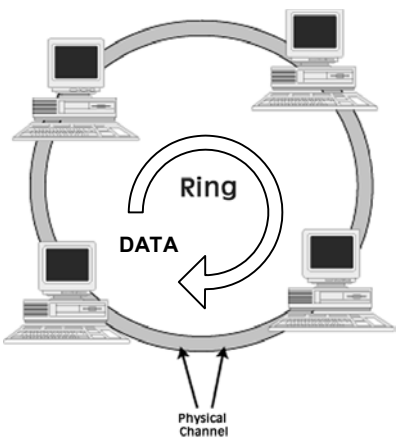
[۱] **تاخیر انتشار ( Propagation Delay ) مدت زمانی است که یک سیگنال حامل پیام (الکتریکی یا نوری) از ابتدای کانال به انتهای آن منتقل می‌شود. در کانالهای فیبر نوری تاخیر انتشار حدود ۳/۳ میکروثانیه و در کانالهای مسی یا رادیویی حدود ۵ میکروثانیه به ازای هر کیلومتر خواهد بود.**

[۲] LAN Topology

است و برای آنکه اطلاعات از یک ایستگاه به ایستگاه غیر مجاور آن در حلقه منتقل شود ، باید ماشینهایی که در مسیر هستند ، بسته‌های اطلاعاتی را دریافت و به کامپیوتر بعدی در حلقه بفرستند تا در نهایت اطلاعات به مقصد برسد. ارتباط هر ایستگاه با ایستگاه بعدی خود در حلقه یکطرفه است و اگر یک ایستگاه خواست به ماشین قبلی خود در حلقه بسته‌ای از داده‌ها را بفرستد آن بسته باید یک دور کامل در حلقه گردش کند تا به ایستگاه مورد نظر برسد. در شکل (٤-١) این توپولوژی نشان داده شده است.

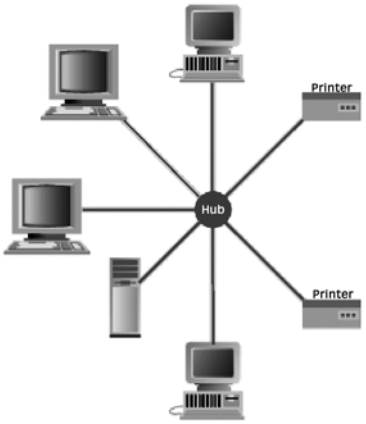
**ج) توپولوژی ستاره -Star-** : در توپولوژی ستاره ارتباط تمامی ماشینهای شبکه از طریق یک "گره" مرکزی برقرار می‌شود. این گره می‌تواند یک سوئیچ بسیار سریع و هوشمند یا یک "هاب"[1] معمولی یا حتی یک کامپیوتر باشد. در مورد انواع هاب و سوئیچ در بخشی مجزا توضیح خواهیم داد زیرا این نوع توپولوژی امروزه رونق گرفته و در حال جایگزین شدن بجای توپولوژی باس است. نمایش کلی این توپولوژی در شکل (٥-١) نشان داده شده است.

شکل (٣-١) ساختار شبکهٔ باس




شکل (٤-١) ساختار شبکهٔ حلقه

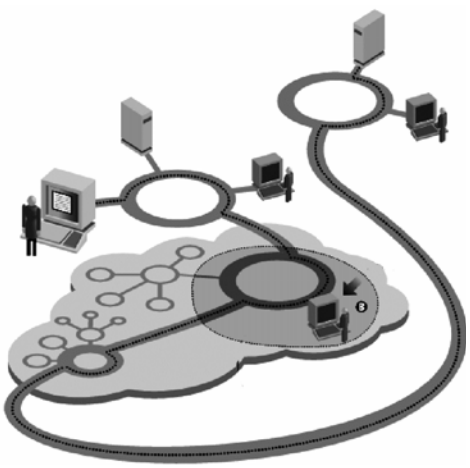
[1] Hub




شکل (۵-۱) ساختار شبکهٔ ستاره

**۲-۲-۳) شبکه‌های بین شهری -MAN-**

برای ایجاد شبکه در سطح یک منطقهٔ وسیع در حد یک شهر یا اتصال چندین شبکهٔ محلی ، از شبکهٔ MAN استفاده می‌شود. این شبکه ، تکنولوژی و توپولوژی مشابه با شبکه‌های محلی دارد. بدلیل طول زیاد کانال (حدود ۱۰۰ کیلومتر) معمولاً از فیبر نوری استفاده می‌شود. در بخشهای آتی دو نوع شبکهٔ MAN معرفی خواهد شد. در شکل (٦-۱) شمای فرضی از یک شبکهٔ MAN به تصویر کشیده شده است.



شکل (٦-۱) شمای فرضی از یک شبکهٔ MAN

**۳-۲-۳) شبکه‌های گسترده -WAN-**

شبکه‌های WAN ، در گستره جغرافیایی یک کشور یا جهان پیاده‌سازی می‌شود و شبکه‌های محلی و بین‌شهری را به هم مرتبط می‌نماید. آشکار است که چنین شبکه‌ای ، نمی‌تواند ساختار همگون و یکسان داشته باشد زیرا اولاً شبکه‌های محلی با توپولوژی متفاوت پیاده‌سازی شده‌اند؛ ثانیاً ماشینهای موجود در این شبکه‌ها ، از سخت‌افزار و نرم‌افزار متنوعی استفاده می‌کنند و بطور ذاتی با هم سازگار نیستند.

نیاز به دسترسی به منابع اطلاعات و ارتباط جهانی ، مجموعهٔ شبکه‌های کوچک و بزرگ را فارغ از ساختار سخت‌افزاری یا نرم‌افزاری آنها ، بهم پیوند زده و شبکهٔ WAN را پدید می‌آورد.

در ادبیات شبکه ، به "ماشینهای نهایی" که در اختیار کاربر قرار دارد و برنامه‌های کاربردی او را اجرا می‌کند ، **"ماشین میزبان"**[۱] می‌گویند.

به یک واحد اطلاعاتی که بصورت مستقل توسط یک ماشین تولید و روی شبکه ارسال می‌شود ، **"بسته"**[۲] گفته می‌شود. بسته‌ها معمولاً دارای اندازه‌ای متغیر بین چند بایت تا چند کیلوبایت هستند؛ ولی اگر بسته‌ها دارای اندازهٔ ثابت و کوچک (زیر صدبایت) باشند به آنها **سلول**[۳] گفته می‌شود.

ماشینهای میزبان توسط قسمت عظیمی از شبکه که از دید کاربر مخفی است بهم متصل شده‌اند. این قسمت از شبکهٔ WAN ، **"زیرساخت ارتباطی"** (یا به اختصار **"زیرشبکه"**) نامیده می‌شود و وظیفهٔ آن ، حمل و انتقال داده‌های یک ماشین میزبان به ماشین دیگر است.[٤]

زیرساخت ارتباطی در شبکهٔ WAN از دو بخش تشکیل شده است:

- **عناصر سوئیچ**[٥] : کامپیوترهای ویژه‌ای هستند که به چندین کانال ارتباطی ورودی/خروجی متصلند و وظیفه دارند پس از دریافت یک بسته ، با در نظر گرفتن مقصدِ آن ، یک کانال خروجی مناسب انتخاب کنند به نحوی که بسته را به مقصدش

---

[۱] **ماشین میزبان یا Host را در ادبیات شبکه End system هم گفته‌اند.**

[۲] Packet

[۳] Cell

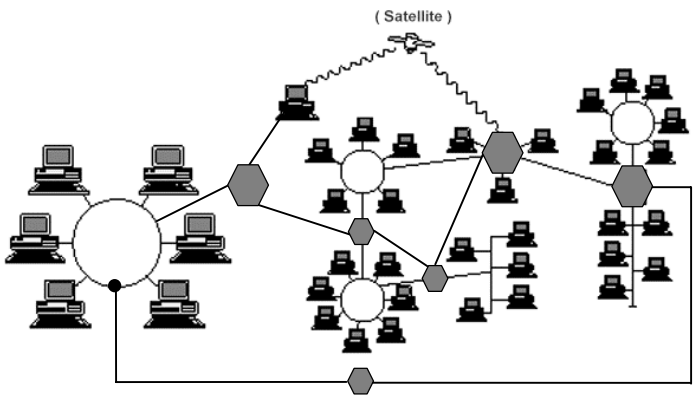
[۴] **اگر شبکهٔ WAN را با شبکهٔ تلفنی مقایسه کنید باید "ماشین میزبان" را در ذهن خودتان گوشی تلفن فرض کرده و تمامی آنچه که از سرپریز تلفنتان شروع می‌شود (شامل مرکز مخابرات محلی، مراکز سوئیچ مرکزی ، مراکز مایکروویو...) را "زیرشبکه" تجسم کنید.(البته مثال فوق را فقط برای روشن شدن قضیه در نظر بگیرید چرا که مرکز سوئیچ تلفن و مراکز سوئیچ داده تفاوت بنیادی دارند)**

[٥] Switching Elements

نزدیک کند؛ سپس بسته را روی آن کانال هدایت کنند.[۱] در سرتاسر کتاب به این عناصر **"مسیریاب"** اطلاق می‌کنیم چرا که توافق بیشتری روی آن وجود دارد.

- **خطوط ارتباطی یا کانالها**[۲] : خطوط انتقال با پهنای باند بالا هستند که ارتباط عناصر سوئیچ را برقرار می‌کنند.

به شکل (۷-۱) توجه کنید. در این شکل نماد ⬢ مسیریابها را مشخص می‌کند و شبکه‌های محلی ، محدوده جغرافیایی کاملاً متفاوتی دارند. همانگونه که از شکل مشخص است بین دو ماشین در شبکه‌های متفاوت ، چندین مسیر وجود دارد.




شکل (۷-۱) یک شبکهٔ فرضی WAN

[۱] عناصر سوئیچ نیز در برخی از کتابها با نامهای زیر نیز معرفی شده‌اند:
Packet Switching Nodes
Intermediate Systems
Data switching Exchange
Router

[۲] برای خطوط ارتباطی نیز اصطلاحات متفاوتی عنوان شده است ، مثل : Channels ، Circuits یا Trunks

برای ارائهٔ یک دیدگاه از شبکهٔ WAN ، فرض کنید در کشور ایران ، بین مراکز استانها طبق نقشهٔ شکل (۸-۱) ، کانال فیبر نوری و در هر مرکز استان یک عنصر سوئیچ نصب شود.( مثلاً سوئیچهای ATM یا MPLS ) با پیاده‌سازی این ساختار که نقش ستون فقرات زیرشبکه را بازی می‌کند ، در هر مرکز استان ، سازمانها و نهادهای مختلف می‌توانند با تخصیص یک کانال با پهنای باند متوسط ، شبکهٔ محلی خود را به این زیرشبکهٔ ارتباطی متصل سازند؛ بدینصورت هر ماشین در شبکه‌های محلی قادر خواهند بود با هر ماشین دیگر در سرتاسر کشور ارتباط داشته باشد. در این نقشه مسیرهای مختلفی بین مراکز سوئیچ استانها وجود دارد و به فرض اگر کانال تهران-شیراز بدلیلی از کار بیفتد ارتباط هیچ ماشینی در شبکه قطع نخواهد شد چون مسیرهای جایگزین دیگری در زیرساخت شبکه وجود دارند.

در این مثال زیرشبکهٔ نصب شده در کل کشور می‌تواند ارتباط اطلاعاتی شبکه‌های تمام مراکز استان را برقرار سازد و یک شبکهٔ WAN محسوب می‌شود ، بدون آنکه اتصال به شبکهٔ اینترنت لازم باشد. حال فرض یک مرکز سوئیچ در شمال کشور به خط فیبر نوری "جادهٔ ابریشم" وصل شود و در تهران و بندرعباس نیز دو کانال ماهواره‌ای با پهنای باند بالا ، ارتباط این زیرساخت را با ستون فقرات شبکهٔ اینترنت در اروپا ، برقرار کند. در این حالت کل شبکه‌های محلی عملاً به اینترنت متصلند و بسته‌های اطلاعاتی آنها از طریق این سه مرکز سوئیچ ، به ستون فقرات شبکهٔ اینترنت هدایت خواهد شد.

در شبکه WAN به همان اندازه که ماشینهای متصل به شبکه متفاوتند ، کانالهای ارتباطی در زیرشبکه و مسیریابها هم متنوعند. کانالهای ارتباطی در زیرشبکه می‌توانند خطوط اجاره‌ای[1] ، خطوط $T_1$ تا $T_4$ از استاندارد شبکهٔ تلفن Bell ، فیبرنوری ، کانالهای ماهواره‌ای یا کانالهای رادیویی باشند.

عناصر سوئیچ و مسیریابها در زیرشبکه از روش **"سوئیچ بسته"** که در بخشهای بعدی با آن آشنا خواهیم شد استفاده می‌کنند. این عناصر اصطلاحاً از قاعدهٔ **"دریافت ، ذخیره و هدایت به جلو"**[2] پیروی می‌کنند ، بدین معنا که وقتی یک بسته در ورودی یک مسیریاب دریافت می‌شود ، بطور کامل در حافظهٔ مرکز سوئیچ یا

---

[1] Leased Line
[2] Store & Forward

مسیریاب ذخیره می‌گردد تا یک خط مناسب و آزاد برای طی مسیر آن پیدا شود؛ سپس آن بسته روی آن خط ارسال می‌شود.

سوال مهم آنست که در یک شبکهٔ WAN که اجزاء آن خودشان یک شبکه محلی با دهها ماشین میزبان هستند و هر کدام از سخت افزار ، سیستم عامل و برنامه‌های کاربردی خاص خود استفاده می‌کنند ، ماشینها به چه صورت می‌توانند با یکدیگر مرتبط شوند در حالیکه از کلیهٔ جهات به شدت ناهمگن و ناسازگار هستند؟ این سوالی است که مطالب این کتاب باید به آن جواب بدهد.


شکل (۸-۱) یک زیرساخت فرضی برای پیاده‌سازی WAN در ایران

**۴-۲-۳) شبکه‌های بی‌سیم -Wireless-**

امروزه کامپیوترهای قابل‌حمل[۱] (مثل کامپیوترهای کیفی یا کامپیوترهای قابل نصب در خودرو) رشدی چشمگیر داشته و به وفور از آن استفاده می‌شود. برای ایجاد شبکه‌ای که در آن برخی از ایستگاهها متحرک هستند ، هیچ راه‌حلی برای انتقال داده‌ها مگر استفاده از کانالهای انتقال رادیویی باقی نمی‌ماند. انگیزه‌های دیگری نیز وجود دارند که حتی برای شبکه‌های غیرمتحرک از کانالهای رادیویی استفاده شود؛ مثلاً در محیطهایی همانند بیمارستان ، موزه و ساختمانهای قدیمی که کابل‌کشی در آنها مقرون به صرفه یا عقلائی نیست ، شبکه‌های بی‌سیم جایگاه ویژه‌ای خواهند داشت. در مجموع بدلیل عدم نیاز به کابل‌کشی ، نصب و راه‌اندازی این نوع شبکه ساده است ولی هزینهٔ بالاتری نسبت به شبکه‌های مبتنی بر کابلهای مسی خواهد داشت.

در شبکه‌های بی‌سیم بدلیل استفاده از کانالهای رادیویی اولاً نرخ ارسال و دریافت پایین است. ثانیاً نرخ خطا در این نوع شبکه‌ها نسبتاً بالاست. ثالثاً امنیت اطلاعات وجود ندارد چون براحتی می‌توان اطلاعات روی کانال را استراق سمع کرد.

در مورد این شبکه نیز به اختصار در فصل بعد توضیح خواهیم داد.

## ۴) روشهای برقراری ارتباط دو ماشین در شبکه

روش برقراری ارتباط بین گیرنده و فرستنده را سوئیچینگ ( راهگزینی ) گویند. در یک شبکه ( چه شبکهٔ انتقال داده و چه شبکهٔ انتقال صدا و تصویر ) معمولاً گیرنده و فرستنده مستقیماً به هم متصل نیستند بلکه بین آنها یک زیرشبکهٔ وسیع ارتباطی وجود دارد ، که نقش برقراری ارتباط بین دو ماشین را برعهده دارد. از این شبکه ایستگاههای مختلفی برای ارتباط با یکدیگر استفاده می‌کنند. برای ایجاد یک ارتباط بین دو ماشین در یک شبکه ، سه روش مختلف وجود دارد:

- سوئیچینگ مداری[۲]
- سوئیچینگ پیام[۳]
- سوئیچینگ بسته[۴] و سلول

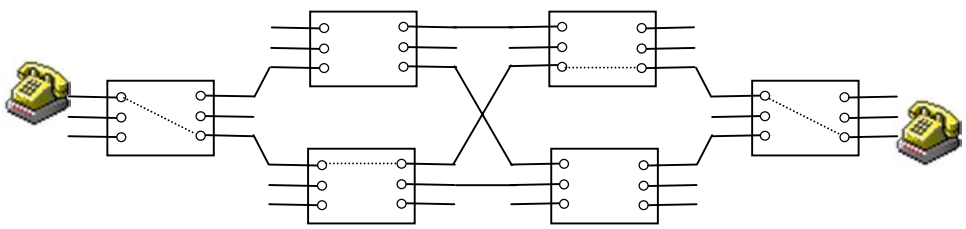
---

۱ Portable
۲ Circuit Switching
۳ Message Switching
۴ Packet Switching / Cell Switching

### ۱-۴) سوئیچینگ مداری

در این روش برای انتقال اطلاعات بین دو ماشین ابتدا یک اتصال فیزیکی بین مبدأ و مقصد برقرار می شود. ( مهمترین مثال آن شبکه‌های تلفن است ) در این روش خطوطِ ارتباطی گیرنده و فرستنده از نظر الکتریکی به یکدیگر متصل می‌شوند ، به همین دلیل به این روش سوئیچینگ مداری گفته می‌شود. در زمانهایی که گیرنده یا فرستنده غیر فعالند ( یعنی در زمان قطع ارتباط ) هیچ گونه ارتباط فیزیکی بین آنها برقرار نیست و مدار اتصال-باز[۱] است. برای برقرار شدن یک ارتباط فیزیکی ، مدت زمان قابل توجهی به عنوان زمان "برقراری و تنظیم ارتباط"[۲] صرف می‌شود تا ارتباطِ مداری برقرار شود. گاهی زمان برقراری ارتباط از چند ثانیه تا چند ده ثانیه طول خواهد کشید ، که این زمان از دیدگاه شبکهٔ کامپیوتری قابل قبول نیست و یک کامپیوتر می تواند در این زمان تلف شده چندین هزار بیت را منتقل کند ، به همین دلیل حتی الامکان سعی می شود در شبکه های کامپیوتری از این روش استفاده نشود. در روش سوئیچ مداری اگر کانالی توسط گیرنده و فرستنده اشغال شود ، برای هیچ ماشین دیگری مقدور نخواهد بود که با این دو ماشین ارتباط برقرار کند و باید تا آزاد شدن کانال ، منتظر بمانند. منظور از برقراری یک ارتباط فیزیکی بین دو ماشین آنست که پس از برقراری ارتباط بین آنها ، عملاً مسیری بین آنها ایجاد می‌شود که همانند یک کانال مستقیم ، سیگنال منتشر شده از یک مبدأ ، در مقصد دریافت می‌شود. ( البته با تاخیر انتشار و مقداری نویز )

در شکل (۹-۱) شمای کلی سوئیچینگ مداری در سیستم تلفن به تصویر کشیده شده است. برای برقراری یک ارتباط ، ابتدا مبدأ باید شماره‌گیری نماید. شماره‌گیری عملی است برای آنکه مراکز سوئیچ میانی ، با وصل کردن سوئیچها ، ارتباط بین مبدأ و مقصد را برقرار نمایند. (فرآیند شماره‌گیری چندین ثانیه طول می‌کشد. ) در این شکل مسیر نقطه‌چین سوئیچ‌هایی است که وصل شده و ارتباط مبدأ و مقصد را ایجاد کرده است.



شکل (۹-۱) شمای کلی سوئیچینگ مدار در سیستم تلفن

---

[۱] Open Circuit
[۲] Setup

### ۲-۴) سوئیچینگ پیام

در روش سوئیچ پیام که صرفاً مختص انتقال داده‌های دیجیتال است ، هر ایستگاه یک اتصال دائمی و "همیشه وصل" با مرکز سوئیچ خود دارد. مرکز سوئیچ یک کامپیوتر با تعداد زیادی پورت دیجیتال ورودی/خروجی است که دارای حافظهٔ اصلی و حافظهٔ جانبی می‌باشد. هر ایستگاه به محض آنکه تمایل به ارسال داشته باشد با اضافه کردن اطلاعات لازم به ابتدای داده‌ها ، آنرا در قالب یک "پیام" تحویل مرکز سوئیچی که به آن متصل است می‌دهد؛ (خواه گیرندهٔ پیام آماده و آزاد باشد و خواه مشغول). اطلاعاتی که به ابتدای پیام اضافه می‌شود شامل آدرس گیرنده و آدرس فرستندهٔ پیام است و مرکز سوئیچ موظف است ، پیام را کاملاً دریافت کرده و آنرا در حافظه خود ذخیره کند. سپس بر اساس آدرس گیرنده ، کانال مناسب خروجی را برای آن انتخاب کرده و آنرا به سمت مرکز سوئیچ بعدی هدایت می‌کند تا نهایتاً با تکرار این روند ، پیام به ایستگاه مقصد برسد. مرکز سوئیچ نهایی که به ایستگاه گیرنده متصل است پیامهای رسیده برای یک ایستگاه را بافر کرده و به ترتیب و بر حسب اولویت برای آن ایستگاه ارسال می‌نماید.

بر خلاف روش سوئیچ مداری ، در این روش هیچ ایستگاهی مجبور نیست قبل از ارسال پیام ، اقدام به برقراری یک مسیر فیزیکی نماید و به محض آماده شدن داده‌ها برای ارسال ، آنها را به سوی مرکز سوئیچ متصل به آن ، ارسال می‌کند و بنابراین زمان "تنظیم مسیر فیزیکی" که بسیار وقتگیر است ، حذف خواهد شد. در ضمن کانال بین دو ایستگاه اشغال نخواهد شد و تمام ایستگاهها می‌توانند برای یکدیگر پیام بفرستند. اگر دو پیام از دو ایستگاه متفاوت برای یک ایستگاه واحد ارسال شود ، پس از دریافت و نگهداری در حافظهٔ مرکز سوئیچ ، به ترتیب برای آن ایستگاه ارسال خواهد شد.

با توجه به نکات عنوان شده ، روش سوئیچ پیام بسیار سریع و کارآمد است و اشغال کانال وجود نخواهد داشت ، ولی این روش یک عیب اساسی دارد : "عدم محدودیت طول پیام"
اگر هر مرکز سوئیچ موظف باشد پیامهای مربوط به ایستگاهها را کاملاً دریافت و سپس آنرا به سمت مسیری مناسب هدایت کند ، بدون آنکه هیچ محدودیتی بر روی طول پیام وجود داشته باشد ، اشکالات عمدهٔ زیر پدید می‌آید:

- هر مرکز سوئیچ باید فضای حافظه بسیار زیادی داشته باشد تا وقتی با حجم انبوهی از پیامهای ایستگاهها مواجه می‌شود بتواند آنها را ذخیره کند و پیامها از دست نروند. حتی ممکن است بدلیل عدم محدودیت روی طول پیام ، مرکز سوئیچ در لحظاتی با کمبود حافظه مواجه شده و مجبور شود از فضای حافظهٔ جانبی (مثل دیسک سخت) استفاده کند که سرعت انتقال پیام را از مبدأ به مقصد ، بشدت کاهش خواهد داد.

- در صورت بروز حتی یک بیت خرابی در پیام (ناشی از خطای کانال) ، حجم بسیار زیادی از داده‌ها باید مجدداً ارسال شوند.
- چون هر مرکز سوئیچ موظف است کل پیام را دریافت کرده و سپس آنرا به کانال مناسب هدایت نماید ، لذا تاخیر رسیدن پیام زیاد خواهد شد ، چراکه اگر زمان دریافت یک پیام بزرگ $t$ ثانیه باشد و در مسیر بین مبدأ و مقصد n مرکز سوئیچ واقع شده باشد ، کل تاخیر معادل $n.t$ ثانیه خواهد بود. برای پیامهای بزرگ این زمان بسیار زیاد خواهد شد ، مثلاً اگر طول پیام ، یک مگابایت باشد و زمان دریافت این پیام در هر مرکز سوئیچ جمعاً ۵ ثانیه باشد ، برای گذر از ۱۰ مرکز سوئیچ در طول مسیر ، ۵۰ ثانیه تاخیر ایجاد می‌شود که بسیار زیاد است و می‌توان آنرا کاهش داد. برای درک این مسئله به شکل (الف-۱۰-۱) دقت کنید. در این شکل فرض شده است که بین دو ایستگاه مراکز سوئیچ A ، B ، C و D وجود داشته باشد. فرستندهٔ پیام آنرا تحویل A داده است و باید به نحو مناسبی به مرکز سوئیچ D تحویل داده شود. محور عمودی نشان دهندهٔ زمان است.

### ۳-۴) سوئیچینگ بسته و سلول

مشکلات ناشی از عدم محدودیت طول پیام ، باعث شد که در روشهای جدید بر روی اندازهٔ پیام ارسالی محدودیت گذاشته شود و ایستگاه اجازه نداشته باشد در هر بار ارسال ، کل یک پیام بزرگ را یکجا بفرستد ، بلکه موظف است آنرا به قطعات کوچکتری به نام "بسته" تقسیم کرده و ضمن اضافه کردن اطلاعات لازم برای بازسازی اصل پیام به هر بسته ، آنها را بطور جداگانه به مرکز سوئیچ ارسال کند. مثلاً ایستگاهی که تمایل دارد پیامی شامل یک مگابایت اطلاعات را برای یک ایستگاه دیگر بفرستد می‌تواند آنرا به هزار بسته تقسیم کرده و آنها را به صورت مستقل ارسال نماید. تمام ایستگاهها موظفند از مکانیزم شماره‌گذاری بسته‌ها تبعیت کنند تا امکان بازسازی اصل پیام وجود داشته باشد.

مجموع تاخیر در روش سوئیچ بسته کمتر از روش سوئیچ پیام خواهد بود؛ برای مقایسهٔ تاخیر ، به شکل (ب-۱۰-۱) دقت کنید. در این شکل فرض شده است پیامی که در شکل (الف-۱۰-۱) به روش سوئیچ پیام ، بصورت یکجا ارسال شده در قالب سه بستهٔ مجزا ارسال شود. چون مراکز سوئیچ پس از دریافت کامل یک بسته قادرند بطور همزمان ضمن ادامهٔ دریافت بسته‌های بعدی ، بستهٔ فعلی را روی کانال مناسب هدایت کنند ، لذا بگونه‌ای که در

شکل نشان داده شده ، به دلیل "روی هم افتادگی"[۱] زمانهائی که مرکز سوئیچ باید معطل بماند تا بستهٔ بعدی دریافت شود ، تاخیر کل کاهش چشمگیری داشته است.

در این روش با توجه به محدود بودن طول هر بسته ، اولاً فضای حافظهٔ مورد نیاز برای هر مرکز سوئیچ قابل تخمین و قابل تأمین خواهد بود. ثانیاً در صورت بروز خرابی در یک بسته فقط بخش ناچیزی از کل پیام خراب شده و ارسال مجدّد خواهد شد.

در مراکز سوئیچ مدرن که با سرعت بسیار بالا عمل می‌کنند ، طول بسته‌ها ثابت و بسیار کوچک است؛ در سوئیچ‌های ATM هر بسته دقیقاً ۵۳ بایت است که از این ۵۳ بایت ، ٤٨ بایت داده و ۵ بایت سرآیند[۲] محسوب می‌شود. به دلیل اندازهٔ کوچک و ثابت این بسته‌ها به آنها **سلول** گفته می‌شود. کوچک بودن اندازه سلول ، مرکز سوئیچ را قادر می‌سازد تا بتواند به سرعت عمل هدایت بسته از یک ورودی به یک خروجی مناسب را انجام بدهد. دقت کنید که در این روش نرخ ارسال و دریافت بسیار بالا و نرخ خرابی سلولها بسیار پایین است و معمولاً از فیبرهای نوری استفاده می‌شود. (مثلاً سوئیچ‌های ATM با نرخ ارسال 155.52 Mbps و 622.08 Mbps در دسترس است.نرخ خرابی سلول در این سوئیچها حدود $10^{-12}$ است).

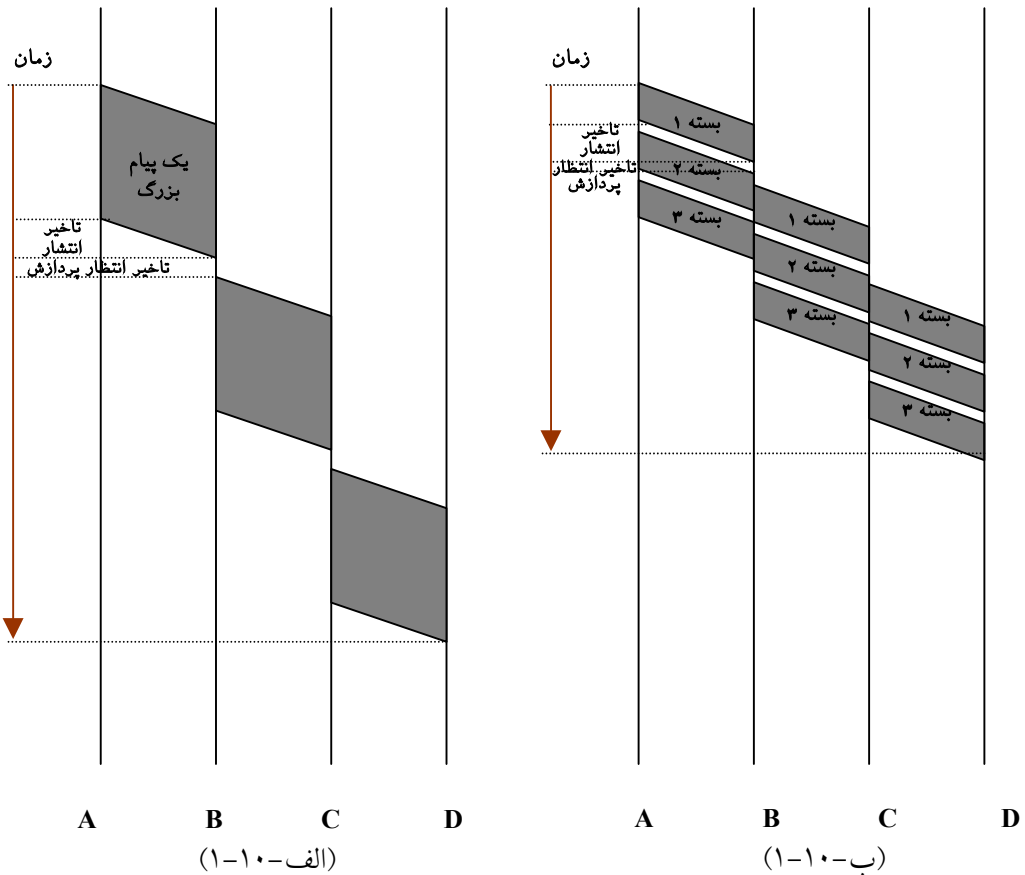
به گونه‌ای که در بخشهای قبلی اشاره شد مراکز سوئیچ بسته یا سلول ، اصطلاحاً از قاعدهٔ **"دریافت ، ذخیره و هدایت به جلو"**[۳] پیروی می‌کنند ، بدین معنا که وقتی یک بسته در ورودی یک مسیریاب دریافت می‌شود ، بطور کامل در حافظهٔ مرکز سوئیچ یا مسیریاب ذخیره می‌گردد تا یک خط مناسب و آزاد برای طی مسیر آن پیدا شود؛ سپس آن بسته روی آن خط ارسال می‌شود. به مراکز سوئیچ بسته معمولاً **"مسیریاب"** گفته می‌شود در حالی که به مراکز سوئیچ سلول به اختصار **"سوئیچ"** می‌گویند. ( مثل سوئیچهای ATM یا MPLS که در ستون‌فقرات شبکه‌ها مورد استفاده قرار می‌گیرد)

هرچند ممکن است بسیاری از کاربران شبکهٔ WAN برای اتصال به مراکز سرویس‌دهندهٔ شبکه[٤] ، از خطوط تلفن که به روش سوئیچ مداری عمل می‌کنند و در عین حال کم ظرفیت هستند ، استفاده نمایند ولیکن برای اتصال شبکه‌ها به یکدیگر ، این روش بهیچوجه قابل قبول نبوده و باید از روشهای سوئیچ بسته یا سلول استفاده

---

۱ Time Overlap
۲ Header
۳ Store & Forward
٤ Network Service Provider

شـود. از همـان روزهـای ابتدایـی ظـهور شـبکه‌های WAN (مثـل شـبکهٔ ARPA در ابتـــدای دههٔ هفتاد میلادی) از روشـــهای سـوئیچ بسـته اسـتفاده می‌شـد.




شکل (۱۰-۱) زمانبندی تاخیر در روشهای سوئیچینگ پیام و بسته

## ۵) طراحی شبکه‌ها و اصول لایه‌بندی

برای طراحی یک شبکهٔ کامپیوتری ، مسائل و مشکلات بسیار گسترده و متنوعی وجود دارد که باید به نحوی حل شود تا بتوان یک ارتباط مطمئن و قابل اعتماد بین دو ماشین در شبکه برقرار کرد. این مسائل و مشکلات همگی از یک سنخ نیستند و منشأ و راه‌حل مشابه نیز ندارند و بخشی از آنها توسط سخت‌افزار و بخش دیگر با تکنیکهای نرم‌افزاری قابل حل هستند. به عنوان مثال نیاز برای ارتباط بی‌سیم بین چند ایستگاه در شبکه ، طراح شبکه را مجبور به استفاده از مدولاسیون آنالوگ در سخت‌افزار مخابراتی خواهد کرد ولی مسئلهٔ هماهنگی در ارسال بسته‌ها از مبدأ به مقصد یا شماره‌گذاری بسته‌ها برای بازسازی پیام و اطمینان از رسیدن یک بسته ، با استفاده از تکنیکهای نرم‌افزاری قابل حل است. بهمین دلیل برای طراحی شبکه‌های کامپیوتری ، باید مسائل و مشکلاتی که برای برقراری یک ارتباط مطمئن ، ساده و شفاف بین دو ماشین در شبکه وجود دارد ، دسته‌بندی شده و راه‌حلهای استاندارد برای آنها ارائه می‌شود. در زیر بخشی از مسائل طراحی شبکه‌ها عنوان شده است :

- اولین موضوع چگونگی ارسال و دریافت بیتهای اطلاعات بصورت یک سیگنال الکتریکی ، الکترومغناطیسی یا نوری است، بسته به اینکه آیا کانال انتقال سیم مسی ، فیبر نوری ، کانال ماهواره‌ای یا خطوط مایکروویو است. بنابراین تبدیل بیتها به یک سیگنال متناسب با کانال انتقال یکی از مسائل اولیهٔ شبکه به شمار می‌رود.
- مساله دوم ماهیت انتقال است که می تواند به یکی از سه صورت زیر باشد:
  - **Simplex** : ارتباط یکطرفه ( یک طرف همیشه گیرنده و طرف دیگر همیشه فرستنده).
  - **Half Duplex** : ارتباط دوطرفهٔ غیرهمزمان ( هر دو ماشین هم می‌توانند فرستنده یا گیرنده باشند ولی نه بصورت همزمان ، بلکه یکی از طرفین ابتدا ارسال می‌کند ، سپس ساکت می‌شود تا طرف مقابل ارسال داشته باشد)
  - **Full Duplex** : ارتباط دو طرفه همزمان ( مانند خطوط مایکروویو )

- مساله سوم مسئلهٔ خطا و وجود نویز روی کانالهای ارتباطی است بدین معنا که ممکن است در حین ارسال داده‌ها بر روی کانال فیزیکی تعدادی از بیتها دچار خرابی شود؛ چنین وضعیتی که قابل اجتناب نیست باید تشخیص داده شده و داده‌های فاقد اعتبار دور ریخته شود مبدأ آنها را از نو ارسال کند.
- با توجه به اینکه در شبکه‌ها ممکن است مسیرهای گوناگونی بین مبدأ و مقصد وجود داشته باشد؛ بنابراین پیدا کردن بهترین مسیر و هدایت بسته‌ها ، از مسائل طراحی شبکه

محسوب می‌شود. در ضمن ممکن است یک پیام بزرگ به واحدهای کوچکتری تقسیم شده و از مسیرهای مختلفی به مقصد برسد بنابراین بازسازی پیام از دیگر مسائل شبکه به شمار می‌آید.

- ممکن است گیرنده به دلایلی نتواند با سرعتی که فرستنده بسته‌های یک پیام را ارسال می‌کند آنها را دریافت کند ، بنابراین طراحی مکانیزمهای حفظ هماهنگی بین مبدأ و مقصد از دیگر مسائل شبکه است.
- چون ماشینهای فرستنده و گیرندهٔ متعددی در یک شبکه وجود دارد مسائلی مثل ازدحام ، تداخل و تصادم در شبکه‌ها بوجود می‌آید که این مشکلات بهمراه مسائل دیگر باید در سخت‌افزار و نرم‌افزار شبکه حل شود.

طراح یک شبکه باید تمام مسائل شبکه را تجزیه و تحلیل کرده و برای آنها راه حل ارائه کند ولی چون این مسائل دارای ماهیتی متفاوت از یکدیگر هستند ، بنابراین طراحی یک شبکه باید بصورت **"لایه به لایه"** انجام شود. به عنوان مثال وقتی قرار است یک شبکه به گونه‌ای طراحی شود که ایستگاهها بتوانند انتقال فایل داشته باشند ، اولین مسئله‌ای که طراح باید به آن بیندیشد طراحی یک سخت‌افزار مخابراتی برای ارسال و دریافت بیتها روی کانال فیزیکی است. اگر چنین سخت‌افزاری طراحی شود ، می‌تواند بر اساس آن اقدام به حل مسئلهٔ خطاهای احتمالی در داده‌ها نماید؛ یعنی زمانی مکانیزمهای کنترل و کشف خطا مطرح می‌شود که قبل از آن سخت‌افزار مخابرهٔ داده‌ها طراحی شده باشد. بعد از این دو مرحلهٔ طراحی ، باید مکانیزمهای بسته‌بندی اطلاعات ، آدرس‌دهی ماشینها و مسیریابی بسته‌ها طراحی شود. سپس برای بقیه مسائل نظیر آدرس‌دهی پروسه‌ها و چگونگی انتقال فایل راه حل ارائه شود.

طراحی لایه‌ای شبکه به منظور تفکیک مسائلی است که باید توسط طراح حل شود و مبتنی بر اصول زیر است[۱]:

- هر لایه وظیفه مشخصی دارد و طراح شبکه باید آنها را به دقت تشریح کند.
- هرگاه سرویسهایی که باید ارائه شود از نظر ماهیتی متفاوت باشد ، باید لایه به لایه و جداگانه طراحی شود.
- وظیفه هر لایه باید با توجه به قراردادها و استانداردهای جهانی مشخص شود.

---

[۱] **طراحی لایه‌ای شبکه را می‌توان با برنامه‌نویسی ماجولار مقایسه کرد ، بدین نحو که روالهای حل یک مسئله به اجزای کوچکتری شکسته می‌شود و برای آن زیربرنامه نوشته می‌شود. در توابع صدازنندهٔ این زیربرنامه‌ها ، جزئیات درونی آنها اهمیت ندارد بلکه فقط نحوهٔ صدازدن آنها و پارامترهای مورد نیاز ورودی به زیربرنامه و چگونگی برگشت نتیجه به صدازننده ، مهم است.**

- تعداد لایه ها نباید آنقدر زیاد باشد که تمایز لایه ها از دیدگاه سرویسهای ارائه شده نامشخص باشد و نه آنقدر کم باشد ، که وظیفه و خدماتِ یک لایه ، پیچیده و نامشخص شود.
- در هر لایه جزئیات لایه‌های زیرین نادیده گرفته می‌شود و لایه‌های بالایی باید در یک روال ساده و ماجولار از خدمات لایهٔ زیرین خود استفاده کنند.
- باید مرزهای هر لایه به گونه‌ای انتخاب شود که جریان اطلاعات بین لایه‌ها ، حداقل باشد.

برای آنکه طراحی شبکه‌ها سلیقه‌ای و پیچیده نشود سازمان جهانی استاندارد[۱] –ISO– ، مدلی هفت لایه‌ای برای شبکه ارائه کرد ، به گونه‌ای که وظائف و خدمات شبکه در هفت لایهٔ مجزا تعریف و ارائه می‌شود. این مدل هفت لایه‌ای ، **OSI**[۲] نام گرفت. هر چند در شبکهٔ اینترنت از این مدل استفاده نمی‌شود و بجای آن یک مدل چهار لایه‌ای به نام **TCP/IP** تعریف شده است ، ولیکن بررسی مدل هفت لایه‌ای OSI ، بدلیل دقّتی که در تفکیک و تبیین مسائل شبکه در آن وجود دارد ، با ارزش خواهد بود. پس از بررسی مدل OSI ، به تشریح مدل TCP/IP خواهیم پرداخت.

## ۶) مدل هفت لایه‌ای OSI از سازمان استاندارد جهانی ISO

در این استاندارد کل وظائف و خدمات یک شبکه در هفت لایه تعریف شده است :

| | |
|---|---|
| **لایه ۱ – لایه فیزیکی** | **Physical Layer** |
| **لایه ۲ – لایه پیوند داده‌ها** | **Data link Layer** |
| **لایه ۳ – لایه شبکه** | **Network Layer** |
| **لایه ٤ – لایه انتقال** | **Transport Layer** |
| **لایه ۵ – لایه جلسه** | **Session Layer** |
| **لایه ٦ – لایه ارائه (نمایش)** | **Presentation Layer** |
| **لایه ۷ – لایه کاربرد** | **Application Layer** |

[۱] International Standard Organization
[۲] Open System Interconnection

از لایه‌های پایین به بالا ، سرویسهای ارائه شده (با تکیه بر سرویسی که لایه‌های زیرین ارائه می‌کنند) پیشرفته‌تر می‌شود.

این مدل به منظور تعریف یک استاندارد جهانی و فراگیر ارائه شد و گمان می‌رفت که تمام شبکه‌ها بر اساس این مدل در هفت لایه طراحی شوند ، به گونه‌ای که در دهۀ هشتاد سازمان ملی علوم در آمریکا عنوان کرد که در آینده فقط از این استاندارد حمایت خواهد کرد ، ولی در عمل ، طراحان شبکه به این مدل وفادار نماندند. مثلاً شرکت ناول مدل پنج لایه‌ای خودش را بکار گرفت و در اینترنت مدل TCP/IP فراگیر شد. در اینجا به دلایل شکست مدل OSI نخواهیم پرداخت زیرا پاره‌ای از این عوامل از مسائل غیرعلمی (همانند انتشار رایگان اصل برنامه‌های TCP/IP توسط دانشگاه برکلی) نشأت می‌گیرد و محل بحث و مناقشه است. در مجموع این مدل ، مرجع بسیار کامل و مناسبی برای بحث در دروس دانشگاهی است.

در ادامه به اختصار وظائف هر لایه در مدل OSI را تعریف خواهیم کرد.

### ۱-۶) لایۀ فیزیکی

وظیفه اصلی در لایۀ فیزیکی ، انتقال بیتها بصورت سیگنال الکتریکی و ارسال آن بر روی کانال می باشد. واحد اطلاعات در این لایه بیت است و بنابراین این لایه هیچ اطلاعی از محتوای پیام ندارد و تنها بیتهای ۰ و ۱ را ارسال یا دریافت می‌کند. پارامترهایی که باید در این لایه مورد نظر باشند عبارتند از :

- ظرفیت کانال فیزیکی و نرخ ارسال[۱]
- نوع مدولاسیون
- چگونگی کوپلاژ با خط انتقال
- مسائل مکانیکی و الکتریکی مانند نوع کابل ، باند فرکانسی و نوع رابط (کانکتور) کابل.

در این لایه که تماماً سخت‌افزاری است ، مسائل مخابراتی در مبادلۀ بیتها ، تجزیه و تحلیل شده و طراحی‌های لازم انجام می‌شود. طراح شبکه می‌تواند برای طراحی این لایه ، از استانداردهای شناخته شدۀ انتقال همانند RS-232 و RS- 422 و RS- 423 و ... که سخت‌افزار آنها موجود است ، استفاده کند.

این لایه هیچ وظیفه‌ای در مورد تشخیص و ترمیم خطا ندارد.

---

[۱] Channel Capacity and Bit Rate

**۲-۶) لایهٔ پیوند داده‌ها**

وظیفهٔ این لایه آن است که با استفاده از مکانیزمهای کشف و کنترل خطا ، داده‌ها را روی یک کانال انتقال که ذاتاً دارای خطا است ، بدون خطا و مطمئن به مقصد برساند. در حقیقت می‌توان وظیفهٔ این لایه را بیمهٔ اطلاعات در مقابل خطاهای احتمالی دانست؛ زیرا ماهیت خطا به‌گونه‌ای است که قابل رفع نیست ولی می‌توان تدابیری اتخاذ کرد که فرستنده از رسیدن یا نرسیدن صحیح اطلاعات به مقصد مطلع شده و در صورت بروز خطا مجدداً اقدام به ارسال اطلاعات کند؛ با چنین مکانیزمی یک کانال دارای خطا به یک خط مطمئن و بدون خطا تبدیل خواهد شد.

یکی دیگر از وظائف لایهٔ پیوند داده ها آن است که اطلاعات ارسالی از لایهٔ بالاتر را به واحدهای استاندارد و کوچکتری شکسته و ابتدا و انتهای آن را از طریق نشانه‌های خاصی که Delimiter نامیده می‌شود ، مشخص نماید. این قالب استاندارد که ابتدا و انتهای آن دقیقاً مشخص شده ، **فریم** نامیده می‌شود؛ یعنی واحد اطلاعات در لایهٔ دوم فریم است.

کشف خطا که از وظایف این لایه می باشد می تواند از طریق اضافه کردن بیتهای کنترل خطا مثل بیتهای Parity Check و Checksum و CRC انجام شود.

یکی دیگر از وظایف لایهٔ دوم کنترل جریان یا به عبارت دیگر تنظیم جریان ارسال فریم‌ها به گونه‌ای است که یک دستگاه کند هیچ گونه فریمی را به خاطر آهسته بودن از دست ندهد.

یکی دیگر از وظایف این لایه آن است که وصول داده ها یا عدم رسید داده‌ها را به فرستنده اعلام کند.

در بخشهای قبل اشاره کردیم که بسیاری از شبکه‌ها از کانال اشتراکی استفاده می‌کنند و ارسال همزمان دو ایستگاه منجر به تصادم (اختلاط سیگنال انتقال) و خرابی داده‌ها خواهد شد. یکی دیگر از وظایف این لایه آن است که قراردادهایی را برای جلوگیری از تصادم سیگنال ایستگاههایی که از کانال اشتراکی استفاده می‌کنند ، وضع کند چراکه فرمان ارسال داده بر روی کانال مشترک از لایهٔ دوم صادر می‌شود. این قراردادها در زیرلایه‌ای به نام MAS[1] تعریف شده است.

وقتی یک واحد اطلاعاتی تحویل یک ماشین متصل به کانال فیزیکی در شبکه شد ، وظیفهٔ این لایه پایان می‌یابد. از دیدگاه این لایه ، ماشینهائی که به کانال فیزیکی متصل نمی‌باشند ، در دسترس نیستند. کنترل سخت‌افزار لایهٔ فیزیکی به عهدهٔ این لایه است.

فراموش نکنید که وظایف این لایه نیز با استفاده از سخت‌افزارهای دیجیتال انجام می‌شود.

---

[1] Medium Access Sublayer

### ۳-۶) لایهٔ شبکه

در این لایه اطلاعات به صورت بسته‌هایی سازماندهی می‌شود و برای انتقال مطمئن تحویل لایهٔ دوم می‌شود. با توجه به آنکه ممکن است بین دو ماشین در شبکه مسیرهای گوناگونی وجود داشته باشد ، لذا این لایه وظیفه دارد هر بستهٔ اطلاعاتی را پس از دریافت به مسیری هدایت کند تا آن بسته بتواند به مقصد برسد. در این لایه باید تدابیری اندیشیده شود تا از ازدحام (یعنی ترافیک بیش از اندازهٔ بسته‌ها در یک مسیریاب یا مرکز سوئیچ) جلوگیری شده و از ایجاد بن‌بست ممانعت بعمل بیاورد.[۱]

هر مسیریاب می‌تواند به صورت ایستا و غیرهوشمند بسته‌ها را مسیریابی کند. همچنین می‌تواند به صورت پویا و هوشمند برای بسته‌ها مسیر انتخاب نماید. در این لایه تمام ماشینهای شبکه دارای یک آدرس جهانی و منحصر به فرد خواهند بود که هر ماشین بر اساس این آدرسها اقدام به هدایت بسته‌ها به سمت مقصد خواهد کرد.

این لایه ذاتاً "بدون اتصال"[۲] است یعنی پس از تولید یک بستهٔ اطلاعاتی در مبدأ ، بدون هیچ تضمینی در رسیدن آن بسته به مقصد ، بسته شروع به طی مسیر در شبکه می‌کند. وظائف این لایه به سیستم نامه‌رسانی تشبیه شده است؛ یک پاکت محتوی نامه پس از آنکه مشخصات لازم بر روی آن درج شد ، به صندوق پست انداخته می‌شود ، بدون آنکه بتوان زمان دقیق رسیدن نامه و وجود گیرندهٔ نامه را در مقصد ، از قبل حدس زد. در ضمن ممکن است نامه به هر دلیلی گم شود یا به اشتباه در راهی بیفتد که مدتها در مسیر بماند و زمانی به گیرندهٔ آن برسد که هیچ ارزشی نداشته باشد.

در این لایه تضمینی وجود ندارد وقتی بسته ای برای یک ماشین مقصد ارسال می‌شود آن ماشین آمادهٔ دریافت آن بسته باشد و بتواند آنرا دریافت کند. در ضمن هیچ تضمینی وجود ندارد وقتی چند بستهٔ متوالی برای یک ماشین ارسال می‌شود به همان ترتیبی که بر روی شبکه ارسال شده ، در مقصد دریافت شوند. همچنین ممکن است که وقتی بسته‌ای برای یک مقصد ارسال می‌گردد ، به دلیل دیر رسیدن از اعتبار ساقط شده و مجدداً ارسال شود و هر دو بسته ( جدید و قدیم ) با هم برسند. این مسائل در لایهٔ بالاتر قابل حل خواهد بود.

هر چند وظائف این لایه می‌تواند بصورت نرم‌افزاری پیاده شود ولی برای بالاتر رفتن سرعت عمل شبکه ، می‌توان برای این لایه یک کامپیوتر خاص طراحی نمود تا در کنار سخت‌افزارِ لایه‌های زیرین ، بسته‌ها را روی شبکه رد و بدل کند.

---

[۱] **این اصطلاحات در فصول آتی تشریح خواهد شد.**

[۲] Connectionless

### ۶-۴) لایهٔ انتقال

در این لایه بر اساس خدمات لایهٔ زیرین ، یک سرویس انتقال بسیار مطمئن و "اتصال‌گرا"[۱] ارائه می‌شود. تمام مشکلاتی که در لایهٔ شبکه عنوان شد در این لایه حل و فصل می‌شود:

- قبل از ارسال بسته‌ها ، نرم‌افزار این لایه اقدام به ارسال یک بستهٔ ویژه می‌نماید تا مطمئن شود که ماشین گیرنده آمادهٔ دریافت اطلاعات است.
- جریان ارسال اطلاعات شماره‌گذاری شده تا هیچ بسته‌ای گم نشود یا دو بار دریافت نشود.
- ترتیب جریان بسته‌ها حفظ می‌شود.
- در این لایه پروسه‌های مختلفی که بر روی یک ماشین واحد اجرا شده‌اند ، آدرس‌دهی می‌شوند به نحوی که هر پروسه بر روی یک ماشین واحد ، به عنوان یک هویت مستقل داده‌های خود را ارسال یا دریافت نماید.

واحد اطلاعات در این لایه قطعه[۲] است. از وظائف دیگر این لایه می‌توان به موارد زیر اشاره کرد:

- تقسیم پیامهای بزرگ به بسته‌های اطلاعاتی کوچکتر
- بازسازی بسته‌های اطلاعاتی و تشکیل یک پیام کامل
- شماره گذاری بسته‌های کوچکتر جهت بازسازی
- تعیین و تبیین مکانیزم نامگذاری ایستگاه‌هایی که در شبکه‌اند.

وظائف این لایه (و لایه های بعدی) با استفاده از نرم‌افزار پیاده‌سازی می‌شود و فقط بر روی ماشینهای نهایی (ماشینهای کاربران) وجود دارد و مراکز سوئیچ به وظائف این لایه احتیاجی ندارند (مگر در موارد خاص).

این لایه در فصلی مجزا بررسی خواهد شد.

### ۶-۵) لایهٔ جلسه

وظیفهٔ این لایه فراهم آوردن شرایط یک جلسه (نشست) همانند ورود به سیستم از راه دور[۱] ، احراز هویت طرفین ، نگهداری این نشست و توانایی از سرگیری یک نشست در هنگام قطع ارتباط می‌باشد.

---

[۱] Connection Oriented
[۲] Segment

در این جا نیز واحد اطلاعات پیام است. وظایف این لایه را می‌توان در موارد زیر خلاصه کرد:

- برقراری و مدیریت یک جلسه
- شناسایی طرفین
- مشخص نمودن اعتبار پیامها
- اتمام جلسه
- حسابداری مشتری‌ها[۲]

### ۶-۶) لایهٔ ارائه (نمایش)

در این لایه معمولا کارهایی صورت می گیرد که اگرچه بنیادی و اساسی نیستند ولیکن به عنوان نیازهای عمومی تلقی می شوند. مثلا :

- فشرده‌سازی فایل[۳]
- رمزنگاری[٤] برای ارسال داده‌های محرمانه
- رمزگشایی[٥]
- تبدیل کدها به یکدیگر وقتی که دو ماشین از استانداردهای مختلفی برای متن استفاده می‌کنند. (مثل تبدیل متون EBCDIC به ASCII و بالعکس)

### ۶-۷) لایهٔ کاربرد

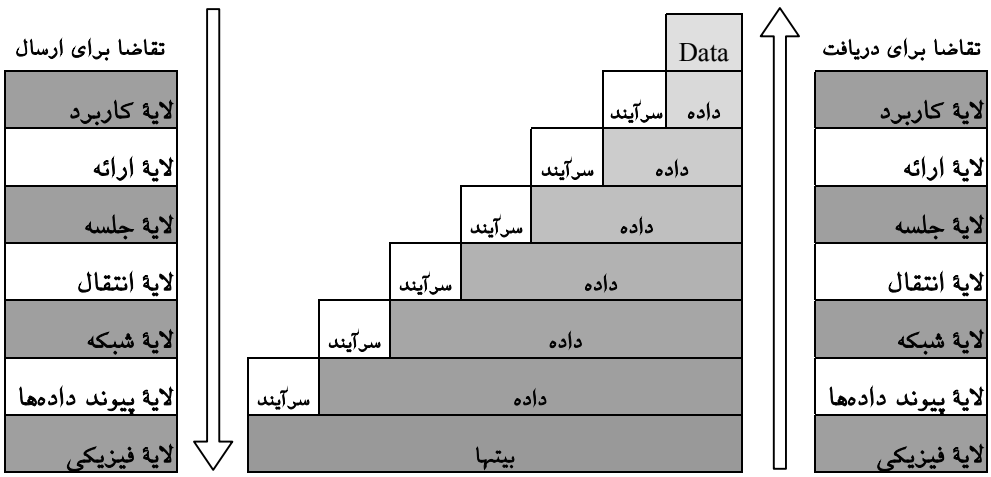
در این لایه ، استاندارد مبادلهٔ پیام بین نرم‌افزارهائی که در اختیار کاربر بوده و به نحوی با شبکه در ارتباطند ، تعریف می‌شود. لایهٔ کاربرد شامل تعریف استانداردهایی نظیر انتقال نامه‌های الکترونیکی ، انتقال مطمئن فایل ، دسترسی به بانکهای اطلاعاتی راه دور ، مدیریت شبکه و انتقال صفحات وب است.

در مدل لایه‌ای شبکه ، وقتی یک برنامهٔ کاربردی در لایهٔ آخر اقدام به ارسال یک واحد اطلاعات می‌نماید ، سرآیند لازم به آن اضافه شده و از طریق صدا زدن توابع سیستمی استاندارد به لایهٔ زیرین تحویل داده می‌شود. لایهٔ زیر نیز پس از اضافه کردن سرآیند لازم ، آنرا

---

۱ Remote Login
۲ Accounting
۳ Data Compression
٤ Encryption
٥ Decryption

به لایهٔ پایین تحویل میدهد و این روند تکرار می‌شود تا آن واحد اطلاعات روی کانال فیزیکی ارسال شود. در مقصد پس از دریافت یک واحد اطلاعات از روی خط فیزیکی ، تحویل لایهٔ بالاتر شده و در هر لایه پس از تحلیل و پردازش لازم ، سرآیند اضافه شده را حذف و به لایهٔ بالاتر تحویل می‌دهد. در شکل (۱۱-۱) روند حذف و اضافه شدن سرآیند در هر لایه به تصویر کشیده شده است.




شکل (۱۱-۱) روند حذف و اضافه شدن سرآیند در هر لایه

## ۷) مدل چهار لایه‌ای TCP/IP

مدل TCP/IP زاده جنگ سرد در دهه شصت بود. در اواخر دههٔ شصت ، آژانس پروژه‌های پیشرفتهٔ تحقیقاتی دولت ایالات متحده (ARPA[1]) با بودجهٔ دولتی ، تصمیم به پیاده‌سازی یک شبکهٔ WAN در نُه ایالت آمریکا گرفت. این شبکه صرفاً اهداف نظامی را دنبال می‌کرد و در عرض دو سال پیاده‌سازی و نصب شد. برای اولین بار روش "سوئیچ بسته" در این شبکه معرفی شد و موفقیت این شبکه مراکز تحقیقاتی مختلف را بر آن داشت تا شروع به کار

[1] Advanced Research Project Agency

مشترک برای توسعهٔ تکنولوژی شبکه نمایند. کمیتهٔ ARPA که به ICCB[۱] مشهور شد روز به روز شهرت یافت و رشد کرد. این کمیته با همکاری بقیهٔ آژانسهای تحقیقاتی ، کار مشترک تبدیل تکنولوژی ARPA به یک پروتکل شبکه‌ای استاندارد به نام TCP/IP[۲] را شروع کردند. در اوایل دههٔ هشتاد محیطهای دانشگاهی نیز از TCP/IP حمایت کردند. دانشگاه برکلی در کالیفرنیا در نسخهٔ یونیکس خود که رایگان بود ، پروتکل TCP/IP را پیاده‌سازی و ارائه کرد. رایگان بودن این سیستم عامل بسیار قدرتمند باعث شد تا دانشکده‌های علوم کامپیوتری به سرعت با TCP/IP آشنا شده و ضمن پیاده‌سازی شبکه‌های مبتنی بر آن ، از این مدل حمایت نمایند. شاید بزرگترین عامل توسعه و رشد TCP/IP همین کار دانشگاه برکلی در ارائهٔ رایگان TCP/IP بر روی یونیکس بود.

در سال ۱۹۸۳ کمیته ICCB بعنوان گروه طراحی اینترنت یا IAB[۳] به جهان معرفی شد. این کمیته یک سازمان مستقل برای طراحی استانداردها و ترویج تحقیقات در زمینهٔ تکنولوژی اینترنت است. کمیتهٔ IAB اکنون نیز وجود دارد و در دو قسمت فعالیت می‌کند:

- گروه IETF[٤] : موارد فنی و مشکلات استانداردها و تکنولوژی بکار گرفته شده در شبکهٔ اینترنت را بررسی و حل می‌کند و جزئیات پروتکلهای فعلی را در اختیار عموم قرار می‌دهد.
- گروه IRTF[٥] : کار تحقیقات به منظور بهبود و ارتقاء اینترنت را بر عهده دارد.

موفقیت IAB در اواسط دههٔ هشتاد سرمایه‌ها را به سمت شبکه سوق داد. سازمان ملی علوم آمریکا تصمیم به سرمایه‌گذاری برای راه‌اندازی یک ستون فقرات در آمریکا گرفت که NSFNET نامیده شد. با پیاده‌سازی موفق این ستون‌فقرات ، اینترنت باز هم رشد کرد و باز هم سرمایه‌ها را به سمت خود کشید و مرزهای آمریکا را در نوردید و به یک پدیدهٔ جهانی تبدیل شد.

مدیریت روزانه و پشتیبانی فنی شبکهٔ اینترنت ، توسط مرکزی در آمریکا به نام INTERNIC[٦] انجام می‌شود. این مرکز مدیریت سطح بالای شبکه ، ثبت اسامی نمادین در اینترنت و ثبت کلاسهای آدرس یکتا را برعهده دارد. شکل (۱۲-۱) سایت

---

[۱] Internet Control and Configuration Board
[۲] Transport Control Protocol/Internet Protocol
[۳] Internet Architecture Board
[٤] Internet Engineering Task Force
[٥] Internet Research Task Force
[٦] Internet Network Information Center

INTERNIC را در اینترنت نشان می‌دهد. این مرکز ، استانداردهای اینترنت و تکنولوژیهای مرتبط با آن را که مورد تایید IAB است ، تحت مستنداتِ دقیق و کاملی به نام RFC[1] به دنیا عرضه می‌کند. برخی از این مستندات خود به اندازهٔ یک کتاب ، مفصل است. سایت اصلی ارائه کنندهٔ این مستندات INTERNIC است ولی سایتهای مختلفی در دنیا موجودند که نسخه‌ای از این مستندات را در اختیار دارند؛ به عنوان مثال می‌توانید از سایت زیر در دانشگاه اوهایو استفاده کنید:

**http://www.cis.ohio-state.edu/hypertext/information/rfc.html**

امروزه TCP/IP به عنوان محبوبترین پروتکل شبکه در تمام سیستمهای عامل حمایت می‌شود و با تمام نقایصی که دارد حتی در پیاده‌سازی اینترانتهایی که حتی به اینترنت متصل نیستند ، مورد استفاده قرار می‌گیرد.


شکل (۱۲-۱) سایت INTERNIC در اینترنت

[1] Request For Comment

### ۱-۷) مولفه‌های TCP/IP

عاملی که تمامی شبکه های مختلف دنیا را به صورت موفقیت‌آمیز به هم پیوند زده است ، تبعیت همه آنها از مجموعه پروتکلی است که تحت عنوان TCP/IP در دنیا شناخته می‌شود. دقت کنید که عبارت خلاصه شده TCP/IP می تواند به دو موضوع متفاوت اشاره داشته باشد:

- **مدل TCP/IP**[1] : این مدل یک ساختار چهار لایه‌ای برای ارتباطات گسترده تعریف می‌نماید که آنرا در ادامه تشریح می‌کنیم.
- **پشتهٔ پروتکلهای TCP/IP**[2] : پشتهٔ TCP/IP مجموعه‌ای شامل بیش از صد پروتکل متفاوت است که برای سازماندهی کلیه اجزاء شبکهٔ اینترنت به کار می‌رود. در این کتاب تعدادی از این پروتکلها را که عمومیت و استفادهٔ فراگیر دارند، معرفی می‌کنیم.

نکته ای که بایستی در اینجا ذکر کرد آنست که برخی از اصطلاحات در مدل TCP/IP نسبت به آنچه که در مدل OSI (هفت لایه‌ای) مطرح می‌شود ، متفاوت است و این موضوعی است که نباید شما را به اشتباه بیندازد . بعنوان مثال اگردر مدل TCP/IP به دیتاگرام اشاره شد ، به یک واحد (بسته ) از اطلاعات اطلاق می‌شود که آیتم و داده های لازم برای مسیریابی روی شبکه به آن اضافه شده است . به مرور با فرهنگِ اصطلاحات و اختصاراتِ مهندسی اینترنت آشنا می شویم و یک لغت نامه کوچک فراهم می کنیم.

### ۲-۷) مدل TCP/IP

همانگونه که اشاره شد این مدل یک ساختار چهارلایه‌ای برای شبکه عرضه کرده است. شکل (۱۳-۱) این مدل را به تصویر کشیده است. اگر بخواهیم این مدل چهار لایه‌ای را با مدل OSI مقایسه کنیم ، لایهٔ اول از مدل TCP/IP یعنی لایهٔ دسترسی به شبکه تلفیقی از وظائف لایهٔ فیزیکی و لایهٔ پیوند داده‌ها از مدل OSI خواهد بود. لایهٔ دوم از مدل TCP/IP معادل لایهٔ سوم از مدل OSI یعنی لایهٔ شبکه است. لایهٔ سوم از مدل TCP/IP همنام و معادل با لایهٔ چهارم از مدل OSI یعنی لایهٔ انتقال خواهد بود .

---

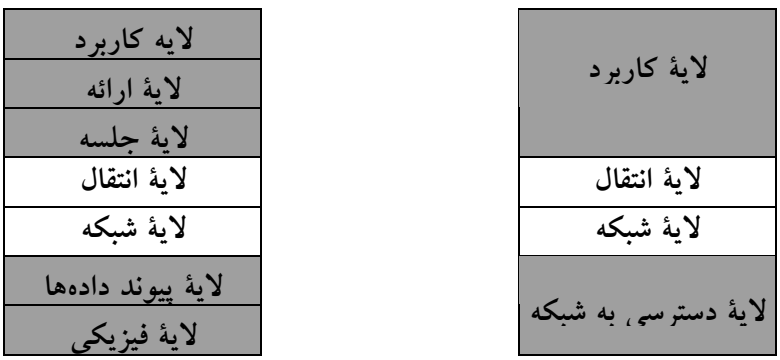
[1] TCP/IP Model
[2] TCP/IP Protocol Stack

| لایه‌ها | نامهای معادل در برخی از کتب |
|---|---|
| **Application layer**<br>**لایه کاربرد** | ➢ لایه سرویسهای کاربردی |
| **Transport layer**<br>**لایه انتقال** | ➢ لایه ارتباط میزبان به میزبان (Host to Host) |
| **Network layer**<br>**لایه شبکه** | ➢ لایه اینترنت<br>➢ لایه ارتباطات اینترنت |
| **Network Interface**<br>**لایه دسترسی به شبکه** | ➢ لایه میزبان به شبکه (Host to network)<br>➢ لایه رابط شبکه |

شکل (۱۳-۱) مدل چهار لایه‌ای TCP/IP

لایه پنجم (جلسه) و لایه ششم (ارائه) از مدل OSI در مدل TCP/IP وجود ندارند و وظائف آنها در صورت لزوم در لایه چهارم از مدل TCP/IP ادغام شده است .
لایه هفتم از مدل OSI معادل بخشی از لایه چهارم از مدل TCP/IP است .
در شکل (۱۴-۱) دو مدل TCP/IP و OSI با هم مقایسه شده‌اند.

در ادامه چهارلایهٔ مدل TCP/IP را بررسی خواهیم کرد.




شکل (۱۴-۱) مقایسهٔ دو مدل TCP/IP و OSI

### ۳-۷) لایهٔ اول از مدل TCP/IP : لایهٔ واسط شبکه

در این لایه استانداردهای سخت‌افزار ، نرم افزارهای راه‌انداز[۱] و پروتکلهای شبکه تعریف می‌شود. این لایه درگیر با مسائل فیزیکی ، الکتریکی و مخابراتی کانال انتقال ، نوع کارت شبکه و راه‌اندازهای لازم برای نصب کارت شبکه می‌باشد. در شبکهٔ اینترنت که می تواند مجموعه ای از عناصر غیر همگن و نامشابه را به هم پیوند بزند انعطاف لازم در این لایه برای شبکه‌های گوناگون و ماشینهای میزبان فراهم شده است. یعنی الزام ویژه‌ای در بکارگیری سخت‌افزار ارتباطی خاص ، دراین لایه وجود ندارد. ایستگاهی که تصمیم دارد به اینترنت متصل شود بایستی با استفاده از پروتکلهای متعدد و معتبر و نرم‌افزار راه‌انداز مناسب ، به نحوی داده‌های خودش را به شبکه تزریق کند. بنابراین اصرار و اجبار خاصی در استفاده از یک استاندارد خاص در این لایه وجود ندارد. تمام پروتکلهای LAN و MAN در این لایه قابل استفاده است. در فصل بعدی ضمن معرفی مختصر پروتکلهای شناخته شدهٔ LAN ، دو پروتکل مهم به نامهای PPP و SLIP را که برای ارسال بسته های اطلاعاتی روی خطوط نقطه به نقطهٔ سریال کاربرد دارد و حالت سنکرون و آسنکرون را پشتیبانی می‌کند ، معرفی می‌کنیم . کاربران اینترنت به خوبی از وجود چنین پروتکلهایی مطلعند و میلیونها نفردر جهان که از خطوط تلفنی[۲] برای اتصال به شبکهٔ اینترنت بهره می‌گیرند ، از پروتکل PPP ( یا در فرم ساده‌ترSLIP) استفاده می‌کنند .

یک ماشین میزبان می‌تواند از طریق شبکهٔ محلی ، فریمهای اطلاعاتی را به زیرشبکه تزریق کند به این نحو که بسته‌های راه دور[۳] را که مقصدشان خارج از شبکهٔ محلی است ، به مسیریاب از پیش تعریف شده ، هدایت نماید. شبکه‌های محلی از طریق یک یا چند مسیریاب می‌توانند به اینترنت متصل شوند . بنابراین یک بسته اطلاعاتی که از لایه بالاتر جهت ارسال به یک مقصد ، به لایه اول در مدل TCP/IP تحویل می‌شود ، نهایتاً در قسمت "فیلد داده"[۴] از فریم شبکه محلی قرار می‌گیرد و مسیر خود را آغاز می‌نماید؛ بنابراین باید با تمام "قالبهای فریم"[۵] از شبکه های محلی مهم مثل اترنت[۶] ، شبکهٔ حلقه (توکن رینگ)[۷] آشنا شویم. پروتکلهائی که

---

۱ Device Driver
۲ Dial up
۳ Distant Packet
۴ Data Field/Payload
۵ Frame Format
۶ Ethernet / IEEE 802.3
۷ Token Ring / IEEE 802.5

در لایه اول از مدل TCP/IP تعریف می‌شوند ، می‌توانند مبتنی بر ارسال رشتهٔ بیت[۱] یا مبتنی بر ارسال رشتهٔ بایت[۲] باشند.

### ۴-۷) لایهٔ دوم از مدل TCP/IP : لایهٔ شبکه

این لایه در ساده ترین عبارت وظیفه دارد بسته های اطلاعاتی را که از این به بعد آنها را بسته های IP می‌نامیم، روی شبکه هدایت کرده و از مبدأ تا مقصد به پیش ببرد. در این لایه چندین پروتکل در کنار هم وظیفه مسیریابی و تحویل بسته های اطلاعاتی از مبدأ تا مقصد را انجام می دهند. کلیدی‌ترین پروتکل در این لایه ، پروتکل IP نام دارد. برخی از پروتکلهای مهم که یک سری وظائف جانبی برعهده دارند عبارتند از : ARP – RARP - RIP - ICMP - IGMP - BOOTP و ... . این پروتکلها را به اختصار توضیح خواهیم داد ولی بیشترین تلاش ما در کالبدشناسی پروتکل IP خواهد بود.

همانگونه که اشاره شد در این لایه یک واحد اطلاعاتی که بایستی تحویل مقصد شود ، دیتاگرام نامیده می شود. پروتکل IP می‌تواند یک دیتاگرام را در قالب بسته‌های کوچکتری قطعه‌قطعه کرده و پس از اضافه کردن اطلاعات لازم برای بازسازی ، آنها را روی شبکه ارسال کند.

لازم است بدانید که در این لایه برقراری ارتباط بین مبدأ و مقصد بروش "بدون اتصال" خواهد بود و ارسال یک بسته IP روی شبکه ، عبور از مسیر خاصی را تضمین نمی‌کند. یعنی اگر دو بسته متوالی برای یک مقصد یکسان ارسال شود هیچ تضمینی در به ترتیب رسیدن آنها وجود ندارد ، چون این دو بسته می‌توانند از مسیرهای متفاوتی به سمت مقصد حرکت نمایند. در ضمن در این لایه پس از آنکه بسته‌ای روی یکی از کانالهای ارتباطی هدایت شد ، از سالم رسیدن یا نرسیدن آن به مقصد هیچ اطلاعی بدست نخواهد آمد ، چرا که در این لایه ، برای بسته‌های IP هیچ گونه پیغام دریافت یا عدم دریافت[۳] بین عناصر واقع بر روی مسیر ، رد و بدل نمی‌شود؛ بنابراین سرویسی که در این لایه ارائه می شود نامطمئن است و اگر به سرویسهای مطمئن و یا اتصال‌گرا نیاز باشد در لایه بالاتر این نیاز تامین خواهد شد.

---

۱ Bit oriented در اینجا کوچکترین واحد اطلاعات که می‌تواند بطور مستقل ارسال شود یک بیت خواهد بود.
۲ Byte Oriented در اینجا کوچکترین واحد اطلاعات که می‌تواند بطور مستقل ارسال شود یک بایت خواهد بود.
۳ Ack/Nack

دراین لایه مسیریابها بایستی از شرایط توپولوژیکی و ترافیکی شبکه اطلاعاتی را کسب نمایند تا مسیریابی بروش پویا انجام شود. همچنین در این لایه باید اطلاعاتی درباره مشکلات یا خطاهای احتمالی در ساختار زیرشبکه بین مسیریابها و ماشینهای میزبان ، مبادله شود. یکی دیگر از وظائف این لایه ویژگی ارسال "چندپخشی"[۱] است یعنی یک ایستگاه قادر باشد به چندین مقصد گوناگون که در قالب یک گروه سازماندهی شده‌اند ، بسته یا بسته‌هائی را ارسال نماید.

### ۷-۵) لایهٔ سوم از مدل TCP/IP : لایهٔ انتقال

این لایه ارتباط ماشینهای انتهایی (ماشینهای میزبان) را در شبکه برقرار می کند یعنی می‌تواند بر اساس سرویسی که لایه دوم ارائه می کند یک ارتباط اتصال گرا و مطمئن[۲] ، برقرار کند. البته در این لایه برای عملیاتی نظیر ارسال صوت و تصویر که سرعت مهمتر از دقت و خطا است سرویسهای بدون اتصالِ سریع و نامطمئن نیز فراهم شده است.

در سرویس مطمئنی که در این لایه ارائه می شود ، مکانیزمی اتخاذ شده است که فرستنده از رسیدن و یا عدم رسید صحیح بسته به مقصد باخبر شود . در مورد سرویسهای مطمئن و نامطمئن بعداً بحث خواهد شد. این لایه از یکطرف با لایهٔ شبکه و از طرف دیگر با لایهٔ کاربرد در ارتباط است. داده‌های تحویلی به این لایه توسط برنامهٔ کاربردی و با صدا زدن توابع سیستمی تعریف شده در "واسط برنامه‌های کاربردی" – API[۳] – ارسال یا دریافت می‌شوند.

### ۷-۶) لایهٔ چهارم از مدل TCP/IP : لایهٔ کاربرد

در این لایه بر اساس خدمات لایه‌های زیرین ، سرویس سطح بالایی برای خلق برنامه‌های کاربردی ویژه و پیچیده ارائه می‌شود. این خدمات در قالب ، پروتکلهای استانداردی همانند موارد زیر به کاربر ارائه می‌شود.

- شبیه سازی ترمینال[۴]

---

[۱] Multicast
[۲] Reliable
[۳] Application Program Interface
[۴] TELNET / Teminal Emulation

- انتقال فایل یا FTP[1]
- مدیریت پست الکترونیکی
- خدمات انتقال صفحات ابرمتنی
- ....

در مورد سرویسهای این لایه در فصولی مجزا بحث خواهیم کرد.

درپایان این قسمت بایستی خاطر نشان کنیم که ارسال یک واحد اطلاعاتی از لایه چهارم پس از انجام پردازشهای لازم در لایه های زیرین به نحو مناسبی روی زیرشبکه تزریق شده و نهایتاً در ماشین مقصد، تحویل یک برنامه کاربردی خاص خواهد شد.

## ۸ ) مراجع این فصل

مجموعهٔ مراجع زیر می‌توانند برای دست آوردن جزییات دقیق و تحقیق جامع در مورد مفاهیم معرفی شده در این فصل مفید واقع شوند.


| "Computer Networks" , Andrew S.Tanenbaum, Third Edition, Prentice-Hall, 1996. | |
|---|---|
| **RFC1360** | "IAB Official Protocol Standards," Postel, J.B.; 1992 |
| **RFC1340** | "Assigned Numbers," Reynolds, J.K.; Postel, J.B.; 1992 |
| **RFC1208** | "Glossary of Networking Terms," Jacobsen, O.J.; Lynch, D.C.; 1991 |
| **RFC1180** | "TCP/IP Tutorial," Socolofsky, T.J.; Kale, C.J.; 1991 |
| **RFC1178** | "Choosing a Name for Your Computer," Libes, D.; 1990 |
| **RFC1175** | "FYI on Where to Start: A Bibliography of Inter-networking Information," Bowers, K.L.; LaQuey, T.L.; Reynolds, J.K.; Reubicek, K.; Stahl, M.K.; Yuan, A.; 1990 |
| **RFC1173** | "Responsibilities of Host and Network Managers: A Summary of the Oral Tradition of the Internet," vanBokkelen, J.; 1990 |
| **RFC1166** | "Internet Numbers," Kirkpatrick, S.; Stahl, M.K.; Recker, M.; 1990 |


[1] File Transfer Protocal

| | |
|---|---|
| **RFC1127** | "Perspective on the Host Requirements RFCs," Braden, R.T.; 1989 |
| **RFC1123** | "Requirements for Internet Hosts—Application and Support," Braden, R.T., ed; 1989 |
| **RFC1122** | "Requirements for Internet Hosts—Communication Layers," Braden, R.T., ed; 1989 |
| **RFC1118** | "Hitchhiker's Guide to the Internet," Krol, E., 1989 |
| **RFC1011** | "Official Internet Protocol," Reynolds, J.R.; Postel, J.B.; 1987 |
| **RFC1009** | "Requirements for Internet Gateways," Braden, R.T.; Postel, J.B.; 1987 |
| **RFC980** | "Protocol Document Order Information," Jacobsen, O.J.; Postel, J.B.; 1986 |

## ۱) لایهٔ واسط شبکه

در فصل اول آموختیم که مسائل مربوط به برقراری ارتباط فیزیکی بین دو ماشین در یک شبکهٔ کامپیوتری که مستقیماً از طریق یک محیط میانی[۱] به هم متصل شده‌اند ، در لایه اول از مدل TCP/IP مطرح می‌شود. این لایه موظف است تدابیری اتخاذ کند تا یک کانال دارای خطا ، به یک خط مطمئن و بدون خطا تبدیل شود. در راستای این وظیفه ، اطلاعاتی که قرار است روی خط ارسال شوند ، در قالب یک "فریم" سازماندهی شده و ابتدا و انتهای آنها با علامتهای ویژه نشانه‌گذاری[۲] می‌شود تا گیرندهٔ اطلاعات بتواند مرز فریمهای متوالی را تشخیص بدهد. همچنین به ابتدا و انتهای هر فریم ، اطلاعات لازم مثل آدرس گیرنده و فرستندهٔ فریم و کدهای کشف خطا اضافه می‌شود. در شبکه‌هائی که از کانال اشتراکی استفاده می‌کنند ، وظیفهٔ جلوگیری از تصادم سیگنال[۳] و مدیریت کانال ، بر عهدهٔ سخت‌افزار این لایه است.

در فصل قبل اشاره شد که از دیدگاه کانال انتقال ، دو دسته شبکهٔ "کانال فراگیر/ اشتراکی" و "کانال نقطه به نقطه" تعریف شده است. پروتکلهای مورد نیاز برای انتقال فریم روی کانالهای نقطه به نقطه ، تفاوت ذاتی با پروتکلهای کانال اشتراکی دارند ، زیرا کانالهای نقطه به نقطه مشکل مدیریت کانال و پدیدهٔ تصادم را نخواهند داشت. در این فصل آن دسته از پروتکلهای شناخته شده و جهانی ، که در لایهٔ "واسط شبکه" از مدل TCP/IP تعریف شده ، بررسی خواهد شد.

برای انعطاف بیشتر در شبکهٔ اینترنت ، که مجموعه‌ای از عناصر غیرهمگن و نامشابه را به هم پیوند زده ، این لایه بسیار باز و منعطف تعریف شده است ، یعنی الزام ویژه‌ای در بکارگیری سخت‌افزار ارتباطی خاص و پروتکل ارتباطی معین ، در این لایه وجود ندارد. ایستگاهی که تصمیم دارد به اینترنت متصل شود باید با بهره‌گیری از یک پروتکل ارتباطی معتبر و نرم‌افزار راه‌انداز مناسب ، به نحوی داده‌های خودش را به شبکه تزریق کند. بنابراین اصرار و اجبار خاصی در استفاده از یک استاندارد خاص در لایهٔ اول از مدل TCP/IP تعیین نشده است.

تقریباً در اکثر شبکه‌های محلی ، ماشینهای شبکه از یک کانال مشترک استفاده می‌کنند. ولی معمولاً دو شبکهٔ مجزا ، با استفاده از مسیریاب و خطوط نقطه به نقطه[۴]

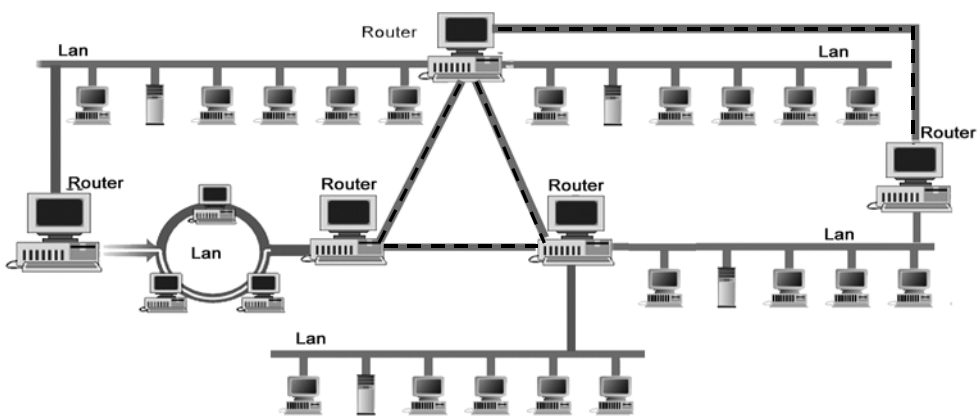
---

۱ Medium ( Channel )
۲ Delimiter
۳ Collision
۴ Point to Point

به یکدیگر متصل می‌شوند. منظور از خطوط نقطه به نقطه خطوطی است که ارتباط دو ماشین روبرو را برقرار می‌کنند و هیچ ماشین ثالثی در این کانال سهیم نیست. این خطوط می‌توانند خطوط اجاره‌ای[1] ، خطوط تلفن معمولی ، کانالهای اختصاصی مایکروویو و یا کانالهای ماهواره‌ای باشند. از آنجایی که تمام ارتباطات زیر شبکه ، از طریق مسیریاب (یا مراکز سوئیچ سلول[2]) انجام می‌شود لذا هر یک از این مسیریابها یک (یا تعدادی) خط اختصاصی با دیگر مسیریابها (یا سوئیچها) دارند که به این خطوط اختصاصی در یک اصطلاح عام ، **"لینک"**[3] گفته می‌شود.

به شکل (۱-۲) نگاه کنید. در این شکل ، پنج شبکهٔ محلی متفاوت از طریق مسیریابها به هم متصل شده‌اند. خطوط بین مسیریابها که در این شکل به صورت نقطه‌چین نشان داده شده‌اند ، خطوط نقطه به نقطه و اختصاصی محسوب می‌شوند. بقیهٔ کانالها ، اشتراکی و فراگیر هستند.

یک بسته برای طی مسیر از مبدأ به مقصد ، باید از شبکه‌ها و کانالهای متفاوت عبور کند. با تغییر شبکه و کانال ، ساختار فریم توسط واسط شبکه تعویض می‌شود. چیزی که در طی مسیر و تغییرات مداوم فریم ، تغییر نخواهد کرد و ساختار آن از مبدأ تا مقصد ، استاندارد و بدون تغییر باقی خواهد ماند ، **"ساختمان داده‌ایست که درون فیلدِ داده از این فریمها قرار گرفته و بستهٔ IP نام دارد"**.




شکل (۱-۲) شمای یک شبکهٔ فرضی

---

[1] Leased line
[2] Cell Switchs
[3] Link

در ادامهٔ این فصل ابتدا مسئلهٔ خطا و چگونگی کشف آن را در داده‌ها مطرح می‌کنیم و سپس مشخصات کانالهای فیزیکی مختلف را که مرتبط با لایهٔ واسط شبکه هستند ، به اختصار یادآوری می‌نماییم. سپس به معرفی پروتکلها و استانداردهای تعریف شده در این لایه خواهیم پرداخت.

### ۱-۱) مختصری در مورد کانالهای انتقال

همانگونه که گفته شد وظیفهٔ سخت‌افزار مخابراتی در لایهٔ واسط شبکه آنست که بدون توجه به نوع و محتوای داده‌ها ، بیتهای داده را بر روی کانال فیزیکی منتقل کند. سخت‌افزار انتقال در این لایه ، بیشتر با مسائل مخابراتی و الکتریکی سروکار دارد. سه جز اصلی این سخت‌افزار ، عبارتند از:

- گیرنده
- فرستنده
- کانال فیزیکی[۱]

مباحث مربوط به گیرنده/فرستنده در محدودهٔ این کتاب نیست ، تنها به معرفی کانالهای ارتباطی که برای اتصال بین ماشینها استفاده می‌شود ، اکتفا می‌کنیم. این کانالها عبارتند از :

- **خطوط تلفن**
- **سیمهای به هم بافته شدهٔ زوجی** (در انواع مختلف مثل UTP[۲] که یک زوج سیم معمولی به هم بافته‌شده است و STP[۳] که زوج سیم معمولی به هم بافته‌شده به همراه یک پوشش آلومینیمی بر روی آنها جهت کاهش اثر نویزهای محیطی بر روی سیم می‌باشد.)
- **کابلهای هم‌محور (کواکسیال)** (در انواع مختلف مثل کابل کواکس ۵۰ اهم ضخیم[۴] ، کابل کواکس ۵۰ اهم نازک[۵] و کابل کوآکس ۷۵ اهم معمولی)
- فیبرهای نوری (در انواع مختلف مثل فیبر تک‌موده و چندموده[۶])
- **کانالهای ماهواره‌ای** : در باندهای فرکانسی مختلف مثل :

  باند **C** : ارسال از زمین به ماهواره در باند 5.925~6.425 GHz

  دریافت از ماهواره در باند 3.7~4.2 GHz

---

۱ Physical Channel
۲ Unshielded Twisted Pair
۳ Shielded Twisted Pair
۴ Tick Coaxial Cable
۵ Thin Coaxial Cable
۶ Mono mode / Multi mode Fiber Optic

باند **Ku** : ارسال از زمین به ماهواره در باند 14.0~14.5 GHz
دریافت از ماهواره در باند 11.7~12.2 GHz
باند **Ka** : ارسال از زمین به ماهواره در باند 27.5~30.5 GHz
دریافت از ماهواره در باند 17.7~21.7 GHz

- **کانالهای رادیویی** ( شامل باندهای فرکانسی مختلف مثل UHF ، VHF )
- **امواج طیف نوری** شامل نور مادون قرمز (با استفاده از این امواج که خاصیت نور دارند می‌توان داده‌ها را به فاصلهٔ چند متر عبور داد. این امواج فقط از محیطهای شفاف عبور می‌کنند و بیشتر برای انتقال اطلاعات در فواصل بسیار کوتاه کاربرد دارد. مثلاً در کامپیوترهای کیفی برای ارتباط بی‌سیم با یک کامپیوتر دیگر مناسب است.)

تمام کانالها دارای مشخصه‌ای بنام پهنای باند هستند. در یک عبارت ساده و غیر دقیق ، پهنای باند هر کانال را می‌توان ، توانایی و ظرفیت آن در ارسال اطلاعات با نرخ B بیت در هر ثانیه ، تعریف کرد. بنابراین وقتی گفته می‌شود پهنای باند یک کانال یک مگابیت بر ثانیه (1Mbps) است یعنی با سرعت بالاتر از یک مگابیت بر ثانیه نمی‌توان اطلاعات را سالم به مقصد رساند. در این خصوص رابطهٔ معروفی بنام رابطهٔ شانون وجود دارد:

$$C=B.\log_2(1+S/N)$$

C : ظرفیت کانال بر حسب بیت بر ثانیه
S : متوسط توان سیگنال
N : متوسط توان نویز
B : پهنای باند کانال بر حسب هرتز

به عنوان مثال اگر پهنای باند کانالهای تلفن معمولی را حداکثر 4KHz فرض کنیم و نسبت توان سیگنال به توان نویز بطور تقریبی ۱۰۰۰ باشد ، در چنین حالتی با استفاده از خط تلفن حداکثر ۳۹۰۰۰ بیت در ثانیه را می‌توان انتقال داد و انتقال اطلاعات در بالاتر از این نرخ منجر به خرابی داده‌ها خواهد شد. انتقال یک فایل یک مگابایتی از طریق خط تلفن با این سرعت حدوداً ۲۱۰ ثانیه طول خواهد کشید. در جدول (۲-۲) مشخصات برخی از کانالهای انتقال با یکدیگر مقایسه شده است.

معیار خطا در کانالهای انتقال ، احتمال بروز یک بیت خطا روی کانال تعریف می‌شود؛ یعنی احتمال آنکه در فرستنده بیت ۱ ارسال و در گیرنده اشتباهاً بیت ۰ آشکارسازی بشود.

| | پهنای باند | خطا | پیاده سازی | قیمت | توضیح |
|---|---|---|---|---|---|
| خطوط تلفن معمولی | کم (حدود 4KHz) | زیاد | ساده | ارزان | از قبل وجود دارد |
| زوج سیم | متوسط ( حدود چند ده تا صد مگاهرتز ) | متوسط | ساده | ارزان | برای فواصل کوتاه مناسب است. |
| کابلهای کواکس | حدود چند صد مگاهرتز | کم | متوسط | متوسط | |
| فیبرهای نوری | حدود چند گیگا هرتز | بسیار کم | پیچیده | متوسط | بهترین کارآیی |
| کانالهای ماهواره | حدود چند صد مگاهرتز | متوسط | بسیار پیچیده | گران | در همه جا تحت پوشش |
| کانالهای رادیویی | حدود چند مگاهرتز | زیاد | نسبتاً پیچیده | نسبتاً گران | در جایی که کابل‌کشی عقلایی نیست مناسب می‌باشد. |

جدول (۲-۲) مقایسهٔ مشخصاتِ برخی از کانالهای انتقال

با توجه به آنکه پهنای باند بعضی از کانالها بسیار زیاد است (مثل کانالهای ماهواره‌ای ) می‌توان یک کانال فیزیکی را بین چندین ایستگاه تقسیم کرد. این تقسیم باعث می‌شود که از یک کانال مشترک چندین ایستگاه استفاده کنند و هزینه‌های ارتباط کاهش یابد. به عمل تقسیم پهنای باند یک کانال بین چند ایستگاه عمل مالتی‌پلکس یا تسهیم گفته می‌شود. تسهیم به دو روش قابل انجام است:

- تسهیم در میدان فرکانس یا FDM[۱]
- تسهیم در میدان زمان یا TDM[۲]

در روش FDM با فرض آنکه حداکثر N ایستگاه در شبکه وجود داشته باشد ، پهنای باند فرکانسی کانال به N باند مجزا تقسیم می‌شود. هر ایستگاه موظف است در یکی از این باندهای فرکانسی ارسال و دریافت داشته باشد و چون این باند فرکانسی به صورت ثابت ، متعلق به خودش خواهد بود ، هرگونه تصادم و تداخل سیگنال منتفی است.

در روش TDM زمان به بازه‌های کوچکی[۳] تقسیم شده و هر ایستگاه مجاز است فقط در بازهٔ زمانی متعلق به خودش ، اطلاعات را روی کانال بفرستد.

---

۱ Frequency Division Multiplexing
۲ Time Division Multiplexing
۳ Time Slot

روشهای FDM و TDM زمانی کارآمد و مفید خواهند بود که : اولاً تعداد ایستگاهها ثابت و محدود باشد. ثانیاً هر ایستگاه حجم ثابت و در عین حال دائمی ارسال داده بر روی کانال داشته باشد. در شبکه‌های کامپیوتری ایستگاهها از نظر تعداد ، نامشخص و زیادند و ارسال داده‌ها نیز "انفجاری"[1] است. انفجاری بودن ترافیک بدین معناست که ایستگاه در لحظاتی ، بصورت ناگهانی حجم انبوهی از فریمها را برای ارسال روی کانال تولید می‌کند و سپس متوقف شده و تا لحظات متمادی هیچ داده‌ای برای ارسال تولید نمی‌کند. در شبکه‌ها تقاضای ارسال روی کانال پدیده‌ایست تصادفی و هیچ قاعدهٔ از پیش تعیین شده‌ای ندارد. آمارها نشان می‌دهد که انفجاری بودن ترافیک روی شبکه ، نسبت ۱۰۰۰/۱ دارد؛ یعنی :

$$\frac{\text{Peak Traffic}}{\text{Mean Traffic}} = \frac{1000}{1}$$

با این توصیف برای تسهیم کانالهای مشترک باید به سمت روشهای پویا حرکت کرد. در این خصوص پروتکلهای متفاوتی عرضه شده که در این فصل شناخته شده‌ترین آنها را که استانداردهای IEEE 802.x هستند ، معرفی خواهیم کرد.

### ۲-۱) مختصری در مورد خطا در شبکه‌های کامپیوتری

خطا در خطوط انتقال جزو حقایقی است که به هیچ وجه نمی‌توان بطور کامل آن را برطرف کرد و همیشه جزو مشکلات عمدهٔ سیستمهای مخابراتی بوده است. ماهیت خطا و علل بوجود آمدن آن را می‌توان در موارد زیر خلاصه کرد:

- **نویز حرارتی** : این نویز به دلیل حرکت اتفاقی الکترونها بوجود می‌آید و با افزایش دما ، شدت این نویز هم به صورت خطی تقویت می‌شود. بخصوص در مداراتی مثل تقویت کننده‌های نیمه هادی با ضریب تقویت و بهرهٔ بالا ، تاثیر این نویز حساسیت بیشتری دارد. اثر این خطا کاملا تصادفی است.
- **شوک‌های الکتریکی** : این نوع از نویز بدلیل قطع و وصل کلیدها ، سیمها و سوئیچ‌های الکتریکی یا رعد و برق بوجود آمده و نوعی خطای انفجاری را باعث می‌شود؛ یعنی مجموعهٔ گسترده‌ای از بیتها که روی کانال در جریانند ، به یکباره خراب می‌شوند. به

[1] Bursty Traffic

عنوان مثال اگر یک شوک الکتریکی به اندازه 10ms ادامه یابد و اطلاعات روی کانال با سرعت 1Mbps در جریان باشد ، با فرض آنکه طول متوسط فریمها 1KB در نظر گرفته شود ، این شوک می‌تواند تا ده فریم را بطور کلی نابود کند؛ به این معنا که فرستنده ده فریم را فرستاده ولی گیرنده هیچ فریمی دریافت نکرده است.

- **نویز کیهانی:** این نوع خطاها ناشی از حرکات کیهانی ، کهکشانها ، وضعیت ستارگان و خورشید و امثال آن می‌باشد و تاثیر آن بیشتر بر روی کانالهای رادیوئی است.

ساده ترین روش کشف خطا ، اضافه کردن بیت توازن به داده‌هاست. در این روش به ازای هر بایت از اطلاعات یک بیت توازن اضافه می‌شود؛ این بیت باید به گونه‌ای انتخاب و اضافه شود که مجموع تعداد بیتهای ۱ ، همیشه زوج یا فرد باشد.

مثال:

| | |
|---|---|
| بایت اصلی : | 01101001 |
| بیت توان فرد | Odd Parity 1 01101001 |
| بیت توان زوج | Even Parity 0 01101001 |

بنابراین گیرنده می‌تواند با بررسی بیت توازن ، خطای احتمالی را کشف کند ، ولی این روش در صورتی موثر است که تعداد خطاهای رخ داده زوج نباشد.

روش **Checksum** : در این روش تمام بایتهای یک فریم که باید توسط فرستنده ارسال شود ، با هم جمع (یا XOR ) شده و یک بایت به نام Checksum بدست می‌آید. این بایت در انتهای فریم به مقصد ارسال می‌شود. در مقصد مجدداً بایت Checksum محاسبه و سپس مقایسه می‌شود. این روش در صورتی قادر به کشف خطا است که تعداد خطاهای رخ داده در بیتهای هم ارزش زوج نباشد.

**کدهای کشف خطای CRC**[1] : در روش CRC ، به ازای مجموعه ای از بیتها (مثلا ۵۱۲۰ بیت یا ۱۰۲۴۰ بیت ...) تعدادی بیت کنترلی به نام CRC محاسبه و به انتهای فریم اضافه می‌شود. مبنای محاسبهٔ کدهای CRC با استفاده از تقسیم چندجمله‌ای است که روش محاسبهٔ آن با ارائهٔ یک مثال توضیح داده شده است :

[1] Cyclic Redundancy Check

دادهٔ اصلی : 11100101

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |

ابتدا از روی دادهٔ اصلی یک چندجمله‌ای تولید می‌شود. نمایش ریاضی چند جمله‌ای بدینصورت است که بیتها از راست به چپ ضرایب یک چند جمله‌ای قرار می‌گیرند که توان هر جمله را موقعیت بیت در رشته مشخص می‌کند. بدین صورت داده به صورت یک چند جمله‌ای نمایش داده خواهد شد. رشتهٔ بیت در این مثال برای سادگی عملیات ، هشت بیتی فرض شده است ، ولی در عمل این رشته می‌تواند دهها هزار بیت طول داشته باشد.

$$D(X)=1 * x^7 + 1 * x^6 + 1 * x^5 + 0 * x^4 + 0 * x^3 + 1 * x^2 + 0 * x + 1$$
$$D(X)= x^7 + x^6 + x^5 + x^2 + 1$$

برای تولید کد CRC ، چند جمله‌ای D(X). $x^n$ بر یک **"چندجمله‌ای مولد"** که بین گیرنده و فرستنده توافق می‌شود و اختیاری است ، تقسیم می‌گردد. ( n بالاترین توان چندجمله‌ای مولد است. ) تقسیم در مبنای ۲ انجام می‌شود ، یعنی ضرایب جملاتِ با توان مساوی با هم XOR خواهد شدو تفریق معنا ندارد. باقیماندهٔ تقسیم به عنوان کدهای کنترل خطا در انتهای داده‌ها ارسال خواهند شد.

مولد CRC = $X^2+1 \rightarrow 101$

Data = $x^7+x^6+x^5+x^2+1 \xrightarrow{*X^2} X^9 + X^8 + X^7 + X^4 + X^2$

$X^9 + X^8 + X^7 + X^4 + X^2 \,|\, X^2+1$

باقیمانده : $+1 = 01$

در روشی که بخواهیم بصورت باینری ( به جای چند جمله‌ای) کد CRC را حساب کنیم ، به تعداد بزرگترین توان جملهٔ مولد ، در سمت راست رشتهٔ داده صفر اضافه می‌کنیم. بنابراین در مثال بالا باید دو صفر به سمت راستِ داده اضافه شود. سپس داده‌ای که به آن صفر اضافه شده است ، بر چند جمله ای مولد تقسیم می‌شود. در تقسیم به نکات زیر باید توجه داشت:

۱- تقسیم از این لحاظ با تقسیم معمولی متفاوت است که شما باید از سمت چپ بیتهای باقیمانده را صفر کنید.

۲- جمع در مبنای ۲ و به صورت XOR انجام می‌شود.

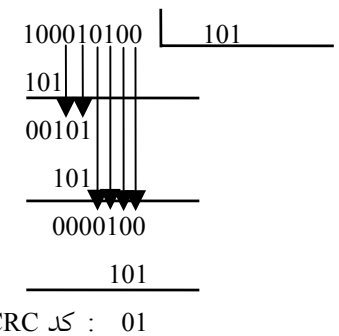
۳- نهایتاً بیتهای باقیماندهٔ تقسیم در سمت راست بیتهای داده قرار می‌گیرد. مثال:

```
1 1 1 0 0 1 0 1 0 0 | 101
1 0 1               | 11010001
  1 0 0
  1 0 1
    1 0 1
    1 0 1
      0 0 0 1 0 0
          1 0 1
            0 1   کد CRC
```

مثال :

Data = $X^6 + X^2 + 1$

CRC Generator = $X^2 + 1$




**مثالی از مولدهای شناخته شده و استاندارد CRC**

CRC-12 = $X^{12} + X^{11} + X^3 + X^2 + 1$

CRC-16 = $X^{16} + X^{15} + X^2 + 1$

CRC-CCITT = $X^{16} + X^{12} + X^5 + 1$

کدهای محاسبه شدهٔ CRC معمولا در انتهای اطلاعات ارسال خواهد شد و پس از دریافت اطلاعات در گیرنده ، مجدداً کدهای CRC برای داده‌ها محاسبه می‌شوند و نتیجه با کد ارسالی CRC مقایسه می‌گردد و در صورت عدم تطابق ، خطایی در داده‌ها وجود دارد و داده‌ها فاقد

اعتبار است. اگر مولد CRC مناسب انتخاب شود ، احتمال آنکه خطایی بروز کند ولی گیرنده قادر به کشف آن نباشد ، کمتر از 0.002 است.

دقت کنید که عمل محاسبهٔ کدهای CRC و همچنین بررسی خطا از طریق تراشه‌های سخت‌افزاری انجام می‌شود تا سرعت عمل بالا برود. این تراشه‌ها بسادگی و از طریق شیفت‌رجیسترهای فیدبک‌دار و گیتهای منطقی ساده مثل XOR پیاده می‌شود.

## ۲) استانداردهای انتقال روی خطوط نقطه به نقطه

در این بخش دو پروتکل ارتباطی برای برقراری یک لینک بین دو ماشین نقطه به نقطه معرفی می‌شود. این دو پروتکل (بالاخص پروتکل دوم یعنی PPP ) زمینه ساز برقراری ارتباط میلیونها نفر در سراسر دنیا با شبکهٔ اینترنت هستند ، چراکه بسیاری از کاربران اینترنت بوسیلهٔ مودم و از طریق خطوط تلفن معمولی به اینترنت متصل می‌شوند که کانالی نقطه به نقطه محسوب می‌شود. معمولاً یکسری از موسسات ارائه‌دهندهٔ خدمات اینترنت که از این به بعد آنها را ISP[1] می‌نامیم ، خدمات اتصال به شبکه اینترنت را برای عموم فراهم می‌کنند. این مراکز ، تعدادی خط تلفن و مودم در اختیار دارند که مشترکین آنها می‌توانند از طریق مودم شماره گیری کرده و پس از برقراری ارتباط ، از خدمات شبکهٔ اینترنت برخوردار شوند.[2] گذشته از ISPهای تجاری ، دانشگاهها و موسسات نیز به کاربران مخصوص خودشان به همین روش سرویس می‌دهند.

سوال آنست که داده‌ها چگونه بین دو ماشین نقطه به نقطه مبادله می‌شوند و چه تمهیداتی برای برقراری یک لینک سریال روی این خطوط اندیشیده شده است.

### ۲-۱) پروتکل SLIP[3]

این پروتکل در سال ۱۹۸٤ توسط ریک آدامز برای اتصال ایستگاههای Sun به وسیلهٔ یک خط سریال مثل خط تلفن ، ابداع شد. این پروتکل که مستندات آن در RFC-1055 تشریح شده است ، فوق‌العاده ساده و در عین حال سریع است. روش کار بدین صورت است که به محض آنکه یک ایستگاه تمایل داشت اطلاعاتی را ارسال

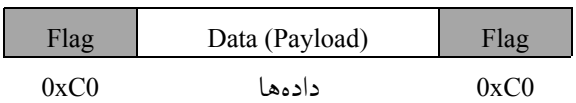
---

[1] Internet Service Providor

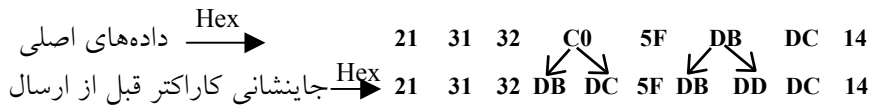
[2] درآمریکا معروفترین آنها AOL , MSN یا همان America online ، Compuserv ، Prodigy هستند.

[3] Serial Line IP

نماید ، علامت مشخصهٔ یک بایتی0xC0 را روی خط ارسال می‌کند و پشت سر آن داده‌ها را روی خط منتقل می‌نماید. برای مشخص کردن انتهای فریم ، پس از ارسال آخرین بایت داده‌ها مجدداً 0xC0 را روی خط می‌گذارد. بنابراین قالب هر فریم در این پروتکل به صورت زیر است :




در ساختار این فریم ، هیچ نکته قابل توضیحی وجود ندارد مگر آنکه بررسی شود اگر در درون قسمت داده‌ها ، کاراکتر 0xC0 وجود داشته باشد ، چه تمهیدی برای جلوگیری از اشتباه در انتهای فریم اندیشیده شده است. پروتکل SLIP برای حل این اشکال از روش "جاینشانی کاراکتر"[1] استفاده کرده است ، یعنی قبل از ارسال داده‌ها روی کانال هرگاه کاراکتر 0xC0 درون داده‌ها پیدا شود ، با دو کاراکتر متوالی (0xDB,0xDC) جایگزین خواهد شد. در ضمن برای آنکه این مشکل مجدداً برای زوج کاراکتر (0xDB,0xDC) تکرار نشود تمام کاراکترهای 0xDB با زوج (0xDB,0xDD) عوض خواهد شد. در چنین حالتی ضمن آنکه داده‌ها در مقصد قابل بازیابی به شکل اصلی هستند ، کاراکتر 0xC0 در درون فیلد داده وجود نخواهد داشت. برای آشنایی با این روش به مثال زیر دقت کنید.




با کمی دقت به قالب فریم در پروتکل SLIP ، متوجه خواهیم شد که این پروتکل از مسائل متعددی رنج می‌برد:

- در این پروتکل هیچ گونه کد کشف خطا گنجانیده نشده است و مسئلهٔ کشف خطاهای احتمالی به لایه‌های بالاتر محول شده است.

[1] Character stuffing

♦ در درون فیلد داده از فریم پروتکل SLIP ، فقط بسته‌های IP قرار می‌گیرد ، در حالی که امروزه در بعضی از شبکه‌ها مثل Novel یا Apple Talk ، ایستگاهها قادرند از طریق خطوط سریال و پروتکلهائی به غیر از IP با ایستگاههای راه دور ارتباط برقرار کنند و SLIP در این شبکه‌ها قابل استفاده نیست.

♦ چون دو ماشین که از طریق پروتکل SLIP با هم ارتباط برقرار می‌کنند دو مرکز رو در رو (نقطه به نقطه) محسوب می‌شوند ، این دو ایستگاه باید آدرسهای IP ثابت و شناخته شده‌ای داشته باشند ولیکن امروزه ارتباطی مورد نیاز است که وقتی یک ماشین میزبان به شبکه وارد شد ، قبل از هر گونه تبادل اطلاعات ابتدا هویت او تائید شده و سپس یک IP موقت به آن تخصیص داده شود. (آدرس IP را بعداً تشریح می‌کنیم.) نهایتاً پس از ختم ارتباط ، آن آدرس IP آزاد شده و برای ماشین دیگری در نظر گرفته شود. پروتکل SLIP چنین ویژگی را پشتیبانی نمی‌کند.

♦ پروتکل SLIP فقط برقرار کننده ارتباط را بعنوان ماشین معتبر می‌شناسد و هیچ راهی برای تائید و احراز هویت کاربر برقرارکننده ارتباط فراهم نکرده است و امنیت شبکه به مخاطره می‌افتد.

♦ متأسفانه بسیاری از سیستمهای عامل از SLIP پشتیبانی نمی‌کنند بهمین دلیل این پروتکل جای خود را به پروتکل جدید تری به نام PPP داده است که در ادامه آنرا معرفی می‌نمائیم. با تمامی معایب گفته شده بدلیل آنکه فریم SLIP فیلدهای سرآیند زیادی ندارد و یک ارتباط بدون انجام مراحل گوناگون ، برقرار می‌شود ، لذا این پروتکل بسیار سریع است.

### ۲-۲) پروتکل PPP[1]

این پروتکل که مستندات آن در RFC-1661 تا RFC-1663 آمده است دارای قالب فریم زیر است:

| 1 Byte | 1 Byte | 1 Byte | 1 or 2 byte | Variable | 2 or 4 | 1 Byte |
|---|---|---|---|---|---|---|
| **Flag 01111110** | Address 11111111 | Control 00000011 | Protocol | Payload | Checksum | **Flag 01111110** |

با یک نگاه ساده و بدون هیچگونه توضیحی به قالب فریم بالا می‌توان متوجه شد که در این پروتکل بسیاری از معایب SLIP رفع شده است. قبل از آنکه

[1] Point to Point Protocol

مشخصات دقیق فریم PPP توضیح داده شود ، بررسی می‌کنیم که وقتی از طریق یک خط سریال نقطه‌به‌نقطه (مثل خط تلفن) می‌خواهید به اینترنت متصل شوید چه اتفاقاتی رخ می‌دهد تا یک ارتباط موفقیت‌آمیز برقرار شود:

در مرحله اول ماشین به کمک مودم شماره‌گیری می‌کند و پس از آنکه مودم طرف مقابل اتصال تلفن را وصل کرد ، ابتدا لازم است یکسری بسته‌های اطلاعاتی کنترلی به نام LCP[۱] بین طرفین رد و بدل شود. فریمهای LCP حاوی اطلاعاتی درون فیلد داده هستند که پارامترهای پروتکل PPP را بصورت توافقی ، انتخاب و تنظیم می‌نمایند (مهمترین بسته‌های LCP در ادامه معرفی خواهد شد). سپس یک سری پارامترهای لایهٔ بالاتر یعنی پروتکل لایهٔ شبکه تنظیم می‌شود. این پارامترها توسط بسته‌هایی که NCP[۲] نامیده می‌شوند ، تنظیم شده و طرفین بر روی این پارامترها توافق می‌کنند. به عنوان مثال کامپیوتر شما که قبل از برقراری ارتباط دارای آدرس IP نیست می‌تواند در هنگام برقراری ارتباط با ISP از طریق رد و بدل کردن فریمهای NCP ، یک IP موقت بگیرد و آدرس IP خودش را تنظیم کند. پس از ختم ارتباط ، آن آدرس IP آزاد شده و می‌تواند به مشترک دیگری اختصاص داده شود. در ضمن در ابتدای برقراری ارتباط طول فیلد داده ، فیلد کشف خطا ، فیلد پروتکل و وجود یا عدم وجود فیلدهای آدرس و کنترل ، توافق می‌شود. به مجموعه این مراحل ، فاز **مذاکره**[۳] گفته می‌شود. سپس مبادلهٔ فریمها آغاز می‌شود. در قسمت فیلد داده[۴] از پروتکل PPP ، می‌تواند بسته‌های IP یا بسته‌های پروتکلهای شناخته شده قرار بگیرد.

حمل بسته‌های مورد نظر ادامه خواهد یافت تا طرفین بر سر ختم ارتباط به توافق برسند. در هنگام ختم ارتباط مجدداً بسته‌های NCP رد و بدل می‌شوند تا همدیگر را از پایان ارتباط مطلع کنند.سپس مجدداً یکسری فریمهای LCP مبادله می‌شود تا طرفین بصورت توافقی ارتباط فیزیکی خودشان را قطع بنماید و خط آزاد شود.

با دقت در قالب فریم PPP مشخص است که ابتدا و انتهای فریم با علامت هشت بیتی 01111110 (0x7E) تعیین می‌شود و بالطبع چنین الگویی نباید در درون اطلاعات وجود داشته باشد. در ضمن برای پیشگیری از اشتباهات ناشی از وجود کاراکترهای کنترلی ASCII ، در این پروتکل وجود کاراکترهای با کد زیر ۳۲ در داده‌ها نیز

---

۱ Link Control Protocol
۲ Network Control Packet
۳ Negotiation Phase
۴ Payload

غیرمجازند. بـهمین دلیـل در ایـن پروتکـل ، در هنگـام وجـود چنیـن الگوهـای غیرمجـاز در درون داده‌ها ، عمل "جاینشانی کـاراکتر" بـه صـورت زیـر انجـام می‌شـود:

بجای کاراکتر با کد 0x7E ، زوج کاراکتر 0x7D-0x5E قرار می‌گیرد.

بجای کاراکتر با کد 0x7D ، زوج کاراکتر 0x7D-0x9D قرار می‌گیرد.

بجای کـاراکتر بـا کدهـای زیـر ۳۲ ابتـدا بیـت ششـم از آن کـاراکتر معکـوس شـده و سـپس کـاراکتر 0x7D قبـل از آن اضافـه می‌شـود. مثـلاً کـاراکتر بـا کـد 0x0A بصـورت 0x7D-0x2A تبدیـل و ارسـال می‌شـود.

حال فیلدهای فریـم را در ایـن پروتکـل بررسـی می‌نمـاییم:

- **Address Field** : تمامـاً ۱ اسـت و طبعـاً بعنـوان یـک آدرس فـراگـیر تلقـی شـده و ایستگاه مقابل موظف است چنین فریمـی را بپذیـرد. ( ایـن فیلـد عمـلاً زائـد اسـت.)

- **Control Field** : ایـن فیلـد در مـورد فریمـهای عـادی مقـدار 00000011 دارد کـه نشـان دهنـده آن اسـت کـه ایـن فریـم شـماره‌گـذاری شـده نیسـت[1] و در نتیجـه ، طرفیـن برای فریمـهای یکدیگـر پیغـام ACK پـس نخواهنـد فرسـتاد. وقتـی کـه یـک فریـم PPP در حـالت عـادی بسـته‌های IP را حمـل می‌کنـد هـر دو فیلـد "آدرس " و "کنـترل" ثـابت و عملاً زائد هستند زیرا کنـترل جریـان داده‌هـا در لایـهٔ سـوم انجـام می‌شـود. البتـه می‌تـوان از ایـن پروتکـل در حـالت شـماره‌گـذاری شـده فریمـها را ارسـال کـرد کـه در اینجـا بـه آن نخواهیـم پرداخـت زیـرا در شـبکهٔ اینـترنت کـاربرد چندانـی نـدارد. ( مسـتندات آن در REC-1663 آمده اسـت ) شـاید سـئوال کنیـد در هنگـام حمـل بسـته‌های IP ، دو فیلـد ثـابت و زائد "آدرس" و "کنترل" چـرا بایسـتی ارسـال شـوند؟ پاسـخ آن اسـت کـه طرفیـن ارتبـاط بـا اسـتفاده از بسـته‌های LCP می‌تواننـد توافـق کننـد کـه در ادامـهٔ ارتبـاط ایـن دو فیلـد حذف شوند. نکته ای که بـاید بـه آن دقـت داشـته باشـید آن اسـت کـه ایـن پروتکـل بـرای یـک اتصـال نقطـه بـه نقطـه تعریـف شـده و بـرای ارتباطـات چنـدگانـه[2] صـادق نیسـت و بهمین دلیل حل بسیاری از مسـائل در چنیـن ارتبـاطی سـاده خواهـد بـود.

- **Protocol** : عددی که در این فیلد قرار می‌گیرد مشخص‌کنندهٔ آنست کـه بسـتهٔ درون فیلـد داده ، مربوط به چه پروتکلی در لایهٔ بالاتر است؛ یعنی پس از دریـافت فریـم ، محتـوای فیلـد

---

[1] Unnumbered Frame
[2] Multidrop

دادهٔ آن برای پردازشهای بعدی باید به کدام پروسس در لایهٔ بالاتر تحویل شود.(مثلاً پروتکلهایی مثل IP ، IPX و ...) در ضمن این عدد می‌تواند مشخص کند که درون فیلد داده ، یک بستهٔ NCP یا LCP قرار دارد.

0xC021 : درون فریم ، بستهٔ LCP قرار دارد.

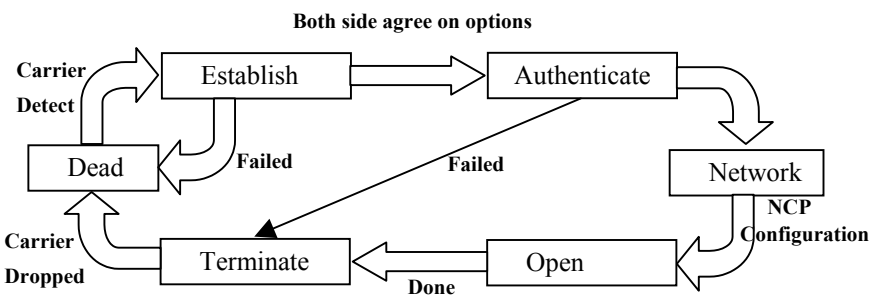
0x8021 : درون فریم ، بستهٔ NCP قرار دارد.

0x0021 : درون فریم ، بستهٔ IP قرار دارد.

- **Payload** : در این فیلد یک بستهٔ مربوط به لایهٔ بالاتر حمل می‌شود. محدودیتی بر روی طول این فیلد وجود ندارد ، ولی اندازهٔ آن توافقی است و در ابتدای برقراری ارتباط بین طرفین توافق می‌شود. در هنگام برقراری ارتباط که هنوز طرفین ارتباط ، روی اندازهٔ مشخصی توافق نکرده‌اند، سایز پیش فرض برای این فیلد ۱۵۰۰ بایت می‌باشد.
- **Checksum** : این فیلد برای کشف خطاهای احتمالی در فریم است و در حالت پیش فرض ۲ بایتی است ولی می‌توان بصورت ٤ بایتی بین طرفین ارتباط توافق کرد.

همانگونه که اشاره شد پروتکل PPP اجازه داده است که درون فیلد داده از هر فریم ، بسته‌های متفاوتی (از لحاظ پروتکل تولید کننده در لایهٔ بالاتر) قرار بگیرد ، بهمین دلیل این انعطاف را دارد که در شبکه‌های گوناگونی بکار گرفته شود.

در شکل (۳-۲) مراحل برقراری و ختم یک ارتباط در پروتکل PPP به صورت یک نمودار حالت تصویر شده است.




شکل (۳-۲) مراحل برقراری و ختم یک ارتباط در پروتکل PPP

شرح "نمودار حالتِ"[۱] شکل (۳-۲) در زیر آمده است :

♦ حالت Dead نشان می‌دهد که خط آزاد است و هیچ سیگنال معتبری روی کانال احساس نمی‌شود.

♦ حالت Establish بدین معناست که روی خط سیگنال معتبری مشاهده شده و یک اتصال فیزیکی برقرار گردیده است ولیکن طرفین ، پارامترهای خود را برای برقراری یک ارتباط تنظیم نکرده‌اند ، بهمین دلیل یک سری بسته‌های LCP بین طرفین رد و بدل شده تا پارامترهای لینک تنظیم شوند. ( پارامترهایی مثل اندازهٔ فیلدها )

♦ حالت Authenticate ، مرحلهٔ بررسی و تائید هویت شروع کنندهٔ ارتباط خواهد بود. اگر یکی از طرفین ، هویت و مشخصات طرف مقابلش را تصدیق نکرد ، ارتباط قطع می‌شود.

♦ حالت Network : در این حالت بسته‌های NCP برای تنظیم پیکربندی پروتکل لایهٔ بالاتر (لایه شبکه) مبادله می‌شوند. پس از تنظیمِ پیکربندی لازم همه چیز برای برقراری یک ارتباط آماده است.

♦ حالت Open : در این حالت ، شرایط برای مبادله و انتقال اطلاعات آماده می‌شود.

♦ حالت Terminate : در این حالت که پس از اتمام مبادلهٔ اطلاعات صورت می‌گیرد ، طرفین با مبادله بسته‌های LCP ، بر سر ختم ارتباط به توافق می‌رسند.

♦ حالت Dead : هر گاه مبادله اطلاعات به اتمام رسید و طرفین با ختم ارتباط موافقت کردند ، کانال به حالت غیرفعال تبدیل شده و عملاً هیچ سیگنال حاملِ معتبری روی آن ارسال نخواهد شد.

### ۱-۲-۲) برخی از بسته‌های مهم LCP

بسته‌های LCP بسیار متنوعند و برای دسترسی به جزئیات دقیق آن بایستی به RFC-1661 مراجعه کرد. در جدول (٤-۲) برخی از این بسته‌ها بصورت اجمالی معرفی شده است. در این جدول ، علامت I[۲] به معنای یکی از طرفین است که به دیگری پیشنهادی را عرضه می‌کند و علامت R[۳] پاسخ‌دهنده به پیشنهاد دهنده است.

---

[۱] State Diagram
[۲] Initiator
[۳] Responder

| نام بسته | جهت | عملکرد |
|---|---|---|
| Configure Request | I ⟶ R | لیستی از گزینه‌ها و مقادیر را برای تنظیم ، پیشنهاد می‌کند. |
| Configure Ack | I ⟵ R | مشخص می‌کند که تمامی پیشنهادات پذیرفته شد. |
| Configure Nack | I ⟵ R | برخی از پارامترها و گزینه‌ها پذیرفته نشد. |
| Configure Reject | I ⟵ R | برخی از پارامترها قابل بحث و توافق نیستند. |
| Terminate Request | I ⟶ R | تقاضا برای خاتمه و قطع ارتباط |
| Terminate Ack | I ⟵ R | موافقت برای قطع ارتباط و کانال |
| Code-Reject | I ⟵ R | تقاضایی رسیده است که شناسایی و فهم نمی‌شود. |
| Echo Request | I ⟶ R | لطفاً عیناً همین بسته را پس بفرستید! |
| Echo Reply | I ⟵ R | بسته پس فرستاده شد! (پاسخ بستهٔ Echo Request ) |
| Discard Request | I ⟶ R | لطفاً این بسته را ندیده بگیرید. (حذف کنید.) |
| Protocol Reject | I ⟵ R | پروتکلی را تعیین کرده‌اید که تشخیص داده نمی‌شود. |

جدول (٤-٢) برخی از بسته‌های LCP

- **Configure Request** : یکی از طرفین ، فهرستی از پارامترها و گزینه‌ها را برای توافق به طرف مقابل عرضه می‌کند. بعنوان مثال می‌توان بر سر اندازهٔ فیلد داده (Payload) ، بودن یا نبودن دو فیلد آدرس و کنترل در خلال ارسال بسته‌های IP (با ارسال این بسته) ، توافق کرد.
- **Configure Ack** : در پاسخ به لیست پارامترهای پیشنهاد شده ، با ارسال این بسته اعلام می‌شود که همهٔ آنها پذیرفته شده و مورد توافق قرار گرفته است.
- **Configure Nack** : برای مخالفت و ابراز عدم توافق بر روی برخی از پارامترهای پیشنهاد شده ، این بسته پس فرستاده می‌شود و در ضمن پارامترهای جدیدی پیشنهاد می‌شود.
- **Configure Reject** : در پاسخ به پیشنهادات عرضه شده ، طرف مقابل با ارسال این بسته ، یکسری از پارامترها را که قابل بحث و توافق نیستند ، مشخص می‌کند. این پارامترها باید بصورت پیش فرض در نظر گرفته شده یا چشمپوشی شود.
- **Terminate Request** : با این بسته یکی از طرفین ، تقاضای پایان دادن به ارتباط را اعلام می‌نماید.

- **Terminate Ack** : با این بسته در پاسخ به تقاضای ختم ارتباط ، طرف مقابل پذیرش ختم ارتباط را اعلام می‌کند.
- **Code-Reject** : هر گاه یکی از طرفین پارامترها و پیشنهاداتی را دریافت کند که آنها را تشخیص نداده یا منظور طرف مقابل را متوجه نشود ، این بسته را در پاسخ به آن ارسال می‌نماید.
- **Echo Request** : یکی از طرفین ارتباط از دیگری می‌خواهد که این بسته را گرفته و مجدداً به خودش برگرداند.
- **Echo Reply** : در پاسخ به تقاضای Echo Request ، طرف مقابل این بسته را پس می‌فرستد.

دو بستهٔ فوق برای عملیات اشکال زدایی و تست ارتباط و همچنین تخمین زمان رفت و برگشت و محاسبهٔ تاخیر کاربرد دارد.

- **Discard Request** : این بسته که می‌تواند توسط هر یک از طرفین ارتباط ارسال شود ، در طرف مقابل نادیده گرفته می‌شود. در حقیقت این بسته نیز برای اشکال‌زدایی فنّی از شبکه به کار می‌رود. ( بعنوان مثال یک ایستگاه که نمی‌داند آیا داده‌هایش روی سیم منتقل می‌شود یا خیر ، می‌تواند با ارسال این بسته موضوع را بررسی کند.)
- **Protocol Reject** : همانگونه که در توضیح قالب فریم پروتکل PPP مشخص شد ، درون فیلد Protocol ( یعنی فیلد چهارم) شمارهٔ پروتکلی قرار می‌گیرد که بستهٔ درون فیلد داده توسط آن پروتکل تولید و ارسال شده است و در طرف مقابل نیز باید تحویل پروسهٔ متناظر با همان پروتکل شود. اگر یکی از طرفین شمارهٔ پروتکل را تشخیص ندهد ، نمی‌داند با بسته درون فیلد داده چکار کند. در این حالت ضمن حذف آن بسته در پاسخ به ارسال کنندهٔ آن ، بستهٔ Protocol Reject را ارسال می‌کند.

در مورد بسته‌های NCP فعلاً مطلبی مطرح نمی‌کنیم ولی همین قدر کافی است که بدانید با استفاده از این بسته‌ها ، می‌توان پارامترهای دینامیکی لایهٔ بالاتر را پیکربندی کرد.

## ۳) استانداردهای واسط شبکه‌های محلی با کانال اشتراکی

در بخش قبلی با دو پروتکل انتقال فریم روی خطوط نقطه به نقطه آشنا شدیم. در این بخش به شبکه‌های متکی به کانالهای مشترک و استانداردهای آنها خواهیم پرداخت. انجمن بین‌المللی مهندسین برق و الکترونیک (IEEE) به عنوان بزرگترین سازمان علمی و تحقیقاتی جهان در زمینهٔ برق ، الکترونیک و کامپیوتر در بسیاری از زمینه‌ها اقدام به تدوین استانداردهای جهانی نموده است که در این بین استانداردهای سری IEEE 802.x در ارتباط با شبکه‌های کامپیوتری تدوین شده‌اند. این استانداردها در خصوص انتقال اطلاعات روی کانال مشترک و مدیریت کانال هستند ، لذا در لایهٔ اوّل از مدل TCP/IP مطرح می‌شوند.[۱] این استانداردها بعداً توسط کمیتهٔ ISO پذیرفته شد و مجدداً تحت نام ISO 8802 معرفی گردید. در این بخش برخی از استانداردهای IEEE سری 802 را توضیح خواهیم داد.

**IEEE 802.1** یک پروتکل شبکه نیست بلکه استانداردی شامل یکسری تعاریف ، تشریح برخی از روشها و مقدمه‌ای در مورد مجموعهٔ استانداردها است. همچنین طریقهٔ دسترسی به سرویس‌های تعریف شده در هر استاندارد و نکات فنی در مورد پروتکلهایی که IEEE برای شبکه‌ها ارائه کرده ، در این استاندارد تشریح شده است.

**IEEE 802.2** یک زیرلایه به نام LLC[۲] تعریف کرده است تا اولاً جزئیات سخت‌افزاری و توپولوژی شبکه را پنهان کند؛ (یعنی با استفاده از این زیرلایه ، شبکه‌های محلی با توپولوژیهای متفاوت همگی از لحاظ سرویسهایی که به لایهٔ بالاتر ارائه می‌دهند ، یکسان‌سازی خواهند شد.) ثانیاً با استفاده از این زیرلایه سرویس انتقال فریمها مطمئن خواهد شد ، به گونه‌ای که ضمن شماره‌گذاری فریمها ، برای آنها پیغام تصدیق (Ack) مبادله شده و بر روی جریان فریمها نظارت می‌شود.[۳] در این فصل به زیرلایهٔ IEEE 802.2 نخواهیم پرداخت زیرا در شبکهٔ اینترنت به خدمات این زیرلایه احتیاجی نیست و کنترل جریان ونظارت بر مبادلهٔ مطمئن داده‌ها به لایهٔ سوم محول شده است.

### ۳-۱) IEEE 802.3 : استاندارد شبکه‌های محلی باس

این استاندارد برای شبکه‌های کانال مشترک با توپولوژی باس تعریف شده است. در این استاندارد ، مدیریت کانال به روش CSMA/CD[۴] انجام می‌شود. یعنی هرگاه

---

[۱] این استانداردها ، لایهٔ اوّل و لایهٔ دوّم از مدل ISO را پیاده‌سازی می‌کند.

[۲] Logic Link Control

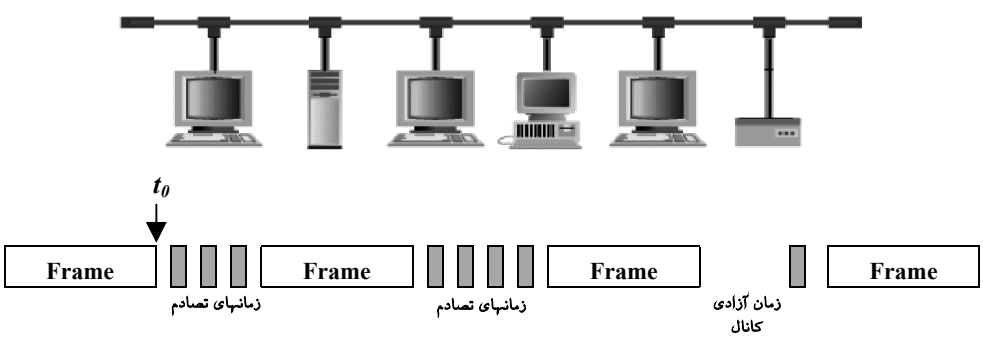
[۳] Flow Control

[۴] Carrier Sense Multiple Access / Collision Detection

ایستگاهی تقاضای ارسال فریم داشته باشد ابتدا به کانال گوش می‌دهد تا تشخیص بدهد آیا روی کانال ، سیگنال معتبر و حامل داده وجود دارد یا آنکه کانال آزاد است. اگر کانال خالی بود ارسال خود را آغاز می‌نماید ولی اگر روی کانال سیگنالی معتبر وجود داشت که نشان می‌داد ایستگاه دیگری در حال ارسال است ، صبر می‌کند و ضمن بررسی مداوم کانال ، مترصد می‌ماند تا کانال آزاد شود. ( یعنی صبر می‌کند تا ایستگاه در حال ارسال ، کار ارسال فریمش را به اتمام برساند.) پس از آزاد شدن کانال ، ایستگاه شروع به ارسال فریم خود روی کانال می‌نماید ، ولی احتمال دارد در این لحظه "تصادم سیگنال" اتفاق بیفتد ، زیرا ممکن است ایستگاههای دیگری نیز برای آزاد شدن کانال منتظر مانده باشند و همگی با آزاد شدن خط اقدام به ارسال فریم خود بنمایند. برای کشف سریع تصادم ، ایستگاهها موظفند در حین ارسال فریم ، به سیگنال خط گوش بدهند تا قادر باشند به محض بروز تصادم ، عمل ارسال فریم را سریعاً متوقف کنند. در این استاندارد کشف تصادم به صورت سخت‌افزاری و آنالوگ انجام می‌شود تا بروز تصادم سریعاً کشف شود. ( روش آنالوگ در کشف تصادم می‌تواند از طریق اندازه‌گیری توان خروجی فرستنده یا اندازه‌گیری طول پالسها پیاده‌سازی شده یا تلفیقی از هر دو باشد.)

هر گاه ایستگاهی که شروع به ارسال می‌کند با تصادم مواجه شود ، موظف است بطور تصادفی یک عدد تولید کرده و بر اساس آن مدت زمانی را صبر کند و مجدداً به خط گوش بدهد. با توجه به آنکه تمام ایستگاههائی که موجب بروز تصادم شده‌اند به اندازهٔ یک زمان تصادفی صبر خواهند کرد ، لذا آن ایستگاهی که زمان تصادفی انتظارش کمتر از بقیه است ممکن است در مرحلهٔ بعد موفق به ارسال شود. ( روال تولید عدد تصادفی و زمان انتظار در ادامه تشریح خواهد شد.)

برای بررسی بیشتر روش CSMA/CD به شکل (۵-۲) دقت کنید.




شکل (۵-۲) بررسی روش CSMA/CD در مدیریت کانال

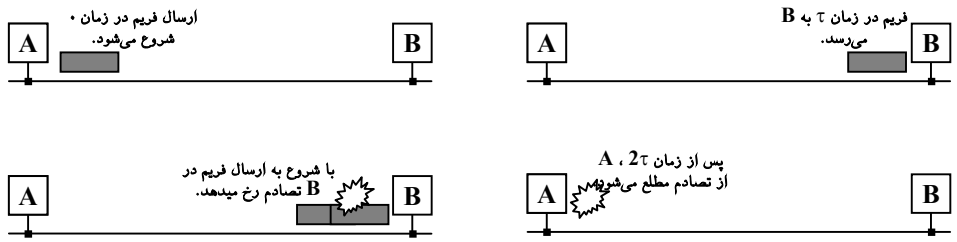
در شکل (۵-۲) یک ایستگاه ارسال فریم خود را در زمان $t_0$ به اتمام رسانده و در این لحظه کانال آزاد شده است. حال موقع ارسال فریم ایستگاههایی است که تقاضا برای ارسال دارند. در این لحظه چند ایستگاه بطور همزمان اقدام به ارسال فریم کرده‌اند و تصادم بوجود آمده است. ایستگاههای تصادم کننده به اندازهٔ یک زمان تصادفی از تلاش برای تصرف کانال کنار می‌کشند و نهایتاً پس از چند مرحله "رقابت"[1] یکی از ایستگاهها موفق به تصرف کانال خواهد شد. زمانهایی که بصورت منقطع بین زمان دو فریم نشان داده شده ، لحظاتی است که ایستگاهها برای تصرف کانال تلاش کرده‌اند ولی تصادم پیش آمده است.

در CSMA/CD هر گاه ایستگاهی از تصادم آگاه شود با تولید سیگنال نویز روی کانال ، به بقیه ایستگاهها کمک می‌کند تا بتوانند به سرعت از بروز تصادم مطلع شوند. سوال مهم آنست که اگر دو ایستگاه دقیقاً در زمان $t_0$ شروع به ارسال نمایند ، چقدر طول می‌کشد تا تصادم کشف شود؟ جواب این سوال از برخی جهات حیاتی است.

مدت زمان کشف تصادم به پارامتر تاخیر انتشار[2] سیگنال بستگی دارد.[3] در یک شبکهٔ باس اگر تاخیر انتشار سیگنال در کل کانال ، $\tau$ ثانیه باشد ، در بدترین حالت به اندازهٔ $2\tau$ **ثانیه** طول می‌کشد تا تصادم کشف شود. به شکل (۶-۲) نگاه کنید. در این شکل فرض شده که ایستگاه A با خالی دیدن کانال شروع به ارسال فریم بنماید. تا رسیدن سیگنال منتشر شده به ایستگاه B در انتهای کانال ، $\tau$ ثانیه طول می‌کشد. اگر در همین لحظه ایستگاه B با خالی دیدن کانال شروع به ارسال فریم خود کند ، تصادم پیش خواهد آمد. با کشف سریع تصادم ، ایستگاه B شروع به تولید نویز می‌کند و $\tau$ ثانیهٔ دیگر طول خواهد کشید تا ایستگاه A از این قضیه مطلع شده و ارسال فریم را قطع کند. این زمان تلف شده در شبکه‌های کامپیوتری بسیار زیاد است. به عنوان مثال اگر طول کانال را هزار متر و نرخ ارسال را 100Mbps در نظر بگیریم ، در زمان $2\tau$ که معادل ده میکروثانیه است ، ایستگاه A ، هزار بیت از فریم خود را ارسال کرده که بدلیل عدم اطلاع از تصادم باید آنرا مجدداً ارسال کند. این زمان جزو زمان تلفاتی کانال محسوب می‌شود. اگر تصادم چند مرحله ادامه یابد ، این زمان تلفاتی به مراتب راندمان کانال را بدتر خواهد کرد.

---

[1] Contention
[2] Propagation delay
[3] **تاخیر انتشار سیگنال در کانالهای مسی ۵ میکروثانیه و در کانالهای نوری یا رادیوئی ۳/۳ میکروثانیه در هر هزارمتر است.**




شکل (٦-٢) مراحل کشف تصادم

در روش CSMA/CD پس از آنکه ایستگاهی بدون تصادم موفق به تصرف کانال شد ، دیگر هیچگاه در خلال ارسال ، تصادم پیش نخواهد آمد و تمام برخوردها بین زمان ارسال دو فریم اتفاق می‌افتد. این زمان که "زمان رقابت" نامیده می‌شود روی راندمان کانال تاثیر منفی دارد. با بالا رفتن تعداد ایستگاهها یا ترافیک آنها ، تعداد تصادمها نیز بصورت تصاعدی زیاد شده و راندمان کانال بصورت بحرانی کاهش خواهد یافت. در ضمن هنگامی که طول کانال زیاد شود ، تاخیر انتشار و به تبع آن زمان رقابت نیز افزایش می‌یابد ، لذا این استاندارد فقط برای شبکه‌های محلی با طول کانال کم و نرخ ارسال پایین مناسب است. بطور کلّی راندمان کانال در این استاندارد به پارامترهای زیر بستگی دارد :

- $F$ : طول فریم بر حسب بیت
- $B$ : پهنای باند کانال
- $C$ : سرعت انتشار
- $L$ : طول کانال
- e : عدد نپرین ( .....2.718 )

$$\text{راندمان کانال} = \frac{1}{1+\frac{2.e.B.L}{C.F}}$$

با دقت در رابطه بالا مشهود است که :

- با کاهش طول فریم راندمان کانال کاهش می‌یابد.
- با افزایش طول کانال راندمان کانال کاهش می‌یابد.
- با افزایش نرخ ارسال راندمان کانال کاهش می‌یابد.

مشخصات فیزیکی استاندارد IEEE 802.3 بطور خلاصه عبارت است از :

- سرعت : ۱۰ مگابیت بر ثانیه
- کدینگ : "منچستر"

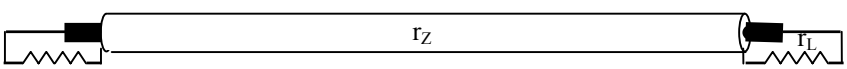
در کدینگ منچستر برای ارسال بیتهای صفر و یک از شکل موجهای زیر استفاده می‌شود :


استفاده از این شکل موجها ، بدلیل لبه‌ای که یقیناً در وسط هر بیت وجود دارد ، به سنکرون شدن ایستگاهها کمک می‌کند.

- سطوح ولتاژ : $\pm 0.85$ V
- کانال : کابل کواکس ۵۰ اهم یا زوج سیم
- حداکثر طول کانال : ۵۰۰ متر با کابل کوآکس ضخیم و ۱۸۵ متر با کابل کوآکس نازک و ۱۰۰ متر با زوج سیم. (برای افزایش طول کابل به "تکرارکننده"[1] نیاز است. با استفاده ازتکرارکننده حداکثر طول کابل تا ۲/۵ کیلومتر قابل افزایش است.[2] )

در شبکه‌های با توپولوژی باس ، اگر انتهای کابل باز باشد سیگنال منتشر شده در درون کابل پس از برخورد به سطح مقطع انتهایی آن ، بازتاب شده و ضمن بازگشت با اختلاف فاز ۱۸۰ درجه ، باعث تداخل با سیگنالهای ارسالی و نهایتاً خرابی بیتها خواهد شد. اگر انتهای کابل با یک مقاومت $\mathbf{r_L}$ اهمی بسته شده باشد ، میزان انعکاس سیگنال در درون کانال طبق رابطهٔ زیر محاسبه می‌شود :

$$\frac{\mathbf{r_Z - r_L}}{\mathbf{r_Z + r_L}}$$




برای آنکه میزان انعکاس سیگنال به صفر برسد باید $r_L=r_Z$ باشد. بهمین دلیل در این استاندارد که کابل کوآکس ۵۰ اهمی بکار رفته است باید یک مقاومت انتهایی ( مقاومت ترمیناتور) ۵۰ اهمی در انتهای کانال بسته شود.

---

[1] Repeater

[2] **تکرار کننده ابزاری است که پس از دریافت بیتها از روی خط ، آنها را مجدداً روی خروجی خود تولید می‌کند و و این عمل باعث تقویت سیگنال و حذف نویز خواهد شد.**

قالب فریمهای داده در استاندارد IEEE 802.3 بصورت زیر است :

| 7 Byte | 1 Byte | 2 or 6 Byte | 2 or 6 Byte | 2 Byte | 0~1500 Byte | 0~46 | 4 Byte |
|---|---|---|---|---|---|---|---|
| **Preamble** | **Start of Frame Delimiter** | **Destination Address** | **Source Address** | **Length of Data Field** | **Data** | **Pad** | **CRC** |

- **Preamble** : ابتدا ایستگاهی که توانسته است بدون تصادم کانال را صاحب شود ، ۷ بایت الگوی 10101010 را روی خط می‌گذارد و چون طریقه ارسال بیتها "منچستر" است ، این ۷ بایت با فرکانس 10MHz بمدت 5.6 میکروثانیه باعث سنکرون شدن تمام گیرنده‌ها با فرستنده خواهد شد.
- پس از این ۷ بایت ، فرستنده **"علامت ابتدای فریم"**[1] را با الگوی 10101011 ، روی خط می‌گذارد. این بایت نقطهٔ شروع فریم را مشخص می‌کند.
- هر فریم دارای دو فیلد آدرس است. طبق استاندارد IEEE 802.3 این آدرسها می‌توانند ۲ بایتی یا ٦ بایتی باشند. (امروزه این آدرسها ٦ بایتی هستند.)

این فیلدها که به نام آدرسهای MAC مشهورند ، عبارتند از :

- آدرس گیرندهٔ فریم (مقصدِ فریم) : همهٔ ایستگاهها به خط گوش می‌دهند و ایستگاهی که آدرس خود را روی خط ببیند فریم را دریافت خواهد کرد.
- آدرس فرستندهٔ فریم (مبدأ فریم)

**تبصره :**

- با ارزشترین بیتِ آدرس برای آدرس شخصی هر ایستگاه 0 است (لزوماً)
- با ارزشترین بیتِ آدرس برای آدرسهای گروهی 1 است. بقیه بیتها شماره گروه را تعیین می‌کند.
- اگر همه بیتهای فیلد آدرس مقصد ۱ باشد ، فریم برای تمامی ایستگاههای شبکه است و باید توسط همهٔ آنها دریافت و پردازش شود. (ارسال فراگیر و همگانی[2])

- **فیلد Lenght Of Data Field** : مقدار این فیلد مشخص می‌کند که چند بایت اطلاعات در فیلد داده وجود دارد.
- **فیلد Data** : در این فیلد حداقل صفر بایت و حداکثر ۱۵۰۰ بایت داده قرار می‌گیرد.

---

[1] Start Delimiter
[2] Boradcasting

- **فیلد PAD** : طبق استاندارد IEEE 802.3 ، فریمهای ارسالی حداقل باید ٦٤ بایت طول داشته باشند. بنابراین اگر اندازهٔ کل فریم ، از ٦٤ بایت کمتر بود باید در قسمت PAD آنقدر صفر اضافه شود تا طول فریم به ٦٤ بایت برسد. دلیل وجود این فیلد آنست که طول فریم نباید آنقدر کم باشد که قبل از زمان $2\tau$ ( بدترین حالتِ زمان کشف تصادم ) ارسال آن به پایان برسد وگرنه ممکن است فرستنده قبل از اطلاع از تصادم ارسال فریم خود را تمام کند. در این استاندارد با در نظر گرفتن حداکثر ۲۵۰۰ متر طول کانال ، زمان $2\tau$ تقریباً معادل ۲۵ میکروثانیه خواهد شد. در این ۲۵ میکروثانیه با سرعت 10 Mbps می‌توان ۲۵۰ بیت (معادل ۳۲ بایت) ارسال کرد که برای اطمینان ، حداقل طول هر فریم ٦٤ بایت انتخاب شده است.
- در انتهای فریم یک کد کشف خطا از نوع CRC-32 جهت بررسی صحت فریم ، اضافه شده است.

به گونه‌ای که اشاره شد در هنگام بروز تصادم هر ایستگاه اقدام به تولید یک عدد تصادفی کرده و بر اساس آن مدت زمانی را صبر می‌کند و مجدداً به خط گوش خواهد داد. روش تولید عدد تصادفی برای انتظار ، از یک الگوریتم خاص تبعیت می‌کند:
در اولین تصادم هر ایستگاه بطور تصادفی یکی از اعداد صفر یا یک را تولید می‌کند. در صورت تولید عدد ۱ ، معادل ۵۱۲ میکروثانیه صبر می‌کند و سپس به خط گوش می‌دهد تا در صورت خالی بودن ارسال را شروع نماید.
در هنگام دومین تصادم متوالی ، ایستگاه بطور تصادفی یکی از اعداد صفر تا سه را تولید می‌کند.
در هنگام سومین تصادم متوالی ، بطور تصادفی یکی از اعداد صفر تا هفت را تولید می‌کند.
در هنگام ***n***امین تصادم متوالی ، بطور تصادفی یکی از اعداد 0 تا $(2^n - 1)$ را تولید می‌کند.

ایستگاه موظف است به ازای عدد تصادفی تولید شده بر مبنای ۵۱۲ میکروثانیه ، منتظر مانده و سپس مجدداً خط را بررسی کند. بعنوان مثال اگر پس از ده تصادم پیاپی ، عدد تصادفی تولید شده ۱۰۰ باشد ، ایستگاه موظف است ۱۰۰×۵۱۲ میکروثانیه (۵۱/۲ میلی ثانیه) منتظر بماند.
پس از ١٦ تصادم پیاپی ، سخت‌افزار شبکه ، یک خرابی جدّی را به کاربر هشدار می‌دهد و پیگیری و کشف علت مسئله برعهده لایه‌های بالاتر است. این روش ، **"الگوریتم عقب‌گرد توانی"**[1] نامیده می‌شود.

[1] Binary Exponential Backoff

شرکت‌های DEC ، Xerox و Intel یک پیاده‌سازی عملی از این استاندارد را که به نام **اترنت**[1] مشهور است ، ارائه کرده‌اند. اترنت کاملاً سازگار با IEEE 802.3 است با این تفاوت که ساختار فریم در اترنت با یک اختلاف جزئی به صورت زیر است :

| 8 Byte | 6 Byte | 6 Byte | 2 Byte | 64~1500 Byte | 4 Byte |
|---|---|---|---|---|---|
| **Preamble** | **Destination Address** | **Source Address** | **Frame Type** | **Data** | **CRC** |

در این فریم ، فیلد Frame Type ، نوع بسته‌ای را که در درون فیلد داده حمل می‌شود ، مشخص می‌نماید.

محصولات مبتنی بر این استاندارد به دلیل ارزان بودن و سادگی نصب ، راه‌اندازی و نگهداری ، برای سالیان سال از پررونق‌ترین سخت‌افزارهای شبکه بوده است.

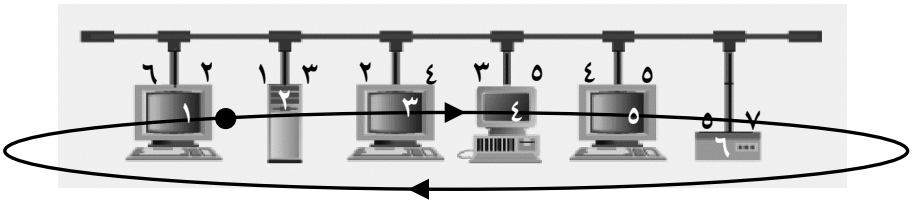
### ۲-۳) IEEE 802.4 : استاندارد شبکه‌های محلی توکن باس

در استاندارد IEEE 802.4 هدف اصلی ، پیاده‌سازی یک حلقهٔ مجازی بر روی یک شبکه با توپولوژی باس است به گونه‌ای که تصادم بر روی کانال بوجود نیاید و همهٔ ایستگاهها طبق یک روش سازمان‌یافته از کانال استفاده کرده و زمان تلف شده‌ای که صرف تصادم می‌شود حذف شود. در این استاندارد زمان انتظار برای استفاده از کانال و ارسال فریم قابل تخمین و تضمین شده می‌باشد ، زیرا اگر n ایستگاه در شبکه موجود و فعال باشد و هر ایستگاه فقط حق استفادهٔ حداکثر T ثانیه از کانال را داشته باشد ، در بالاترین حدّ ترافیک ، تاخیر حداکثر n.T ثانیه خواهد بود.

در این استاندارد کلیه ایستگاهها روی یک حلقهٔ مجازی فرض می‌شوند و نوبت ارسال ایستگاهها به ترتیبی است که روی حلقهٔ مجازی واقع شده‌اند؛ بنابراین هر ایستگاه موظف است: اولا آدرس ایستگاه چپ و راست خود را در حلقه حفظ نماید. ثانیاً هرگاه ایستگاهی ارسال فریم خود را به اتمام برساند ، باید یک فریم کنترلی تحت عنوان **توکن**[2] برای آدرس بعدی خود در حلقه بفرستد. در حقیقت **توکن** مجوز ارسال روی کانال محسوب می‌شود. اگر قاعده بر این باشد که فقط ایستگاهی که توکن را دریافت کرده حق ارسال داشته باشد ، هیچگاه تصادم رخ نمی‌دهد. شکل (۷-۲) یک شبکهٔ باس با حلقهٔ مجازی را نشان می‌دهد.

---

[1] Ethernet
[2] Token




شکل (۷-۲) حلقهٔ مجازی بر روی شبکهٔ باس

استاندارد IEEE 802.4 از لحاظ پیاده‌سازی بسیار پیچیده است و حداقل به ۱۰ زمان‌سنج سخت‌افزاری جهت کنترل و نظارت بر استاندارد محتاج است. برخی از مشخصات این استاندارد عبارت است از :

- نوع کانال : کابل کوآکس ۷۵ اهم تلویزیون
- در این استاندارد سطوح اولویت ۰ ، ۲ ، ٤ و ٦ وجود دارد که اولویت ٦ بالاترین سطح محسوب می‌شود. کلیهٔ ایستگاهها به سطوح اولویتی مجزا ، تقسیم می‌شوند. در سطح اولویت ٦ می‌توان برای انتقال بلادرنگ ، مثل ارسال صدا و تصویر از شبکه بهره برد. وجود اولویت یکی از نکات مثبتی است که شبکهٔ نوع CSMA/CD فاقد آن می‌باشد.
- قالب فریمهای داده در استاندارد IEEE 802.4 بصورت زیر است :

| > 1 Byte | 1 Byte | 1 Byte | 2 or 6 Byte | 2 or 6 Byte | 0~8182 Byte | 4 Byte | 1 Byte |
|---|---|---|---|---|---|---|---|
| **Preamble** | **Start of Frame Delimiter** | **Frame Control** | **Destination Address** | **Source Address** | **Data** | **CRC** | **End Delimiter** |

- **Preamble** : همانند استاندارد 802.3 ، یک الگوی خاص از بیتها جهت سنکرون شدن گیرنده‌ها با فرستنده ، روی خط ارسال می‌شود و تعداد آن می‌تواند از یک تا چند بایت باشد.
- **Start Delimiter** : یک بایت جهت مشخص نمودن ابتدای فریم
- **End Delimiter** : یک بایت جهت مشخص انتهای فریم

کدینگ این دو بایت با کدینگ ارسال اطلاعات متفاوت است بدینگونه که روش عادی ارسال ، روش منچستر است ولی این دو بایت بروش NRZ-Half Binary-Unipolar ارسال می‌شوند تا ابتدا و انتهای فریم ، بصورت آنالوگ و سخت افزاری کشف شود و بهمین دلیل در این استاندارد ، به فیلد مشخص کنندهٔ طول داده ( Data Length ) نیازی نیست.

• **Frame Control** : در استاندارد IEEE 802.4 این فیلد انواع مختلف فریمهای کنترلی را برای عملیات نظارت بر حلقهٔ مجازی ، مشخص می‌نماید. اعداد این فیلد و معنای آن در زیر فهرست شده است:

♦ Claim Token ( 00H ) : هر ایستگاه دارای یک زمان‌سنج (تایمر) است که بطور مرتب خط را بررسی می‌کند ، بنابراین روی خط باید یا توکن یا یک فریم داده ببیند؛ زیرا در این استاندارد بیکار بودن کانال معنا ندارد و اگر ایستگاهها ، فریم داده برای ارسال نداشته باشند ، بطور متوالی فریم توکن را برای یکدیگر ارسال می‌کنند. هر گاه اولین مدت زمان مجاز به سر برسد ولی ایستگاهی که نوبت ارسال اوست خط را بیکار ببیند ، تقاضای دریافت توکن می‌کند؛ یعنی یک فریم که در فیلد Frame Control آن کد 00 قرار دارد ، روی خط قرار می‌دهد تا ایستگاهی که توکن پیش اوست آنرا روی کانال بگذارد. اگر ایستگاهی که توکن را دریافت کرده ، بنحوی از خط خارج شده باشد ، ایستگاهها طبق الگوریتمی نظیر آنچه در استاندارد IEEE 802.3 گفته شد ، جهت رقابت و تصرف توکن مبارزه می‌کنند.

♦ Solicit-Succesor-1 (01H) : هر ایستگاه زمانسنجی دارد که بطور متناوب ایستگاههایی را که در حلقه نیستند و تقاضای ورود به شبکه را دارند ، دعوت می‌کند. طریقه دعوت آنها با ارسال فریمی است که در فیلد Frame Control آن ، کد 01 قرار دارد. ایستگاه ارسال کنندهٔ این فریم شماره خود و شماره‌هایی را که می‌توانند بعد از او وارد حلقه شوند ، معین می‌کند. در چنین زمانی ممکن است بین چند ایستگاه متقاضی ورود به شبکه تصادم رخ دهد ، بنابراین ایستگاه با قرار دادن فریم کنترلی 04H یا Resolve-Contention به رفع تصادم اقدام می‌نماید.

♦ گاهی اوقات توکن گم می‌شود مثلا ایستگاه شمارهٔ x ، توکن را برای آدرسی ارسال کرده است که اصلا روی حلقه وجود ندارد. در این حالت ایستگاهی که توکن را آزاد کند وجود ندارد؛ برای رفع این مشکل ، ایستگاه ارسال کنندهٔ توکن با تنظیم یک زمان‌سنج و بررسی خط ، پس از انقضای مهلتِ پاسخ دهی ، با ارسال فریم 03 یا Who-Follows ، آدرس ایستگاه بعد از خود را در حلقه سوال می‌کند. حال ایستگاهی که آدرس اعلام شده توسط این فریم را بعنوان آدرس ایستگاه قبل از خود می‌بیند با ارسال یک فریم کنترلی به نام Set-Successor به آن پاسخ می‌دهد و ایستگاه مربوط را از حلقه حذف کرده و حلقه اصلاح می‌شود. یعنی فرستندهٔ فریم Set-Successor جانشین ایستگاه خراب می‌شود.

♦ اگر ایستگاهی خراب یا از خط خارج شود و این اتفاق برای ایستگاه بعد از او هم بیفتد ، دیگر هیچ ایستگاهی به فریم Who-Follows پاسخ نمی‌دهد؛ در چنین حالتی با

ارسال فریم Solicit-Successor-2 همهٔ ایستگاه‌ها دعوت به بازسازی مجدد شماره‌های حلقه مجازی می‌شوند.

- 08H یا Token : فریمی است که هر ایستگاه با دریافت آن حق ارسال اطلاعات دارد.

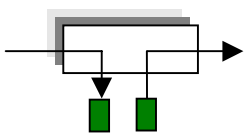
- بقیهٔ فیلدهایی که به آن اشاره نشد دقیقاً عملکردی شبیه به آنچه در استاندارد IEEE 802.3 توصیف شد ، دارند.

استاندارد IEEE 802.4 برای استفاده در محیطهای صنعتی و کاربردهای بلادرنگ ارائه شد و امروزه اهمیت خود را از دست داده است.[۱]

### ۳-۳) IEEE 802.5 : استاندارد شبکه‌های محلی حلقه

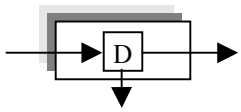
این استاندارد مختص توپولوژی حلقه است و اولین بار توسط IBM پیاده‌سازی شده است. در این استاندارد هر ایستگاه موظف است فریمهای داده را از ایستگاه قبلی دریافت و به ایستگاه بعدی در حلقه ارسال کند. ایستگاه حتی اگر اطلاعات برای او ارسال شده باشد ، موظف است ضمن برداشتن اطلاعات از روی خطّ مجدداً آنرا به ایستگاه بعدی ارسال کند. بنابراین اگر یک ایستگاه فریمی را روی کانال بگذارد ، نهایتاً خودش فریم خود را دریافت می‌کند. ایستگاههایی که اطلاعات برای آنها ارسال نشده است فقط وظیفه تقویت و انتقال آن را روی کانال برعهده دارند. هر ایستگاه در انتقال یک فریم حداقل یک بیت تاخیر ایجاد می‌کند. در شبکهٔ حلقه ، هر ایستگاه می‌تواند یکی از سه حالت زیر را داشته باشد:

- **حالت ارسال** : ایستگاه فقط زمانی مجوز ارسال دارد که یک فریم کنترلی خاصّ ۲٤ بیتی که توکن (نشانه) نام دارد ، دریافت کند. هرگاه یک ایستگاه توکن را دریافت کند ، اگر فریمی را برای ارسال آماده داشته باشد ، می‌تواند آنرا ارسال کرده و سپس توکن را آزاد کند. اگر ایستگاه فریمی جهت ارسال نداشته باشد ، توکن دریافتی را به ایستگاه بعدی در حلقه می‌فرستد. فرستنده پس از ارسال فریم خود و انتقال کامل آن در طول حلقه خودش آنرا دریافت کرده و از حلقه خارج می‌کند. ایستگاهِ در حال ارسال ، پس از دریافت آخرین بیتِ فریمِ خود ، توکن را آزاد می‌کند.

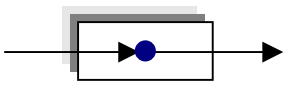


[۱] اولین بار این شبکه برای کاربردهای صنعتی در کمپانی جنرال موتورز (GMS) آمریکا پیاده‌سازی شد.

- **حالت شنود** : در این حالت ایستگاه فریم دریافتی را با یک بیت تاخیر برای ایستگاه بعدی در حلقه ارسال می‌کند.




- **حالت غیر فعال**[۱] : در این حالت ایستگاه از شبکه خارج شده و عملاً ورودی و خروجی ، اتصال کوتاهند.



در بار سبک (یعنی وقتی تقاضا برای ارسال کم است ) ، توکن بسرعت و به ازای هر ایستگاه فقط با یک بیت تاخیر می‌چرخد و بنابراین هرگاه یک ایستگاه تقاضای ارسال داشته باشد ، پس از چند بیت تاخیر قادر به دریافت توکن و ارسال فریم خود خواهد بود.

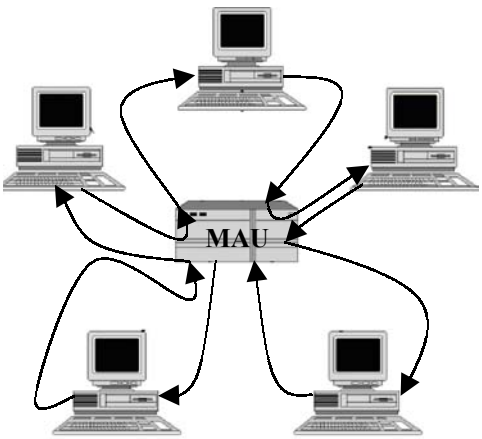
ایستگاهی که توکن را دریافت می‌کند ، مجاز است آنرا تا زمان محدودی ، برای ارسال فریمش نگه دارد. در استاندارد IEEE 802.5 ، "زمان مجاز در اختیار داشتن توکن"[۲] ، بطور پیش فرض ده میلی‌ثانیه در نظر گرفته شده است. چون هر ایستگاه موظف به ارسال فریم در زمان مجاز می‌باشد و پس از این زمان باید توکن را به ایستگاه بعدی تحویل بدهد لذا در بار سنگین همه در یک صف منظم سرویس‌دهی شده و از کانال استفادهٔ بهینه می‌شود. بنابراین می‌توان ادعا کرد در استاندارد IEEE 802.5 ، راندمان کانال در بار سنگین ، با تقریب خوبی نزدیک به ٪۱۰۰ است.

در شبکه‌های حلقه اگر یکی از ایستگاهها خراب شود کل شبکه مختل خواهد شد؛ بهمین دلیل باید تدبیری اتخاذ گردد تا یک ایستگاه به محض خرابی ، از حلقه خارج شود. در شبکهٔ حلقه ، از یک ابزار به نام MAU[۳] استفاده می‌شود تا به محض خروج یک ایستگاه از شبکه ، حلقه اصلاح شود. بزبان ساده تمام کابلهای شبکه از طریق MAU به هم وصل می‌شوند تا هنگام خرابی یک ایستگاه ، ورودی و خروجی آن ایستگاه را اتصال کوتاه کند و ایستگاه از حلقه خارج شود. شکل (۸-۲) یک شبکهٔ حلقه را با MAU ، نشان می‌دهد. هرچند در شبکهٔ

---

[۱] Idle
[۲] Token Holding Time
[۳] Multi Access Unit

حلقه وجود MAU ضروری نیست ولی شبکه را قابل اعتماد می‌کند. وجود یک MAU مرکـزی برای شبکه‌های با محدوده جغرافیایی وسیع مشکل‌آفرین است و در بعضی از مواقع می‌توان از چند MAU مجزا ، بصورت منطقه‌ای استفاده کرد.




شکل (۸-۲) شبکهٔ حلقه با MAU

در استاندارد IEEE 802.5 باید یک ایستگاه بعنوان ناظر[1] وظائف زیر را برعهده بگیرد: (هـر ایستگاه این قابلیت را دارد که نقش ناظر را بازی کند و نیازی به ایستگاه مجزا نیست.)

- **بررسی وجود توکن** : هر ایستگاه حداکثر ۱۰ میلی ثانیه می‌تواند توکن را در اختیار داشته باشد و ایستگاه ناظر باید با تنظیم یک زمان‌سنج ، از چرخش توکن مطمئن شود.
- **از بین بردن و پاک کردن فریمهای سرگردان و آشغال** ( فریمهای سـرگـردان ، فریمـهای کوتاهی هستند که فرستندهٔ آنها ، پس از ارسال از حلقه خـارج شـده و آنـها روی حلقـه ، سرگردان می‌چرخند.)
- **ایجاد و تضمین حداقل ۲٤ بیت تاخیر در حلقه** : حداقل تأخیر در شبکهٔ حلقه مبتنـی بـر استاندارد IEEE 802.5 ، باید بقدری باشـد کـه توکـن بتوانـد یـک دور کـامل بزنـد؛ یعنـی ایستگاهی که توکن را ایجاد می‌کند نباید در حین تولیـد آنـرا دریـافت کنـد. پـس در ایـن استاندارد که توکن ، ۲٤ بیتی است ، حداقل باید ۲٤ بیت تاخیر وجود داشته باشـد و ایـن تاخیر برای عملکرد درست شبکه ضروری است. مساله بسـیار مـهم در طراحـی و تحلیـل

[1] Monitor

شبکهٔ حلقه آنست که طول فیزیکی یک بیت[۱] چقدر است؟ در یک کانال به طول L متر ، هرگاه یک پالس در ابتدای آن قرار گیرد ، مدت τ ثانیه طول خواهد کشید تا به انتهای کانال برسد. در شبکهٔ حلقه ، با نرخ ارسال " R مگابیت بر ثانیه" (روی کانال مسی با سرعت انتشار 200 m/μs ) طول فیزیکی یک بیت 200/R تعریف می‌شود. (R برحسب مگابیت بر ثانیه است.) بعنوان مثال برای شبکه ای با طول ۱۰۰۰ متر و نرخ ۱ مگابیت بر ثانیه ، طول فیزیکی بیت برابر با ۲۰۰ متر است؛ یعنی تا زمانی که بیت بعدی روی کانال شود ، بیت اول ۲۰۰ متر طی از کانال را طی خواهد کرد. در نتیجه تا زمانی که بیت اول به انتهای کانال برسد ، فرستنده ۵ (۱۰۰۰/۲۰۰) بیت بعدی فریم را ارسال کرده است.
نسبت طول کانال به طول فیزیکی بیت ، "تاخیر انتشار کانال بر حسب بیت" خواهد بود.
در شبکهٔ حلقه با استاندارد IEEE 802.5 ، مجموع تعداد ایستگاههای فعال (که بصورت پیش‌فرض یک بیت تاخیر ایجاد می‌کنند) و تاخیر انتشار کانال بر حسب بیت ، نباید از ۲٤ کمتر شود؛ در غیر اینصورت ایستگاه ناظر موظف است تاخیر مصنوعی ایجاد کند.

روش کدینگ در این استاندارد ، منچستر تفاضلی[۲] با سطوح 3± ولت تا 4.5 ± ولت می‌باشد و نرخ ارسال ۱ تا ٤ مگابیت بر ثانیه است.

قالب فریمهای داده در استاندارد IEEE 802.5 بصورت زیر است. بدلیل مفصل بودن عملکرد برخی از فیلدها ، فقط به معرفی آنها بسنده کرده‌ایم.

| 1 Byte | 1 Byte | 1 Byte | 2 or 6 Byte | 2 or 6 Byte | No Limit | 4 Byte | 1 Byte | 1 Byte |
|---|---|---|---|---|---|---|---|---|
| **Start of Frame Delimiter** | **Access Control** | **Frame Control** | **Destination Address** | **Source Address** | **Data** | **CRC** | **End Delimiter** | **Frame Status** |

- **Start Delimiter** : یک بایت جهت مشخص نمودن ابتدای فریم
- **End Delimiter** : یک بایت جهت مشخص کردن انتهای فریم

کدینگ این دو بایت با کدینگ ارسال اطلاعات متفاوت است تا ابتدا و انتهای فریم ، بصورت آنالوگ و سخت افزاری کشف شود.

- **Access Control** : این فیلد هشت بیتی دارای چهار قسمت است:

---

[۱] Physical Length
[۲] Differential Manchester

- بیت توکن : این بیت مشخص کنندهٔ آن است که فریم جاری ، یک فریم داده نیست بلکه فریم توکن است و به ایستگاه مجوز ارسال می‌دهد. اگر این بیت فعال (۱) باشد ، ساختار فریم به صورت زیر خواهد بود:


- بیت مانیتور : این بیت را ایستگاه ناظر ، ۱ می‌کند تا یک فریم داده ، بیش از یکبار در حلقه نچرخد.
- بیتهای رزرو توکن و بیتهای اولویت : در این استاندارد سطوح اولویت وجود دارد و کلیهٔ ایستگاهها ، فریمهای ارسالی خود را به سطوح اولویتی مجزا تقسیم می‌کنند. توکن ابتدا با سطح اولویت بالا می‌چرخد و ایستگاههایی که یک فریم با اولویتِ مطابق با اولویت تعیین شده در توکن آمادهٔ ارسال دارند ، حق دارند آنرا ارسال نمایند. در غیر این صورت در بیتهای رزرو توکن ، اولویت فریم خود را رزرو می‌کند.

- **Access Control** : این فیلد هشت بیتی ، انواع مختلف فریم را برای نظارت بر عملکرد صحیح حلقه ، تعریف می‌کند. در جدول زیر برخی از انواع این فریمها و کاربرد آنها ، به اختصار فهرست شده است:

| شماره | نام فریم | عملکرد کنترلی فریم |
|---|---|---|
| 00000000 | Duplicate Address Test | زمانی که دو ایستگاه شمارهٔ یکسان داشته باشند ، از این فریم استفاده می‌شود. |
| 00000010 | Beacon | از این فریم برای کشف محل پارگی کانال ، استفاده می‌شود. |
| 00000011 | Claim Token | با این فریم یک ایستگاه تلاش می‌کند تا خود را نامزد "ایستگاه ناظر" نموده و بر شبکه نظارت کند. |
| 00000100 | Purge | با این فریم از تمام ایستگاهها تقاضا می‌شود تا حلقه را از نو بازسازی کنند. |
| 00000101 | Active Monitor Present | ایستگاه ناظر به طور متناوب این فریم را ارسال می‌کند تا بقیه را از حضور خود مطمئن کند. |
| 00000110 | Standby Monitor Present | در این استاندارد برای ایستگاه ناظر یک ایستگاه پشتیبان در نظر گرفته می‌شود تا در مواقع اضطراری جایگزین ایستگاه ناظر شود. ایستگاه پشتیبان به طور متناوب این فریم را ارسال می‌کند تا بقیه را از حضور خود مطمئن کند. |

• **Frame Status** : در این فیلد که در آخر فریم قرار گرفته ، فقـط دو بیـت بـا ارزش A و C تعریف شده است. این دو بیت که در ابتدا صفر است باید توسط ایستگاه مقصـدِ فریـم تنظیـم شود. پس از چرخش فریم روی حلقه و بـازگشـت بـه ایسـتگاه تولیـد کننـدهٔ آن ، از حـالات مختلف این دو بیت می‌توان اطلاعات زیر را استنتاج کرد:

| | |
|---|---|
| ایستگاه مقصد ، در شبکه نیست و فریم دریافت نشده است. | **A=0 , C=0** |
| ایستگاه مقصد ، در شبکه وجود دارد ولی فریم پذیرفته نشد. | **A=1 , C=0** |
| فریم توسط ایستگاه مقصد ، سالم دریافت شده است. | **A=1 , C=1** |

شبکه‌های مبتنی بر استاندارد IEEE 802.5 ، بدلیل راندمان بسیار خوب کانال ، مـورد توجـه قرار گرفتند ، به گونه‌ای که IBM آنرا به عنوان شبکهٔ داخلـی خـود بـرگزیـد و بـر اسـاس آن شبکه‌های بین شهری با سرعت بالا (مثل شبکهٔ بین‌شهری FDDI ) پیاده‌سازی شدند.

### ۴-۳) مقایسهٔ سه استاندارد معرفی شده برای شبکه‌های محلی

برای تعیین یک طرح اولیه برای نصب و پیاده‌سازی شبکه ، باید شرایط حـاکم و همچنیـن الزاماتی که در محیط وجود دارد ، در نظر گرفته شود و بالطبع یک استاندارد هیچ رجحانی بـر دیگری ندارد مگر در قیاس با شرایطِ حاکم بر محیطی که قرار است در آنجا شبکه نصب شود. در زیر مشخصات کلیدی استانداردهای معرفی شده را فهرست می‌کنیم:

**IEEE 802.3 - CSMA/CD**

- دسترسی به کانال قطعیت و روال منظم ندارد.[1]
- در بار پایین ، تاخیر چندانی وجود ندارد و راندمان کانال مناسب است.
- در بار بالا به دلیل افزایش تصادم ، این استاندارد راندمان خوبی ندارد.
- در سرعت بالا و کاهش طول فریم ، راندمان کانال کاهش می‌یابد.
- فریمها سطوح اولویت ندارند و ارسال صوت و تصویر در آن گنجانده نشده است.

[1] Nondeterministic Channel Allocation

- با تمام کاستیها ، هزینهٔ نصب و راه‌اندازی این نوع شبکه کم است.

**IEEE 802.4 – Token Bus**

- دسترسی به کانال دارای روال باقطعیت و منظم‌تری نسبت به استاندارد قبلی است.
- این استاندارد برای فریمها اولویت قائل است و میتوان در اولویت بالا ارسال همزمان و بلادرنگ صوت و تصویر را ارائه کرد .
- استاندارد بسیار پیچیده است و قسمتی از سخت افزار آنالوگ می‌باشد .
- استفاده از کانال در بار بالا صحیحتر و با راندمان بهتری صورت می‌گیرد .
- برای فریمهای با طول کوتاه راندمان پائین می‌آید .
- برای سیستمهای بلادرنگ[1] قابل استفاده می‌باشد .

**IEEE 802.5 – Token Ring**

- سخت افزار کاملاً دیجیتال است و تصادم معنا ندارد.
- کابلهای زوج سیم تا فیبر نوری قابل استفاده می‌باشد.
- این استاندارد برای فریمها اولویت قائل است و می‌توان در اولویت بالا ارسال همزمان و بلادرنگ صوت و تصویر را ارائه کرد.
- فریمهای کوتاه قابل ارسالند بدون آنکه راندمان کانال بصورت بحرانی کم شود.
- راندمان در بار بالا بسیار عالی است. ( نزدیک ۱۰۰٪ )
- وجود ایستگاه ناظر درسیستم حساسیت حلقه را به آن ایستگاه افزایش می‌دهد. عملکرد بدِ[2] ایستگاه ناظر روی کل شبکه تاثیر می‌گذارد.
- در بار پایین مقداری تاخیر وجود خواهد داشت. ( حداقل معادل زمان ۲٤ بیت )

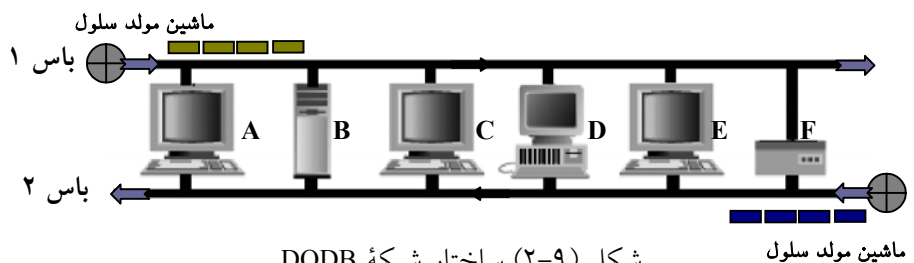
---

[1] Real Time Systems
[2] Malfunctioning

### ۴) IEEE 802.6 - DQDB : استاندارد شبکهٔ بین‌شهری

هیچیک از استانداردهای IEEE که در بخشهای قبل معرفی شدند ، کارآیی لازم را برای بکارگیری در شبکه‌های بین‌شهری نداشتند. مسئلهٔ زیاد بودن طول کانال و تعداد بسیار زیاد ایستگاهها در شبکهٔ MAN ، تاثیر مخربی بر روی راندمان کانال دارد و باید برای چنین شبکه‌هایی ، استاندارد ویژه‌ای طراحی می‌شد.

با توجه به آنکه بهترین کانال انتقال برای شبکهٔ بین‌شهری فیبر نوری است ، لذا IEEE استانداردی به نام DQDB[1] که مبتنی بر دو رشته فیبر نوری است ، ارائه کرد. شبکهٔ مبتنی بر این استاندارد قادر است ناحیه‌ای به وسعت ۱۶۰ کیلومتر را با نرخ ارسال 44.736Mbps پوشش بدهد. در شکل (۹-۲) ساختار این شبکه به تصویر کشیده شده است. به گونه‌ای که دیده می‌شود دو رشتهٔ فیبر نوری که به هر کدام "باس" گفته می‌شود با طول بسیار زیاد ، ارتباط بین ایستگاهها را برقرار می‌کند. مسیر و جهت ارسال اطلاعات در هریک از این باسها یکطرفه است. در انتهای هر یک از باسها یک ماشین ویژه وجود دارد که بطور دائم ، سلولهای مشخص و ثابت ۵۳ بایتی تولید می‌کند. این ماشینهای تولید سلول ، در دو سمت مخالف به انتهای رشتهٔ فیبر متصل می‌شوند و برای هر فیبر فقط یک ماشین وجود دارد. تولید سلولها توسط یک ماشین خاص ، باعث می‌شود که علیرغم طول بسیار زیاد کانال و تاخیر انتشار ، ایستگاهها قادر باشند خود را با این سلولها سنکرون کرده و از لحاظ روال دسترسی به کانال مشکلی ایجاد نشود. سلولها در هنگام تولید ، خالی هستند و قادرند ٤٤ بایت داده را حمل نمایند. هر ایستگاه ضمن دریافت بیتهای سلول ، "بدون هیچ تاخیری" آنرا روی قطعهٔ بعدی فیبر تقویت و ارسال می‌نماید.




شکل (۹-۲) ساختار شبکهٔ DQDB

[1] Distributsed Queue Dual Bus

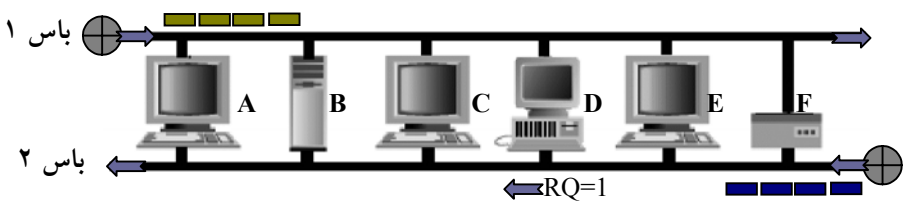
در هر سلول ، ۹ بایت ابتدایی سرآیند سلول هستند. در سرآیند هر سلول دو بیت کنترلی ویژه تعریف شده است که نام و عملکردشان به شرح زیر است:

- بیت اشغال **-Busy–** : در صورتی که این بیت ۱ باشد ، مشخص کنندهٔ آنست که سلول خالی نیست و محتوی داده می‌باشد. بدین معنا که سلول توسط ایستگاه دیگری که به ماشین تولید سلول نزدیکتر بوده پر شده است.
- **بیت تقاضا -Request–** : هرگاه ایستگاهی تقاضای ارسال داشته باشد ، با ۱ کردن این بیت ، تقاضای خود را اعلام می‌کند.

هر ایستگاه که تقاضای ارسال یک سلول دارد ، باید تشخیص بدهد که گیرندهٔ سلول (ایستگاه مقصد) در سمت چپ او واقع شده یا در سمت راست او قرار دارد؛ چراکه مسیر ارسال یکطرفه است و برای ارسال سلول به سمت چپ باید از یک باس و برای سمت راست از باس دیگر استفاده کرد. در شکل (۹–۲) اگر ایستگاه B برای D ارسال داشته باشد باید از باس ۱ استفاده کند در حالیکه برای ارسال از B به A باید از باس ۲ استفاده شود.

در این استاندارد روش ارسال یک سلول بر روی کانال ، زمانبندی FIFO[1] است و هر ایستگاه موظف است خودش را طبق الگوریتم زیر نوبت‌بندی کند:

- هر ایستگاه دارای دو عدد شمارندهٔ سخت‌افزاری است که در ابتدای کار صفر است. این دو شمارنده ، RC[2] و CD نام دارند. شمارندهٔ RC بر اساس ۱ بودن "بیت تقاضا" در سرآیند سلول افزایش می‌یابد یعنی هرگاه یک ایستگاه سلولی را انتقال بدهد که "بیت تقاضا" در آن یک باشد یک واحد به RC اضافه می‌کند ، بدین معنی که یک ایستگاه "بالادست"[3] تمایل دارد روی باس مخالف ، ارسال داشته باشد. به شکل (۱۰–۲) نگاه کنید. ایستگاه D با ۱ کردن بیت تقاضا در یک سلول که روی باس ۲ جریان دارد ، به A ، B و C اعلام می‌کند که تمایل دارد روی باس ۱ ، سلولی را ارسال کند. (مثلاً برای E یا F )




شکل (۱۰–۲) اعلام تقاضا به ایستگاه‌های بالادست

---

[1] First In First Out
[2] Request Counter
[3] Upstream

این کار برای آنست که ایستگاههای بالادست که به ماشین تولید سلول نزدیکترند ، سلولهای خالی را قبضه نکنند و به ماشینهای پایین‌دست نیز اجازه بدهند تا ارسال داشته باشند. دقت کنید که هر ایستگاه فقط می‌تواند تقاضای ارسال یک سلول را داشته باشد. وقتی شمارندهٔ RC در یک ایستگاه مقدار غیر صفر دارد ، به معنای آنست که ایستگاههایی تقاضای ارسال دارند. با ارسال هر سلول یک واحد از RC کم می‌شود چراکه قطعاً یکی از متقاضیان آنرا پر کرده و از صف خارج شده است.

ایستگاهی که تقاضای ارسال ندارد ، با کم و زیاد کردن شمارندهٔ RC جریان سلولها و تقاضاهای ارسال را تعقیب می‌کند.

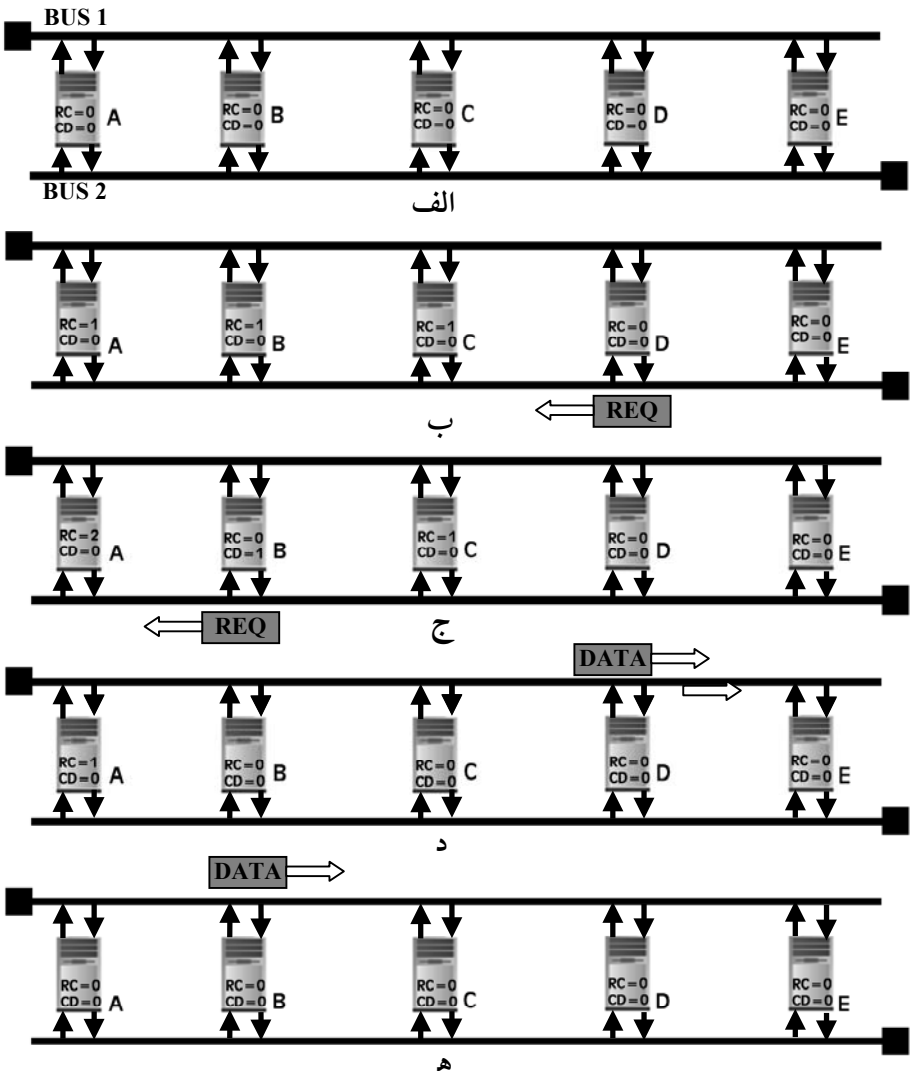
- به محض آنکه ایستگاهی تقاضای ارسال روی یکی از باسها داشته باشد و سلولی را روی باس مخالف دریافت کند که بیت تقاضا در آن صفر باشد ، آنرا ۱ کرده و با اعلام آن ، شمارندهٔ RC را در شمارندهٔ CD کپی کرده و RC را صفر می‌کند. این کار برای آنست که ایستگاه بداند در زمانی که او تقاضای ارسال را اعلام کرده ، چند ایستگاه قبل از او تقاضا داده‌اند و حق دارند زودتر ارسال کنند. پس از صفر شدن RC ، مقدار جدیدِ RC تقاضاهائی است که بعد از تقاضای ایستگاه ، اعلام شده است. با ارسال هر سلول روی باس یک واحد از CD کم می‌شود تا به صفر برسد. به محض صفر شدن CD ، نوبت ارسال ایستگاه فرا می‌رسد و با دریافت سلول خالی آنرا پر کرده و ارسال می‌نماید. به عنوان مثال اگر در لحظه‌ای مقدار CD=4 و RC=3 باشد به این معناست که در لحظهٔ اعلام تقاضا ٤ ایستگاه قبل از او تقاضای ارسال داده‌اند و ۳ ایستگاه نیز پس از او در صف هستند؛ بنابراین موظف است به اندازهٔ زمان ٤ سلول صبر کند تا نوبتش فرا برسد.

دقت کنید که ارسال روی یک باس فقط به مقصد ایستگاههای "پایین‌دست"[۱] مقدور است چرا که جریان سلولها بصورت فیزیکی یکطرفه است. چون تقاضا باید به اطلاع ایستگاههای بالادست برسد تا ایستگاه را در صف وارد کنند ، لذا باید روی باس مخالف اعلام شود.

مراحل فوق را با مثال شکل (۱۱-۲) بررسی می‌کنیم. در شکل (الف) همهٔ ایستگاه‌ها آزاد هستند و چون هیچیک از آنها تقاضای ارسال ندارند ، لذا تمام شمارنده‌ها صفر است. در شکل (ب) ایستگاه D روی باس مخالف به ایستگاههای بالادست خود ، اعلام تقاضا کرده است. ایستگاههای A ، B و C با دیدن این تقاضا ، شمارندهٔ RC خود را یک واحد افزایش داده و متوجه می‌شوند که ایستگاهی قبل از آنها تقاضا داده است. در شکل (ج) فرض شده که B نیز تقاضای ارسال داده و مقدار RC در ایستگاه A باز هم افزایش یافته است.

---

[۱] Downstream




شکل (۱۱-۲) نوبت‌بندی FIFO در استاندارد DQDB

در شکل (د) یک سلول خالی تولید شده و چون D زودتر تقاضا داده و شمارندهٔ CD صفر است ، بنابراین مجاز به ارسال می‌باشد. پس از ارسال تمام شمارنده‌های غیر صفر یک واحد کاهش می‌یابد. در شکل (ه) یک سلول خالی دیگر تولید شده و B مجاز به ارسال می‌باشد.

در این روش هیچ ایستگاهی قبل از آنکه شمارندهٔ CD صفر نشود ، حق ارسال ندارد و با این روال یک "صف توزیع شده"[۱] پیاده‌سازی می‌شود.

در استاندارد IEEE 802.6-DQDB روال تولید سلولها بسیار سریع است و در فواصل ۲۳ نانوثانیه‌ای یک سلول جدید تولید خواهد شد و چون تصادم در این روش معنا ندارد ، ایستگاهها با تاخیر نامعقول مواجه نخواهند شد. تنها ایرادی که می‌توان بر این استاندارد وارد کرد سرآیند ۹ بایتی هر سلول است که تقریباً ۱۷ درصد از پهنای باند مفید کانال را هدر می‌دهد.

شبکه‌های مبتنی بر این استاندارد در کشورهای زیادی مورد استقبال قرار گرفت و به عنوان شبکه بین‌شهری نصب و پیاده‌سازی شد.

### ۵) IEEE 802.11 – Wireless Lan : استاندارد شبکه‌های بی‌سیم

با پیشرفت تکنولوژی و همه‌گیر شدن کامپیوترهای شخصی و ظهور کامپیوترهای قابل حمل و نقل ، نیاز به یک ارتباط بی‌سیم احساس می‌شد؛ بهمین دلیل در دههٔ نود تحقیقاتی در این زمینه انجام شد و شبکه‌های محلی بی‌سیم با پیکربندی[۲] زیر توسعه یافتند:

- ایستگاههای متحرک (همانند کامپیوترهای کیفی[۳]) باید بتوانند در بُرد محدود ( در حدّ چند ده متر ) روی باند UHF ، داده‌ها را انتقال بدهند.
- در محدودهٔ پیاده‌سازی چنین شبکه‌ای ، باید تعدادی ایستگاه ثابت[۴] وجود داشته باشد. (ارتباط آنها نیز با ایستگاههای متحرک بی‌سیم است.)
- پهنای باند کانال بین یک تا دومگابیت بر ثانیه ، مطلوب خواهد بود.
- ایستگاههای متحرک دارای توان انتقال ثابت و محدودی هستند. ( یعنی بُرد سیگنال تمام ایستگاهها یکسان است )

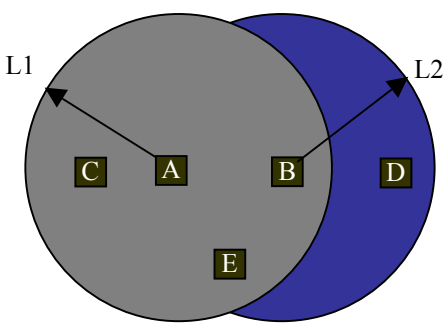
---

۱ Distributed Queue
۲ Configuration
۳ Notebook
۴ Base Station

- به دلیل پراکندگی تصادفی ایستگاهها ، فقط تعداد محدودی از ایستگاههای متحرک در محدودهٔ برد یکدیگر هستند.

هیچیک از استانداردهای IEEE مثل CSMA/CD یا Token Ring برای مدیریت کانال در چنین شبکه‌ای جوابگو نیست ، زیرا:
اولاً اکثر ایستگاهها متحرکند و مبادلهٔ توکن بین ایستگاههایی که زمانی در شبکه هستند و زمانی از شبکه خارج می‌شوند ، معقول نخواهد بود.
ثانیاً پراکندگی تصادفی و برد محدود ایستگاهها باعث می‌شود که نتوان از روشهای مبتنی بر گوش دادن به کانال استفاده کرد ، چراکه بسیاری از ایستگاهها بدلیل بعد مسافت ، سیگنال یکدیگر را نمی‌شنوند.

در استاندارد IEEE 802.11 ایستگاهها قبل از ارسال روی کانال ، باید عملیات "دست‌تکانی"[1] انجام بدهند. عملیات دست‌تکانی به منظور اطلاع دادن به ایستگاههای متحرکی است که در محدودهٔ برد یکدیگر هستند و می‌توانند عامل بروز تصادم باشند. مراحل عملیات دست‌تکانی با یک مثال تشریح می‌شود. به شکل (۱۲-۲) دقت کنید. در این مثال پراکندگی اتفاقی ایستگاهها نشان داده شده است. دایره‌های L1 و L2 ، محدودهٔ برد سیگنال منتشره از ایستگاههای A و B را مشخص کرده‌اند. در این مثال ایستگاه C بدلیل بُعد مسافت در محدودهٔ سیگنال ایستگاه B نیست؛ D هم در محدودهٔ ایستگاه A نمی‌باشد. حال فرض کنید ایستگاه A تمایل دارد برای ایستگاه B فریمی را ارسال کند. عملیات دست‌تکانی به شرح زیر است:




شکل (۱۲-۲) پراکندگی اتفاقی ایستگاهها در شبکهٔ بی‌سیم

[1] Handshaking

- قبل از ارسال ، A موظف است فریمی کوتاه (۳۰ بایتی ) به نام RTS[1] را در محدودهٔ برد سیگنال خود ارسال نماید. درون این فریم ۳۰ بایتی ، آدرس گیرنده ، آدرس فرستنده و طول فریمی که قرار است ارسال شود ، مشخص می‌شود.

- در صورتی که B آماده برای دریافت باشد در پاسخ ، فریم کوتاه CTS[2] را منتشر می‌کند. هنگامی که A این فریم را دریافت کند ، اجازه دارد فریمش را ارسال نماید.

- اصل بر این است که هر ایستگاهی که سیگنال RTS را احساس می‌کند ( یعنی قادر است سیگنال منتشره از A را بشنود ) ، قاعدتاً به A نزدیک است و باید به مدت کافی صبر کند تا CTS بدون تصادم به A برگردد. ( این زمان پس از اتمام ارسال فریم RTS ، تقریباً به اندازهٔ زمان لازم برای ارسال ۳۰ بایت است.)

- هر ایستگاهی که CTS را می‌شنود به B نزدیک است و باید به اندازهٔ مدت انتقال فریم داده صبر کند تا انتقال فریم تمام شود. ( طول فریم در RTS و CTS به همهٔ ایستگاهها اعلام می‌شود و همه می‌توانند تخمینی از زمان انتقال فریم داشته باشند. )

در شکل (۱۲-۲) ایستگاه A فریمی را برای B آمادهٔ ارسال دارد ، بنابراین فریم RTS را ارسال می‌کند. چون C در حوزهٔ شنوایی سیگنال A قرار دارد ، RTS را می‌شنود ولی چون در حوزهٔ شنوایی B قرار ندارد ، CTS را نمی‌شنود و پس از تاخیری معادل ۳۰ بایت آزاد است برای ایستگاههای دیگر ارسال داشته باشد ، زیرا ارسال C به دلیل دور بودن از B منجر به تصادم نخواهد شد. ایستگاه D هر چند در حوزهٔ شنوایی A نیست و RTS را نمی‌شنود ولی قادر است CTS را بشنود. بنابراین حق ندارد تا ارسال کامل فریم توسط A ، ارسال داشته باشد چراکه ارسال از طرف D منجر به تصادم در ایستگاه B خواهد شد.

در این پروتکل وقوع تصادم فقط در حین ارسال فریمهای RTS و CTS محتمل است. هرگاه در هنگام ارسال این دو فریم تصادم رخ بدهد ، الگوریتم **"عقب‌گرد توانی"** که در استاندارد IEEE 802.3 معرفی شد ، اجرا می‌شود.

در استاندارد IEEE 802.11 اولاً توپولوژی شبکه ثابت نیست و با زمان تغییر می‌کند و بهمین دلیل ایستگاهها موظفند بطور متناوب اطلاعاتی را از وضعیت شبکه بدست آورده و در جدولی ذخیره کنند. ثانیاً برای برقراری ارتباط بین ایستگاههایی که در بُرد یکدیگر نیستند باید مسیریابی انجام شود.

---

[1] Request To Send
[2] Clear To Send

## ۶) مراجع این فصل

مجموعهٔ مراجع زیر می‌توانند برای دست آوردن جزییات دقیق و تحقیق جامع در مورد مفاهیم معرفی شده در این فصل مفید واقع شوند.

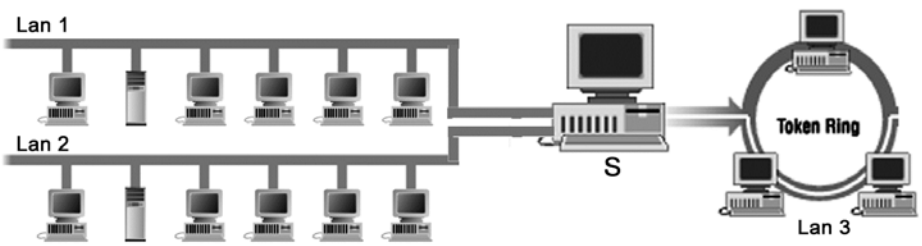

| | |
|---|---|
| **"Computer Networks" , Andrew S.Tanenbaum, Third Edition, Prentice-Hall, 1996.** | |
| IEEE, "IEEE Standards for Local Area Networks: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications", IEEE, New York, New York, 1985. | |
| IEEE, "IEEE Standards for Local Area Networks: Token-Passing Bus Access Method and Physical Layer Specification", IEEE, New York, New York, 1985. | |
| IEEE, "IEEE Standards for Local Area Networks: Token Ring Access Method and Physical Layer Specifications", IEEE, New York, New York, 1985. | |
| IEEE, "IEEE Standards for Local Area Networks: Logical Link Control", IEEE, New York, New York, 1985. | |
| IEEE 802.3/ISO 8802-3 Information processing systems - Local area networks - Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications, 1993. | |
| **RFC1661** | The Point-to-Point Protocol (PPP). W. Simpson, Editor. July 1994. |
| **RFC1663** | PPP Reliable Transmission. D. Rand. July 1994. |
| **RFC1717** | The PPP Multilink Protocol (MP). K. Sklower, B. Lloyd, G. McGregor, D. Carr. November 1994. |
| **RFC1042** | Standard for the transmission of IP datagrams over IEEE 802 networks. J. Postel, J.K. Reynolds. Feb-01-1988. |
| **RFC1230** | IEEE 802.4 Token Bus MIB. K. McCloghrie, R. Fox. May-01-1991. |
| **RFC1231** | IEEE 802.5 Token Ring MIB. K. McCloghrie, R. Fox, E. Decker. |
| **RFC1055** | Nonstandard for transmission of IP datagrams over serial lines: SLIP. J.L. Romkey. Jun-01-1988. |

## ۱) مقدمه

وظیفهٔ لایهٔ اوّل در مدل TCP/IP آنست که یک فریم اطلاعاتی را بین دو کامپیوتر که بر روی یک کانال فیزیکی مشترک واقعند ، منتقل کرده و مسائلی را که در این انتقال ممکن است بروز کند (مثل خطاهای احتمالی کانال) حلّ و فصل نماید. در این نقطه رسالت لایهٔ اوّل پایان می‌یابد.

یک شبکهٔ محلی با توپولوژی Bus را در نظر بگیرید؛ وظیفهٔ لایهٔ اوّل در این شبکه آنست که یک فریم را از ماشینی که به کانال مشترک متصل است به ماشینی دیگر منتقل نماید و از دیدگاه این لایه ماشینی قابل دسترسی خواهد بود که مستقیماً به واسط فیزیکیِ انتقال[۱] متصل شده و آمادهٔ دریافت داده‌ها باشد و در ضمن قواعد شبکهٔ Bus را دقیقاً رعایت نماید. ساختار لایه فیزیکی شدیداً به توپولوژی و سخت‌افزار شبکه وابسته است.

حال ساختار شبکهٔ (۱-۳) را در نظر بگیرید.




شکل (۱-۳) ساختار به هم‌بندی سه شبکهٔ محلی مجزا

در شکل (۱-۳) سه شبکه محلی مستقل وجود دارد که یکی از آنها توپولوژی حلقه و دو شبکهٔ دیگر توپولوژی Bus دارند و هیچگونه ارتباطِ مستقیمِ فیزیکی بین کانالهای انتقال در این سه شبکه وجود ندارد. ماهیت انتقال در شبکهٔ حلقه با نوع BUS متفاوت است و امکان اتصال مستقیم این شبکه‌ها ذاتاً میسّر نیست. تنها نکته قابل توجه در ساختار فوق ایستگاه S است که همزمان به هر سه شبکه متصل است. (ایستگاه S را کامپیوتری با سه عدد کارت شبکه در نظر بگیرید که دوتا از کارتها از نوع Ethernet و یکی از نوع Token Ring است.) S بعنوان عضوی از شبکهٔ حلقه و دقیقاً هماهنگ با پروتکل IEEE 802.5 ، اقدام به ارسال و دریافت اطلاعات روی شبکهٔ محلی LAN3 می‌نماید و همچنین قادر است بعنوان عضوی از دو شبکهٔ BUS

[۱] Medium

اقدام به ارسال و دریافت از شبکه‌های LAN1 و LAN2 کند. سؤالی که مطرح است آنست که : " آیا فارغ از ساختار این سه شبکه، ایستگاه S می‌تواند داده‌هائی را از یک ایستگاه در شبکهٔ LAN1 دریافت کرده و آنرا به ایستگاهی در شبکهٔ LAN3 برساند؟"

جواب مثبت است و رسالت لایهٔ دوم از همین جا آغاز می‌شود یعنی "هدایت بسته‌های اطلاعاتی از شبکه‌ای به شبکهٔ دیگر". در ادبیات شبکه به ایستگاه S ، مسیریاب[۱] و به عمل هدایت بسته‌های اطلاعاتی از مبداء به مقصد مسیریابی[۲] گفته می‌شود.

پس برای ارتباط اطلاعاتی بین دو ایستگاه روی LAN1 و LAN3 (با در نظر گرفتن اختلافِ ساختار فیزیکی دو شبکه) ایستگاه S بوسیلهٔ سخت‌افزار کارت شبکه، داده‌ها را از کانال فیزیکیِ LAN1 دریافت می‌نماید (این داده‌ها در فیلد داده[۳] از فریم لایهٔ فیزیکی شبکه Ethernet قرار گرفته‌اند) و پس از استخراج داده‌ها، مجدّداً آنها را درون فیلد داده از فریم شبکهٔ حلقه قرار داده و روی شبکه تزریق می‌کند.

به قالب فریم در شبکهٔ Ethernet شکل (۲-۳) دقت نمائید.

| Preamble | Start of Frame Delimiter | Destination Address | Source Address | Frame Length | Data (Payload) | CRC |
|---|---|---|---|---|---|---|

شکل (۲-۳) قالب فریم در شبکه Ethernet

فیلدهائی که در فریم چنین شبکه‌ای تعریف شده فقط به این کار می‌آید که فریمی از یک طرف کانال ارسال و طرف دیگر دریافت شود. دو فیلد آدرسی که در این فریم تعریف شده فقط و فقط روی همین شبکه معتبر است و خارج از این شبکه هیچگونه مورد استفاده‌ای ندارند. چرا که وقتی یک واحد اطلاعاتی از یک شبکهٔ محلی به شبکهٔ محلی دیگر منتقل می‌شود ، کلاً قالب فریم و به تبع آن محتوای فیلدهای آدرس باید عوض شود.

به آدرسهائی که در لایه فیزیکی (لایهٔ اوّل) تعریف می‌شود و فقط برای انتقال فریمها روی کانال مورد استفاده هستند "آدرسهای MAC" گفته می‌شود. در حقیقت این آدرسها روشی برای تحریکِ سخت‌افزار کارت شبکه هستند تا اطلاعات را از روی کانال مشترک بردارد. بنابراین چگونگی تعریف این آدرسها و اصول آدرس‌دهی و اندازهٔ این آدرسها (برحسب بایت) شدیداً به ساختار شبکه وابسته است. مثلاً در پروتکل SLIP که ارتباط فقط دوبه دو است اصلاً

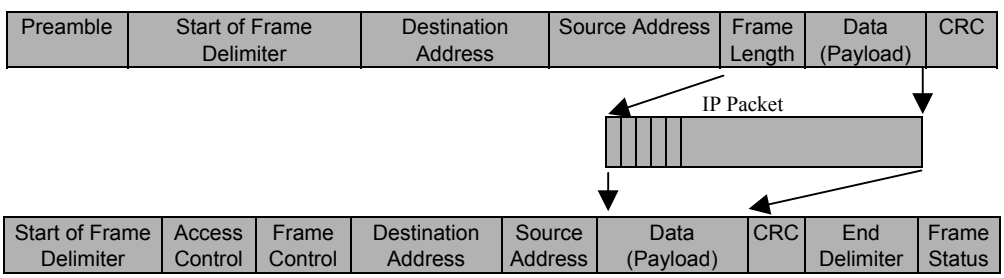
---

۱ Router
۲ Routing
۳ Payload

احتیاجی به فیلد آدرس MAC وجود ندارد در حالی که در پروتکل CSMA/CD (شبکهٔ Ethernet) این آدرسها شش بایتی هستند.

بی‌نظمی در شبکه‌های مختلف و تنوع توپولوژی و پروتکلها و روشهای آدرس‌دهی ، ایجاب می‌کند که برای یکپارچه‌سازی شبکه‌ها و امکان برقراری ارتباط بین آنها در لایهٔ دومِ شبکه تمهیدی اندیشیده شود. اوّلین کار بنیادی ، تعریف آدرسهای جهانی و استاندارد برای تمامی ایستگاههای موجود بر روی شبکه‌های مختلف جهان می‌باشد. در ضمن باید ساختار بسته‌ای که درون فیلدِ داده از فریم هر شبکه قرار می‌گیرد برای تمام شبکه‌ها یکسان و استاندارد باشد بگونه‌ای که هیچ وابستگی به نوع شبکه و سخت‌افزار آن نداشته باشد.

در مدل TCP/IP به واحد اطلاعاتی که باید درون فیلد داده از فریم لایه فیزیکی قرار بگیرد بستهٔ IP[1] گفته می‌شود. تعریف قالب استاندارد یک بستهٔ IP و چگونگی آدرس‌دهی ماشینهای مختلف شبکه و روشهای مسیریابی در لایهٔ دوم از مدل TCP/IP (لایهٔ اینترنت) تعریف شده است. این بسته برای عبور از یک شبکه به شبکهٔ دیگر تغییری نخواهد کرد بلکه از فیلد داده در فریم لایهٔ فیزیکی استخراج شده ، در فریم دیگری قرار می‌گیرد و بدینگونه در شبکه‌ها طی مسیر می‌کند؛ به شکل (۳-۳) دقت کنید. در این شکل فرض شده که یک مسیریاب یک بسته از شبکهٔ Ethernet تحویل گرفته و می‌خواهد آنرا به شبکهٔ حلقه هدایت نماید؛ برای این کار بسته را از فیلد دادهٔ فریمِ شبکهٔ اول استخراج کرده و آنرا درون فریم شبکهٔ دوم قرار داده آنرا ارسال می‌نماید.



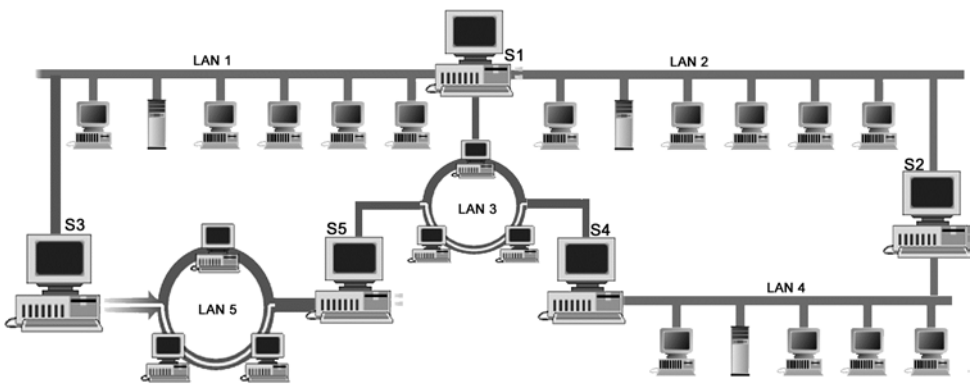

شکل (۳-۳) تغییر قالب فریم و آدرسهای فیزیکی در یک مسیریاب

[1] IP Packet

درون یک بسته تعدادی فیلد به منظور تسهیل در هدایت داده‌ها از یک شبکه به شبکهٔ دیگر در نظر گرفته شده است. دو تا از این فیلدها آدرس مبداء و مقصد هستند که این دو، آدرسهای جهانی محسوب می‌شوند و دو ماشین را فارغ از ساختار شبکه‌ای که به آن متصل هستند بصورت یکتا مشخص می‌کند؛ در شبکهٔ اینترنت به این آدرسها ، آدرسهای IP گفته می‌شود. وقتی یک بستهٔ IP از یک شبکه روی شبکه ای دیگر منتقل می‌شود آدرس‌های MAC (یا کلاً فریم آن) عوض می‌شود ولیکن ساختار بسته ای که درون فیلد داده قرار گرفته و همچنین آدرسهای IP عوض نخواهد شد. در ادامهٔ این فصل ساختار بستهٔ IP را دقیقاً بررسی خواهیم کرد؛ قبل از آن مقدمه‌ای در مورد مسیریاب خواهیم داشت.

### ۱-۱) مسیریاب

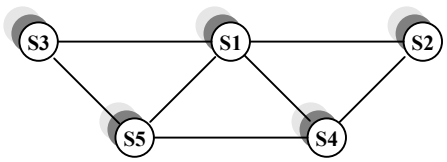
در ساختار شبکهٔ شکل (٤-۳) برای ارتباط دو ماشین روی شبکه‌های متفاوت تنها راه ارتباطی ماشین S است که آنرا مسیریاب نامیدیم. حال ساختار شبکه شکل (٤-۳) را در نظر بگیرید.




شکل (٤-۳) به هم بندی چند شبکهٔ محلی

در این ساختار فرضی $S_1$ همزمان در شبکه‌های محلی ۱ و ۲ و ۳ حضور دارد؛ $S_2$ در شبکه‌های ۲ و ٤ ؛ $S_3$ در شبکه‌های ۵ و ۱ ؛ $S_4$ در شبکه‌های ٤ و ۳ ؛ $S_5$ در شبکه‌های ۳ و ۵ حضور دارد. هر یک از ماشینهای $S_1$ تا $S_5$ می‌توانند نقش هدایت بسته‌ها را از یک شبکه به شبکهٔ دیگر بر عهده بگیرند. اگر فقط چگونگی ارتباط مسیریابها را با هم در نظر بگیریم و ماشینهای دو شبکهٔ محلی (یعنی ماشینهای میزبان) را که هیچ نقشی در مسیریابی بازی نمی‌کنند حذف نمائیم ، ساختار مسیرها بصورت شکل (۵-۳) در می‌آید. در این شکل کلاً ساختار

شبکه‌های محلی حذف شده و فقط وجود یا عدم وجود ارتباط بین مسیریابها نشان داده شده است. مثلاً اگر چه ارتباط $S_1$ و $S_4$ از طریق شبکهٔ محلی LAN3 برقرار می‌شود ولی در شکل (۵-۳) فقط وجود ارتباط (بوسیلهٔ یک خط مستقیم) نشان داده شده است. حال وقتی تعداد شبکه‌های متصل بهم را متعدّد فرض کنید شکل بصورت گرافی پیچیده در خواهد آمد؛ این گراف زیرساخت ارتباطی شبکه‌ها را تشکیل می‌دهد که "زیرشبکه"[۱] نامیده می‌شود.




شکل (۵-۳) زیرساخت ارتباطی مسیریابها

در زیرساخت ارتباطی شبکه‌ها ، مسیرهای متعددی بین دو مسیریاب وجود دارد و بنابراین هر یک از مسیریابها به غیر از وظیفهٔ هدایت بسته‌ها بایستی راهی را برای طی مسیر بسته به سمت مقصد برگزینند که بهینه باشد.

در ساده‌ترین تعریف، مسیریاب ماشینی است که تعدادی ورودی / خروجی داشته و بسته‌های اطلاعاتی را از ورودیها تحویل گرفته و براساس آدرس مقصد، یکی از کانالهای خروجی را برای انتقال بسته انتخاب می‌کند؛ به نحوی که بسته را به مقصد نزدیک نماید.

ماشینی را که هیچ نقشی در هدایت بسته‌های اطلاعاتی روی شبکه ندارد و فقط تولید کننده یا مصرف کنندهٔ بسته‌های اطلاعاتی است ، "ماشین میزبان"[۲] می‌گویند.

از این به بعد هرگاه ساختار زیرشبکهٔ ارتباطی را در قالب یک گراف نشان دادیم، گره‌های این گراف ، مسیریابها را تصویر می‌کند و خطوط بین دو گره در گراف ، کانال ارتباطی بین دو مسیریاب را نشان می‌دهد و بنابراین ماشینهای میزبانِ شبکه را نشان نخواهیم داد.

مجموعهٔ مسیریابها و کانالهای ارتباطی بین آنها ، توپولوژی زیر شبکه را تشکیل می‌دهد و خرابی یکی از مسیریابها یا یکی از کانالهای ارتباطی، توپولوژی زیر شبکه را تغییر داده و در نتیجه ، عملِ مسیریابی در شبکه تحت تأثیر قرار می‌گیرد.

در بیانی ساده ترافیکِ یک کانال ، متوسط تعداد بسته‌های اطلاعاتی است که در واحد زمان روی یکی از کانالهای ورودیِ یک مسیریاب ارسال شده و مسیریاب موظّف به دریافت و

---

[۱] Subnet
[۲] Host Machine

پردازش آن می‌باشد. با توجّه به آنکه تولید بسته‌های اطلاعاتی کاملاً تصادفی و نامعین است بنابراین ترافیک لحظه‌ای هر کانال کاملاً متغیر با زمان خواهد بود.

با این مقدمه ابتدا لایهٔ اینترنت ( لایهٔ شبکه) از مدل TCP/IP را معرفی کرده و در فصلی مجزّا به الگوریتمهای مسیریابی در شبکهٔ اینترنت خواهیم پرداخت.

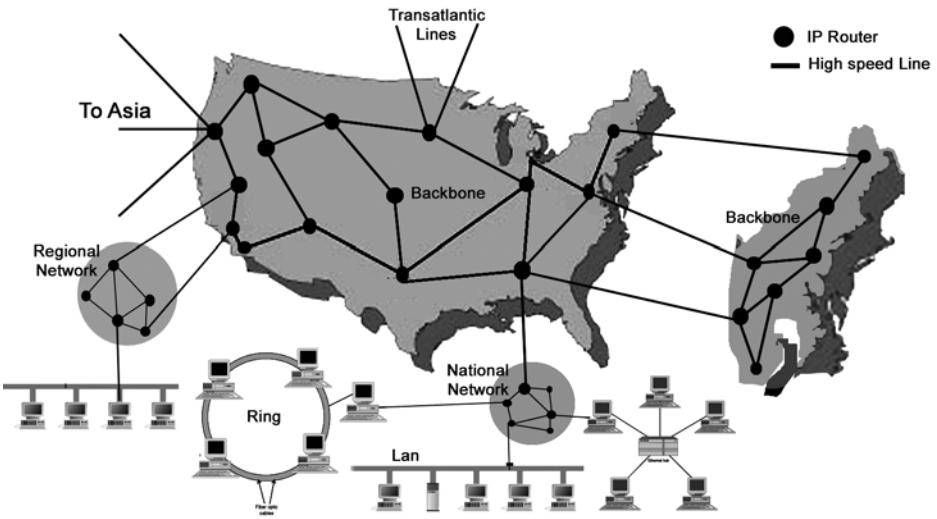
## ۲) لایهٔ اینترنت

جوهرهٔ اینترنت به گونه ای شکل گرفته است که مجموعه ای از شبکه‌های خودمختار[۱] را به همدیگر وصل می‌نماید . هیچگونه ساختار حقیقی و ثابتی نمی‌توان برای اینترنت متصوّر شد . این نکته را بایستی یادآور شویم که در قسمت "زیرشبکه" از شبکهٔ اینترنت تعدادی از خطوط ارتباطی با پهنای باند (نرخ ارسال) بسیار بالا و مسیریابهای بسیار سریع و هوشمند ، برای پیکرهٔ شبکهٔ جهانیِ اینترنت یک "ستون فقرات"[۲] تشکیل داده است. شبکه‌های منطقه ای و محلّی پیرامون این ستون فقرات شکل گرفته و ترافیک دادهٔ آنها به نحوی از این ستون فقرات خواهد گذشت . ستون فقرات در شبکهٔ اینترنت که با سرمایه گذاری عظیمی در آمریکا، اروپا و قسمتهایی از اقیانوسیه و آسیا ایجاد شده است حجم بسیار وسیعی از بسته‌های اطلاعاتی را در هر ثانیه حمل می‌کنند و اکثر شبکه‌های منطقه‌ای و محلی یا ارائه دهندگان سرویسهای اینترنت[۳] به نحوی با یکی از گره‌های این ستون فقرات در ارتباطند . در شکل (٦-۳) سیمای کلّی و ساده از مفهوم ستون فقرات را می‌بینید.

به گونه ای که در بخش قبلی اشاره شد قراردادی که حمل و تردّد بسته‌های اطلاعاتی و همچنین مسیریابی صحیح آنها را از مبدأ به مقصد ، مدیریّت و سازماندهی می‌نماید پروتکل IP[٤] نام دارد. درحقیقت پروتکل IP که روی تمامی ماشینهای شبکه اینترنت وجود دارد بسته‌های اطلاعاتی را (بسته‌های IP) از مبداء تا مقصد هدایت می‌نماید ، فارغ از آنکه آیا ماشینهای مبدأ و مقصد روی یک شبکه هستند یا چندین شبکهٔ دیگر بین آنها واقع شده است.

---

۱ Autonomous **این اصطلاح در فصل بعدی معرفی شده است.**
۲ Backbone
۳ Internet Service Provider ( ISP )
٤ Internet protocol




شکل (٦-٣) سیمای کلّی و تجسّمی ستون فقرات در شبکهٔ اینترنت

ساده ترین تعریف برای پروتکل IP روی شبکهٔ اینترنت بصورت زیر خلاصه می‌شود :

لایهٔ IP یک واحد از داده‌ها را از لایهٔ بالاتر تحویل می‌گیرد؛ به این واحد اطلاعات معمولاً یک "دیتاگرام" گفته می‌شود[1]. امکان دارد طول این دیتاگرام بزرگ باشد ، در چنین موردی لایهٔ IP آنرا به واحدهای کوچکتری که هر کدام "قطعه"[2] نام دارد شکسته و با تشکیل یک بستهٔ IP به ازای هر قطعه ، اطلاعات لازم برای طی مسیر در شبکه را به آنها اضافه میکند و سپس آنها را روی شبکه به جریان می‌اندازد؛ هر مسیریاب با بررسی و پردازش بسته‌ها ، آنها را تا مقصد هدایت می‌کند . هر چند طول یک بسته IP می‌تواند حدّاکثر 64Kbyte باشد و لیکن در عمل عموماً طول بسته‌ها حدود ۱۵۰۰ بایت است . ( این قضیه به دلیل آنست که اکثر شبکه‌های محلی دنیا اعمّ از Bus، حلقه ، ستاره ، ... طول فریمی نزدیک به یک تا چند کیلو بایت دارند) پروتکل IP مجبور است هنگام قطعه قطعه کردن یک دیتاگرام ، برای کلّ آن یک شمارهٔ مشخصه و برای هر قطعه یک شمارهٔ ترتیب در نظر بگیرد تا آن دیتاگرام بتواند در مقصد برای تحویل به لایهٔ بالاتر یعنی لایهٔ انتقال بازسازی شود .

---

[1] **اصطلاح دیتاگرام در ادبیّات شبکه‌های کامپیوتری به معانی متفاوت و در موارد متعدّد استفاده شده است. لذا به مورد استفادهٔ آن دقّت داشته باشید.**

[2] Fragment

(مجدداً تأکید می‌کنیم که در این مبحث ، دیتاگرام یک واحد اطلاعات است که به صورت یکجا از لایهٔ IP به لایهٔ انتقال تحویل داده می‌شود یا بالعکس لایه انتقال آنرا جهت ارسال روی شبکه به لایهٔ IP تحویل داده و ممکن است شکسته شود)

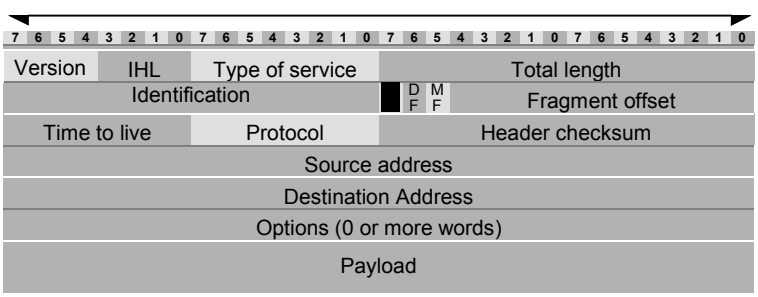
در کنار پروتکل IP چندین پروتکل دیگر مثل ICMP, ARP, RARP, RIP و ... تعریف شده که پروتکل IP را در عملکرد بهتر ، مسیریابی صحیح، مدیریت خطاهای احتمالی یا کشف آدرسهای ناشناخته کمک می‌کنند.

تواناییهایی که پروتکل IP و پروتکلهای جانبی آن عرضه می‌کنند این امکان را فراهم آورده است که تمامی شبکه‌ها و ابزارهای شبکه ای (مثل ماشینهای میزبان ، مسیریابها ، پلها ، و... ) فارغ از نوع ماشین و نوع سخت افزار و حتی با وجود تفاوت در سیستم عامل مورد استفادهٔ آنها ، بتوانند بسته‌های IP را با یکدیگر مبادله کنند . پروتکل IP ساختاری استاندارد دارد و به هیچ سخت افزار یا سیستم عامل خاصّ وابسته نیست.

بعنوان اولین گام در شناخت پروتکل IP لازم است قالب یک بستهٔ IP را کالبد شکافی کرده و در گامهای بعدی چگونگی آدرس دهی ماشینها و انواع کلاسهای آدرس در شبکهٔ اینترنت را معرفی نموده و نهایتاً به روشهای مسیریابی و همچنین تعریف پروتکلهای وابسته به IP بپردازیم.

### ۱-۲) قالب یک بستهٔ IP

شکل (۷-۳) قالب یک بسته IP را به تصویر کشیده است. یک بستهٔ IP از دو قسمت سرآیند و قسمت حمل داده تشکیل شده است. مجموعهٔ اطلاعاتی که در سرآیند بستهٔ IP درج می‌شود توسط مسیریابها مورد استفاده و پردازش قرار می‌گیرد.




شکل (۷-۳) قالب یک بستهٔ IP

➢ فیلد **Version** : اولین فیلد در سرآیند یک بستهٔ IP که چهار بیت است نسخهٔ پروتکلِ IP که این بسته بر اساس آن سازماندهی و ارسال شده است را تعیین می‌کند. در حال حاضر تمامی شبکه‌ها و مسیریابها از نسخهٔ شماره ٤ پروتکل IP پشتیبانی می‌کنند. اگرچه امروزه نسخهٔ شمارهٔ ٦ پروتکل IP به نامهای IPng یا IPv6 معرفی و در حال بررسی و نصب است ولیکن بسیاری از مسیریابها در شبکه‌های دنیا هنوز برای پذیرش این پروتکل آمادگی ندارند و به نظر می‌رسد که تا سال ۲۰۰۵ نگارشِ جدید جهانی نشود. عددی که در حال حاضر در این فیلد قرار می‌گیرد ٤ یا $(0100)_2$ است .

➢ فیلد **IHL**[1] : این فیلد هم چهاربیتی است و طول کل سرآیند بسته را بر مبنای کلمات ۳۲ بیتی مشخص می‌نماید. بعنوان مثال اگر در این فیلد عدد ۱۰ قرار گرفته باشد بدین معناست که کل سرآیند ۳۲۰ بیت معادل چهل بایت خواهد بود . اگر به ساختار یک بستهٔ IP دقّت شود به غیر از فیلد Options که اختیاری است ، وجود تمامی فیلدهای سرآیند الزامی می‌باشد . طول قسمت اجباریِ سرآیند ۲۰ بایت است و بهمین دلیل حداقل عددی که در فیلد IHL قرار می‌گیرد ۵ یا $(0101)_2$ خواهد بود و هر مقدار کمتر از ۵ به عنوان خطا تلقّی شده و منجر به حذف بسته خواهد شد. با توجه به طول ٤ بیتی این فیلد ، بدیهی است که حداکثر مقدار آن ۱۵ یا $(1111)_2$ خواهد بود که در این صورت طول قسمت سرآیند ٦۰ بایت (۱۵×٤) و طول قسمت اختیاری ٤۰ بایت می‌باشد. قسمت اختیاری در سرآیند برای اضافه کردن اطلاعاتی مثل آدرس مسیرهای پیموده شده ، "مهر زمان" و برخی دیگر از گزینه‌هاست که در ادامه توضیح داده خواهد شد.

➢ فیلد **Type of service** : این فیلد هشت بیتی است و توسط آن ماشین میزبان ( یعنی ماشین تولید کنندهٔ بستهٔ IP) از مجموعهٔ زیرشبکه (یعنی مجموعهٔ مسیریابهای بین راه) تقاضای سرویس ویژه‌ای برای ارسال یک دیتاگرام می‌نماید . بعنوان مثال ممکن است یک ماشین میزبان بخواهد دیتاگرامِ صدا یا تصویر برای ماشین مقصد ارسال نماید؛ در چنین شرایطی از زیرشبکه تقاضای ارسال سریع و به موقع اطلاعات را دارد نه قابلیت اطمینان صد در صد ، چرا که اگر یک یا چند بیت از داده‌های

[1] IP Header Length

ارسالی در مسیر دچـار خرابـی شـود تـاثیر چندانـی در کیفیـت کـار نخواهـد گذاشـت ولـی اگـر بسـته‌های حـاوی اطلاعـات صـدا یـا تصویـر بـه سـرعت و سـر موقـع تحویـل نشـود اشکال عمـده بوجـود خواهـد آمـد. در چنیـن مواقعـی ماشـین میزبـان از زیرشـبکه تقاضـای سـرویس سـریع (و لاجـرم غـیر قـابل اطمینـان ) می‌نمـاید. در برخـی از محیطـهای دیگـر مثـل ارسـال نامـهٔ الکـترونیکی یـا مبادلـهٔ فـایل انتظـار اطمینـان[1] صـد درصـد از زیـر شـبکه وجود دارد و سرعت تاثیر چندانـی بـر کیفیّـت کـار نـدارد .

از طریق این فیلد نوع سرویس درخواستی مشخص می‌شود، این فیلد خودش به چنـد بخـش تقسیم شده است :

| P2 | P1 | P0 | D | T | R | – | – |
|---|---|---|---|---|---|---|---|
| تقدّم بسته | | | تأخیر | توان خروجی | قابلیّت اطمینان | بلا استفاده | |

الف) سه بیت سمت چـپ ، اولویـت بسـتهٔ IP را تعییـن می‌کنـد . اگـر در ایـن سـه بیـت صفر قرار گرفته باشـد بسـته اطلاعـاتی از نـوع معمولـی تلقـی می‌شـود ، یعنـی دارای پـایین تریـن مقـدار اولویـت اسـت و اگـر مقـدار ۷ یعنـی 2(۱۱۱) در ایـن سـه بیـت قـرار گرفتـه باشد بالاترین اولویـت بـرای بسـته درنظـر گرفتـه می‌شـود . مسـیریاب در بیـن بسـته‌های IP کـه از کانالـهای مختلـف وارد شـده‌اند ، بسـته‌هایی را زودتـر پـردازش و مسـیریابی می‌کنـد که دارای حقّ تقدم و اولویـت بـالاتری باشـند. بسـته‌های بـا حـقّ تقـدم بـالا بـرای عملیـاتی نظیر ارسال بسـته‌های اطلاعـاتی بـه منظـور تنظیـم و پیکربنـدی پارامترهـای زیرشـبکه مـورد استفاده قـرار میگـیرد. ( مثـلاً بـرای گـزارش یـک خرابـی در زیـر شـبکه یـا مبادلـهٔ جـداول مسـیریابی )

ب ) بیتهای **R , T , D** : بیـت D بـه معنـای تـأخیر[2] ، بیـت R بـه معنـای قـابلیت اطمینـان و بیـت T بـه معنـای تـوان خروجـی خط[3] اسـت و ماشـین میزبـان بـا قـرار دادن ۱ در ایـن بیتها انتظارش را از زیرشـبکه بیـان می‌کنـد . مسـیریابها بـا بررسـی ایـن سـه بیـت می‌تواننـد در مـورد انتخـاب مسـیر مناسـب تصمیـم بگـیرند . بعنـوان مثـال یـک کانـال مـاهواره‌ای دارای تـوان خروجـی بسـیار بـالا ( از لحـاظ نـرخ ارسـال) ولیکـن تـاخیر نامناسـب اسـت ، در صورتی که یـک خـط اجـاره‌ای می‌توانـد دارای تـاخیر کمـتر و همچنیـن تـوان خروجـی

[1] Reliability
[2] Delay
[3] Throughput

همچنین توان خروجی کمتر باشد . اگر در ارسال یک بستهٔ IP ، تاخیر پذیرفتنی نباشد با یک کردن بیتِ D مسیریاب را وادار می‌کند که حتی الامکان از خطوط پرتاخیر مثل خط ماهواره‌ای استفاده نکند؛ با یک کردن بیت R مسیریاب موظف خواهد بود تا از بین خطوط خروجی امن ترین و کم خطاترین آنها را انتخاب کند. ( البته در صورت امکان)

اکثر مسیریابهای تجاری فیلد Type of Service را نادیده می‌گیرند و اهمیتی به محتوای آن نمی‌دهند .

- فیلد **Total Length** : در این فیلدِ ۱۶ بیتی عددی قرار می‌گیرد که طول کل بستهٔ IP را که شامل مجموع اندازهٔ سرآیند و ناحیه داده است ، تعیین می‌کند . مبنای طول برحسب بایت است و بنابراین حداکثر طول کل بستهٔ IP می‌تواند ۶۵۵۳۵ بایت باشد.

- فیلد **Identification** : همانگونه که قبلاً اشاره شد برخی از مواقع مسیریابها یا ماشینهای میزبان مجبورند یک دیتاگرام را به قطعات کوچکتر بشکنند و ماشین مقصد مجبور است آنها را بازسازی کند ، بنابراین وقتی یک دیتاگرامِ واحد شکسته می‌شود باید مشخصه‌ای داشته باشد تا در هنگام بازسازیِ آن در مقصد بتوان قطعه‌های آن دیتاگرام را از بقیه جدا کرد. در این فیلدِ ۱۶ بیتی عَددی قرار می‌گیرد که شمارهٔ یک دیتاگرام واحد را مشخص می‌کند . کلیه بسته‌های IP که با این شماره وارد می‌شوند قطعه‌های مربوط به یک دیتاگرام بوده و باید پس از گردآوری قطعه‌ها ، آن را مجدداً بازسازی کرد. بعنوان مثال اگر در این فیلد عدد ۱۶۵۲ قرار بگیرد تمامی بسته‌های IP که مشخصهٔ ۱۶۵۲ دارند قطعه‌های مربوط به یک دیتاگرام هستند و پس از دریافت کل قطعه‌ها باید بازسازی شوند. البته برای حفظ ترتیب ، هر قطعه گذشته از یک شمارهٔ مشخصه بایستی دارای شمارهٔ ترتیب نیز باشد تا بتوان آنها را طبق این شماره مرتب و بازسازی کرد.

- فیلد **Fragment offset** : این فیلد در سه بخش سازماندهی شده است :

الف) بیت DF[1] : با یک شدن این بیت در یک بستهٔ IP هیچ مسیریابی حقّ ندارد آن را قطعه قطعه کند ، چرا که مقصد قادر به بازسازی دیتاگرام‌های تکه تکه شده نیست . بعنوان مثال وقتی که یک کامپیوتر بدون دیسک از طریق ROM بوت می‌شود اطلاعات هسته اصلی سیستم عامل باید در قالب یک دیتاگرام واحد برای آن کامپیوتر ارسال شود چرا که آن کامپیوتر در حال حاضر نرم افزار لازم برای بازسازی بسته‌های قطعه قطعه شده را ندارد. اگر این بیت به ۱ تنظیم شده باشد و مسیریابی نتواند آنرا به دلیل بزرگی اندازهٔ آن ، انتقال بدهد لاجَرَم آنرا حذف خواهد کرد.

ب) بیت MF[2] : این بیت مشخص می‌کند که آیا بستهٔ IP آخرین قطعه از یک دیتاگرام محسوب می‌شود یا باز هم قطعه‌های بعدی وجود دارد. در آخرین قطعه از یک دیتاگرام بیت MF صفر خواهد بود و در بقیه الزاماً ۱ است.

ج) Fragment offset : این قسمت که سیزده بیتی است در حقیقت شماره ترتیب هر قطعه در یک دیتاگرام شکسته شده محسوب می‌شود. با توجه به سیزده بیتی بودن این فیلد، یک دیتاگرام حداکثر می‌تواند به ۸۱۹۲ تکه تقسیم شود.

نکته بسیار مهم در مورد این فیلد آن است که اندازهٔ هر قطعه باید ضریبی از ۸ باشد یعنی به استثنای قطعهٔ آخر ، اندازه بقیهٔ قطعه‌ها بایستی بگونه ای انتخاب شود که ضریبی از ۸ بایت باشد؛ مثلاً اگر در فیلد آفست مقدار ۷ قرار بگیرد نشان می‌دهد که محل قرار گرفتن قطعهٔ جاری در دیتاگرام بازسازی شده در موقعیت بایت پنجاه وششم (۵۶=۸*۷) خواهد بود. به عنوان مثالی دیگر فرض کنید مسیریابی مجبور است یک دیتاگرام به طول ۵۰۰۰ بایت را قطعه قطعه کند به گونه‌ای که اندازهٔ هر قطعه زیر ۱۵۰۰ بایت باشد. در چنین موردی نمیتواند اندازهٔ هر قطعه را ۱۲۵۰ بایت در نظر بگیرد چرا که ضریبی از ۸ نیست ولی اندازهٔ ۱۲۸۰ مناسب است. در این حالت مسیریاب ، دیتاگرام را به سه بستهٔ ۱۲۸۰ بایتی و یک بستهٔ ۱۱۶۰ بایتی می‌شکند. در این مثال فرض کنید مسیریاب شمارهٔ ۲۳۲۲ را به عنوان مشخصهٔ دیتاگرام انتخاب کرده است؛ بنابراین برای هر یک از چهار قطعهٔ دیتاگرام ، فیلد آفست و مشخصه به صورت زیر خواهد بود:

---

[1] Don’t Fragment
[2] More fragment

| شمارهٔ قطعه | Identification | Fragment Offset | بیت MF | آدرس محّل قرار گرفتن قطعه در دیتاگرام | طول هر قطعه |
|---|---|---|---|---|---|
| قطعهٔ شماره ۱ | 2322 | 0 | 1 | 8*0=0 | ۱۲۸۰ |
| قطعهٔ شماره ۲ | 2322 | 160 | 1 | 8*160=1280 | ۱۲۸۰ |
| قطعهٔ شماره ۳ | 2322 | 320 | 1 | 8*320=2560 | ۱۲۸۰ |
| آخرین قطعه | 2322 | 480 | 0 | 8*480=3840 | ۱۱۶۰ |

(بیت اول یعنی بیت سمت چپ از این فیلد که در شکل (۷-۳) به رنگ سیاه علامت گذاری شده مورد استفاده ندارد )

ممکن است یک دیتاگرامِ واحد از یک ماشین میزبان روی زیر شبکه تزریق شود و در طول مسیر به مسیریابی برسد که به دلیلی مجبور به شکستن آن به قطعات کوچکتر شود. در چنین حالتی باز هم وظیفهٔ بازسازی قطعات به عهدهٔ ماشین مقصد می‌باشد. به عبارت ساده تر عمل شکستن یک دیتاگرام در هر جای زیر شبکه ممکن است اتفاق بیفتد ولیکن عمل باز سازی فقط در ماشین مقصد انجام می‌شود.

- فیلد **Time To Live** : این فیلد هشت بیتی در نقش یک شمارنده ، طول عمر بسته را مشخص می‌کند. طول عمر یک بسته بطور ضمنی به زمانی اشاره می‌کند که یک بسته IP می‌تواند بر روی شبکه سرگردان باشد . حداکثر طول عمر یک بسته ، ۲۵۵ خواهد بود که به ازای عبور از هر مسیریاب[1] از مقدار این فیلد یک واحد کم می‌شود. هر گاه یک بستهٔ IP به دلیل بافر شدن در حافظهٔ یک مسیریاب زمانی را معطّل بماند ، به ازای هر ثانیه یک واحد از این فیلد کم خواهد شد. به محض آنکه مقدار این فیلد به صفر برسد بستهٔ IP در هر نقطه از مسیر باشد حذف شده و از ادامهٔ سیرِ آن به سمت مقصد جلوگیری خواهد شد. ( البته معمولاً یک پیام هشدار به ماشینی که آن بسته را تولید کرده باز پس فرستاده خواهد شد.)

اگرچه بزرگترین عددی که در فیلد طول عمر بسته قرار می‌گیرد ۲۵۵ است ولی در عمل مقداری که سیستمهای عامل در این فیلد قرار می‌دهند چیزی حدود ۳۰ است. ( البته می‌توان مقدار پیش فرض آن را عوض کرد )

این فیلد برای پاکسازی زیر شبکه از بسته‌های IP که به هر دلیل در یک مسیرِ بسته می‌چرخند بسیار حیاتی است وگرنه پس از مدتی کلّ زیر شبکه از بسته‌های

[1] در ادبیات شبکه به عبور بسته از یک مسیریاب یک جهش یا Hop گفته می‌شود.

آشغال پر خواهد شد. بسته‌های سرگردان گاهاً به این دلیل بوجود می‌آیند که جداول مسیریابی در بعضی از مسیریابها آلوده به اطلاعات نادرست[۱] شده‌اند. سرگردانی یک بسته در زیرشبکه مسئلهٔ غیر ممکنی نیست و گاهی اتفاق می‌افتد.

- فیلد **Protocol** : دیتاگرامی که در فیلد داده از یک بستهٔ IP حمل می‌شود با ساختمان دادهٔ خاصّ از لایهٔ بالاتر تحویلِ پروتکل IP شده تا روی شبکه ارسال شود. بعنوان مثال ممکن است این داده‌ها را پروتکل TCP در لایه بالاتر ارسال کرده باشد و یا ممکن است این کار توسط پروتکل UDP انجام شده باشد. بنابراین مقدار این فیلد شمارهٔ پروتکلی است که در لایه بالاتر تقاضای ارسال یک دیتاگرام کرده است؛ بسته‌ها پس از دریافت در مقصد باید به پروتکل تعیین شده تحویل داده شود. فیلد پروتکل ۸ بیتی است و پروتکلهای لایهٔ بالاتر دارای یک شمارهٔ هشت بیتیِ منحصر بفرد و استاندارد هستند که درصورت نیاز به دانستن شمارهٔ آنها می‌توانید به انتهای این فصل مراجعه کنید .

- فیلد **Header Checksum** *:* این فیلد که شانزده بیتی است به منظور کشف خطاهای احتمالی در سرآیند هر بستهٔ IP استفاده می‌شود . برای محاسبهٔ کد کشف خطا ، کلّ سرآیند بصورت دو بایت، دوبایت با یکدیگر جمع می‌شود. نهایتاً حاصل جمع به روش "مکمل یک"[۲] منفی می‌شود و این عدد منفی در این فیلد از سرآیند قرار می‌گیرد .

در هر مسیریاب قبل از پردازش و مسیریابی ابتدا صحّتِ اطلاعات درون سرآیند بررسی می‌شود. روش بررسی بدینصورت است که اگر تمامی سرآیند بصورت دو بایت ، دوبایت در مبنای مکمل یک با یکدیگر جمع شود باید حاصل جمع ، صفر بدست آید؛ در غیر این صورت بستهٔ IP فاقد اعتبار بوده حذف خواهد شد.
دقت کنید که فیلد Checksum در هر مسیریاب باید از نو محاسبه و مقداردهی شود زیرا وقتی یک بستهٔ IP وارد یک مسیریاب می‌شود حداقّل فیلد TTL از آن بسته عوض خواهد شد.

فیلد Checksum برای کشف خطاهای احتمالی درون داده‌های فیلد Payload استفاده نمی‌شود چرا که اینگونه خطاها در لایه پایینتر یعنی لایه فیزیکی معمولاً

---

[۱] Corrupt
[۲] One's Compelement

توسط کدهای CRC نظارت می‌شود؛ در ضمن لایه‌های بالاتر نیز مسئلهٔ خطا را بررسی می‌کنند.

در حقیقت این فیلد برای کشف خطاهایی است که یک مسیریاب در تنظیم سرآیند یک بستهٔ IP مرتکب شده است .

- فیلد **Source Address** : هر ماشین میزبان در شبکهٔ اینترنت یک آدرس جهانی و یکتای ۳۲ بیتی دارد. بنابراین هر ماشین میزبان در هنگام تولید یک بستهٔ IP باید آدرس خودش را در این فیلد قرار بدهد .

بحث آدرسها در اینترنت یکی از مسائل بسیار مهمی است که در فصلی مجزا به آن خواهیم پرداخت . (به این آدرس از این بعد ، "آدرسِ IP" می‌گوئیم)

- فیلد **Destination Address** : در این فیلد آدرس ۳۲ بیتیِ مربوط به ماشین مقصد که باید بسته IP تحویل آن بشود ، قرار می‌گیرد.

- فیلد اختیاری **Options** : در این فیلدِ اختیاری می‌توان تا حداکثر ٤٠ بایت قرار داد و محتوی اطلاعاتی است که می‌تواند به مسیریابها در مورد یافتن مسیر مناسب کمک کند. البته به گونه ای که اشاره شد حداکثر فضای این فیلد ٤٠ بایت است که بسیار کم به نظر می‌رسد.

از آنجایی که در فضای ٤٠ بایتی این فیلد چندین گزینه می‌تواند قرار بگیرد و هر گزینه نیز اندازهٔ متفاوتی دارد ( بر حسب بایت ) لذا هر گزینه با یک کد یک بایتی مشخص می‌شود:

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Copy Flag | Option Class | | Option Number | | | | |

- بیت Copy Flag : ۱ بودن این بیت مشخص میکند که اگر مسیریابی مجبور به شکستن بستهٔ فعلی شود ، این گزینه در یکایک قطعات بسته تکرار شود. صفر بودن این بیت به معنای آنست که در هنگام شکسته شدن بسته این گزینه فقط در اولین قطعه وجود داشته باشد.
- دو بیت Option Class : این دو بیت نوع عملکرد گزینه را تعیین میکند:

  00 : عملکرد گزینه

  10 : عملکرد گزینه برای اشکالزدایی و مدیریت شبکه می‌باشد.

01 و 11 : تعریف نشـــده اسـت.

♦ پنـج بیـت Option Number : ایـن پنـج بیـــت نـــوع و معنـــای گزینـــه را مشـــخص میکند. تاکنون پنج گزینهٔ متفـاوت در ایـن فیلـد تعریـف شـده اسـت:

| Option Class | Option Number | Name Of Options | شرح |
|---|---|---|---|
| 00 | 0 | End of Options List | **۱-تعیین پایـان لیسـت گزینـه‌ها** |
| 00 | 1 | Null Option | **۲-گزینهٔ پوچ (فقط بـرای پـر کـردن فضـا )** |
| 00 | 2 | Security | **۳-گزینهٔ امنیـت** |
| 00 | 3 | Loose Source Routing | **۴-گزینهٔ تعیین مسـیر بصـورت نـاقص** |
| 00 | 7 | Record Route | **۵-گزینهٔ ثبت مسـیر** |
| 00 | 9 | Strict Source Routing | **۶-گزینهٔ تعیین مسـیر بصـورت دقیـق و صریـح** |
| 10 | 4 | Timestamp | **۷-گزینهٔ ثبت مسـیر و زمـان** |

**گزینهٔ اوّل** : با این گزینـه پایـان مجموعـهٔ گزینـه‌ها مشـخص می‌شـود.

**گزینـهٔ دوم** : ایـن گزینـه هیـچ ارزش اجرایـی نـدارد و فقـط بـرای آنسـت کـه فضـــای فیلد Options به گونه ای پر شـود تـا ضریبـی از ٤ بـاقی بمـاند.

**گزینـهٔ سـوم** : مشـخص می‌کنـد کـه بسـته IP تـا چـه حـدّ محرمانـه اســت و در ایـــن شـرایط مسـیریاب خواهـد دانسـت کـه ایـن بسـته را از طریـق چـه مسـیرهائی بـــه ســـمت مقصد هدایت نماید تـا امنیـت بسـته تـامین شـود و از چـه مسـیرهائی بـاید احـتراز نمـاید . اکثر مسیریابهای تجـاری از ایـن گزینـه چشمپوشـی می‌نمـایند .

**گزینه چـهارم**: بـا ایـن گزینـه می‌تـوان مسـیری را بـرای عبـور بسـته (بصـورت نـاقص) مشـخص کـرد و بسـته بـاید قطعـاً از مسـیریابهای مشـخص شـده عبـور نمـاید ولـــی از آن جائی کـه ایـن گزینـه مسـیر کـامل را مشـخص نکـرده اسـت بقیـه مسـیر توسـط مسـیریاب تعیین میشود . بـــرای مثـال فـرض کنیـد بخواهیـد بسـته ای را کـه بـاید از لنـدن بـه سـیدنی طی مسـیر کنـد ، بجـای عبـور از شـرق بـه غـرب از مسـیرهای نیویـورک ، لـس آنجلـس و هانولولو به سـمت سـیدنی ارسـال شـود . کـافی اسـت فقـط آدرس مسـیرهای ابتدائـی را بـا این گزینه مشخص کرده و بقیـه مسـیر بعـهده مسـیریابها گذاشـته شـود.

**گزینه پنجم:** با درج این گزینه در بستهٔ IP از تمامی مسیریابها خواسته می‌شود که قبل از ارسال بسته به مسیریاب بعدی آدرس خودشان را در فیلد Option ثبت نمایند . با بررسی مسیرهائی که یک بسته از مبداء به سمت مقصد پیموده است می‌توان به اشکالات احتمالی در الگوریتمهای مسیریابی هر مسیریاب پی برد. دقّت کنید که پروتکل IP زمانی وضع شده است که فضای ٤٠ بایتی فیلد Options برای تمامی شبکه‌ها کافی بود؛ چرا که این پروتکل برای اولین بار در ARPANET پیاده شد که حداکثر تعداد مسیریابها در طولانیترین مسیر ، ۹ عدد بود . بنابراین فضای چهل بایتی برای امروزه که هزاران مسیریاب در جهان نصب و راه‌اندازی شده است بسیار ناکافی به نظر می‌رسد.

**گزینه ششم** : با این گزینه می‌توان مسیر از پیش تعیین شده ای را برای بستهٔ IP تعیین کرد و مسیریابها نیز موظفند از مسیر تعیین شده تبعیت نمایند. با توجه به آنکه در زیرشبکه ، مسیریابی به روشهای پویا انجام می‌شود ، استفاده از این گزینه چندان منطقی و مناسب به نظر نمی‌رسد بلکه فقط بعنوان یک ابزار برای مدیران سیستم جهت آزمایش و بررسی شرایط یک مسیر و تخمین جداول مسیریابی ( بصورت دستی) مفید خواهد بود.

**گزینه هفتم:** این گزینه از تمامی مسیریابها می‌خواهد که زمان دریافت بسته را در فیلد Options درج کنند. این گزینه برای اشکال زدائی از الگوریتمهای مسیریابی مناسب است.

- فیلد **Payload** : در این فیلد داده‌های دریافتی از لایهٔ بالاتر قرار می‌گیرد.

پس از شناسائی ساختار یک بسته IP ، بایستی به مبحث آدرسها در پروتکل IP بپردازیم . مفاهیم آدرسهای IP شما را در درک واقعیت چگونگی مسیریابی بهتر کمک می‌کند . سپس به پروتکلهائی خواهیم پرداخت که به پروتکلIP در لایه شبکه کمک می‌کنند تا یک مسیریابی صحیح امکان پذیر باشد .

## ۳) مبحث آدرسها در اینترنت و اینترانت

همانگونه که در مباحث قبلی بدان اشاره کردیم پروتکل اینترنت در ارتباطات بین شبکه ای[1] از آدرسهای منحصر به فرد و یکتای ۳۲ بیتی بهره می‌برد. (هر چند که در نسل بعدیِ پروتکل اینترنت که تا سال ۲۰۰۵ همه گیر خواهد شد این آدرسها ۱۲۸ بیتی می‌شوند.) هر ابزار شبکه اعم از ماشینهای میزبان ، مسیریابها و چاپگرهای شبکه در اینترنت با یک آدرس IP شناسائی می‌شوند.

در ادامهٔ این فصل باید موارد زیر را بررسی و مطالعه کنیم:

- قالب هر آدرس IP چگونه سازماندهی می‌شود؟
- کلاسهای مختلف آدرسهای IP به چه منظور و چگونه سازماندهی می‌شوند؟
- چگونه آدرسهای IP به آدرسهای سخت افزاری لایه فیزیکی تبدیل خواهد شد و قراردادهای نمایش آدرسهای IP چگونه هستند؟
- یک مسیریاب چگونه می‌تواند از یک آدرس چهاربایتی ، محل دقیق یک ماشین را بین دهها میلیون ماشین متصّل به شبکه پیدا نماید؟

آدرسهای IP درون یک عدد دودویی ۳۲ بیتی درج می‌شوند ولیکن برای سادگی نمایش به چهار بایت تقسیم شده و بصورت چهار عدد دهدهی که با نقطه از هم جدا شده‌اند نوشته می‌شود؛ یعنی معادل دهدهی هر یک از بایتهای آدرس بصورت مجزا نوشته شده و هر عدد با یک علامتِ . از دیگری تفکیک می‌شود. بعنوان مثال آدرس زیر یک آدرس IP معتبر می‌باشد که در قالب چهار قسمت دهدهی نوشته شده است:

34.21.225.1

این آدرس بصورت زیر در فیلد آدرس از یک بستهٔ IP تنظیم میشود:


پرارزشترین بایت یعنی اولین بایتِ سمت چپ از آدرس IP ، کلاس آدرس را مشخص می‌کند و از این رو دارای اهمیت ویژه است. ولی قبل از آنکه کلاسهای آدرس را تشریح نماییم بازهم روی این نکته تکیه می‌کنیم که وقتی یک ماشین میزبان به شبکه اینترنت متصل می‌شود بایستی آدرس IP آن منحصر به فرد و یکتا[2] باشد . در

---

[1] Internetworking
[2] Unique

در حقیقت هر ماشین روی شبکه با یک آدرس یکتا هویت پیدا میکند . برای اطمینان از یکتا بودن آدرسهای IP برای ارتباطات عمومی، مرکز InterNIC[1] کنترل و نظارت بر روی آدرسهای IP را بر عهده گرفته است.

IANA[2] قدرت اجرائی برای اختصاص آدرسهای IP منحصر به فرد را فراهم کرده است . هر چند شبکه‌های خصوصی که به اینترنت وصل نیستند می‌توانند از آدرسهای IP دلخواه استفاده کنند ولی اگر این شبکه‌ها زمانی بخواهند به اینترنت وصل شوند دوگانگی آدرسهای غیر یکتا و نهایتاً تناقض و اشکال در مسیریابی[3] رخ خواهد داد؛ به همین دلیل پیشنهاد شده است که حتی شبکه‌های خصوصی نیز برای اختصاص آدرس به ماشینهای میزبان از مرکز InterNIC مجوز بگیرند و از آدرسهای معتبر و اختصاصی استفاده کنند.

### ۱-۳) کلاسهای آدرسِ IP

از آنجا که TCP/IP برای شبکه‌های با مقیاس بزرگ طراحی شده است لذا نمی‌توان انتظار داشت که فضای ۳۲ بیتی آدرس که حدود چهار میلیارد و سیصد میلیون (4,294,967,295) آدرس را در اختیار می‌گذارد ، بدون هیچ نظم و سیاق خاص به ماشینهای شبکه اختصاص داده شود . این کار همانند آن خواهد بود که تمامی آپارتمانها و منازل در کل جهان با شماره‌های ده رقمی مشخص شود بدون آنکه هیچ ضابطه ای در شماره گذاری آنها رعایت شده باشد . آنگاه منزلی با شماره ۱۰٦٥٤۳۲۳۹۰ چگونه پیدا می‌شود !

آدرسهای پستی ساختاری سلسله مراتبی به صورت زیر دارند، به گونه ای که هر منزل در هر کجای دنیا قابل آدرس دهی است و به راحتی پیدا می‌شود:

شماره/کوچه/خیابان/ناحیه/شهر/کشور

فلسفهٔ کلاسهای آدرس IP به همین منظور است :

آدرس ماشین/آدرس زیرشبکه/آدرس شبکه

---

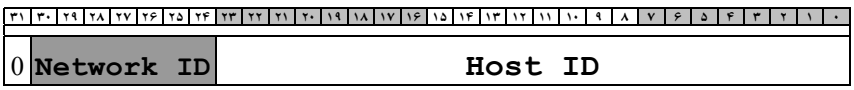
[1] Internet Network Information Center
[2] Internet Assigned Number Authority
[3] Conflict

با توجه به آنکه اینترنت مجموعه‌ای از شبکه‌های متصل شده به هم است برای آدرس دادن به ماشینهای میزبان بهتر است ۳۲ بیت آدرسِ IP به قسمتهای زیر تقسیم شود:

الف) آدرس شبکه
ب) آدرس زیر شبکه (درصورت لزوم)
ج) آدرس ماشین میزبان

آدرسهای IP درپنج کلاس E,D,C,B,A معرفی شده‌اند که شما بایستی آنها را بدقت بشناسید و تحلیل کنید. در زیر قالب کلاسهای پنج گانهٔ آدرس IP مشخص شده است:

- **آدرسهای کلاس A** : قالب ۳۲ بیتیِ آدرس در کلاس A به صورت زیر است:




در کلاس A ، پرارزشترین بیت از آدرس ، مقدار صفر دارد و این بیت کلاس A را از دیگر کلاسها متمایز می‌کند؛ ۷ بیت بعدی "مشخصهٔ آدرس شبکه" و سه بایت باقیمانده ، آدرس ماشین میزبان را تعیین می‌کند. بنابراین در کلاس A بایت پرارزش در محدوده صفر تا ۱۲۷ تغییر می‌کند. چون با ۲٤ بیت می‌توان حدود هفده میلیون ماشین میزبان را آدرس دهی کرد ، می‌توان به این نتیجه رسید که آدرسهای کلاس A بایستی برای آژانسهای ستون فقرات اینترنت یا شبکه‌ها بسیار عظیم مثل NSFNet یا ARPANet اختصاص داده شده باشد . مشخصهٔ شبکه در این کلاس بهیچوجه نمی‌تواند اعداد صفر یا ۱۲۷ انتخاب شود چرا که این دو عدد در شبکه معنای دیگری خواهند داشت و بعداً به آن اشاره خواهیم کرد. بنابراین تعداد شبکه‌هائی که در جهان می‌توانند از کلاس A استفاده کنند ۱۲٦ تا خواهد شد که بسیار کم است . امروزه اختصاص آدرسهای کلاس A غیر ممکن است چرا که همهٔ آنها توسط پیشگامان شبکه سالها قبل تملیک شده‌اند.

وقتی به یک آدرس IP که در قالب دهدهی نوشته شده است نگاه می‌کنید براحتی می‌توانید کلاس آنرا تشخیص بدهید . اگر عدد سمت چپ آدرس ، بین صفر تا ۱۲۷ باشد ، آن آدرس از کلاس A خواهد بود:

74. 103 .14 .138

Net ID  Host ID

آدرس IP (127.0.0.0) در پروتکل اینترنت ، یک شبکه را تعیین نمی‌کند بلکه بصورت قراردادی بعنوان آدرس "حلقه بازگشت"[1] جهت اهداف اشکال زدایی استفاده شده است چرا که این آدرس عملاً معادل آدرس خود ماشین محلی است .

- **آدرسهای کلاس B**: قالب ۳۲ بیتیِ آدرس در کلاس **B** به صورت زیر است:


هر گاه دو بیت پرارزش از آدرس IP مقدار 10 داشته باشد آن آدرس از کلاس B خواهد بود . ١٤ بیت باقیمانده از ۲ بایت سمت چپ ، آدرس شبکه را تعیین می‌کند و دو بایت اول از سمت راست (١٦ بیت) آدرس ماشین میزبان خواهد بود .

در آدرسهای کلاس B ، تعداد ١٦٣٨٢ ($2^{14}$-2) شبکهٔ گوناگون قابل تعریف خواهد بود و هر شبکه می‌تواند ٦٥٥٣٤ ($2^{16}$-2) ماشین میزبان تعریف نماید. اختصاص آدرسهای کلاس B برای شبکه‌های بسیار عظیم مناسب است . هر چند تعداد این شبکه در جهان می‌تواند تا حدود شانزده هزار عدد باشد ولیکن امروزه عملاً نمی‌توان آدرس کلاس B گرفت چرا که تقریباً همهٔ آنها آن تخصیص داده شده‌اند .

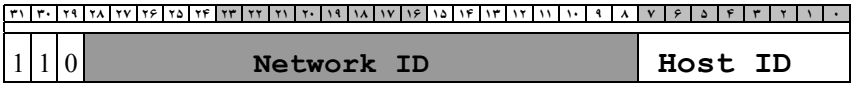
اگر آدرس IP به صورت دهدهی نوشته شود و عدد سمت چپِ آن بین ۱۲۸ تا ۱۹۱ باشد ، آن آدرس ، کلاس B خواهد بود:

134. 64. 143 . 24

Net ID Host ID

[1] Loopback

➢ **آدرس کلاس C** : قالب ۳۲ بیتیِ آدرس در کلاس **C** به صورت زیر است:




کلاس C مناسب ترین و پرکاربردترین کلاس از آدرس‌های IP است . همانگونه که از شکل مشخص است در این کلاس ، سه بیت پرارزش دارای مقدار 110 است و ۲۱ بیت بعدی از سه بایت سمت چپ برای تعیین آدرس شبکهٔ مورد نظر بکار رفته است . بنابراین در این کلاس می‌توان حدود دو میلیون شبکه را در جهان آدرس دهی کرد و هر شبکه می‌تواند تا ۲۵٤ عدد ماشین میزبان تعریف نماید . برای تشخیص آدرس‌های کلاس C به عدد سمت چپ از آدرس IP که به صورت دهدهی نوشته شده است نگاه کنید. اگر عدد بین ۱۹۲ تا ۲۲۳ بود آن آدرس از کلاس C خواهد بود:

**199.164. 78 .132**

**Net ID  Host ID**

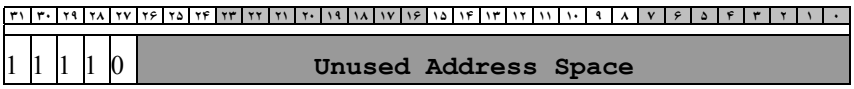
➢ **آدرس کلاس D** : قالب ۳۲ بیتیِ آدرس در کلاس **D** به صورت زیر است:


در این کلاس ، چهار بیت پرارزش دارای مقدار 1110 است و ۲۸ بیت باقیمانده از کلّ آدرس برای تعیین آدرسهای "چند مقصده"[1] (آدرسهای گروهی) است .

از این آدرسها برای ارسال یک دیتاگرام به طور همزمان برای چندین ماشین میزبان کاربرد دارد و بمنظور عملیات رسانه ای و چند پخشی بکار می‌رود. توضیح بیشتر در مورد این کلاس در بخشی مجزا ارائه خواهد شد .

[1] Multicast

➢ **آدرس کلاس E** : قالب ۳۲ بیتیِ آدرس در کلاس **E** به صورت زیر است:




فعلاً این دسته از آدرسها که پنج بیت پرارزش آنها در سمت چپ 11110 است کاربرد خاصّی ندارند و برای استفاده در آینده بدون استفاده رها شده‌اند. البته گاهی بصورت آزمایشی از این آدرسها استفاده شد ولی تاکنون جهانی نشده‌اند .

### ۲-۳) آدرسهای خاصّ

در بین تمامی کلاسهای آدرسIP پنج گروه از آدرسها ، معنای ویژه ای دارند و با آنها نمی‌توان یک شبکهٔ خاص را تعریف و آدرس دهی کرد. این پنج گروه آدرس عبارتند از:

الف) آدرس 0.0.0.0 : هر ماشین میزبان که از آدرس IP خودش مطلع نیست این آدرس را بعنوان آدرس خودش فرض می‌کند. البته از این آدرس فقط به عنوان آدرس مبداء و برای ارسال یک بسته می‌توان استفاده کرد و گیرندهٔ بسته نمی‌تواند پاسخی به مبداء بسته برگرداند.[۱]

ب) آدرس 0.HostID زمانی به کار می‌رود که ماشین میزبان ، آدرس مشخصهٔ شبکه ای که بدان متعلق است را نداند. در این حالت در قسمت NetID مقدار صفر و در قسمت HostID شمارهٔ مشخصهٔ خود را قرار می‌دهد.

ج) آدرس 255.255.255.255 : برای ارسال پیامهای فراگیر برای تمامی ماشینهای میزبان بر روی شبکهٔ محلی که ماشین ارسال کننده به آن متعلق است .

د) آدرس NetID.255 : برای ارسال پیامهای فراگیر برای تمامی ماشینهای یک شبکهٔ راه دور که ماشین میزبان فعلی متعلق به آن نیست. آدرس شبکهٔ مورد نظر در قسمت NetID تعیین شده و تمامی بیتهای قسمت مشخصهٔ ماشین میزبان ۱ قرار داده می‌شود. البته بسیاری از مسیریابها برای مصون ماندن شبکه از مزاحمتهای بیرونی ، چنین بسته‌هایی را حذف می‌کنند.

---

[۱] **استفاده از این آدرس مانند آنست که در آدرس فرستندهٔ یک بستهٔ پستی نوشته شود : "خودم" . بسته می‌تواند به مقصد برسد ولی پاسخی نخواهد داشت.**

ه) 127.xx.yy.zz بعنوان "آدرس بازگشت" شناخته می‌شود و آدرس بسیار مفیدی برای اشکالزدایی از نرم افزار می‌باشد. به عنوان مثال اگر بسته ای به آدرس 127.0.0.1 ارسال شود ، بسته برای ماشین تولیدکنندهٔ آن بر خواهد گشت[1]؛ در این حالت اگر نرم افزارهای TCP/IP درست و بدون اشکال نصب شده باشد فرستنده بسته باید آنرا مجدداً دیافت کند. همچنین از این آدرس می‌توان برای آزمایش برنامه‌های تحت شبکه ، قبل از نصب آنها بر روی ماشینهای میزبان استفاده کرد.

### ۳-۳) آدرسهای زیرشبکه

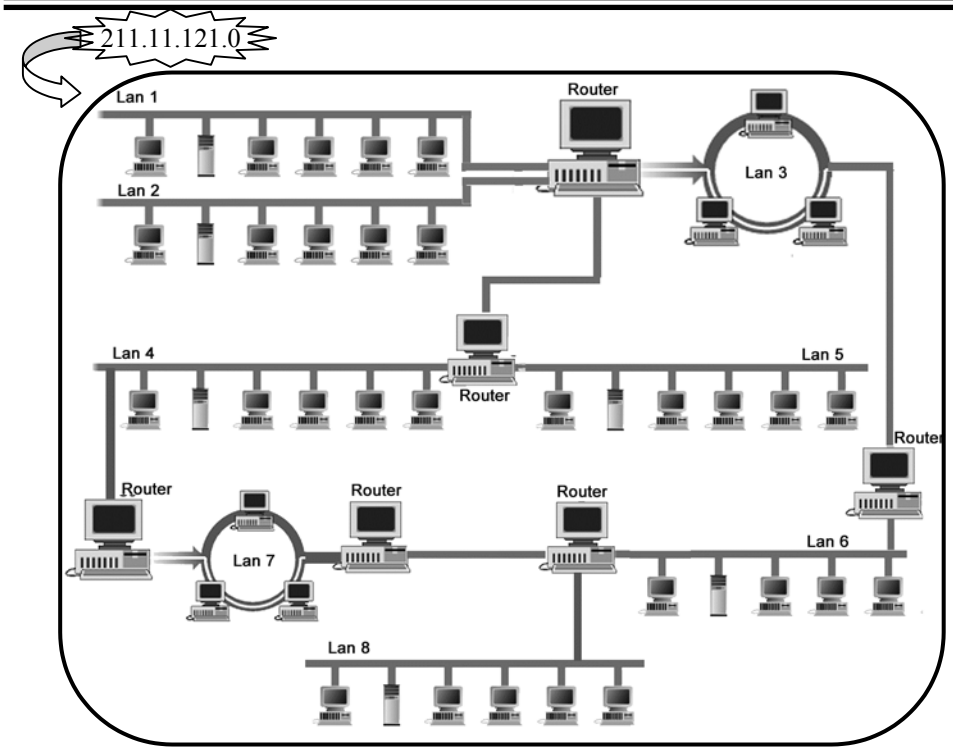
در ادامهٔ بحث بایستی مسئله زیر شبکه را در خصوص آدرس دهی‌ها مطرح نمائیم . مبحث را با یک مثال آغاز می‌نمائیم:

فرض کنید دانشگاه شما یک کلاس C با قابلیت تعریف ۲۵٤ ماشین میزبان ثبت می‌نماید (مثلاً 211.11.121.0) ؛ یعنی شبکهٔ دانشگاه توانایی آدرس دهی ۲۵۵ ایستگاه را در شبکه دارد . در نظر بگیرید که دانشگاه دارای یک شبکهٔ محلی واحد و یکپارچه برای کل دانشگاه نیست بلکه دارای هشت شبکهٔ محلی مجزا است که برای هر دانشکده تهیه دیده شده است؛ ( همانند ساختاری که در شکل (۸-۳) ترسیم شده است). هر کدام از این شبکه‌ها که می‌تواند توپولوژی متفاوتی داشته باشد ، از طریق مسیریاب به هم متصل شده‌اند و طبعاً برای ارتباط بین شبکه‌های هر دانشکده باید مسیریابی صورت گیرد. از دیدگاه بیرونی کلّ مجموعهٔ شبکه‌های محلی دانشگاه با یک آدرس مشخصه یعنی 211.11.121.0 شناخته میشود و مسیریابهای بیرونی هیچ شناختی از ساختار شبکه بندی داخلی دانشگاه ندارند. ( هر یک از شبکه‌های محلی داخل دانشگاه یک زیرشبکه نامیده می‌شود. ) بنابراین باید روشی وجود داشته باشد تا از طریق آدرسهای کلاس C ( یا هر کلاس دیگر ) بتوان زیر شبکه‌ها را نیز مشخص کرد تا مسیریابهای داخلی نیز قادر باشند زیر شبکه‌های مختلف را شناسایی و تفکیک کنند.

این مسئله برای آدرسهای کلاس B و A بسیار ضروری و اجتناب ناپذیر می‌نماید چرا که نمی‌توان انتظار داشت که یک موسسه که آدرس کلاس B با قابلیت تعریف حدود ٦٦ هزار ماشین میزبان ثبت کرده است فقط یک شبکهٔ یکپارچه داشته باشد بلکه چنین موسسه ای ممکن است دارای صدها زیر شبکهٔ کوچک و بزرگ باشد.

---

[1] **این آدرس همانند آنست که فرستندهٔ یک بستهٔ پستی آدرس دقیق خودش را به عنوان گیرندهٔ آن درج نماید. بنابراین با آدرس 0.0.0.0 تفاوت ذاتی دارد.**




شکل (۸-۳) یک شبکهٔ خود مختار که کلاً با یک آدرس مشخصهٔ شبکه شناسایی میشود.

برای آنکه بتوان زیرشبکه‌ها[۱] را تفکیک کرد جدای از قسمت آدرس شبکه که کل شبکهٔ دانشگاه شما را مشخص می‌کند بایستی در قسمت مشخصهٔ ماشین میزبان نیز به گونه ای زیر شبکه‌ها مشخص شوند . این کار از طریق مفهومی به نام "الگوی زیرشبکه"[۲] انجام میشود.

شما با نگاه اول به اولین عدد سمت چپ متوجه خواهید شد که این آدرس از چه کلاسی است ولی هنوز موارد مبهمی وجود دارد : آیا شبکه ای که آدرس آنرا پیش رو دارید فقط یک شبکه است یا خودش زیر شبکه بندی شده است؛ یعنی از چند شبکهٔ محلی متصل بهم تشکیل شده است؟

---

[۱] Subnetworks
[۲] Subnet Mask

این اطلاعات برای شبکه‌های مبتنی بر TCP/IP که قابلیت مسیریابی دارند بسیار مهم است ، چرا که هر ماشینِ میزبان بایستی قادر به درک این مطلب باشد که آیا یک ماشین مقصد با آدرس خاصّ و مشخص ، بر روی شبکهٔ محلی خودش واقع است یا آنکه آن آدرس متعلق به زیر شبکه دیگری است . بر اساس این اطلاعات ماشین میزبان تصمیم می‌گیرد که آیا انتقال اطلاعات باید مستقیماً بر روی شبکهٔ محلی انجام شود یا آنکه باید از طریق یک مسیریاب روی شبکه ای دیگر ارسال شود .

تمامی ماشینهای میزبان برای تشخیص محل مقصد یک بستهٔ IP در شبکه احتیاج به یک مشخصهٔ دیگر دارند و آن "الگوی زیرشبکه" نامیده می‌شود.

الگوی زیرشبکه یک عدد ۳۲ بیتی دودویی است که برای ماشین میزبان نقش یک مقایسه‌گر را بازی می‌کند تا با استفاده از آن بتواند تشخیص دهد که آیا مقصد روی همین شبکه محلی است که خودش به آن تعلق دارد یا روی شبکه دیگری است .

فرآیند استفاده از "الگوی زیرشبکه" را با استفاده از مثال قبل ولی با آدرس کلاس B آموزش می‌دهیم:

فرض کنید شما کاربری روی یک ایستگاه در شبکه دانشگاه خودتان هستید, آدرس IP متعلق به دستگاه شما بصورت زیر اختصاص داده شده است :

131.55.213.73

با یک نگاه متوجه می‌شوید که آدرس از کلاس B است که مشخصهٔ شبکهٔ آن معادل 131.55.0.0 و مشخصهٔ ماشین شما 0.0.213.73 است؛ ولی هنوز نمی‌دانید شبکه ای که مشخصهٔ آن معادل 131.55 است آیا زیر شبکه دارد یا خیر؟

فرض کنید که دانشگاه شما با آدرس شبکه 131.55.0.0 ، می‌خواهد حداکثر دارای ۲۵٤ زیر شبکه باشد ، بهمین دلیل فرض کرده است که در فیلد مشخصهٔ ماشین میزبان (Host ID ) که در کلاس B دو بایت سمت راست را شامل میشود ، بایت دومِ آن به عنوان مشخصهٔ مربوط به زیر شبکه تعریف شود. یعنی فیلد دوبایتی مربوط به مشخصهٔ ماشین میزبان به دو بخش تقسیم شده است:

الف) مشخصهٔ زیرشبکه ب) مشخصهٔ ماشین میزبان

| ۳۱ | ۳۰ | ۲۹ | ۲۸ | ۲۷ | ۲۶ | ۲۵ | ۲۴ | ۲۳ | ۲۲ | ۲۱ | ۲۰ | ۱۹ | ۱۸ | ۱۷ | ۱۶ | ۱۵ | ۱۴ | ۱۳ | ۱۲ | ۱۱ | ۱۰ | ۹ | ۸ | ۷ | ۶ | ۵ | ۴ | ۳ | ۲ | ۱ | ۰ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | Network ID | | | | | | | | | | | | | | Subnet ID | | | | | | | | Host ID | | | | | | | |

ماشین شما تصمیم دارد بسته ای را برای یک ماشین میزبان با آدرس IP معادل 131.55.108.75 بفرستد؛ ماشین از کجا می‌تواند بفهمد که مقصد روی همین شبکه محلی که شما بدان متعلق هستید واقع است یا آنکه به شبکهٔ محلی در یک دانشکدهٔ دیگر متعلق است . دانستن این موضوع بسیار با اهمیت خواهد بود چرا که اگر ماشین میزبان مورد نظر روی شبکهٔ دیگری باشد بسته باید با آدرس فیزیکی "مسیریاب پیش فرض"[1] روی کانال ارسال شود. بنابراین تمام ماشینهای روی شبکه بایستی از وضعیت زیر شبکه‌ها مطلع باشند .

با توجه به آنچه که در بالا اشاره شد دومین بایت از سمت راست بعنوان مشخصهٔ زیر شبکه اختصاص داده شده است و بهمین دلیل هر ماشین برای دانستن آنکه آیا ماشین مقصد در شبکه محلی خودش واقع است یا در خارج از شبکه قرار دارد باید قسمت "مشخصهٔ شبکه" و "مشخصهٔ زیرشبکه" از آدرس IP خودش را با همین مشخصه‌ها از آدرس مقصد مقایسه نماید.

اینجاست که یک الگوی ۳۲ بیتی تعریف می‌شود که یک عدد ۳۲ بیتی و در این مثال بصورت 255.255.255.0 است:

| ۳۱ | ۳۰ | ۲۹ | ۲۸ | ۲۷ | ۲۶ | ۲۵ | ۲۴ | ۲۳ | ۲۲ | ۲۱ | ۲۰ | ۱۹ | ۱۸ | ۱۷ | ۱۶ | ۱۵ | ۱۴ | ۱۳ | ۱۲ | ۱۱ | ۱۰ | ۹ | ۸ | ۷ | ۶ | ۵ | ۴ | ۳ | ۲ | ۱ | ۰ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۰ | ۰ | ۰ | ۰ | ۰ | ۰ | ۰ | ۰ |

هر گاه ماشین بخواهدیک آدرس IP را تحلیل کند . الگوی فوق را با آدرس IP خودش AND می‌کند. ( با اینکار در حقیقت Host ID خودش را صفر می‌نماید ) سپس مجدداً الگو را با آدرس IP مقصد AND می‌کند (مشخصهٔ ماشین مقصد هم صفر می‌شود) حال نتیجه دو مرحله را با هم مقایسه می‌نماید . اگر نتیجه دو مرحله یکسان بود ، هم مشخصهٔ شبکه و هم مشخصهٔ زیرشبکه از آدرسهای مبدأ و مقصد یکی است و هر دو روی یک شبکهٔ محلی قرار دارند. در صورت عدم تساوی ، ماشین مبدأ به این نتیجه می‌رسد که مقصد مورد نظر روی شبکه محلی خودش نیست و آن بسته بایستی به آدرس فیزیکی مسیریابِ پیش فرض ارسال شود.

فرض کنید بسته ای با آدرسهای مشخص زیر بخواهد ارسال شود:

آدرس ماشین مبدأ: 131.55.213.73

[1] Default Gateway

آدرس ماشین مقصد: 131.55.108.75

الگوی زیرشبکه: 255.255.255.0

آدرس ماشین مبدأ در قالب دودویی


AND

الگوی زیرشبکه در قالب دودویی


حاصل مرحلهٔ ۱ :


آدرس ماشین مقصد در قالب دودویی


AND

الگوی زیرشبکه در قالب دودویی


حاصل مرحلهٔ ۲ :


حاصل مراحل ۱ و ۲ با هم مساوی نیستند و بنابراین ماشین مبدأ متوجه خواهد شد که ماشین مقصد روی شبکه محلی خودش نیست و بسته اطلاعاتی را بایستی به آدرس فیزیکی مسیریاب پیش فرض ارسال نماید.

به عنوان مثالی دیگر فرض کنید ماشین شما می‌خواهد برای ماشینی با آدرس IP زیر بسته ای را ارسال نماید:

131.55.213.84

آدرس ماشین مبدأ در قالب دودویی


AND

الگوی زیرشبکه در قالب دودویی


حاصل مرحلهٔ ۱ :


آدرس ماشین مقصد در قالب دودویی


AND

الگوی زیرشبکه در قالب دودویی


حاصل مرحلهٔ ۲ :


همانگونه که مشاهده میشود حاصل مراحل ۱ و ۲ مساوی هستند و بالطبع مبدأ و مقصد روی یک شبکه محلی واقعند و هیچ لزومی ندارد که ارسال بسته به آدرس فیزیکی مسیریاب پیش فرض انجام شود ، بلکه باید مستقیماً از آدرس فیزیکی ایستگاه مقصد استفاده شود .

ذکر این نکته ضروری است که الگوی زیرشبکه باید به عنوان یکی از پارامترهای پیکربندی TCP/IP تنظیم شود و فقط برای تشخیص محل شبکهٔ مقصد کاربرد دارد.

الگوی زیرشبکه در مثالهای بالا ساده ترین حالت بود که به آنها "الگوی زیرشبکهٔ استاندارد"[1] گفته می‌شود چرا که الگوها دقیقاً هشت بیتی هستند .

[1] Standard Subnet Mask

## ۴) زیر شبکه‌های غیر استاندارد

الگوهای زیرشبکه برای تقسیم فضای آدرس‌دهی در شبکه‌های کلاس A ، B و C به تعدادی زیرشبکه ، تعریف می‌شوند. در مثالهائی که بررسی کردیم الگوی شبکه بصورت زیر تعریف شده بود:

255.255.255.0

حال الگوی زیر را در نظر بگیرید:

255.255.240.0

عدد ۲٤۰ در الگوی زیرشبکه چه چیزی را تعریف می‌کند؟
به فرم دودویی الگوی بالا دقت کنید:

| ۳۱ | ۳۰ | ۲۹ | ۲۸ | ۲۷ | ۲۶ | ۲۵ | ۲۴ | ۲۳ | ۲۲ | ۲۱ | ۲۰ | ۱۹ | ۱۸ | ۱۷ | ۱۶ | ۱۵ | ۱۴ | ۱۳ | ۱۲ | ۱۱ | ۱۰ | ۹ | ۸ | ۷ | ۶ | ۵ | ۴ | ۳ | ۲ | ۱ | ۰ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۰ | ۰ | ۰ | ۰ | ۰ | ۰ | ۰ | ۰ | ۰ | ۰ | ۰ | ۰ |

بفرض اگر الگوی بالا برای زیر شبکه بندی آدرس کلاس B به کار رفته باشد ، نشان دهنده آن است که چهار بیت پرارزش از بایت دوم برای تعیین شماره زیر شبکه به کار رفته است و ۱۲ بیت باقیمانده بعنوان "مشخصهٔ ماشین میزبان" استفاده شده است؛ بدین معنا که با این الگو می‌توان ١٤ زیر شبکه ($2^4$-2) تعریف کرد بگونه ای که در هر زیرشبکه ٤۰۹٤ ، ($2^{12}$-2) ماشینِ میزبان قابل آدرس دهی خواهد بود.

فراموش نکنید که همیشه تعداد زیرشبکه‌ها و ماشینهای میزبان از کلّ تعداد قابل تعریف ، دوتا کمتر است؛ چراکه زیرشبکه یا ماشینی که تمام بیتهای آن صفر یا تماماً یک باشد قابل تعریف نیست.

کلاً برای آنکه با تنظیم الگوهای زیر شبکه آشنا شوید الگوریتم آنرا در زیر تشریح می‌کنیم:
الف ) با دقت و دوراندیشی کافی تعیین کنید چه تعداد زیر شبکه و چه تعداد ماشین میزبان روی هر زیرشبکه خواهید داشت. به تعداد زیرشبکه‌ها و ماشینها عدد ۲ را اضافه کرده و سپس تعیین کنید هر کدام از این اعداد به حداقل چند بیت نیاز دارند.
ب ) الگوی ۳۲ بیتی زیرشبکه را بگونه ای تنظیم کنید که در سمت راست آن به تعداد بیتی که برای آدرس دهی ماشینهای میزبان نیاز است صفر قرار بگیرد . بیتهای باقیمانده را هم تماماً ۱ قرار بدهید.
ج) الگوی دودویی را بفرم دهدهی نقطه دار تبدیل کنید.
د ) زیرشبکه‌ها و ماشینهای میزبان روی هر زیرشبکه را تعریف نمائید.

حال با مثالی این روند را بهتر بررسی می‌کنیم:

فرض کنید یک شرکت بزرگ دارای حداکثر ۲۵ شبکه محلی است که هر شبکه محلی حداکثر تا ۱۰۰۰ میزبان را حمایت می‌نماید . الگوی زیر شبکه را تنظیم نمائید.

الف ) برای آدرس دهیِ ۲۵ زیرشبکهٔ محلی ۵ بیت کفایت می‌نماید . (با ۵ بیت می‌توان سی زیرشبکه (2-32) را تعریف کرد )

ب ) آدرس از کلاس B است پس قسمت ۱۶ بیتی از مشخصهٔ ماشین میزبان ، پس از کسر ۵ بیت که بعنوان زیر شبکه استفاده شد مقدار ۱۱ بیت خواهد بود. حداکثر تعداد ماشین میزبان قابل تعریف بصورت زیر حساب می‌شود:

$$2^{11}-2=2046$$

تعداد ماشین قابل تعریف از تعدادی که نیاز داشتیم خیلی بیشتر است که اجازه می‌دهد زیرشبکه در آینده گسترش یابد . پس الگوی زیرشبکه برای شبکهٔ این شرکت به صورت زیر تنظیم می‌شود:

| ۳۱ | ۳۰ | ۲۹ | ۲۸ | ۲۷ | ۲۶ | ۲۵ | ۲۴ | ۲۳ | ۲۲ | ۲۱ | ۲۰ | ۱۹ | ۱۸ | ۱۷ | ۱۶ | ۱۵ | ۱۴ | ۱۳ | ۱۲ | ۱۱ | ۱۰ | ۹ | ۸ | ۷ | ۶ | ۵ | ۴ | ۳ | ۲ | ۱ | ۰ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۱ | ۰ | ۰ | ۰ | ۰ | ۰ | ۰ | ۰ | ۰ | ۰ | ۰ | ۰ |

**الگوی زیرشبکه 255.255.248.0**

دقت کنید که اگر دور اندیشی کافی نداشته باشید و نیاز باشد که زمانی ساختار زیر شبکه را توسعه بدهید، برای تغییر یک الگوی زیر شبکه به الگوی دیگر ، باید پیکربندی[۱] تمام ماشینهای شبکه بصورت دستی تنظیم شود که دردسر زیادی دارد. لذا از میزان رشد آدرسهای زیرشبکهٔ خود اطمینان حاصل کنید و طرح را مطابق با نیازتان بریزید. هرگونه اشتباه در تنظیم الگوی زیرشبکه منجر به عدم کارکرد صحیح شبکه خواهد شد.

بگونه ای که اشاره شد وقتی تعداد زیر شبکه‌ها را حساب می‌نمائید نهایتاً عدد ۲ را با آن جمع کنید و سپس تعداد بیتهای مورد نیاز را برای عدد بدست آمده محاسبه نمائید. مثلاً فرض کنید که کل شبکه دقیقاً چهار زیرشبکه داشته باشد؛ پس برای الگوی زیرشبکه باید سه بیت اختصاص بدهید.

[۱] Configuration

## ۵) پروتکل ICMP[۱]

پروتکل IP ، پروتکلی "بدون اتصّال"[۲] و "غیر قابل اعتماد" [۳] است! بدون اتصّال بدین معنا که مسیریاب هر بسته را بدون هیچگونه هماهنگی با مقصد بسته یا مسیریاب بعدی ارسال می‌نماید ، بدون آنکه بتواند اطلاعی از وجود یا عدم وجود مقصد داشته باشد. در ضمن هر مسیریاب پس از ارسال یک بسته آنرا فراموش می‌کند و منتظر "پیام دریافت بسته"[٤] از گیرندهٔ آن نخواهد ماند. اگر یک بستهٔ IP با خطا به مقصد برسد و یا اصلا به مقصد نرسد این پروتکل هیچ اطلاعی در مورد سرنوشت آن به فرستندهٔ بسته نمی‌دهد.

دلایل مختلفی برای نرسیدن یک بسته به مقصد وجود دارد: ممکن است "زمان حیات[٥]" بسته قبل از رسیدن به مقصد منقضی شود؛ ممکن است مسیر یاب بسته را به مسیری اشتباه هدایت کند؛ ممکن است در هنگام قطعه قطعه کردن بسته و ارسال آنها ، یکی از قطعات دچار خطا شود یا به هر دلیلی به مقصد نرسد بنابراین کل دیتاگرام قابل بازسازی نخواهد بود؛ ممکن است مقصدِ بسته آمادگی دریافت بسته را نداشته باشد یا اصلاً وجود خارجی نداشته باشد. در هنگام بروز هرگونه خطا ، پروتکل IP به فرستندهٔ بسته هیچ اطلاعی در مورد سرنوشت آن نخواهد داد .

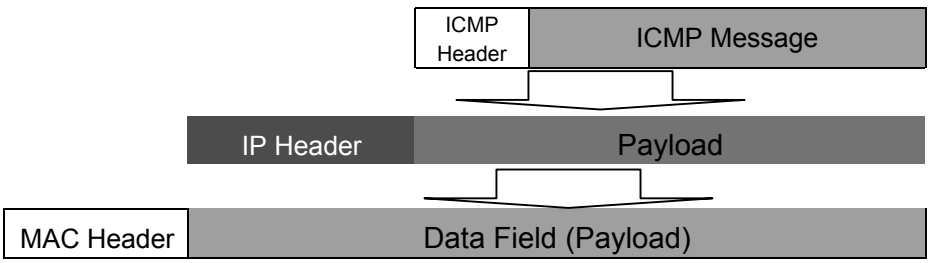
عدم گزارشِ خطا به تولید کنندهٔ یک بسته منجر به تکرار خطا و حمل بیهوده و زائد بسته‌هایی میشود که محکوم به فنا و حذف در شبکه هستند. به عنوان مثال عدم گزارش در مورد آماده نبودن مقصد برای دریافت بسته باعث خواهد شد که فرستندهٔ آن اقدام به ارسال بسته‌های دیگر کند در حالی که این کار بی ثمر خواهد بود و فقط بار ترافیک شبکه را افزایش می‌دهد و حتی می‌تواند منجر به بروز "ازدحام"[٦] شود.

پروتکلICMP در کنار پروتکل IP ، برای بررسی انواع خطا و ارسال پیام برای مبدأ بسته در هنگام بروز اشکالات ناخواسته استفاده می‌شود. در حقیقت ICMP یک سیستم گزارش خطا است که بر روی پروتکلIP نصب می‌شود تا در صورت بروز هرگونه خطا به فرستندهٔ بسته پیام مناسب را بدهد تا آن خطا تکرار نشود. در واقع ICMP وظیفه ای در قبال وقوع خطا ندارد بلکه فقط پیامی که بیانگر بروز خطا و نوع آن است به فرستنده برمیگرداند . این پروتکل اشکالات موجود را در قالب یکسری پیام گزارش می‌کند که این پیام خود در یک بسته IP قرار

---

[۱] Internet Control Message Protocol
[۲] Connectionless
[۳] Unreliable
[٤] Acknowledgement Message
[٥] Time To Live
[٦] Congestion

می‌گیرد که از جانب یک مسیریاب یا ماشین مقصد به آدرس فرستنده باز می‌گردد. در شکل (۹-۳) چگونگی قرار گرفتن یک پیام ICMP درون یک بستهٔ IP تصویر شده است.




شکل (۹-۳) چگونگی قرار گرفتن یک پیام ICMP درون یک بستهٔ IP

با توجه به آنکه پیام ICMP خود درون یک بسته IP جاسازی می‌شود بنابراین فیلد Protocol در سرآیند بستهٔ IP باید با شمارهٔ مشخصهٔ پروتکل ICMP ( یعنی ۱ ) تنظیم شود.

دقّت کنید که خود بسته‌های ICMP نیز ممکن است دچار خطا شوند که برای این گونه خطا پیامی ارسال نخواهد شد .

شکل کلّی و قالب پیام ICMP در زیر مشخص شده است:

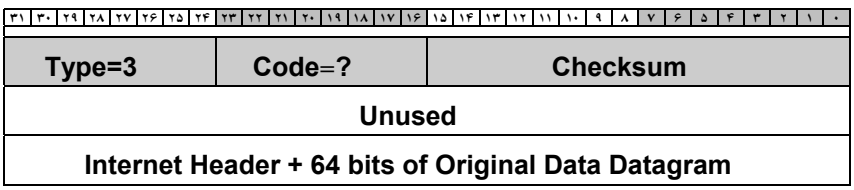
| ۳۱ | ۳۰ | ۲۹ | ۲۸ | ۲۷ | ۲۶ | ۲۵ | ۲۴ | ۲۳ | ۲۲ | ۲۱ | ۲۰ | ۱۹ | ۱۸ | ۱۷ | ۱۶ | ۱۵ | ۱۴ | ۱۳ | ۱۲ | ۱۱ | ۱۰ | ۹ | ۸ | ۷ | ۶ | ۵ | ۴ | ۳ | ۲ | ۱ | ۰ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Type** | | | | | | | | **Code** | | | | | | | | **Checksum** | | | | | | | | | | | | | | | |
| **Parameters** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Data** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

- فیلد **Type** : در این فیلد عددی قرار می‌گیرد که بیانگر نوع پیام می‌باشد و ساختار فیلدهای Parameters و Data بسته به عددی که در این فیلد قرار می‌گیرد متفاوت خواهد بود.
- فیلد **Code** : گاهی خود نوع پیام به چند زیر نوع دیگر تقسیم می‌شود که کد زیر نوع در این فیلد قرار می‌گیرد.

➢ فیلد **Checksum** : محتوای این فیلد برای سنجش اعتبار و سلامت بستهٔ ICMP مورد استفاده قرار می‌گیرد. تمامی بستهٔ ICMP بصورت دوبایت دوبایت جمع شده و نهایتاً از مکمل ۱ حاصل جمع ، عددی ۱۶ بیتی بدست می‌آید که درون این فیلد قرار می‌گیرد.

در ادامه نوع و ساختار پیامهای ICMP را توضیح می‌دهیم:

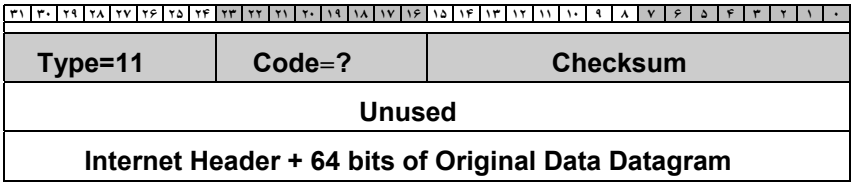
♦ پیام Destination Unreachable : این پیام زمانی صادر می‌شود که زیر شبکه یا یک مسیریاب نتواند آدرس مقصد را تشخیص بدهد یا به هر دلیلی بسته توسط ماشین میزبان تحویل گرفته نشود . (مثلاً بدلیل بزرگ بودن اندازهٔ بسته‌ها و عدم اجازه به مسیریاب برای شکستن آن )
ساختار بستهٔ حامل این پیام به صورت زیر است:




معنای شماره‌های مختلف در فیلد Code به شرح زیر است:
0 : شبکهٔ مورد نظر در دسترس نمی‌باشد.
1 : ماشین میزبان مورد نظر در دسترس نمی‌باشد.
2 : پروتکل مورد نظر تعریف نشده است.
3 : شمارهٔ پورت مورد نظر وجود ندارد.
4 : اندازهٔ بسته بزرگ است و نیاز به شکستن دارد در حالی که اجازه داده نشده است.

♦ پیام Time Exceeded : این پیام زمانی صادر می‌شود که مهلت قانونی یک بسته منقضی شده باشد و یک مسیریاب مجبور شود آنرا حذف کند؛ در چنین حالتی این پیام به آدرس فرستنده بسته IP برای آگاهی ارسال خواهد شد .

ساختار بستهٔ حامل این پیام به صورت زیر است:



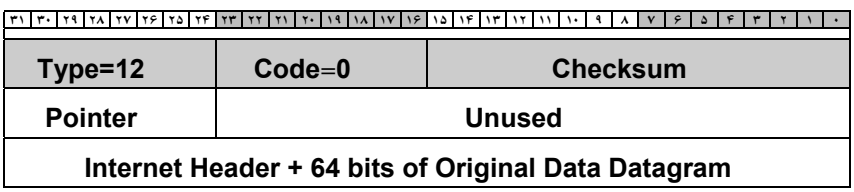

معنای شماره‌های مختلف در فیلد Code به شرح زیر است:

0 : زمان حیات بسته منقضی شده است. ( این پیام معمولاً توسط مسیریاب صادر میشود)

1 : زمان بازسازی قطعات یک دیتاگرام منقضی شده است. ( این پیام توسط ماشین میزبان صادر میشود)

♦ پیام Parameter Problem : این پیام زمانی صادر خواهد شد که مقداری نامعتبر در یکی از فیلدهای سرآیند در بسته IP قرار گرفته باشد و مسیریاب قادر به تشخیص و تفسیر سرآیند آن بسته IP نباشد. بعنوان مثال در فیلد Version از بسته IP عدد ۵ قرار گرفته باشد و یا Checksum با سرآیند تناقض داشته باشد.

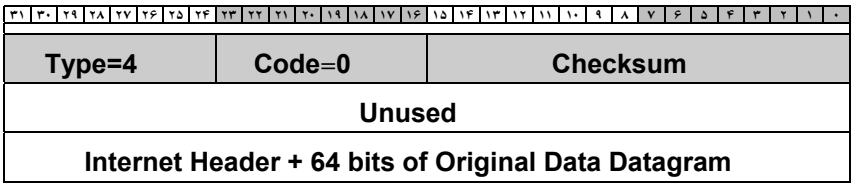
ساختار بستهٔ حامل این پیام به صورت زیر است:




فیلد Pointer محل بایتی را در بسته مشخص می‌کند که خطا در آن ناحیه بوده است.

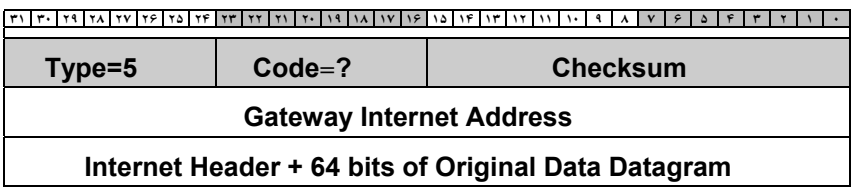
♦ پیام Source Quench : این بسته زمانی برای یک ماشین میزبان ارسال می‌شود که از آن خواسته شود حجم ارسال بسته‌هایش را کاهش بدهد چرا که در غیر اینصورت ازدحام پیش خواهد آمد . در مجموع هر گاه ازیک ماشین میزبان تقاضای کاهش نرخ تولید و ارسال

بسته‌های IP را داشته باشد این پیام را صادر می‌نماید. اگر ماشین میزبان پس از طی مدّت مشخصی این پیام را دریافت نکرد می‌تواند سرعت تولید بسته‌ها را به حالت اول برگرداند.
ساختار بستهٔ حامل این پیام به صورت زیر است:




♦ پیام Redirect : این پیام زمانی صادر می‌شود که یک مسیریاب احساس کند بسته یا بسته‌هایی که برای او ارسال شده است در مسیر صحیح نیستند و احتمالاً اشکالی در مسیریابی وجود دارد . این پیام می‌تواند برای هشدار خطاهای احتمالی موثر باشد.

ساختار بستهٔ حامل این پیام به صورت زیر است:




معنای شماره‌های مختلف در فیلد Code به شرح زیر است:

0 : باید تغییر مسیر به شبکه ای که آدرس آن مشخص شده است انجام شود.

1 : باید تغییر مسیر به ماشینی که آدرس آن مشخص شده است انجام شود.

2 : برای برآورده شدن سرویس ویژهٔ درخواستی که در فیلد Type of service مشخص شده ، باید تغییر مسیر به شبکه ای که آدرس آن مشخص شده است انجام شود.

3 : برای برآورده شدن سرویس ویژهٔ درخواستی که در فیلد Type of service مشخص شده ، باید تغییر مسیر به ماشینی که آدرس آن مشخص شده است انجام شود.

فرض کنید به مسیریاب R1 بسته ای ارسال شده و او با بررسی جدول مسیریابی آنرا به مسیریاب R2 فرستاده تا او آنرا به مقصد X برساند .حال اگر R2 با مقایسهٔ الگوی زیرشبکه به این نتیجه رسید که خود او و فرستنده آن بسته در یک شبکه واقعند با ارسال این پیام به فرستنده اعلام میکند اگر از این به بعد بسته‌هایش به جای اینکه به R1 ارسال شود به R2 داده شود ، زودتر به مقصد خواهد رسید؛ ضمناً آدرس IP خودش را نیز در فیلد Gateway Internet Address قرار می‌دهد .

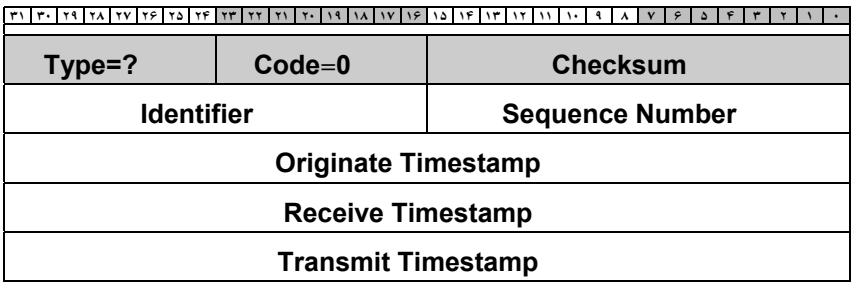
♦ پیغاهای Echo Request , Echo Reply : پیام Echo Request وقتی صادر می‌شود که یک مسیریاب بخواهد بداند آیا یک ماشین خاصّ شبکه قابل دسترس و موجود است یا خیر. در پاسخ به دریافت Echo Request ، مقصد با ارسال پیام Echo Reply به آن پاسخ می‌دهد . با این پرسش و پاسخ ، یک ماشین می‌تواند از قابل دسترس بودن یک مسیریاب یا ماشین میزبان در شبکه مطلع شود.
ساختار بستهٔ حامل این پیامها به صورت زیر است:

| ۳۱ | ۳۰ | ۲۹ | ۲۸ | ۲۷ | ۲۶ | ۲۵ | ۲۴ | ۲۳ | ۲۲ | ۲۱ | ۲۰ | ۱۹ | ۱۸ | ۱۷ | ۱۶ | ۱۵ | ۱۴ | ۱۳ | ۱۲ | ۱۱ | ۱۰ | ۹ | ۸ | ۷ | ۶ | ۵ | ۴ | ۳ | ۲ | ۱ | ۰ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Type=? | | | | | | | | Code=0 | | | | | | | | Checksum | | | | | | | | | | | | | | | |
| Identifier | | | | | | | | | | | | | | | | Sequence Number | | | | | | | | | | | | | | | |
| Data | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

معنای شماره‌های مختلف در فیلد Type به شرح زیر است:
8 : برای مشخص کردن پیام Echo Request
0 : برای مشخص کردن پیام Echo Reply

ابتدا پیام Echo Request به سمت ماشین مقصد ارسال میشود و ماشینی که آنرا دریافت کند ، آدرسهای مبدا و مقصد را عوض کرده و شمارهٔ نوع آنرا از ۸ به صفر تغییر داده ، پس از محاسبهٔ مجدد کد کشف خطا ، آنرا برمیگرداند. فیلدهای Identifier و Sequence number برای پیشگیری از اشتباه در همخوانی و تطابق پیامهای رفت و برگشتی است تا مبدأ بداند یک پاسخ مربوط به کدام تقاضای اوست .

♦ پیامهای Timestamp Reply و Timestamp Request : این دو پیام دقیقاً شبیه دو پیام تعریف شده در قبل هستند با این تفاوت که دریافت کنندهٔ آن ، زمان دریافت و زمان ارسال بسته را نیز در پاسخ به آن اضافه خواهد کرد. بنابراین ارسال کننده پیام Timestamp Request پس از دریافت پاسخ نه تنها از قابل دسترس بودن مقصد باخبر می‌شود بلکه زمان رفت و برگشت یک بسته را نیز می‌تواند تخمین بزند و به کمک آن جداول مسیریابی و همچنین کارائی شبکه را اندازه گیری نماید.
ساختار بستهٔ حامل این پیامها به صورت زیر است:




معنای شماره‌های مختلف در فیلد Type به شرح زیر است:
13 : برای مشخص کردن پیام Timestamp Request
14 : برای مشخص کردن پیام Timestamp Reply

Identifier & Sequence Number همانند پیامهای قبلی برای پیشگیری از اشتباه در همخوانی و تطابق پیامهای رفت و برگشتی است. Originate Timestamp زمانی است که مبدأ ، آن پیام را ارسال کرده است ( زمان بر حسب میلی ثانیه گذشته از نیمه شب و بر اساس زمان جهانی گرینویچ است) . Receive Timestamp زمانی است که گیرنده آن را دریافت کرده است و Transmit Timestamp زمان ارسال پاسخ بسته از طرف مقابل است . اگر زمان بر حسب میلی ثانیه آماده نبود بیت پرارزش از فیلد زمان یک می‌شود تا معلوم شود که آن فیلد معتبر نیست.

در پروتکل ICMP چهار پیام دیگر نیز وجود دارد که با استفاده از آنها یک ماشین میزبان می‌تواند آدرس IP شبکه محلی خود را در هنگامیکه چندین شبکه محلی از آدرسهای IP مشترک استفاده می‌کند پیدا نماید.

برای بدست آوردن اطلاعات جزئی تر و دقیق در مورد وظایف و پیامهای پروتکل ICMP به RFC-792 مراجعه نمائید.

## ۶) پروتکل ARP[1]

نکته ظریفی که در مورد شبکه اینترنت وجود دارد آن است که اگر چه تمامی ماشینهای میزبان و ابزارهای شبکه ای از آدرس IP که آدرس منحصر به فرد و یکتا است استفاده می‌کنند ولیکن یک بسته IP فقط در لایه شبکه قابل شناسائی و تحلیل است. یک بسته IP قبل از ارسال روی کانال از لایه اول یعنی لایهٔ فیزیکی عبور می‌کند و ضمن اضافه شدن اطلاعات لازم و تشکیل یک فریم ، روی کانال فیزیکی ارسال می‌شود . بعبارت روشنتر بسته IP قبل از ارسال درون فیلد داده از فریمی قرار می‌گیرد که بعداً در لایه اول تشکیل می‌شود؛ لایه اول وظیفه ای در قبال مسیریابی و کارهائی از این قبیل ندارد و فقط با آدرسهای فیزیکی کار می‌کند . بعنوان مثال اگر ماشین شما بخواهد بسته ای را برای ماشینی که روی شبکه محلی خودتان واقع است بفرستد، در لایه اول الزاماً بایستی آدرس فیزیکی ماشین شما (مبداء) و آدرس فیزیکی ماشین طرف مقابل (مقصد) معین باشد . ( این آدرسها بصورت سخت افزاری در کارت شبکه درج شده است) عدم دانستن آدرسهای فیزیکی عملاً مساوی عدم توانایی برای ارتباط خواهد بود چرا که روی کانال انتقال آدرسهای IP بی معنا هستند .

حال فرض کنید ماشین شما می‌خواهد بسته ای را برای ماشین دیگر ارسال کند که روی شبکه فعلی شما نیست. در این حالت هم لایه اول یک فریم برای ارسال روی کانال فیزیکی تشکیل می‌دهد و نیاز به آدرس MAC از مقصد دارد؛ آدرس فیزیکی مقصد چیست؟

در لایه اول هر گاه بسته ای قرار است به خارج از شبکه ارسال شود آدرس فیزیکی مقصد ، آدرس مسیریاب پیش فرض شما خواهد بود . بنابراین آدرسهای MAC مقوله ای جدا هستند و آدرسهای IP مقوله ای دیگر.

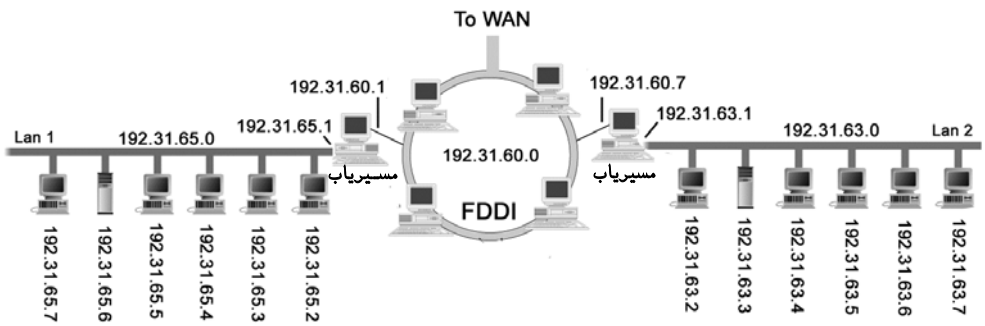
با مقدمه فوق به این نتیجه خواهیم رسید که هر ماشینی روی اینترنت گذشته از آن که بایستی آدرسهای IP خودش و مقصدش را بشناسد و بداند ، نیازمند به دانستن

[1] Address Resolution Protocol

آدرسهای فیزیکی ماشینهائی که مستقیماً با او در ارتباطند ، می‌باشد . بعنوان مثال شبکه اترنت که در تمام دنیا شناخته شده است از آدرسهائی استفاده می‌کند که منحصر به فرد و ٤٨ بیتی (٦ بایتی) است . بنابراین کامپیوتری که به یک کارت اترنت مجهز است گذشته از آن که بایستی یک آدرس IP منحصر به فرد داشته باشد یقیناً دارای یک آدرس ٤٨ بیتی یکتاست که این آدرس یکتا در کارخانه سازندهٔ آن ، تنظیم شده است . بنابراین وقتی پروتکل IP می‌خواهد یک بسته اطلاعاتی را روی شبکه بفرستد باید به نحوی آدرس فیزیکی اولین ماشینی که با آن بایستی ارتباط برقرار کند را بداند؛ این ماشین می‌تواند مسیریاب پیش فرض او باشد یا می‌تواند آدرس فیزیکی مقصد روی همین شبکه محلی باشد .

حال فرض کنید ایستگاهی آدرس IP ماشینی را که میخواهد با آن ارتباط برقرار کند ، می‌داند ولی آدرس فیزیکی او را نمی‌داند . چه کاری می‌تواند انجام بدهد؟ باید از پروتکل ARP بهره ببرد! در این پروتکل فرض بر آن است که تمامی ماشینهایِ روی یک شبکه محلی آدرس IP خود را می‌دانند.

برای روشن شدن وظیفهٔ پروتکل ARP به شکل (۱۰-۳) نگاه کنید.




شکل (۱۰-۳) شبکه بندی و آدرس دهی آنها در یک دانشکده

در مثال شکل (۱۰-۳) فرض کنید سه شبکه در دانشگاه شما نصب شده است . شبکهٔ محلی اول در دانشکدهٔ کامپیوتر با آدرس کلاس C به شماره 192.31.65.0 و شبکهٔ دوم در دانشکده برق با آدرس کلاس C به شمارهٔ 192.31.63.0 نصب شده است. ( هر دو شبکه از نوع اترنت هستند )

این دو شبکه از طریق یک شبکه فیبر نوری با استاندارد FDDI و با آدرس IP شماره 192.31.60.0 به همدیگر متصل شده‌اند . هر ماشین در شبکهٔ اترنت یک آدرس

۴۸ بیتی یکتا دارد. مسیریابها در شکل مشخص شده‌اند و ارتباط دو شبکه اترنت را با FDDI برقرار می‌کنند. شبکهٔ FDDI از طریق یک خط اختصاصی به شبکهٔ جهانی اینترنت متصل شده است . هر مسیریاب به دو شبکهٔ متفاوت متصل شده و به عنوان عضوی از هر دو شبکه دارای دو آدرس IP مجزا می‌باشد، که هر یک از آنها در یکی از شبکه‌های محلی تعریف شده است .

حال فرض کنید که ماشینی مایل است به آدرس خاصی مثلاً 192.31.65.5 بستهٔ IP بفرستد. در لایهٔ شبکه یک بستهٔ IP با مشخصات لازم ساخته می‌شود و در قسمت آدرس مقصد مقدار 192.31.65.5 قرار می‌گیرد. از دیدگاه لایه شبکه پس از تشکیل بستهٔ IP ، کار تمام است و لیکن از دیدگاه لایه اول که بایستی آن بسته را روی کانال ارسال کند دانستن آدرس فیزیکی (آدرس MAC ) ماشین مقصدی که آدرس IP آن 192.31.65.5 است ، حیاتی است .

وظیفه پروتکل ARP در اینجا آن است که یک "بسته فراگیر"[1] روی کل شبکه محلی منتشر کند که این بسته در حقیقت سوال می‌کند:

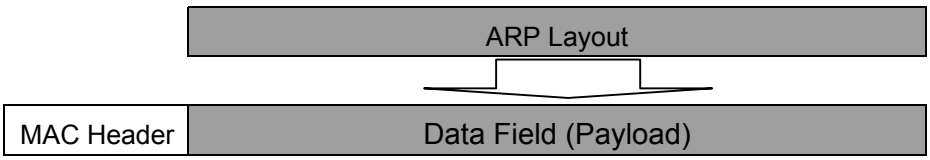
" آیکی IP او 192.31.65.5 است ، آدرس فیزیکی او چیست؟"

با توجه به آنکه بسته‌های فراگیر توسط تمامی ماشینهای روی شبکهٔ محلی دریافت می‌شود ، ماشینی که آدرس IP خودش را درون این بسته می‌بیند ، بدان پاسخ می‌دهد و آدرس فیزیکی خود را برای ارسال کنندهٔ آن بسته می‌فرستد. پس از آنکه آدرس فیزیکیِ مقصد بدست آمد ، یک فریم اترنت ساخته شده بر روی کانال منتقل می‌شود.

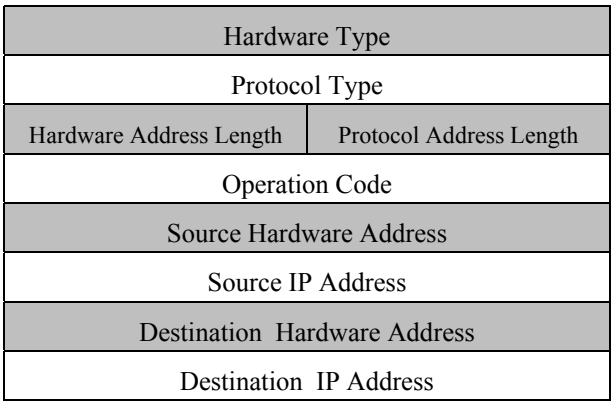
به این نکته توجه داشته باشید که هر ماشین بر روی شبکهٔ محلی از پروتکل ARP حمایت می‌کند و این پروتکل عملیات پرسش و پاسخ را برای هر ماشین که تقاضای ارسال بستهٔ IP دارد ، انجام می‌دهد.

بر خلاف پروتکل ICMP که روی پروتکل IP قرار می‌گیرد ، پروتکل ARP مستقیماً بر روی پروتکل لایهٔ فیزیکی عمل میکند؛ یعنی یک بستهٔ ARP ساخته شده و درون فیلد داده از فریم لایهٔ فیزیکی قرار گرفته و روی کانال ارسال می‌شود. در شکل (۱۱-۳) چگونگی ساخته شدن یک پیام ARP به تصویر کشیده شده است. در شکل (۱۲-۳) ساختار درونی بستهٔ ARP تشریح شده است.

[1] Broadcast




شکل (۱۱-۳) چگونگی قرار گرفتن یک پیام ARP درون فریم لایهٔ فیزیکی




شکل (۱۲-۳) ساختار پیامهای ARP

| شماره نوع | عنوان سخت افزار کارت شبکه |
|---|---|
| 1 | **Ethernet** |
| 2 | **Experimental Ethernet** |
| 3 | **X.25** |
| 4 | **Proteon ProNET (Token Ring)** |
| 5 | **Chaos** |
| 6 | **IEEE 802.X** |
| 7 | **ARCnet** |

جدول (۱۳-۳) تعریف استاندارد سخت افزار کارت شبکه

➢ **Hardware Type** : شمارهٔ نوع سخت افزار کارت شبکه که در لایهٔ اول وظیفهٔ انتقال اطلاعات روی کانال فیزیکی را بر عهده دارد. این شمارهها در جدول (۱۳-۳) مشخص شدهاند.

➢ **Protocol Type** : نوع پروتکلی که لایهٔ دوم از آن استفاده می‌شود. این پروتکلها و شمارهٔ آنها در جدول (۱٤-۳) مشخص شده‌اند. برای شبکه‌های مبتنی بر TCP/IP این شماره ۲۰٤۸ است.

➢ **Hardware Address Length** : با توجه به آنکه طول آدرسهای فیزیکی در شبکه‌ها ، متفاوت است در این فیلد طول آدرس (بر حسب بایت) مشخص میشود.

➢ **Protocol Address Length** : طول آدرسهای IP که در پروتکل TCP/IP مقدار ٤ است.

➢ **Operation Code (Opcode)** : ۱ برای ARP request

۲ برای ARP reply

➢ **Source Hardware Address** : آدرس فیزیکی مبدأ

➢ **Source IP Address** : آدرس IP ماشین مبدأ

➢ **Destination Hardware Address** : آدرس فیزیکی ماشین مقصد

➢ **Destination IP Address** : آدرس IP ماشین مقصد

برای بالا بردن سرعت پروتکل ARP ، وقتی برای یکبار آدرس فیزیکی متناظر با آدرس IP از یک ایستگاه بدست آمد ، پروتکل ARP این دو آدرس را در جدولی درون حافظهٔ اصلی که ARP Cache نامیده می‌شود ذخیره می‌کند تا اگر مجدداً به این آدرس نیاز شد به سرعت در اختیار قرار بگیرد . ساختار هر رکورد از این جدول بصورت زیر است:

| IF Index | Physical Address | IP Address | Type |
|---|---|---|---|

♦ **IF Index** : شماره پورت سخت افزاری متناظر با آن کارت شبکه

♦ **Physical Address** : آدرس سخت افزاری کارت شبکه

♦ **IP Address** : آدرس IP متناظر با آدرس سخت افزاری

♦ **Type** : مقداری که در این فیلد قرار می‌گیرد وضعیت هر رکورد را در این جدول مشخص میکند: مقدار ۱ : یعنی این رکورد باید بطور متناوب به هنگام شود. دقت کنید که ARP Cache هر دقیقه یکبار "بهنگام سازی"[۱] می‌شود. مقدار ٤: بدین معناست که این رکورد ثابت و بدون تغییر است و نباید بهنگام شود. مقدار ۱: یعنی رکورد چون بهنگام نشده از اعتبار ساقط است.

[۱] Update

| شمارهٔ پروتکل | نام پروتکل |
|---|---|
| **512** | XEROX PUP |
| **513** | PUP Address Translation |
| **1536** | XEROX NS IDP |
| **2048** | Internet Protocol (IP) |
| **2049** | X.75 |
| **2050** | NBS |
| **2051** | ECMA |
| **2052** | Chaosnet |
| **2053** | X.25 Level 3 |
| **2054** | Address Resolution Protocol (ARP) |
| **2055** | XNS |
| **4096** | Berkeley Trailer |
| **21000** | BBN Simnet |
| **24577** | DEC MOP Dump/Load |
| **24578** | DEC MOP Remote Console |
| **24579** | DEC DECnet Phase IV |
| **24580** | DEC LAT |
| **24582** | DEC |
| **24583** | DEC |
| **32773** | HP Probe |
| **32784** | Excelan |
| **32821** | Reverse ARP |
| **32824** | DEC LANBridge |
| **32823** | AppleTalk |

جدول (۱٤–۳) شمارهٔ پروتکلهای لایهٔ دوم

مسئله دیگری که ممکن است در هنگام بکارگیری پروتکل ARP رخ بدهد آن است که وقتی آدرس IP مربوط به ایستگاهی روی شبکه محلی سوال می‌شود ، ممکن است آن ایستگاه روی شبکه محلی دیگری باشد و بالطبع پاسخی نمی‌رسد. در چنین حالتی دو راه حل وجود دارد:

الف : وقتی مسیریابی که به آن شبکه متصل است می‌بیند آدرس مقصدی که توسط ARP سوال شده روی یک شبکه محلی دیگر واقع است در پاسخ به آن ، آدرس فیزیکی خودش را به ایستگاه سوال کننده ارسال می‌دارد؛ به این روش Proxy ARP گفته می‌شود.

ب: ایستگاهها خودشان موظفند به روشی که در مبحث "الگوی زیرشبکه" اشاره شد مستقلاً محلی بودن یا خارجی بودن ماشین مقصد را تشخیص داده و در صورت خارجی بودن ، آدرس فیزیکی یک مسیریاب مناسب را انتخاب نمایند.

نکته آخری که در مورد پروتکل ARP بایستی توضیح بدهیم آن است که در مسیریابها نیز برای شناسائی آدرس ایستگاههای یک شبکهٔ محلی متصل به آنها بهمین روش عمل می‌شود. برای جزئیات دقیقتر پروتکل ARP به REC-826 مراجعه کنید.

## ۷) پروتکل RARP[1]

پروتکل ARP برای یافتن آدرس‌های فیزیکی ایستگاههائی است که آدرس IP خود را می‌دانند. پروتکل RARP دقیقاً عکس پروتکل ARP عمل می‌کند . گاهاً اتفاق می‌افتد که ایستگاه آدرس فیزیکی مورد نظرش را میداند ولیکن آدرس IP آنرا نمی‌داند؛ این قضیه برای ایستگاههائی که بدون دیسکند و از طریق سرویس دهنده بوت می‌شوند صادق است .

در این پروتکل برای شناسایی آدرس IP متناظر با یک آدرس فیزیکی یک بسته فراگیر روی خط ارسال می‌شود که در آن آدرس فیزیکی یک ایستگاه قرار دارد . تمامی‌ایستگاههایی که از پروتکل RARP حمایت می‌کنند و بسته‌های مربوطه را تشخیص می‌دهند ، در صورتی که آدرس فیزیکی خودشان را درون بسته ببینند در پاسخ به آن ، آدرس IP خود را در قالب یک بستهٔ RARP Reply برمی‌گردانند. بعنوان

[1] Reverse Address Resolution Protocol

مثال فرض کنید ایستگاهی با قرار دادن بستهٔ RARP و آدرس ٦ بایتی اترنت -14-04 D5-C8-01-25 روی خط ، آدرس IP آنرا طلب می‌کند . هر ماشین که آدرس IP متناظر با آن را می‌داند به این بسته RARP پاسخ می‌دهد.

دقت کنید که بسته‌های PARP,ARP از نوع "فراگیر محلی"[1] هستند و بالطبع توسط مسیریابها منتقل نمیشوند و فقط در محدوده شبکه محلی عمل می‌کنند. مستندات RARP در RFC903 آمده است .

## ۸) پروتکل BootP

با توجه به آنچه که در مورد RARP گفته شد بسته‌های سوال کننده آدرس IP از نوع محلی هستند و بالطبع این گونه بسته‌ها از مسیریابها به خارج از شبکه منتقل نخواهد شد.

گاهی نیاز است که یک آدرس IP روی چند شبکه محلی جستجو شود که در این حالت RARP جوابگو نیست. ( این نیاز برای ایستگاههای بدون دیسک بوجود می‌آید چرا که پس از روشن شدن بایستی از طریق سرویس دهندهٔ شبکه[2] بوت شوند)

پروتکل BOOTP در چنین محیطهایی کاربرد دارد و از دیتاگرام‌های نوع UDP که در آینده به آنها خواهیم پرداخت استفاده می‌کند و مسیریابها موظف به انتقال آنها هستند . در این پروتکل نکته جالبی وجود دارد و آن هم آنست که در پاسخ به چنین بسته‌هائی به غیر از آدرس IP ایستگاه مورد نظر، اطلاعات لازم جهت بوت شدن سیستم و همچنین "الگوی زیرشبکه" برای ایستگاه تقاضا کننده که احتمالاً یک ایستگاه بدون دیسک است در قالب یک بستهٔ UDP ارسال خواهد شد.

## ۹) شمارهٔ پروتکل‌های استاندارد در لایهٔ سوم

دیتاگرامی که در فیلد داده از یک بستهٔ IP حمل می‌شود با ساختمان دادهٔ خاصّ از لایهٔ بالاتر تحویلِ پروتکل IP می‌شود تا روی شبکه ارسال شود. بعنوان مثال ممکن است این داده‌ها را پروتکل TCP در لایه بالاتر ارسال کرده باشد و یا ممکن است این

---

[1] Local Broadcast
[2] Network Server

کار توسط پروتکل UDP انجام شده باشد. بنابراین مقدار این فیلد شمارهٔ پروتکلی است که در لایه بالاتر تقاضای ارسال یک دیتاگرام کرده است؛ بسته‌ها پس از دریافت در مقصد باید به پروسهٔ متناظر با پروتکل تعیین شده ، تحویل داده شود. پروتکلهای لایهٔ بالاتر دارای یک شمارهٔ هشت بیتی منحصر بفرد و استاندارد هستند که در جدول (۱۵-۳) شماره و نام این پروتکلها ارائه شده است.

| | | |
|---|---|---|
| 0 | Reserved | [JBP] |
| 1 | ICMP | Internet Control Message [RFC792,JBP] |
| 2 | IGMP | Internet Group Management [RFC1112,JBP] |
| 3 | GGP | Gateway-to-Gateway [RFC823,MB] |
| 4 | IP | IP in IP (encasulation) [JBP] |
| 5 | ST | Stream [RFC1190,IEN119,JWF] |
| 6 | TCP | Transmission Control [RFC793,JBP] |
| 7 | UCL | UCL [PK] |
| 8 | EGP | Exterior Gateway Protocol [RFC888,DLM1] |
| 9 | IGP | any private interior gateway [JBP] |
| 10 | BBN-RCC-MON | BBN RCC Monitoring [SGC] |
| 11 | NVP-II | Network Voice Protocol [RFC741,SC3] |
| 12 | PUP | PUP [PUP,XEROX] |
| 13 | ARGUS | ARGUS [RWS4] |
| 14 | EMCON | EMCON [BN7] |
| 15 | XNET | Cross Net Debugger [IEN158,JFH2] |
| 16 | CHAOS | Chaos [NC3] |
| 17 | UDP | User Datagram [RFC768,JBP] |
| 18 | MUX | Multiplexing [IEN90,JBP] |
| 19 | DCN-MEAS | DCN Measurement Subsystems [DLM1] |
| 20 | HMP | Host Monitoring [RFC869,RH6] |
| 21 | PRM | Packet Radio Measurement [ZSU] |
| 22 | XNS-IDP | XEROX NS IDP [ETHERNET,XEROX] |
| 23 | TRUNK-1 | Trunk-1 [BWB6] |
| 24 | TRUNK-2 | Trunk-2 [BWB6] |
| 25 | LEAF-1 | Leaf-1 [BWB6] |
| 26 | LEAF-2 | Leaf-2 [BWB6] |
| 27 | RDP | Reliable Data Protocol [RFC908,RH6] |
| 28 | IRTP | Internet Reliable Transaction [RFC938,TXM] |
| 29 | ISO-TP4 | ISO Transport Protocol Class 4 [RFC905,RC77] |
| 30 | NETBLT | Bulk Data Transfer Protocol [RFC969,DDC1] |
| 31 | MFE-NSP | MFE Network Services Protocol [MFENET,BCH2] |
| 32 | MERIT-INP | MERIT Internodal Protocol [HWB] |
| 33 | SEP | Sequential Exchange Protocol [JC120] |
| 34 | 3PC | Third Party Connect Protocol [SAF3] |
| 35 | IDPR | Inter-Domain Policy Routing Protocol [MXS1] |
| 36 | XTP | XTP [GXC] |
| 37 | DDP | Datagram Delivery Protocol [WXC] |
| 38 | IDPR-CMTP | IDPR Control Message Transport Proto [MXS1] |
| 39 | TP++ | TP++ Transport Protocol [DXF] |
| 40 | IL | IL Transport Protocol [DXP2] |
| 41 | SIP | Simple Internet Protocol [SXD] |
| 42 | SDRP | Source Demand Routing Protocol [DXE1] |
| 43 | SIP-SR | SIP Source Route [SXD] |

| | | | |
|---|---|---|---|
| 44 | SIP-FRAG | SIP Fragment | [SXD] |
| 45 | IDRP | Inter-Domain Routing Protocol | [Sue Hares] |
| 46 | RSVP | Reservation Protocol | [Bob Braden] |
| 47 | GRE | General Routing Encapsulation | [Tony Li] |
| 48 | MHRP | Mobile Host Routing Protocol | [David Johnson] |
| 49 | BNA | BNA | [Gary Salamon] |
| 50 | SIPP-ESP | SIPP Encap Security Payload | [Steve Deering] |
| 51 | SIPP-AH | SIPP Authentication Header | [Steve Deering] |
| 52 | I-NLSP | Integrated Net Layer Security TUBA | [GLENN] |
| 53 | SWIPE | IP with Encryption | [JI6] |
| 54 | NHRP | NBMA Next Hop Resolution Protocol | |
| 55-60 | | Unassigned | [JBP] |
| 61 | | any host internal protocol | [JBP] |
| 62 | CFTP | CFTP | [CFTP,HCF2] |
| 63 | | any local network | [JBP] |
| 64 | SAT-EXPAK | SATNET and Backroom EXPAK | [SHB] |
| 65 | KRYPTOLAN | Kryptolan | [PXL1] |
| 66 | RVD | MIT Remote Virtual Disk Protocol | [MBG] |
| 67 | IPPC | Internet Pluribus Packet Core | [SHB] |
| 68 | | any distributed file system | [JBP] |
| 69 | SAT-MON | SATNET Monitoring | [SHB] |
| 70 | VISA | VISA Protocol | [GXT1] |
| 71 | IPCV | Internet Packet Core Utility | [SHB] |
| 72 | CPNX | Computer Protocol Network Executive | [DXM2] |
| 73 | CPHB | Computer Protocol Heart Beat | [DXM2] |
| 74 | WSN | Wang Span Network | [VXD] |
| 75 | PVP | Packet Video Protocol | [SC3] |
| 76 | BR-SAT-MON | Backroom SATNET Monitoring | [SHB] |
| 77 | SUN-ND | SUN ND PROTOCOL-Temporary | [WM3] |
| 78 | WB-MON | WIDEBAND Monitoring | [SHB] |
| 79 | WB-EXPAK | WIDEBAND EXPAK | [SHB] |
| 80 | ISO-IP | ISO Internet Protocol | [MTR] |
| 81 | VMTP | VMTP | [DRC3] |
| 82 | SECURE-VMTP | SECURE-VMTP | [DRC3] |
| 83 | VINES | VINES | [BXH] |
| 84 | TTP | TTP | [JXS] |
| 85 | NSFNET-IGP | NSFNET-IGP | [HWB] |
| 86 | DGP | Dissimilar Gateway Protocol | [DGP,ML109] |
| 87 | TCF | TCF | [GAL5] |
| 88 | IGRP | IGRP | [CISCO,GXS] |
| 89 | OSPFIGP | OSPFIGP | [RFC1583,JTM4] |
| 90 | Sprite-RPC | Sprite RPC Protocol | [SPRITE,BXW] |
| 91 | LARP | Locus Address Resolution Protocol | [BXH] |
| 92 | MTP | Multicast Transport Protocol | [SXA] |
| 93 | AX.25 | AX.25 Frames | [BK29] |
| 94 | IPIP | IP-within-IP Encapsulation Protocol | [JI6] |
| 95 | MICP | Mobile Internetworking Control Pro. | [JI6] |
| 96 | SCC-SP | Semaphore Communications Sec. Pro. | [HXH] |
| 97 | ETHERIP | Ethernet-within-IP Encapsulation | [RXH1] |
| 98 | ENCAP | Encapsulation Header | [RFC1241,RXB3] |
| 99 | | any private encryption scheme | [JBP] |
| 100 | GMTP | GMTP | [RXB5] |
| 101-254 | | Unassigned | [JBP] |

جدول (۱۵–۳) شماره و نام پروتکلهای استانداردِ تولیدکننده و دریافت کنندهٔ دیتاگرام

## ۱۰) مراجع این فصل

مجموعهٔ مراجع زیر می‌توانند برای دست آوردن جزییات دقیق و تحقیق جامع در مورد مفاهیم معرفی شده در این فصل مفید واقع شوند.

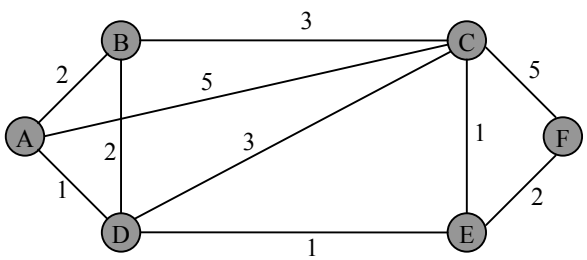

| | |
|---|---|
| RFC 1219 | "On the Assignment of Subnet Numbers," Tsuchiya, P.F.; 1991 |
| RFC 1112 | "Host Extensions for IP Multicasting," Deering, S.E.; 1989 |
| RFC 1088 | "Standard for the Transmission of IP Datagrams over NetBIOS Networks," McLaughlin, L.J.; 1989 |
| RFC 950 | "Internet Standard Subnetting Procedure," Mogul, J.C.; Postel, J.B.; 1985 |
| RFC 932 | "Subnetwork Addressing Schema," Clark, D.D.; 1985 |
| RFC 922 | "Broadcasting Internet Datagrams in the Presence of Subnets," Mogul, J.C.; 1984 |
| RFC 919 | "Broadcasting Internet Datagrams," Mogul, J.C.; 1984 |
| RFC 886 | "Proposed Standard for Message Header Munging," Rose, M.T.; 1983 |
| RFC 815 | "IP Datagram Reassembly Algorithms," Clark, D.D.; 1982 |
| RFC 814 | "Names, Addresses, Ports, and Routes," Clark, D.D.; 1982 |
| RFC 792 | "Internet Control Message Protocol," Postel, J.B.; 1981 |
| RFC 791 | "Internet Protocol," Postel, J.B.; 1981 |
| RFC 781 | "Specification of the Internet Protocol (IP) Timestamp Option," Su, Z.; 1981 |

## ۱) مفاهیم اولیهٔ مسیریابی

در فصل قبل اشاره شد که مسیریاب ابزاری است که ارتباط دو یا چند شبکه را برقرار می‌نماید. مجدداً با مراجعه به فصل قبلی به شکلهای (٤-۳) و (٥-۳) دقت کنید. در این دو شکل ، کامپیوترهای S1 تا S5 نقش مسیریاب را ایفا می‌کنند؛ مجموعهٔ این مسیریابها و کانالهای فیزیکی مابین آنها "زیرساخت ارتباطی"[۱] شبکه را تشکیل می‌دهد. در نشان دادن زیرساخت ارتباطی از یک شبکه ، تمامی ماشینهای میزبان حذف خواهند شد چراکه این ماشینها هیچ تاثیری در برقراری ارتباط و حمل ترافیک بسته‌ها نداشته و به عنوان استفاده‌کنندهٔ نهایی[۲] مطرح هستند.

در شکل (۱-٤) به زیرساخت ارتباطی یک شبکهٔ فرضی که در قالب یک گراف نشان داده شده است ، دقت کنید. در این گراف گره‌های A تا F مسیریابها هستند و خطوط ارتباطی بین هر دو گره (لبه[۳]) نشان‌دهندهٔ وجود یک کانال فیزیکی بین آنها می‌باشد. اعدادی که روی هر لبه نوشته شده است پارامتر هزینهٔ رسیدن از یک گره به گره دیگر خواهد بود. در ساده‌ترین حالت می‌توان معیار هزینه را زمان تاخیر در نظر گرفت. البته پارامتر تاخیر را باید ترکیبی از "تاخیر فیزیکی انتشار" و "زمان پردازش" یعنی زمان اجرای الگوریتم مسیریابی بر روی یک بسته در نظر گرفت. مجموع این دو زمان می‌تواند به عنوان پارامتر هزینه در نظر گرفته شود. البته در بخشهای آتی خواهیم دید که معیار هزینه در برخی از مسیریابها بر اساس پارامترهای پیچیده‌تری همانند امنیت ، سیاست و اقتصاد ارزیابی می‌شود.

حال فرض کنید بسته‌ای وارد مسیریاب A شده تا پس از طی مسیر ، به F تحویل داده شود. اصلیترین وظیفهٔ الگوریتمهای مسیریابی ، پیدا کردن مسیری بهینه از A به F می‌باشد به‌گونه‌ای که هزینهٔ کل مسیر به حداقل برسد.




شکل (۱-٤) زیرساخت ارتباطی یک شبکهٔ فرضی

---

[۱] Communication Subnet
[۲] End User
[۳] Edge

به‌گونه‌ای که از شکل مشخص است دوازده مسیر متفاوت برای رسیدن یک بسته از A به F وجود دارد که در زیر فهرست شده اند:

| | | | | | |
|---|---|---|---|---|---|
| A→B→C →F | ≺ | با هزینهٔ ۱۰ | ≺ | A→D→C →F | با هزینهٔ ۹ |
| A→B→C →E→F | ≺ | با هزینهٔ ۸ | ≺ | A→D →E→F | با هزینهٔ ۴ |
| A→B→C→D→E→F | ≺ | با هزینهٔ ۱۱ | ≺ | A→D→C→ E→F | با هزینهٔ ۷ |
| A →C →F | ≺ | با هزینهٔ ۱۰ | ≺ | A→D→B→C →F | با هزینهٔ ۱۱ |
| A→ C →E→F | ≺ | با هزینهٔ ۸ | ≺ | A→D→B→C→ E→F | با هزینهٔ ۹ |
| A→ C→D→E→F | ≺ | با هزینهٔ ۱۱ | ≺ | A→ D→C→E→F | با هزینهٔ ۷ |

از بین این مسیرها بهترین مسیر برای رسیدن از A به F ، مسیر A→D →E→F خواهد بود.
دو مسئلهٔ مهم در مسیریابی مطرح است:
الف) هر یک از مسیریابها چگونه از پارامتر هزینهٔ کل کانالهای شبکه مطلع شوند ، تا بتوانند گراف زیرساخت ارتباطی شبکه را تشکیل داده و بهترین مسیر را انتخاب نمایند؟
ب) چه الگوریتمی برای یافتن بهترین مسیر انتخاب شود تا از لحاظ سرعت پردازش و تصمیم‌گیری ، بهینه بوده و بسته ها را با تاخیر و انتظار مواجه نکند؟ (یعنی از لحاظ پیچیدگی زمانی الگوریتم بهینه باشد).

بهتر است قبل از ادامهٔ بحث به اصطلاحات کلیدی از فصل قبل که به صورت خلاصه در جدول (۲-٤) ارائه شده است دقت کنید.

### ۱-۱) روشهای هدایت بسته‌های اطلاعاتی در شبکه‌های کامپیوتری

دو روش برای انتقال بسته‌های اطلاعاتی در شبکه‌های کامپیوتری مطرح است که هر کدام از آنها در شبکه‌های امروزی به نحوی مورد استفاده قرار می‌گیرد:
الف) روش "مدار مجازی"[۱] که اختصاراً روش VC گفته می‌شود.
ب ) روش "دیتاگرام"[۲]

[۱] Virtual Circuit
[۲] Datagram
**۲ مفهوم روش دیتاگرام در مسیریابی را با مفهوم دیتاگرام به معنای "یک واحد اطلاعاتی" در لایهٔ اینترنت اشتباه نکنید.**

**آدرسهای MAC** : آدرسهائی هستند که در لایه فیزیکی (لایهٔ اوّل) تعریف می‌شوند و فقط برای انتقال فریمها روی کانال مورد استفاده قرار می‌گیرند. در حقیقت این آدرسها روشی برای تحریکِ سخت‌افزار کارت شبکه هستند تا اطلاعات را از روی کانال مشترک بردارد. بنابراین چگونگی تعریف این آدرسها و اصول آدرس‌دهی و اندازهٔ این آدرسها (برحسب بایت) شدیداً به پروتکل و توپولوژی شبکه وابسته است.

**آدرسهای IP** : آدرسهای جهانی و منحصر به فرد که یک ماشین را فارغ از نوع سخت‌افزار و نرم‌افزار آن ، مشخص می‌نماید.

**بستهٔ IP** : یک واحد اطلاعاتی با اندازه محدود ( ولی متغیر ) که باید در زیرساخت ارتباطی یک شبکه از مبدأ به سمت مقصد هدایت شود.

در هدایت بسته‌های اطلاعاتی از یک مسیریاب به مسیریاب دیگر آدرسهای MAC دائماً تغییر می‌کنند ولی آدرسهای IP ثابت و جهانی هستند. مسیریاب بر اساس این آدرسها هدایت بسته را به سمت مقصد انجام می‌دهد.

**مسیریاب** : ابزاری است که تعدادی ورودی و خروجی داشته و بسته‌های اطلاعاتی را از ورودی تحویل گرفته و بر اساس آدرسهای جهانی یکی از کانالهای خروجی را برای انتقال بسته انتخاب می‌نماید ، به نحوی که بسته را به مقصد نزدیک نماید.

**شرایط توپولوژیکی شبکه** : مجموعهٔ مسیریابها و کانالهای فیزیکی مابین آنها در زیرساخت ارتباطی یک شبکه ، توپولوژی آن شبکه را تشکیل می‌دهد. با توجه به آنکه با ورود یک مسیریاب جدید به شبکه یا خرابی یکی از کانالهای ارتباطی یا حذف یک مسیریاب ، توپولوژی زیرساخت ارتباطی تغییر خواهد کرد لذا توپولوژی شبکه متغیر با زمان خواهد بود.

**شرایط ترافیکی شبکه** : تعداد متوسط بسته‌های اطلاعاتی را که در واحد زمان روی یک کانال ارسال (یا دریافت ) می‌شود ، ترافیک آن کانال گویند. چون تولید بسته‌های اطلاعاتی توسط ماشینهای میزبان کاملاً تصادفی و نامعین است بنابراین ترافیک در شبکه نیز کاملاً متغیر با زمان خواهد بود.

**گام یا Hop** : به عبور بسته از یک مسیریاب ، گام و به تعداد مسیریابهایی که یک بسته در طی مسیر خود به سمت مقصد می‌پیماید "تعداد گام" گفته می‌شود.

**ازدحام یا Congestion** : زمانی که تعداد متوسط بسته‌های ورودی به یک مسیریاب از تعداد متوسط بسته‌های خروجی از آن بیشتر شود ، ازدحام رخ داده و تاخیر ارسال بسته‌ها در آن مسیریاب شروع به افزایش خواهد کرد. هرگاه تاخیر به حدی برسد که طول عمر بسته‌ها تمام شود ، اصطلاحاً بن‌بست –Deadlock– پدید آمده و مسیریاب عملاً مسدود شده است.

جدول (۲-٤) تعریف برخی از اصطلاحات کلیدی در مبحث مسیریابی

در روش مدار مجازی قبل از شروع به ارسال بسته‌های اطلاعاتی از یک ماشین ، ابتدا یک مسیر بین مبدأ و مقصد برقرار می‌شود؛ بدینصورت که مبدأ ابتدا با ارسال یک بستهٔ کنترلی خاص با یک شمارهٔ ویژه بر روی شبکه اعلام می‌کند که خواستار برقراری ارتباط با یک مقصد خاص می‌باشد. هر مسیریاب که آن بسته را دریافت کند ضمن پیداکردن یک مسیر مناسب برای آن بسته شمارهٔ آن را در جدولی درج میکند و از آن به بعد هر بسته‌ای که با این شماره وارد شود از همان مسیری که برای بستهٔ اول انتخاب شده بود به سمت مقصد هدایت می‌شود. بنابراین تمامی بسته‌های اطلاعاتی که بعد از برقراری یک مسیر ، از مبدأ به سمت مقصد ارسال می‌شوند نیاز به مسیریابی جداگانه نخواهند داشت. به این مسیر که فقط یکبار ایجاد می‌شود ، "مدار مجازی" گفته می‌شود. "مدار مجازی" تا وقتی با اطلاع طرفین ارتباط و اعلام به مسیریابهای واقع بر روی مدار خاتمه داده نشود ، برقرار خواهد ماند. از آنجایی که در روش VC تمام بسته های اطلاعاتی از یک مسیر واحد حرکت می‌کنند ، این تضمین وجود دارد که بسته‌های اطلاعاتی در مقصد به همان ترتیبی که در مبدأ ارسال شده‌اند ، دریافت شوند.

خصوصیات روش VC را می‌توان به صورت زیر خلاصه کرد:

- برای ارسال بسته‌های اطلاعاتی به آدرسهای جهانی مبدأ و مقصد نیازی نیست بلکه فقط به شمارهٔ VC نیاز است.
- برای هدایت بسته‌های اطلاعاتی از مبدأ به سمت مقصد نیاز به اجرای الگوریتم مسیریابی به ازای تک‌تک بسته‌ها نیست بلکه فقط یک جستجو در جدول هر مسیریاب انجام می‌شود.
- بسته‌ها الزاماً به ترتیب به مقصد خواهند رسید.
- احتمال گم شدن بسته‌ها ناشی از اشتباه در عمل مسیریابی در شبکه وجود ندارد.

در روش دیتاگرام هر ماشین میزبان پس از آنکه بسته‌ای را تولید کرد تحویل اولین مسیریاب در دسترس می‌دهد. مسیریابها مختارند بر اساس شرایط ترافیکی و توپولوژیکی زیرساخت ارتباطی شبکه ، مسیری را برای آن بسته انتخاب کرده و آن بسته را روی آن مسیر ارسال نمایند. بنابراین هیچ مسیر ثابت و از قبل مشخصی برای بسته‌های متوالی وجود ندارد. یعنی وقتی دو بسته از یک مبدأ تولید و به سمت یک مقصدِ واحد ارسال می‌شود ممکن است مسیرهای متفاوتی را طی نمایند؛ در ضمن ممکن است بسته‌ها به ترتیبی که تولید می‌شوند به مقصد نرسند.

خصوصیات روش دیتاگرام را می‌توان به صورت زیر خلاصه کرد:

- هر بستهٔ اطلاعاتی به آدرسهای جهانی مبدأ و مقصد نیازمند است.
- برای هر بسته باید مسیریابی جداگانه انجام شود.

- توزیع و هدایت بسته‌ها روی مسیرهای متفاوت ، بر اساس شرایط توپولوژیکی و ترافیکی لحظه‌ای شبکه خواهد بود.
- چون بسته‌ها به ترتیب نمی‌رسند باید فرآیندی برای تنظیم ترتیب بسته‌ها اتخاذ شود.
- در لایهٔ بالاتر باید نظارتهای ویژه بر گم شدن یا دوبله شدن بسته‌ها انجام شود.

### ۲-۱) انواع الگوریتمهای مسیریابی

الگوریتمهای مسیریابی را با دو دیدگاه میتوان دسته‌بندی کرد :
الف) از دیدگاه روش تصمیم‌گیری و میزان هوشمندی الگوریتم
ب) از دیدگاه چگونگی جمع‌آوری و پردازش اطلاعاتِ زیرساخت ارتباطی شبکه

- با دیدگاه اول الگوریتمهای مسیریابی را میتوان به دو دستهٔ **"ایستا"** و **"پویا"** تقسیم‌بندی کرد. در الگوریتمهای **ایستا** هیچ اعتنایی به شرایط توپولوژیکی و ترافیک لحظه‌ای شبکه نمی‌شود. معمولاً در این الگوریتمها برای هدایت یک بسته ، هر مسیریاب از جداولی استفاده می‌کند که در هنگام برپایی شبکه تنظیم شده و در طول زمان ثابت است. در هنگام وقوع هرگونه تغییر در توپولوژی زیرساختِ شبکه ، این جداول باید توسط مسئول شبکه بصورت دستی مجدداً تنظیم شود. اگرچه این الگوریتمها بسیار سریعند ولی چون ترافیک لحظه‌ای شبکه متغیر است ، نمی‌توانند بهترین مسیرها را انتخاب نمایند و هرگونه تغییر در توپولوژی زیرساخت ارتباطی شبکه ، یک مشکل عمده و جدی ایجاد خواهد کرد.

در الگوریتمهای **پویا** مسیریابی بر اساس آخرین وضعیت توپولوژیکی و ترافیک شبکه انجام می‌شود. جداول مسیریابی در این نوع الگوریتمها هر T ثانیه یکبار به‌هنگام می‌شود. این الگوریتمها بر اساس وضعیت فعلی شبکه تصمیم‌گیری می‌نمایند ولی ممکن است پیچیدگی این الگوریتمها به قدری زیاد باشد که زمان تصمیم‌گیری برای انتخاب بهترین مسیر ، طولانی شده و منجر به تاخیرهای بحرانی شده و نهایتاً به ازدحام بیانجامد؛ بهمین دلیل در مسیریابهای سریع از تکنیکهای چندپردازنده‌ای و پردازش موازی استفاده می‌شود.

- از دیدگاه دوم الگوریتمهای مسیریابی به دو دستهٔ "سراسری / متمرکز"[۱] و "غیرمتمرکز"[۲] تقسیم می‌شود.

در "الگوریتمهای سراسری" هر مسیریاب باید اطلاعات کاملی از زیرساخت ارتباطی شبکه داشته باشد. یعنی هر مسیریاب باید تمامی مسیریابهای دیگر ، ارتباطات بین آنها و هزینهٔ هر

[۱] Global Routing Algorithm
[۲] Decentralized Routing Algorithm

خط را دقیقاً شناسایی نماید. سپس با جمع‌آوری این اطلاعات "ساختمان دادهٔ" مربوط به گراف زیرساخت شبکه را تشکیل بدهد. در چنین شرایطی برای یافتن بهترین مسیر بین هر دو مسیریاب ، از الگوریتمهای کوتاهترین مسیر نظیر "الگوریتم دایجکسترا"[۱] استفاده می‌شود. به چنین الگوریتمهایی که برای مسیریابی به اطلاعات کاملی از زیرساخت شبکه و هزینهٔ ارتباط بین هر دو مسیریاب نیازمندند ، اختصاراً الگوریتمهای LS[۲] گفته می‌شود و در مسیریابهای مدرن و جدید از آن استفاده می‌شود.

در الگوریتمهای "غیر متمرکز" ، مسیریاب اطلاعات کاملی از زیرساخت شبکه ندارد بلکه فقط قادر است هزینهٔ ارتباط با مسیریابهایی که بطور مستقیم و فیزیکی با آنها در ارتباط است محاسبه و ارزیابی نماید. سپس در فواصل زمانی منظم ، هر مسیریاب جدول مسیریابی خود را برای مسیریابهای مجاور ، ارسال می‌نماید. مسیریاب با دریافت این جداول و مقادیری که خودش مستقیماً اندازه‌گیری کرده ، با یک الگوریتم بسیار ساده جدول خودش را به‌هنگام می‌نماید و برای هدایت هر بسته ، از آن استفاده می‌کند. در این الگوریتمها برای مسیریابی هر بسته ، فقط یک جستجو در جدول مسیریابی کافی است و در نتیجه پیچیدگی زمانی بسیار مناسبی دارد چراکه درگیر اجرای الگوریتمهای وقتگیری شبیه "دایْجْکِسترا" نخواهند شد. به این نوع الگوریتمها به اختصار "الگوریتمهای DV"[۳] گفته می‌شود.

این روشها را با تفصیل بیشتری بررسی می‌کنیم ولی قبل از آن یکی از روشهای ایستا را که در برخی از موارد خاص کاربرد دارد ، معرفی می‌نماییم.

### ۳-۱) روش ارسال سیل‌آسا[۴]

این روش که برای ارسال بسته‌های همگانی (فراگیر[۵]) کاربرد دارد سریعترین الگوریتم برای ارسال اطلاعات به یک مقصد در شبکه به شمار می‌رود. طریقهٔ ارسال در این روش آنست که هر مسیریاب با دریافت این‌گونه بسته‌ها موظف است آنرا روی تمامی مسیرهای خروجی خود (به غیر از مسیری که بسته را از آن دریافت کرده است) ارسال نماید. در چنین حالتی این تضمین وجود دارد که اولاً هر بستهٔ اطلاعاتی به تمامی مسیریابهای زیرشبکه خواهد رسید. ثانیاً هر بسته در سریعترین زمان ممکن به مقصد می‌رسد.

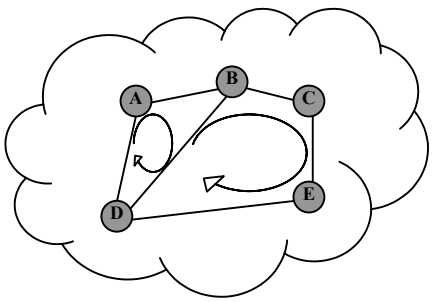
---

[۱] Dijkstra Shortest Path Algorithm
[۲] Link State Algorithms
[۳] Distance Vector Algorithms
[۴] Flooding Algorithm
[۵] Broadcast

از روش سیل‌آسا در موارد خاص و برای ارسال پیامهای فراگیر و کنترلی (مثل اعلام جداول مسیریابی) استفاده می‌شود؛ زیرا استفاده از این روش ، کل شبکه را در ترافیک زائد و بیهوده غرق خواهد کرد و بنابراین روشی قابل اتکا و عمومی برای مسیریابی نخواهد بود.

روش سیل‌آسا مشکل عمده‌ای دارد که باید رفع شود:

اگر قاعده براین باشد که همهٔ مسیریابها یک بستهٔ نوع فراگیر را روی تمامی خروجیهای خود ارسال نمایند ، ممکن است پس از چند لحظه خودشان آن بسته را دریافت کرده و چون مجدداً آنرا روی خروجی‌های خود ارسال می‌کنند این عمل تا بی‌نهایت ادامه خواهد یافت و در یک دور باطل کلّ شبکه از این بسته‌ها پر شده و روند ارسال هیچگاه متوقف نمی‌شود و عملاً شبکَه از کار خواهد افتاد. در شکل (۳-٤) حلقه‌های موجود در زیرساخت ارتباطی شبکه باعث شده است که روند تکرار بسته‌های فراگیر هیچگاه خاتمه نیابد.




شکل (۳-٤) حلقه‌های بینهایت در روش سیل آسا

برای رفع این مشکل دو راه حل وجود دارد:

- قرار دادن شمارهٔ شناسایی برای هر بسته[1]: در این روش برای هر بستهٔ فراگیر ، یک شمارهٔ منحصر به فرد و یکتا درج میشود و مسیریابی که بسته‌ای را یکبار دریافت کند شمارهٔ آنرا در جدولی ثبت می‌نماید. با دریافت یک بستهٔ فراگیر ، شمارهٔ شناسایی آنرا در جدول جستجو می‌کند و در صورت وجود ، آنرا حذف می‌نماید.
- قرار دادن طول عمر برای بسته‌ها : در این روش یک فیلد شمارنده در سرآیند بسته قرار داده می‌شود و به ازای عبور بسته از هر مسیریاب یکی از آن کم شده و وقتی این شماره به صفر رسید آن بسته از شبکه حذف خواهد شد.

[1] Selective Flooding

## ۲) الگوریتمهای LS

در الگوریتمهای LS هر مسیریاب باید پنج عمل زیر را انجام بدهد:

الف) مسیریابهای مجاور خود را که بصورت فیزیکی به آنها متصل است شناسایی کرده و آدرس آنها را بدست آورد.

ب ) تاخیر (یا بطور کلی هزینهٔ) مسیریابهای مجاور خود را اندازه گیری نماید.

ج ) یک بسته بسازد و تمام اطلاعاتی که از مسیریابهای مجاور خود دارد در آن قرار بدهد.

د ) بستهٔ ساخته شده را به روش "سیل‌آسا" برای تمامی مسیریابهای شبکه ارسال نماید و همچنین بسته‌هایی را که از مسیریابهای دیگر می‌رسد دریافت و ذخیره کند.

ه ) با استفاده از الگوریتمی مناسب ، بهینه‌ترین مسیر را بین هر دو مسیریاب در شبکه ، پیدا نماید.

هر یک از مراحل پنج‌گانهٔ فوق را بطور مجزا توضیح می‌دهیم:

### ۲-۱) شناسایی مسیریابهای مجاور

وقتی یک مسیریاب شروع به کار می‌کند (به عبارت ساده بوت می‌شود) اولین کاری که باید انجام بدهد شناسایی مسیریابهای مجاور خود می‌باشد. مسیریاب این کار را با ارسال یک بستهٔ خاص به نام "بستهٔ سلام" یا HELLO Packet روی تمامی خروجی‌های خود انجام می‌دهد. مسیریابهایی که از طریق یک کانال فیزیکی مستقیم به آن وصلند ضمن پاسخ به این بسته آدرس جهانی (آدرس IP ) خود را اعلام می‌نمایند. پس از دریافت بسته‌های پاسخ این اطلاعات در جدولی درج می‌شود.

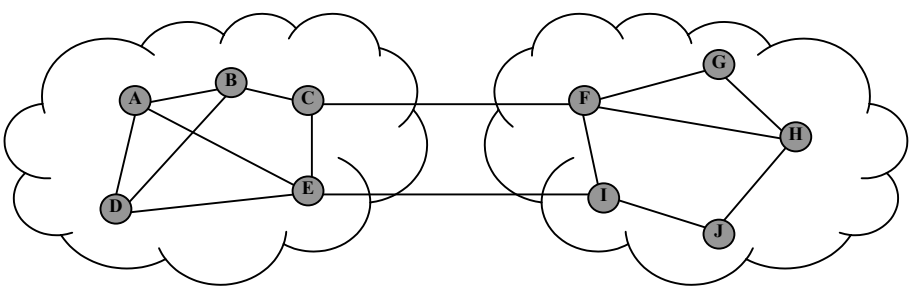
### ۲-۲) اندازه‌گیری هزینه

هر مسیریاب موظف است تاخیر هر یک از خطوط خروجی خود را اندازه‌گیری نماید؛ مسیریاب این کار را از طریق ارسال یک بستهٔ خاص به نام Echo Packet روی تمام خطوط خروجی خود انجام می‌دهد. تمام مسیریابهای گیرندهٔ این بسته ، با ارسال بسته‌ای به نام Echo Reply به فرستنده پاسخ می‌دهند. اگر مسیریاب موظف باشد که با دریافت بستهٔ Echo خارج از نوبت و به سرعت به آن پاسخ بدهد ، "زمان رفت و برگشت"[1] این بسته فقط تاخیر فیزیکی بین دو مسیریاب را به عنوان معیار هزینه مشخص می‌کند؛ مسیریاب این زمان را با استفاده از یک زمان‌سنج اندازه‌گیری کرده و آنرا بر ۲ تقسیم و در جدولی درج می‌نماید. در این حالت

[1] Round Trip Time

"میزان بار" و زمان انتظار بسته‌های به صف شده و منتظرِ پردازش در مسیریاب ، به حساب نمی‌آید که چندان صحیح نیست. مسیریاب می‌تواند بسته‌های Echo را همانند بسته‌های معمولی به انتهای صف بفرستد و پس از فرا رسیدن نوبت پردازشِ بسته ، به آن پاسخ بدهد؛ در این حالت معیار دقیقتری از تاخیر بدست می‌آید.

لازم به ذکر است که برای برخی از کانالهای فیزیکی مثل کانالهای ماهواره‌ای زمان تاخیر انتشار سیگنال بسیار زیاد و نامتعارف است چراکه زمان ارسال یک سیگنال به ارتفاع حدود ۳۷۰۰۰ کیلومتری[1] و برگشت آن به زمین چیزی حدود ۲۷۰میلی‌ثانیه طول میکشد. بنابراین ارسال یک بستهٔ Echo و برگشتِ پاسخ آن ، با چشمپوشی از زمان تاخیرِ پردازش ، چیزی حدود ۵۴۰ میلی ثانیه خواهد شد که در دنیای شبکه‌های کامپیوتری زمان بسیار زیادی محسوب می‌شود. بهترین حالت ، کانالهای فیبر نوری هستند که به ازای ۱۰۰۰کیلومتر حدود ۳ میلی ثانیه تاخیر دارند. بنابراین نمی‌توان از تاخیر فیزیکی انتشار سیگنال چشمپوشی کرد.
از طرف دیگر تاخیر پردازش برای مسیریابهایی که در برخی از زمانها با ترافیک بالایی روبرو می‌شوند ممکن است از چند ده میلی ثانیه تا چندین ثانیه! طول بکشد تا جایی که حتی طول عمر بسته به پایان رسیده و منجر به حذف آن شود. بنابراین زمان پردازش را نیز باید در معیار هزینه به حساب آورد ولی منظور کردن این پارامتر در معیار هزینه مشکلی دیگر پدید می‌آورد که برای رفع آن باید الگوریتم را پیچیده‌تر کرد. برای تشریح این مسئله به شکل (٤-٤) دقت کنید.




**شبکهٔ I** **شبکهٔ II**

شکل (٤-٤) دو شبکه متفاوت که با دو کانال ارتباطی به هم متصلند

[1] **به ارتفاع حدود ۳۷۰۰۰ کیلومتری از سطح زمین که ماهواره‌های مخابراتی در آن مدار قرار می‌گیرند مدار Geosynchrone گفته می‌شود.**

در شکل (٤-٤) تمام مسیریابها زمان تاخیر انتشار و میزان بار (زمان انتظار پردازش برای هر بسته) را به عنوان پارامتر هزینه در نظر می‌گیرند. فرض کنید تمامی مسیریابها با اجرای الگوریتم کوتاهترین مسیر ، مسیر بهینه از شبکهٔ I به شبکهٔ II را از طریق خط CF تشخیص بدهند و تمامی بسته‌ها را به این مسیر هدایت نمایند؛ چراکه این خط کمترین تاخیر و ترافیک را داشته است. با این کار در عرض چند ثانیه خط CF با ترافیک زیادی روبرو شده و تاخیر آن زیاد خواهد شد. با به‌هنگام شدن جدول مسیریابی ، کانال CF یک خط با تاخیر زیاد و "بد" گزارش می‌شود و از آن به بعد تمامی مسیریابها از خط EI بعنوان بهترین مسیر از شبکهٔ I به شبکهٔ II استفاده خواهند کرد. مجدداً خط EI با بار زیادی مواجه شده و به بدترین مسیر تبدیل می‌شود و این روند "نوسان"[1] تا بینهایت تکرار خواهد شد. این قضیه بسیار طبیعی است که اگر همهٔ مسیریابها از بهترین مسیر برای هدایت بسته‌ها استفاده کنند ، پس از مدت کوتاهی بهترین مسیر به بدترین مسیر تبدیل خواهد شد. برای اصلاح این مشکل میتوان یا از معیار زمان انتظار و پردازش چشمپوشی کرد یا آنکه روشی انتخاب کرد که درصدی از ترافیک بسته‌ها روی خطوط غیر بهینه توزیع شود. مثلاً وقتی مسیریاب C بهترین کانال را خط CF تشخیص میدهد ۷۰٪ از بسته ها را روی CF و ۳۰٪ باقیمانده را روی خط CE ارسال کند تا از طریق EI به سمت شبکهٔ II هدایت شوند. پیاده‌سازی چنین روشی پیچیدگی زمانی الگوریتم را افزایش می‌دهد.

### ۳-۲) تشکیل بسته‌های LS

هر مسیریاب موظف است پس از جمع‌آوری اطلاعات لازم از مسیریابهای مجاور خود بسته‌ای از این اطلاعات تشکیل بدهد؛ به این بسته ، "بستهٔ LS"[2] گفته می‌شود. در حالت کلّی این بسته باید شامل اطلاعات زیر باشد:

الف) آدرس جهانی مسیریاب تولیدکنندهٔ بسته

ب ) یک شمارهٔ ترتیب (تا بسته‌های تکراری از بسته‌های جدید تشخیص داده شوند.)

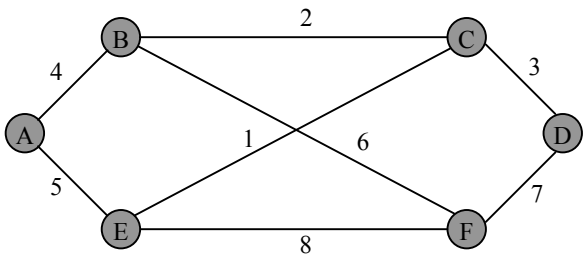
ج ) طول عمر بسته (تا اطلاعات بسته ، زمان انقضای اعتبار داشته باشد.)

د ) آدرس جهانی مسیریابهای مجاور و هزینهٔ تخمینی

برای روشن شدن قضیه به شکل (٤-٥) دقت کنید. در این شکل شش مسیریاب A تا F و خطوط ارتباطی مابین آنها و هزینه هر خط نشان داده شده است. قالب کلّی جدولی که هر

[1] Oscillation
[2] Link State Packet

مسیریاب باید بسازد در شکل (٦-٤) آورده شده است. ساختن این بسته‌ها مشکل نیست بلکه قسمت مشکل مسئله ، زمان تولید و توزیع آنها بر روی شبکه میباشد.
یک راه حل آنست که بسته‌ها در فواصل زمانی منظم برای مسیریابهای دیگر ارسال شود؛ در این حالت در بازه‌های زمانی مشخص ، کلّ شبکه برای لحظاتی غرق در بسته‌های LS خواهد شد. راه حل دوم آنست وقتی این جداول ارسال شوند که یک تغییر اساسی در زیرساخت شبکه رخ بدهد (مثل اضافه شدن یا حذف یک مسیریاب از شبکه).
وظیفهٔ فیلدهای "شماره ترتیب" و "طول عمر" در بخش بعدی مشخص شده است.




شکل (٥-٤) یک زیرساخت از یک شبکهٔ فرضی

| A | |
|---|---|
| Seq. | |
| Age | |
| B | 4 |
| E | 5 |

| B | |
|---|---|
| Seq. | |
| Age | |
| A | 4 |
| C | 2 |
| F | 6 |

| C | |
|---|---|
| Seq. | |
| Age | |
| B | 2 |
| D | 3 |
| F | 1 |

| D | |
|---|---|
| Seq. | |
| Age | |
| C | 3 |
| F | 7 |

| E | |
|---|---|
| Seq. | |
| Age | |
| A | 5 |
| C | 1 |
| F | 8 |

| F | |
|---|---|
| Seq. | |
| Age | |
| B | 6 |
| D | 7 |
| E | 8 |

شکل (٦-٤) بسته‌های LS

### ۴-۲) توزیع بسته‌های LS روی شبکه

یکی از مسائل مهم در این نوع الگوریتمِ مسیریابی ، چگونگی توزیع بسته‌های LS روی شبکه است. قاعدهٔ کلی ارسال بسته‌های LS ، ارسال به روش "سیل‌آسا" است. برای آنکه این بسته‌ها در یک حلقهٔ بینهایت تکرار نشوند ، هر بسته دارای یک شمارهٔ ترتیب است که با ورود

به هر مسیریاب ابتدا بررسی می‌شود که آیا این بسته قبلاً دریافت شده است یا آنکه یک بستهٔ جدید است. در صورت جدید بودن ، مسیریاب ضمن درج اطلاعات درون بسته در یک جدول موقت ، آنرا روی تمامی خروجی‌های خود (به غیر از کانالی که بسته را از آن دریافت کرده) ارسال می‌نماید. بنابراین هر مسیریاب موظف است که شمارهٔ ترتیب بسته‌های LS خود را که در هر مرحله ارسال می‌کند ، حفظ نماید و در هر بار ارسال ، یکی به آن اضافه کرده تا در مراحل بعدی ، بسته‌های LS با شمارهٔ جدید و غیر تکراری ارسال شود. شمارهٔ ترتیب معمولاً ۳۲ بیتی است و اگر شمارهٔ بسته‌ها از صفر شروع و به فرض در هر ثانیه یکی به آن اضافه شود تکرار شمارهٔ یک بسته ، ۱۳۷ سال طول می‌کشد!

وقتی مسیریاب بسته‌ای را با یک شمارهٔ خاص دریافت کرد دیگر بسته‌های با شمارهٔ کوچکتر از آن شماره را دریافت نخواهد کرد. بعنوان مثال اگر مسیریابی یک بستهٔ LS را از یک مسیریاب خاص با شمارهٔ ترتیب ٦٥٤٣٢٠ دریافت کند و آنرا در جدولش درج نماید ، از آن به بعد بسته‌هایی را که شمارهٔ کمتر از ٦٥٤٣٢١ داشته باشد ، از آن مسیریاب قبول نخواهد کرد. همین مسئله مشکل بزرگی برای مسیریابهایی که به ناگاه از کار می‌افتند و می‌خواهند مجدداً به شبکه وارد شوند ، پدید خواهد آورد؛ چونکه یک مسیریاب که از شبکه خارج شده و می‌خواهد مجدداً به شبکه وارد شود مجبور است شمارهٔ بسته‌ها را از صفر شروع کند و هیچ مسیریابی آنرا نخواهد پذیرفت. برای رفع این مشکل برای هر بسته یک طول عمر در نظر گرفته می‌شود که به ازای هر ثانیه یک واحد از آن کم می‌شود؛ هرگاه مسیریابی بسته‌ای را دریافت و آنرا در جدولی درج نماید ولی در خلال طول عمر بسته ( مثلاً ۱۰ دقیقه معادل ٦٠٠ ثانیه ) بستهٔ جدیدی دریافت نکند آن مسیریاب از جدول مسیریابی حذف خواهد شد. با این قاعده وقتی مسیریابی که از جدول حذف شده ، بخواهد به شبکه برگردد بسته‌های ارسالی از طرف او با هر شماره‌ای دریافت خواهد شد.با این روش مشکل دریافت بسته‌های تکراری حل خواهد شد.

مسئلهٔ بحرانی دیگر آنست که اگر یک مسیریاب به هر دلیلی اطلاعات غلط ارائه کند ، کلّ مسیریابی شبکه با اشکال مواجه شده و تمام جداول مسیریابی با اطلاعات آلوده تنظیم خواهد شد. این مشکل زمانی بروز می‌کند که یک مسیریاب به اشتباه اعلام کند که کانالی فیزیکی با مسیریاب دیگر دارد در حالی که نداشته باشد؛ یا کانالی فیزیکی با یک مسیریاب داشته باشد ولی اعلام نکند. گاهی این مشکلات بصورت عمدی توسط اخلالگران بوجود می‌آید چراکه ایجاد بسته‌های LS بصورت مصنوعی و تزریق آن به شبکه توسط کاربران فقط نیاز به برنامه‌نویسی دارد. برای رفع این مشکل ، در مسیریابهای مدرن بسته‌های LS زمانی پذیرفته می‌شود که قبل از آن ، هویت ارسال کنندهٔ بسته احراز شود.

### ۵-۲) محاسبهٔ مسیرهای جدید

وقتی یک مسیریاب بسته‌های LS را از تمامی مسیریابهای شبکه دریافت کرد ، می‌تواند ساختمان دادهٔ گراف زیرشبکه را تشکیل داده و بر اساس آن اقدام به پیدا کردن بهترین مسیرها (یعنی با کمترین هزینه) بین هر دو گره بنماید.

بگونه‌ای که اشاره شد برای یافتن بهترین مسیر بین دو گره در یک گراف ، می‌توان از الگوریتم دایجْکِسترا بهره برد. معرفی این الگوریتم در اینجا خالی از لطف نیست چراکه بسیاری از مسیریابهای مدرن از آن استفاده می‌کنند؛ این الگوریتم در سال ۱۹۵۹ معرفی شد.

فرض کنید الگوریتم بخواهد بهترین مسیر بین گره‌های V1 و V2 را پیدا کند.

- گراف زیرشبکه را تشکیل داده و گره‌های V1 و V2 را مشخص کنید. فرض کنید گراف زیرشبکه با استفاده از "ماتریس همجواری"[1] نشان داده شود. در این ماتریس عنصر a[i.j] هزینهٔ رسیدن از گره Vi به گره Vj می‌باشد. اگر هیچ مسیر مستقیمی بین گره Vi و گره Vj وجود نداشته باشد مقدار a[i.j] ، بینهایت در نظر گرفته می‌شود.

```
int dist[MAX_NODES] [MAX_NODES];
```

- برای هر گره از گراف یک "رکورد حالت" ایجاد کنید که شامل سه فیلد باشد:
  - فیلد اول نشان‌دهندهٔ گره قبلی در مسیر است؛ این فیلد را predecessor می‌نامیم.
  - فیلد دوم نشان‌دهندهٔ مجموع هزینهٔ رسیدن از گره V1 به این گره می‌باشد؛ این فیلد را length می‌نامیم.
  - فیلد سوم حالت گره را مشخص می‌کند که می‌تواند دو حالت "دائمی" (Permanent) و "موقت" (Tentative) داشته باشد. این فیلد را "برچسب" (label) می‌نامیم.

این رکوردها را می‌توان در زبان C بصورت زیر تعریف کرد:

```
struct state {
    int predecessor;
    int length;
    enum {permanent, tentative} label;
    } State[MAX_NODES];
```

- برای تمامی گره‌ها ، متغیرهای حالت را بصورت زیر مقداردهی اولیه نمایید:

```
State[i].prdecessor=NULL;
State[i].length=INFINITY;
```

[1] Adjacency Matrix

```
State[i].label=tentative;
```

- گره t=V1 را به عنوان نقطهٔ کار انتخاب نمایید.
- (*) برچسب گره **t** را به صورت دائمی درآورید. گره‌ای که برچسب آن دائمی است دیگر هیچگاه برچسب آن تغییر نخواهد کرد.
- برای تمامی "گره‌های Vi مجاور با **t** که برچسبشان موقت" است ، رکوردهای حالت باید طبق رابطهٔ زیر تغییر کنند.

```
if (State[t].length+dist[t][i]<State[i].length) {
    State[i].prdecessor=t;
    State[i].length= State[t].length+dist[t][i];
    }
```

یعنی برای تمامی گره‌های Vi مجاور با **t** که برچسبشان موقت است ، بررسی می‌شود که اگر هزینهٔ رسیدن از V1 به Vi از طریق گره **t** کمتر از هزینهٔ مسیر قبلی آنها به V1 است ، اصلاح لازم روی رکورد حالت آن گره انجام شود.

- از بین تمامی گره‌هایی که برچسبشان موقتی است گره‌ای که کمترین هزینه را تا V1 دارد پیدا کرده و بعنوان نقطهٔ کار t انتخاب کنید.
- اگر **t** به گره مقصد (V2) نرسیده است به مرحلهٔ (*) برگردید.
- اگر **t** به گره مقصد رسید ، از آن شروع کرده و گره ماقبل آنرا از رکورد حالت استخراج کرده و اینکار را تکرار کنید تا به گره مبدأ برگردید.

برای درک راحتتر الگوریتم فوق به مثال شکل (۷-٤) دقت کنید. در این مثال فرض شده است می‌خواهیم بهترین مسیر از A به D را پیدا کنیم.

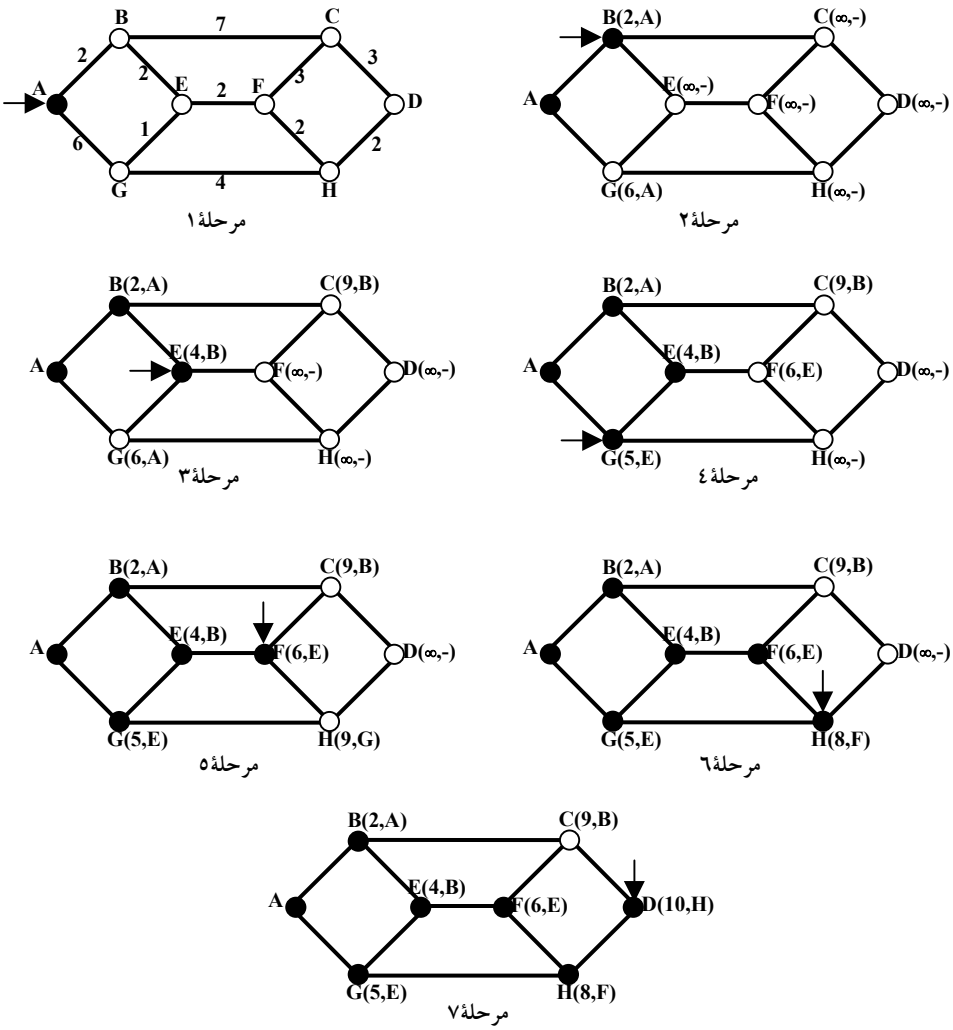
در مرحلهٔ ۱ گراف زیرساخت ارتباطی یک شبکهٔ فرضی به همراه هزینهٔ هر ارتباط به تصویر کشیده شده است. گره شروع به عنوان نقطهٔ کار انتخاب شده و حالت آن بصورت "دائمی" علامت خورده است. (گره‌های دائمی با دایرهٔ توپر نشان داده شده است)

در مرحلهٔ ۲ هزینهٔ گره‌های مجاور A (یعنی B و G ) محاسبه شده و رکورد حالت آنها اصلاح شده است. سپس از بین تمام گره‌های با حالت "موقت" ، گره B که کمترین هزینه را تا A داشته به عنوان گره نقطهٔ کار انتخاب و حالت آن بصورت دائمی علامت زده شده است.

در مرحلهٔ ۳ هزینهٔ گره‌های مجاور B که علامت موقتی دارند (یعنی C و E ) تا A محاسبه شده و رکورد حالت آنها اصلاح شده است. از بین تمام گره‌های با حالت "موقت" ، گره E که

کمترین هزینه را تا A داشته به عنوان گره نقطهٔ کار انتخاب و حالت آن بصورت دائمی علامت زده شده است.

در مرحلهٔ ٤ هزینهٔ گره‌های مجاور E که علامت موقتی دارند (یعنی G و F ) تا A محاسبه شده و رکورد حالت آنها اصلاح شده است. (به چگونگی اصلاح رکورد حالت در گره G در این مرحله دقت کنید) از بین تمام گره‌های با حالت "موقت" ، گره G که کمترین هزینه را تا A داشته به عنوان گره نقطهٔ کار انتخاب و حالت آن بصورت دائمی علامت زده شده است.




شکل (۷-٤) مراحل پیدا کردن کوتاهترین مسیر از A به F

در مرحلهٔ ۵ هزینهٔ گره‌های مجاور G که علامت موقتی دارند (یعنی H ) تا A محاسبه و رکورد حالت آن اصلاح شده است. از بین تمام گره‌های با حالت "موقت" ، گره F که کمترین هزینه را تا A داشته به عنوان گره نقطهٔ کار انتخاب و حالت آن بصورت دائمی علامت زده شده است. ( دقت کنید که F مجاور G نیست )

در مرحلهٔ ٦ هزینهٔ گره‌های مجاور F که علامت موقتی دارند (یعنی C و H ) تا A محاسبه و رکورد حالت گره H اصلاح شده است. از بین تمام گره‌های با حالت "موقت" ، گره H که کمترین هزینه را تا A داشته به عنوان گره نقطهٔ کار انتخاب و حالت آن بصورت دائمی علامت زده شده است.

در مرحلهٔ ۷ هزینهٔ گره‌های مجاور H که علامت موقتی دارند (یعنی D ) تا A محاسبه و رکورد حالت آن اصلاح شده است. از بین تمام گره‌های با حالت "موقت" ، گره D که کمترین هزینه را تا A داشته به عنوان گره نقطهٔ کار انتخاب می‌شود ولی چون D گره مقصد است الگوریتم در این مرحله پایان می‌یابد. برای پیدا کردن مسیر ، از رکورد حالتِ گره D شروع کرده و گره قبلی آن را پیدا می‌کنیم ، از این گره مجدداً گره قبلیِ آن پیدا می‌شود و این کار ادامه می‌یابد تا به نقطهٔ شروع برسیم.

در جدول (۷-٤) زیربرنامهٔ کوتاهترین مسیر بر طبق الگوریتم دایْجْکِسترا به زبان C ارائه شده است.

برای شبکه‌هایی که دارای n مسیریاب و هر مسیریاب دارای حداکثر k کانال ورودی/خروجی است ، در بدترین حالت به فضای n×k رکورد اطلاعاتی برای ذخیره‌سازی جداول LS ، نیاز خواهد بود که برای شبکه‌های وسیع با هزاران مسیریاب ، مشکل‌ساز خواهد بود.

در بسیاری از مسیریابهای مدرن از روشهای مبتنی بر الگوریتم LS ، استفاده شده است. برخی از این پروتکلها عبارتند از:

- پروتکل OSPF[1] در برخی مسیریابهای تجاری CISCO
- پروتکل IS-IS[2] که توسط DECnet ارائه شده است.
- پروتکل NLSP که توسط شرکت Novell معرفی و ارائه شده است.
- پروتکل CLNP که توسط ISO معرفی و ارائه شده است.

در ادامه الگوریتمهای DV را معرفی خواهیم کرد.

[1] Open Shortest Path First
[2] Intermediate System-Intermediate System

```
 #define MAX_NODES 100  /*تعریف تعداد حداکثر گره‌های گراف*/
#define INFINITY 1000000000 /*تعریف یک عدد بسیار بزرگ به عنوان هزینهٔ بی‌نهایت */
int n, dist[MAX_NODES][ MAX_NODES]; /*تعریف ماتریس همجواری */
                                    /* n تعداد گره‌های گراف می‌باشد */
void shortest_path(int s, int t, int path[])
    { struct state {
          int predecessor; /*گره قبلی در مسیر*/
          int length; /*هزینهٔ گره تا مبدأ */
          enum {permanent, tentative} label; /*حالت گره */
          } State[MAX_NODES]; /*تشکیل رکوردهای حالت برای هر گره درون یک آرایه*/
int  i, k, min;
struct state *p;
for(p  = &State[0];p<&State[n];p++) { /*مقداردهی اولیه به رکوردهای حالت */
      p->predecessor=NULL;
      p->length=INFINITY;
      p->label=tentative;
}
State[t].length=0;
State[t].label=permanent;
k=t;  /* k گره نقطهٔ کار برای شروع می‌باشد */
do {
    for(i=0;i<n;i++)
      if(dist[k][i]!=0 && State[i].label==tentative){
         if(State[k].length+dist[k][i]<State[i].length){
                                          /*آیا مسیر بهتری از گره فعلی به گره مبدا وجود دارد؟*/
           State[I].predecessor=k;
           State[i].length= State[k].length+dist[k][i];
        }
      }
k=0;   min=INFINITY; /*یافتن گره‌ای از بین گره‌های با علامت "موقتی" که کمترین هزینه را دارد */
    for(i=0;i<n;i++)
      if(State[i].label==tentative && State[i].length<min){
      min= State[i].length;
      k=i;
      }
      State[k].label=permanent;
      } while (k!=s);
/* path[] قرار دادن مسیر بهینه از آخر به اول در آرایهٔ */
      i=0;  k=s;
      do {
      path[i++]=k; k=State[k].predecessor;
      } while (k>=0);
      }
```

جدول (۷-٤) الگوریتم دایجکسترا برای پیدا کردن بهترین مسیر بین دو گره در یک گراف

## ۳) الگوریتمهای DV[1]

یکی از روشهای پویا در مسیریابی ، روش "بردار فاصله" یا DV است که در سالهای اولیه راه اندازی شبکهٔ ARPA مورد استفاده قرارگرفت و پس از عمومی شدن اینترنت تحت نام پروتکل RIP عرضه شد و هنوز هم در مسیریابهای کوچک مورد استفاده قرار می‌گیرد و در نسخه های جدید ویندوز NT پشتیبانی می‌شود . نامهای متفاوتی برای این روش ارائه شده که همهٔ آنها از یک الگوریتم استفاده می‌کنند. این نامها عبارتند از:

- پروتکل RIP
- الگوریتم مسیریابی Bellman–Ford
- الگوریتم مسیریابی Ford-Fulkerson
- الگوریتم Distance Vector Routing

در این روش بر خلاف الگوریتم LS ، هر مسیریاب بدون آنکه اطلاعی از هزینهٔ خطوط ارتباطی در زیرشبکه داشته باشد ، جدولی را در حافظه خود نگه می‌دارد که جدول مسیریابی[2] نام دارد. در این جدول به ازای هر مسیریاب در زیرشبکه یک رکورد وجود دارد؛ هر رکورد دارای دو فیلد مجزا است:

الف: **فیلد مسیر** : این فیلد خط خروجی مناسب برای رسیدن به یک مسیریابِ خاص در شبکه را مشخص می‌کند.

ب: **فیلد مقدار تقریبی هزینه** : این فیلد هزینهٔ تقریبی رسیدن یک بسته تا مسیریابِ مقصد را مشخص می‌نماید.

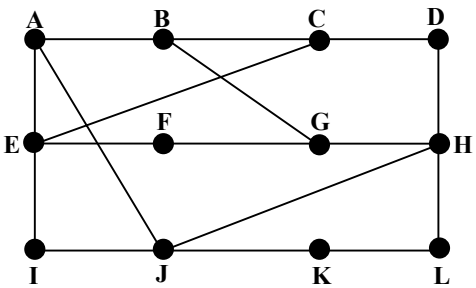
برای روشن شدن قضیه به شکل (۸–٤) که زیرساخت ارتباطی یک شبکهٔ فرضی را نشان می‌دهد ، دقت کنید. در این مثال تعداد ۱۲ مسیریاب با نامهای A تا L زیرساخت ارتباطی شبکه را تشکیل داده‌اند؛ کانالهایی که بین این ۱۲ مسیریاب وجود دارد در شکل مشخص است ولی هیچیک از مسیریابها اطلاعی از هزینهٔ هر یک از خطوط ارتباطی ندارند ، بهمین دلیل مقادیر هزینهٔ هر خط در شکل نشان داده نشده است. جدول (۹–٤) جدول مسیریابی مربوط به J است و هر سطر نشان می‌دهد که اگر J بخواهد بسته‌ای را به یک مسیریاب دیگر بفرستد از چه خطی باید استفاده کند. بعنوان مثال اگر مسیریاب J خواست برای G بسته‌ای ارسال کند ، با

[1] در برخی از کتابها ، الگوریتمهای DV به "الگوریتم بردار فاصله" ترجمه شده است.

[2] Routing Table

مراجعه به این جدول نتیجه می‌گیرد که باید آنرا به سمتِ H ارسال نماید و هزینهٔ تقریبی رسیدن بسته به G تقریباً ۱۸ است. حال شاید بپرسید در این روش معیار هزینه و واحد آن چیست؟ پاسخ آنست که معیار هزینه میتواند تاخیر یا "تعداد گام" (Hop) در نظر گرفته شود؛ در چنین حالاتی واحد تاخیر ، میلی‌ثانیه و واحد گام ، "تعداد" خواهد بود. در این مثال معیار هزینه ، زمانِ تاخیر بر حسب میلی‌ثانیه انتخاب شده است.

تا اینجا به این نکته اشاره کردیم که هر یک از مسیریابها جدولی در حافظه خود تشکیل می‌دهند ولی سوال اصلی اینجاست که این جداول چگونه ایجاد و به‌هنگام می‌شوند. چرا که در زمانهای متفاوت شرایط ترافیکی و توپولوژیکی شبکه عوض شده و بالطبع این جداول باید با زمان تغییر داده شود تا همیشه بهترین وضعیت را برای مسیریابی ارائه بدهد.




شکل (۸-٤) زیرساخت ارتباطی یک شبکهٔ فرضی با دوازده مسیریاب

| | هزینهٔ تقریبی | خط |
|---|---|---|
| A | 8 | A |
| B | 20 | A |
| C | 28 | I |
| D | 20 | H |
| E | 17 | I |
| F | 30 | I |
| G | 18 | H |
| H | 12 | H |
| I | 10 | I |
| J | 0 | — |
| K | 6 | K |
| L | 15 | K |

جدول (۹-٤) جدول مسیریابی مربوط به مسیریاب J

در روش DV اصول کار بصورت زیر خلاصه می‌شود:

- هر مسیریاب موظف است هزینهٔ خطوطی را که بصورت فیزیکی با مسیریابهای دیگر دارد ، محاسبه کرده و در جدول خود درج نماید. هزینهٔ خطوطی که مسیریاب با آنها در ارتباطِ مستقیم نیست ، در این جدول بینهایت در نظر گرفته می‌شود.
- هر مسیریاب موظف است در بازه‌های زمانی مشخص ، ستون هزینه از جدول مسیریابی خودش را برای مسیریابهای مجاور ارسال نماید، ("یعنی فقط برای مسیریابهائی که با آن در ارتباط است نه تمام مسیریابها "). بنابراین هر مسیریاب در فواصل T ثانیه‌ای ، اطلاعاتی را از مسیریابهای مجاور دریافت می‌کند که جدید است و می‌تواند بر اساس آن ، جدول مسیریابی خود را به‌هنگام کند.
- هر مسیریاب موظف است پس از دریافت جداول مسیریابی از مسیریابهای مجاور ، جدول خود را طبق یک الگوریتم بسیار ساده به‌هنگام نماید. ( این الگوریتم با یک مثال ، تشریح خواهد شد )

برای مشخص شدن چگونگی به‌هنگام شدن جداول مسیریابی ، به مثال قبلی مراجعه کنید. فرض کنید مسیریابهای مجاور J ( یعنی مسیریابهای A ، I ، H ، K ) جداول مسیریابی خود را برای J ارسال کرده باشند. این جداول در شکل (۱۰-٤) نشان داده شده‌اند.

مسیریاب J می‌تواند تخمین بزند که از لحاظ زمانی تا مسیریابهای A ، I ، H و K چقدر تاخیر وجود دارد . این تخمین بسادگی از طریق ارسال بسته های Echo و دریافت پاسخ آن ومحاسبهٔ زمان رفت و برگشت آن امکانپذیر است . فرض کنید با این روش مسیریاب J مقادیر تاخیر را بصورت زیر ارزیابی کرده باشد:

- J تا A ← ۸ میلی‌ثانیه
- J تا I ← ۱۰ میلی‌ثانیه
- J تا H ← ۱۲ میلی‌ثانیه
- J تا K ← ٦ میلی‌ثانیه

حال فرض کنید پس از رسیدن چهار جدول فوق به J و اندازه گیری J از مقدار تاخیر تا مسیریابهای A ، I ، H و K ، مسیریاب J بخواهد در جدول مسیریابی خود ، بهترین کانال را برای ارسال بسته به هر یک از مسیریابهای A تا L بیابد. بعنوان مثال J می‌خواهد بداند بهترین مسیر برای رسیدن به G کدام است . ابتدا به جدول رسیده از A مراجعه می‌کند؛ A ادعا کرده است که برای رسیدن به G ، تاخیری معادل ۱۸ میلی‌ثانیه دارد. پس اگر J بخواهد ازطریق A بسته‌ای برای G بفرستد معادل ۲٦ میلی‌ثانیه تاخیر خواهد داشت؛ (یعنی ۸ میلی‌ثانیه از J به A و ۱۸ میلی‌ثانیه از A به G بنابر ادعای A ). J این مقدار را بطور موقتی در حافظه ذخیره می‌نماید.

| | A | I | H | K |
|---|---|---|---|---|
| A | 0 | 24 | 20 | 21 |
| B | 12 | 36 | 31 | 28 |
| C | 25 | 18 | 19 | 36 |
| D | 40 | 27 | 8 | 24 |
| E | 14 | 7 | 30 | 22 |
| F | 23 | 20 | 19 | 40 |
| G | 18 | 31 | 6 | 31 |
| H | 17 | 20 | 0 | 19 |
| I | 21 | 0 | 14 | 22 |
| J | 9 | 11 | 7 | 10 |
| K | 24 | 22 | 22 | 0 |
| L | 29 | 33 | 9 | 9 |

شکل (۱۰-٤) جداول ارسالی توسط مسیریابهای مجاور J

حال به جدول رسیده از I مراجعه می‌کند. I ادعا کرده است که تا G معادل ۳۱ میلی‌ثانیه تاخیر دارد ، بنابراین اگر J بخواهد از طریق I به G بسته ای بفرستد معادل ٤۱ میلی‌ثانیه تاخیر خواهد داشت. (۱۰ میلی‌ثانیه از J به I و ۳۱ میلی ثانیه از I به G ) این مقدار نیز بطور موقتی در حافظه ذخیره می‌شود.

در جدول رسیده از H ، تاخیر زمانی تا G معادل ٦ میلی‌ثانیه ادعا شده ، بنابراین اگر J بخواهد از طریق H به G بسته ای بفرستد معادل ۱۸ میلی‌ثانیه تاخیر خواهد داشت. (۱۲ میلی‌ثانیه از J به H و ٦ میلی ثانیه از H به G )

بهمین ترتیب هزینه رسیدن به G از طریق K معادل ۳۷ (۳۱+٦) میلی‌ثانیه محاسبه و در حافظه ذخیره می‌شود .

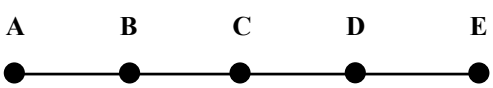
با محاسبات فوق J می‌تواند با پیدا کردن حداقل مقدار از بین مقادیر محاسبه شده ، بهترین مسیر برای ارسال یک بسته به G را پیدا کند. در مثال فوق از بین چهار مقدار ذخیره شده در حافظه ، هزینهٔ ارسال از طریق مسیریابِ H حداقل خواهد بود و تاخیری معادل ۱۸ میلی‌ثانیه خواهد داشت. مسیریاب J در جدول خَود در رکورد متناظر با G آدرس مسیریاب H و هزینهٔ ۱۸ را درج می‌کند.

برای تمام مسیریابهای دیگر این روند تکرار و بهترین مسیر برای رسیدن به یکایک آنها محاسبه شده و در جدول جدید درج خواهد شد.

این جدول تا زمان به‌هنگام‌سازی بعدی ، که جداول جدید از مسیریابهای مجاور می‌رسند قابل استناد بوده و بهترین مسیر را برای ارسال بسته‌ها تعیین می‌کند. بعنوان مثال برای ارسال بسته‌ای به مسیریاب F با استناد به جدول محاسبه شده ، باید آن بسته تحویل مسیریاب I شود.

این روش در عین سادگی، پویا است و تغییرات ترافیکی شبکه با زمان را در جداول مسیریابی دخالت خواهد داد. از طرفی حجم جدولی که بایستی هر مسیریاب در حافظه خود نگه دارد از درجه ۱ است یعنی به ازای n مسیریاب فقط n رکورد کافی است در حالی که برای نگهداری جداول مسیریابی در روش LS درجهٔ ۲ است.( یعنی در بدترین حالت به یک ماتریس با n×n رکورد نیاز می‌باشد)

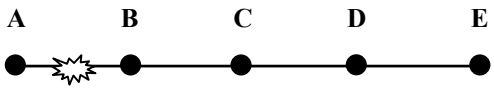
پروتکلهای DV از یک مشکل عمده رنج می‌برند و آن "عدم همگرایی سریع جداول مسیریابی" در هنگام خرابی یک مسیریاب یا یک کانال ارتباطی می‌باشد. این مشکل "شمارش تا بینهایت"[1] نام گرفته است. این اشکال زمانی پیش خواهد آمد که یکی از مسیریابها دچار خرابی شود یا آنکه مسیر ارتباطی او با دیگران قطع شود . بعنوان مثال شکل (۱۱-٤) را در نظر بگیرید؛ در این شکل ، A تا E مسیریابها هستند و هر کدام برای رسیدن به دیگری فقط یک مسیر در اختیار دارند. بعنوان مثال هزینه A تا B ، ۱ میلی ثانیه و از A تا E ، ٤ میلی ثانیه است. در حالت عادی جداول مسیریابی هر یک از مسیریابها ، طبق جداولِ شکل (۱۱-٤) تنظیم خواهد شد.




| | | | | | |
|---|---|---|---|---|---|
| A | 0,- | 1,A | 2,B | 3,C | 4,D |
| B | 1,B | 0,- | 1,B | 2,C | 3,D |
| C | 2,B | 1,C | 0,- | 1,C | 2,D |
| D | 3,B | 2,C | 1,D | 0,- | 1,D |
| E | 4,B | 3,C | 2,D | 1,E | 0,- |

شکل (۱۱-٤) زیرساخت ارتباطی از یک شبکهٔ فرضی و جداول مسیریابی

[1] Count to infinity




| | | | | | |
|---|---|---|---|---|---|
| هزینهٔ رسیدن به A در هنگام بروز خرابی | A | ∞,A | 2,B | 3,C | 4,D |
| هزینهٔ رسیدن به A پس از اولین بهنگام‌سازی | A | 3,C | 2,B | 3,C | 4,D |
| هزینهٔ رسیدن به A پس از دومین بهنگام‌سازی | A | 3,C | 4,B | 3,C | 4,D |
| هزینهٔ رسیدن به A پس از سومین بهنگام‌سازی | A | 5,C | 4,B | 5,C | 4,D |
| هزینهٔ رسیدن به A پس از چهارمین | A | 5,C | 6,B | 5,C | 6,D |
| هزینهٔ رسیدن به A پس از پنجمین بهنگام‌سازی | A | 7,C | 6,B | 7,C | 6,D |
| هزینهٔ رسیدن به A پس از ششمین بهنگام‌سازی | A | 7,C | 8,B | 7,C | 8,D |
| هزینهٔ رسیدن به A پس از بهنگام‌سازی n ام | A | …. | …. | …. | …. |
| ∞ | A | ∞ | ∞ | ∞ | ∞ |

شکل (۱۲-٤) عدم همگرایی سریع جدول مسیریابی در هنگام وقوع یک خرابی

حال فرض کنید ناگهان خط ارتباطی A بـه B قطـع شـود. بنـابراین در ایـن حـالت هزینـهٔ مسیریاب B به A بینهایت خواهد شد و هزینهٔ ارسال بسته به A در جداول هر یک از مسیریابها ، طبق روند شکل (۱۲-٤) تغییر خواهد کرد. دقت کنید که پس آنکه ارتباط B با A قطع شد ، B در جدول خود ، هزینهٔ رسیدن به A را مقدار ∞ درج می‌کند ولـی پـس از گذشـت T ثانیـه جدول مسیریابی از C می‌رسد و C اعلام کرده که مقدار هزینه تـا A مقـدار ۲ اسـت زیـرا C نمی‌داند که کانال B به A قطع شده است و در نتیجه همان مقدار قبل از وقوع خرابی را اعـلام می‌کند. B که در جدولش مقدار هزینه تا A را بینهایت درج کرده است به خیال آنکه C بـه A مسیری مجزا دارد ( با هزینه ۲ ) در جدول خـود هزینـه رسـیدن بـه A را از ∞ بـه ۳ تبدیـل می‌کند. (۱ واحد تا C و ۲ واحد از C به A طبق ادعای C) بنابراین B فرض کرده که از C به A کانالی مستقل وجود دارد و هزینه آنهم ۲ است در حالی که چنین فرضـی اشـتباه اسـت. در T ثانیه بعد مجدداً جداول مسیریابی بین همسایه ها رد و بدل می‌شود . C جـدول B را گرفتـه و متوجه می‌شود که هزینه رسیدن به A مقدار ۳ شده است بنـابراین بـا احتسـاب مقـدار جدیـد هزینه رسیدن از C به A در جدولش ٤ می‌شود . در T ثانیه بعد ، مسیریاب B با دریافت جدول مسیریابی C ، باز هم به اشتباه هزینه رسیدن بـه A را در جـدول خـودش ۵ درج می‌کنـد. (۱ واحد تا C + ٤ واحد از C به A طبق ادعای C). این روند تا بی نهایت ادامه دارد و بطور مداوم بین مسیریابها اطلاعات غلط در مورد A مبادله می‌شود . مشکل از آن جائی است کـه C ، D و E نمی‌دانند که تنها مسیرشان به A از طریق B است و به B اعلام می‌کنند که راهی به A دارنـد

و هزینهٔ آن را اعلام می‌کنند ، در حالیکه که تمامی این راهها همانی است که فعـلاً قطـع شـده است !

روشهای گوناگونی برای حل این مسأله پیشنهاد شده که عمدتاً پیچیده‌اند یا مقـرون بـه صرفـه نیستند. ساده تریـن راه‌حـل آن اسـت کـه وقتـی یـک مسـیریاب می‌خواهـد اطلاعـاتی را بـه همسایه‌هایش بدهد هزینه رسیدن به آنهایی را که قطعاً باید از همان مسیریاب بگذرند را اعلام نمی‌کند. (یا ∞ اعلام می‌کنند) بعنوان مثـال C چـون می‌دانـد مسـیر A از B مـی‌گـذرد وقتـی خواست جدول مسیریابی خود را به B اعلام کنـد هزینـه رسـیدن بـه A را همیشـه ∞ اعـلام می‌کند چرا که برای رسیدن به A قطعاً باید از B عبور کرد؛ در ایـن حـالت جـداول مسـیریابی سریعاً اصلاح خواهد شد.[1]

الگوریتمهای DV برای مسیریابی در یک شبکهٔ کوچک ( با حداکـثر ۳۰ مسـیریاب ) هنـوز کاربرد دارد ولی مسیریابهای جدید ، به دلیل نقص یاد شده و روشهای بهتری که ابداع شـده‌اند به سمت روشهای اصلاح شده تری مثل روش OSPF رفته‌اند . ایـن روشـها سـریعتر و دقیقـتر جداول مسیریابی را نسبت به تغییرات توپولوژیکی و ترافیکی شبکه تنظیم می‌کنند و در ضمـن می‌توانند بار ترافیک را روی چند مسیر که نزدیک به بهینگی هستند تقسیم نمایند چرا که اگـر همه بار روی بهترین مسیر ارسال شود در اندک زمانی بهترین تبدیل به بدترین مسـیر خواهـد شد.

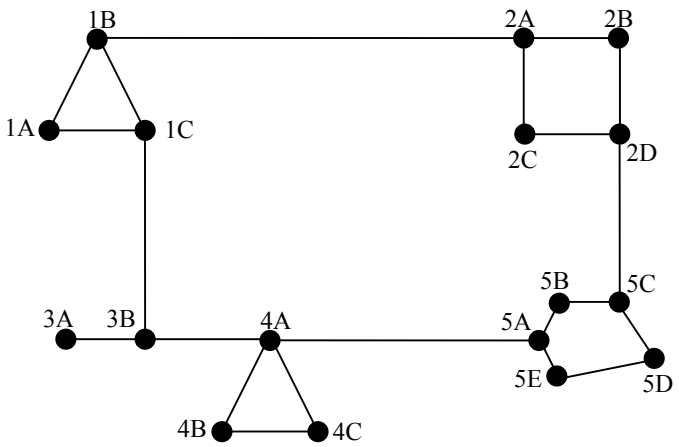
در بخشهای آتی باز هم به سراغ نسخه‌های کـاربردی و پیاده‌سـازی شـدهٔ ایـن الگوریتمـها خواهیم آمد.

## ۴) مسیریابی سلسله‌مراتبی[2]

در الگوریتمهای مسیریابی LS و DV هر مسیریاب باید جدولی را به‌عنوان جدول مسیریابی تشکیل بدهد. وقتی یک شبکه رشد می‌کند و شـبکه‌های محلـی و مسـیریابها اضافـه می‌شـوند حجم جداول مسیریابی و زمان لازم برای تعیین مسیر یک بسته افزایش پیدا می‌کند؛ تا جاییکـه ممکن است این زمان به تاخیرهای بحرانی بیانجامد و عملاً کـارآیی شـبکه کـاهش چشـمگیر داشته باشد. در شکل (۱۳-٤) به زیرساخت ارتباطی از یک شبکهٔ فرضی دقـت کنیـد. در ایـن شبکه ۱۷ مسیریاب وجود دارد و مسیریابی به روش DV انجام می‌شود. در این مثال معیار هزینه

[1] به این راه حل Split Horzon گفته می شود.

[2] Hierarchical Routing



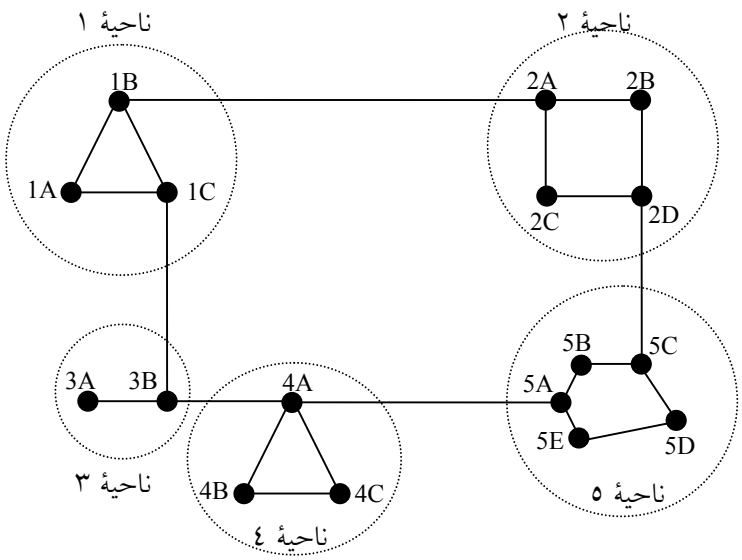

شکل (۱۳-٤) زیرساخت ارتباطی از یک شبکهٔ فرضی و جداول مسیریابی

| مقصد | خطّ | هزینه |
|---|---|---|
| **1A** | – | - |
| **1B** | 1B | 1 |
| **1C** | 1C | 1 |
| **2A** | 1B | 2 |
| **2B** | 1B | 3 |
| **2C** | 1B | 3 |
| **2D** | 1B | 4 |
| **3A** | 1C | 3 |
| **3B** | 1C | 2 |
| **4A** | 1C | 3 |
| **4B** | 1C | 4 |
| **4C** | 1C | 4 |
| **5A** | 1C | 4 |
| **5B** | 1C | 5 |
| **5C** | 1B | 5 |
| **5D** | 1C | 6 |
| **5E** | 1C | 5 |

شکل (۱٤-٤) جدول مسیریابی 1A

"تعداد گام" می‌باشد. ( وقتی معیار هزینه تعداد گام است ، براحتی هزینهٔ هر خط را ثابت و مساوی ۱ فرض کنید ) در شکل (۱۴-٤) جدول مسیریابی 1A نشان داده شده است و بگونه‌ای که ملاحظه می‌شود دارای ۱۷ رکورد است. تمام مسیریابها دارای چنین جدولی هستند و در فواصل منظم باید آن را برای مسیریابهای مجاور خود ارسال نمایند.

حال فرض کنید که تعداد صدهزار مسیریاب در شبکه موجود باشد. ( صدهزارمین شبکهٔ دنیا که به اینترنت پیوست در سال ۱۹۹٦ ثبت شده است! ) در چنین حالتی مسیریاب قادر به ذخیره ، پردازش و ارسال جداول مسیریابی برای دیگر مسیریابها نخواهد بود.




شکل (۱۵-٤) زیرساخت ارتباطی از یک شبکهٔ فرضی و جداول مسیریابی

| مقصد | خطّ | هزینه |
|---|---|---|
| **1A** | – | – |
| **1B** | 1B | 1 |
| **1C** | 1C | 1 |
| **Region** | 1B | 2 |
| **Region** | 1C | 2 |
| **Region** | 1C | 3 |
| **Region** | 1C | 4 |

شکل (۱٦-٤) جدول مسیریابی 1A

در مسیریابی سلسله‌مراتبی ، مسیریابها در گروههایی به نام **"ناحیه"**[1] دسته بندی می‌شوند. هر مسیریاب فقط **"نواحی"** و مسیریابهای درون ناحیهٔ خود را می‌شناسد و هیچ اطلاعی از مسیریابهای درون نواحی دیگر ندارد. در شکل (۱۵-٤) ، شبکهٔ مثال قبل در قالب پنج ناحیه تقسیم بندی شده و در شکل (۱٦-٤) جدول مسیریابِ 1A به تصویر کشیده شده است. در این جدول به ازای هر "ناحیه" و هر مسیریاب درون ناحیه ، یک رکورد در حافظه نگهداری می‌شود. به عنوان مثال مسیریاب 1A برای ارسال یک بسته به مسیریابی که در ناحیهٔ ۲ واقع است ، آنرا به سمت مسیریاب 1B هدایت می‌نماید؛ طبق این جدول برای ارسال بسته به نواحی ۳ و ٤ و ۵ ، باید از طریق 1C اقدام شود.

به گونه‌ای که از جدول (۱٦-٤) مشهود است با استفاده از مسیریابی سلسله‌مراتبی تعداد رکوردهای جدول مسیریابی از ۱۷ به ۷ رکورد کاهش یافته است. (۳ رکورد برای مسیریابهای درون ناحیه و ٤ رکورد برای بقیهٔ نواحی )

حال فرض کنید شبکه‌ای با ۷۲۰ مسیریاب ایجاد شده است؛ در روش معمولی ( DV ) جدول مسیریابی دارای ۷۲۰ رکورد خواهد بود. اگر این شبکه به ۲٤ ناحیه تقسیم و در هر ناحیه ۳۰ مسیریاب تعریف شود ، در این حالت مسیریاب نیاز به ۳۰ رکورد به ازای هر مسیریاب در درون ناحیه و ۲۳ رکورد به ازای ۲۳ ناحیهٔ دیگر دارد. (جمعاً ۵۳ رکورد)

برای ساده‌تر شدن جداول مسیریابی می‌توان از روش **"سلسله‌مراتبی سه‌سطحی"** استفاده کرد. در این روش کل شبکه به صورت زیر تقسیم‌بندی می‌شود:

- کل شبکه به چندین "دسته"[2] تقسیم می‌شود.
- هر "دسته" به چند "ناحیه" تقسیم می‌شود.
- هر ناحیه در برگیرندهٔ چند مسیریاب است.

به عنوان مثال شبکه‌ای با ۷۲۰ مسیریاب به ۸ "دسته" ، هر دسته دارای ۹ "ناحیه" و هر ناحیه دارای ۱۰ مسیریاب باشد. در این حالت هر مسیریاب به حداکثر ۲۵ رکورد احتیاج دارد:

۱. ۱۰ رکورد برای مسیریابهای هم‌ناحیه
۲. ۸ رکورد برای بقیهٔ نواحی (۸=۱-۹)
۳. ۷ رکورد برای بقیهٔ دسته‌ها (۷=۱-۸)

---

[1] Region
[2] Cluster

سلسله‌مراتب در مسیریابی می‌تواند بسته به بزرگی شبکه ، تا چندین سطح ادامه یابد. بعنوان مثال یک شبکه می‌تواند بصورت زیر تقسیم‌بندی شود:

- کل شبکه به تعدادی "ناحیه" تقسیم شود. Region
- هر ناحیه به تعدادی "دسته" تقسیم شود. Cluster
- هر دسته به تعدادی "حوزه" تقسیم شود. Zone
- هر حوزه به تعدادی "گروه" تقسیم شود. Group
- در هر گروه تعدادی مسیریاب تعریف شود.

برای آنکه بتوانید میزان صرفه‌جویی در اندازهٔ جداول مسیریابی سلسله‌مراتبی را ارزیابی کنید به جدول (۱۷-٤) دقت کنید. در این جدول تعداد رکوردی که هر مسیریاب ، باید در حافظه نگهداری کند ، ارائه شده است. (در یک شبکه با ۷۲۰ مسیریاب )

تنها اشکالی که می‌توان برای روش سلسله‌مراتبی برشمرد ، آنست که چون در این روش کل توپولوژی زیرشبکه برای هر مسیریاب مشخص نیست ، لذا ممکن است مسیر انتخابی برای ارسال بسته به یک مسیریابِ خاصّ درون یک ناحیه بهینه نباشد ولی در مجموع این روش به حالت بهینه نزدیک بوده و تغییرات توپولوژیکی و ترافیکی در جداول مسیریابی تاثیر داده خواهد شد.

در شبکهٔ اینترنت از روش سلسله‌مراتبی در مسیریابی استفاده می‌شود و پروتکلهای متفاوتی برای مسیریابی در درون یک "ناحیه" و مسیریابی بین نواحی تعریف و ارائه شده است.

| | تعداد ناحیه Regions | تعداد دسته Clusters | تعداد حوزه Zones | تعداد مسیریاب | تعداد رکورد در جدول |
|---|---|---|---|---|---|
| **مسیریابی DV بدون سلسله‌مراتب** | ۱ | – | – | ۷۲۰ | ۷۲۰ |
| **مسیریابی DV با سلسله‌مراتب دوسطحی** | ۲٤ | – | – | ۳۰ | ۵۳ |
| **مسیریابی DV با سلسله‌مراتب سه‌سطحی** | ۹ | ۸ | – | ۱۰ | ۲۵ |
| **مسیریابی DV با سلسله‌مراتب سه‌سطحی** | ۹ | ۵ | ٤ | ٤ | ۱۹ |

جدول (۱۷-٤) مقایسهٔ اندازهٔ جدول مسیریابی در روشهای سلسله‌مراتبی

## ۵) مسیریابی در اینترنت

اینترنت مجموعه‌ای از "شبکه‌های خودمختار[۱] و مستقل" است که به نحوی به هم متصل شده‌اند. "شبکهٔ خودمختار" که اختصاراً AS[۲] نامیده می‌شود ، شبکه‌ای است که تحت نظارت و سرپرستی یک مجموعه یا سازمان خاص پیاده و اداره می‌شود. مثلاً یک دانشگاه می‌تواند برای خود یک شبکهٔ AS مهیّا نماید بگونه‌ای که هر دانشکده دارای یک شبکهٔ محلی باشد و این شبکه‌های محلی طبق صلاحدید مسئول شبکه (از طریق نصب مسیریاب) به هم متصل شوند. اضافه شدن یک شبکهٔ محلی یا حتی یک ماشین میزبان به شبکه باید با مجوز و نظارت مسئول شبکه انجام شود. شکل (۱۸-٤) می‌تواند مثالی از یک شبکهٔ AS باشد. این مسئله واضح است که برای برقراری ارتباط بین زیرشبکه‌های درون یک AS ، باید مسیریابی انجام شود؛ مسئول شبکه موظف است "الگوهای زیرشبکه"[۳] را طراحی کرده و مسیریابها را بگونه‌ای تنظیم نماید که مسیریابی در درون شبکهٔ تحت نظارتِ او به درستی انجام شود.

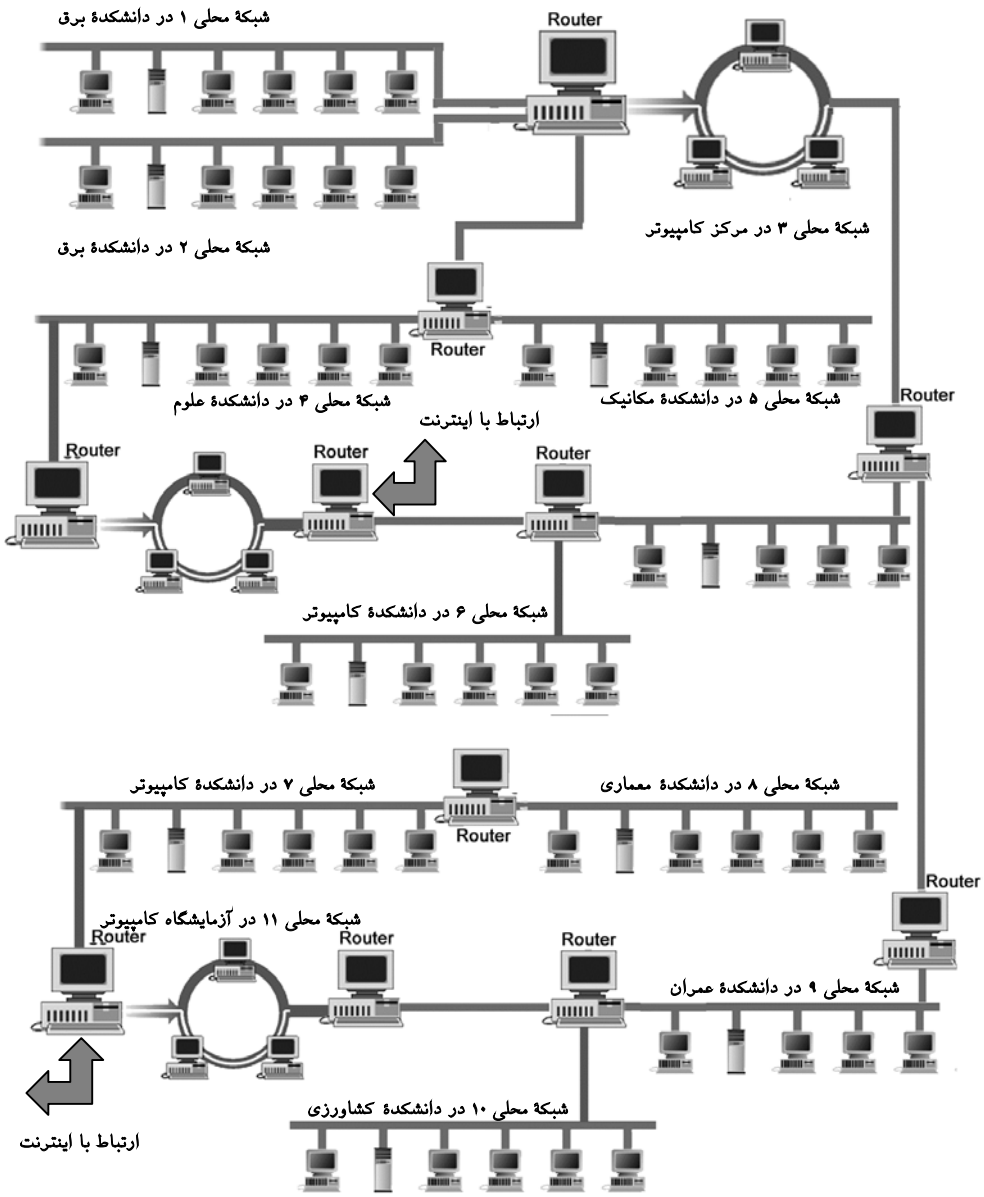
مسئول شبکهٔ خودمختار می‌تواند بر روی شبکهٔ تحت نظارت خود "حاکمیّت"[٤] داشته باشد یعنی می‌تواند بر روی تک‌تک اجزای شبکه (ماشینهای میزبان)، توپولوژی کل شبکه ، سیستم عامل ، طراحی زیرساخت ارتباطی و طریقهٔ اتصال شبکه‌های محلی و نوع پروتکل مسیریابی اعمال نفوذ کرده و نظرات خود را پیاده نماید.

مسیریابی بسته‌های IP در درون یک شبکهٔ خودمختار بیشتر تابع پارامترهایی نظیر سرعت و قابل اعتماد بودن الگوریتم مسیریابی است. حال فرض کنید شبکه‌های خودمختار از طریق یک زیرساخت ارتباطی بسیار سریع و جهانی به هم متصل شوند.[۵] مسیریابی بسته‌های اطلاعاتی بر روی شاهراه‌هایی که شبکه‌های AS را بهم متصل کرده ، مسائلی کاملاً متفاوت با مسیریابی در درون یک شبکهٔ خودمختار دارد. در مسیریابی بین شبکه‌های AS ، مسائلی نظیر امنیت ، پرداخت حقّ اشتراک و سیاست نیز می‌تواند در انتخاب بهترین مسیر دخیل باشد.

به گونه‌ای که در مبحث آدرس‌دهی اشاره شد یک شبکهٔ عظیم می‌تواند خودش از چندین زیرشبکه تشکیل شده باشد که مسیریابی بسته‌ها در درون این شبکه بر اساس الگوهای زیرشبکه انجام می‌شود ولی از دیدگاه شبکهٔ اینترنت ، کل این شبکه (با تمام ماشینهای میزبان و مسیریابهای داخلی) فقط یک "گره" تلقی شده و مسیریابهای بیرونی آنرا یک سیستم واحد می‌بینند.[٦] در حقیقت کلاسهای آدرس IP به شبکه‌های خودمختار اختصاص داده می‌شود.

---

[۱] Autonomous
[۲] Autonomous System
[۳] **به فصل قبلی مراجعه کنید.**
[٤] Authority
[۵] **در فصل قبل اشاره شد که به زیرساخت ارتباطی شبکه‌های جهانی ستون فقرات یا** Backbone **می‌گویند.**
[٦] **این شبکه با قسمت** NetID **در آدرس** IP **مشخص می‌شود.**




شکل (۱۸-٤) مثالی از یک سیستم خودمختار (AS) در یک دانشگاه

قبل از معرفی روشهای مسیریابی در اینترنت ، ارائهٔ یک مثال ساده می‌تواند به فهم کلیّت موضوع کمک کند:

کشورمان ایران را در نظر بگیرید؛ در این کشور مسیرهای هوایی زیادی بین مراکز استان و برخی از شهرستانها وجود دارد. بنابراین یک فرد ایرانی وقتی می‌خواهد از نقطه‌ای به نقطهٔ دیگر سفر کند با یک ارزیابی ساده ( از میزان هزینه و زمان ) مسیر خود را انتخاب و بر اساس آن سفرش را آغاز می‌نماید. یک ایرانی مشکل خاصی برای سفر آزادانه در درون کشور خود ندارد و براحتی می‌تواند بین هر دو نقطه سفر کند؛ ضوابط حاکم بر راههای هوایی و زمینی کشور نیز توسط دولت تبیین می‌شود. حال فرض کنید یک ایرانی در شیراز بخواهد به شهر کلن در آلمان سفر کند؛ برای انتخاب مسیر خود : اولاً نیاز به اخذ مجوزهای لازم از کشور مبدأ و مقصد دارد. ثانیاً چون پرواز مستقیم بین این دو شهر وجود ندارد ، باید با یک پرواز داخلی خود را به یک فرودگاه بین‌المللی (مثلاً مهرآباد تهران) رسانده و از طریق یک پرواز خارجی به یکی از شهرهای آلمان ( برلین ، فرانکفورت ) سفر کند و پس از ورود به آلمان طبق ضوابط دولت آلمان و از یک مسیر مناسب به کلن آلمان سفر کند. تفاوتهای ویژه در پروازهای داخلی و پروازهای خارجی از لحاظ سطح کنترل مدارک ، اخذ هزینه و مسائل امنیتی وجود دارد. یک پرواز خارجی ممکن است در طول مسیر از کشورهای ثالثی عبور کند و این کشورها بر اساس تعرفه‌هایی برای عبور یک پرواز خارجی اخذ هزینه کنند یا آنکه بدلایل امنیتی اصلاً اجازهٔ عبور صادر نکنند.

این مثال تا حدود زیادی به انواع مسیریابی در شبکهٔ اینترنت شباهت دارد. بگونه‌ای که در شکل (۱۸-٤) مشهود است مسیرهای داخلی زیادی در درون شبکهٔ دانشگاه وجود دارد که بسته‌های IP می‌توانند از طریق آنها بین هر دو ماشین درون این شبکه مبادله شوند ولی برای ارسال بسته‌ها به خارج از شبکه ، ابتدا بسته‌ها باید به سمت یکی از دو مسیریاب که با دنیای خارج در ارتباطند هدایت شوند ( مسیریابی درونی )؛ سپس این دو مسیریاب بسته‌ها را روی زیرساخت خارجی شبکه به جریان می‌اندازند. مسیریابهایی که در درون شبکهٔ AS نصب شده‌اند و نقش برقراری ارتباطات داخلی را برعهده دارند به نام "دروازه‌های درونی"[۱] مشهورند.[۲] مسیریابهایی که ارتباط دو شبکهٔ خودمختار متفاوت را برقرار می‌کنند و تمامی ارتباطات بین‌شبکه‌ای از طریق آنها انجام می‌شود به نام "دروازه‌های مرزی"[۳] شناخته می‌شوند. به شکل (۱۹-٤) دقت کنید. در این شکل زیرساخت ارتباطی چهار شبکهٔ AS که هر کدام با اهداف ویژه ، تحت نظارت و سرپرستی یک سازمان خاص پیاده‌سازی شده‌اند ، به تصویر

[۱] Interior Gateway

[۲] **"دروازه یا Gateway" نام سنتی و اولیّه مسیریاب بوده است.**

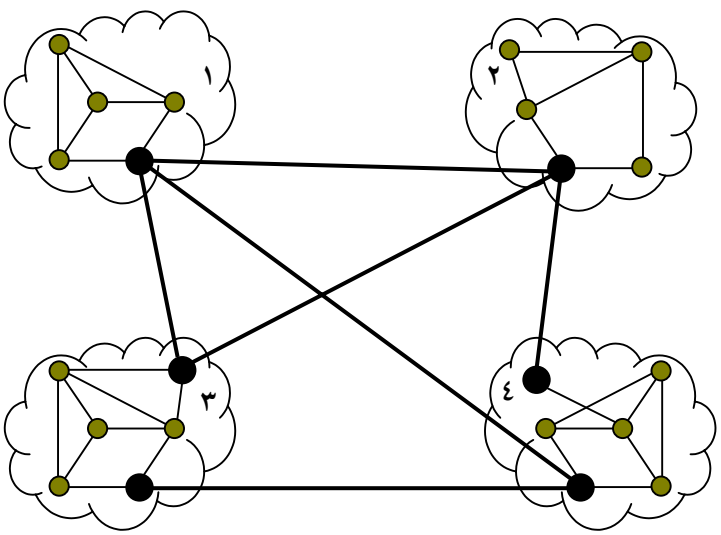
[۳] Border Gateway

کشیده شده است. از بین مسیریابهای این چهار شبکه فقط تعداد محدودی از مسیریابها ، ارتباطات بین شبکه‌ای را برعهده دارند. این "مسیریابهای مرزی" با رنگ تیره‌تری نسبت به مسیریابهای درونی نشان داده شده‌اند. مسیریابهای مرزی و ساختار ارتباطی بین آنها تابع قواعد "مسیریابی برونی" و مسیریابهای داخلی تابع الگوریتمهای "مسیریابی درونی" است که می‌تواند کاملاً با هم متفاوت باشد. به مسیریابهای مرزی ، "مسیریابهای BGP" نیز گفته می‌شود.

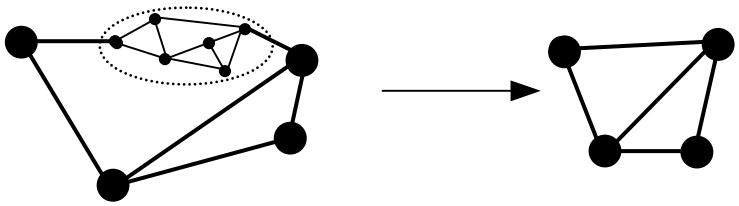
در ساختار مثال فوق اگر یک ماشین میزبان در شبکهٔ ۱ بخواهد بسته‌ای برای ماشین دیگر در شبکهٔ ٤ بفرستد سه مرحله مسیریابی لازم است:

- مسیریابی در درون شبکهٔ ۱ تا رسیدن بسته به مسیریاب مرزی
- مسیریابی روی خطوط ارتباطی بین‌شبکه‌ای تا رسیدن به شبکهٔ ٤
- مسیریابی درون شبکهٔ ٤ تا رسیدن به ماشین مقصد

ممکن است بین دو مسیریاب مرزی یک شبکه قرار گرفته باشد که کلّ آن را می‌توان یک خطّ واحد در نظر گرفت. در شکل (۲۰-٤) این موضوع به تصویر کشیده شده است.




شکل (۱۹-٤) مثالی از چهار شبکهٔ AS متصل به هم

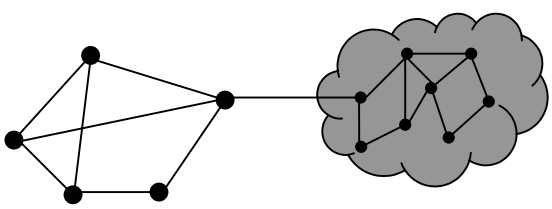


شکل (۲۰-٤) ساختار ارتباطی یک شبکهٔ فرضی

مجموعهٔ اینترنت از دهها هزار شبکهٔ خودمختار تشکیل شده که از طریق یک ساختار ارتباطی به نام "ستون فقرات" به هم متصل شده‌اند. در ادامه طریقهٔ اتصال این شبکه‌ها به اینترنت تشریح شده است.

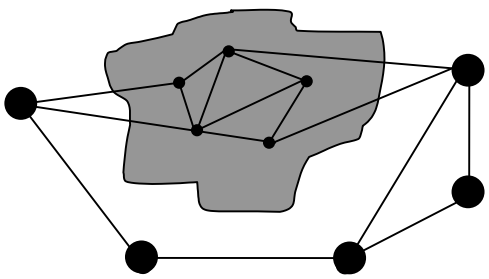
سه دسته شبکه می‌توانند با مسیریابهای BGP در ارتباط باشند:

- **"شبکه‌های پایانی"**- Stub - : این نوع از شبکه ها فقط با یک مسیریاب نوع BGP در ارتباطند و بنابراین نمی‌توانند در ستون فقرات اینترنت نقش ایفا کنند و کمکی به توزیع ترافیک بر روی شبکهٔ اینترنت نمی‌کنند. معمولاً برای وصل شبکه‌های پایانی به یکی از مسیریابهای BGP باید هزینهٔ قابل توجهی در هر ماه پرداخت شود. اکثر شبکه‌های متصل به اینترنت در ایران بخاطر عدم وجود ستون‌فقرات ارتباطی سریع بین شهرها و استانهای مختلف کشور ، از نوع شبکه‌های پایانی -Stub- بشمار می‌روند.



شکل (۲۱-٤) ساختار ارتباطی یک شبکهٔ پایانی

➢ **"شبکه‌های چندارتباطی"**[۱]: این گونه از شبکه‌ها بین مسیریابهای نوع BGP واقعند و می‌توانند برای توزیع و حمل ترافیک در شبکهٔ اینترنت مورد استفاده قرار بگیرند مگر آنکه بدلایل امنیتی ، تمایل به چنین کاری نداشته باشند.



شکل (۲۲-٤) ساختار ارتباطی یک شبکهٔ چندارتباطی

➢ **"شبکه‌های ترانزیت"**[۲] : این گونه شبکه ها که به نحوی به روی ستون فقرات شبکه اینترنت واقعند وظیفه عمده‌ای در حمل و توزیع بسته های IP بعهده دارند. (همانند شبکهٔ NSFNet در آمریکا )

با مقدمهٔ فوق باید برخی از الگوریتمهای کاربردی مسیریابی درونی و برونی در شبکهٔ اینترنت را معرفی کنیم.

## ۶) پروتکل RIP در مسیریابی درونی

پروتکل RIP[۳] یکی از پروتکلهای پرسابقه در مسیریابی درونی در شبکهٔ اینترنت بشمار می‌رود و ابداع کنندهٔ اصلی آن شرکت "زیراکس"[٤] است. شاید نقطهٔ عطف گسترش این پروتکل سال ۱۹۸۲ بود که بهمراه نسخهٔ "یونیکس برکلی" (BSD)[٥] ، نرم‌افزار پیاده شدهٔ

[۱] Mulitconnected Networks
[۲] Transit Networks
[۳] Routing Information Protocol
[٤] Xerox Network System
[٥] Berkeley Software Distribution

پروتکل RIP به بازار ارائه شد. بعد از آن ، در نسخه‌های بعدی یونیکس ، پروتکل RIP نیز بهینه‌سازی شد.[۱]

پروتکل RIP ذاتاً مبتنی بر الگوریتم بردار فاصله (DV) است؛ با این مشخصه که ، معیار هزینه در این پروتکل "تعداد گام" است یعنی هزینهٔ هر خط بین دو مسیریاب ثابت و مساوی ۱ است. با این معیار وقتی هزینهٔ مسیری ٤ اعلام می‌شود بدین معناست که ٤ مسیریاب در سر راه این مسیر وجود دارد. ( برای درک دقیق این پروتکل ، باید الگوریتم DV را به دقّت بررسی کنید.)

در پروتکل RIP جداول مسیریابی هر ۳۰ ثانیه یکبار بین مسیریابهای مجاور مبادله می‌شوند و اگر مسیریابی این جداول را از یک همسایه به مدت ۱۸۰ ثانیه (۳ دقیقه) دریافت نکند ، متوجه یک خرابی در آن همسایه شده و ضمن درج ∞ در جدول خود ، آنرا به تمام مسیریابهای مجاور خود اعلان می‌کند. ارسال جداول مسیریابی به همسایه‌ها در فواصل ۳۰ ثانیه‌ای ، **"اعلان"**[۲] گفته می‌شود.

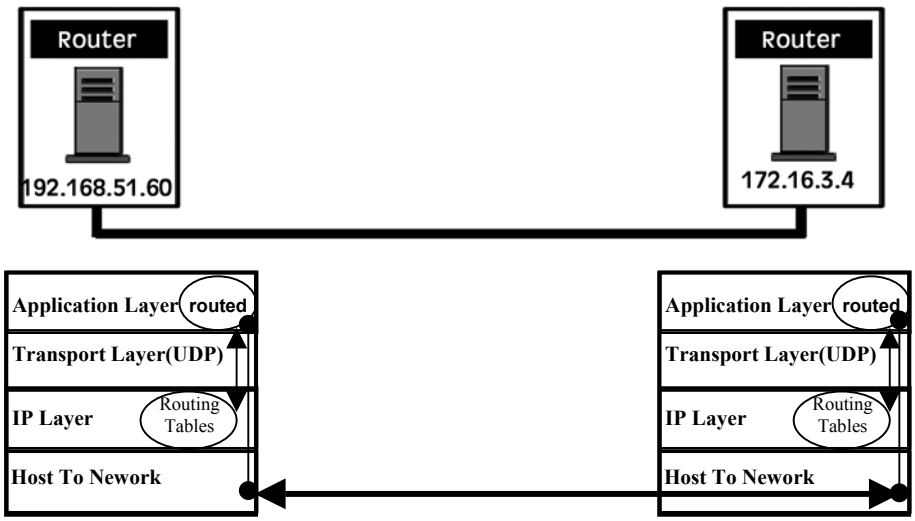
در پروتکل اولیهٔ RIP ، حداکثر تعداد طول مسیر به ۱۵ محدود شده بود یعنی در یک زیرساخت ارتباطی با هر تعداد مسیریاب ، طول بزرگترین مسیر نباید از ۱۵ زیادتر شود. در این پروتکل هر مسیریاب می‌تواند با ارسال یک "پیام تقاضا"[۳] از همسایهٔ خود در مورد هزینهٔ رسیدن به یک مسیریاب خاص در شبکه سوال نماید.

RIP از پورت شمارهٔ ۲۵۰ و پروتکل UDP برای مبادلهٔ جداول مسیریابی استفاده می‌نماید. ( موضوع شمارهٔ پورت و طریقهٔ برنامه‌نویسی تحت شبکهٔ اینترنت در فصول بعدی بررسی می‌شوند.) در شکل (۲۳-٤) ساختار پروتکل RIP که در لایهٔ کاربرد از مدل TCP/IP تعریف می‌شود ، نشان داده شده است. بگونه‌ای که مشاهده می‌شود جداول مسیریابی در لایهٔ دوم و برای مسیریابی بسته‌های IP ، مورد استفاده هستند ولی مبادلهٔ جداول و عملیاتِ به‌هنگام‌سازی توسط یک برنامهٔ کاربردی در لایهٔ چهارم انجام می‌شود. نام این برنامه در سیستم عامل یونیکس routed[٤] می‌باشد و با استفاده از برنامه‌نویسی سوکت که در فصلی مجزا به آن خواهیم پرداخت نوشته شده است.

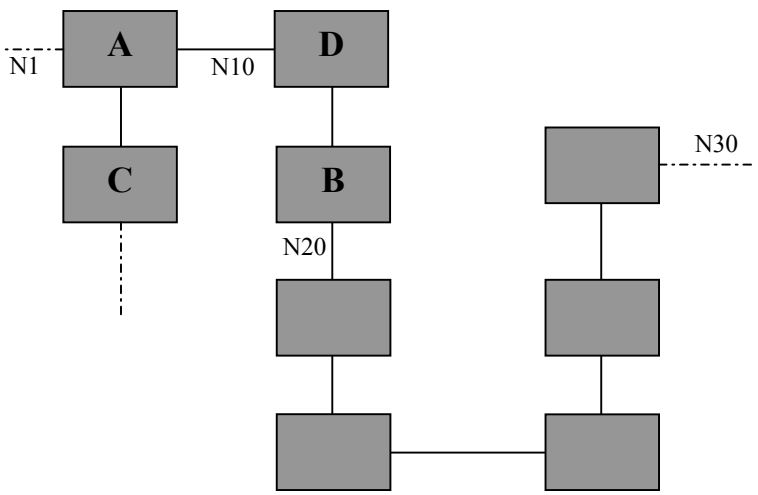
هرچند پروتکل RIP دقیقاً مشابه الگوریتمی است که در روش DV به آن پرداختیم ولی در اینجا به مثالی دیگر خواهیم پرداخت. در شکل (۲٤-٤) بخشی از زیرساخت ارتباطی یک شبکهٔ AS نشان داده شده و طبق معمول ماشینهای میزبان حذف شده‌اند. خطوط نقطه‌چین به

---

[۱] RIP Version 2
[۲] Advertisement
[۳] Request Message
[٤] **کلمهٔ routed را route dee تلفظ کنید چراکه مخفف** Routing Daemon است.




(۲۳-٤) پروتکل RIP در لایهٔ کاربرد




شکل (۲٤-٤) بخشی از زیرساخت ارتباطی یک شبکهٔ AS

معنای آنست که گراف شبکه ادامه دارد ولی در تصویر نشان داده نشده است.
در جدول (۲۵-٤) بخشی از جدول مسیریابی D نشان داده شده است.

| مسیریاب مقصد | مسیریاب بعدی | تعداد گام تا مقصد |
|---|---|---|
| **N1** | A | 2 |
| **N20** | B | 2 |
| **N30** | B | 7 |
| **N10** | - | 1 |
| **....** | .... | .... |

جدول (۲۵-٤) بخشی از جدول مسیریاب D

فرض کنید در لحظهٔ *t=t0* جدول مسیریابی (۲٦-٤) ، توسط مسیریاب A به D اعلان شده باشد.

| مسیریاب مقصد | مسیریاب بعدی | تعداد گام تا مقصد |
|---|---|---|
| **N1** | - | 1 |
| **N10** | - | 1 |
| **N30** | C | 4 |
| **....** | .... | .... |

جدول (۲٦-٤) بخشی از جدول اعلان شده توسط مسیریاب A به D

با استفاده از این جدول و طبق الگوریتمی که در روش DV اشاره شد ، D نتیجه می‌گیرد که برای رسیدن به شبکهٔ N30 مسیر کوتاهتری از طریق A وجود دارد و بنابراین جدول خود را به صورت جدول (۲۷-٤) اصلاح می‌کند.

| مسیریاب مقصد | مسیریاب بعدی | تعداد گام تا مقصد |
|---|---|---|
| **N1** | A | 2 |
| **N20** | B | 2 |
| **N30** | A | 5 |
| **N10** | - | 1 |
| **....** | .... | .... |

جدول (۲۷-٤) بخشی از جدول مسیریاب D پس از اصلاح

قالب پیامها در پروتکل RIP در شکل (۲۸-٤) نشان داده شده است. در "سرآیند پیام"[۱] سه فیلد زیر تعریف شده است:

- **Command** : این فیلد نوع پیام را مشخص می‌کند :

**۱** : پیام از نوع تقاضا است.[۲] اگر در بدنهٔ پیام آدرس IP درج شده باشد بدین معناست که هزینهٔ یک مسیریاب خاص تقاضا شده است ولی اگر هیچ آدرسی درج نشده باشد یعنی کل جدول مسیریابی باید برای تقاضا دهنده ارسال شود.

**۲** : پیام از نوع پاسخ است.[۳] یعنی درون بدنهٔ پیام کلّ یا قسمتی از جدول مسیریابی قرار گرفته است.

- **Version** : این فیلد نسخهٔ پروتکل RIP که بسته بر طبق آن تولید و ارسال شده است را معین می‌کند. ( ۱ ، ۲ و ...)
- **Reserved** : این فیلد بدون استفاده است و با صفر پر می‌شود.

بدنهٔ پیام در این پروتکل شامل ۳ فیلد مهم است :

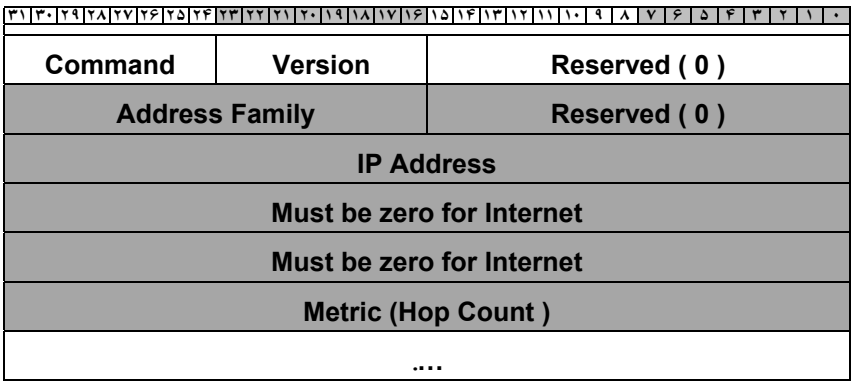
- **Address Family** : در این فیلد برای شبکهٔ اینترنت مقدار ثابت **۲** قرار می‌گیرد.
- **IP Address** : آدرس IP مسیریاب را تعیین می‌کند.
- **Metric** : معیار هزینهٔ رسیدن به مسیریاب مشخص شده در فیلد قبلی را بر حسب گام تعیین می‌کند. در پروتکل RIP حداکثر مقدار این فیلد ۱۵ است. " اگر یک مسیریاب در دسترس نباشد ، به جای مقدار ∞ در این فیلد مقدار **۱٦** قرار می‌گیرد.

بدنهٔ پیام (قسمت تیره‌رنگ در قالب پیام )می‌تواند به ازای هر رکورد در جدول مسیریابی تکرار شود.

در مثالهای این فصل ، تمام شبکه‌ها و مسیریابها با اسامی نمادین (همانند A ، B ،N1 و N30) نشان داده شده‌اند در حالی واضح است که شبکه‌ها و مسیریابها با آدرس IP تعیین می‌شوند. در جدول (۲۹-٤) حالت واقعی از یک جدول مسیریابی که در حافظهٔ مسیریاب نگهداری می‌شود ، آورده شده است. این جدول در سیستم عامل یونیکس با استفاده از دستور netstat –rn بدست آمده است.

---

[۱] Message Header
[۲] Request Message
[۳] Response Message




شکل (۲۸-٤) قالب پیامها در پروتکل RIP

| Destination | Gateway | Flags | Ref | Use | Interface |
|---|---|---|---|---|---|
| 127.0.0.1 | 127.0.0.1 | UH | 0 | 26492 | lo0 |
| 192.168.2. | 192.168.2.5 | U | 2 | 13 | fa0 |
| 193.55.114. | 193.55.114.6 | U | 3 | 58503 | le0 |
| 192.168.3. | 192.168.3.5 | U | 2 | 25 | qaa0 |
| 224.0.0.0 | 193.55.114.6 | U | 3 | 0 | le0 |
| default | 193.55.114.129 | UG | 0 | 143454 | |

جدول (۲۹-٤) حالت واقعی از یک جدول مسیریابی

آدرس شبکه‌های مقصد که در این جدول درج شده‌اند به شرح زیرند :

♦ **192.168.2. و 193.55.114. و 192.168.3. :** این سه آدرس ، سه شبکه با آدرس کلاس C را تعیین می‌کند. در هنگام ارسال یک بسته برای یک ماشین ، آدرس IP ماشین مقصد ، بررسی شده و اگر قسمت "مشخصهٔ شبکهٔ"[1] آن یکی از سه مقدار بالا باشد ، بسته به سمت مسیریابی هدایت می‌شود که آدرس آن در قسمت Gateway درج شده است. آدرس نمادین کانال خروجی که بسته باید روی آن ارسال شود ( یعنی آدرس نمادین کارت شبکه ) در قسمت Interface مشخص شده است.

[1] NetID

♦ **127.0.0.1 :** این آدرس همان آدرس "بازگشت" است که در فصل قبل توضیح داده شد. ارسال بسته‌ای به این مقصد باعث بازگشت آن به تولیدکنندهٔ آن خواهد شد.

♦ **224.0.0.0 :** این آدرس یک آدرس کلاس D است و برای پخش همگانی کاربرد دارد.

♦ **default :** هرگاه بسته‌ای برای یک ماشین میزبان ارسال شود ولی آدرس شبکهٔ مقصد آن جزو شبکه‌های مشخص شده در جدول نباشد آن بسته به سمت مسیریابی هدایت می‌شود که آدرس آن در قسمت Gateway درج شده است.

به گونه‌ای که اشاره شد با وجود تمام مشکلاتی که در این پروتکل وجود دارد ، به دلیل سادگی و سرعت ، هنوز هم در شبکه‌های کوچک مورد استفاده قرار می‌گیرد و در نسخه های جدید ویندوز NT پشتیبانی می‌شود.

## ۷) پروتکل OSPF در مسیریابی درونی

بکارگیری پروتکل RIP در شبکه‌های کامپیوتری بیشتر به دلیل شرایط زمان بوده است. در دههٔ هفتاد و هشتاد حافظه و پردازنده‌های سریع ، گران قیمت بودند و پیاده‌سازی الگوریتمهای مسیریابی مبتنی بر روشهایی نظیر LS که هم به حافظه و هم به پردازندهٔ سریع نیاز دارند ، مقرون به صرفه نبود. از طرفی شبکه‌ها نیز آنقدر توسعه نیافته بودند که نیاز به الگوریتمهای بهینه‌تر احساس شود. با گسترش اینترنت و توسعهٔ شبکه‌های خودمختار در اواخر دههٔ هشتاد ، کاستیهای پروتکل RIP نمود بیشتری پیدا کرد و با سریع شدن پردازنده‌ها و ارزان شدن سخت‌افزار ، نیاز به طراحی یک پروتکل بهینه ، IETF[۱] را واداشت تا در سال ۱۹۹۰ ، OSPF[۲] را به عنوان یک پروتکل استاندارد ارائه نماید. مسیریابهای زیادی مبتنی بر این پروتکل به بازار عرضه شده‌اند و احتمال می‌رود که در آینده تبدیل به مهمترین پروتکل مسیریابی درونی در شبکه‌های AS شود.

ابتدا مشخصه‌های پروتکل OSPF را در مقایسه با پروتکل RIP ارائه می‌کنیم:

---

[۱] Internet Engineering Task Force

**استانداردسازی تمام پروتکلها و مستندات شبکهٔ اینترنت توسط IETF که کمیته‌ای بین‌المللی است انجام می‌شود.**

[۲] Open Shortest Path First

**کلمهٔ Open در نام این پروتکل به معنای آنست که این پروتکل بطور عام در اختیار تمام دنیا قرار دارد و طراحان مسیریاب می‌توانند بدون هیچ مشکلی از آن استفاده کنند؛ یعنی امتیاز این پروتکل در اختیار گروه یا شرکت خاصی نیست. ( بر خلاف پروتکلی مثل IGRP که امتیاز آن متعلق به شرکت CISCO در آمریکاست و کسی حق استفادهٔ بدون مجوز از آنرا ندارد. ( RFC1247 )**

- بر خلاف پروتکل RIP ، در این پروتکل از الگوریتم LS برای محاسبهٔ بهترین مسیر استفاده می‌شود و بنابراین مشکل "شمارش تا بینهایت" وجود ندارد.
- برخلاف پروتکل RIP ، در این پروتکل معیار هزینه فقط "تعداد گام" نیست بلکه می‌تواند چندین معیار هزینه را در انتخاب بهترین مسیر در نظر بگیرد.
- بر خلاف پروتکل RIP ، در این پروتکل حجم بار و ترافیک یک مسیریاب در محاسبهٔ بهترین مسیر دخالت داده می‌شود و در ضمن در هنگام خرابی یک مسیریاب ، جداول مسیریابی سریعاً همگرا می‌شود.
- بر خلاف پروتکل RIP ، در این پروتکل ، فیلد Type of Service در بستهٔ IP می‌تواند در نظر گرفته شود و بر اساس نوع سرویس درخواستی ، برای یک بسته مسیر مناسب انتخاب گردد.
- بر خلاف پروتکل RIP ، در پروتکل OSPF تمام بسته‌های ارسالی برای یک مقصد خاص ، روی بهترین مسیر هدایت نمی‌شود بلکه درصدی از بسته‌ها روی مسیرهایی که از لحاظ حداقل هزینه در رتبهٔ ۲ ، ۳ و ... قرار دارند ارسال می‌شود تا پدیدهٔ "نوسان" که قبلاً به آن اشاره شد رخ ندهد. به این کار **"موازنهٔ بار"**[1] گفته می‌شود.
- بر خلاف پروتکل RIP ، در این پروتکل از مسیریابی سلسله‌مراتبی پشتیبانی می‌شود.
- بر خلاف پروتکل RIP ، در این پروتکل مسیریابها جداول مسیریابی را از دیگر مسیریابها قبول نمی‌کنند مگر آنکه هویت ارسال کنندهٔ آن احراز شود. به همین دلیل مسئول شبکه برای هر مسیریاب یک "کلمهٔ عبور" تعیین می‌کند تا کاربران اخلالگر نتوانند با برنامه‌نویسی ، جداول مسیریابی مصنوعی تولید کرده و با ارسال آنها ، مسیریابی در شبکه را با مشکل مواجه کنند.

قبل از مطالعهٔ این بخش روش سلسله‌مراتبی در مسیریابی و الگوریتم LS را به دقّت بررسی کنید ، زیرا OSPF مبتنی بر این دو مفهوم است.

در ادبیات پروتکل OSPF ، سلسله‌مراتب تعیین شده برای نواحی بصورت زیر می‌باشد:

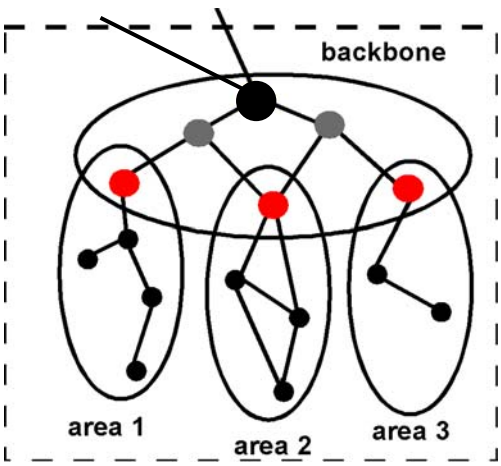
- یک شبکهٔ خودمختار (AS) به تعدادی "ناحیه"[2] تقسیم می‌شود. تمام مسیریابهای درون یک ناحیه باید مسیریابهای هم‌ناحیهٔ خود و هزینهٔ ارتباط بین آنها را بدانند و در جدولی ذخیره کنند. در لحظات به‌هنگام‌سازی ، این جداول برای تمام مسیریابهای هم‌ناحیه ارسال خواهد

[1] Load Balancing
[2] Area

شد. مسیریاب هیچ اطلاعی از وضعیت مسیریابهای درون نواحی دیگر ندارد. در شکل (۳۰-٤) این مسیریابها با علامت ● نشان داده شده است.

➢ درون هر ناحیه یک یا چند مسیریاب وجود دارند که ارتباط بین نواحی را برقرار می‌کنند؛ به آنها ، "مسیریابهای مرزی"[۱] گفته می‌شود. مجموعهٔ مسیریابهای مرزی و مسیریابهایی که در خارج از هر ناحیه نقش توزیع ترافیک بین نواحی را بر عهده دارند ( بهمراه ساختار ارتباطی بین این مسیریابها ) **"ستون‌فقرات"** شبکهٔ AS را تشکیل می‌دهد. در شکل (۳۰-٤) ، مسیریابهای مرزی با علامت ● نشان داده شده است.

➢ درون ستون‌فقراتِ شبکهٔ AS ممکن است مسیریابهایی وجود داشته باشند که با دیگر شبکه‌های AS در ارتباط باشد. به این مسیریابها "دروازه‌های مرزی"[۲] یا BGP گفته می‌شود. در شکل (۳۰-٤) این مسیریاب با دایرهٔ بزرگ و تیره‌رنگ نشان داده شده است.




شکل (۳۰-٤) سلسله‌مراتب مسیریابی در پروتکل OSPF

در پروتکل OSPF جداول زیر توسط مسیریابها "اعلان" می‌شود:

➢ **جدول مسیریابی محلی درون یک ناحیه**[۳] : این جداول ، محتوی اطلاعاتی در مورد گراف هزینهٔ ناحیه‌ای است که یک مسیریاب به آن متعلق است و توسط هر مسیریاب درون آن ناحیه، به تمام مسیریابها اعلان می‌شود.

---

۱ Area Border Routers
۲ Boundary Gateway
۳ Router Links Advertisement

- **جدول مسیریابی شبکه درون یک ناحیه**[۱] : این جداول که محتوی اطلاعاتی در مورد مسیریابها و کانالهای بین آنها در یک شبکه است ، توسط مسیریابهای درون یک ناحیه به تمامی مسیریابها اعلان می‌شود.
- **جدول خلاصهٔ مسیریابیِ مسیریابهای مرزی**[۲] : این جداول محتوی اطلاعاتی خلاصه ، در مورد مسیرهای موجود در خارج از نواحی است و توسط مسیریابهای مرزی به تمامی مسیریابهای نواحی مختلف اعلان می‌شود.
- **جدول مسیریابی شبکه**[۳] : این جداول محتوی اطلاعاتی در مورد مسیریابها و کانالهای بین آنها در خارج از شبکهٔ AS است و توسط مسیریابهای واقع بر ستون فقرات شبکهٔ AS به تمامی مسیریابهای نواحی مختلف اعلان می‌شود ولی فقط در مسیریابهای مرزی مورد استفاده قرار می‌گیرد.

بر خلاف RIP که از پروتکل UDP در لایهٔ انتقال استفاده می‌کند و به عنوان یک برنامهٔ کاربردی مطرح است ، پروتکل OSPF مستقیماً از پروتکل IP استفاده می‌کند و به عنوان یک برنامه در لایهٔ انتقال انجام وظیفه می‌نماید. اگر در فیلد "پروتکل" از بستهٔ IP عدد ۸۹ قرار گرفته باشد ، آن بسته تحویل پروسهٔ OSPF در لایهٔ سوم می‌شود. تعریف پروتکل OSPF در لایهٔ سوم ( بجای لایهٔ چهارم ) ، سرعت عمل آنرا افزایش داده است.

پیامهای تعریف شده در پروتکل OSPF متنوع و زیاد هستند ولی برای ارائهٔ یک دید کلی از امکاناتی که این پروتکل ارائه می‌دهد ، انواع و قالب این پیامها و فیلدهای تعریف شده در آن را بطور اجمالی معرفی می‌نماییم. بررسی کلّی برخی از فیلدهای این پیامها می‌تواند مشخصات کلی این پروتکل را روشنتر کند.

پنج نوع پیام در پروتکل OSPF تعریف شده است:

- پیام **"سلام"** : وقتی یک مسیریاب روشن و بوت می‌شود موظف است به تمام مسیریابهایی که مستقیماً به آنها متصل است[۴] یک پیام "سلام" بفرستد تا آنها از حضور این مسیریاب در شبکه مطلع شوند. قالب این پیام در شکل (۳۱-٤) نشان داده شده است و به گونه ای که دیده می‌شود سرآیند این پیام دو قسمت است:

---

۱ Router Links Advertisement
۲ Summary Links Advertisement
۳ Autonomous System Extended Links Advertisement
٤ Point To Point Adjacent Router

قسمت ٢٤ بایتی اول که به رنگ تیره نشان داده شده است. این قسمت از سرآیند ، در تمام پیامهای دیگر نیز وجود دارد و شامل اطلاعات مهمی است که به آن اشاره خواهیم کرد.
قسمت دوم سرآیند ، فقط مختص به "پیام سلام" می‌باشد.

- پیام Link State Update : هر مسیریاب موظف است که در بازه‌های زمانی مشخص ، جدول مسیریابی خودش را به روش سیل‌آسا به اطلاع دیگر مسیریابهای هم‌ناحیه برساند. این کار را با ارسال این پیام انجام می‌دهد؛ در ضمن هر مسیریاب وقتی هزینهٔ یکی از خطوط مستقیم او تغییر کرد یا مسیریاب مجاورش از شبکه بیرون رفت ( یا به شبکه برگشت ) سریعاً با این پیام آنرا به اطلاع دیگران می‌رساند. قالب این پیام در شکل (٣٢-٤) نشان داده شده است.

- پیام Database Description : هر مسیریاب به ازای تک‌تک رکوردهای هزینه که در یک بانک اطلاعاتی درون حافظهٔ اصلی ذخیره کرده ، یک فیلد شمارهٔ ترتیب در نظر می‌گیرد. مسیریاب ارسال کننده این پیام ، شمارهٔ ترتیب و تمام رکوردهایی را که در بانک اطلاعاتی خود ذخیره کرده است ، ارسال می‌نماید. گیرندهٔ این پیام با مقایسهٔ شماره‌های ترتیب رکوردها با رکوردهایی که در بانک اطلاعاتی خود دارد می‌تواند رکوردهای قدیمیتر را با رکوردهای جدید جایگزین نماید. قالب این پیام در شکل (٣٣-٤) نشان داده شده است .

- پیام Link State Request : با این پیام هر مسیریاب می‌تواند اطلاعات جدول مسیریابی را از از یک مسیریاب خاص تقاضا نماید. با این کار مسیریاب می‌تواند ضمن درخواست جدول مسیریابی همسایه‌های خود و مقایسهٔ شمارهٔ ترتیب رکوردهای آن اقدام به تازه‌سازی جدول خود نماید. قالب این پیام در شکل (٣٤-٤) نشان داده شده است .

- پیام Link State Ack : این پیام توسط گیرندهٔ پیام Link State Update برای ارسال کنندهٔ آن ارسال می‌شود و به منظور تصدیق دریافت جدول مسیریابی می‌باشد.

به گونه‌ای که اشاره شد تمام پیامهای تعریف شده در پروتکل OSPF دارای سرآیند ٢٤ بایتی هستند و در اشکال (٣١-٤) تا (٣٤-٤) به رنگ تیره نشان داده شده است. این فیلدها به شرح زیرند:

- **فیلد Version** : نسخهٔ پروتکل OSPF را تعیین می‌کند. ( فعلاً این شماره ۱ است )
- **فیلد Type** : این فیلد یکی از انواع پنج‌گانهٔ پیام را که در بالا معرفی شد ، مشخص می‌کند. معنای مقادیر مختلف این فیلد عبارتند از :

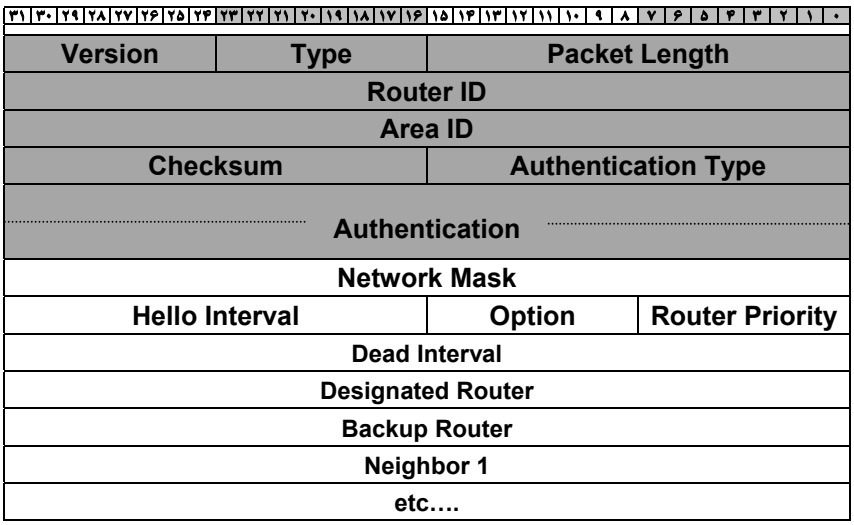
| | |
|---|---|
| ۱ | **Hello Message** |
| ۲ | **Database Description Message** |
| ۳ | **Link State Request Message** |
| ٤ | **Link State Update Message** |
| ٥ | **Link State Acknowledgement Message** |

- **فیلد Packet Length** : این فیلد طول کل پیام را بر حسب بایت مشخص می‌نماید. ( طول پیام شامل قسمت سرآیند پیام نیز هست )
- **فیلد Router ID** : این فیلد چهاربایتی "مشخصهٔ مسیریاب"[1] ارسال‌کنندهٔ پیام را تعیین می‌نماید.
- **فیلد Area ID** : این فیلد چهاربایتی "مشخصهٔ ناحیه‌ای" که مسیریاب ارسال‌کنندهٔ پیام به آن ناحیه متعلق است ، تعیین می‌نماید.
- **فیلد Checksum** : این فیلد ، یک کد کشف خطای ١٦ بیتی از کل پیام (شامل سرآیند و بدنهٔ پیام ) است که با روشی که برای بسته‌های IP گفته شد ، محاسبه می‌شود.
- **فیلد Authentication Type** : این فیلد که فعلاً دو مقدار برای آن تعریف شده مشخص می‌کند که آیا برای این پیام ، احراز هویت انجام شود یا خیر :

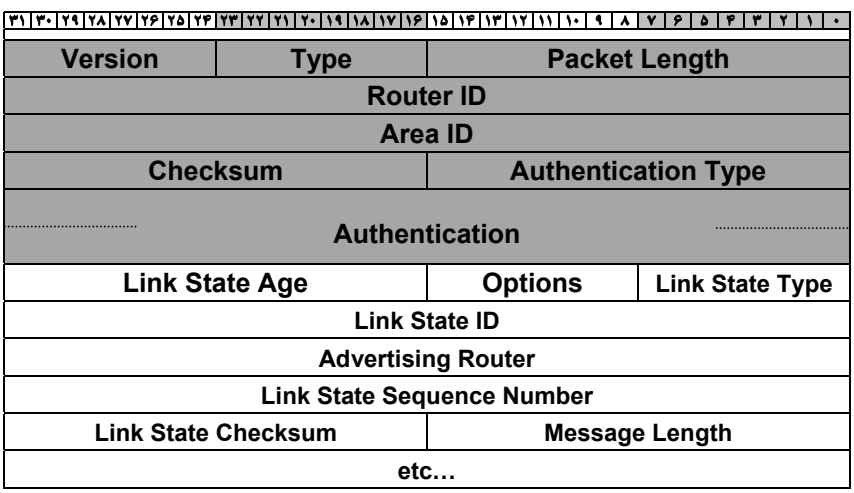
  مقدار ۰ : بدون احراز هویت (با این مقدار فیلد ٦٤ بیتی بعدی اهمیتی نخواهد داشت.)

  مقدار ۱ : احراز هویت وجود دارد و مقدار فیلد ٦٤ بیتی بعدی کلمهٔ عبور مسیریاب را تعیین می‌کند.
- فیلد Authentication : این فیلد در حقیقت کلمهٔ عبور مسیریاب به شمار می‌آید ودر هنگام نصب و تنظیم شبکه ، توسط مسئول آن تعیین می‌شود. مسیریابهای دریافت کنندهٔ این بسته به شرطی آنرا می‌پذیرند که کلمهٔ عبور آن مسیریاب معتبر و تعریف‌شده باشد.

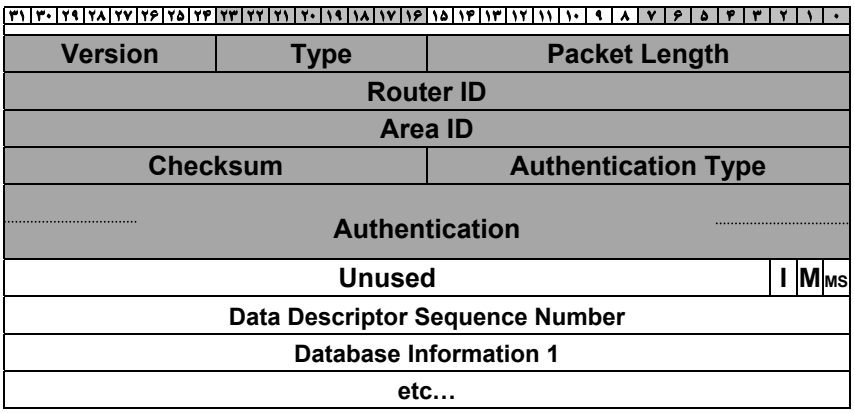
[1] Router Identifier



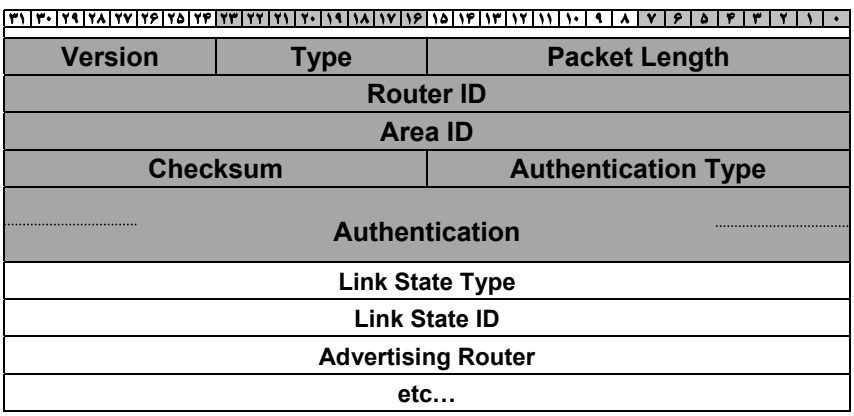

شکل (۳۱-٤) قالب پیام **"سلام"** -Hello Packet- در پروتکل OSPF




شکل (۳۲-٤) قالب پیام Link State Update در پروتکل OSPF




شکل (۳۳-٤) قالب پیام ارسال بانک اطلاعاتی در پروتکل OSPF




شکل (۳٤-٤) قالب پیام "تقاضا" در پروتکل OSPF

**دقت کنید که هدف از ارائهٔ قالب پیامهای پروتکل OSPF ، آشنایی کلی با ساختار آنها و مقایسه با نکاتی است که در مورد تشکیل بستههای LS در بخشهای قبلی به آن اشاره شد. برای آشنایی دقیق با این پروتکل نیاز به مستندات IETF دارید که خود به اندازهٔ یک کتاب ، مفصل است. مراجع مختلفی که میتوانید به آنها مراجعه کنید در آخر این کتاب آمده است. در ضمن از سایت زیر هم میتوانید استفاده کنید:**

**http://www.cis.ohio-state.edu/hypertext/information/rfc.html**

## ۸) پروتکل BGP : پروتکل مسیریابی برونی[۱]

همانگونه که مؤکداً اشاره کردیم در شبکه‌ای مثل اینترنت ، مسیریابی در درون یک شبکهٔ خودمختار (AS) با مسیریابی خارج از شبکه‌های خودمختار ، کاملاً متفاوت است و از پارامترهای متنوع و متناقضی تبعیت می‌کند. "مسیریابی برونی" نه تنها تابع شرایط ترافیکی ، توپولوژیکی ، پهنای باند و سرعت پردازش مسیریابها است ، بلکه از یکسری سیاستهای اقتصادی ، امنیتی و ملّی نیز تاثیر می‌پذیرد. شاید این نکته جالب باشد که مسیریابهائی که بین شبکه‌های خودمختار عمل می‌کنند نه تنها بایستی در مورد بهترین مسیر از لحاظ تاخیر کمتر و سرعت بیشتر تصمیم بگیرند بلکه بایستی در این تصمیم گیری مسائل سیاسی ، اقتصادی و امنیتی را نیز بعنوان جزئی از الگوریتم دخالت بدهند. بعنوان مثال یک مسیریاب در آمریکا شاید نخواهد بسته های IP که مبداً آنها مراکز نظامی و امنیتی آمریکا هستند از مسیرهائی عبور کنند که کشورهایی مثل کره شمالی ، عراق و لیبی در آن مسیر قرار دارند؛ یا مثلاً امنیت اقتصادی ایجاب می‌کند ، بسته‌های IP که مبداً آن IBM است بهیچوجه از مسیریاب Microsoft عبور ننمایند.

الگوریتمهای مسیریابی بین شبکه‌های خودمختار در اینترنت ، BGP[۲] نامیده می‌شود و تمامی این پارامترها را در تصمیم گیری لحاظ می‌کند. شبکه‌های خودمختار را یک ستون‌فقرات خارجی به هم متصل کرده که مسیریابهای نوع BGP بر روی آن قرار گرفته‌اند. بنابراین مسیریابهای نوع BGP از طریق ستون فقرات شبکهٔ اینترنت با مسیریابهای دیگر از نوع BGP در ارتباطند . ممکن است بین دو مسیریاب از نوع BGP یک شبکهٔ خودمختار قرار گرفته باشد که کل آن شبکه یک خط واحد در نظر گرفته می‌شود (به شکل (۲۰-٤) دقت کنید). حال به اختصار بررسی می‌کنیم که این مسیریابها چگونه عمل می‌نمایند:

در پروتکل BGP بجای آنکه جداول مسیریابی و هزینه‌ها بین مسیریابهای مجاور مبادله شود ، در بازه‌های زمانی T ثانیه‌ای ، فهرستی از مسیرهای کامل بین هر دو مسیریاب در شبکه ، برای مسیریابهای مجاور ارسال می‌شود. ( بدون تعیین هزینه )

برای روشن شدن قضیه فرض کنید در شکل (۳٤-٤) مسیریاب F از مسیریابهای مجاور خود یعنی B ، I ، G و E ، اطلاعاتی در مورد D بصورت زیر دریافت کند:

تعیین مسیر رسیده از B From B : "I use BCD"

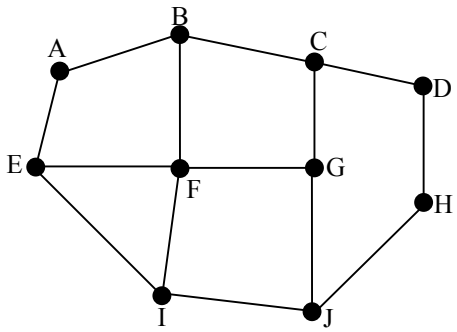
تعیین مسیر رسیده از G From G : "I use GCD"

تعیین مسیر رسیده از I From G : "I use IFGCD"

تعیین مسیر رسیده از E From G: "I use EFGCD"

---

[۱] The Exterior Gateway Routing Protocol
[۲] Border Gateway Protocol




شکل (۳۵-٤) ساختار فرضی از ارتباط بین مسیریابهای BGP

البته از B و G و I و E مسیرهای دیگری که به سایر مسیریابها وجود دارد ، اعلان می‌شود ولی برای سادگی در مثال فوق ، فقط مسیرهائی که به D ختم می‌شوند ارائه شده است. مسیریاب F برای یافتن راهی به D چهار مسیر پیشنهاد شده ، توسط همسایگانش را بررسی می‌کند و همان ابتدا مسیرهای I و E را کنار می‌گذارد چرا که این مسیرها از خود F عبور می‌کنند و بنابراین مسیری مستقل محسوب نمی‌شوند.

بنابراین F برای رسیدن به D فقط دو راه خواهد داشت که یکی BCD و دیگری GCD است و بایستی یکی از این دو مسیر را انتخاب کند. در اینجاست که باید تمامی پارامترهای ترافیکی ، توپولوژیکی ، اقتصادی ، سیاسی و امنیتی برای انتخاب بهینه‌ترین مسیر ، محاسبه شود. بهمین دلیل نحوه **"امتیازدهی"**[۱] به مسیرها بعنوان بخشی ازپروتکل BGP محسوب نمی‌شود و به ملاحظات مدیریتی هر مسیریاب وابسته است و توسط مسئول شبکه تنظیم می‌شود.

الگوریتم BGP مشکل "شمارش تا بینهایت"[۲] را نخواهد داشت : بعنوان مثال فرض کنید خط بین F و G قطع شود. F اطلاعاتی را از سه مسیریاب مجاور دریافت می‌کند که اعلان کرده‌اند برای رسیدن به D ، از مسیرهای BCD ، IFGCD ، EFGCD استفاده میکنند و چون در بین مسیرهای IFGCD ، EFGCD ، خط FG قرار دارد ، بنابراین نمی‌توانند به عنوان مسیرهایی مستقل مورد استفاده قرار گیرند و تنها مسیر باقیمانده BCD خواهد بود.

---

۱ Scoring Function
۲ Count To Infinity

الگوریتمهائی که در تبادل اطلاعات با همسایگان مسیرهای کامل را به اطلاع یکدیگر می‌رسانند اولاً مشکل "شمارش تا بینهایت" را نخواهد داشت؛ ثانیاً مسیریابهای دیگر میتوانند بر روی کل مسیر ، بررسی‌های امنیتی ، اقتصادی ، سیاسی و ملی انجام دهند و بر اساس این پارامترها مسیر مناسب را انتخاب نمایند.

در ادبیات پروتکل BGP ، مسیریابهای مجاور "مسیریابهای همتا"[۱] نامیده شده است. اطلاعات مسیریابی (فهرست مسیرها) در قالب پیامهایی بین مسیریابهای همتا مبادله می‌شود. در پروتکل BGP چهار نوع پیام تعریف شده است:

- **پیام OPEN** : مسیریابهای همتا از طریق پروتکل TCP و شمارهٔ پورت ۱۷۹ با یکدیگر مبادلهٔ اطلاعات می‌کنند. ( در فصل بعد خواهید دید که استفاده از پروتکل TCP علیرغم اندکی کاهش سرعت انتقال ، صحت داده‌ها را تضمین می‌کند. ) وقتی یک مسیریاب به شبکه وارد می‌شود با ارسال این پیام برای مسیریابهای همتای خود ، ضمن اعلام موجودیت ، خود را معرفی می‌کند تا دیگران بتوانند هویت او را احراز کنند. اگر مسیریابهای همتا او را بپذیرند ، در پاسخ پیام KEEPALIVE را به عنوان تصدیق ، برخواهند گرداند.
- **پیام KEEPALIVE** : این پیام دو نوع کاربرد دارد:

کاربرد اول زمانی است که یک مسیریاب با پیام OPEN در شبکه اعلام حضور می‌کند؛ در این حالت مسیریابهای همتا پس از احراز هویت ، این پیام را به عنوان پیغام تصدیق (Ack) برمی‌گردانند.

کاربرد دوم زمانی است که یک مسیریاب BGP در موعد اعلان "فهرست مسیرها" ، چیزی برای ارسال نداشته باشد؛ در این حالت با ارسال این پیام اعلام می‌کند که در شبکه حضور داشته و فعال است. اگر مسیریابی در یک بازهٔ زمانی مشخص هیچ پیامی از مسیریاب همتای خود دریافت نکند ، فرض خواهد کرد که آن مسیریاب به هر دلیلی از شبکه خارج شده است و بر این اساس فهرست مسیرهای خود را اصلاح خواهد کرد.

- **پیام NOTIFICATION** : با این پیام یک مسیریاب به همتای خود اعلام می‌کند که در دریافت پیام قبلی او خطایی رخ داده است. در حقیقت این پیام نقش "اعلان عدم تصدیق" (NAck[۲]) را بازی می‌کند.
- **پیام UPDATE** : مسیریاب با این پیام ، مسیر مورد استفاده‌اش برای رسیدن به یک مقصد خاص در شبکه را ، به اطلاع همتای خود می‌رساند. در ضمن اگر مسیری که قبلاً اعلان شده ، دیگر معتبر و در دسترس نباشد با این پیام به اطلاع مسیریابهای همتا می‌رسد.

---

[۱] Peer BGP Router
[۲] Not Acknowledged

برای اطلاع بیشتر از جزئیات پروتکل BGP به مستندات RFC-1263 و RFC-1654 مراجعه نمائید.

## ۹) مراجع این فصل

مجموعهٔ مراجع زیر می‌توانند برای دست آوردن جزییات دقیق و تحقیق جامع در مورد پروتکلهای معرفی شده در این فصل مفید واقع شوند.

### مراجع مفید در مبحث مسیریابی


| | |
|---|---|
| **Cisco97** | "Interior Gateway Routing Protocol and Enhanced IGRP " http://www.cisco.com/univercd/cc/td/doc/csintwk/ito_doc/55182.htm |
| **Halabi 97** | B. Halabi, Internet Routing Architectures, Cisco Systems Publishing, Indianapolis, 1997. |
| **Huitema** | C. Huiteman, Routing in the Internet, Prentice Hall, New Jersey, 1995. |
| **RFC1058** | "Routing Information Protocol," Hedrick, C.L.; 1988 |
| **RFC1074** | "NSFNET Backbone SPF-Based Interior Gateway Protocol," Rekhter, J.; 1988 |
| **RFC1136** | "Administrative Domains and Routing Domains: A Model for Routing in the Internet," Hares, S.; Katz, D.; 1989 |
| **RFC1163** | "Border Gateway Protocol (BGP)," Lougheed, K.; Rekhter, Y.; 1990 |
| **RFC1164** | "Application of the Border Gateway Protocol in the Internet," Honig, J.C.; Katz, D.; Mathis, M.; Rekhter, Y.; Yu, J.Y.; 1990 |
| **RFC1195** | "Use of OSI IS-IS for Routing in TCP/IP and Dual Environments," Callon, R.W.; 1990 |
| **RFC1222** | "Advancing the NSFNET Routing Architecture," Braun, H.W.; Rekhter, Y.; 1991 |
| **RFC1247** | "OSPF version 2," Moy, J.; 1991 |
| **RFC1256** | S. Deering, "ICMP Router Discovery Messages," RFC 1256, Sept. 1991. |
| **RFC1267** | "A Border Gateway Protocol 3 (BGP-3)," Lougheed, K.; Rekhter, Y.; 1991 |
| **RFC1584** | J. Moy, "Multicast Extensions to OSPF," RFC 1584, March 1994. |
| **RFC1723** | G. Malkin, RIP Version 2 - Carrying Additional Information. RFC 1723, November 1994. |
| **RFC1771** | Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP-4)," RFC 1771, March 1995. |
| **RFC1772** | Y. Rekhter and P. Gross, "Application of the Border Gateway Protocol in the Internet," RFC 1772, March 1995. |
| **RFC1773** | P. Traina, "Experience with the BGP-4 protocol," RFC 1773, March 1995 |
| **RFC2002** | C. Perkins, "IP Mobility Support," RFC 2002, 1996. |
| **RFC2178** | J. Moy, "Open Shortest Path First Version 2", RFC 2178, July 1997. |
| **RFC823** | "DARPA Internet Gateway," Hinden, R.M.; Sheltzer, A.; 1982 |
| **RFC827** | "Exterior Gateway Protocol (EGP)," Rosen, E.C.; 1982 |
| **RFC888** | "STUB Exterior Gateway Protocol," Seamonson, L.; Rosen, E.C.; 1984 |
| **RFC904** | "Exterior Gateway Protocol Formal Specification," Mills, D.L.; 1984 |
| **RFC911** | "EGP Gateway under Berkeley UNIX 4.2," Kirton, P.; 1984 |

## مراجع مفید در مبحث کارایی و سیاستهای مسیریابی


| | |
|---|---|
| **RFC1102** | "Policy Routing in Internet Protocols," Clark, D.D.; 1989 |
| **RFC1104** | "Models of Policy-Based Routing," Braun, H.W.; 1989 |
| **RFC1124** | "Policy Issues in Interconnecting Networks," Leiner, B.M.; 1989 |
| **RFC1125** | "Policy Requirements for Inter-Administrative Domain Routing," Estrin, D.; 1989 |
| **RFC1245** | "OSPF Protocol Analysis," Moy, J., ed; 1991 |
| **RFC1246** | "Experience with the OSPF Protocol," Moy, J., ed.; 1991 |
| **RFC1254** | "Gateway Congestion Control Survey," Mankin, A.; Ramakrishnan, K.K, eds.; 1991 |

## ۱) لایهٔ انتقال[۱] در شبکهٔ اینترنت

بگونه‌ای که در فصل قبل اشاره شد پروتکل IP وظیفهٔ هدایت و مسیریابی بسته های اطلاعاتی را از یک ماشین میزبان به ماشینی دیگر برعهده دارد و مشکلاتی که در طی مسیر ممکن است برای یک بستهٔ IP اتفاق بیفتد ، توسط این لایه قابل حل نیست.

وظیفهٔ لایهٔ انتقال در شبکه ، " فراهم آوردن خدمات سازماندهی شده ، مطمئن و مبتنی بر اصول سیستم عامل ، برای برنامه های کاربردی در لایهٔ بالاتر است ، بگونه‌ای که مشکلات و ناکارآمدی لایهٔ IP جبران و ترمیم شود ".

در مقام مقایسه ، وظیفه‌ای را که لایهٔ انتقال بر عهده دارد ، می‌توان با وظایفی که "سیستم مدیریت فایل"[۲] به عنوان بخشی از سیستم عامل بر عهده دارد ، قیاس کرد. سیستم مدیریت فایل از یک طرف با ابزارهای ذخیره سازی اطلاعات که ذاتاً سخت افزاری ، متنوع و ناهمگون هستند ، سر و کار دارد و از طرف دیگر با برنامه های کاربردی در ارتباط است که برای ذخیره و بازیابی اطلاعات فقط مفهومی به نام فایل ، در اختیار دارد و از دید برنامه نویس نوع ابزار و چگونگی و محل فیزیکی ذخیرهٔ داده هایش مهم نیست ، بلکه فقط عملیات لازم را برنامه ریزی میکند. از دیدگاهِ ابزارهای ذخیره و بازیابی اطلاعات ، چیزی به نام فایل ، درایوهای منطقی (مجازی) و جدول FAT[۳] معنایی ندارد ، بلکه این ابزار میتوانند یک بلوک داده را با اندازهٔ ثابت ، تحویل گرفته و بر روی محل مشخصی از فضای فیزیکی ذخیره سازی اطلاعات بنویسند ( یا بخوانند ) . سیستم مدیریت فایل که بین این ابزار فیزیکی و برنامه های کاربردی قرار میگیرد از یک ابزار فیزیکیِ خام ، یکپارچه و پیچیده ، خدماتی را در قالب مفهوم فایل به برنامه های کاربردی ارائه میکند که کاملاً قابل اعتماد ، شفاف ، ساده و عاری از هرگونه پیچیدگی سخت افزاری است. سیستم مدیریت فایل برای ارائهٔ چنین خدماتی باید جداول FAT ، جدول درایوهای منطقی[۴] ، سیستم فهرست فایلها[۵] و ... را ایجاد و سازماندهی نماید. تنها کاری که برنامه نویس برای بهره گیری از خدمات سیستم فایل باید انجام بدهد آنست که فایلی را بگشاید و تقاضای خواندن از آن یا نوشتن در آن را بدهد. پیچیدگی هایی که در این بین وجود دارد توسط مدیر فایل حلّ و فصل می‌شود.

وظیفهٔ لایه انتقال همین مفهوم را دنبال میکند یعنی : " بهره گیری از خدمات لایهٔ IP که سریع و ساده و در عین حال غیرمطمئن و ناکارآمد است و ارائهٔ خدماتی مطمئن ، ساختاریافته

---

۱ Transport Layer
۲ File Management System
۳ File Allocation Table
۴ Partition Table
۵ Root Directory

و شفاف به برنامه های کاربردی در لایهٔ بالاتر ، به گونه‌ای کـه برنامـه نویـس از درگـیری بـا جزییات زیرشبکه و مشکلات کانالهای انتقال و مسایلی از این قبیل به دور باشد".

برای تشریح وظایف لایهٔ انتقال باید کاستی های لایهٔ IP را بررسی کرده و سپس روشـی را که لایهٔ انتقال برای جبران آنها برگزیده است ، توضیح بدهیم. دقت کنید که منشأ کاستی هـای لایهٔ IP ، ذاتِ کانالهای انتقال و مشکلات فیزیکی در زیرشبکهٔ ارتباطی است. عمدهٔ این کاستی ها عبارتند از:

- تضمینی وجود ندارد وقتی بسته‌ای برای یک ماشین مقصد ارسـال میشـود آن ماشـین آمـادهٔ دریافت آن بسته باشد و بتواند آنرا دریافت کند.
- تضمینی وجود ندارد وقتی چند بستهٔ متوالی برای یک ماشین ارسال می‌شود به همان ترتیبـی که بر روی شبکه ارسال شده‌اند ، در مقصد دریافت شوند.
- تضمینی وجود ندارد که وقتی بسته‌ای برای یک مقصد ارسال می‌شود ، به دلیل دیـر رسـیدن مجدداً ارسال نشود و در چنین حالتی ممکن است بسته‌ای به اشتباه دو بار[1] در مقصد دریـافت شود. لایهٔ IP قادر نیست تمایزی بین دو بستهٔ عین هم ، که یکی از آنها زائد است قائل شـود و هر دو را تحویل ماشین مقصد می‌دهد.
- لایهٔ IP هیچ وظیفه‌ای در قبال توزیع بسته ها بین پروسه های مختلفی که بر روی یک ماشین واحد اجرا شده‌اند ندارد. در یک محیط "چند کاربره"[2] یا "چند وظیفه‌ای"[3] ممکن است چندین پروسهٔ متفاوت تقاضای ارسال یا دریافت داده داشته باشند. حال فرض کنید بسته‌ای به لایـهٔ IP از یک ماشین واحد ، تحویل داده شود. داده های درون این بسته متعلق به کدامین پروسـهٔ در حال اجرا روی آن ماشین است؟ از دیدگاه لایهٔ IP مفهومی به نام "پروسه های متفاوتِ در حال اجرا" ، رسمیّت و هویّت ندارد.
- لایهٔ IP هیچ وظیفه‌ای در قبال تنظیم سرعت تحویل بسته ها به یک ماشین ندارد. مثلاً ممکـن است یک ماشین با سرعت بسیار زیاد بسته هایی را تولید کرده و تحویـل لایـهٔ IP بدهـد ولـی ماشین مقصد قادر نباشد بسته ها را با این سرعت دریافت کند و بسته هـا در مقصـد بـه دلیـل عدم توانایی در دریافت ، از بین بروند.

در لایهٔ انتقال دو پروتکل به نامهای TCP[4] و UDP[1] تعریف شده‌اند که ابتدا پروتکـل TCP را که تمام کاستی های عنوان شده را جبران کرده معرفـی می‌کنیـم و نـهایتاً بـه پروتکـل UDP و مشخصات آن خواهیم پرداخت.

---

1 Duplication Problem
2 Multi User
3 Muti Task
4 Transmission Control Protocol

## ۲) راهکارهای پروتکل TCP برای جبران کاستی های لایهٔ IP

در این بخش مفهوم عملیاتی که پروتکل TCP برای جبران کاستیهای لایهٔ IP انجام میدهد ، بررسی می‌شود و سپس جزئیات این عملیات را در بخشهای آتی ارائه می‌دهیم.

اولین کاستی در لایهٔ IP ، عدم تضمین در آماده بودن و توانایی دریافت داده ها توسط ماشین مقصد ، عنوان شد. در پروتکل TCP راهکاری ساده وکارآمد برای این مشکل اتخاذ شده است: " برقراری یک ارتباط و اقدام به هماهنگی بین مبدأ و مقصد ، قبل از ارسال هرگونه داده".

برای تشریح این راه حل ، فرض کنید پروسهٔ A تمایل داشته باشد برای پروسهٔ B بر روی یک ماشین مشخص ، داده هایی را ارسال کند؛ قبل از اقدام به ارسال داده به صورت زیر عمل می‌کند:

الف) A یک بستهٔ خاص را به عنوان درخواست برای ارتباط ، به آدرس ماشین B می‌فرستد و منتظر می‌ماند.

ب ) B درخواست ارتباط را دریافت کرده و بر حسب شرایط ، آمادگی یا عدم آمادگی خود را به A اعلام مینماید. ( ممکن است B اصلاً وجود خارجی نداشته باشد و طبعاً هیچ پاسخی بر نمیگردد. )

ج ) در صورتی که A در یک مهلت زمان مشخص ، پاسخ مثبت مبنی بر آماده بودن B دریافت نماید میتواند به ارسال داده ها اقدام نماید.

به پروتکلهایی که قبل از مبادلهٔ داده ها سعی در برقراری یک ارتباط و ایجاد هماهنگی قبلی می‌نمایند پروتکلهای "اتصال گرا"[۲] گفته میشود. در این پروتکلها خاتمهٔ مبادلهٔ داده ها نیز بایستی در یک روند هماهنگ و با اطلاع قبلی انجام شود:

الف) A خاتمهٔ ارسال داده های خود را اعلام می‌کند ولی باید منتظر بماند و به دریافت داده های ارسالی از طرف B ادامه بدهد تا آ نکه B نیز اعلام ختم ارتباط را تایید کند.

ب ) B نیز اعلام ختم ارتباط کرده و ارتباط ، در یک روند هماهنگ خاتمه می‌یابد.

البته ممکن است اتفاقاتی در خلال مبادلهٔ داده ها رخ بدهد که توسط لایهٔ TCP قابل حل نباشد ( مثل خرابی خط یا خاتمه ناهنگام یکی از برنامه های A یا B توسط سیستم عامل در اثر

---

[۱] User Datagram Protocol
[۲] Connectuion Oriented

بروز یک مشکل داخلی[۱]، یا هر اتفاق دیگر ) این مشکلات در پروتکل TCP توسط زمان سنج های خاصّی کشف و مدیریت می‌شوند که در بخشی مجزا به آنها خواهیم پرداخت.

معضلات بعدی در لایهٔ IP تضمین به ترتیب رسیدن داده ها و صحّت آنهاست. حلّ این مسایل چندان مشکل نیست. مجدداً فرض کنید پروسهٔ A تمایل داشته باشد برای پروسهٔ B بر روی یک ماشین مشخص ، داده هایی را ارسال کند و قبل از اقدام به ارسال داده ها یک ارتباط موفق برقرار کرده باشد. برای تضمین صحّت و ترتیب داده ها روند زیر قابل انجام است:

الف) A بخشی از داده هایی که باید ارسال شوند را در قالب یک بسته سازماندهی کرده و در سرآیند آن یک "شمارهٔ ترتیب"[۲] تنظیم می‌نماید؛ سپس ضمن نگهداری آن بسته درون یک بافر ، آن را جهت هدایت به سمت مقصد ، تحویل لایهٔ IP می‌دهد و یک "زمان سنج" تنظیم مینماید. همچنین برای نظارت بر خطاهای احتمالی یک کد ۱۶ بیتی کشف خطا در سرآیند بسته قرار می‌دهد.

ب ) در صورتی که B بستهٔ ارسالی از A را سالم دریافت کرد ، یک "پیغام تصدیق" که اختصاراً Ack نامیده میشود برای A پس می‌فرستد.[۳]

ج ) اگر A در زمان مقرّر پیغام Ack را دریافت کرد ، بافر مربوط به آن بسته را آزاد کرده و اقدام به ادامهٔ ارسال داده ها به همین روال مینماید. اگر به دلیل خرابی داده ها ( یا خرابی پیغام Ack در مسیر برگشت ) در مهلت مقرر پیغام تصدیق دریافت نشود ، بسته بافرشده از نو ارسال میشود.[۴]

با قرار دادن شمارهٔ ترتیب برای داده ها می‌توان تضمین کرد که جریان داده ها به ترتیب می‌رسند و به هر دلیلی اگر بسته‌ای دو بار دریافت شوند ، با مقایسهٔ شماره های ترتیب ، یکی از آنها دور انداخته می‌شود.

با تنظیم یک کد ۱۶ بیتی کشف خطا در مبدأ و بررسی مجدّد آن در مقصد ، می‌توان از صحت داده ها نیز مطمئن شد. جزئیات این عملیات با تشریح پروتکل TCP مشخص خواهد شد.

---

[۱] Bug

[۲] Sequence Number

[۳] ارسال Ack معمولا بصورت مجزا ارسال نمیشود بلکه ضمیمهٔ اطلاعاتی میشود که قرار است در پاسخ ، ارسال شود ، مگر آنکه داده ای برای ارسال وجود نداشته باشد؛ به این روش Piggy Backing گفته میشود.

[۴] به پروتکلهایی که فقط در هنگام دریافت صحیح داده ها پیغام Ack برمیگردانند و در صورت دریافت بستهٔ خراب ساکت میمانند ، پروتکلهای Positive Acknowledgement with Retransmission- PAR- گفته میشود.

در پروتکل TCP برای به رسمیّت شناختن پروسه های مختلفی که بر روی یک ماشین در حال اجرا هستند راه حل زیر ارائه شده است:

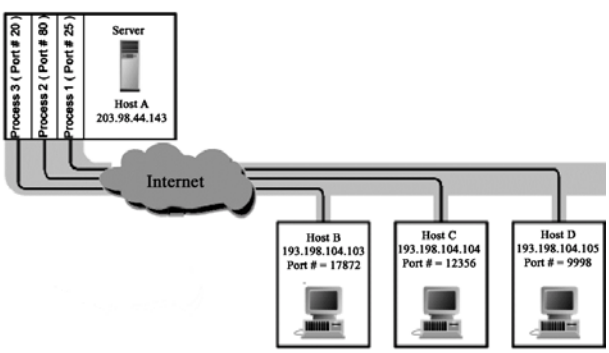
هر پروسه برای تقاضای برقراری یک ارتباط با پروسه‌ای دیگر روی شبکه ، یک شمارهٔ شناسایی برای خود برمی‌گزیند. به این شمارهٔ شناسایی "آدرس پورت"[1] گفته می‌شود. در سرآیند بسته‌ای که توسط پروتکل TCP سازماندهی می‌شود آدرس پورت پروسهٔ فرستنده و آدرس پورت پروسهٔ گیرندهٔ آن درج می‌شود. یکتا بودن شماره های پورت که به پروسه ها رسمیت و هویت می‌بخشد ، توسط پروتکل TCP به عنوان جزئی از سیستم عامل نظارت خواهد شد. سیستم عامل جدولی را نگهداری میکند که شماره شناسایی تقاضا دهندهٔ ارتباط در آن وجود دارد.

به شکل (۱-۵) دقت کنید. آدرس IP ، یک ماشین یکتا را در کل شبکه مشخص می‌نماید؛ شمارهٔ پورت نیز از بین پروسه های اجرا شده بر روی آن ماشین ، یکی از آنها را به عنوان مبدأ (یا مقصد) تعیین می‌کند. بنابراین زوجِ آدرس IP و آدرس پورت می‌تواند یک پروسهٔ یکتا و واحد را بر روی هر ماشین در دنیا مشخص نماید. در ادبیات شبکه به این زوج آدرس ، "آدرس سوکت" گفته می‌شود:

( IP Address **:** Port Number ) = Socket Address

مثال : 193.142.22.121**:**80

( البته اصطلاح "آدرس سوکت" نباید با مفهوم "برنامه نویسی سوکت" اشتباه شود ، آنرا در فصلی جداگانه به تفصیل بررسی خواهیم کرد.)




شکل (۱-۵) آدرس دهی پروسه ها بوسیلهٔ شماره پورت روی یک ماشین واحد

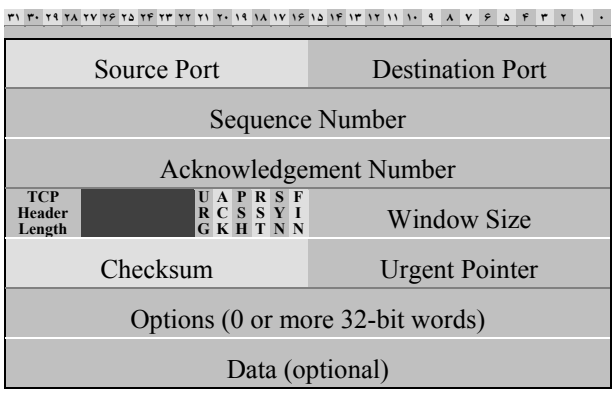
[1] Port Number

برای حل مسئلهٔ هماهنگی سرعت ارسال و دریافت در پروتکل TCP الگوریتمی پویا برای تنظیم مجموعهٔ زمان سنجهایی که در این رابطه انجام وظیفه می‌نمایند بکار گرفته شده است که در بخشی مجزا تشریح خواهد شد.

قبل از وارد شدن به جزئیات پروتکل TCP بهتر است ساختار بسته ای را که این پروتکل برای تحویل به لایهٔ IP تنظیم و سازماندهی میکند ، مورد بررسی قرار بدهیم چرا که بسیاری از مسائل با بررسی ساختار این بسته آشکار خواهد شد. ( بسته‌ای که در لایهٔ انتقال تولید و تنظیم می‌شود ، "قطعهٔ TCP"[۱] یا TPDU[۲] نام دارد ، که به اختصار به آن بستهٔ TCP خواهیم گفت. )

## ۳) ساختار بسته های پروتکل TCP

در این بخش یک دید کلی از پروتکل TCP ارائه می‌نمائیم و ساختار سرآیند بسته ها را در این پروتکل ، توضیح خواهیم داد. در شکل (۲-۵) ساختار یک بستهٔ TCP به تصویر کشیده شده است.




شکل (۲-۵) ساختار یک بستهٔ TCP

[۱] TCP Segment
[۲] Transport Protocol Data Unit

♦ **فیلد Source Port** : در این فیلد یک شمارهٔ ۱۶ بیتی بعنوان آدرس پورت پروسهٔ مبدأ که این بسته را جهت ارسال ، تولید کرده ، قرار خواهد گرفت.

♦ **فیلد Destination Port** :در این فیلد ، آدرس پورتِ پروسهٔ مقصد که آنرا تحویل خواهد گرفت ، تعیین خواهد شد .

همانگونه که در بخش قبلی اشاره شد این دو آدرس مشخص می‌کنند که این بستهٔ از چه برنامهٔ کاربردی در لایه بالاتر تولید و باید به چه برنامه‌ای در ماشین مقصد تحویل داده شود. برخی از پروسه های کاربردی و استاندارد دارای شماره پورت استاندارد و جهانی هستند؛ مثلاً سرویس دهنده پست الکترونیکی دارای شماره پورت ۲۵ است. به جدول شماره پورتهای استاندارد در انتهای این فصل نگاهی بیندازید.

♦ **فیلد Sequence Number** : این فیلد سی و دو بیتی ، شمارهٔ ترتیب آخرین بایتی را که در "فیلد داده" از بستهٔ جاری قرار دارد ، نشان می‌دهد.

در پروتکل TCP شمارهٔ ترتیب ، بر حسب شمارهٔ آخرین بایتی است که در بستهٔ جاری قرار گرفته و ارسال شده است. بعنوان مثال اگر در این فیلد عددی معادل ۱۹۳٤۱ قرار بگیرد به این معناست که داده ها تا بایت شمارهٔ ۱۹۳٤۱ درون فیلد داده قرار دارد . دقت کنید که این عدد بمعنای آن نیست که به تعداد ۱۹۳٤۱ بایت ، درون قسمت داده قرار دارد ، بلکه همیشه به شمارهٔ ترتیب آخرینِ بایت داده ، اشاره مینماید. یعنی ممکن است که کلاً درون فیلد داده فقط یک بایت قرار داشته باشد در حالی که در فیلد شمارهٔ ترتیب عدد ۱۹۳٤۱ قرار داشته باشد. دقّت شود که شمارهٔ ترتیبِ اولین بایت ، از صفر شروع نمی‌شود بلکه از یک عدد تصادفی که در هنگام برقراریِ ارتباط به اطلاع طرفین میرسد ، شروع خواهد شد.

♦ **فیلدAcknawledgement Number** : این فیلد ۳۲ بیتی نیز شمارهٔ ترتیب بایتی که فرستندهٔ بسته منتظرِ دریافت آن است را تعیین می‌کند. بعنوان مثال اگر دراین فیلد عددی معادل ۳٤۲۳۱۰ قرار گرفته باشد بدین معناست که از رشته داده ها (که مشخص نیست چند بایت است) تا شماره ۳٤۲۳۱۰ صحیح و کامل دریافت شده است و منتظر بایتهای از ۳٤۲۳۱۱ به بعد می‌باشد .

♦ **فیلد TCP Header Length** : عددی که در این فیلد قرار می‌گیرد ، طول سرآیند بستهٔ TCP را بر مبنای کلمات ۳۲ بیتی تعیین می‌کند . بعنوان مثال اگر در این فیلد عدد ۷ قرار بگیرد طول سرآیند مقدار ۲۸=۷×٤ بایت خواهد بود. ( این فیلد کلاً چهار بیتی است ) دقّت کنید که قسمت ثابت و اجباری در یک بستهٔ TCP حداقل ۲۰ بایت است ولی در فیلد اختیاری Options می‌تواند اطلاعاتی قرار و بنابراین گیرندهٔ یک

بستهٔ TCP باید بتواند مرز بین سرآیندِ بسته و قسمت داده را تشخیص بدهد . پس عددی که در این فیلد قرار می‌گیرد می‌تواند بعنوان یک "اشاره گر"[1] ، محل شروع داده ها را در یک بستهٔ TCP تعیین کند (توجه دارید که مبنای این عدد کلمات ۳۲ بیتی ( چهار بایتی ) هستند)

- **٦ بیت بلا استفاده** : پس از فیلد TCP Header Length شش بیت بلا استفاده رها شده است که شاید برای استفاده در آینده رزرو شده‌اند .
- **بیتهای Flag** : شش بیت بعدی در بستهٔ TCP هر کدام نقش یک بیت پرچم را که معنا و کاربرد مختلفی دارند را بازی میکنند.

| URG | ACK | PSH | RST | SYN | FIN |
|---|---|---|---|---|---|

تک تک این بیتها و معنای آنها را به ترتیب بررسی می‌کنیم:

- **بیت URG** : در صورتی که این بیت مقدار ۱ داشته باشد ، معین می‌کند که در فیلد Urgent Pointer که در ادامه معرفی خواهد شد مقداری قابل استناد و معتبر قرار دارد و بایستی مورد پردازش قرار گیرد . درصورتی که این بیت صفر باشد فیلد Urgent Pointer شامل مقدار معتبر و قابل استنادی نیست و از آن چشمپوشی می‌شود.
- **بیت ACK** : اگر در این بیت مقدار ۱ قرار گرفته باشد ، نشان می‌دهد که عددی که در فیلد Acknowledgement Number قرار گرفته است ، دارای مقداری معتبر و قابل استناد است . بیتACK و بیت SYN نقش دیگری نیز دارند که در ادامه بدان اشاره خواهد شد.
- **بیت PSH**[2] : اگر در این بیت مقدار ۱ قرار گرفته باشد فرستندهٔ اطلاعات از گیرنده تقاضا می‌کند که داده های موجود در این بسته را بافر نکند و در اسرع وقت آنرا جهت پردازشهای بعدی تحویل برنامهٔ کاربردی صاحب آن بدهد. این عمل گاهی برای برنامه هائی مشابه Telnet ضروری است؛ بعنوان مثال فرض کنید یک کاربر با کامپیوتر شخصی خود از نوع سازگار با IBM به کامپیوتری در فاصله هزاران کیلومتری خود وصل شده و تصمیم دارد از طریق یک محیط شبیه سازی شده با کامپیوتر سرویس دهنده ارتباط برقرار کرده و دستورات سیستم عامل UNIX را تمرین نماید. فرض کنید کاربر دستور ls (معادل dir در DOS ) را اجرا می‌کند و بالطبع توقع دارد پس از ارسال این دو کاراکتر و تحویل آن به برنامه Telnet سریعاً پاسخ لازم را روی صفحهٔ نمایشش ببیند ولی نرم افزار TCP

---

[1] Pointer
[2] Push

معمولاً در هنگام دریافت بسته های کوچک ، آنها را بافر کرده تا وقتی حجم بافر به اندازهٔ مشخصی پر شد ، آنرا یکجا تحویل برنامه کاربردی بدهد. در چنین حالتی فرستندهٔ بسته با ۱ کردن بیت PSH ، از گیرندهٔ آن می‌خواهد که آنرا بافر نکند.

- **بیت RST** : اگر در این بیت مقدار ۱ قرار بگیرد ارتباط بصورت یکطرفه و ناتمام قطع خواهد شد[۱]، بدین معنا که به هر دلیلی ( اعم از نقص سخت افزاری یا نرم افزاری) اشکالی بوجود آمده که یکی از طرفین ارتباط مجبور به خاتمه ارتباط فعلی شده است. همچنین بیت RST می‌تواند بعنوان علامت عدم پذیرش برقراری ارتباط بکار برود.اگر یکی از طرفین ارتباط یک بسته دریافت کند که در آن بیت RST مقدار۱ داشته باشد ، ارتباط بصورت ناهماهنگ و نامتعادل ، قطع خواهد شد.
- **بیت SYN** : این بیت نقش اساسی در برقراری یک ارتباط بازی می‌کند . برقراریِ یک ارتباطِ TCP از روند زیر تبعیت میکند:

الف) شروع کننده ارتباط یک بستهٔ TCP بدون هیچگونه داده و با تنظیم بیتهای (ACK=0, SYN=1) ، برای طرف مقابل ارسال می‌کند . در حقیقت ارسال چنین بسته ای به معنای "تقاضای برقراری ارتباط"[۲] تلقی می‌شود.

ب) در پاسخ به درخواست ارتباط ، در صورتیکه طرف مقابل به برقراری ارتباط تمایل داشته باشد بسته ای بر می‌گرداند که در آن بیت SYN=1 و بیت ACK=1 است . این بسته نقش "پذیرش یک ارتباط"[۳] را بازی می‌کند.

برقراری ارتباط را بیشتر توضیح خواهیم داد.

- **بیت FIN** : اگر یکی از طرفین ارتباط ، داده دیگری برای ارسال نداشته باشد در هنگام ارسال آخرین بستهٔ خود این بیت را ۱ می‌کند و در حقیقت ارسال اطلاعات خودش را یکطرفه قطع می‌کند. در این حالت اگر چه ارسال اطلاعات قطع شده و لیکن طرف مقابل هنوز ممکن است به ارسال اطلاعات مشغول باشد. زمانی ارتباط کاملاً خاتمه می‌یابد که طرف مقابل نیز در یک بسته با ۱ کردن بیت FIN ، ارسال اطلاعات را خاتمه بدهد.

♦ **فیلد Windows Size** : مقدار قرار گرفته در این فیلد مشخص می‌کند که فضای بافرِ گیرنده چند بایت دیگر ظرفیت خالی دارد. یعنی به طرف مقابل اعلام می‌کند که مجاز است از بایت با شمارهٔ ترتیبی که در فیلد Acknowledgement مشخص شده

---

[۱] Abnormaly Ended
[۲] Connection Request
[۳] Connection Accept

است، حداکثر به اندازهٔ مقداری که در این فیلد درج شده ، ارسال داشته باشد و در غیر اینصورت فضای کافی برای دریافت داده ها وجود نداشته و ناگزیر دور ریخته خواهد شد. اگر مقدار این فیلد صفر باشد بدین معناست که بافر گیرنده تماماً پر شده است و امکان دریافت داده های بعدی وجود ندارد و پروسهٔ فرستنده متوقف خواهد شد؛ در این مورد نیز بیشتر توضیح خواهیم داد.

♦ **فیلد Checksum** : در این فیلد ۱٦ بیتی ، کد کشف خطا قرار می‌گیرد. طریقه محاسبه این کد اندکی متفاوت از روشهای معمول است:

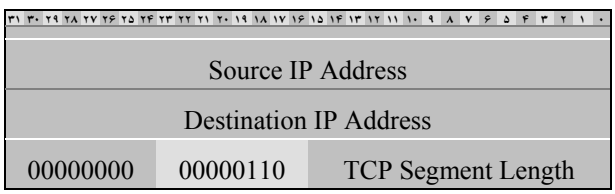
الف) کلّ بستهٔ TCP شامل قسمت سرآیند بسته و قسمت داده ، در قالب کلمات ۱٦ بیتی در نظر گرفته می‌شود. ( منهای قسمت Checkum )

ب ) یک "سرآیند فرضی"[۱] که در بستهٔ TCP وجود ندارد ، با قالب زیر ساخته شده و بصورت کلمات ۱٦ بیتی در نظر گرفته می‌شود.

ج ) تمامی کلمات در "مبنای مکمل ۱"[۲] با هم جمع شده و سپس عدد بدست آمده در مبنای مکمل ۱ منفی می‌شود. این عدد نهایتاً در فیلد Checksum قرار می‌گیرد.

اگر در حین ارسال داده ها خطائی بروز نکند ، پس از دریافت بسته در مقصد ، جمع کل کلمات ۱٦ بیتی موجود در یک بستهٔ TCP به انضمام "سرآیند فرضی" بایستی صفر شود ، در غیر اینصورت داده های غیرمعتبر و خراب هستند.

ساختار "سرآیند فرضی" که بصورت ساختگی تولید می‌شود بصورت زیر است:




همانگونه که دیده می‌شود این سرآیند ساختگی شامل فیلدهای زیر است:

- ۳۲ بیت آدرس IP مربوط به ماشین مبدأ
- ۳۲ بیت آدرس IP مربوط به ماشین مقصد
- یک فیلد هشت بیتی کاملاً صفر
- فیلد هشت بیتی پروتکل که برای پروتکل TCP یقیناً مقدار ٦ دارد.

---

[۱] Pseudo Header
[۲] 1's Compelement

- فیلد TCP Segment Length که در آن طول کل بستهٔ TCP مشخص می‌شود.

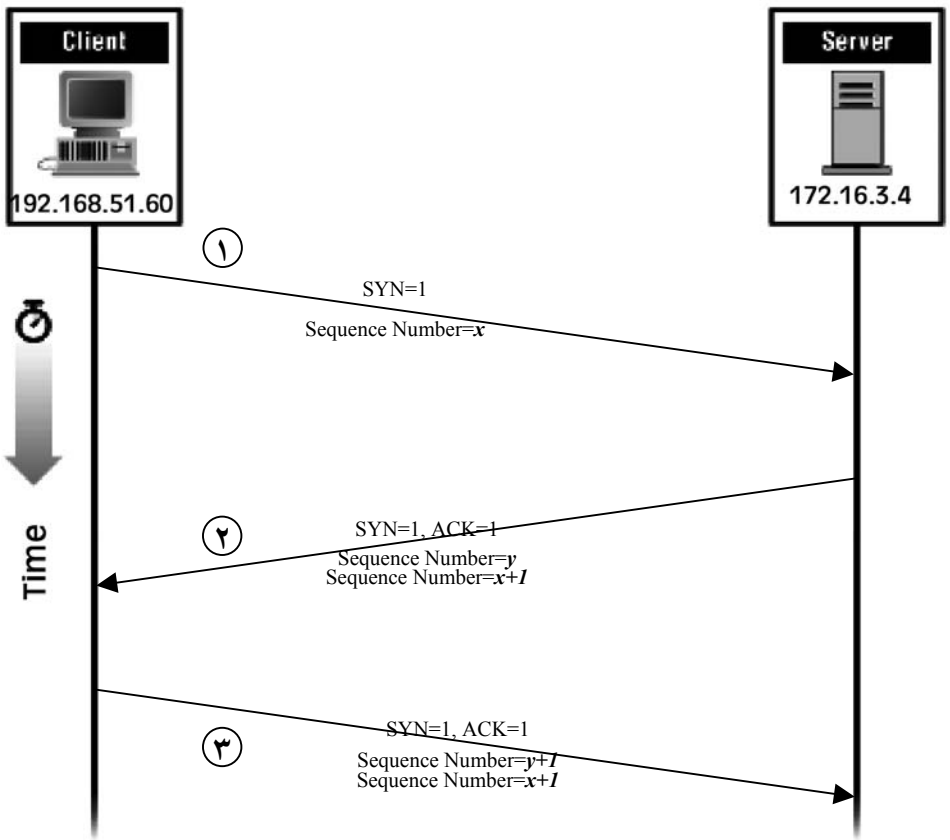
♦ **فیلد Urgent Pointer** : در این فیلد یک عدد بعنوان اشاره گر قرار می‌گیرد که موقعیت داده های اضطراری را درون بستهٔ TCP معین می‌کند . این داده ها ، زمانی اتفاق می‌افتند و ارسال می‌شوند که عملی شبیه وقوع وقفه ها در هنگام اجرای یک برنامهٔ کاربردی رخ بدهد. بدون آنکه ارتباط قطع شود داده های لازم در همین بستهٔ جاری ارسال خواهد شد. دقّت کنید که داده های اضطراری توسط برنامهٔ کاربردی در لایهٔ بالاتر پردازش خواهد شد و برای پروتکل TCP کاربردی ندارد.

♦ **فیلد Options** : در این فیلد اختیاری است و مقداری نظیر حداکثر طول بستهٔ TCP در آن قرار میگیرد. برای آنکه طول بسته ضریبی از ٤ باقی بماند از این فیلد با کدهای بی ارزش استفاده می‌شود. گزینهٔ خاص دیگری در این فیلد تعریف نشده است.

## ۴) روش برقراری ارتباط در پروتکل TCP

برای برقراری ارتباط در پروتکل TCP از روش "دست تکانی سه مرحله ای" استفاده می‌شود . البته برقراری ارتباط منوط به این قضیه است که طرفین ارتباط آمادهٔ برقراری یک ارتباط باشند یعنی یکطرف که فعلاً آنرا سرویس دهنده می‌نامیم برای برقراری ارتباط از طریق توابع سیستمی ()listen و ()accept اعلام آمادگی کرده باشد و طرف مقابل نیز یعنی مشتری با فراخوانی تابع سیستمی ()connect و تعیین آدرس IP و آدرس پورت پروسهٔ مقصد ، تمایل خود را برای ارتباط ، ابراز نماید. این توابع در فصل برنامه نویسی تحت شبکه بطور مبسوط توضیح داده خواهد شد. در چنین حالتی بین طرفین اتفاقات ۳ مرحله ای زیر خواهد افتاد (در حالت طبیعی).
در شکل (۳-۵) این مراحل به تصویر کشیده شده است.

در مرحله اوّل ، از طرف شروع کنندهٔ ارتباط ، یک بستهٔ TCP (خالی از داده ) ارسال خواهد شد که در آن بیت SYN=1 و بیت ACK=0 است و درون فیلد شماره ترتیب عدد $x$ قرار داده شده که در آن $x$ یک عدد تصادفی است. در حقیقت با این شماره به طرف مقابل اطلاع داده می‌شود که ترتیب داده های ارسالی از شماره $x+1$ شروع می‌شود . در پروتکل TCP شماره ترتیب ۳۲ بیتی است لذا برای پیشگیری از مشکلات احتمالی ناشی از مساوی بودن شمارهٔ ترتیب بسته های ارسالی ، داده ها از شماره ۰ شروع نمی‌شوند ، بلکه از یک عدد تصادفی (که بصورت خودکار تولید می‌شود) ، شروع می‌گردد و در همان مرحلهٔ اول ، این




شکل (۳-۵) مراحل دست تکانی سه مرحله ای برای برقراری ارتباط در پروتکل TCP

شمارهٔ ترتیب به طرف مقابل اعلام خواهد شد. بعنوان مثال اگر SEQ=145500 باشد بدین معناست که داده هائی که قرار است ارسال شوند شماره ترتیب آنها از ۱٤٥٥۰۱ آغاز خواهد شد. طرف مقابل حتماً باید از این موضوع باخبر باشد.

در مرحلهٔدوم ، طرف مقابل با دریافت بسته ای با مشخصات فوق الذکر اگر تمایل به برقراری ارتباط نداشته باشد با ارسال یک بستهٔ خالی که در آن بیت RST به ۱ تنظیم شده ، این تقاضا را ردّ می‌کند ولی اگر متمایل به برقراری ارتباط بود یک بستهٔ خالی از داده با مشخصات زیر تولید می‌کند:

- بیت SYN را یک می‌کند.
- بیت ACK را یک می‌کند.

- مقدار فیلد Acknowledgement Number را $x+1$ قرار می‌دهد.
- مقدار فیلد Sequence Number را مقدار تصادفی $y$ قرار می‌دهد.

در این مرحله که به معنای پذیرش ارتباط است طرف مقابل با قرار دادن مقدار فیلد Ack=$x+1$ نشان می‌دهد که شماره ترتیب $x$ را پذیرفته و منتظرداده ها از شماره ترتیب $x+1$ به بعد است . در ضمن خودش عدد تصادفی $y$ را در فیلد .Seq.No قرار می‌دهد و به طرف مقابل اعلام می‌کند که شماره ترتیب داده های ارسالی از $y$ خواهد بود.

در مرحلهٔ سوّم ، شروع کننده ارتباط با قرار دادن مقادیر زیر شروع ارتباط را تصدیق می‌کند :

- بیت SYN را یک می‌کند.
- بیت ACK را یک می‌کند.
- فیلد Seq. No.=x+1 را قرار می‌دهد.
- فیلد Ack را y+1 قرار می‌دهد.

با قرار دادن Seq.No.=$x+1$ و Ack=$y+1$ شروع کنندهٔ ارتباط اعلام می‌کند که بر روی پارامترهای شماره ترتیب توافق شده است و او پذیرفته که داده های طرف مقابل را از شماره $y+1$ بپذیرد. پس از این مرحله ارسال و دریافت داده ها توسط طرفین تا هنگامی که ارتباط با اطلاع طرفین خاتمه داده نشده است آزاد است.

برای خاتمه ارتباط روند زیر صورت می‌گیرد:

طرفی که داده هایش برای ارسال تمام شده است یک بستهٔ TCP ارسال می‌نماید که در سرآیند آن بیت FIN را یک قرار داده است. طرف مقابل این درخواست را دریافت می‌کند و با ختم یک طرفهٔ آن موافقت می‌کند. ولی چون ارتباط بصورت یکطرفه ختم می‌شود طرف مقابل می‌تواند تا جائیکه داده دارد ، آنها را ارسال کند و نهایتاً در آخرین بسته ، بیت FIN را یک بگذارد تا پس از تصدیقِ آن ، ارتباط به صورت دو طرفه ختم شود.

نکته ای که وجود دارد آنست که اگر یکی از طرفین ارتباط در اثر بروز مشکلی سخت افزاری یا نرم افزاری ارتباط را بدون هماهنگی قطع کند حق ندارد تا ۱۲۰ ثانیه به ارتباط مجدد با همان پروسه اقدام کند و این نکته ناشی از آن است که مطمئن

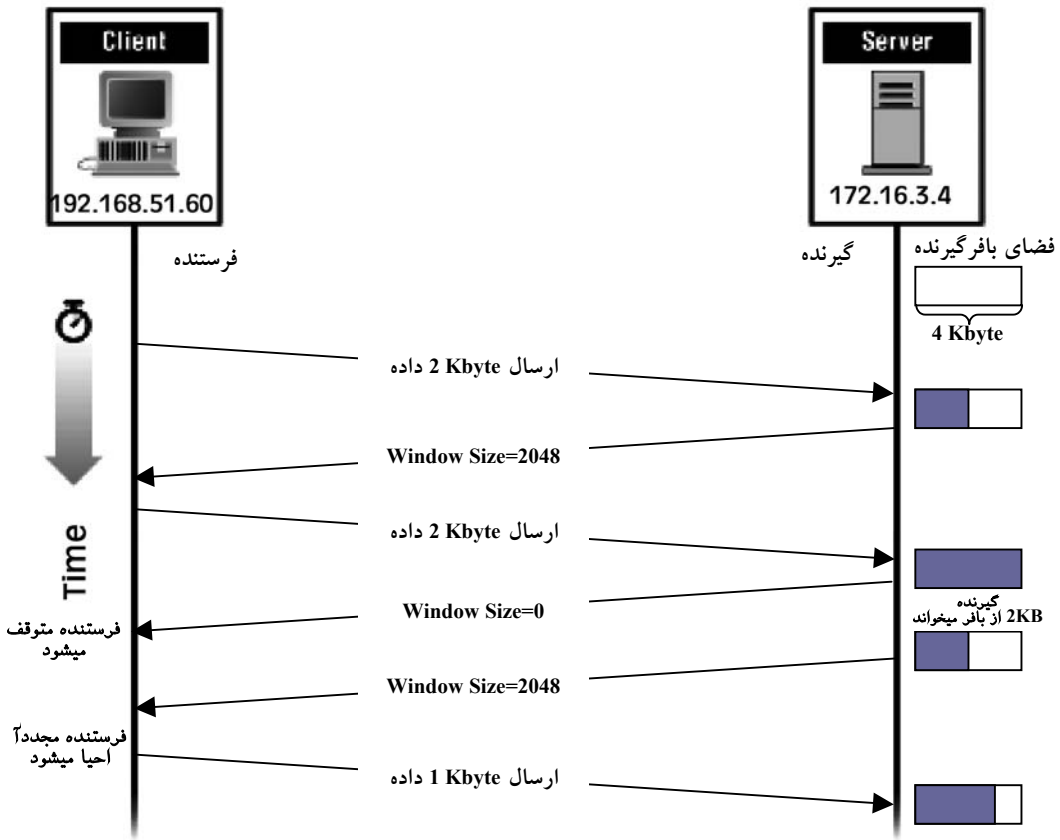
باشد بسته هـای قبلـی کـه ارسـال کـرده یـا آنکـه برایـش ارسـال شـده از زیرشـبکه حـذف شـده‌اند .

## ۵) کنترل جریان در پروتکل TCP

در اینجـا بـد نیسـت کـه اندکـی در مـورد نقـش فیلـد Window size بحـث کنیـم . همانگونـه کـه قبـلاً اشـاره شـد در پروتکـل TCP بـرای کنـترل جریـان داده هـا از بـافر اسـتفاده می‌شـود و داده هـا قبـل از ارسـال بـه برنامـه کـاربردی لایـه بـالاتر بـافر شـده و بصـورت دسـته ای تحویـل خواهـد شـد و گاهـاً ممکـن اسـت برنامـهٔ کـاربردی اقـدام بـه دریافت داده های بافر شـدهٔ خـود در مـهلت مقـرر نکـرده و بـافر پـر شـود. در ایـن حـالت گیرنده دیگر قادر بـه دریـافت و ذخـیرهٔ داده هـا در بـافرش نخواهـد بـود ، بـهمین دلیـل در هر بستهٔ TCP کـه بـه طـرف دیگـر ارسـال می‌شـود حجـم فضـای آزاد بـافر ، در ایـن فیلـد اعـلام خواهـد شـد. بعنـوان مثـال اگـر در یـک بسـتهٔ دریـافتی مقـدار فیلـد Window Size مقدار ٤٠٩٦ باشد بدیـن معناسـت کـه از کـل فضـای بـافر موجـود ، فعـلاً چـهار کیلوبـایت از آن خالی است. برای روشن شـدن قضیـه بـه شـکل (۵-۵) توجـه کنیـد.

| نام متغیر | توضیح |
|---|---|
| | **متغیرهای نظارت بر ارسال داده ها** |
| SND.UNA | شمارهٔ ترتیب آخرین بسته ای که ارسال شده ولی هنوز پیغام Ack آن برنگشته است. |
| SND.NXT | شمارهٔ ترتیب آخرین بایت که داده ها از آن شماره به بعد در بستهٔ بعدی که باید ارسال شود. |
| SND.WND | میزان فضای آزاد در بافر ارسال |
| SND.UP | شمارهٔ ترتیب آخرین داده های اضطراری که تحویل برنامهٔ کاربردی شده است. |
| SND .WL1 | |
| SND.WL2 | |
| SND.PUSH | شمارهٔ ترتیب آخرین داده هایی که باید آنی به برنامهٔ کاربردی گسیل ( Push ) شود. |
| SND.ISS | مقدار اولیهٔ شمارندهٔ ترتیب داده های دریافتی که در حین ارتباط بر روی آن توافق می‌شود. |
| | **متغیرهای نظارت بر دریافت داده ها** |
| RCV.NXT | شمارهٔ ترتیب آخرین بایت در بستهٔ بعدی که از آن شماره به بعد انتظار دریافت آنرا دارد. |
| RCV.WND | میزان فضای آزاد در بافر دریافت |
| RCV.UP | شمارهٔ ترتیب آخرین داده های اضطراری که برای برنامهٔ طرف مقابل ارسال شده است. |
| RCV.IRS | مقدار اولیهٔ شمارندهٔ ترتیب داده های ارسالی که در حین ارتباط بر روی آن توافق می‌شود. |

جدول (٤-۵) برخی از متغیرهای ساختمان دادهٔ TCB




شکل (۵-۵) مثالی از روند کنترل جریان در پروتکل TCP

در این پروتکل به ازای هر ارتباط TCP که موفقیت آمیز برقرار شود ، یک "ساختمان دادهٔ" خاص برای آن ایجاد خواهد شد که اطلاعاتی از آخرین وضعیت ارسال یا دریافت جریان داده ها در آن نگهداری می‌شود. این ساختمان داده ، "بلوک نظارت بر انتقال"[1] یا اختصاراً TCB نامیده می‌شود. برخی از متغیرهای تعریف شده درون ساختمان دادهٔ TCB در جدول (٤-۵) معرفی شده است.

عملکرد این متغیرها با تعریفی که از فیلدهای یک بستهٔ TCP داشتیم واضح و مشخص است.

[1] Transmission Control Block

## ۶) زمان سنجها در پروتکل TCP[۱]

عملکرد صحیح پروتکل TCP وابستگی شدیدی به استفادهٔ درست و منطقی از زمان سنجها دارد. در این بخش مهمترین زمان سنجهای بکار رفته در این پروتکل را بررسی می‌نماییم :

- Retransmission Timer : به گونه ای که اشاره شد پس از برقراری یک ارتباط ، وقتی بسته ای برای پروسهٔ مقصد ارسال میشود ، ضمن نگهداری موقت آن در یک بافر ، برای آن یک زمان سنج تنظیم و فعّال میشود و اگر در مهلت مقرّر پیغام دریافت آن (Ack) نرسید ، آن بسته از نو برای مقصد ارسال خواهد شد. این زمان سنج که اختصاراً **RT** نامیده میشود به یک مقدار پیش فرض ، مقداردهی میشود و شروع به شمارش معکوس زمان می‌نماید؛ هرگاه مقدار آن زمان سنج به صفر برسد ولی پیغام دریافت بسته برنگردد ، "رخداد انقضای زمان تکرار"[۲] حادث شده و پروسهٔ TCP را وادار به ارسال مجدد آن بسته می‌کند و مراحل قبلی از نو تکرار می‌شود.

عملکرد این زمان سنج بسیار ساده است ولی مسئلهٔ بغرنج در شبکه آنست که : اولاً پیش فرض این زمان سنج چه مقداری باشد؟ ثانیاً عمل ارسال مجدّد یک بسته چند بار تکرار شود؟
در شبکه های محلی سریع ، زمان رفت یک بسته و برگشت پیغام دریافتِ آن ، حدود چند هزارم ثانیه طول خواهد کشید در حالی که در شبکهٔ WAN این زمان رفت و برگشت میتواند تا چندین ثانیه طول بکشد.
اگر قرار باشد زمان پیش فرض زمان سنج RT به مقداری کم تنظیم شود ، آنگاه وقتی مقصد روی یک شبکهٔ راه دور واقع است ، قبل از آنکه بسته بتواند به مقصد برسد ، مهلت این زمان سنج منقضی شده و بسته مجدداً ارسال میشود و این کار برای هر بسته بطور متوالی تکرار می‌شود و ترافیک زائد و بیهوده ای را به شبکه تحمیل میکند.
از طرف دیگر اگر قرار باشد زمان پیش فرض این زمان سنج با مقداری بزرگ تنظیم شود در شبکه های محلی و سریع ، هنگام بروز یک خطا تاخیر زیادی بوجود خواهد آمد.
بهترین راه تنظیم زمان سنج استفاده از روشهای وفقی و پویا است چراکه راندمان پروتکل TCP به شدت به آن وابسته است. الگوریتم پویایی که معمولاً در پیاده سازی TCP بکارگرفته میشود در زیر معرفی شده است[۳] :

---

[۱] TCP Timers
[۲] Retransmission Timeout Event
[۳] این الگوریتم در سال ۱۹۸۸ توسط Jacobson معرفی شد.

الف) وقتی یک ارتباط TCP برقرار میشود یک متغیر حافظه به نـام RTT متنـاظر بـا آن ایجـاد شده و به یک مقدار پیش فرض مقداردهی میشود. این مقدار کـه حدکـثر زمـان انتظـار بـرای برگشت پیغام دریافت بسته محسوب میشود ممکن است اصلاً بهینه و مناسب نباشد.
ب ) به ازای هر بسته که ارسال میشود یک زمان سنج متناظر با آن بسته تنظیم می‌شود که زمان رفت بسته و برگشت پیغام دریافت آنرا اندازه میگیرد؛ فرض کنید برای یـک بسـته ایـن زمـان مقدار M محاسبه شده باشد.
ج ) مقدار پیش فرض زمان سنج طبق رابطهٔ زیر بهنگام میشود:

$$RTT_{new}=RTT_{old}+4*D_{new}$$

$$D_{new}=\alpha.D_{old}+(1-\alpha).(RTT_{old}-M)$$

$\alpha=7/8$ ، مقدار اولیهٔ D می‌تواند صفر باشد.

تبصره : مقدار جدید برای زمان سنج فقط به شرطی اعمال می‌شود که ارسال بستهٔ TCP تکـرار نشده باشد و فقط یکبار ارسال شده باشد[۱].

- **Keep-Alive Timer** : ممکن است طرفین یک ارتباط به هر دلیلی ارسال اطلاعات را موقتاً متوقف کنند و هیچ دادهای مبادله نشود ، هرچند ارتباط TCP فعّال و باز باشد. از سوی دیگـر ممکن است یکی از طرفین به دلیلی مثل خرابی سخت افزاری یا نرم افـزاری ، بـدون اطـلاع ، ارتباط را رها کرده باشد. برای تمایز بین این دو حالت ، فرستندهٔ اطلاعات بـا اسـتفاده از ایـن زمان سنج در بازههای زمانی منظم یک بستهٔ TCP که خالی از هرگونـه دادهای می‌باشـد بـرای مقصد ارسال می‌شود و در صورتی که پیغام دریافت آن باز گشـت ، نشـان دهنـدهٔ آنسـت کـه ارتباط TCP فعال و باز است؛ در غیر این صورت ارتباط TCP بصورت یکطرفه قطـع شـده و تمام بافرها و فضای ایجاد شده آزاد می‌شوند. زمان پیش فرض این زمان سنج مقداری بین ۵ تا ٤٥ ثانیه می‌باشد.
- **Persistence Timer** : در پروتکل TCP وقتی یکی از طرفین ارتباط ، مقدار فضای بـافر آزاد خود را در فیلد Window Size صفر اعلام کند ، ناگزیر پروسهٔ طـرف مقـابل متوقـف (بلوکـه) خواهد شد. در چنین حالتی پس از آنکه مقداری از فضـای بـافر پـر شـده تخلیـه شـد ، ایـن موضوع باید به طرف مقابل گزارش شود تا سیستم عامل ، پروسهٔ بلوکه شـده را احیـا کـرده و ادامهٔ ارسال ممکن باشد. در غیر اینصورت "بن بست"[۲] و تاخیر بینهایت بـرای پروسـه بوجـود خواهد آمد.

---

[۱] Karn's Algorithm
[۲] Deadlock

با استفاده از این زمان سنج پس از آزاد شدن فضای بافر ، در فواصل زمانی منظم یک بستهٔ TCP برای پروسهٔ بلوکه شده ارسال می‌شود تا ضمن آگاهی از آخرین وضعیت فضای بافر پروسه بتواند احیا شود.

- **Quiet Timer** : ممکن است یک ارتباط TCP بسته شود ولی هنوز بستههایی سرگردان بر روی شبکه وجود داشته باشد که پس از بسته شدن ارتباط TCP به مقصد برسند ، لذا در این پروتکل پس از بسته شدن یک ارتباط با شمارهٔ پورت خاص ، بقیهٔ پروسه ها حق استفاده از شمارهٔ پورتی که اخیراً بسته شده را ندارند. مقدار پیش فرض این زمان سنج دقیقاً دو برابر مقدار پیش فرض زمان حیات بستهٔ IP بر حسب ثانیه است. ( چیزی بین ۳۰ تا ۱۲۰ ثانیه )
- **Idle Timer** : این زمان سنج برای آن است که اگر تلاش برای تکرار ارسال یک بسته بیش از حد متعارف انجام شود ارتباط TCP را بصورت یکطرفه رها و قطع نماید. مقدار معمول این زمان سنج ۳٦۰ ثانیه ( ٦ دقیقه ) است.

## ۷) پروتکل UDP

پروتکل TCP پروتکلی "اتصالگرا" است و لزوم برقراری یک ارتباط قبل از هرگونه مبادلهٔ داده ، می‌تواند بین چند میلی ثانیه ( برای شبکه های محلی سریع ) تا چندین ثانیه ( برای شبکه های WAN ) طول بکشد؛ در ضمن تامل برای بازگشت پیغامهای Ack ، یک پروسهٔ کاربردی را با تاخیر مواجه خواهد کرد. برای برخی از کاربردها این زمان قابل تحمل نیست و سرعت در رسیدن یک بسته به مقصد ، ضروریتر از پرداختن به مسائلی از قبیل بررسی شمارهٔ ترتیب و ارسال پیغامهای کنترلی محسوب میشود. ( کاربردهایی مثل سیستم DNS یا TFTP که در بخشهای آتی بررسی میشوند.)

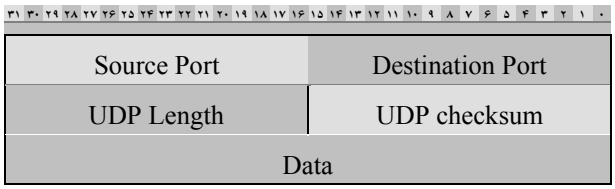
در لایهٔ انتقال از مدل TCP/IP برای چنین کاربردهایی یک پروتکل ساده و سریع به نام UDP معرفی شده است که به صورت ذاتی "بدون اتصال"[۱] است ، یعنی بدون هیچ اطلاعی از سرنوشتی که در انتظار یک بسته است ، به سمت مقصد ارسال میشود. هرگونه اطلاعی از رسیدن یا نرسیدن داده ها باید در لایه بالاتر بررسی و مدیریت شود.

پروتکل UDP ، تمام کاستی های لایهٔ IP را دارد ( به غیر از نظارت بر خطای کانال که میتواند وجود داشته باشد ) و تنها ارمغان این پروتکل برای پروسه ها سرعت ارسال و کم شدن تأخیرات ناشی از نظارت بر جریان بسته هاست.

---

[۱] Connectionless

در ادامه ساختار بسیار سادهٔ یک بستهٔ UDP را بررسی میکنیم. در شکل (٦-٥) ساختار یک بستهٔ UDP به تصویر کشیده شده است.




شکل (٦-٥) ساختار یک بستهٔ UDP

- **فیلد Source Port** : در این فیلد ، یک شمارهٔ ١٦ بیتی بعنوان آدرس پورت پروسهٔ مبدأ که این بسته را جهت ارسال ، تولید کرده ، قرار خواهد گرفت.
- **فیلد Destination Port** :در این فیلد ، آدرس پورتِ پروسهٔ مقصد که آنرا تحویل خواهد گرفت ، تعیین خواهد شد .

همانگونه که در بخش قبلی اشاره شد این دو آدرس مشخص می‌کنند که این بسته از کدام برنامهٔ کاربردی در لایه بالاتر تولید و باید به چه برنامه ای در ماشین مقصد تحویل داده شود.

- **فیلد UDP Length** : در این فیلد طول بستهٔ UDP برحسب بایت ، شامل سرآیند و داده ها ، درج میشود.
- **فیلد UDP Checksum** : در این فیلدِ ١٦ بیتی کد کشف خطا درج میشود. روش محاسبهٔ این کد دقیقاً همانند روشی است که در پروتکل TCP معرفی شد. تنها تفاوت در آنست که بکارگیری این فیلد اختیاری است و در صورت عدم نیاز به آن ، تمام بیتهای آن به صفر تنظیم می‌شود. ( برای کاربردهایی مثل ارسال دیجیتال صدا یا تصویر )

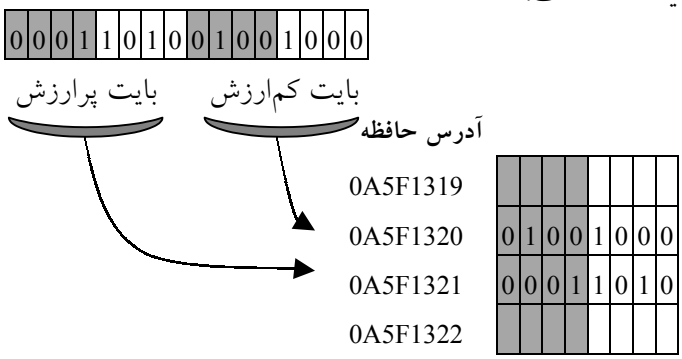
مناسبترین کاربرد پروتکل UDP برای پروسه هایی است که عملیاتشان مبتنی بر یک تقاضا و یک پاسخ است. (سیستم DNS)

نکات بکارگیری و ظرائف پروتکلهای TCP و UDP در فصل برنامه نویسی تحت شبکه مجدداً بررسی می‌شوند.

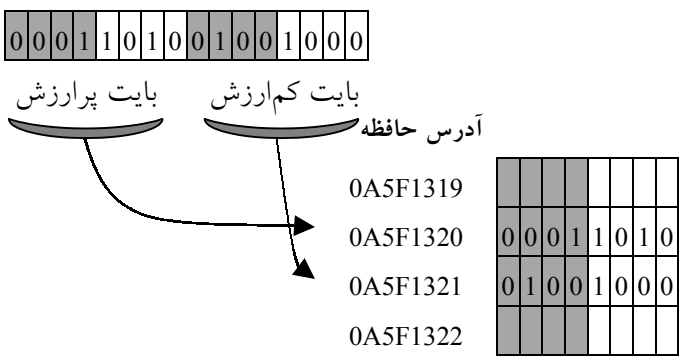
## ۸) ماشینهای Big Endian و Little Endian

ساختار بسته‌های IP و TCP در قالب کلمات ۳۲ بیتی سازماندهی می‌شوند. ماشینهای متفاوتی که در دنیا وجود دارد کلمات دوبایتی و چهاربایتی (۳۲ بیتی) را به یک نحو درون حافظه ذخیره نمی‌کنند . در برخی از ماشینها برای ذخیره یک کلمه در حافظه ابتدا بایت کم ارزش در حافظه ذخیره شده و پشت سر آن بایت پرارزش ذخیره می‌شود ، در حالی که در برخی دیگر دقیقاً برعکس است. بعنوان مثال فرض کنید کلمهٔ ۲ بایتی (0x1A48) با دستور Mov به حافظهٔ اصلی منتقل شود. ماشینها به دو روش آنرا در حافظهٔ خود ذخیره می‌کنند :

- ماشینهایی که ابتدا بایت کم ارزش و سپس بایت پرارزش را ذخیره می‌کنند ، ماشینهای Little Endian نامیده می‌شوند. کامپیوترهای شخصی با پردازنده سری 80X86 و پنتیوم از این دسته هستند. در مثال زیر طریقهٔ ذخیره‌سازی یک کلمهٔ ۱۶ بیتی در فضای حافظهٔ یک ماشین .L.E به تصویر کشیده شده است.




- ماشینهایی که ابتدا بایت پرارزش و سپس بایت کم‌ارزش را ذخیره می‌کنند ، ماشینهای Big Endian نامیده می‌شود. کامپیوترهای سری SUN از این دسته هستند. در مثال زیر طریقهٔ ذخیره‌سازی یک کلمهٔ ۱۶ بیتی در فضای حافظهٔ یک ماشین .B.E به تصویر کشیده شده است.



از آنجایی که بسته‌های IP ابتدا در حافظه تشکیل و سپس از طریق سخت‌افزار کارتِ شبکه ارسال می‌شوند لذا اگر بسته IP که بصورت سریال روی خط ارسال می‌شود از یک ماشین Little Endian تولید و توسط یک ماشین Big Endian دریافت شود ، ممکن است جای بایتها عوض شده و محتوی بسته‌ها اشتباه و فاقد ارزش دریافت شوند.

پروتکل TCP/IP ، استاندارد ماشینهای Big Endian را مبنا قرار داده است ، لذا در تمام ماشینهای Little Endian ، قبل از ارسال بسته‌های IP باید فیلدهای دوبایتی و چهاربایتی درون حافظه ، بگونه‌ای تنظیم و مقداردهی شوند تا در هنگام ارسال بسته روی خط ، ابتدا بایت پرارزش ارسال شده و استاندارد ماشینهای .B.E رعایت شود.

## ۹) شمارهٔ پورتهای استاندارد

در جدول (۷-۵) شمارهٔ پورتهای استاندارد که توسط IETF به عنوان استاندارد پذیرفته شده‌اند ، ارائه شده است.

| پورت | نام اختصاری | نام پروسه |
|---|---|---|
| 1 | TCPMUX | TCP Port Service Multiplexer |
| 5 | RJE | Remote Job Entry |
| 7 | ECHO | Echo |
| 9 | DISCARD | Discard |
| 11 | USERS | Active Users |
| 13 | DAYTIME | Daytime |
| 17 | Quote | Quote of the Day |
| 19 | CHARGEN | Character Generator |
| 20 | FTP-DATA | File Transfer (Data Channel) |
| 21 | FTP | File Transfer (Control Channel) |
| 23 | TELNET | TELNET |
| 25 | SMTP | Simple Mail Transfer |
| 27 | NSW-FE | NSW User System FE |
| 29 | MSG-ICP | MSG-ICP |
| 31 | MSG-AUTH | MSG Authentication |
| 33 | DSP | Display Support Protocol |
| 35 | PPS | Private Printer Server |
| 37 | TIME | Time |
| 39 | RLP | Resource Location Protocol |
| 41 | GRAPHICS | Graphics |
| 42 | NAMESERVER | Host Name Server |
| 43 | NICNAME | Who Is |
| 49 | LOGIN | Login Host Protocol |
| 53 | DOMAIN | Domain Name Server |

| | | |
|---|---|---|
| **67** | BOOTPS | Bootstrap Protocol Server |
| **68** | BOOTPC | Bootstrap Protocol Client |
| **69** | TFTP | Trivial File Transfer Protocol |
| **79** | FINGER | Finger |
| **101** | HOSTNAMENIC | Host Name Server |
| **102** | ISO-TSAP | ISO TSAP |
| **103** | X400 | X.400 |
| **104** | X400SND | X.400 SND |
| **105** | CSNET-NSCSNET | Mailbox Name Server |
| **109** | POP2 | Post Office Protocol v2 |
| **110** | POP3 | Post Office Protocol v3 |
| **111** | SUNRPC | SUN RPC Portmap |
| **137** | NETBIOS-NS | NETBIOS Name Service |
| **138** | NETBIOS-DGMNET | BIOS Datagram Service |
| **139** | NETBIOS-SSNNET | BIOS Session Service |
| **146** | ISO-TP0 | ISO TP0 |
| **147** | ISO-IP | ISO IP |
| **150** | SQL-NET | SQL-NET |
| **153** | SGMP | SGMP |
| **156** | SQLSRV | SQL Service |
| **160** | SGMP-TRAP5 | SGMP TRAPS |
| **161** | SNMP | SNMP |
| **162** | SNMPTRAP | SNMPTRAP |
| **163** | CMIP-MANAGE | CMIP/TCP Manager |
| **164** | CMIP-AGENT | CMIP/TCP Agent |
| **165** | XNS-COURIER | Xerox Network |
| **179** | BGP | Border Gateway Protocol |

(۷-۵) شمارهٔ پورتهای استاندارد

## ۱۰) مراجع این فصل

مجموعهٔ مراجع زیر می‌توانند برای دست آوردن جزییات دقیق و تحقیق جامع در مورد مفاهیم معرفی شده در این فصل مفید واقع شوند.


| | |
|---|---|
| RFC 1072 | "TCP Extensions for Long-Delay Paths," Jacobson, V.; Braden, R.T.; 1988 |
| RFC 896 | "Congestion Control in IP/TCP Internetworks," Nagle, J.; 1984 |
| RFC 879 | "TCP Maximum Segment Size and Related Topics," Postel, J.B.; 1983 |
| RFC 813 | "Window and Acknowledgment Strategy in TCP," Clark, D.D.; 1982 |
| RFC 793 | "Transmission Control Protocol," Postel, J.B.; 1981 |
| RFC 768 | "User Datagram Protocol," Postel, J.B.; 1980 |

## ۱) سرویس دهندهٔ نامهای حوزه[1]

در فصول گذشته به کرّات صحبت از آدرسهای IP شد و کلاسهای آدرس و الگوهای زیر شبکه ، مورد بررسی قرار گرفتند و لیکن آدرسهائی که در دنیای واقعی بعنوان آدرسهای اینترنت دیده می‌شود ، نمادی متفاوت و شبیه به مثالهای زیر دارد:

www.ibm.com

cs.ucsb.edu

آنچه مسلم است از آدرسهائی با قالب و ساختار بالا فقط بخاطر راحتی کاربرد و سادگی در بخاطر سپردن استفاده شده است و قطعاً در هنگام برقراری یک ارتباط و مبادلهٔ داده ، این آدرسهای نمادین بایستی به معادل عددی آن (یعنی عدد ۳۲ بیتی آدرس IP) ترجمه شود؛ به این آدرسهای نمادین " نَام حوزه" گفته می‌شود. بخاطر داشته باشید که آدرسهای نمادین فوق می‌توانند با مقدار آدرسIP آن تعویض شود؛ یعنی آدرسهای زیر نیز در دنیای اینترنت معتبرند ولی به دلیل سختی در به خاطر سپردن و استفاده از آنها ، به ندرت استفاده می‌شود:

alavi@203.41.3.81

http://131.11.70.14

حال سوال اینجاست که وقتی ماشینی تمایل دارد داده‌هایی را به ماشین دیگری با آدرس:

cs.rayan.ir

بفرستد چگونه می‌تواند قبل از ارسال پیام ، آدرس IP معادل با این آدرس نمادین را پیدا کند در حالی که دهها میلیون از این گونه آدرسها روی اینترنت تعریف شده اند.

به غیر از آدرسهای نمادین همانند مثالهای فوق ، "شبکه‌های خودمختار" میتوانند برای ماشینهای خود از روشهای نامگذاری اختصاصی استفاده کنند. در این گونه شبکه‌ها یک ماشین می‌تواند دارای دو نام متفاوت باشد ، مثل :

kiti.cs.ucsb.edu

hpss_kiti_510

آدرس نمادین اول در کلّ شبکهٔ اینترنت معتبر است ولی نام دوم فقط درون شبکه‌ای قابل شناسایی است که آن ماشین متعلق به آن است.

---

[1] Domain Name System

در سالهای نخستین راه اندازی شبکهٔ ARPANet راه حلی بسیار ساده برای ترجمهٔ نامهای نمادین به آدرس IP وجود داشت و آن تعریف تمام نامها و آدرسهای IP معادل ، در یک فایل بنام hosts.txt بود. در آن تاریخ تعداد ماشینهای میزبان زیاد نبود و حجم چنین فایلی ، چندان بزرگ نمی‌شد. هر ماشین میزبان ساعت 24.00 هر شب این فایل را تازه سازی و به‌روز می‌کرد تا هر گونه تغییر احتمالی و تعریف آدرسهای جدید اعمال شود. تابع مترجم نام روی هر ماشین میزبان برای ترجمهٔ یک نام نمادین مستقیماً به این فایل مراجعه می‌کرد و معادل IP آن را استخراج کرده و به برنامهٔ کاربردی برمی‌گرداند.

بدیهی است که داشتن یک فایل متمرکز و قرار دادن تمام آدرسها و معادل آدرسIP در آن ، امروزه با حجم میلیونی آدرسها در اینترنت امکان پذیر نیست و سالهاست از روشی برای تبدیل آدرسهای نمادین به آدرسهای IP استفاده می‌شود که DNS نام دارد. ( امروزه روش متمرکز فقط برای شبکه‌های داخلی و کوچک و برای سیستمهای نامگذاری خاص استفاده میشود )

DNS یا "سیستم نامگذاری حوزه" ، روشی سلسله مراتبی است که بانک اطلاعاتی مربوط به نامهای نمادین و معادل IP آنها را روی کل شبکهٔ اینترنت توزیع کرده است و هر ایستگاه می‌تواند در یک روال منظم و سلسله مراتبی آدرس IP معادل با ایستگاه مورد نظرش را در نقطه‌ای از شبکه پیدا کند؛ این سیستم در سال ۱۹۸٤ معرفی شد.

در DNS ، کلّ آدرسهای اینترنت درون بانکهای اطلاعاتی توزیع شده ای هستند که هیچ تمرکزی روی نقطه‌ای خاصّ از شبکه ندارند. روش ترجمهٔ نام بدین صورت است که وقتی یک برنامهٔ کاربردی مجبور است برای برقراری یک ارتباط ، معادل آدرس IP از یک ماشین با نامی مثل cs.ucsb.edu را بدست بیاورد ، قبل از هر کاری یک تابع کتابخانه‌ای[۱] را صدا می‌زند؛ به این تابع کتابخانه‌ای "تابع تحلیلگر نام"[۲] گفته می‌شود. تابع تحلیلگر نام ، یک آدرس نمادین را که بایستی ترجمه شود ، بعنوان پارامتر ورودی پذیرفته و سپس یک بستهٔ درخواست[۳] به روش UDP تولید کرده و به آدرس یک سرویس دهندهٔ DNS (که به صورت پیش فرض مشخص می‌باشد) ، ارسال می‌کند. همهٔ ماشینهای میزبان ، حداقل باید آدرس IP از یک سرویس دهندهٔ

[۱] Library Function
[۲] Name Resolver
[۳] Query Packet

DNS را در اختیار داشته باشند. این "سرویس دهندهٔ محلی"[۱] پس از جستجو، آدرس IP معادل با یک نام نمادین را بر می‌گرداند. "تابع تحلیلگر نام" نیز آن آدرس IP را به برنامهٔ کاربردی تحویل می‌دهد. با پیدا شدن آدرس IP ، برنامهٔ کاربردی می‌تواند عملیات مورد نظرش را ادامه بدهد.

حال باید روشهای جستجو را در سیستم DNS بررسی کنیم:

همانگونه که اشاره شد بانک اطلاعاتی که اسامی اینترنت را با معادل IP آنها در خود دارد متمرکز نیست بلکه روی کل اینترنت توزیع شده است. حال باید دید اسامی اینترنت چگونه سازماندهی می‌شود تا نهایتاً بتوان روش جستجو روی یک بانک اطلاعاتی توزیع شده را توضیح داد.

اسامی نمادین زیر را در نظر بگیرید:

www.president.ir

www.bristol.edu

khatami@president.ir

elvis@hware.cs.mit.edu

بدیهی است که نامهای حوزه همانند مثالهای بالا بدون مسمّی و دلیل انتخاب نمی‌شوند بلکه اطلاعاتی ارزشمند برای جستجو در بانک اطلاعاتی توزیع‌شدهٔ نامهای نمادین در خود دارند. بگونه ای که مشهود است یک نام حوزه از چند بخش مجزا که با علامت "." از هم تفکیک شده ، تشکیل می‌شود. هر کدام از این بخشها که "سطح" نام دارد به یک قسمت از بانک اطلاعاتی توزیع شده اشاره می‌نماید که به محدودتر شدن فضای جستجو کمک می‌کند.[۲]

برای تحلیل یک نام حوزه ، سطوح از سمت راست به چپ تفکیک می‌شوند و در یک روند سلسله مراتبی ، سرویس دهندهٔ متناظر با آن سطح پیدا می‌شود.

فعلاً از بالاترین سطح که در سمت راست نام حوزه قرار می‌گیرد (مثل com. ، edu. ، net. و... ) شروع می‌کنیم. نامهای حوزه به هفت منطقهٔ عمومی و حدود صد و اندی منطقهٔ کشوری تقسیم بندی شده است. حوزه بدین معناست که شما با یک

---

[۱] Local DNS Server

[۲] **نامهای حوزه شباهت ویژه‌ای به سیستم سلسله مراتبی ذخیره‌سازی فایلها روی یک ماشین دارد؛ به عنوان مثال اگر شما تمام فایلهایتان را فقط روی یک شاخه ذخیره می کردید جستجو و یافتن یک فایل از بین آنها ، کاری بسیار پر زحمت و وقتگیر می‌شد درحالی که تقسیم فضای دیسک به شاخه‌های متعدد و دسته‌بندی و ذخیرهٔ فایلها روی شاخه‌های متفاوت این مشکل را حل می‌کند.**

نگاه ساده به انتهای آدرس نمادین ، می‌توانید ماهیت آن نام و سرویس دهندهٔ متناظر با آن را حدس بزنید. یعنی اگر انتهای نامهای حوزه متفاوت باشد منطقهٔ جستجو برای یافتن آدرس IP معادل نیز متفاوت خواهد بود.

هفت حوزه عمومی که همه آنها سه حرفی هستند عبارتند از:

- **com.** صاحب این نام جزو موسسات اقتصادی و تجاری به شمار می‌آید.[۱]

www.sony.com

- **edu.** صاحب این نام جزو موسسات علمی یا دانشگاهی به شمار می‌آید.[۲]

www.sharif.edu

- **gov.** این مجموعه از نامها برای آژانسهای دولتی آمریکا اختصاص داده شده است.[۳]

www.whitehouse.gov

- **int.** صاحب این نام یکی از سازمانهای بین المللی (مثل یونسکو ، یونیسف ، فائو و ... ) محسوب می‌شود.[٤]

www.unicef.int

- **mil.** صاحب این نام یکی از سازمانهای نظامی دنیا به شمار می‌آید.[٥]
- **net.** صاحب نام جزو یکی از "ارائه‌دهندگان خدمات شبکه"[٦] به شمار می‌رود.

www.pegah.net

- **org.** صاحب نام جزو یکی سازمانهای عام‌المنفعه و غیرانتفاعی محسوب می‌شوند.[۷]

www.ieee.org

بنابراین اگر در انتهای نام یک آدرس یکی از حالتهای هفتگانه فوق را دیدید می‌توانید سریعاً به ماهیت آن آدرس و منطقهٔ جستجو پی ببرید.

نامهای حوزهٔ بسیار زیادی در اینترنت تعریف شده اند که هیچیک از حوزه‌های سه حرفی هفتگانه را در انتهای آنها نمی‌بینید. معمولاً در انتهای این آدرسها یک رشتهٔ

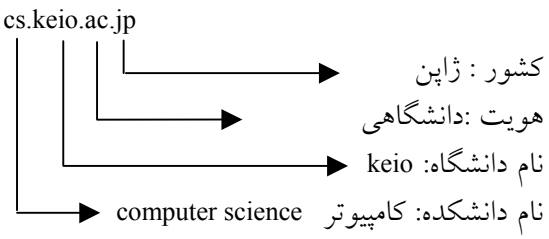
---

[۱] مخفّف commercial
[۲] مخفّف educational
[۳] مخفّف government
[٤] مخفّف international
[٥] مخفّف military
[٦] مخفّف Network Service Provider
[۷] مخفّف organization

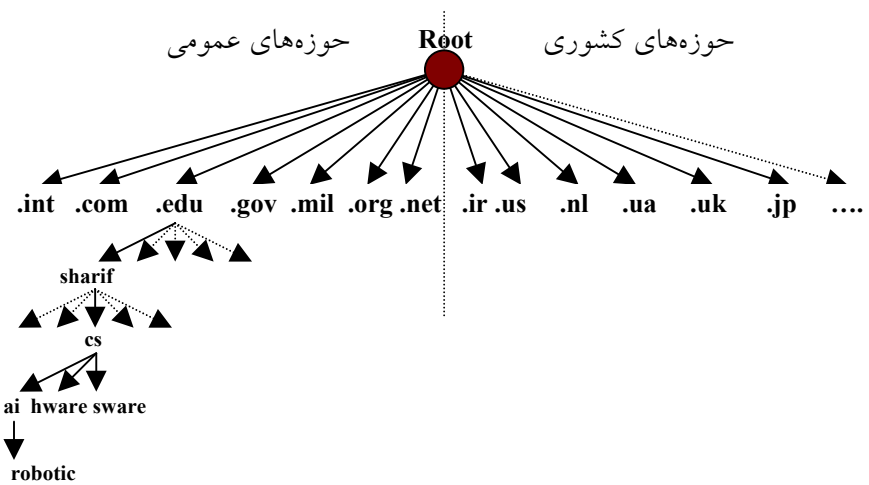
دوحرفی مثل **ir.** یا **nl.** قرار گرفته است. این رشتهٔ دو حرفی مخفّف نام کشوری است که آن آدرس و ماشین صاحب آن نام ، در آن کشور واقع است. مثل:

| | | | |
|---|---|---|---|
| ir. | ایران | ca. | کانادا |
| nl. | هلند | jp. | ژاپن |
| us. | ایالات متحده | us. | امارات متحده |

هر حوزه می‌تواند به زیر حوزه‌های کوچکتری تقسیم شود. بعنوان مثال نامهای مربوط به حوزه ژاپن با مخفف jp. به دو حوزه کوچکتر تقسیم می‌شود: ac.jp. و co.jp. که اولی یک موسّسهٔ علمی و دانشگاهی و دومی یک موسّسهٔ بازرگانی یا تجاری را در ژاپن تعیین می‌نماید؛ یعنی محل جستجو برای ترجمهٔ یک نام متفاوت خواهد بود. بعنوان مثال آدرس زیر بسادگی قابل تحلیل است:




برای سادگی در درک حوزه و زیرحوزه به نمودار زیر دقّت کنید:



در شکل قبل چگونگی شکسته شدن یک آدرس به زیرحوزه‌های کوچکتر به تصویر کشیده شده است.

با این ساختار برای ترجمهٔ یک نام حوزه مثل robotic.ai.cs.sharif.edu ، عملیات از فایلی به نام "ریشه" شروع می‌شود؛ سپس آدرس ماشینی که فایل راهنمای **edu.** در آنجا واقع شده بدست می‌آید. با مراجعه به چنین فایلی مجدداً آدرس ماشینی که فایل راهنمای **sharif.edu.** در آنجا قرار دارد به دست می‌آید؛ این روند تا رسیدن به آدرس IP معادل ادامه می‌یابد. این عملیات در چند مرحلهٔ محدود تکرار می‌شود و با توجه به آنکه از پروتکل UDP استفاده می‌شود ، تاخیر بحرانی نخواهد داشت.

دقّت کنید شما نمی‌توانید هر نام دلخواه را برای شرکت یا سازمان خودتان انتخاب نمائید بلکه برای اینکار باید نام مورد نظر را ثبت[۱] نمائید؛ درغیر این صورت چنین آدرسی در اینترنت هویت نخواهد داشت. برای ثبت آدرس باید به سایتهای ثبت‌کنندهٔ نام حوزه مراجعه کرده و تقاضای ثبت آدرس نمائید.[۲] این مؤسسات ضمن ثبت و درج نام در بانک اطلاعاتی یکی از حوزه‌های سطح بالا ، تضمین خواهند کرد که نام انتخابیتان در کل شبکهٔ اینترنت منحصر بفرد باشد.

## ۲) روشهای جستجو در سرویس دهنده‌های نام

همانگونه که اشاره شد اسامی نمادین در شبکهٔ اینترنت که خود در قالب حوزه‌ها و زیرحوزه‌ها سازماندهی شده اند در یک فایل متمرکز ذخیره نمی‌شوند بلکه روی کل شبکه اینترنت توزیع شده اند، به همین دلیل برای ترجمه یک نام به آدرس IP ممکن است چندین مرحله "پرس‌وجو" صورت بگیرد تا یک آدرس پیدا شود. طبیعی است که یک پرس‌وجو برای تبدیل یک نام حوزه همیشه موفقیت آمیز نباشد و ممکن است به پرس و جوهای بیشتری نیاز شود یا حتی ممکن است یک آدرس نمادین اشتباه باشد وهیچ معادل IP نداشته باشد.

سه روش برای پرس و جوی نام در سرویس‌دهنده‌های نام وجود دارد:

- پرس‌وجوی تکراری[۳]

---

[۱] Register

[۲] سایتهای ثبت نام حوزه ، متعددند ولی مشهورترین آنها www.internic.net می باشد که هزینه‌ای را نیز دریافت می‌کند.

[۳] Iterative Query

- پرس‌وجوی بازگشتی[۱]
- پرس‌وجوی معکوس[۲]

این سه روش را بررسی می‌کنیم:

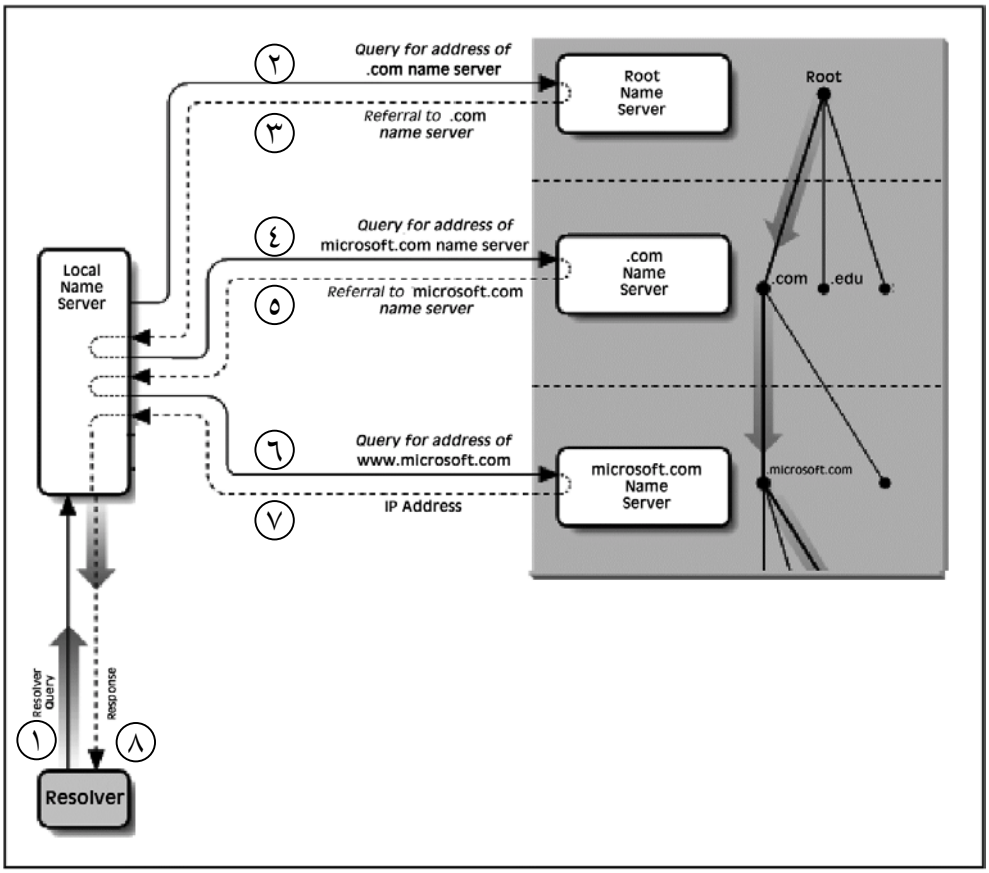
### ۱-۲) پرس و جوی تکراری

در پرس‌وجوهای تکراری قسمت اعظم تلاش برای تبدیل یک نام بر عهدهٔ سرویس‌دهندهٔ محلی است؛ این DNS حداقل به آدرس ماشین Root ، به عنوان نقطهٔ شروع نیاز دارد. وقتی یک تقاضای ترجمه آدرس به سرویس‌دهندهٔ محلی ارسال می‌شود در صورتی که قادر به ترجمهٔ نام به معادل IP آن باشد ، معادل آدرس IP نام مورد نظر را به تقاضا کننده برمی‌گرداند. ( این حالت وقتی است که سرویس‌دهندهٔ محلی قبلاً آن نام را ترجمه و در یک فایل ذخیره کرده باشد.) در غیر این صورت سرویس‌دهندهٔ محلی خودش یک تقاضا برای DNS سطح بالا ارسال می‌کند. این سرویس‌دهنده ، آدرسِ ماشینی را که می‌تواند برای ترجمهٔ نام مورد نظر مفید باشد ، به سرویس دهندهٔ محلی معرفی می‌کند؛ سرویس‌دهندهٔ محلی مجدداً یک تقاضا به ماشین معرفی شده در مرحلهٔ قبل ارسال میکند. در این حالت هم سرویس‌دهندهٔ نام می‌تواند در صورت یافتن آدرس IP معادل با آن نام حوزه ، آنرا ترجمه کند و یا آنکه آدرس سرویس‌دهندهٔ سطح پایینتری را به او برگرداند . این روند ادامه می‌یابد تا DNS نهائی نام مورد نظر را به آدرس IP ترجمه نماید. برای درک بهتر از روند کار به شکل (۱-٦) دقت کنید. در این مثال فرض شده است که یک برنامه کاربردی با فراخوانی "تابع تحلیلگر نام" ، تقاضای ترجمهٔ نام www.microsoft.com را می‌نماید. مراحلی که انجام میشود به شرح ذیل است:

- در مرحلهٔ اول برنامهٔ کاربردی با فراخوانی "تابع تحلیل نام" ، تقاضای ترجمه آدرس www.microsoft.com را برای سرویس‌دهندهٔ محلی ارسال کرده و منتظر می‌ماند.
- در مرحلهٔ دوم ، سرویس‌دهندهٔ محلی از سرویس دهندهٔ Root (که حوزه‌های متفاوت را تفکیک می‌کند) آدرسِ ماشین یک DNS که متولّی حوزه com. است را سؤال می‌کند.
- در مرحلهٔ سوم ، آدرس سرویس‌دهندهٔ مربوط به حوزهٔ com. برمی‌گردد.

---

[۱] Recursive Query
[۲] Reverse Query




شکل (۱-٦) روند ترجمهٔ نام به روش پرس‌وجوی تکراری

- درمرحلهٔ چهارم ، سرویس‌دهندهٔ محلی ، از ماشین معرفی شده در مرحلهٔ قبلی ، آدرس سرویس دهندهٔ مربوط به حوزه microsoft.com را سؤال می‌نماید.
- درمرحلهٔ پنجم فهرستی از سرویس دهنده‌های DNS مربوط به microsoft.com برمی‌گردد.
- در مرحلهٔ ششم ، سرویس‌دهندهٔ محلی تقاضای ترجمهٔ آدرس نمادین www.microsoft.com را از DNS متعلق به حوزه microsoft.com می‌کند.
- در مرحلهٔ هفتم ، معادل آدرس IP نام www.microsoft.com برمی‌گردد.
- در مرحلهٔ هشتم ، آدرس IP خواسته شده در اختیار برنامه کاربردی قرار می‌گیرد.

"تابع سیستمی تحلیل نام" در فصل برنامه‌نویسی تحت شبکه معرفی خواهد شد.

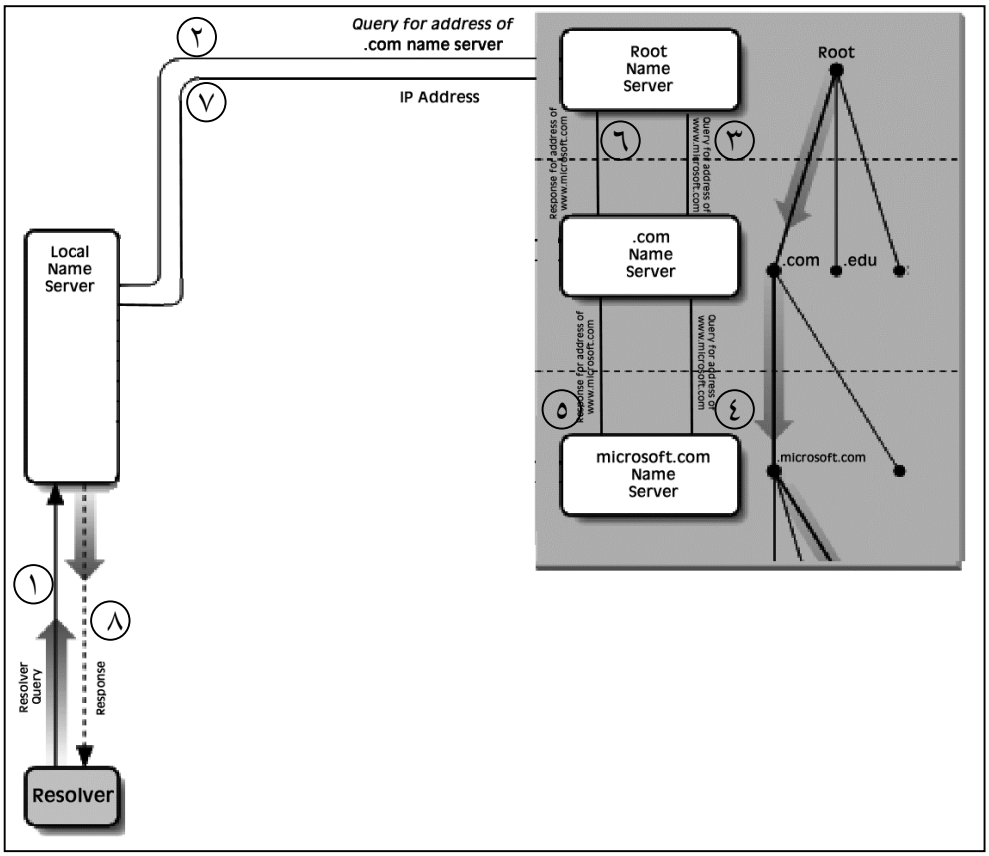
### ۲-۲) پرس و جوی بازگشتی

در این روش هر گاه برنامه‌ای بخواهد آدرس IP معادل یک نام مثل cs.yale.edu را بدست آورد بگونه ای که قبلاً اشاره شد ، "تابع سیستمی تحلیل نام" را فراخوانی می‌کند. این تابع یک ماشین را بعنوان سرویس‌دهندهٔ محلی از قبل می‌شناسد و بنابراین تقاضای تبدیل نام را به روش UDP برای آن ارسال کرده و منتظر جواب می‌ماند ( پاسخ نهایی DNS طبیعتاً باید یک آدرس ۳۲ بیتی معادل آدرس IP یک ماشین باشد ) دو حالت ممکن است اتفاق بیفتد:

- ممکن است در بانک اطلاعاتی مربوط به سرویس‌دهندهٔ محلی ، آدرس IP معادل با آن نام از قبل وجود داشته و بالطبع به سرعت مقدار معادل IP آن برمی‌گردد.
- ممکن است در بانک اطلاعاتی سرویس‌دهندهٔ محلی ، معادل IP آن نام وجود نداشته باشد. مثلاً سرویس‌دهندهٔ محلی در بانک اطلاعاتی خودش معادل آدرس IP نام cs.mit.edeu را نداشته و طبیعتاً نمی تواند آن را ترجمه کند. در چنین حالتی سرویس‌دهندهٔ محلی موظف است بدون آنکه به تقاضا دهنده خبر بدهد ، خودش رأساً به سرویس‌دهندهٔ سطح بالاتر تقاضای ترجمه آدرس بدهد. دراین حالت هم DNS سطح بالاتر بهمین نحو ترجمه آدرس را پیگری می‌کند یعنی اگر معادل IP آن نام را داشته باشد آنرا برمی‌گرداند و در غیر اینصورت خودش از سرویس‌دهندهٔ سطح پایینتر تقاضای ترجمهٔ آن نام را می‌نماید و این مراحل تکرار میشود. در روش پرس‌وجوی بازگشتی ماشین سرویس‌دهندهٔ محلی این مراحل متوالی را نمی‌بیند و هیچ کاری جز ارسال تقاضای ترجمهٔ یک آدرس برعهده ندارد و پس از ارسال تقاضا برای سرویس‌دهندهٔ سطح بالا منتظر خواهد ماند. باز هم تکرار می‌کنیم ، روشی که DNS برای ترجمه آدرس بکار می‌برد می‌تواند بدون اتصال ( UDP ) باشد که این کار به سرعت عمل ترجمهٔ آدرس می‌افزاید.

دقت کنید که در روش پرس‌وجوی تکراری نسبت به روش پرس‌وجوی بازگشتی ، حجم عمدهٔ عملیات برعهدهٔ سرویس‌دهندهٔ DNS محلی است و مدیریت خطاها و پیگیری روند کار ساده‌تر خواهد بود و روش منطقی‌تری برای بکارگیری در شبکهٔ اینترنت محسوب می‌شود. روش پرس‌وجوی بازگشتی برای شبکه‌های کوچک کاربرد دارد.

برای درک بیشتر این روش به شکل (۲-٦) دقّت کنید.




شکل (۲-٦) روند ترجمهٔ نام به روش پرس‌وجوی بازگشتی

### ۲-۳) پرس و جوهای معکوس

فرض کنید حالتی بوجود بیاید که یک سرویس‌دهندهٔ DNS ، آدرس IP یک ماشین را بداند ولی نام نمادین معادل با آن را نداند. بعنوان مثال DNS مایل است بداند که چه نامی در شبکهٔ اینترنت معادل با 195.13.42.7 می‌باشد. در چنین حالتی مسئله کمی حادّتر به نظر می‌رسد ، چرا که برای ترجمهٔ نامهای نمادین ، چون این نامها دارای حوزه و زیرحوزه هستند ، تحلیل آدرسها ساده است ولی ترجمهٔ آدرس IP به معادل نام حوزه ، از چنین روابطی تبعیت نمی‌کند؛ بعبارت بهتر هیچ ارتباط مستقیم و متناظری بین آدرسهای IP و اسامی انتخاب شده در اینترنت وجود ندارد.

برای یافتن نامهای متناظر با یک آدرس IP باید یک جستجوی کامل و در عین حال وقتگیر انجام بشود.

روش کاربدین صورت است که سرویس‌دهندهٔ محلی یک تقاضا برای DNS متناظر با شبکه‌ای که مشخصهٔ آن در آدرس IP ، مشخص شده ، ارسال می‌کند.[۱]

بعنوان مثال آدرس IP شبکه‌ای را 138.14.7.13 در نظر بگیرید ، آدرس کلاس B و مشخصهٔ آن 138.14.0.0 است. زمانی که موسّسه‌ای یک کلاس IP ثبت می‌دهد یک سرویس‌دهندهٔ DNS ، متناظر با شبکهٔ خود ایجاد کرده و آنرا نیز معرفی می‌کند. سرویس‌دهندهٔ محلی بایستی آدرس DNS متناظر با شبکهٔ 138.14.0.0 را پیدا کرده و سپس برای آن یک تقاضا ارسال کند. DNS مربوط به این شبکه ، براساس زیرشبکه‌هایی که دارد این سؤال را از طریق سرویس‌دهنده‌های متناظر با هر زیرشبکه پیگیری می‌کند.(چون هر زیر شبکه یک سرویس‌دهندهٔ DNS مخصوص به خود دارد) نهایتاً یک نامِ نمادین حوزه معادل با آن آدرس IP برخواهد گشت.

## ۳) ساختار بانک اطلاعاتی سرویس دهنده‌های نام

یک سرویس دهندهٔ نام در دو قسمت سازماندهی می‌شود:

- پروسهٔ سرویس‌دهنده : یک برنامهٔ اجرایی است که تقاضاهای ترجمهٔ نام از ماشینهای دیگر را گرفته و پردازش می‌کند و پاسخ مناسب را برای تقاضادهنده برمی‌گرداند. قالب هر تقاضا در شبکهٔ اینترنت ، استاندارد و مشخص است تا هر ماشینی فارغ از ساختار و سیستم عامل ، بتواند تقاضا بدهد و پاسخ دریافت کند. قالب استانداردِ تقاضا و پاسخ را در بخش بعدی بررسی خواهیم کرد.
- بانک اطلاعاتی: در این بانک اطلاعاتی داده‌های لازم برای تحلیل یک نام نمادین ، ذخیره می‌شود. هر سرویس دهنده می‌تواند بنابر روش مورد نظر خود ، این بانک اطلاعاتی را ایجاد کرده و از آن استفاده کند. بهمین دلیل ساختار این بانک اطلاعاتی در سرویس‌دهنده‌های گوناگون ، اندکی متفاوت است ولی تقریباً همهٔ آنها از اصول یکسانی پیروی می‌کنند. این بانک اطلاعاتی "بانک رکوردهای منبع"[۲] نام دارد که به اختصار "فایل RR " گفته میشود. برای بالا بردن سرعت جستجو در این بانکِ

---

[۱] یعنی فیلد NetID در آدرس IP

[۲] Resource Records

اطلاعاتی ، این فایل معمولاً در حافظهٔ اصلی نگهداری می‌شود. ( به همین دلیل برای ماشینهای سرویس‌دهندهٔ نام ، حافظهٔ زیاد و سریع پیشنهاد می‌شود.)

بایستی به این نکته دقت داشته باشید که وقتی یک سرویس‌دهندهٔ محلیِ نام ، از طریق پرس‌وجوهای تکراری ، یک آدرس نمادین را یکبار به آدرس IP ترجمه می‌کند نتیجه را در این فایل نگهداری می‌نماید تا آنکه در تقاضاهای بعدی بتواند از آن استفاده کند و مراحل وقتگیرِ پرس‌وجو تکرار نشود. البته این عمل اشکالی را خواهد داشت که باید برای آن تمهیدی اندیشیده شود:

اگر نام حوزه و معادل IP آن در یک فایل ، ثابت و همیشگی باشد ، تغییرات احتمالی و تعویض آدرسها چگونه در این فایل اعمال می‌شود؟ مثلاً هر گاه آدرس www.sbw.com و معادل IP آن 197.140.11.3 در فایلِ RR ذخیره شود، چه تضمینی وجود دارد که چند روز بعد معادل IP آن بدلیل خاصی به 197.140.12.16 تغییر نکند.

به همین دلیل هر رکورد درون فایل RR دارای زمان اعتبار است و پس از انقضای زمان باید از آن فایل حذف شده یا آنکه با پرس‌وجوی مجدّد به‌هنگام گردد. البته در فایل RR فقط نامهای نمادین و آدرسهای IP درج نمی‌شود بلکه آیتم‌های دیگری هم وجود دارد که به سرویس‌دهندهٔ نام برای حل بهینهٔ تبدیل آدرسها ، کمک می‌کند.

در حقیقت وقتی تقاضای تحلیل یک نام نمادین ، به یک سرویس‌دهندهٔ نام ارسال می‌شود ابتدا درون این فایل ، روی تک تک رکوردها جستجو انجام می‌شود و در صورت موفق بودن عملِ جستجو ، رکورد مربوطه به متقاضی بر خواهد گشت.

فایل RR یک فایل کاملاً متنی است (یعنی به راحتی و با یک ویرایشگر میتوان آن را مشاهده کرد یا تغییر داد.) هر رکورد درون این فایل معمولاً دارای پنج فیلد است:

| Domain Name | Time to live | Class | Type | Value |
|---|---|---|---|---|

به گونه‌ای که اشاره شد چون الزامی به استاندارد بودن این فایل وجود نداشته در برخی از سرویس دهنده‌های نام ، ساختار هر رکورد بصورت زیر سازماندهی شده است:

| Domain Name | Type | Class | Time to Live | Length | Value |
|---|---|---|---|---|---|

- **Domain Name** : دراین قسمت نام حوزه یا نام مربوط به یک ماشین (نام نمادین) قرار می‌گیرد. دقت کنید که چندین رکورد می‌تواند وجود داشته باشد که نام

نمادین آنها یکسان باشد ( چراکه ممکن است بقیه فیلدها متفاوت باشند ) بهمین دلیل این فیلد ، فیلدی "منحصر به فرد"[1] نیست.

- **Time to Live** : این گزینه نشان می‌دهد که رکورد تا چه مدت (بر حسب ثانیه) معتبر و قابل استناد است. با این گزینه مشکل عنوان شده در ابتدای این بخش حل می‌شود چراکه هر رکورد یک زمان اعتبار دارد که پس از منقضی شدن زمان ، باید از فایل RR حذف شده یا آنکه به‌هنگام شود. معمولاً در این فیلد مقدار ۸٦٤۰۰ قرار گیرد که معادل یک شبانه روز بر حسب ثانیه می‌باشد.

- **Class** : این فیلد مشخص می‌کند که ماهیّت نامِ نمادین مربوط به چه شبکه‌ای است ، چرا که هر سرویس‌دهندهٔ نام می‌تواند به غیر تعریف اسامی مبتنی بر شبکهٔ اینترنت ، روش نامگذاری خاص خود را در شبکهٔ محلی خود نیز اعمال کند.
اگر رکوردی مربوط به یک نام در شبکهٔ اینترنت باشد ، در این فیلد رشتهٔ دوحرفی IN قرار می‌گیرد؛ این مشخصه نشان میدهد که این رکورد ، در ارتباط با تعریف یک نام حوزه روی شبکه اینترنت است. دو نوع رکورد دیگر با کلاسهای CHAOS و Hesiod نیز تعریف شده‌اند که به آنها نخواهیم پرداخت زیرا به شبکهٔ اینترنت مربوط نمی‌شوند.

- **Type** : این فیلد نوع رکورد و معنای آن را مشخص می‌کند. مهمترین مقادیری که در این فیلد قرار می‌گیرد در جدول (۳-٦) فهرست شده است. به گونه‌ای که از جدول مشخص است در این فیلد میتواند یک گزینهٔ حرفی یا معادل عددی آن قرار بگیرد ولی برای سادگیِ ویرایش و تغییر ، از گزینهٔ حرفی استفاده میشود. مثال :

```
ns.nic.ddn.mil   99999999   IN   A   192.112.36.4
```

برای آنکه راحتتر بتوانیم این گزینه‌ها را تعریف کنیم ، به قسمتی از یک فایل RR در یک سرویس دهندهٔ نام که در مورد نام حوزهٔ cs.vu.nl رکوردهائی را ذخیره کرده است در جدول (٤-٦) دقت کنید.

لازم به تذکّر است هر خطی که در فایل RR با علامت ; شروع شود به عنوان خط توضیح تلقی شده و نادیده گرفته می‌شود. ( خطوط خالی و فواصل نیز مهم نیست )

---

[1] Unique

| *Number* | *Code* | *Description* |
|---|---|---|
| **1** | A | Network address |
| **2** | NS | Authoritative name server |
| **3** | MD | Mail destination; now replaced by MX |
| **4** | MF | Mail forwarder; now replaced by MX |
| **5** | CNAME | Canonical alias name |
| **6** | SOA | Start of zone authority |
| **7** | MB | Mailbox domain name |
| **8** | MG | Mailbox member |
| **9** | MR | Mail rename domain |
| **10** | NULL | Null resource record |
| **11** | WKS | Well-known service |
| **12** | PTR | Pointer to a domain name |
| **13** | HINFO | Host information |
| **14** | MINFO | Mailbox information |
| **15** | MX | Mail exchange |
| **16** | TXT | Text strings |
| **17** | RP | Responsible person |
| **18** | AFSDB | AFS-type services |
| **19** | X.25 | X.25 address |
| **20** | ISDN | ISDN address |
| **21** | RT | Route through |

جدول (۳-٦) انواع رکوردها در بانک اطلاعاتی سرویس‌دهندهٔ نام

```
;Authoritative  data for  cs.vu.nl
cs.vu.nl.  86400 IN SOA  star boss (952771,7200,2419200,86400)
cs.vu.nl.  86400 IN TXT  "Faculteit wiskunde en informatica"
cs.vu.nl.  86400 IN TXT  "Virje universiteit Amsteradam"
cs.vu.nl.  86400 IN MX    1 zephyr.cs.vu.nl.
cs.vu.nl.  86400 IN MX    2 top .cs.vu.nl.

flits.cs vu.nl.    86400  IN HINFO    SUN  UNIX
flits.cs vu.nl.    86400  IN  A        130.37.231.165
flits.cs vu.nl.    86400  IN  A        192.31.231.165
flits.cs vu.nl.    86400  IN  MX       1 flits.cs.vu.nl
flits.cs vu.nl.    86400  IN MX        2 zephyr .cs.vu.nl
flits.cs vu.nl.    86400  IN  MX       3 top.cs.vu.nl
www.cs.vu.nl.  86400  IN CNAME  star.cs.vu.nl
ftp.cs.vu.nl.     86400  IN CNAME  zephyr.cs. vy.nl

rowboat                IN    A        130.37.56.201
                       IN    MX       1 rowboat
                       IN    MX       2 zephyr

little-sister          IN    A        130.37.62.23
                       IN    HINFO  Mac MacOS

laserjet               IN       A        192.31.231.216
                       IN       HINFO  "HP LaserJet IIISi Proprietary"
```

جدول (٤-٦) قسمتی حقیقی ازفایل RR در یک سرویس دهندهٔ نام (در دانشگاه ویرجی هلند)

♦ **SOA** [۱]: یکسری اطلاعات ابتدائی پیرامون "ناحیهٔ آدرس نمادین"[۲]، یک شماره سریال، مدیر مسئول ( مسئول تعریف اسامی ) و مهلت اعتبار، ارائه می‌کند. در حقیقت این رکورد آغاز تمام رکوردهائی است که در مورد یک ناحیهٔ آدرس مثل cs.vu.nl تعریف می‌شود.

```
; named.hosts files;
;Start Of Authority RR
tpci.com. IN SOA merlin.tpci.com
                root.merlin.tpci.com ( 2     ; Serial number
                7200                 ;Refresh (2 hrs)
                3600                 ;Retry (1 hr)
                151200               ;Expire (1 week)
                86400 )              ;min TTL
```

در مثال فوق رکورد SOA، آغاز تعریف تمام رکوردهایی است که به نحوی در حوزهٔ tpci.com تعریف می‌شوند. بعنوان مثال، تمام اسامی که دو سطح آخر نام حوزهٔ آنها tpci.com است، باید در این "ناحیه" تعریف شوند:

```
artemis.tpci.com.  IN A 143.23.25.7
merlin.tpci.com.   IN A 143.23.25.9
pepper.tpci.com.   IN A 143.23.25.72
```

ماشینهای سرویس دهندهٔ نام و همچنین ماشین دریافت‌کنندهٔ نامه‌های الکترونیکی که با آدرسهایی بصورت *person*@tpci.com ارسال میشوند باید در این حوزه تعریف شوند.

♦ **A** [۳]: این نوع رکورد که مهمترین نوع بشمار می‌رود معادل آدرس IP نامی را که در فیلد اول آمده است، تعیین می‌کند. چون یک ماشین می‌تواند بیش از یک آدرس IP داشته باشد لذا به ازای یک آدرس نمادین که در فیلد Domain Name مشخص شده می‌تواند چندین رکورد وجود داشته باشد. ذکر این نکته ضروری است که هرگاه

---

[۱] Start of Authority
[۲] Zone
[۳] IP Address

جای فیلدی خالی بماند مقدار آن همان مقدارِ فیلد در سطر قبلی در نظر گرفته می‌شود.

artemis.tpci.com. 86400 IN A 143.23.25.7

♦ انواع **MD** و **MF** هر دو امروزه بلااستفاده شده اند و بجای آنها از نوع **MX** استفاده میشود. این دو گزینه برای آدرسهای پست الکترونیکی کاربرد دارند.

♦ **MX** [۱]: این نوع رکورد برای آن است که یک ماشین تمام نامه‌های الکترونیکی که با آدرسهای مختلف از ماشینهای متفاوتی می‌رسند را دریافت کند. بعنوان مثال فرض کنید که شخصی دارای یک آدرس پست الکترونیکی با نامِ حوزه cs.vu.nl دارد ولی پس از مدتی به جای دیگری منتقل می‌شود که نام حوزه آن electrobrain.com است ولی تمایل دارد تمام نامه‌های او با نام حوزهٔ electrobrain.com به همان آدرس قبلی او ارسال شود. او از مسئول حوزهٔ com. خواهد خواست که این کار را برای او با ثبت رکورد زیر در فایل RR انجام بدهد:

electrobrain.com 86400 IN MX mailserver.cs.vu.nl

در حقیقت با این رکورد پس از تعریف و ثبت یک آدرس همانند electrobrain.com تمام نامه‌هایی را که با آدرسهایی نظیر *person*@electrobrain.com ارسال می‌شوند ، به ماشین دیگری ارجاع داده می‌شود؛ چراکه صاحب این نام (مثلاً) نخواسته است سیستم سرویس دهندهٔ پست الکترونیکی برای خودش ایجاد نماید و آنرا به ماشینی دیگر ارجاع داده است.

♦ **NS** [۲]: رکوردهای از این نوع ، یک ماشین سرویس دهندهٔ نام ، ویژه یک حوزه را معرفی می‌کند. ( برای تکرار جستجو توسط تقاضادهنده) احتمالاً کاربرد حیاتی این رکورد را احساس کرده‌اید چراکه در یک روند پرس‌وجوی تکراری ، وقتی از یک ماشین سرویس دهندهٔ نام ، تقاضای تحلیل یک نام حوزه می‌شود ، یا معادل IP آن نام را بر می‌گرداند یا آنکه آدرس یک ماشین دیگر که می‌تواند آن نام را تحلیل کند برگشت داده می‌شود. مثلاً در فایل RR از یک سرویس دهندهٔ حوزهٔ com. ، وجود رکوردهایی مثل رکورد زیر کاملاً ضروری است:

---

[۱] Mail Exchange
[۲] Name Server

yahoo.com 86400 IN NS ns.yahoo.com

ns.yahoo.com 86400 IN A 143.231.221.22

این دو رکورد بدین معناست که تمام نامهایی که به yahoo.com ختم می‌شوند در ماشینی با آدرس 143.231.221.22 قابل تحلیل هستند. این رکوردها مبنای یک پرس‌وجوی تکراری هستند.

به گونه‌ای که اشاره شد هر سرویس دهنده برای آنکه قادر به شروع عملیاتِ پرس‌وجوی تکراری باشد ، باید حداقل آدرس سرویس دهنده‌های ROOT را داشته باشد. بنابراین وجود رکوردهایی مثل رکوردهای زیر در یک فایل RR الزامی است:

```
; servers for the root domain;
.     99999999  IN  NS  ns.nic.ddn.mil.
      99999999  IN  NS  ns.nasa.gov.
      99999999  IN  NS  ns.internic.net
; servers by address
;
ns.nic.ddn.mil  99999999 IN A 192.112.36.4
ns.nasa.gov     99999999 IN A 192.52.195.10
ns.internic.net 99999999 IN A 198.41.0.4
```

( دقت کنید در سرویس‌دهندهٔ نام ، حوزه‌ای که با . مشخص میشود به معنای بالاترین حوزه نام یا همان ریشه تلقی میشود.[1])

♦ **CNAME** [2]: نامهای مستعار و راحتتر را برای یک آدرس تعیین می‌کند. بعنوان مثال شاید کسی حدسی بزند آدرس شبکهٔ دانشکدهٔ کامپیوتر در دانشگاه MIT بصورت cs.mit.edu است ولی نمیداند آدرس اصلی آن به صورت lcs.mit.edu می‌باشد ، که کمتر قابل حدس زدن و بخاطر سپردن است؛ در چنین حالتی با رکورد زیر می‌توان نام cs.mit.edu را بعنوان نامی مشابه با نام اصلی تعریف کرد:

cs.mit.edu 86400 IN CNAME lcs.mit.edu

---

[1] Top Level Domain
[2] Canonical Name

از طرفی اسامی نمادین گاهی طولانی هستند ( مثل www.altavista.com) و صاحبان این نامها ترجیح می‌دهند برای از دست ندادن مراجعین کم حوصله ، اسامی کوتاه معادل با نام اصلی داشته باشند. (مثل www.av.com) در چنین حالاتی این نوع از رکورد بکار می‌آید:

www.av.com 86400 IN CNAME www.altavista.com

♦ **PTR**[1] : این نوع رکورد هم شبیه قبلی است با این تفاوت که بجای قرار دادن آدرسِ نمادین معادل با www.av.edu در مثال فوق دقیقاً آدرس IP آنرا قرار می‌دهد:

www.av.com 86400 IN A 203.145.11.121

www.av.com 86400 IN PTR 203.145.11.121

در مجموع استفاده از گزینهٔ CNAME بهتر است چراکه اگر آدرس IP معادل با یک نام عوض شد فقط یک رکورد باید اصلاح شود در حالی که اگر از گزینهٔ PTR استفاده شود تمام رکوردها باید اصلاح شوند.

♦ **HINFO**[2] : رکوردهائی از این دسته ، ارزش عملیاتی برای تعیین و تبدیل آدرس ندارند بلکه سیستم عامل و نوع ماشین متناظر با یک آدرس یا نام را تعریف می‌کند:

flits.cs.vu.nl 86400 IN HINFO SUN UNIX

یعنی ماشینی با نام flits.cs.vu.nl ، دارای سیستم عامل یونیکس و سخت‌افزار آن سازگار با مینی کامپیوترهای نوع Sun است .

♦ **MINFO** : همانند گزینهٔ قبلی با این نوع رکورد می‌توان اطلاعاتی در مورد سیستم پست الکترونیکیِ متناظر با یک حوزه ارائه کرد.

♦ **MG** : با این گزینه مدیر مسئول شبکه می‌تواند افراد حقیقی را به عنوان دارندهٔ صندوق پستی در یک حوزه تعریف نماید. به عنوان مثال اگر شما یک آدرس پست الکترونیکی با نام ali@refah.com داشته باشید ، طبیعتاً باید نام ali در فایل RR از سرویس‌دهندهٔ refah.com تعریف شده باشد.

---

[1] Pointer
[2] Host Information

♦ **TXT** : این نوع رکورد ارزش عملیاتی ندارد و توضیحاتی را در مورد صاحب این نام و هویت آن ارائه میکند. تفاوتی این رکورد با خطوط توضیح (که با **;** شروع می‌شوند ) آنست که این رکوردها می‌توانند در اختیار متقاضیِ آن قرار بگیرد.

cs.mit.edu. 86400 IN TXT "Faculty of computer science "

cs.mit.edu. 86400 IN TXT "Massachusetts Institute of Technology"

♦ **WKS** [1]: با استفاده از این نوع رکورد ، گذشته از تعریف یک ماشین و آدرس IP آن ، سرویسهای معروفی را که آن ماشین ارائه می‌دهد ، معرفی می‌شود.

tpci_sco.tpci.com IN WKS 143.23.1.34. FTP TCP SMTP TELNET

♦ **NULL** : استفاده از این گزینه رکورد را غیرعملیاتی و بی‌ارزش مینماید و برای زمانی کاربرد دارد که ترجیح داده شود بجای حذف ، رکوردی تبدیل به یک رکورد بی ارزش شود.

### ۱-۳) رکوردهای پرس‌وجوی معکوس

بگونه‌ای که اشاره شد هرگاه یک سرویس‌دهنده ، آدرس IP یک ماشین را بداند ولی نام نمادین معادل با آن را نداند به پرس‌وجوی معکوس متوسل می‌شود. برای آنکه یک سرویس‌دهندهٔ نام ، بتواند نام نمادینِ معادل با یک آدرس IP را برگرداند باید در رکوردهای موجود در فایل RR به دنبال آن آدرسِ بگردد. رکوردهائی که در پرس‌وجوی معکوس جستجو می‌شوند باید در اولین فیلد ، آدرس IP یک ماشین یا یک شبکه را تعریف کنند ( البته در ادامهٔ آن باید رشتهٔ IN-ADDR-ARPA درج شده باشد). مثال زیر گویای کل کاری است که این رکوردها برای پرس‌وجوی معکوس انجام می‌دهند:

23.1.45.143.IN-ADDR-ARPA. IN PTR TPCI_HPWS_4.TPCI.COM

1.23.64.147.IN-ADDR-ARPA. IN PTR TPCI_SERVER.MERLIN.COM

3.12.6.123.IN-ADDR-ARPA. IN PTR BEAST.BEAST.COM

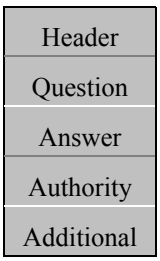
23.143.IN-ADDR-ARPA. IN PTR MERLINGATEWAY.MERLIN.COM

[1] Well-Known Services

## ۴) قالب پیام در سرویس‌دهنده‌های نام

در بخش قبلی اشاره شد که اگرچه ساختار فایلهای RR می‌تواند در سرویس‌دهنده‌های متفاوت اندکی متفاوت باشد ولی قالب "پیامهای پرس‌وجو"[۱] که بین سرویس‌دهنده‌ها مبادله می‌شوند باید استاندارد باشد تا تمام این سیستمها بتوانند آنها را "تجزیه وتحلیل"[۲] کرده و اطلاعات لازم را از آن استخراج کنند.

ساختار پیامهایی که بین سرویس‌دهنده‌های نام مبادله میشود در شکل (۵-٦) به تصویر کشیده شده است.




شکل (۵-٦) ساختار پیامهای سرویس‌دهندهٔ نام

همانگونه که از شکل مشخص می‌شود هر پیام دارای پنج بخش زیر است که آنها را به ترتیب در بخشهای بعدی توضیح خواهیم داد:

- بخش سرآیندِ پیام
- بخش پرسش
- بخش پاسخ
- بخش اطلاعات ناحیه
- بخش اطلاعات اضافی

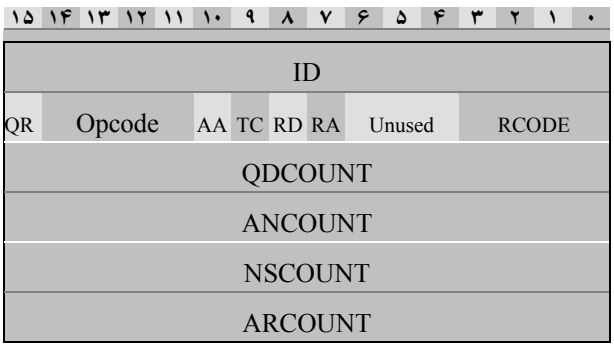
هر یک از بخشهای پنج‌گانهٔ فوق دارای چندین زیربخش است که باید عملکرد آنها بررسی شود:

---

۱ Query Message
۲ Parse

### ۱-۴) فیلدهای بخش سرآیند

در شکل (٦-٦) فیلدهای متفاوتِ بخش سرآیند از یک پیام نشان داده شده است. عملکرد و معنای هر فیلد به شرح زیر است:




شکل (٦-٦) ساختار "بخش سرآیند" از پیامهای سرویس‌دهندهٔ نام

- فیلد **ID** : وقتی یک سرویس‌دهنده پرسشی را در قالب یک پیام برای سرویس‌دهندهٔ دیگر ارسال میکند یک شمارهٔ منحصر به فرد برای آن برمیگزیند تا بتواند از بین پاسخهای رسیده پاسخ متناظر با هر پرسش را تشخیص بدهد. (تعداد پیامهای پرسش و پاسخ زیاد است)

- فیلد **QR** : این فیلد تک‌بیتی مشخص میکند که پیام از نوع پرسشی ( 0 ) است یا از نوع پاسخ ( 1 ) می‌باشد.

- فیلد **Opcode** : این فیلد چهاربیتی نوع پیام را تعریف می‌کند:
  - 0000 : قالب پیام از نوع پرس‌وجوی معمولی است.
  - 0001 : قالب پیام از نوع پرس‌وجوی معکوس است.
  - 0010 : قالب پیام از نوع پرس‌وجو در مورد وضعیت سرویس‌دهنده است.
  - 0011 تا 1111 : فعلاً بدون استفاده هستند.

- فیلد **AA** : این فیلد تک‌بیتی که فقط برای پیامهای پاسخ معنا دارد مشخص میکند که محتوای پاسخ توسط سرویس‌دهندهٔ مسئول و تعریف کنندهٔ یک نام صادر شده است ( 0 ) یا

آنکه محتوای پاسخ معرفی یک سرویس‌دهندهٔ دیگر است که میتواند در ترجمه نام کمک نماید.

- فیلد **TC** : ۱ بودن این فیلد تک‌بیتی مشخص میکند که محتوای پاسخ به دلیل زیاد بودن تعداد رکوردهای ارسالی ، کامل نیست و تعدادی از رکوردهای پاسخ حذف شده‌اند.

- فیلد **RD** : این فیلد تک‌بیتی که فقط برای پیامهای پرسش معنا دارد از یک سرویس‌دهندهٔ نام تقاضا میکند که در صورت امکان برای ترجمهٔ یک نام از روش "پرس‌وجوی بازگشتی" استفاده کرده و نتیجهٔ نهایی را برگرداند. (روش پرس‌وجوی بازگشتی رادر ابتدای این فصل معرفی کردیم)

- فیلد **RA** : این فیلد تک‌بیتی که فقط برای پیامهای پاسخ معنا دارد به یک سرویس‌دهندهٔ نام اعلام می‌کند که آیا قادر است پرس‌وجوی بازگشتی انجام بدهد ( 1 ) یا خیر ( 0 ) .

- بیتهای **Unused** : این سه بیت فعلاً هیچ کاربردی نداشته و در تمام پیامها باید صفر شوند.

- فیلد **RCODE** : این فیلد چهاربیتی نیز در پیامهای نوع پاسخ کاربرد دارد و نتیجهٔ رسیدگی به یک پیام پرسش را تعیین می‌کند:
  - 0000 : مشکلی وجود ندارد و پاسخ لازم ضمیمهٔ پیام است.
  - 0001 : سرویس دهنده موفق نشده است تا نوع تقاضا را تشخیص بدهد.
  - 0010 : سرویس دهنده فعلاً مشکلی داخلی دارد.
  - 0011 : نامِ تقاضا شده وجود ندارد.
  - 0100 : سرویس‌دهنده از پاسخ به تقاضای ارسالی ناتوان است و آنرا نمی‌پذیرد.
  - 0101 : سرویس‌دهنده از پاسخ به تقاضای ارسالی به دلیل مسائل امنیتی معذور است.
  - 0110 تا 1111 : تعریف نشده و بلااستفاده هستند.

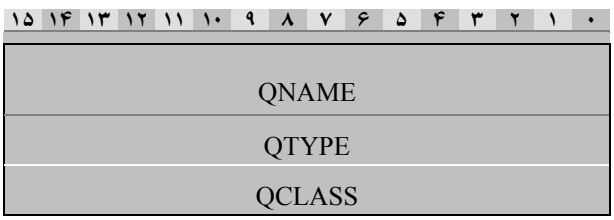
- فیلد **QDCOUNT** : این فیلد ۱۶ بیتی مشخص می‌کند در بخشِ "تقاضا" از قالبِ پیام ، چه تعداد رکورد پرسش وجود دارد. ( به قالب یک پیام در بخش قبلی دقت کنید )

- فیلد **ANCOUNT** : این فیلد ۱۶ بیتی مشخص می‌کند در بخش "پرسش" از قالب پیام ، چه تعداد رکورد RR ارسال شده است.
- فیلد **NSCOUNT** : این فیلد ۱۶ بیتی مشخص می‌کند در بخش "اطلاعات ناحیه" از قالب پیام ، چه تعداد رکورد ضمیمه شده است. ( به قالب یک پیام در بخش قبلی دقت کنید )

- فیلد **ARCOUNT** : این فیلد ۱۶ بیتی مشخص می‌کند در بخش "اطلاعات اضافی" چه تعداد رکورد ضمیمه شده است.

### ۴-۲) فیلدهای بخش پرسش

در شکل (۷-۶) فیلدهای متفاوت در بخش پرسش از قالب یک پیام نشان داده شده است. عملکرد و معنای هر فیلد به شرح زیر است:




شکل (۷-۶) ساختار "بخش پرسش" از پیامهای سرویس‌دهندهٔ نام

- فیلد **QNAME** : در این فیلد نامی که باید به آدرس IP ترجمه شود و یا اطلاعاتی در مورد آن به‌دست آید قرار می‌گیرد. این نام در قالب کاراکترهای متوالی ۸ بیتی ذخیره شده و انتهای آن با کد **0**\ مشخص می‌شود.

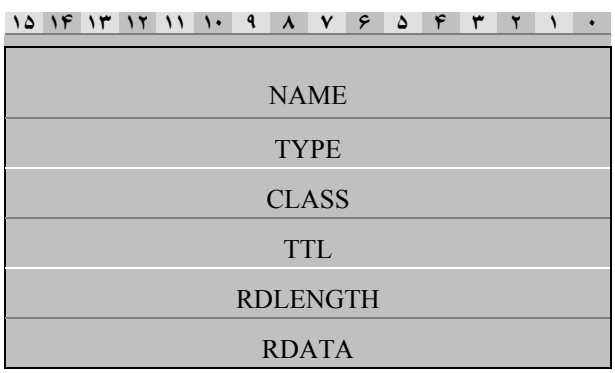
- فیلد **QTYPE** : در این فیلد نوع رکورد درخواستی مشخص می‌شود. ( مثلاً نوع SOA ، HINFO و ... ) تمام انواع رکوردها که در جدول (۳-۶) معرفی شده‌اند میتواند تقاضا داده شود؛ فقط دقت کنید که در این فیلد ۱۶ بیتی کد مربوط به نوع رکورد قرار می‌گیرد نه نام نمادین آن. ( مثلاً ۱ برای A ، ۵ برای CNAME ، ۱۳ برای HINFO ) نکتهٔ مهم در این فیلد آنست که اگر مقدار آن ۲۵۵ باشد پرسش‌کننده از سرویس‌دهنده تقاضا می‌کند که تمام

رکوردهای موجود در فایل RR که به نحوی با نـام حـوزهٔ مشـخص شـده در فیلـد QNAME مربوط است ، در پاسخ ارسال شود.

- فیلد **QCLASS** : در این فیلد ، کلاس آدرس درخواستی مشخص می‌شود و به گونه‌ای کـه اشاره شد برای آدرسهای شبکهٔ اینترنت از نوع IN است. ( به بخش قبلی مراجعه کنید )

### ۳-۴) فیلدهای تعریف شده در بخشهای پاسخ ، اطلاعات نامیه و بخش اطلاعات اضافی

وقتی یک پیامِ پرسش به یک سرویس‌دهندهٔ نام ارسـال می‌شـود ، آن سـرویس‌دهنده درون فایل RR به جستجوی رکوردهای متناظر با نامِ حوزهٔ درخواستی می‌پـردازد و در صورتـی کـه رکورد یا رکوردهایی یافت شدند باید تمام آنها برای سـؤال‌کننده ارسـال شـود. ایـن رکوردهـا بطور متوالی در یک پیام جاسازی می‌شوند. بار دیگر در شکل (۵-٦) به سـاختار پیامـهایی کـه بین سرویس‌دهنده‌های نام مبادله میشود دقت کنید. سه بخش آخر که فقـط در پیامـهای پاسـخ وجود دارند دارای قالب یکسانی هستند و محل قرارگرفتن رکوردهای پیدا شـده در خصـوص یک نامِ حوزه محسوب می‌شود. ساختار این بخشها در شکل (۸-٦) مشخص شده‌اند.




شکل (۸-٦) ساختار سه بخش آخر از پیامهای سرویس‌دهندهٔ نام

- فیلد **NAME** : در این فیلد نام حوزه قرار میگیرد. این نام در قـالب کاراکترهـای متوالـی ۸ بیتی ذخیره شده و آخر نام با کد **\0** مشخص می‌شود.

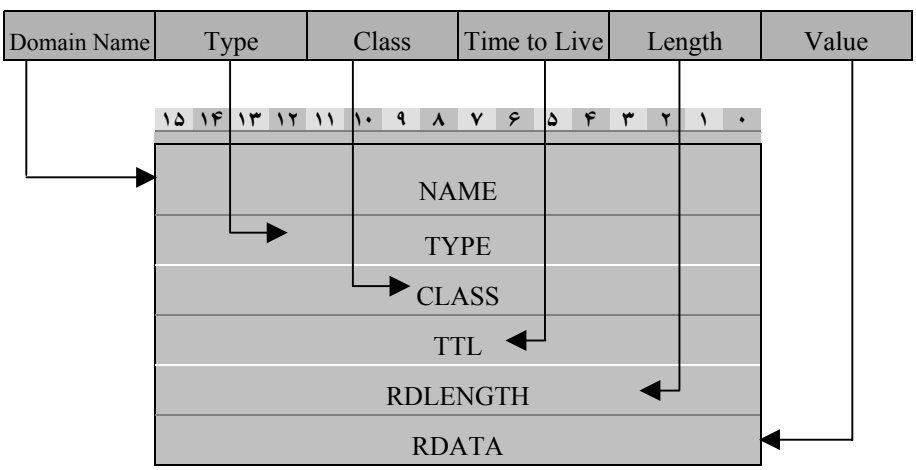
- فیلد **TYPE** : در این فیلد نوع رکورد مشخص می‌شود. انواع رکوردها که در جدول (۳-٦) معرفی شده‌اند ؛ فقط دقت کنید که در این فیلد ١٦ بیتی کد مربوط به نوع رکورد قرار می‌گیرد نه نام نمادین آن. ( مثلاً ۱ برای A ، ۵ برای CNAME ، ۱۳ برای HINFO )

- فیلد **CLASS** : در این فیلد کلاسِ نام حوزه مشخص میشود و به گونه‌ای که اشاره شد برای آدرسهای شبکهٔ اینترنت از نوع IN است.

- فیلد **RDLENGTH** : در این فیلد طول داده‌های موجود در فیلد RDATA بر حسب تعدادِ کاراکتر مشخص می‌شود.

- فیلد **RDATA** : در این فیلد رشتهٔ کاراکتری متناظر با نام حوزه قرار میگیرد.

برای درک ساده‌تر از فیلدهای فوق مجدداً ساختار رکوردهای فایل RR را در نظر بگیرید. وقتی رکوردی قرار است در پاسخ به یک پرسش ارسال شود طبق شکل (۹-٦) در "بخش پاسخ" از یک پیام جاسازی می‌شود.




شکل (۹-٦) جاسازی یک رکورد در یک پیامِ ارسالی از سرویس‌دهندهٔ نام

## ۵) مقدمه‌ای بر مدیریت شبکه

در اوایل پیدایش آرپانت در صورت بروز هرگونه مشکل در شبکه ، به علت کم بودن تعداد مسیریابها و ایستگاهها ، امکان بررسی همه آنها وجود داشت و با انجام برخی آزمایشهای فیزیکی توسط یک فرد خبره مشکل رفع می‌شد. ولی زمانی که آرپانت به اینترنت جهانی توسعه یافت به علت گستردگی آن ، این روش اشکالزدایی امکانپذیر نبود و راهکار بهتری را برای مدیریت شبکه اقتضا می‌کرد. در شبکه‌های قدیمی پروتکلهای مدیریت و نظارت بر شبکه بخشی از لایه پیوند داده بودند. مدیر شبکه می‌توانست برای مشخص شدن مشکل ، مسیریابها را مورد سئوال قرار دهد[۱] و یا مسیرها را آزمایش[۲] کرده و یا تغییر بدهد و همچنین هر یک از واسطهای ارتباطی شبکه[۳] را بصورت فیزیکی بررسی نماید و در صورتِ رفعِ مشکل ، به مسیریابها دستور ادامه عملیات را بدهد. برخلاف شبکه‌های محلی که از سخت‌افزار همگون تشکیل می‌شوند ، شبکه اینترنت مبتنی بر TCP/IP دارای یک پروتکل پیوند دادهٔ واحد نمی‌باشد و شامل شبکه‌های ناهمگونی است که از طریق "دروازه‌ها"[۴] به هم ارتباط یافته‌اند ، بنابراین مدیریت یک شبکهٔ محلی با مدیریت شبکهٔ گسترده‌ای همانند اینترنت کاملاً متفاوت خواهد بود.

نظارت بر وضعیت شبکه و اجزای آن و همچنین توانایی اعمال مدیریت بر روی ماشینهای میزبان و اجزای یک زیرشبکه (شامل مسیریابها ، پلها و ... ) از ملزومات شبکهٔ اینترنت محسوب می‌شود. در این راستا پروتکلهای مدیریت شبکه بوجود آمده‌اند. بدلیل ناهمگونی سخت‌افزار ارتباطی شبکه‌ها ، لاجرم نرم‌افزارهای مدیریت شبکه باید در لایه کاربرد پیاده‌سازی شوند. پیاده‌سازی نرم افزار مدیریت در لایه کاربرد باعث می‌شود پروتکل‌های مدیریت ، مستقل از سخت‌افزار شبکه طراحی گردند که در اینصورت مدیر شبکه می‌تواند با انواع مسیریابها و ماشینهای میزبان به یک روش مشابه ارتباط برقرار نماید و پروتکل مدیریت شبکه برای تمام اجزای آن یکسان و واحد باشد. از معایب پیاده‌سازی نرم افزار مدیریت در لایه کاربرد ، آن است که قابلیت اطمینان آن پائین می‌آید چراکه به هنگام بروز هرگونه مشکلِ پشتهٔ

---

[۱] توسط برنامه‌های کمکی همانند ping
[۲] توسط برنامه‌هایی مثل tracert (مخفف trace route )
[۳] کارت شبکه یا NIC ( مخفف Network Interface Card )
[۴] در گذشته به مسیریابهایی که دو شبکه با پروتکلهای متفاوت را به هم متصل می‌کردند دروازه ( Gateway ) گفته می‌شد و این اصطلاح هنوز هم بجای مسیریاب استفاده می‌شود.

TCP/IP در یکی از عناصر شبکه ، قبل از آنکه بتوان راهی را برای کشف و رفع عیب آن ارائه کرد ، نرم‌افزار مدیریت را از کار خواهد انداخت.

در ادامه به معرفی مختصر یکی از پروتکلهای مدیریت شبکه (SNMP[۱]) که در اینترنت مورد استفاده قرار می‌گیرد ، خواهیم پرداخت.

### ۱-۵) معماری پروتکلهای مدیریت شبکه

بطور کلی مسئله مدیریتِ شبکه به دو بخش عمده تقسیم شده و برای هر بخش ، استاندارد مخصوص به آن تعریف شده است:

- بخش اول ، تعریفِ استانداردِ مبادله اطلاعات لازم برای نظارت و مدیریت شبکه بین ماشینها و مدیر شبکه می‌باشد. بدین معنا که یک نرم افزار مشتری که در یک ماشین میزبان قرار دارد چگونه با ماشینی که نقش مدیر شبکه را بر عهده دارد ، ارتباط برقرار نماید ، از آن اطلاعات دریافت کند و یا باعث تغییر اطلاعات آن شود.

- بخش دوم ، شامل تعریف استاندارد نظارت[۲] و کنترل و همچنین تعریف اطلاعات مدیریتی[۳] می‌باشد. در این بخش از استاندارد معین می‌شود که چه اطلاعاتی برای نظارت و مدیریت شبکه لازم است و این اطلاعات با چه اسامی و چه قالبی باید جمع‌آوری و نگهداری و به‌هنگام‌سازی شوند.

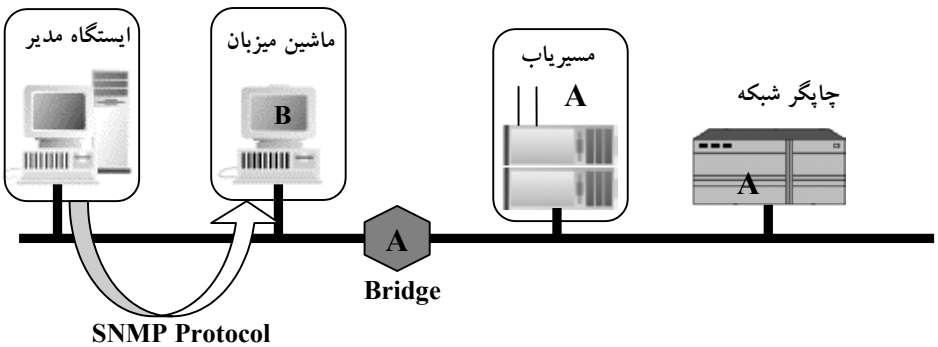
تاکنون برای مدیریت شبکه ، راهکارها و استانداردهای مختلفی تعریف شده است. اولین استانداردهای مدیریت در شبکهٔ اینترنت توسط IETF در RFC-1068 و RFC-1028 تعریف شدند که بدلیل نقائص و قابلیت اطمینان پایین ، طول عمرشان زیاد نبود. در ماه مه ۱۹۹۰ نسخه اول پروتکل مدیریت سادهٔ شبکه[۵] منتشر شد که بعداً با مشخص شدن عیوب آن ، نسخه اصلاح شده و پیشرفته‌تر آن به نام SNMPv2 ارائه گردیده و به عنوان استاندارد مدیریت در شبکهٔ اینترنت مطرح شد. از جمله استانداردهای مدیریت شبکه می‌توان به CMOT[۴] و RMON[۵] نیز اشاره کرد.

---

۱ Simple Network Management Protocol
۲ Monitoring
۳ Management Information
۴ Common standard Management information protocol Over TCP/IP
۵ Remote Monitor

### ۲-۵) مدل SNMP

در مدل SNMP کلیهٔ عناصرِ یک شبکهٔ خودمختار به چهار ردهٔ زیر تقسیم‌بندی می‌شوند. در شکل (۱۰-٦) این تقسیم‌بندی نشان داده شده است:




شکل (۱۰-٦) اجزای مدل مدیریت در SNMP

- **نودهای تحت مدیریت**[۱] : شامل ماشینهای میزبان ، مسیریابها ، پلها، چاپگرها و هر ماشین دیگری که بتواند اطلاعاتی از وضعیت خود ، به ایستگاههای مدیر ارسال نماید و از فرامین آنها تبعیت کند. یک نود تحت مدیریت باید قادر به اجرایِ پروسهٔ کاربردی SNMP باشد. در این حالت به آن ایستگاه **نمایندگی SNMP**[۲] گفته می‌شود. هر نود تحت مدیریت ممکن است در کنترل چند ایستگاه مدیریت باشد که هر یک از این ایستگاههای مدیر ، سطوح دسترسی متفاوتی به آن ایستگاه دارند.
- **ایستگاههای مدیریت**[۳] : این ایستگاهها مراکز مدیریت شبکه می‌باشند و معمولاً کامپیوترهای همه‌منظوره‌ای هستند که نرم‌افزار لازم برای مدیریت بر روی آنها نصب شده است. این ایستگاهها با نمایندگیها (نودهای تحت مدیریت) ارتباط برقرار کرده ، دستوراتی را صادر و پاسخهایی را دریافت می‌کنند. ممکن است نرم افزار مدیریت ، دارای رابط گرافیکی[٤] باشد که مسئول شبکه به سادگی وضعیت شبکه را نظارت کند.

---

۱ Managed Nodes
۲ SNMP Agent
۳ Management Stations
٤ Graphical User Interface

- **اطلاعات مدیریت** : هر ایستگاه یک یا چند "متغیر وضعیت" را در حافظه سازماندهی و نگهداری می‌کند که این متغیرها وضعیت فعلی آنرا توصیف می‌کنند. در ادبیات پروتکل SNMP ، این متغیرها **"اشیاء"** نامیده شده‌اند.
- **قرارداد مدیریت** : روشی است استاندارد و مستقل که بر اساس آن ، ایستگاه مدیر با نمایندگیها ارتباط برقرار می‌کند و قادر است حالت **اشیاء** (متغیرهای وضعیت) آنها را تقاضا کرده و در صورت لزوم آنها را تغییر دهد.

هرگاه در شبکه واقعهٔ پیش‌بینی نشده‌ای رخ بدهد و یکی از نمایندگیها متوجه شود ، وقوع آن را به تمام ایستگاههای مدیریت اطلاع می‌دهد. این گزارش به دلایل تاریخی **"تله"** ( TRAP ) نامیده می‌شود. از آنجایی که ارتباط ایستگاه مدیریت با گره‌های تحت مدیریت ، قابل‌اعتماد نیست[۱] ، بهمین دلیل در فواصل زمانی مشخص ، نودهای تحت مدیریت توسط ایستگاه مدیریت سرکشی[۲] می‌شوند؛ به سرکشی ایستگاههای مدیر "سرکشی مستقیم تله"[۳] اطلاق می‌شود. برای ماشینها و شبکه‌های قدیمی یا هر ماشینی که قادر به اجرای پروسهٔ SNMP نیستند از "نمایندگی وکالت"[۴] استفاده می‌شود ، یعنی سیستمی که قادر است به هر نحو آنها را مدیریت کند ، باید اطلاعات لازم را از آنها اخذ و سازماندهی کرده و در قالب پروتکل SNMP در اختیار ایستگاههای مدیر قرار دهد.

امنیت و قابلیت اطمینان[۵] در شبکه حساسیت ویژه‌ای دارد؛ از آنجایی که یک ایستگاه مدیر می‌تواند اطلاعات زیادی از گره‌های تحت مدیریت بدست آورده و حتی آنها را از کار بیاندازد ، لذا قبل از برقراری هرگونه ارتباط و مبادلهٔ اطلاعات ، هر یک از نمایندگیها باید به نوعی متقاعد گردند که تقاضاها و فرمانها از طرف یک ایستگاه مدیریتِ مجاز صادر شده است. برای این منظور در SNMPv2 کلمهٔ رمز و روشهای رمزنگاری جدید استفاده می‌شود ، زیرا بدون احراز هویت صادر کنندهٔ پیام ، یک کاربر اخلالگر قادر خواهد بود پیامهای مدیریتی را بصورت مصنوعی تولید و منتشر نماید ، یا فرمان از کار افتادن بخشی از اجزای شبکه را صادر نماید.

---

[۱] **عدم اطمینان از آنجایی ناشی می‌شود که هنگام ارسال هر گونه گزارش به ایستگاههای مدیریت ، هیچگونه پیغام تصدیق ( Ack ) برنمی‌گردد و یک ایستگاه پس از ارسال اطلاعات مدیریتی هیچ اطلاعی از سرنوشت آنها نخواهد داشت.**
[۲] Pulling
[۳] Trap Directed Polling
[۴] Proxy Agent
[۵] Security/Reliability

### ۳-۵) استانداردهای مدیریت داده

همانطورکه اشاره شد ، هر نود تحت مدیریت ، باید دارای مجموعهٔ استانداردی از متغیرها برای توصیف وضعیت خود ( از قبیل میزان ترافیک ورودی و خروجی ، نرخ خرابی بسته‌های داده ، وضعیت اجزای مرتبط و ... ) باشد. بر این اساس **"استاندارد مدیریت داده"** در شبکه بوجود آمده است. به مجموعهٔ اطلاعات مدیریتی و ساختار پیاده‌سازی آن "پایگاه دادهٔ اطلاعات مدیریتی"[۱] (MIB) اطلاق می‌شود.

دو سازمانی که SNMP و CMOT را ابداع نمودند ، یک استاندارد مدیریت داده در شبکه به نام MIB پیشنهاد کرده‌اند که در واقع یک بانک اطلاعاتی جهت ذخیره سازی اشیاء و متغیرها می‌باشد. استاندارد MIB مستقل از پروتکلهای مدیریت شبکه تعریف شده است که این نکته باعث شده ، امکان تغییر پروتکل مدیریت ، بدون نیاز به تغییر MIB وجود داشته باشد ولی عیب اساسی آنست که پروتکلهای مدیریت شبکه باید از اطلاعات مدیریتی یکسان استفاده کنند. MIB دارای ۱۰ گروه از اشیاء است که در جدول (۱۱-۶) فهرست شده‌اند. برای تعریف و کدگذاری استاندارد اشیاء نیز از یک زبان به نام ASN.1[۲] استفاده می‌شود که در ادامه به معرفی آن می‌پردازیم.

| نام گروه | شمارهٔ شیئ | نوع متغیرهای وضعیتی که هر شیئ نگهداری می‌کند |
|---|---|---|
| System | 7 | **Name,location and description of the equipment** |
| Intefaces | 23 | **Network interfaces and their measured traffic** |
| AT | 3 | **Address translation (deprecated)** |
| IP | 42 | **IP packet statistics** |
| ICMP | 26 | **Statistics about ICMP messages received** |
| TCP | 19 | **TCP algorithms,parameters and statistics** |
| UDP | 6 | **UDP traffic statistics** |
| EGP | 20 | **Exterior gateway protocol traffic statistics** |
| Transmission | 0 | **Reserved for media specific MIBs** |
| SNMP | 29 | **SNMP traffic statistics** |

جدول (۱۱-۶) گروههای اشیاء MIB-II در اینترنت

---

[۱] Management Information Base
[۲] Abstract Syntax Notation 1

### ۴-۵) زبان توصیفی ASN.1

هستهٔ اصلی پروتکل SNMP ، مجموعه‌ای از اشیاء (متغیرهای وضعیت) است که می‌تواند توسط ایستگاههای مدیریت ، خوانده یا نوشته شوند. برای آنکه تمام ایستگاههای شبکه با هر گونه اختلاف بنیادی از لحاظ سخت‌افزار و نرم‌افزار ، قادر به ارتباط با مدیر شبکه باشند ، باید ساختار اشیاء دقیقاً استاندارد باشد. زبان توصیفی ASN.1 ، استانداردی است که با آن اشیاء و متغیرهای حالت تعریف می‌شوند. همانند زبان C که قادرید طبق یک روال استاندارد متغیرها و استراکچرها را تعریف نمایید ، ASN.1 نیز یک روش برای تعریف[1] متغیرها و اشیاء ارائه کرده است. نکتهٔ دوم آنست که روش ارسال این اشیاء روی کانال باید دقیقاً تبیین شود؛ چیزی که در زبانC به آن نیاز نبوده است ولی در SNMP حیاتی است. به عنوان مثال شما نمی‌دانید یک استراکچر با دو متغیر از نوع long ، چگونه و به چه ترتیبی روی کانال ارسال می‌شود. ( اول کدام بایت از کدام متغیر ) یا مثلاً در C نمی‌توانید یک متغیر ۱۳ بیتی تعریف کنید ، در حالی که این کار در ASN.1 ممکن است.

اطلاق کلمهٔ **شیئ** به متغیرهای حالت ، پیشنهاد خوبی نیست زیرا در مفهوم شیئ‌گرایی ، یک شیئ مجموعه‌ای از متغیرهای حالت و متودهاست ، در حالی که در ASN.1 ، اشیاء هیچ متودی ندارند. اطلاق "متغیر حالت" بجای شیئ مناسبتر است ولی چاره‌ای جز تبعیت از ادبیات استاندارد ASN.1 نداریم.

ASN.1 که بخشی از آن توسط SNMP مورد استفاده قرارمی‌گیرد ، دارای دو مجموعهٔ استاندارد می‌باشد:

- یک نوع زبان توصیف اشیاء که توسط کاربر قابل استفاده است.
- یک روش کدگذاری[2] برای مبادلهٔ اطلاعات بین ایستگاههایی که از پروتکل SNMP پشتیبانی می‌کنند.

استاندارد ASN.1 ، هرگونه ابهامی را در شکل و محتوای اشیاء از بین می‌برد. به عنوان مثال برای یک شیئ می‌توان نوع و محدوده مقادیر آن را تعریف کرد که این دقت در بیان متغیرها ، با توجه به وجود کامپیوترهای متنوع در شبکه اهمیت بیشتری پیدا می‌کند.[3] همچنین این زبان باعث آسانتر شدن پیاده‌سازی پروتکلهای مدیریت

---

[1] Declaration
[2] Encoding Rule
[3] **به عنوان مثال یک متغیر صحیح ( Integer ) در ماشینهای متفاوت می‌تواند طول متفاوتی داشته باشد. ( ۲ بایت یا ۴ بایت ) بنابراین در هنگام تعریف اشیاء به نحوی که فارغ از ساختار ماشین باشند ، توصیف یک شیئ باید کاملاً دقیق باشد.**

شبکه می‌گردد چراکه براساس زبان ASN.1 داده‌ها و متغیرها در یک پیغام قابل رمزگذاری هستند. ( رمزگذاری به معنای تبیین قالب ارسال اشیاء –Encoding–)

با استفاده از این زبان کاربر اشیاء اولیه را تعریف کرده و سپس اشیاء پیچیده‌تری را به کمک آنها ایجاد می‌کند. در حقیقت تعریف اشیاء و متغیرها در ASN.1 همانند تعریف متغیرها و استراکچرها در زبان C البته با نماد و اصول متفاوت می‌باشد. چند قاعده را در نامگذاری اشیاء باید درنظر گرفت :

- انواع داده‌ای قبل تعریف‌شده با حروف بزرگ نوشته می‌شوند. ( مثل INTEGER ) در جدول (۱۲–٦) برخی از این انواع از پیش تعریف شده فهرست شده است. ( در SNMP انواع داده‌ای اعشاری و بولیَن مورد استفاده نیستند. )
- انواع داده که توسط کاربر تعریف می‌شود اگرچه می‌تواند با حروف بزرگ نوشته شود ولی باید حداقل دارای یک حرف غیر بزرگ نیز باشد.
- نام شناسه‌ها[1] با حروف کوچک شروع می‌شود ولی می‌تواند در ادامه حاوی حروف بزرگ نیز باشند.
- خطوط توضیح با دو کاراکتر -- شروع شده و تا انتهای خط یا -- بعدی ادامه می‌یابند.

مثلاً در زیر یک متغیر صحیح با محدودهٔ خاص تعریف شده است :

Packetsize **::=** INTEGER (0..1023)

- **Object Identifier** روشی برای شناسایی اشیاء است. هر شیئ باید بطور منحصر بفرد قابل شناسایی باشد. مکانیزیم شناسایی اشیاء ، استفاده از درخت استانداردها می‌باشد. همانطور که در شکل (۱۳–٦) مشاهده می‌شود در سطح فوقانی این درخت سازمانهای تعیین استاندارد وجود دارند که بخشی از زیردرخت های آن به **SNMPv2** اختصاص داده شده است. هر نود در درخت دارای برچسب و کد مخصوص به خود است. برای توصیف اشیاء می‌توان از برچسب یا کدهای درخت ASN استفاده کرد. مثلاً برای نامگذاری متغیر **IpInReceive** که با کد شماره ۳ در زیرگروه **ip** قرار دارد می‌توانیم از روشهای زیر استفاده کنیم:

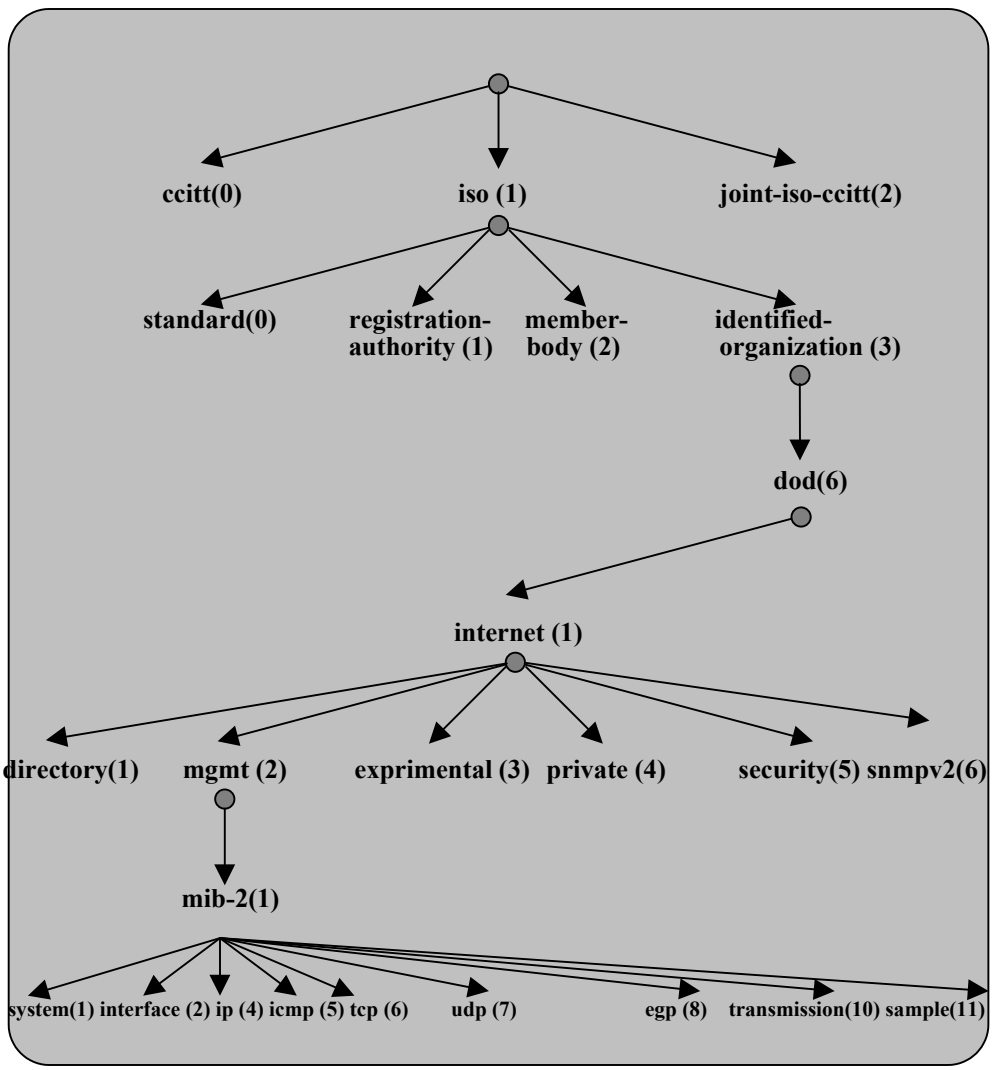
**{ iso.org.dod.internet.mgmt.mib.ip.IpInReceives }**

**{1.3.6.1.2.1.4.3} یا**

[1] Identifiers

| *Primitive type* | *Meaning* | *Code* |
|---|---|---|
| **INTEGER** | Arbitrary length integer | 2 |
| **BIT STRING** | A string of 0 or more bits | 3 |
| **OCTET STRING** | A string of 0 or more signed bytes | 4 |
| **NULL** | A place holder | 5 |
| **OBJECT IDENTIFIER** | An officially defined data type | 6 |

جدول (۱۲-٦) داده های از پیش‌تعریف شده در ASN.1 که در SNMP استفاده می‌شود.




شکل (۱۳-٦) بخشی از درخت نامگذاری اشیاء در ASN.1

مثالی از تعریف یک عدد صحیح و مقداردهی اولیه بر اساس زبان ASN:

count INTEGER ::= 100

count متغیری صحیح با مقدار اولیهٔ ۱۰۰ تعریف شده است.

در **ASN.1** چندین روش برای تعریف انواع جدید داده وجود دارد. **SEQUENCE** لیستی مرتب از انواع داده را همانند رکورد در پاسکال بوجود می‌آورد. **SEQUENCE OF** یک آرایهٔ تک‌بُعدی از یک نوع دادهٔ خاص را تعریف می‌کند. در اینجا به یک مثال کلی و ذکر چند نکته می‌پردازیم: در گروه **ip** از **MIB** یک شیئ به نام **ipAddrTable** تعریف شده است که شامل آرایه‌ای از مشخصه‌های آدرس **IP** می‌باشد و کد ۲۰ به آن تخصیص یافته است. تعریف این شیئ بر اساس ASN.1 بصورت زیر است:

**IpAddrTable::= SEQUENCE OF IpAddrEntry**

**آرایه‌ای از**

**IpAddrEntry::= SEQUENCE {**

**ipAdEnt Addr IpAddress,**

**ipAdEntIfIndex INTEGER,**

**ipAdEntNetMask IpAddress,**

**ipAdEntBcastAddr IpAddress,**

**ipAdEntReasmMaxSize INTEGER(0.. 65535)**

**}**

بر طبق تعریف بالا IpAddrTable آرایه‌ای از اشیاء IpAddrEntry است. شیئ IpAddrEntry خودش یک رکورد حالت با پنج متغیر می‌باشد. از این پنج متغیر ، سه تای آنها از نوع IpAddress تعریف شده که عملاً یک عدد صحیح ٤ بایتی بدون علامت است. در ASN اینگونه از انواع را می‌توانید به صورت زیر تعریف کنید:

**IpAddress::= [APPLICATION 1] INTEGER (0..4294967295)**

بدین معنا که نوع IpAddress با کد شمارهٔ ۱ ، عددی صحیح و ٤ بایتی با محدودهٔ تغییر از0 تا $2^{32}$-1 است. از آن به بعد IpAddress یک نوع شناخته شده محسوب می‌شود.

در ASN برخلاف اغلب زبانها به جای اندیس در آرایه ، از پسوند در انتهای نام متغیر استفاده می‌شود.

### ۵-۵) نحوه انتقال در ASN.1

در ASN.1 ، روش تبدیل متغیرها و داده‌ها به دنباله‌ای از بایتها ، نحوه کدگذاری و نیز دیکود دنبالهٔ بایتها ، دقیقاً مشخص شده است؛ این روش به نام **BER**[1] مشهور است. در این روش ، قالب هر شیئ که روی کانال ارسال می‌شود ، اعم از مقادیر اولیه و استراکچرها شامل چهار فیلد زیر است:

| Identifier | Data Length Field | Data | End-of-Content Flag |
|---|---|---|---|

- **شناسه (نوع یا برچسب)** : این فیلد خود دارای سه قسمت می‌باشد که دو بیت با ارزش آن ، نوع شناسه را مشخص می‌کند. فیلد یک بیتی بعدی مشخص می‌کند که نوع متغیر ابتدایی[2] است (0) و یا ساخته‌شده[3] (1) می‌باشد. ۵ بیتِ باقیمانده برای تعیین شمارهٔ نوع متغیر یا شیئ است. شمارهٔ انواع متغیر در جدول (۱۲-٦) آمده است. در زیر قالب بایت شناسه یعنی اولین بایتی که در استاندارد ASN.1 روی کانال انتقال می‌یابد ، مشخص شده است:

| 2 Bits | 1 Bit | 5 Bits |
|---|---|---|
| Tag | Type | Number |

00 Universal — 0 Primitive Type

01 Application — 1 Constructed Type

10 Context specific

11 Private

- **طول فیلد داده‌ها بر حسب بایت** : اگر طول بایتهای داده کمتر از ۱۲۸ باشد مقدار آن در بایتی که بیت با ارزش آن صفر است نوشته می‌شود. برای داده‌های با طول بیشتر از ۱۲۸ ، این فیلد چند بایتی است ، بطوریکه بیت با ارزشِ بایت اول و هفت بیت کم ارزش دارای حداکثر مقدار ۱۲۷ می‌باشد. مثلاً اگر طول داده ۱۰۰۰ بایت باشد ، ابتدا در بایت اول مقدار ۱۳۰ قرار

---

[1] Basic Encoding Rules
[2] Primitive
[3] Constructed Type

می‌گیرد تا نشان بدهد دو بایت بعدی دارای مقدار هستند (۲+۱۲۸) سپس در دو بایت بعدی معادل دودویی عدد ۱۰۰۰ قرار می‌گیرد.

- **فیلد داده** : داده‌های از نوع صحیح به صورت مکمل ۲ کد می‌شوند. رشته‌های بیتی به همان شکل خودشان کدگذاری و از بیت پرارزش به کم ارزش روی کانال ارسال می‌شوند. چون ممکن است تعداد بیتهای رشته ضریبی از ۸ نباشد ، بایت اول که قبل از رشته ارسال می‌شود ، تعداد بیتهای بلااستفادهٔ بایت آخر از رشتهٔ بیتی را مشخص می‌کند؛ مثلاً برای رشته ۹ بیتی "۰۱۰۰۱۱۱۱۱" ، رشتهٔ زیر انتقال می‌یابد:

| 00000111 | 01001111 | 10000000 |
|---|---|---|
| 07 | 4F | 80 |

برای انتقال شناسه‌های اشیاء از دنباله اعداد صحیح استفاده می‌شود. مثلاً برای تعریف "شیئِ اینترنت"[1] با دقت به درخت ASN.1 ، باید رشته {۱و ٦ و ۳ و ۱ } ارسال شود.

- **نشانگر انتهای داده** : پایان داده‌ها را مشخص می‌کند. در پروتکل SNMP از این فیلد استفاده نمی‌شود ، چراکه فیلد طول داده‌ها حتماً باید ذکر شود و از اینرو دیگر نیازی به نشانگر انتهای داده نیست. در زیر مثالهایی از نحوه کدگذاری برای داده ها و اشیاء دیده می‌شود:

- **ارسال عدد صحیح ۴۹ با قاعدهٔ BER**

| | Tag | | Type Code | Length | Value |
|---|---|---|---|---|---|
| **Integer 49** | **00** | **0** | **00010** | **00000001** | **00110001** |

- **ارسال رشتهٔ سه بیتی 110 با قاعدهٔ BER**

| | Tag | | Type Code | Length | Value | Value |
|---|---|---|---|---|---|---|
| **Bit String 110** | **00** | **0** | **00011** | **00000010** | **00000101** | **11000000** |

- **ارسال رشتهٔ دو بایتی "xy" با قاعدهٔ BER**

| | Tag | | Type Code | Length | Value | Value |
|---|---|---|---|---|---|---|
| **String "xy"** | **00** | **0** | **00100** | **00000010** | **01111000** | **01111001** |

[1] Internet Object

- **ارسال بایت NULL با قاعدهٔ BER**

| | Tag | | Type Code | Length |
|---|---|---|---|---|
| **NULL** | 00 | 0 | 00101 | 00000000 |

- **ارسال شیئ Internet Object در درخت ASN با قاعدهٔ BER**

| | Tag | | Type Code | Length | Value | Value | Value |
|---|---|---|---|---|---|---|---|
| **Internet Object** | 00 | 0 | 00110 | 00000011 | 00101011 | 00000110 | 00000001 |

## ۶-۵) ساختار اطلاعات مدیریتی SMI[1]

با توجه به اینکه متغیرهای وضعیت در یک ایستگاه بسیار متنوعند لذا در SMI زیرمجموعهٔ مهمی از اشیاء در ASN.1 ، تحت نام ساختار اطلاعات تعریف شده که برای تشکیل ساختمان داده‌های لازم و مقداردهی آنها کاربرد دارد. این استاندارد شامل چهار ماکرو و هشت نوع دادهٔ جدید است. مثلاً اگرچه متغیر **Counter** و متغیر **Gauge** هر دو یک عدد ۳۲ بیتی هستند ولی یک تفاوت بنیادی دارند : متغیر **Counter** پس از رسیدن به مقدار $2^{32}$-1 ، صفر خواهد شد ولی متغیر **Gauge** پس از رسیدن به مقدار $2^{32}$-1 ، دیگر نمی‌شمارد تا زمانیکه یا صفر شود یا از مقدار آن کاسته شود.

برای اینکه پروتکل مدیریت شبکه ساده باشد محدودیتهایی بر روی متغیرهای بانک اطلاعات مدیریتی اعمال شده است. SMI ضمن تعریف انواع دادهٔ جدید امکان استفاده از آنها را در تعریف اشیاء جدید فراهم کرده است. مهمتر از همه اینکه SMI نحوه رجوع به جداول مقادیر ، مانند جدول مسیریابی را شرح می‌دهد. در جدول (۱۴-۶) انواع داده را که در ساختار مدیریت اطلاعات توسط استاندارد SMI تعریف شده است ، مشاهده می‌کنید.

[1] Structure for Managemenent Information

| *Name* | *Type* | *Bytes* | *Meaninig* |
|---|---|---|---|
| **INTEGER** | Numeric | 4 | integre (32 bits in current implementaions |
| **Counter32** | Numeric | 4 | Unsigned 32-bit counter that wraps |
| **Gauge32** | Numeric | 4 | Unsigned value that does not wrap |
| **Integer32** | Numeric | 4 | 32 Bits, even on a 64-bit CPU |
| **UInteger32** | Numeric | 4 | Like Inetger 32,but unsigned |
| **Counter64** | Numeric | 8 | A 64 bit counter |
| **TimeTicks** | Numeric | 4 | In hundredths of a second since some epoch |
| **BIT STRING** | Numeric | 4 | Bit map of 1 to 32 bits |
| **OCTET STRING** | String | >= 0 | Variable length byte string |
| **Opaque** | String | >= 0 | Obfolete; for backward compatibility only |
| **Object Identifier** | String | >0 | A list of integers form Fig 3 |
| **IpAddress** | String | 4 | A dotted decimal interner address |
| **Nsap Address** | String | <22 | An OSI NSAP address |

شکل (۱٤-٦) انواع دادهٔ جدید تعریف شده در SMI برای استفاده در SNMP

### ۵-۷) پروتکل سادهٔ مدیریت شبکه (SNMP)

به گونه‌ای که اشاره شد مدیریت شبکه درخواستی را به نمایندگی می‌فرستد و اطلاعاتی را از وی خواسته و یا در صدد تغییر حالت آن برمی‌آید. در SNMP حدود ۱۷۵ متغیر حالت و وضعیت تعریف شده که هر یک به نحوی برای نظارت و مدیریت شبکه بکار می‌آید. ارسال داده‌های حالت نیز براساس استاندارد انتقال **ASN.1** انجام می‌شود. ( البته ممکن است خطاهای متعددی مانند عدم وجود متغیر گزارش گردد.)

در یک پروتکل مدیریتِ شبکه ، ممکن است انواع مختلفی از دستورات موجود باشد که این باعث پیچیدگی زیاد می‌شود چراکه برای هر نوع عملیاتی ، نیاز به اضافه کردن دستورات جدید خواهد بود. در SNMP از یک روش جالب استفاده می‌شود ، به این ترتیب که تمامی عملیات و فرمانها در قالب روش واکشی و ذخیرهٔ متغیرهای حالت انجام می‌شود. مثلاً برای بوت کردن دوباره یک ایستگاه تحت مدیریت ، از تغییر مقدار در یک متغیر ویژه استفاده می‌شود. پروتکل SNMP مجموعاً دارای هفت فرمان است. شش فرمان از هفت فرمان موجود در جدول (۱۵-٦) آمده است. فرمان Get-request برای درخواست مقدار یک یا چند متغیر حالت است.

فرمان **get-next-request** امکان درخواست متغیر بعدی را می‌دهد. فرمان **get_bulk_request** برای واکشی جداول بزرگ است. فرمان **set-request** برای بهنگام‌سازی متغیرهای حالت می‌باشد. با فرمان **inform-request** یک مدیر به مدیر دیگر متغیری را که در حال مدیریت آن است اعلام می‌کند و فرمان **SnmpV2-trap** جهت گزارش یک رخداد از طرف یکی از ایستگاهها به مدیر ، کاربرد دارد.

| *Message* | *Description* |
|---|---|
| **Get-request** | Requests the value of one or more variables |
| **Get-next-rquest** | Request the variable foolowing this one |
| **Get-bulk-request** | Fetches a large table |
| **Set-request** | Updates one or more variables |
| **Inform-request** | Managaer to manager message describing local MIB |
| **SnmpV2-trap** | Agent-to-manager trap report |

جدول (۱۵-٦) انواع پیغامهای SNMP

همانطور که گفته شد در ASN برای مشخص کردن اعضای آرایه به جای اندیس از یک پسوند که به دنبال شناسه شیئ می‌آید ، استفاده می‌شود. حال مسئله این است که ایستگاه مدیریت باید بتواند عناصر موجود در جدول را واکشی نماید و برای اینکار باید پسوندها را به دست آورد. دستور **get-next-request** متغیر بعدی در جدول را براساس نام متغیری که به آن ارسال می‌شود ، واکشی می‌کند و بطور پیش‌فرض در صورتیکه نام متغیر بدون پسوند فرستاده شود عنصر اول در جدول واکشی می‌شود و اگر نام متغیر بطور کامل ارسال شود ، عنصر بعدی موجود در جدول را برمی‌گرداند. پروسهٔ SNMP در ایستگاه تحت مدیریت ، بر اساس کد ارسال شده ، اولین متغیر حالتی را که کد آن از لحاظ ترتیب در جدول بزرگتر از کد موردنظر است ، برمی‌گرداند. با این روش می‌توان عناصر یک جدول را به ترتیب واکشی کرد.

پیغامهای SNMP دارای طول مشخصی می‌باشند و رمزگشایی آنها توسط افراد عادی ساده نیست. یک پیغام **SNMP** شامل سه بخش مهم می‌باشد؛ شماره نسخهٔ

پروتکل SNMP ، یک شناسه که گروه ایستگاههای تحت نظارت یک مدیر را مشخص می‌کند؛ و بخش داده که به چند واحد داده تقسیم می‌شود. هر واحد دادهٔ پروتکل شامل یک درخواست از طرف ایستگاه مدیریت و یک پاسخ از سوی ایستگاه تحت مدیریت می‌باشد. در زیر قالب پیغام به زبان ASN نمایش داده شده است:

```
SNMP-Message ::=
                SEQUENCE {
                    version INTEGER {
                                version-1 (0)
                    },
                community
                    OCTET STRING,
                data
                    ANY
            }
```

پروتکل SNMP بر خلاف اسم آن – **"پروتکل سادهٔ مدیریت شبکه"** – چندان هم ساده نیست و مستندات آن ۶۰۰ صفحه را در بر می‌گیرد. بسیاری از منتقدان معتقدند هرگاه سازمان استاندارد جهانی ISO قدم در استاندارد کردن یک پروتکل بر می‌دارد به دلیل وسواسی که در کامل بودن آن استاندارد دارد ، چنان آنرا پیچیده می‌کند که پیاده‌سازی ، استفاده و فهم آن مشکل می‌شود. ASN.1 نیز از این قاعده مستثنی نیست و کلاً پیچیده و مبهم است. علاقمندانی که به رسالت پروتکلهای مدیریت شبکه همانند SNMP و زبان توصیف اشیاء مثل استاندارد ASN پی برده‌اند برای درک کامل جزئیاتِ آنها ، باید به مراجع معرفی شده مراجعه کنند.

## ۶ ) مراجع این فصل

مجموعهٔ مراجع زیر می‌توانند برای دست آوردن جزییات دقیق و تحقیق جامع در مورد مفاهیم معرفی شده در این فصل مفید واقع شوند.


| | |
|---|---|
| **"Computer Networks" , Andrew S.Tanenbaum, Third Edition, Prentice-Hall, 1996.** **"Internetworking with TCP/IP", Commer D.E. ,Prentice-Hall, 1996.** | |
| RFC 1101 | "DNS Encoding of Network Names and Other Types," Mockapetris, P.V.; 1989 |
| RFC 1035 | "Domain Names—Implementation and Specification," Mockapetris, P.V.; 1987 |
| RFC 1034 | "Domain Names—Concepts and Facilities," Mockapetris, P.V.; 1987 |
| RFC 1033 | "Domain Administrators Operations Guide," Lottor, M.; 1987 |
| RFC 1032 | "Domain Administrators Guide," Stahl, M.K.; 1987 |
| RFC 974 | "Mail Routing and the Domain System," Partridge, C.; 1986 |
| RFC 920 | "Domain Requirements," Postel, J.B.; Reynolds, J.K.; 1984 |
| RFC 799 | "Internet Name Domains," Mills, D.L.; 1981 |
| RFC 1157 | Simple Network Management Protocol (SNMP). J.D. Case, M. Fedor, M.L. Schoffstall, C. Davin. May-01-1990 |
| RFC 1161 | SNMP over OSI. M.T. Rose. Jun-01-1990. |
| RFC 1187 | Bulk Table Retrieval with the SNMP. M.T. Rose, K. McCloghrie, J.R. Davin. Oct-01-1990. |
| RFC 1902 | Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2). SNMPv2 Working Group, J. Case, K. McCloghrie, M. Rose & S. Waldbusser. January 1996. |
| RFC 2578 | Structure of Management Information Version 2 (SMIv2). K. McCloghrie, D. Perkins, J. Schoenwaelder. April 1999. |

## ۱) مقدمه

در فصول گذشته ساختار پروتکلهای TCP و IP را بررسی کردیم و طریقه آدرس‌دهی ماشینها و پروسه های روی هر ماشین را بوسیله آدرس IP و آدرس پورت آموختیم. معمولاً پیاده‌سازی این پروتکلها توسط طراح هر سیستم عامل انجام و بعنوان جزئی از سیستم عامل همراه آن ارائه و نصب می‌شود. در سیستم عامل هایی مثل یونیکس یا MS-Windows که اصل[۱] برنامه آن در دسترس نیست این انعطاف وجود ندارد که بتوان در محیطهای آزمایشگاهی برنامه های TCP و IP یا پروتکلهای مرتبط با آنها را تغییراتی داد و نتیجه تغییرات را بررسی و تحلیل کرد لذا نظر علاقمندان به این مورد را به سیستم عامل " لینوکس" جلب می‌نمائیم.

حال فرض می‌کنیم یک برنامه نویس بخواهد در یک محیط برنامه نویسی مثل C بگونه ای برنامه نویسی کند که محتویات یک فایل درون یک کامپیوتر راه دور را تغییر بدهد یا آنرا روی کامپیوتر خودش منتقل نماید یا فرض کنید یک برنامه نویس موظف شده است که یک محیط پست الکترونیکی خاص و با امکانات ویژه برای بکارگیری در یک محیط اداری طراحی نماید. برای طراحی چنین برنامه هایی که تماماً در لایه چهارم یعنی لایه کاربرد تعریف می‌شود برنامه نویس باید به نحوی با مفاهیم برنامه نویسی تحت شبکه آشنا باشد.

در این فصل اصول کلی برنامه نویسی شبکه و مفهوم "سوکت"[۲] را مورد بررسی قرار می‌دهیم و با مثالهای ساده روش نوشتن برنامه های کاربردی تحت پروتکل TCP/IP را تشریح خواهیم کرد. برای سادگی کار و همچنین ارائه دید عمیق ، کدهائی که در این فصل ارائه می‌شوند به زبان C هستند که در محیط سیستم عامل لینوکس و با مترجم **gcc** به زبان ماشین ترجمه شده‌اند.

در حقیقت این فصل نقطه آغازی است برای تمام برنامه نویسانی که به نحوی مجبور خواهند شد برنامه کاربردی تحت شبکه اینترنت بنویسند.

سنگ بنای تمام برنامه های کاربردی لایه چهارم مفهومی بنام سوکت است که این مفهوم توسط طراحان سیستم عامل یونیکس به زیبایی به منظور برقراری ارتباط برنامه های تحت شبکه و تبادل جریان داده بین پروسه ها ابداع شد و در این فصل باید مفهوم آنرا کالبد شکافی کنیم.

---

[۱] Source
[۲] Socket

شاید شما این جمله را شنیده باشید **"در دنیای یونیکس هر چیزی می‌تواند بصورت یک فایل تلقی و مدل شود"**. تمام عوامل و انواع ورودی و خروجیها (I/O) می‌تواند توسط سیستم فایل مدل شود. مثلاً چاپگر می‌تواند یک فایل باشد (مثلاً فایلی با نام PRN ). حال وقتی سیستم عامل چاپگر را بصورت یک فایل استاندارد مدل کرده باشد شما با مفاهیمی که از فایلها و چگونگی بکارگیری آنها در محیط برنامه نویسی آموخته اید، برای راه اندازی چاپگر و چاپ یک متن، می‌توانید در برنامه خود عملیات ساده و در عین حال استاندارد زیر را انجام بدهید:

**الف**: چاپگر را همانند یک فایل با نام استاندارد آن بصورت فایلی "فقط نوشتنی" باز می‌کنید. (با دستورات open() یا fopen() ).

**ب**: اگر نتیجه مرحله قبل موفقیت آمیز بود سیستم عامل یک مشخصه فایل[1] بعنوان اشاره گر فایل برمی‌گرداند.

**ج**: داده‌هائی که قرار است بر روی چاپگر ارسال شوند را با همان دستورات معمولی نوشتن در فایل ( دستور معمولی write() یا fwrite() ) ، درون فایل باز شده از مرحله قبل می‌نویسید.

**د**: پس از اتمام کار فایل را می‌بندید. (دستور close() یا fclose() )

کلیت کاری که باید انجام بشود همین چند مرحله است و برنامه نویس به هیچ عنوان درگیر ساختار چاپگر و اعمالی که برای راه اندازی و چاپ یک متن لازم است نخواهد شد. این وظائف را راه انداز چاپگر بعنوان بخشی از پوسته سیستم عامل بعهده دارد.

چهار مرحله‌ای که برای بکارگیری چاپگر معرفی شد دقیقا میتواند برای نوشتن بر روی صفحه نمایش یا خواندن از آن مورد استفاده قرار گیرد ، فقط باید نام فایل صفحه نمایش "کنسول" (con) در نظر گرفته شود.

یونیکس قادر است تمام دستگاههای ورودی و خروجی را بعنوان فایل مدل نماید. بنابراین تمام عملیاتی که برنامه نویس برای بکارگیری دستگاههای مختلف بایستی بداند و بکار بگیرد یکسان و ساده و شفاف خواهد بود. آیا می‌توانید گزاره های زیر را بپذیرید:

- چاپگر فایلی است فقط نوشتنی

---

[1] File Descriptor

- پویشگر[۱] فایلی است فقط خواندنی
- صفحه نمایش بعنوان کنسول فایلی است خواندنی و نوشتنی
- پورت سریال فایلی است خواندنی و نوشتنی
- یک فایل واقعی روی دیسک سخت فایلی است خواندنی و نوشتنی
- یک فایل واقعی روی دیسک فشرده فایلی است فقط خواندنی
- صف FIFO یا خط لوله[۲] در محیط یونیکس فایلهایی هستند خواندنی و نوشتنی

حال که ذهن شما این نکته را پذیرفت که هر نوع I/O در دنیای سیستم عامل بصورت یک فایل استاندارد قابل عرضه و مدل کردن است ، شما را با یک سوال کلیدی مواجه می‌کنیم :

*"آیا ارتباط دو کامپیوتر روی شبکه و مبادله اطلاعات بین آن دو، ماهیت ورودی / خروجی (I/O) ندارد؟"*

اگر جوابتان منفی است این فصل را رها کنید ولی اگر تردید دارید یا یقیناً جوابتان مثبت است تا انتها این فصل را دنبال نمایید.

اگر ساختار فایل را برای ارتباطات شبکه ای تعمیم بدهیم آنگاه برای برقراری ارتباط بین دو برنامه روی کامپیوترهای راه دور روال زیر پذیرفتنی است :

**الف**: در برنامه خود از سیستم عامل بخواهید تا شرایط را برای برقراری یک "ارتباط" با کامپیوتری خاص (با آدرس IP مشخص) و برنامه ای خاص روی آن کامپیوتر (با آدرس پورت مشخص) فراهم کند یا اصطلاحاً سوکتی را بگشاید. اگر این کار موفقیت آمیز بود سیستم عامل برای شما یک اشاره گر فایل برمی‌گرداند و در غیر اینصورت طبق معمول مقدار پوچ (NULL) به برنامه شما برخواهد گرداند.

**ب**: در صورت موفقیت آمیز بودن عمل در مرحله قبل ، به همان صورتی که درون یک فایل می‌نویسید یا از آن می‌خوانید ، می‌توانید با توابع send() [یا write() ] و recv() [یا read() ] اقدام به مبادله داده‌ها بنمائید.

**ج**: عملیات مبادله داده‌ها که تمام شد ارتباط را همانند یک فایل معمولی ببندید. (با تابع close())

برای آنکه در برنامه خود همانند فایل یک "اشاره گر ارتباط" را از سیستم عامل طلب کنید تا برایتان مقدمات یک ارتباط را فراهم کند بایستی تابع سیستمی socket()

---

[۱] Scanner
[۲] Pipe

را در برنامه خود صدا بزنید. درصورتی که عمل موفقیت آمیزبود ، یک اشاره‌گر غیر پوچ بر می‌گردد که از آن برای فراخوانی توابع و روالهای بعدی استفاده خواهد شد.

پس از این هرگاه از "سوکت باز" یا مبادله داده‌ها روی سوکت یاد کردیم منظورمان اشاره به یک ارتباط باز یا مبادله اطلاعات بین دو نقطه TSAP[1] روی دو سیستم شبکه کامپیوتری می‌باشد.

دقیقاً همانند فایلها که میتوان همزمان چندین فایل را در یک برنامه واحد باز کرد و روی هر یک از آنها (با استفاده از اشاره گر فایل ) نوشت یا از آنها خواند، در یک برنامه تحت شبکه می‌توان بطور همزمان چندین ارتباط فعال و باز داشت و با مشخصه هر یک از این ارتباطها روی هر کدام از آنها مبادله داده انجام داد.

## ۲) انواع سوکت و مفاهیم آنها

اگر بخواهیم از نظر اهمیت انواع سوکت را معرفی کنیم دو نوع سوکت بیشتر وجود ندارد. (انواع دیگری هم هستند ولی کم اهمیت ترند). این دو نوع سوکت عبارتند از :

- سوکتهای نوع استریم که سوکتهای اتصال گرا[2] نامیده می‌شود.
- سوکتهای نوع دیتاگرام که سوکتهای بدون اتصال[3] نامیده میشود.

اگر عادت به پیش داوری دارید برای تمایز بین مفهوم این دو نوع سوکت، تفاوت بین مفاهیم ارتباط نوع TCP و UDP را مد نظر قرار بدهید. روش ارسال برای سوکتهای نوع استریم همان روش TCP است و بنابراین داده‌ها با رعایت ترتیب و مطمئن با نظارت کافی برخطاهای احتمالی مبادله می‌شوند. سوکتهای نوع دیتاگرام نامطمئن است و هیچگونه تضمینی در ترتیب جریان داده‌ها وجود ندارد.

اکثر خدمات و پروتکلهائی که در لایه چهارم تعریف شده‌اند و در فصول بعدی آنها را بررسی می‌کنیم نیازمند حفظ اعتبار و صحت داده‌ها و همچنین رعایت ترتیب جریان داده‌ها هستند. بعنوان مثال پروتکل انتقال فایل ( FTP )، پروتکل انتقال صفحات ابرمتن (HTTP) یا پروتکل انتقال نامه های الکترونیکی (SMTP) همگی

---

[1] Transport Service Access Point
[2] Connection Oriented
[3] Connectionless

نیازمند برقراری یک ارتباط مطمئن هستند و طبعاً از سوکتهای نوع استریم بهره می‌برند.

همانگونه که قبلاً در مورد پروتکل TCP آموختیم پروتکلی است که داده‌ها را با رعایت ترتیب و خالی از خطا مبادله می‌نماید و پروتکل IP که در لایه زیرین آن واقع است با مسیریابی بسته‌ها روی شبکه سروکار دارد. سوکتهای نوع استریم دقیقاً مبتنی بر پروتکل TCP بوده و طبیعتاً قبل از مبادله داده‌ها باید یک اتصال[۱] به روش دست‌تکانی سه‌مرحله‌ای[۲] برقرار بشود.

سوکتهای نوع دیتاگرام مبتنی بر پروتکل UDP است و بدون نیاز به برقراری هیچ ارتباط و یا اتصال ، داده‌ها مبادله میشوند و بنابراین تضمینی بررسیدن داده‌ها ، صحت داده‌ها و تضمین ترتیب داده‌ها وجود ندارد ولی با تمام این مشکلات باز هم در برخی از کاربردها مثل انتقال صدا و تصویر یا سیستم DNS که قبلاً آنرا بررسی کردیم مورد استفاده قرار می‌گیرد. تنها حسن این روش سرعت انتقال داده‌ها می‌باشد.

در حقیقت شما با استفاده از سوکتها میخواهید یک ابزار برای استفاده از پروتکلهای TCP یا UDP در اختیار داشته باشید.

**" سوکت یک مفهوم انتزاعی از تعریف ارتباط در سطح برنامه‌نویسی خواهد بود و برنامه‌نویس با تعریف سوکت عملاً تمایل خود را برای مبادله داده‌ها به سیستم عامل اعلام کرده و بدون درگیر شدن با جزئیات پروتکل TCP یا UDP از سیستم عامل می‌خواهد تا فضا و منابع مورد نیاز را جهت برقراری یک ارتباط ، ایجاد کند."**

## ۳) مفهوم سرویس دهنده /مشتری[۳]

دربرنامه نویسی شبکه این نکته قابل توجه است که هر ارتباطی دو طرفه می‌باشد یعنی عملاً ارتباط مابین دو پروسه تعریف می‌شود لذا طرفین ارتباط بایستی در لحظه شروع تمایل خود را برای مبادله داده‌ها به سیستم عامل اعلام کرده باشند. در هر ارتباط یکی از طرفین ، شروع کننده ارتباط تلقی می‌شود و طرف مقابل در صورت

---

[۱] Connection
[۲] Tree Way Handshake
[۳] Client/Server

آمادگی این ارتباط را می‌پذیرد. در صورت پذیرش ارتباط، مبادله داده‌ها امکان پذیر خواهد بود. اگر برنامه‌ای را که شروع کننده ارتباط است ، "برنامه مشتری" بنامیم قاعدتاً برنامه‌ای که این ارتباط را می‌پذیرد (و منتظر آن بوده) "سرویس دهنده" نام خواهد گرفت.

**تعریف عمومی: مشتری (Client) پروسه‌ای است که اصولاً نیازمند مقداری اطلاعات است. سرویس دهنده (Server) پروسه ای است که اطلاعاتی را در اختیار دارد و تمایل دارد تا این اطلاعات را به اشتراک بگذارد و منتظر می‌ماند تا یک متقاضی ، واحدی از این اطلاعات را طلب کند و او آنرا تحویل بدهد.**

بعنوان مثال وقتی سخن از سرویس دهنده وب[۱] در میان است در یک عبارت ساده ، منظور سیستمی است که اطلاعاتی را در قالب صفحات وب[۲] در اختیار دارد و در عین حال منتظر است که کسی تقاضای یکی از این صفحات را نموده و او این درخواست را اجابت کرده و داده‌های لازم را در پاسخ به این تقاضا ارسال نماید.

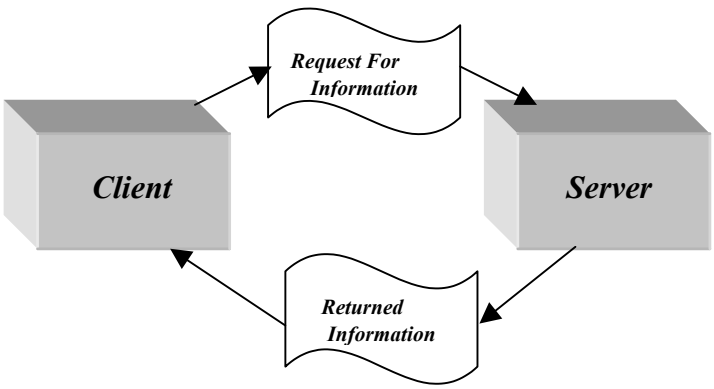
برنامه سمت سرویس دهنده[۳] برنامه‌ای است که روی ماشین سرویس‌دهنده نصب میشود و منتظر است تا تقاضائی مبنی بر برقراری یک ارتباط دریافت کرده و پس از پردازش آن تقاضا ، پاسخ مناسب را ارسال نماید بنابراین در حالت کلی برنامه سرویس دهنده شروع کننده یک ارتباط نیست.

در طرف مقابل برنامه های سمت مشتری[۴] هستند که بنابر نیاز ، اقدام به درخواست اطلاعات می‌کنند. تعداد مشتریها روی ماشینهای متفاوت یا حتی روی یک ماشین می‌تواند متعدد باشد و لیکن معمولاً تعداد سرویس دهنده ها یکی است. (مگر در سیستم های توزیع شده که مورد بحث ما نیستند.) برای مثال جلسه پرسش و پاسخی را در نظر بگیرید که یکنفر صاحب اطلاعات ، پاسخگو و منتظر سوال است – سمت سرویس‌دهنده –. در طرف دیگر تعدادی سوال کننده هستند که مختارند در رابطه با موضوع مورد بحث سوال نمایند – سمت مشتری–.

---

۱ Web Server
۲ Web Page
۳ Server Side
۴ Client Side

به نظر می‌رسد با دقت در مفهوم سرویس‌دهنده/ مشتری متقاعد بشوید که ساختار برنامه‌ای که در سمت سرویس دهنده در حال اجراست با برنامه ای که در سمت مشتری اجرا می‌شود ، متفاوت خواهد بود.




شکل (۱-۷) ارتباط بین سرویس دهنده و مشتری

قبل از آنکه وارد مقوله برنامه نویسی سوکت بشویم بد نیست الگوریتم کل کاری که بایستی در سمت سرویس دهنده و همچنین در سمت مشتری انجام بشود، بررسی نمائیم :

برنامه شما در سمت سرویس دهنده به عملیات زیر نیازمند خواهد بود :

**الف** : یک سوکت را که مشخصه یک ارتباط است ، بوجود بیاورید. تا اینجا فقط به سیستم عامل اعلام کرده‌اید که نیازمند تعریف یک ارتباط هستید. در همین مرحله به سیستم عامل نوع ارتباط درخواستی خود را (TCP یا UDP) معرفی می‌نمائید. این کار در محیط سیستم عامل یونیکس توسط تابع سیستمی ()socket انجام می‌شود.

**ب** : به سوکتی که باز کرده‌اید یک آدرس پورت نسبت بدهید. این کار توسط تابع سیستمی ()bind انجام می‌شود و در حقیقت با این کار به سیستم عامل اعلام می‌کنید که تمام بسته های TCP ( یا UDP ) را که آدرس پورت مقصدشان با شماره مورد نظر شما مطابقت دارد ، به سمت برنامه شما هدایت کند. در حقیقت با این کار خودتان را بعنوان پذیرنده دسته ای از بسته های TCP یا UDP با شماره پورت خاص معرفی کرده‌اید. (دقت کنید که در برنامه سمت سرویس دهنده استفاده از دستور ()bind الزامی است)

**ج** : در مرحله بعد به سیستم عامل اعلام می‌کنید که کارش را برای پذیرش تقاضاهای ارتباط TCP شروع نماید. این کار توسط تابع سیستمی listen() انجام می‌شود و چون ممکن است تعداد تقاضاهای ارتباط متعدد باشد باید حداکثر تعداد ارتباط TCP را که می‌توانید پذیرای آن باشید ، تعیین نمائید چرا که سیستم عامل باید بداند برای پذیرش ارتباطات TCP چقدر فضا و منابع شامل بافر در نظر بگیرد. دقت کنید که اعلام پذیرش تقاضاهای ارتباط به معنای پذیرش داده‌ها نیست بلکه فضای لازم را جهت ارسال و دریافت داده‌ها ایجاد می‌کنید. معمولاً تعیین تعداد ارتباطات TCP که میتواند بطور همزمان پذیرفته شده و به روش اشتراک زمانی[1] پردازش شود ، در اختیار شماست ولی باید این تعداد کمتر از مقداری باشد که سیستم عامل بعنوان حداکثر تعیین کرده است. بازهم یادآوری می‌کنیم که سرویس دهنده می‌تواند بصورت همزمان، چندین ارتباط متفاوت با چندین برنامه روی ماشینهای متفاوت را بصورت باز و فعال داشته باشد. بعنوان یک مقایسه با سیستم فایل تعداد حداکثر ارتباط باز را تعداد فایلی تصور کنید که می‌تواند توسط برنامه شما بطور همزمان باز شود.

**د**: نهایتاً با استفاده از تابع accetp() از سیستم عامل تقاضا کنید یکی از ارتباطات معلق را ( در صورت وجود ) به برنامه شما معرفی کند. تابع accetp() نکات ظریفی دارد که به تفصیل بررسی خواهد شد.

**ه**: از دستورات send() و recv() برای مبادله داده‌ها استفاده نمائید.

**و**: نهایتاً ارتباط را خاتمه بدهید. این کار به دو روش امکان پذیر خواهد بود:

- قطع ارتباط دو طرفه ارسال و دریافت (توسط تابع close() )
- قطع یکطرفه یکی از عملیات ارسال یا دریافت (توسط تابع shutdown() )

در برنامه سمت مشتری بایستی اعمال زیر انجام شود :

**الف** : یک سوکت را که مشخصه یک ارتباط است ، بوجود بیاورید. تا اینجا فقط به سیستم عامل اعلام شده است که نیازمند تعریف یک ارتباط هستید.

**ب** : در مرحله بعد لازم نیست همانند برنامه سرویس دهنده به سوکت خود آدرس پورت نسبت بدهید یعنی لزومی به استفاده از دستور bind() وجود ندارد چرا

---

[1] Time sharing

که برنامه سمت مشتری منتظر تقاضای ارتباط از دیگران نیست بلکه خودش متقاضی برقراری ارتباط با یک سرویس دهنده است. بنابراین در مرحله دوم به محض آنکه نیازمند برقراری ارتباط با یک سرویس دهنده شدید آن تقاضا را با استفاده از تابع سیستمی connect() به سمت آن سرویس دهنده بفرستید.

اگر مراحل دست تکانی سه مرحله ای را در برقراری یک ارتباط TCP بخاطر داشته باشید، دستور connect() عملاً متولی شروع و انجام چنین ارتباطی است.

مجدداً تاکید می‌کنیم از تابع bind() زمانی استفاده می‌شود که پذیرای ارتباطات TCP با شماره پورت خاصی باشید ولی در طرف مشتری چنین کاری لازم نخواهد بود چرا که برنامه سمت مشتری شروع کننده ارتباط است.

اگر عمل connect() موفقیت آمیز بود به معنای موفقیت در برقراری یک ارتباط TCP با سرویس دهنده است و می‌توانید بدون هیچ کار اضافی به ارسال و دریافت داده‌ها اقدام نمائید.

**ج** : از توابع send() , recv() برای ارسال یا دریافت داده‌ها اقدام نمائید.

**د** : ارتباط را با توابع close() یا shutdown() بصورت دوطرفه یا یکطرفه قطع نمائید.

پس از بررسی الگوریتم کلی برنامه های سمت سرویس دهنده و سمت مشتری وقت آن رسیده است که ساختمان داده‌ها و همچنین توابع و روالهای مورد نیاز در برنامه نویسی را با دقت بیشتری مورد بررسی قرار بدهیم.

## ۴) ساختمان داده‌های مورد نیاز در برنامه نویسی مبتنی بر سوکت

برای آغاز برنامه نویسی بهترین کار آنست که متغیرها و انواع ساختمان داده مورد نیاز در برنامه نویسی سوکت، تحت بررسی قرار بگیرد. (تمام قطعه کدها با C هستند )

اولین نوع داده ، "مشخصه سوکت[1]" است که همانند اشاره‌گر فایل ، برای ارجاع به یک ارتباط باز مورد استفاده قرار می‌گیرد و یک عدد صحیح دوبایتی است یعنی با تعریف زیر ، متغیر **a** می‌تواند مشخصه یک سوکت باشد:

```
int    a;
```

[1] Socket Descriptor

دومین نوع داده برای برقراری ارتباط ، یک استراکچر است که آدرس پورت پروسه و همچنین آدرس IP ماشین طرف ارتباط را درخود نگه می‌دارد. فعلاً در تعریفی ساده ساختار آن بصورت زیر است:

```
struct sockaddr {
        unsigned short sa_family;       /* address family, AF_xxxx */
        char sa_data[14];               /* 14 bytes of protocol address */
      };
```

**sa_family** : خانواده یا نوع سوکت را مشخص می‌کند. در حقیقت این گزینه تعیین می‌کند که سوکت مورد نظر را در چه شبکه و روی چه پروتکلی بکار خواهید گرفت؛ لذا در سیستمی که با پروتکلهای متفاوت و سوکتهای متفاوت سروکار دارد ، باید نوع سوکت درخواستی را تعیین کنید. فعلاً در کل این فصل که بحث ما شبکه اینترنت با پروتکل TCP/IP است خانواده سوکت را با ثابت AF_INET مشخص می‌کنیم. در مورد شبکه های دیگر مثل Appletalk این گزینه متفاوت خواهد بود.

**sa_data** : این چهارده بایت مجموعه ای است از آدرس پورت ، آدرس IP و قسمتی اضافی که باید با صفر پر شود و دلیل آنرا بعداً اشاره می‌کنیم.

همانگونه که اشاره شد این استراکچر بایستی آدرس پورت و آدرس IP را نگه دارد ولی در تعریف بالا چنین فیلدهائی مشاهده نمی‌شود. برای سادگی در برنامه نویسی ، استراکچر دیگری معرفی میشود که دقیقاً معادل استراکچر قبلی است ولی تعریف متفاوتی دارد و شما می‌توانید از هر کدام به دلخواه بهره بگیرید:

```
struct sockaddr_in {
        short int sin_family;           /* Address family */
        unsigned short int sin_port;  /* Port number */
        struct in_addr sin_addr;      /* Internet address */
        unsigned char sin_zero[8];   /* Same size as struct sockaddr */
        };
```

**sin_family** : همانند ساختار قبلی خانواده سوکت را تعیین می‌کند و برای شبکه اینترنت بایستی مقدار ثابت AF_INET داشته باشد.
**sin_port** : این فیلد دو بایتی ، آدرس پورت پروسه مورد نظر را مشخص می‌نماید.

**in_addr** : آدرس IP ماشین مورد نظر را مشخص می‌کند. این فیلد خودش یک استراکچر است که در ادامه تعریف خواهد شد ، فقط بدانید که کلاً عددی صحیح ، بدون علامت و چهاربایتی است.

**sin_zero[8]** : این هشت بایت در کاربردهای مهندسی اینترنت کلاً باید مقدار صفر داشته باشد. دلیل وجود این فیلد ، آنست که مفهوم سوکت برای تمام شبکه‌ها با پروتکلهای متفاوت ، بصورت معادل استفاده شده و بنابراین استراکچر فوق باید بگونه ای تعریف شود که برای تمام پروتکلهای شبکه قابل استفاده باشد. در شبکه اینترنت فعلاً آدرس IP چهاربایتی و آدرس پورت دوبایتی است در حالی که در برخی دیگر از شبکه ها طول آدرس بیشتر است. بنابراین هنگامی که از استراکچر فوق در کاربردهای برنامه نویسی شبکه اینترنت بهره می‌گیرید این هشت بایت اضافی است ولی حتماً باید با توابعی مثل (memset تماماً صفر شود.

دقت نمایید که دو استراکچر قبلی دقیقاً معادلند و می‌توان در فراخوانی توابع ، هر کدام از آنها را با تکنیک "تطابق نوع[1]" بجای دیگری بکار برد ولی در مجموع استفاده از تعریف دوم راحت ترخواهدبود.

در تعریف استراکچر دوم یک استراکچر دیگر بنام in_addr تعریف شده که ساختار آن بصورت زیر است:

```
/* Internet IP address (a structure for historical reasons) */
 struct in_addr {
                unsigned long s_addr;
               };
```

این فیلد چهاربایتی برای نگهداری آدرس IP کاربرد دارد و تعریف آن بصورت فوق کمی عجیب به نظر می‌رسد چرا که می‌توانستیم در استراکچر قبلی بطور مستقیم

[1] Casting

آنرا unsigned long معرفی کنیم ولی بهر حال بصورت بالا تعریف شده است. برای مقداردهی به فیلدهای بالا می‌توانید از هر روشی که دلخواه شماست استفاده کنید ولیکن توابعی ساده برای این کار وجود دارند که در ادامه معرفی خواهند شد.

## ۵) مشکلات ماشینها از لحاظ ساختار ذخیره سازی کلمات در حافظه

در گذشته تفاوت ماشینهای نوع BE[۱] و نوع LE[۲] را بررسی کردیم و اشاره شد که در پروتکل TCP/IP ترتیب بایتها بصورت BE توافق شده است لذا وقتی قرار است برنامه شما روی ماشینی که ساختار LE دارد نصب شود ترتیب بایتهای ارسالی روی شبکه بهم خواهد خورد. بعنوان مثال وقتی که روی ماشینی از نوع LE دستور زیر اجرا می‌شود:

```
struct sockaddr_in   as;
as.sin_port=0xB459;
```

چون بایت کم ارزش اول ذخیره می‌شود و بعد از آن بایت پرارزش قرار میگیرد لذا پس از قرار گرفتن این دو بایت در بسته TCP آدرس پورت بصورت زیر (و قطعاً اشتباه) تنظیم خواهد شد:

| 59 | B4 |
|---|---|

بنابراین وقتی قرار است برنامه نویس مقداری را درون فیلدی قرار بدهد که دو بایتی یا چهار بایتی است بایستی نگران نوع ماشین و ترتیب بایتها باشد. بهمین دلیل معرفی توابع زیر بعنوان ابزار کار برنامه نویس شبکه اینترنت ضروری است:

**htons()** : تابع تبدیل کلمات دوبایتی به حالت BE
**htonl()** : تابع تبدیل کلمات چهاربایتی به حالت BE
**ntohs()** : تابع تبدیل کلمات دوبایتی از BE به حالت فعلی ماشین
**ntohl()** : تابع تبدیل کلمات چهاربایتی از BE به حالت فعلی ماشین

---

[۱] Big Endian
[۲] Little Endian

برنامه نویس لازم است ساختار ماشین مورد استفاده جهت نصب نهایی برنامه اش را بداند تا در صورت LE بودن حتماً قبل از قرار دادن مقادیر در فیلدهای دوبایتی یا چهاربایتی از توابع فوق استفاده کند.

تذکر: فقط وقتی از توابع فوق استفاده می‌شود که نهایتاً فیلد مورد نظر در بسته TCP یا IP تنظیم شود. بعنوان مثال در استراکچر sock_addr_in در فیلد sin_family مقدار AF_INET که مقداری ثابت است قرار می‌گیرد و این فیلد فقط برای سیستم عامل تعریف شده و روی شبکه منتقل نخواهد شد لذا برای مقدار دهی به این فیلد لازم نیست از توابع فوق استفاده نمائیم. در ادامه با مثالهائی که خواهیم داشت با موارد استفاده توابع فوق آشنا می‌شویم.

### ۱-۵) مشکلات تنظیم آدرس IP درون فیلد آدرس

در مبحث آدرسهای IP آموختید که آدرسهای IP درقالب چهار فیلد هشت‌تایی[۱] ده دهی نوشته می‌شوند :

192.140.11.211

در حالی که در استراکچر sock_addr_in فیلد آدرس IP عددی است چهاربایتی که با یک عدد از نوع long پر می‌شود. بنابراین دو تابع زیر جهت انتساب آدرسهای IP با ساختار فوق الذکر کاربرد دارد:

**تابع inet_addr()** : این تابع یک رشته کاراکتری بفرم "187.121.11.44" را گرفته و به یک عدد چهاربایتی با قالب BE تبدیل می‌کند. مثال:

```
struct sockaddr_in  ina;
ina.sin_addr.s_addr=inet_addr("130.421.5.10")
```

در مثال بالا آدرس IP رشته‌ای است و پس از تبدیل به عددی چهاربایتی در قالب BE در فیلد مربوطه قرار می‌گیرد.

**تابع inet_ntoa()** : این تابع عکس عمل تابع قبلی را انجام می‌دهد یعنی یک آدرس چهاربایتی درقالب BE را گرفته و آنرا بصورت یک رشتهٔ کاراکتری که آدرس IP را

---

[۱] Octet

بصورت نقطه‌دار تعریف کرده ، تبدیل می‌نماید. پارامتر ورودی تابع فوق از نوع struct in_addr و خروجی آن نوع رشته ای است. به مثال زیر دقت کنید:

```
printf("%s",inet_ntoa(ina.sin_addr));
```

در مثال فوق محتوای آدرس IP بصورت رشته‌ای نقطه دار و در مبنای ده روی خروجی چاپ خواهد شد. مثلاً خروجی به فرم زیر است:

130.141.5.10

در بخشهای آتی چگونگی تبدیل آدرسهای حوزه بفرم www.ibm.com را به آدرس IP در محیط برنامه نویسی توضیح خواهیم داد. قبل از آن باید توابع لازم برای تعریف و برقراری ارتباط تعریف شوند.

## ۶) توابع مورد استفاده در برنامه سرویس دهنده (مبتنی بر پروتکل TCP)

### ۶-۱) تابع socket()

فرم کلی این تابع بصورت زیر است:

```
#include <sys/types.h>
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

**domain** : این پارامترنشان‌دهندهٔ خانواده سوکت است و به نحوی که قبلاً اشاره شد در برنامه‌نویسی شبکهٔ اینترنت ، با مقدار ثابت AF_INET تنظیم می‌شود.

**type** : با این پارامتر نوع سوکت دلخواهتان را اعلام می‌کنید که می‌تواند نوع استریم یا از نوع دیتاگرام باشد. اگر سوکت دلخواهتان نوع استریم بود در فیلد type مقدار ثابت SOCK_STREAM قرار بدهید و اگر نوع دیتاگرام خواستید درآن مقدار SOCK_DGRAM تنظیم کنید.

**protocol** : در این فیلد شماره شناسائی پروتکل مورد نظرتان را تنظیم می‌کنید که برای کاربردهای شبکه اینترنت همیشه مقدار آن صفر است.

مقادیری که در فیلدهای اول و سوم قرار می‌دهید در برنامه نویسی تحت شبکه اینترنت همیشه ثابت خواهند بود.

مقدار بازگشتی توسط تابع socket() همان مشخصه سوکت است که از آن برای توابع بعدی استفاده خواهد شد ( دقیقاً مثل اشاره گر یک فایل ) لذا مشخصه سوکت بایستی تا زمانی که ارتباط خاتمه می‌یابد بدقت نگهداری شود.

اگر مقدار برگشتی تابع socket() ، ۱- باشد عمل موفقیت آمیز نبوده و روند کار باید متوقف شود و شما بعنوان برنامه نویس موظفید حتماً خروجی این تابع را بررسی کنید چرا که عملیات بقیه توابع که در ادامه معرفی خواهند شد به خروجی همین تابع بستگی دارد.

وقتی مقدار برگشتی تابع socket() مقدار ۱- باشد متغیر سراسری errno شماره خطای رخداده می‌باشد. برای پردازش شماره خطا تابع سیستمی perror() می‌تواند استفاده شود که روش بکارگیری آن در مثالها آمده است. این دو متغیر و تابع نیاز به تعریف ندارد و سیستمی هستند.

### ۲-۶) تابع bind()

وقتی سیستم عامل برای شما یک سوکت باز کرد در حقیقت شما فقط سنگ بنای یک ارتباط را بنا نهاده‌اید ولی هنوز هیچ کاری برای مبادله داده‌ها انجام نشده است. تابع bind() که معمولاً در برنامه سمت سرویس دهنده معنا می‌**یابد "عملی است جهت نسبت دادن آدرس پورت به یک سوکت باز شده**". این تعریف احتمالاً ابهام دارد پس به تعریف ساده زیر دقت کنید:

از طریق تابع bind() از سیستم عامل خواهش می‌کنید که تمام بسته های TCP یا UDP و همچنین تقاضاهای ارتباط با شماره پورت خاص را به سمت برنامه شما هدایت نماید.

بعنوان مثال وقتی گفته می‌شود که پروتکل HTTP به پورت 80 گوش می‌دهد به این معناست که برنامهٔ سرویس‌دهنده ، تمام بسته های TCP را که وارد ماشین محلی می‌شوند و شماره پورت مقصد آنها 80 است ، تحویل می‌گیرد و پردازش می‌نماید.

فرم کلی تابع bind() بصورت زیر است:

```
#include <sys/types.h>
#include <sys/socket.h>

int bind(int sockfd, struct sockaddr *my_addr, int addrlen);
```

**sockfd** : همان مشخصه سوکتی است که قبلاً با استفاده از تابع socket() بازکرده‌اید. در حقیقت شما می‌خواهید به سوکت باز شده یک آدرس پورت نسبت بدهید.
**my_addr** : یک استراکچر که خانواده سوکت، آدرس پورت و آدرس IP ماشین محلی را در خود دارد. ساختار این استراکچر قبلاً تعریف شد.
**addr_len** : طول استراکچر my_addr بر حسب بایت

برای آشنایی با چگونگی استفاده از توابع فوق به قطعه کد زیر دقت کنید:

```
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>

#define MYPORT 3490

main()
   {
    int sockfd;
    struct sockaddr_in my_addr;

    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0))!= NULL){
      my_addr.sin_family = AF_INET;         /* host byte order */
      my_addr.sin_port = htons(MYPORT);  /* short, network byte order */
      my_addr.sin_addr.s_addr = inet_addr("132.241.5.10");
      bzero(&(my_addr.sin_zero), 8);          /* zero the rest of the struct */

     if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr))!=-1){
       .
       .
       .
```

در مورد تابع bind() نکاتی وجود دارد که اشاره به آنها خالی از لطف نیست:

الف: در محیط یونیکس اگر فیلد آدرس پورت با مقدار صفر تنظیم شود آنگاه سیستم عامل در بین آدرسهای پورت از شماره ۱۰۲٤ تا ٦٥٥٣٥ ، یک شمارهٔ تصادفی انتخاب کرده و آنرا بعنوان شماره پورت در نظر می‌گیرد.

ب: برنامه کاربردی شما نباید شماره پورتی را بر گزیند که بین صفر تا ۱۰۲۳ باشد چرا که این شماره پورتها برای سرویس دهنده های استاندارد و سرویس دهنده

هـای یونیکـس رزرو شـده اسـت و سیسـتم عـامل اجـازه اسـتفاده از ایـن شـماره پورتـها را به برنامه هـای کـاربران نخواهـد داد.

ج: در محیـط یونیکـس اگـر فیلـد آدرس IP را بـا مقـدار ثـابت INADDR_ANY تنظیـم کنید ، آنگـاه سیسـتم عـامل بصـورت خودکـار آدرس IP ماشـین محلـی شـما را اسـتخراج و در آن قـرار خواهـد داد.

د: نکتـه ای کـه ممکـن اسـت بـر آن خـرده بگـیرید آن اسـت کـه چـرا در مقداردهی فیلدهای بـالا سـعی نکردیـم بـا بـهره‌گـیری از توابـع ()htons آن را بـه حـالت BE تبدیل کنیـم در حـالی کـه چنیـن کـاری لازم می‌باشـد. دلیـل آن بسـیار سـاده اسـت: هـر دو مقدار صفر دارند و حالت صفـر نیـازی بـه تبدیـل نـدارد.

ه: اگر شماره پورتی کـه انتخـاب می‌کنیـد برنامـه دیگـری قبـل از شـما بـرای خـود رزرو کرده باشـد یعنـی آنـرا در برنامـه خـود بـه سـوکتی ()bind کـرده باشـد آنگـاه عمـل ()bind موفقیت آمیز نبـوده ومقـدار (۱-) بـه برنامـه شـما بـاز خواهـد گشـت. بـرای پـردازش نـوع خطـا ، متغـیر سراسـری errno شـماره خطـا و تـابع ()perror مشـخصات خطـا را بـر می‌گردانـد.

### ۳-۶) تابع ()listen

این تـابع فقـط در برنامـه سـرویس دهنـده معنـا می‌یـابد و در یـک عبـارت سـاده اعـلام به سیستم عامل برای پذیـرش تقاضاهـای ارتبـاط TCP اسـت. بـه عبـارت بـهتر توسـط ایـن تابع به سیستم عـامل اعـلام می‌کنیـد کـه از ایـن لحظـه بـه بعـد (یعنـی زمـان اجـرای تـابع) تقاضاهـای ارتبـاط TCP ماشـینهای راه دور بـا شـماره پـورت مـورد نظرتـان را بـه صـف کرده و منتظـر نگـه دارد.

با توجه بـه آنکـه ممکـن اسـت پـس از راه انـدازی برنامـه سـرویس‌دهنده ، در لحظـاتی چندیـن پروسـه متفـاوت بطـور همزمـان تقاضـای برقـراری ارتبـاط TCP بـه یـک آدرس پورت بدهنـد بنـابراین سیسـتم عـامل بـاید بدانـد کـه حداکـثر چنـد تـا از آنـها را بپذیـرد و ارتبـاط آنـها را بـه روش دسـت تکـانی سـه مرحلـه ای برقـرار نمـوده و آنـها را در صـف سـرویس دهـی قـرار بدهـد. توسـط تـابع ()listen بـاید بـه سیسـتم عـامل اعـلام شـود کـه حداکثر تعداد ارتباطـات فعـال و بـاز روی یـک شـماره پـورت خـاص چنـد تـا باشـد. فـرم کلی تابع بصورت زیـر اسـت:

```
int listen(int sockfd, int backlog);
```

**sockfd** : همان مشخصه سوکت است که در ابتدا آنرا ایجاد کرده‌اید.
**backlog** : حداکثر تعداد ارتباطات معلّق و به صف شدهٔ منتظر. در بسیاری از سیستمها مقدار backlog به ۲۰ محدود شده است.

همانند توابع قبلی در صورت بروز خطا مقدار برگشتی این تابع ۱- خواهد بود و متغیر errno شماره خطای رخداده می‌باشد.

### ۴-۶) تابع accept()

این تابع اندکی مرموز به نظر می‌رسد و بایستی به مفهوم آن دقت شود :
پس از آنکه تابع listen() اجرا شد تقاضای ارتباط TCP پروسه های روی ماشینهای راه دور ( درصورت وجود ) پذیرفته ، به صف شده و معلق نگاه داشته میشود. وقتی که تابع accept() اجرا می‌شود در حقیقت برنامه شما از سیستم عامل تقاضا می‌کند که از بین تقاضاهای به صف شده یکی را انتخاب کرده و آنرا با مشخصات پروسه طرف مقابل تحویل برنامه بدهد. بنابراین برنامه باید از بین ارتباطات معلق یکی را به حضور بطلبد تا عملیات لازم را انجام بدهد. بهمین دلیل سیستم عامل یک مشخصه سوکت جدید ایجاد کرده و آنرا به برنامه بر می‌گرداند. در اینجا شما یک سوکت جدید دارید. مشخصه سوکت اول که توسط تابع socket() بدست آمده و مشخصه سوکت دوم که با تابع accept() به برنامه شما برگشته است. تفاوت این دو سوکت در چیست؟

الف: از سوکت اول برای پذیرش یکی از ارتباطات معلق در دستور accept() استفاده می‌کنید. در حقیقت این سوکت مشخصه کل ارتباطات به صف شدهٔ منتظر میباشد.

ب: از سوکت دوم برای دریافت و ارسال اطلاعات روی یکی از ارتباطات معلق استفاده می‌کنید. این سوکت مشخصه یکی از ارتباطات به صف شده می‌باشد.

فرم کلی تابع به صورت زیر است :

```
#include <sys/socket.h>

int accept(int sockfd, void *addr, int *addrlen);
```

**sockfd** : مشخصه سوکت است که در ابتدا با تابع socket() بدست آمده است.

**addr** : اشاره‌گر بـه اسـتراکچری اسـت کـه شـما آنـرا بعنـوان پـارامتر بـه ایـن تـابع ارسـال می‌کنیـد تـا سیسـتم عـامل پـس از پذیـرش یـک ارتبـاط معلـق آدرس پـورت و آدرس IP طـرف مقـابل ارتبـاط را در آن بـه برنامـه شـما بـرگردانـد. سـاختار ایـن اسـتراکچر قبـلاً معرفی شـد.

**addrlen** : طول اسـتراکچر addr بـر حسـب بـایت

مقـدار بـرگشـتی ایـن تـابع یـک مشـخصه سـوکت اسـت کـه در روالـهای بعـدی مـورد استفاده قرار می‌گیرد. اگـر مقـدار بـرگشـتی (۱-) باشـد خطـائی رخـداده اسـت کـه شـماره آن خطا در متغـیر سراسـری errno قـابل بررسـی اسـت.

مثال ناتمام زیر برای روشن شدن کلیـت کـار بسـیار سـودمند خواهـد بـود:

```
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>

#define MYPORT 3490    /* the port users will be connecting to */
#define BACKLOG 10     /* how many pending connections queue will hold */

main()
{
   int sockfd, new_fd;  /* listen on sock_fd, new connection on new_fd */
   struct sockaddr_in my_addr;    /* my address information */
   struct sockaddr_in their_addr; /* connector's address information */
   int sin_size;

   if ((sockfd = socket(AF_INET, SOCK_STREAM, 0))!= NULL){

   my_addr.sin_family = AF_INET;         /* host byte order */
   my_addr.sin_port = htons(MYPORT);     /* short, network byte order */
   my_addr.sin_addr.s_addr = INADDR_ANY; /* auto-fill with my IP */
   bzero(&(my_addr.sin_zero), 8);        /* zero the rest of the struct */

   if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr))!=-1){

   listen(sockfd, BACKLOG);

   sin_size = sizeof(struct sockaddr_in);
   new_fd = accept(sockfd, &their_addr, &sin_size);
   .
   .
   .
```

بار دیگر تأکید می‌کنیم که برای ارسال یا دریافت داده‌ها بایستی از سوکت جدیدی که مشخصه آن توسط تابع ()accept برمی‌گردد ، استفاده کنید.

### ۶-۵) توابع send() و recv()

این دو تابع در برنامه سمت سرویس دهنده و برنامه سمت مشتری قابل استفاده بوده و برای مبادله داده‌ها کاربرد دارند. فرم کلی دو تابع به صورت زیر است :

```
int send(int sockfd, const void *msg, int len, int flags);

int recv(int sockfd, void *buf, int len, unsigned int flags);
```

**sockfd** : مشخصه سوکتی که از تابع ()accept بدست آمده است.
**msg** : محلی در حافظه (مثل آرایه یا استراکچر) که داده‌های ارسالی از آنجا استخراج شده و درون فیلد داده[1] از یک بسته TCP قرار گرفته و ارسال می‌شوند.
**len** : طول داده‌های ارسالی یا دریافتی بر حسب بایت
**flag** : برای پرهیز از پیچیدگی بحث در این مورد توضیح نمی‌دهیم. فقط در آن صفر بگذارید.
**buf** : این پارامتر در تابع ()recv آدرس محلی در حافظه است که داده‌های دریافتی در آنجا قرار گرفته و به برنامه باز گردانده می‌شود.

مقدار برگشتی این دو تابع در صورت بروز هر گونه خطا ۱- خواهد بود ولی در صورت برگشت یک عدد مثبت ، تعداد بایتهای ارسالی یا دریافتی را بر حسب بایت مشخص می‌کند. دقت کنید که ممکن است تعداد بایتهای ارسالی یا دریافتی با تعدادی که در متغیر len تقاضا داده‌اید یکسان نباشد. بعنوان مثال فرض کنید شما در متغیر len مقدار ۱۰۰۰ قرار داده‌اید ولی مقداری که تابع برگردانده است ۲۰۰ باشد. در این صورت ۸۰۰ بایت از کل داده‌های ارسالی (یا دریافتی) باقی مانده است که برنامه شما باید تکلیف آنها را مشخص کند.

---

[1] Payload

**توصیه** : در هر مرحله سعی کنید حجم داده‌هائی که توسط تابع send() ارسال می‌کنید حول و حوش یک کیلو بایت باشد.
**نکته** : توابع send() , recv() فقط برای ارسال و دریافت روی سوکتهای نوع استریم کاربرد دارد ولی اگر می‌خواهید به روش UDP و با سوکتهای دیتاگرام داده‌هایتان را ارسال کنید اندکی صبر کنید؛

### ۶-۶) توابع close() و shutdown()

تا زمانی که نیاز داشتید میتوانید یک ارتباط را باز نگه‌داشته و داده ارسال یا دریافت نمائید ولیکن همانند فایلها هرگاه نیازتان برطرف شد باید ارتباط را ببندید.
فرم کلی تابع close() بصورت زیر است :

```
close(int sockfd);
```

**sockfd** : مشخصه سوکت مورد نظر. این سوکت همان مشخصه ای است که تابع accept() برگردانده است. دقت کنید که اگر sockfd مشخصه ای باشد که توسط تابع socket() برگشته است تمام ارتباطات معلق و منتظر نیز بسته خواهد شد.

ارتباطی که توسط تابع close() بسته می‌شود دیگر برای ارسال و دریافت قابل استفاده نخواهد بود.

هر گاه سوکتی را ببندید در حقیقت یکی از ارتباطات TCP را بسته اید و سیستم عامل می‌تواند بجای آن تقاضای ارتباط دیگری را قبول کرده ، برای پردازش به صف ارتباطات معلق اضافه کند.

راه دیگر بستن یک سوکت تابع shutdown() می‌باشد که فرم کلی آن بصورت زیر است :

```
int shutdown(int sockfd, int how);
```

**sockfd** : مشخصه سوکت مورد نظر
**how** : روش بستن سوکت که یکی از سه مقدار زیر را می‌پذیرد:

- مقدار صفر: دریافت داده را غیر ممکن می‌سازد ولی سوکت برای ارسال داده ، همچنان باز است. سیستم عامل بافر ورودی[۱] مربوط به آن سوکت را آزاد می‌کند.
- مقدار ۱ : ارسال داده را غیر ممکن می‌سازد در حالی که سوکت برای دریافت داده‌ها همچنان باز است. سیستم عامل بافر خروجی[۲] مربوط به آن سوکت را آزاد می‌کند.

- مقدار ۲: ارسال و دریافت را غیر ممکن کرده سوکت کاملاً بسته می‌شود. این حالت دقیقاً همانند تابع close() عمل می‌نماید.

همانند توابع قبلی در صورت بروز خطا مقدار برگشتی این توابع ۱- خواهد بود و متغیر سراسری errno شماره خطا را برای پردازش مشخص می‌کند.

## ۷) توابع مورد استفاده در برنامه مشتری ( مبتنی بر پروتکل TCP )

تا اینجا توابعی که معرفی شدند توابع پایه‌ای بودند که در سمت سرویس دهنده به نحوی استفاده می‌شوند. حال باید ببینیم در سمت مشتری چه توابعی مورد استفاده قرار می‌گیرند:

الف : ابتدا دقیقاً مانند برنامه سرویس دهنده یک سوکت بوجود بیاورید. برای اینکار از تابع socket() که در بخش قبلی معرفی شد استفاده کنید. تا اینجا هیچ تفاوتی برای بکارگیری این تابع در سمت سرویس دهنده و سمت مشتری وجود ندارد.

ب : در هنگام نیاز مستقیماً تقاضای برقراری ارتباط را به سمت سرویس دهنده بفرستید و آنقدر منتظر شوید تا این تقاضا پذیرفته شود. این عمل توسط تابع connect() انجام می‌شود که در ادامه توضیح داده خواهد شد.

ج – از توابع send() و recv() برای ارسال و دریافت داده‌ها استفاده کنید.

د– نهایتاً ارتباط ایجاد شده را توسط تابع close() یا shutdown() ببندید.

---

[۱] Inbound buffer
[۲] Outbound buffer

### ۱-۷) تابع connect()

برای برقراری ارتباط با یک سرویس دهنده از تابع connect() استفاده می‌شود و در صورتی که برنامه سرویس دهنده روی ماشین مورد نظر اجرا شده باشد و توابع listen() و accept() در برنامه فراخوانی شده باشند آنگاه نتیجه تابع connect() موفقیت آمیز خواهد بود.

فرم کلی تابع connect() به صورت زیر است :

```
#include <sys/types.h>
#include <sys/socket.h>

int connect(int sockfd, struct sockaddr *serv_addr, int addrlen);
```

**sockfd** : مشخصه سوکتی است که با فراخوانی تابع socket() بدست آمده است.
**serv_addr** : استراکچری از نوع sockaddr است که قبلاً معرفی شد. در این استراکچر آدرس IP ماشین مقصد و آدرس پورت برنامه مقصد تعیین خواهد شد.
**addrlen** : اندازه استراکچر قبلی را بر حسب بایت معرفی می‌کند و می‌توان براحتی در این پارامترمقدار sizeof(struct sockaddr) قرار داد.

به این نکته دقت کنید که شما آدرس پورت خودتان را تنظیم نمی‌کنید بلکه سیستم عامل بطور خودکار یک شماره پورت تصادفی برای شما انتخاب می‌کند و مقدار این شماره برای برنامه سمت مشتری اصلاً مهم نیست چرا که وقتی شما به یک سرویس دهنده متصل می‌شوید و سرویس دهنده این تقاضا را می‌پذیرد پاسخ سرویس دهنده به همان آدرس پورتی خواهد بود که سیستم عامل برای سوکت انتخاب کرده است. در حقیقت برنامه شما بعنوان شروع کنندهٔ ارتباط ، آدرس پورت خود را نیز به طرف مقابل اعلام می‌کند. در مقابل آدرس پورت برنامه سرویس دهنده قطعاً باید ثابت و مشخص باشد تا برنامه مشتریها بتوانند ارتباط را شروع نمایند.
در صورت عدم موفقیت در برقراری یک ارتباط TCP مقدار برگشتی این تابع ۱- خواهد بود و متغیر errno شماره خطای رخداده می‌باشد.

مثال ناتمام زیر تا حدودی این دیدگاه را به شما عرضه می‌کند:

```
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
```

```
#define DEST_IP  "132.241.5.10"
#define DEST_PORT 23

main()  {

    int sockfd;
    struct sockaddr_in dest_addr;   /* will hold the destination addr */

    if (( sockfd = socket(AF_INET, SOCK_STREAM, 0))!=NULL) {

     dest_addr.sin_family = AF_INET;             /* host byte order */
     dest_addr.sin_port = htons(DEST_PORT);  /* short, network byte order */
     dest_addr.sin_addr.s_addr = inet_addr(DEST_IP);
     bzero(&(dest_addr.sin_zero), 8);            /* zero the rest of the struct */

   if ((connect(sockfd, (struct sockaddr *)&dest_addr, sizeof(struct sockaddr)))!=-1) {
     .
     .
     .
```

## ۸) ارسال و دریافت به روش UDP با سوکتهای دیتاگرام

توابع ارسال ، دریافت و پذیرش برای سوکتهای نوع استریم کاربرد دارد. حال باید دید که به چه صورت می‌توان ارسال و دریافت را به روش UDP روی سوکتهای نوع دیتاگرام انجام داد.

- **برنامه سمت سرویس دهنده**

الف : یک سوکت از نوع دیتاگرام ایجاد کنید. این کار با فراخوانی تابع socket() با پارامتر SOCK_DGRAM انجام می‌شود.

ب : به سوکت ایجاد شده آدرس پورت مورد نظرتان را نسبت بدهید. (با تابع bind() )

ج : بدون هیچ کار اضافی میتوانید منتظر دریافت داده‌ها بشوید. (تا موقعی که داده‌ای دریافت نشود ارسال معنی نمی‌دهد.) وقتی داده‌ای دریافت و پردازش شد آدرس برنامه مبدا (آدرس IP و پورت) مشخص شده و ارسال امکان پذیر خواهد بود.

ارسال و دریافت روی سوکتهای نوع دیتاگرام بوسیله توابع recvfrom() و sendto() انجام می‌شود.

د: نهایتاً سوکت ایجاد شده را ببندید.

- **برنامه سمت مشـــتری**

الـف: یـک سـوکت از نـوع دیتـــاگـــرام ایجـــاد کنیـــد. ( بـــا تـــابع ()socket و پـــارامتر SOCK_DGRAM )

ب: هر گاه نیـاز شـد بـدون هیـچ کـار اضـافی داده‌هایتـان را بـه سـمت سـرویس دهنـده ارسـال نمـایید. تـا وقتـی کـه بـه سـمت سـرویس دهنـده ارسـالی نداشـته باشـید، دریـــافت داده‌ها معنا نمی‌دهد چرا کـه شـما بـرای سـرویس دهنـده شـناخته شـده نیسـتید مگـر اینکـه داده‌ای را ارسال نمائید. ارسال و دریافت را تـا زمـانی کـه نیـاز اسـت انجـام بدهیـد.

ج: سوکت ایجـاد شـده را ببندیـد.

فرم کلی تابع ارسال داده مبتنی بر ســوکتهای دیتـاگـرام بصـورت زیـر اسـت:

```
int sendto(int sockfd, const void *msg, int len, unsigned int flags, const struct sockaddr *to,
        int tolen);
```

**sockfd** : مشخصه سوکت دیتاگرام کـه بـا تـابع ()socket بوجـود آمـده اسـت.

**msg** : آدرس محـل قـرارگرفتـن پیـام در حافظـه کـه داده‌هـــای ارســـالی بایســـتی از آنجـــا استخراج شده و درون یـک بسـته UDP قـرار گرفتـه و ارسـال شـود.

**len** : طول پیام ارسالی بـــر حسـب بـایت

**flags** : برای پرهیز از پیچیدگی بحـث فعـلاً آنـرا صفـر در نظـر بگـیرید.

**to** : اسـتراکچری از نـوع sockaddr کـــه قبـــلاً ســـاختار آنـــرا مشـــخص کردیـــم. در ایـــن اسـتراکچر بـاید آدرس IP مربـوط بـه ماشـین مقصـد و همچنیـن شـماره پـــورت ســـرویس دهنده تنظیم شـــود.

**tolen** : طـول اسـتراکچرsockaddr اسـت کـــه بـــه ســـادگـــی می‌توانیـــد آنـــرا بـــه مقـــدار sizeof(struct sockaddr) تنظیـم نمـــایید.

مقـدار بـرگشـتی ایـن تـابع همـانند تـابع ()send تعـداد بـایتی اسـت کـه سیسـتم عـــامل موفق بـــه ارسـال آن شـده اسـت. دقـت کنیـد کـه اگـر مقـدار بـرگشـتی (1-) باشـد خطـائی بروز کرده که می‌توانیـد شـماره خطـا را در متغـیر سراسـری errno بررسـی نمـائید. بـاز هـم تکرار می‌کنیم دلیلی ندارد تعداد بـایتی کـه تقاضـای ارسـال آنـها را داده‌ایـد بـا تعـداد بـایتی که ارسـال شـده یکـی باشـد. بنـابراین حتمـاً ایـن مـورد را در برنامـه خـود بررسـی کـرده و همچنین تقاضاهای ارسال در هر مرحلـه را نزدیـک یـک کیلـو بـایت در نظـر بگـیرید.

فرم کلی تابع دریافت داده مبتنی بر سوکتهای دیتاگرام بصورت زیر است:

```
int recvfrom(int sockfd, void *buf, int len, unsigned int flags,struct sockaddr *from,
        int  *fromlen);
```

**sockfd** : مشخصه سوکت دیتاگرام که با تابع socket() بوجود آمده است.
**buf** : آدرس محلی از حافظه که سیستم عامل داده‌های دریافتی را در آن محل قرار خواهد داد.
**len** : طول پیامی که باید دریافت شود ( بر حسب بایت)
**from** : استراکچری است از نوع sockaddr که قبلاً ساختار آن بررسی شد و سیستم عامل آنرا با مشخصات آدرس IP و آدرس پورت برنامه مبداء تنظیم و به برنامه شما برمی‌گرداند.
**flag** : آنرا به صفر تنظیم کنید.
**len** : طول استراکچری است که سیستم عامل آنرا برگردانده است.

مقدار برگشتی این تابع نیز تعداد بایتی است که دریافت شده است. این پارامتر برای پردازش داده‌های دریافتی اهمیت حیاتی دارد.

## ۹) توابع مفید در برنامه نویسی شبکه

بغیر از توابع سیستمی معرفی شده توابع دیگری هم هستند که برای برنامه نویسی شبکه بسیار مفید و کارآمد هستند. در ادامه برخی از مهمترین آنها را تشریح خواهیم کرد:

### ۹-۱) تابع getpeername()

```
#include <sys/socket.h>

int getpeername(int sockfd, struct sockaddr *addr, int *addrlen);
```

با استفاده از این تابع می‌توانید هویت طرف مقابل ، شامل آدرس IP و آدرس پورت پروسهٔ طرف مقابل ارتباط را استخراج نمائید. پارامترهای این تابع بصورت ذیل تعریف شده است :

**sockfd** : مشخصه سوکت مورد نظر

**addr** : استراکچری است از نوع sockaddr که قبلاً ساختار آن تعریف شده است. این استراکچر توسط سیستم عامل با آدرس IP و آدرس پورت طرف مقابل پر شده است.

**addrlen** : طول استراکچر sockaddr

در صورت عدم موفقیت تابع فوق مقدار برگشتی (۱-) خواهد بود و در متغیر سراسری errno شماره خطا برای بررسی نوع خطا تنظیم خواهد شد.

نکته ای که ممکن است برنامه نویس فراموش کند آن است که ترتیب آدرس IP و آدرس پورت بصورت BE است و اگر ماشین شما از نوع LE است باید حتماً آنرا تبدیل کنید.

### ۹-۲) تابع gethostname()

این تابع نام ماشینی را که برنامهٔ شما روی آن اجرا می‌شود ، بر خواهد گرداند. این نام یک رشته کاراکتری معادل با نام نمادین ماشین است نه آدرس IP آن (مثلاً www.ibm.com). فرم کلی تابع بصورت زیر است:

```
#include <unistd.h>

int gethostname(char *hostname, size_t size);
```

**hostname** : یک آرایه از کاراکترها (یا بعبارت بهتر یک رشته کاراکتری) است که پس از بازگشت تابع نام ماشین در آنجا ذخیره خواهد شد.

**size** : طول رشته کاراکتری بر حسب کاراکتر

اگر مقدار برگشتی (۱-) باشد خطائی بروز کرده و مقدار errno همانند قبل شماره خطا را نگه می‌دارد ولی اگر تابع فوق موفق عمل کند مقدار برگشتی صفر خواهد بود.

### ۹-۳) بکارگیری سیستم DNS برای ترجمه آدرسهای حوزه

قبلاً در مورد سیستم DNS و طرز عملکرد آن بحث شد. در اینجا وقت آن فرا رسیده است که بتوانید در محیط برنامه نویسی تقاضای ترجمه نام حوزه[1] یک سرویس دهنده را به این سیستم ارائه کرده و نتیجه را در برنامه خود استفاده نمائید.

---

[1] Domain Name

مثالهای کوچکی که تا اینجا داشته ایم همگی برای برقراری یک ارتباط با ماشین خاص مستقیما از آدرس IP آن استفاده می‌کردند و لیکن فرض کنید که شما بخواهید برنامه ای بنویسید که کاربر بتواند آدرس نام حوزه یک سرویس دهنده را بعنوان آدرس مقصد وارد نماید. تابعی که در این مورد بکار می‌آید دارای فرم کلی زیر است:

```
#include <netdb.h>

struct hostent *gethostbyname(const char *name);
```

**name** : رشته کاراکتری نام حوزه یک سرویس دهنده

مقدار برگشتی تابع ، آدرس استراکچری است از نوع hostent که ساختار آن بصورت زیر تعریف شده است :

```
struct hostent {
      char   *h_name;
      char   **h_aliases;
      int    h_addrtype;
      int    h_length;
      char   **h_addr_list;
   };

#define h_addr h_addr_list[0]
```

**hname** : نام رسمی ماشین ( برای شبکه اینترنت این رشته نام حوزه خواهد بود مثلاً (www.ibm.com

**h_aliases** : نام مستعار ماشین (این رشته با \0 ختم می‌شود )

**h_addrtype** : خانواده آدرس (همانگونه که اشاره شد در شبکه اینترنت این فیلد مقدار AF_INET خواهد داشت.)

**h_length** : طول آدرس بر حسب بایت

**h_addr_list** : یک رشته کاراکتری که در آن آدرس IP مربوط به ماشین سرویس دهنده قرار دارد. این رشته با \0 ختم می‌شود.

دقت کنید که در تابع بالا در صورت موفقیت آمیز بودن، یک اشاره گر به استراکچر بر می‌گرداند و در غیر اینصورت مقدار NULL برخواهد گشت و برخلاف

توابع قبلی متغیر errno تنظیم نخواهد شد و بجای آن متغیر سراسری herror که متغیری سیستمی است تنظیم می‌شود و در ضمن تابع سیستمی herror() برای کشف نوع خطا بکار گرفته می‌شود.

برای رفع ابهاماتی که در این تابع وجود دارد طرح یک مثال ضروری به نظر می‌رسد. به برنامه کوچک و اجرائی زیر دقت نمائید :

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>

int main(int argc, char *argv[])
{
   struct hostent *h;

   if (argc != 2) {  /* error check the command line */
      fprintf(stderr,"usage: getip address\n");
      exit(1);
   }

   if ((h=gethostbyname(argv[1])) == NULL) {  /* get the host info */
      herror("gethostbyname");
      exit(1);
   }

   printf("Host name  : %s\n", h-h_name);
   printf("IP Address : %s\n",inet_ntoa(*((struct in_addr *)h-h_addr)));

   return 0;
}
```

نام این برنامه getip است که یک آدرس نام حوزه را بعنوان ورودی دریافت کرده و نتیجه ترجمه آن را به آدرس IP و بقیه مشخصات را روی خروجی چاپ می‌کند.
نکات زیر درمورد برنامه بالا ارزش بازگوئی دارد:

الف: طریقه بکارگیری برنامه فوق بدینصورت است که نام برنامه را روی خط فرمان تایپ کرده و سپس در جلوی آن نام حوزه را با یک فاصله خالی نوشته و کلید Enter را فشار می‌دهید. مثال :

```
$ getip  www.ibm.com
```

ب: آدرس IP معادل با آدرس نام حوزه در متغیر h_addr_list واقع است و هر چند که بصورت یک رشته است که با کد 0\ ختم می‌شود ولی برای شبکه اینترنت که آدرسهای IP فعلاً چهار بایتی هستند شما فقط به چهار بایت اول آن که بصورت BE ذخیره شده‌اند نیازمندید. در برنامه فوق برای تبدیل آدرس چهاربایتی به حالت رشته ای نقطه دار بفرم (مثلاً 213.190.140.187 ) از تابع (inet_ntoa برای چاپ روی خروجی بهره گرفته شده است.

ج: عمل "تطبیق نوع" در تابع (inet_ntoa به آن دلیل بوده است که طبق تعریف اصلی متغیر h→h_addr بصورت رشته معمولی تعریف شده ولی در تابع (inet_ntoa آرگومان ورودی آن یک استراکچر از نوع in_addr است که در ابتدای فصل ساختار آن تعریف شد و چهاربایتی است. بنابراین مجبوریم با عمل "تطبیق نوع" سازگاری پارامتر ورودی را تضمین کنیم ولی در عمل اتفاق خاصی نمی‌افتد.

## ۱۰) برنامه های نمونه

پس از معرفی توابع ساده برای برنامه نویسی شبکه دو مثال ساده به شما کمک می‌کند تا با بررسی آنها اشکالات و ابهامات خود را رفع نمائید.

### ۱-۱۰) مثالی از مبادله اطلاعات به روش TCP مبتنی بر سوکتهای استریم

درمثال اول یک سیستم ساده مبتنی بر مفهوم سرویس دهنده /مشتری بررسی میشود که مطابق با آنچه گفته شد در دو برنامه مجزا باید نوشته شود : برنامه سمت سرویس دهنده و برنامه سمت مشتری. این مثال از سوکتهای نوع استریم استفاده می‌کند یعنی مبادله داده مبتنی بر روش TCP است. در ابتدا برنامه سمت سرویس دهنده را بررسی می‌نمائیم:

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/wait.h>
```

```
#define MYPORT 3490    /* the port users will be connecting to */

#define BACKLOG 10     /* how many pending connections queue will hold */

main()
{
   int sockfd, new_fd;  /* listen on sock_fd, new connection on new_fd */
   struct sockaddr_in my_addr;    /* my address information */
   struct sockaddr_in their_addr; /* connector's address information */
   int sin_size;

   if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
      perror("socket");
      exit(1);
   }

   my_addr.sin_family = AF_INET;         /* host byte order */
   my_addr.sin_port = htons(MYPORT);     /* short, network byte order */
   my_addr.sin_addr.s_addr = INADDR_ANY; /* auto-fill with my IP */
   bzero(&(my_addr.sin_zero), 8);        /* zero the rest of the struct */

   if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr)) \
                                                == -1) {
      perror("bind");
      exit(1);
   }

   if (listen(sockfd, BACKLOG) == -1) {
      perror("listen");
      exit(1);
   }

   while(1) {  /* main accept() loop */
      sin_size = sizeof(struct sockaddr_in);
      if ((new_fd = accept(sockfd, (struct sockaddr *)&their_addr, \
                                        &sin_size)) == -1) {
         perror("accept");
         continue;
      }
      printf("server: got connection from %s\n", \
                                inet_ntoa(their_addr.sin_addr));
      if (!fork()) { /* this is the child process */
```

```
            if (send(new_fd, "Hello, world!\n", 14, 0) == -1)
                perror("send");
            close(new_fd);
            exit(0);
        }
        close(new_fd);  /* parent doesn't need this */

        while(waitpid(-1,NULL,WNOHANG)>0); /* clean up child processes */
    }
}
```

عملی که این برنامهٔ ساده انجام می‌دهد آن است که هرگاه برنامهٔ سمت مشتری با این برنامه و شماره پورت ۳۴۹۰ ارتباط برقرار کند پیغام "Hello, world!\n" را دریافت خواهد کرد. بنابراین برنامه سمت سرویس دهنده دقیقاً پس از ()accept کردن یک ارتباط بدون هیچ پردازش خاصی رشته چهارده بایتی فوق را برای طرف مقابل فرستاده و سوکت متناظر راخواهد بست.

برنامه تا رسیدن به دستور (1)while نیاز به توضیح خاصی ندارد چرا که فقط یک سوکت نوع استریم ایجاد شده و به این سوکت آدرس پورت ۳۴۹۰ نسبت داده شده و با تابع ()listen اجازه داده شده تا حداکثر ده ارتباط معلق پذیرفته شود و سپس وارد حلقه بینهایت شده است. پس از آنکه برنامه وارد حلقه (1)while شد ابتدا اولین ارتباط معلق (درصورت وجود ) پذیرفته شده و مشخصهٔ سوکت جدید برای آن ایجاد شده و به برنامه برگردانده می‌شود.

پس از این کار یک فراخوان سیستمی یونیکس به نام ()fork برای ایجاد یک پروسه فرزند انجام می‌شود. بد نیست برای آشنایی بیشتر در این مورد توضیحی ارائه نمائیم:

()fork تنها راه ایجاد یک پروسس جدید در محیط یونیکس است و وظیفه آن ساختن یک پروسس تکراری دقیقاً یکسان با پروسس اولیه شامل تمام مشخصه های فایل ، رجیسترها و منابع دیگر است. پس از اجرای ()fork پروسس اولیه و پروسس نسخه برداری شده راه جداگانه ای را در پیش خواهند گرفت. از آنجائیکه ()fork بعنوان داده‌های پدر برای ساختن فرزند نسخه برداری می‌شوند ، همه متغیرها در زمان ()fork مقادیر یکسان دارند اما پس از آغاز پروسه فرزند تغییرات بعدی در هر کدام از آنها تاثیری بر روی دیگری نخواهد گذاشت (متن برنامه که غیر قابل تغییر است بین پدر و فرزند به اشتراک گذاشته می‌شود). تابع سیستمی ()fork یک مقدار

برمی‌گرداند که برای پروسس فرزند برابر صفر و برای پروسس پدر شناسه پروسه فرزند [1] خواهد بود. با استفاده از pid بازگشتی می‌توان فهمید که بین دو پروسس کدامیک فرزند و کدام پدر است.

بنابراین در برنامه فوق به ازای هر ارتباط که پذیرفته می‌شود یک پروسه جدید که بعد از تابع سیستمی ()fork شروع می‌شود بعنوان پروسه فرزند تولید شده و همانند دیگر پروسسها بصورت اشتراک زمانی از سیستم عامل سرویس می‌گیرد.

دلیل آنکه در برنامه فوق از این روش استفاده شده آن است که ارتباطات معلق بروش Polling پردازش نشوند بلکه بصورت همروند اجرا گردند. این کار باعث می‌شود که هر گونه تاخیر در یکی از ارتباطات بقیه را با تاخیر مواجه نکند بلکه به ازای هر ارتباط معلق یک پروسه فرزند ایجاد شود و همه در یک سطح بصورت اشتراک زمانی سهمی از زمان CPU را دریافت کرده و اجرا شوند. هر پروسه فرزند که به اتمام رسید یک پروسه فرزند جدید برای ارتباطی جدید ساخته می‌شود.

تابع waitpid(-1,NULL,WNOHANG) پروسه پدر را به حالت تعلیق خواهد برد تا زمانی که تمام پروسسهای فرزندش به اتمام برسند.

حال به برنامه سمت مشتری دقت نمائید. این برنامه با توجه به توضیحاتی که تا اینجا ارائه شده احتیاج به توضیح ندارد. برنامه سمت مشتری زمانی موفق عمل خواهد کرد که قبل از اجرای آن برنامه سمت سرویس دهنده اجرا شده باشد.

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>

#define PORT 3490   /* the port client will be connecting to */

#define MAXDATASIZE 100 /* max number of bytes we can get at once */
```

[1] pid ( Process Identifier )

```
int main(int argc, char *argv[])
{
   int sockfd, numbytes;
   char buf[MAXDATASIZE];
   struct hostent *he;
   struct sockaddr_in their_addr; /* connector's address information */

   if (argc != 2) {
      fprintf(stderr,"usage: client hostname\n");
      exit(1);
   }

   if ((he=gethostbyname(argv[1])) == NULL) {  /* get the host info */
      herror("gethostbyname");
      exit(1);
   }
   if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
      perror("socket");
      exit(1);
   }

   their_addr.sin_family = AF_INET;      /* host byte order */
   their_addr.sin_port = htons(PORT);    /* short, network byte order */
   their_addr.sin_addr = *((struct in_addr *)he-h_addr);
   bzero(&(their_addr.sin_zero), 8);     /* zero the rest of the struct */

   if (connect(sockfd, (struct sockaddr *)&their_addr, \
                                 sizeof(struct sockaddr)) == -1) {
      perror("connect");
      exit(1);
   }
   if ((numbytes=recv(sockfd, buf, MAXDATASIZE, 0)) == -1) {
      perror("recv");
      exit(1);
   }

   buf[numbytes] = '\0';
   printf("Received: %s",buf);

   close(sockfd);
   return 0;
}
```

### ۲-۱۰) مثالی از مبادله اطلاعات به روش UDP مبتنی بر سوکتهای دیتاگرام

ابتدا برنامه سمت سرویس دهنده را ارائه می‌نمائیم. این برنامه در سمت سرویس دهنده منتظر دریافت بسته ها باقی می‌ماند و هر گاه بسته ای را از یک مشتری دریافت کرد به همراه آدرس آن بر روی خروجی نمایش خواهد داد. برنامه نیاز به توضیح خاصی ندارد.

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/wait.h>
#define MYPORT 4950    /* the port users will be connecting to */
#define MAXBUFLEN 100

main()
{
   int sockfd;
   struct sockaddr_in my_addr;    /* my address information */
   struct sockaddr_in their_addr; /* connector's address information */
   int addr_len, numbytes;
   char buf[MAXBUFLEN];

   if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
      perror("socket");
      exit(1);
   }

   my_addr.sin_family = AF_INET;         /* host byte order */
   my_addr.sin_port = htons(MYPORT);     /* short, network byte order */
   my_addr.sin_addr.s_addr = INADDR_ANY; /* auto-fill with my IP */
   bzero(&(my_addr.sin_zero), 8);        /* zero the rest of the struct */

   if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr)) \
                                             == -1) {
      perror("bind");
      exit(1);
   }

   addr_len = sizeof(struct sockaddr);
   if ((numbytes=recvfrom(sockfd, buf, MAXBUFLEN, 0, \
               (struct sockaddr *)&their_addr, &addr_len)) == -1) {
      perror("recvfrom");
```

```
    exit(1);
}

printf("got packet from %s\n",inet_ntoa(their_addr.sin_addr));
printf("packet is %d bytes long\n",numbytes);
buf[numbytes] = '\0';
printf("packet contains \"%s\"\n",buf);

close(sockfd);   }
```

در سمت مشتری ، برنامه رشته‌ای را که بعنوان آرگومان دریافت کرده ، مستقیماً برای سرویس دهنده ارسال می‌کند. بعنوان مثال اگر برنامه را با نام talker.c نوشته و سپس کامپایل و بصورت زیر در خط فرمان اجرا نمائیم:

```
$ talker www.hserver.edu hello
```

رشته hello توسط برنامه به سمت سرویس‌دهنده ارسال خواهد شد و برنامهٔ سمت سرویس‌دهنده طبق توضیحی که ارائه شد آنرا روی خروجی چاپ خواهد کرد.

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <sys/socket.h>
#include <sys/wait.h>

#define MYPORT 4950    /* the port users will be connecting to */

int main(int argc, char *argv[])
{
    int sockfd;
    struct sockaddr_in their_addr; /* connector's address information */
    struct hostent *he;
    int numbytes;

    if (argc != 3) {
        fprintf(stderr,"usage: talker hostname message\n");
        exit(1);
```

```
    }
    if ((he=gethostbyname(argv[1])) == NULL) {  /* get the host info */
       herror("gethostbyname");
       exit(1);
    }

    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
       perror("socket");
       exit(1);
    }

    their_addr.sin_family = AF_INET;      /* host byte order */
    their_addr.sin_port = htons(MYPORT);  /* short, network byte order */
    their_addr.sin_addr = *((struct in_addr *)he-h_addr);
    bzero(&(their_addr.sin_zero), 8);     /* zero the rest of the struct */

    if ((numbytes=sendto(sockfd, argv[2], strlen(argv[2]), 0, \
        (struct sockaddr *)&their_addr, sizeof(struct sockaddr))) == -1) {
       perror("sendto");
       exit(1);
    }

    printf("sent %d bytes to %s\n",numbytes,inet_ntoa(their_addr.sin_addr));
    close(sockfd);

    return 0;
}
```

## ۱۱) بلوکه شدن پروسه های تحت شبکه

مفهوم بلوکه شـــدن یـک پروسـه از مبـاحث طراحـی سیسـتم عـامل اسـت کـه نمی‌تـوان در اینجـا کـاملا آنـرا تشـریح کـرد ولـی نکـاتی از آن را کـه بـه مبحـث مـا مرتبـــط اســـت توضیح می‌دهیـــم.

در یـک عبـارت سـاده دسـتورات ورودی / خروجـی یـــک پروســـه در حـــال اجـــرا را متوقف کرده و تـا زمـانی کـه ورودی /خروجـی آن کـامل نشـود و مجـدداً از سیسـتم عـامل بـرش زمـانی دریـافت نکنـد متوقـف خواهـد مـاند. توابـع recvto() و recv() و accept() از همین دسـته هسـتند ( یعنـی بـه نوعـی ورودی / خروجـی محسـوب می‌شـوند) و بـالطبع

برنامه هایی که ایـن توابـع را اجـرا نمـایند توسـط سیسـتم عـامل بلوکـه خواهنـد شـد و تـا کامل شدن آنها بلوکـه بـاقی می‌مـانند.

بعنوان مثال تـابع accept() یکـی از ارتباطـات معلـق و بـه صـف شـده TCP را بـه برنامـه شـما تحویـل می‌دهـد. حـال وقتـی هیـچ ارتبـاط معلقـی وجـود نـدارد یعنـی هیـچ ماشـینی تقاضای برقـراری ارتبـاط نـداده اسـت ایـن تـابع منجـر بـه بلوکـه شـدن برنامـه می‌شـود تـا زمانی که تقاضائی برسد، در ایـن حـالت سیسـتم عـامل برنامـه بلوکـه شـده را احیـا کـرده و اجرا می‌نماید. ایـن روش کـلاً بسـیار مفیـد و کـارآمد اسـت و لیکـن راهـی وجـود دارد کـه سیسـتم عـامل پـس از اجـرای ایـن تـابع (وبقیـه توابـع) برنامـه شـما را بلوکـه نکنـد. بـرای اینکـار از فراخـوان سیسـتمی fcntl() بـه نحـوی کـه در مثـال بعـدی آمـده اسـت اسـتفاده کنید. در ایـن حـالت بعـد از فراخوانـی توابـع accept() یـا recv() چـه موفقیـت آمـیز و چـه ناموفق برنامه شـما بلوکـه نخواهـد شـد بلکـه خـود برنامـه نویـس موظـف اسـت در برنامـهٔ خـود امکـان پذیـرش ارتبـاط یـا دریـافت داده‌هـا را بررسـی نمـاید. در حقیقـت ایـن روش همـان روش سرکشـی[1] اسـت کـه در محیطـهای چنـد کـاربره روش مناسـبی محسـوب نمیشود چرا که در ایـن روش برنامـه شـما در یـک حلقـه بـی نـهایت وقـت CPU را گرفتـه و پشت سـرهم سـوکتها را سرکشـی می‌نمـاید.

دقـت کنیـد کـه اگـر نتیجـه accept() مقـدار (۱-) باشـد چـون برنامـه شـما بلوکـه نمی‌شود ، اگـر سـعی کنیـد داده‌ای را دریـافت یـا ارسـال کنیـد بـا خطـای سیسـتمی و قطـع برنامه مواجه خواهیـد شـد.

برنامه‌نویسی تحت شبکه ، ابزارهای بهتر و قوی‌تری نسبت به زبان معمولی C دارد ، ولیکن ارائهٔ مفاهیم سوکت و توابع لازم برای برنامه‌نویسی تحت شبکه ، با استفاده از زبان C ، مفاهیم را بهتر و بنیادی‌تر آموزش می‌دهد ، زیرا برای ارائهٔ مفهوم سوکت و برنامه‌نویسی تحت شـبکه با زبانهای شیئ‌گرا ، مجبور خواهیم بود حجم بسیار زیادی از کدهـای یـک شـیئ را در زبـانی مثل جاوا بررسی و تحلیل کنیم.

در بخش آتی سعی خواهد شد ضمن معرفی زبان جاوا ، قابلیتهای شبکه‌ای این زبان ، بصورت فهرست‌وار مرور شود.

---

[1] Polling

## ۱۲) امکانات زبان JAVA در برنامه‌نویسی شبکه

### ۱-۱۲) مقدمه

**جاوا** یک زبان برنامه‌نویسی شیءگراست که می‌توان گفت بطور مستقیم از C و C++ گرفته شده است و اهدافی مثل "عدم وابستگی به ماشینِ اجرا"[۱] ، که C++ در عمل نتوانست بدان دست یابد را به نحو زیبایی پیاده‌سازی کرده است. یعنی بدون هیچ دغدغه‌ای می‌توان بر روی یک ماشین مبتنی بر سیستم عامل MS-Windows برنامه‌ای به زبان جاوا نوشت و آنرا بر روی ماشینی مبتنی بر یونیکس اجرا کرد. این قابلیت در واقع به نوعی نیاز شبکهٔ اینترنت محسوب می‌شد و باعث شد تا جاوا در دوران اوج زبان C++ ، ناگهان نگاهها را معطوفِ خود کند و همانند **وِب** در عرض چند سال به ابزاری مطمئن برای برنامه‌نویسی شبکه تبدیل شود.

بزرگترین ضعف برنامه‌های نوشته‌شده به زبان C++ ، آن دسته از "اشکالاتِ پنهان"[۲] است که در اثر آزادی برنامه‌نویس در مدیریت حافظه و کار با اشاره‌گرها ، در برنامه پدید می‌آید. جاوا با حذف اشاره‌گرها و تقبل مدیریت حافظه ، این دو ضعف را برطرف کرد و به یک زبان برنامهٔ نویسی **امن** مبدل شد.

هنگامیکه مهندسین شرکت سان توجه خود را به پروژهٔ **گرین**[۳] معطوف کردند تا برای لوازم الکترونیکی این شرکت نرم‌افزار پیشرفته بسازند ، دریافتندکه کامپایلرهای C و C++ برای اینکار نارسایی دارند ، از اینرو به فکر خلق زبانی جدید افتادند که در ابتدا اُآک Oak نام گرفت و پس از مدتی به جاوا تغییر نام داد. به دنبال این پیشرفت ، شرکت سان برای جاوا یک مرورگر ساخت که می‌توانست در محیط وب قطعه برنامه‌های جاوا را اجرا کند.

جاوا زبانی است ساده ، ایمن ، قابل حمل ، شیءگرا ، توانمند در حمایت از برنامه‌های "چند ریسمانی"[٤] ، با "معماری خنثی"[٥] ، که با زبانهایC و C++ تفاوتهایی دارد. این تفاوتها را می‌توان در موارد زیر خلاصه کرد:

- **اشاره‌گرها** : همانطور که قبلاً نیز اشاره شد در جاوا اشاره‌گری وجود ندارد ، این درحالیست که در C/C++ می‌توان از اشاره‌گر استفاده کرد. این امر باعث می‌شود

---

[۱] Platform Independence
[۲] Bug
[۳] Green
[٤] Multithread
[٥] Achitecture-neutral

نتوان حافظه را بخوبی مدیریت کرد؛ هرگونه استفادهٔ نامناسب در بکارگیری اشاره‌گرها در برنامه‌های C/C++ ، می‌تواند حداقل برنامه را متوقف کند.

- **استراکچرها و یونیون‌ها**[۱] : در زبان C++ سه نوع از "انواع داده"[۲] وجود دارد : کلاس ، استراکچر و یونیون ، در حالیکه جاوا فقط شامل کلاس است. در یک زبان برنامه‌نویسی شیئ‌گرا مثل C++ وجود "کلاس" ، برنامه‌نویس را از داده‌هایی نظیر استراکچر و یونیون بی‌نیاز می‌کند ، ولی C++ برای سازگاری با C مجبور بود از آنها پشتیبانی کند در حالی که در جاوا هیچ الزامی در تعریف آنها وجود نداشت.
- **توابع** : جاوا هیچ تابعی ندارد ، چون شیئ‌گرا است و برنامه‌نویس را مجبور به استفاده از متودهای کلاس[۳] می‌کند ، در حالیکه در C++ به غیر از کلاس ، توابع نیز تعریف شده‌اند ، که چندان با مفهوم شیئ‌گرایی مطابقت ندارد.
- **وراثت چندگانه**[۴] : وراثت چندگانه به این معناست که یک کلاس را از چند کلاس دیگر مشتق کنیم که این عمل در جاوا براحتی امکان‌پذیر است ، در حالیکه در C/C++ این کار بسیار مشکل بوده و باعث پیچیدگی و خطا می‌شود.
- **رشته‌ها** : در جاوا ، رشته‌ها را بعنوان اشیاء کلاس اولیه داریم در حالیکه در C/C++ ساختاری شیئ‌گرا برای پشتیبانی رشته‌های متنی نداریم.
- **دستور goto** : در C/C++ این دستور کمابیش استفاده می‌شود ولی در جاوا اگرچه این دستور جزو کلمات کلیدی است ولی استفاده از آن پشتیبانی نمی‌شود. عدم پشتیبانی از دستورات پرش غیرساختاریافته ، باعث کاهش خطا در جاوا شده است.
- **Operator overloading** : در جاوا بر خلاف C/C++ از توانایی تغییر عملکرد اپراتورها پشتیبانی نمی‌شود تا پیچیدگی زبان کمتر شود.
- **تبدیل خودکار نوع**[۵] : در زبان C/C++ شما می‌توانید یک متغیر را از نوعی مثل float تعریف کنید و سپس مقداری مثل int به آن نسبت بدهید ، ولی اگر این عمل را در زبان جاوا انجام دهید فوراً با پیغام خطا مواجه خواهید شد. اینگونه سخت‌گیریها ، امنیت ذاتی جاوا را افزایشی چشمگیر داده است.

---

۱ Union
۲ Data Type
۳ Method
۴ Multiple Inheritance
۵ Automatic Conversion

➢ **آرگومانهای خط فرمان**[۱] : C/C++ دو پارامتر argc و argv را به برنامه ارسال می‌کند که argc تعداد آرگومانهای ذخیره شده در argv را مشخص می‌کند ، در حالیکه argv یک اشاره‌گر به آرایه‌ای از کاراکترهاست. با حذف اشاره‌گرها در جاوا ، به جای argv از args استفاده شد؛ [0]args اولین پارامتر خط فرمان است.

برای تاکید بیشتر تکرار می‌شود که مشکل عمده C/C++ اشاره‌گرها و مدیریت حافظه است در حالیکه مدیریت حافظه در جاوا بصورت خودکار انجام می‌شود. برای مشخص شدن قضیه به این نکته دقت کنید که وقتی در C++ یک بلوک حافظه یا یک کلاس را new() می‌کنید ، خودتان موظف به آزادسازی آن هستید و انجام ندادن این کار یک اشکال محسوب می‌شود. پاکسازی حافظه از اشیاء بی‌مصرف برعهدهٔ خود جاوا است. کار با آرایه‌ها در جاوا بسیار آسانتر و مطمئن‌تر است چون آرایه‌ها در این زبان ، عضوی از یک کلاس می‌باشند .

C++ از اصول شیئ‌گرایی به موازات برنامه‌نویسی به سبک قدیم حمایت می‌کند که این حالت در جاوا وجود ندارد و جاوا صد در صد شیئ‌گراست.

کامپایلر جاوا یک برنامهٔ نوشته شده را به کدهای اجرایی یک ماشین خاص مثل IBM یا Apple تبدیل نمی‌کند ، بلکه آنرا به کدهای اجرایی یک ماشین فرضی به نام JVM[۲] ترجمه می‌کند که مختص به هیچ پردازنده‌ای نیست ، بلکه زبان اسمبلی یک ماشین مجازی است. به کدهای اجرایی این ماشین مجازی **"بایت‌کُد"**[۳] گفته می‌شود. بنابراین نتیجهٔ ترجمهٔ یک برنامهٔ جاوا یک فایل میانی حاوی بایت‌کد است. هر ماشین که بخواهد یک برنامهٔ جاوا را اجرا کند موظف است از **"مفسر زمان اجرای جاوا"** استفاده کند تا دستورات مجازی JVM به کدهای اجرایی واقعی از یک ماشین تبدیل شود. هر ماشین برای خودش JVM خاص دارد. بدین گونه جاوا فارغ از ساختار ماشین ، ترجمه و اجرا می‌شود. نحوهٔ اجرای برنامه‌های کاربردی جاوا در یک ماشین بصورت زیر است:


---

۱ Command-line Arguments
۲ Java Virtual Machine
۳ Bytecode

ماشین مجازی جاوا (JVM) دارای پنج بخش مهم زیر می‌باشد :

- **مجموعه دستورات بایت‌کد** : بایت‌کدها به عنوان دستورالعملهای اجرایی (ولی مجازی) شامل دو قسمت عملوند و عملگر می‌باشند. هر نوع دادهٔ اولیه در جاوا یک بایت‌کد مخصوص به خود دارد. این دستورالعملهای مجازی توسط JVM به یک یا چند دستورالعمل اجرایی از یک ماشین تبدیل می‌شوند.
- **مجموعه رجیسترها** : مجموعهٔ رجیسترها در ماشین مجازی جاوا ، همگی ۳۲ بیتی هستند.
- **پشته**[۱] : پشته در JVM دقیقاً مثل پشته‌هایی است که در دیگر زبانهای برنامه‌نویسی برای ارسال پارامتر به توابع و ذخیرهٔ متغیرهای محلی[۲] از آن استفاده می‌شود. هر متود از یک کلاس در زبان جاوا برای خود پشته‌ای دارد که متغیرهای محلیِ متود ، محیط اجرای آن و پارامترهای ارسالی به متود ، در این پشته قرار می‌گیرند.
- **فضای کاری**[۳] : فضای کاریِ برنامهٔ JVM ، قسمتی از حافظه است که برنامه‌نویس قادر به دخالت در آن نیست ، بلکه خودِ کامپایلر ، حافظهٔ این قسمت را مدیریت می‌کند و در صورت لزوم فضایی را تخصیص داده یا آنرا آزاد می‌کند. یعنی دست برنامه‌نویس از دسترسی غیر مجاز به حافظه کوتاه شده و عمل تخصیص و آزادسازی فضای مورد نیاز حافظه به کامپایلر محول شده است.
- **فضای ذخیره‌سازی متدها** : فضای متدهای JVM یک فضای ۸ بیتی است که در قسمت خاصی از حافظه ذخیره می‌شود.

### ۲-۱۲) انواع داده در جاوا

جاوا ۸ نوع داده اصلی دارد که در جدول (۲-۷) فهرست شده‌اند. هر نوع ، یک اندازهٔ مشخص بر حسب بایت دارد . برخلاف زبان C که برای دادهٔ نوع صحیح بسته به نوع معماری ماشین ، ۱٦ ، ۳۲ یا ٦٤ بیت در نظر گرفته می‌شود ، زبان جاوا برای این نوع داده فقط ۳۲ بیت در نظر می‌گیرد که این نکته مزایایی برای زبان جاوا به وجود می‌آورد. یکی از این مزایا آن است که باعث می‌شود برنامه روی انواع ماشینهای ۱٦ ، ۳۲ و ٦٤ بیتی به یک شکل کار کند و نتیجه واحد ارائه دهد.

---

۱ Stack
۲ Local Variable
۳ Work Space

| نوع | اندازه | توضیح |
|---|---|---|
| byte | 1 Byte | یک عدد صحیح علامت‌دار با محدودهٔ ۱۲۸- تا ۱۲۷+ |
| short | 2 Byte | یک عدد صحیح علامت‌دار دو بایتی |
| int | 4 Byte | یک عدد صحیح علامت‌دار چهار بایتی |
| long | 8 Byte | یک عدد صحیح علامت‌دار هشت بایتی |
| float | 4 Byte | یک عدد اعشاری چهار بایتی با استاندارد IEEE |
| double | 8 Byte | یک عدد اعشاری هشت بایتی با استاندارد IEEE |
| boolean | 1 Bit | یک پرچم تک بیتی که دو حالت True یا False دارد |
| char | 2 Byte | یک تک‌کاراکتر یونی‌کد ( دو بایت ) |

جدول (۲-۷) انواع دادهٔ اصلی در جاوا

در زبانهای شیئ‌گرا نظیر جاوا هر چیزی یک شیئ است. هر شیئ دارای مجموعه‌ای از **رفتار**[۱] و **صفات**[۲] است و یکسری **متود**[۳] نیز برای دسترسی به اشیاء وجود دارد. رفتار تنها راه برای این است که به شی بگوییم چه عملی را انجام دهد. برای اینکه رفتار یک شیئ را بسازید باید ابتدا یک متود ایجاد کنید که این متودها شبیه به توابع در دیگر زبانها می‌باشند. برخلاف C++ ، جاوا توابعی که خارج از کلاسها و جداگانه تعریف شده باشند ، ندارد. اگر یک شیئ را تعریف کردید می‌توانید تعیین کنید که چه برنامه‌هایی و با چه سطحی از دسترسی به این شیئ می‌توانند وجود داشته باشند. یک برنامه جاوا ، شامل یک یا چند **بسته**[٤] می‌باشد که هر کدام از این بسته ها خود شامل تعاریف کلاسها هستند و توسط بقیهٔ برنامه‌ها قابل استفاده می‌باشند.

اشیاء در جاوا می‌توانند به صورت پویا و در حین اجرای برنامه تولید شوند. هر کلاس می‌تواند زیر کلاس یا کلاس والد (Super Class) داشته باشد. این کلاس همیشه از کلاس والد خصوصیات و رفتارها و روشها را به ارث می‌برد . البته همیشه نمی‌توان به متغیرهای داخلی کلاس والد دسترسی مستقیم داشت. دسترسی مستقیم به متغیرهای کلاس والد ، به شرطی امکان‌پذیر است که آن متغیرها بصورت public تعریف شده باشند.

---

۱ Behavior
۲ Attributes
۳ Method
٤ Pakage

حال به ارائه مثالی می‌پردازیم که مفهومی از شیئ‌گرایی را در خود دارد. در ارائهٔ مثال فرض کرده‌ایم که با اصول C++ آشنایی دارید. در این مثال ، یک **بسته** (pakage) داریم که دو کلاس را مشخص می‌کند. یک عدد مختلط را در نظر بگیرید؛ می‌دانید که این نوع عدد شامل دو قسمت حقیقی و موهومی[۱] است:

```
class ComplexNumber {
یک شیئ برای متغیر مختلط ایجاد می‌شود. //
protected double re, im;
تعریف قسمتهای حقیقی و موهومی ( این قسمتها در خارج از کلاس در دسترس نیستند و مخفیند ) //
پنج متود زیر متغیرهای پنهان بالا را استفاده و پردازش می‌کنند. //
pablic void Complex (double x, double y) { re=x;  im=y;}
public double Real()  {return re;}
public double Imaginary()  { return im; }
public double Magnitude ( ) { return Math.sqrt (re*re+im+im); }
public double Angle()  {return Math.atan(im/re);}
class test {
تعریف یک کلاس جدید برای استفاده از کلاس بالا //
public static void main (String aras[ ] ) {
ComplexNumber C;
تعریف یک شیئ از نوع متغیر مختلط با تعریف بالا //
C=new ComplexNumber();
شیئ از نوع متغیر مختلط در حافظه تولید می‌شود. //
C. Complex (3.0, 4.0);
مقداردهی اولیه به یک شیئ از نوع متغیر مختلط//
System.out.println ("The magnitude of C is " + C. Magnitude() );
   }
}
```

[۱] Real & Imaginary Part

در این مثال هر شیئ شامل دو متغیر re و im می‌باشد که هر دو اعداد اعشاری ٦٤ بیتی هستند. این متغیرها نمی‌توانند توسط کلاسهای دیگر دستکاری شوند. (یعنی قابل دسترسی نیستند ، چون از نوع protected تعریف شده‌اند.) اگر این متغیرها را از نوع public تعریف می‌کردیم برای هر بسته در هر جا این متغیرها قابل رویت و دسترسی بودند و این حالت مطلقاً مفید نیست.

حال به مثالی دیگر توجه کنید:

```
class Factorial {
public static void main(int argc, String args[ ])  // بدنهٔ برنامهٔ اصلی
long i, f, lower=1, upper=20; //  تعریف چهار متغیر صحیح چهار بایتی
  for (i=lower ; i<=upper ; i<=upper; i++) { // یک حلقه دقیقاً مشابه با زبان C++
    F= factorial(i) ;                                    //f=i!
    System.out.println (i+ "  " +f); // print i and t
    }
}
  static long factorial (long k) {  // تعریف یک تابع خود فراخوان فاکتوریل
     if (k = =0)
          return 1;                          // 0! = 1
          else
  return k * factorial (k-1);    //  k! = k+ (k-1)!  }
}
```

در مثال بالا ابتدا کلاس اصلی برنامه به نام Factorail تعریف شده و سپس تابع factorial به عنوان متودی از این کلاس تعریف گردیده است. در متود اصلیِ کلاس برنامه که همیشه main نام دارد ، متود factorial فراخوانی شده است.

### ۱۲-۳) اپلت Applet

**اپلت** ریزبرنامه یا برنامهٔ کوچکی است که درون یک صفحهٔ وب قرار می‌گیرد و روی یک سرویس‌دهندهٔ اینترنت قابل دسترسی بوده و به عنوان بخشی از یک سند وب بر روی ماشین مشتری اجرا می‌شود. ( البته به شرطی که مرورگر مجهز به مفسر جاوا[1] باشد می‌توان آن را

[1] Java enabled browser

مشاهده کرد.) اپلت‌ها با برچسب APPLET درون صفحهٔ وب تعریف می‌شوند ولی فایلی خارجی به حساب می‌آیند. چون اپلت جهت استفاده در محیط وب نوشته می‌شود لذا کمی پیچیده‌تر از یک برنامهٔ معمولی است. هنگامیکه می‌خواهید یک اپلت را در صفحه وب قرار دهید باید ابتدا تمام کلاسهای مورد نیاز آن اپلت را بسازید ، سپس آنرا کامپایل کرده و بعد با استفاده از زبان HTML یک صفحه وب ساخته و اپلت را درون صفحه وب تعریف کنید. چون اپلتها دارای خط فرمان نیستند ، برای فرستادن آرگومانهای متفاوت باید از برچسب <APPLET> استفاده کنیم. دو راه برای اجرای یک اپلت وجود دارد:

- اجرا نمودن اپلت داخل یک مرورگر سازگار با جاوا مثل Netscape Navigator
- استفاده از Applet Viewer که این برنامه ، اپلت را خارج از مرورگر و در یک پنجره ، اجرا می‌کند ، که برای اشکال‌زدائی از اپلتها راهی سریع و آسان محسوب می‌شود.

اپلت یک برنامهٔ اجرایی است و برای اجرا در محیط مرورگر در نظر گرفته شده تا قابلیتهایی که صفحات وب ندارند از طریق آنها فراهم شود. اپلتها به همراه صفحات وب برای کاربران وب ، ارسال و روی ماشین کاربر اجرا می‌شود. این برنامهٔ اجرایی نباید عمداً یا سهواً قادر باشد صدمه‌ای به سیستم کاربر وارد کند؛ لذا اپلتها در مقایسه با برنامه‌های معمولی که به زبان جاوا نوشته می‌شوند ، دارای محدودیتهای زیر است :

- اپلت جز در موارد محدود و تحت نظارت شدید و آنهم برای خواندن ، قادر به دسترسی به سیستم فایل نیست.
- اپلت قادر به فراخوانی و اجرای هیچ برنامه‌ای روی ماشین اجراکنندهٔ خود نیست.

در برنامه‌های کاربردی ، بدنهٔ برنامهٔ اصلی با بلوک **main(){** شروع می‌شود و درنهایت با علامت **}** پایان می‌پذیرد ، ولی اپلتها در جاوا متود main() ندارند. صورت کلی بدنهٔ یک اپلت به شکل زیر تعریف می‌شود:

```
public class Example extends  java.applet.Applet {
…
}
```

با این تعریف ، کلاس Example ، کلاس از پیش تعریف شدهٔ Applet را به ارث برده و تمام مقدماتی را که یک اپلت برای فعل و انفعال با مرورگر دارد ، فراهم می‌آورد. چندین اپلت می‌توانند بصورت مستقل در یک صفحهٔ وب ( روی مرورگر ) اجرا شوند.

وقتی یک کلاس را به صورت اپلت تعریف می‌کنید ، چندین متود اصلی و بنیادی را به ارث می‌برد که این متودها به خودی خود کاری انجام نمی‌دهند. برای آنکه یک اپلت عملیاتی شود ،

برنامه‌نویس باید این متودها را "باطل و دوباره‌نویسی"[۱] کند. پنج مورد از حیاتی‌ترین این متودها عبارتند از:

**init()  start()  stop()  destroy()  paint()**

دو متودی که بیش از بقیه احتیاج به دوباره‌نویسی دارند متودهای ()paint و ()init می‌باشند.

- متود **()paint** یکی از مهم‌ترین متودهای هر اپلتی است که برنامه‌نویس بدان احتیاج دارد. هر چیزی که بخواهد در پنجرهٔ خروجی اپلت نمایش داده شود ، با استفاده از این متود امکان‌پذیر خواهد بود. متود ()paint فقط یک آرگومان می‌پذیرد و بازنویسی آن بصورت زیر است:

```
public void paint(Graphics screen) {

        / / display statements go here

}
```

در مثال بالا آرگومان ورودی متود ، یک شیئ گرافیکی است. کلاس Graphics از اشیایی تشکیل شده که می‌توانند همه صفات و رفتارها را که نیاز است به عنوان متن ، گرافیک و بقیه اطلاعات روی پنجره ، نمایش داده شوند کنترل کنند. اگرشما از یک شی Graphics در اپلتتان استفاده می‌کنید ، دستور import را قبل از تعریف class در ابتدای فایل برنامه بیاورید:

**import java.awt.Graphics;**

- متود **()init** فقط یکبار و آنهم هنگام بارگذاری اپلت در مرورگر ، اجرا می‌شود؛ بنابراین متود ()init در واقع برای تنظیم کردن و مقداردهی اولیه به اشیاء و متغیرهایی که در طول اجرای اپلت مورد نیازند استفاده می‌شود. به عنوان یک پیشنهاد ، این متود جایی مناسب برای تنظیم نوع قلم (فونت) ، رنگ قلم و رنگ پس‌زمینه صفحه می‌باشد.

- متود **()start** : با این متود ، اپلت راه‌اندازی شده و آغاز به کار خواهد کرد. اگر اپلت با استفاده از متود ()stop موقتاً متوقف شده باشد ، با این متود از سرگرفته می‌شود. عملیاتی که برای راه‌اندازی یک اپلت مورد نیاز است ، در این متود دوباره‌نویسی می‌شود.

- متود **()stop** : هنگامی که این متود صدا زده شود ، اجرای اپلت موقتاً متوقف خواهد شد. زمانیکه کاربر یک صفحه وب (شامل اپلت) را رها می‌کند و به سراغ صفحه‌ای دیگر می‌رود ، این متود بطور خودکار فراخوانی می‌شود. (البته می‌توان این متود را به صورت مستقیم صدا

---

[۱] Override

زده و اپلت را متوقف کرد.) برنامه‌نویس تمهیدات لازم برای توقف اپلت را با دوباره‌نویسی این متود ، فراهم می‌آورد.

- متود **destroy()** : این متود درست برخلاف متود init() ، به منظور پایان دادن به اجرای اپلت فراخوانی می‌شود. برنامه‌نویس موظف است کارهایی را که باید در هنگام خاتمهٔ اپلت انجام شود ، در این قسمت دوباره‌نویسی کند.

مثلاً فرض کنید بخواهیم یک اپلت بنویسیم که در محیط مرورگر اجرا شده و پیغام سادهٔ Hello Web! بر روی پنجرهٔ آن نمایش یابد. چنین اپلتی فقط نیاز به بازنویسی متود paint() دارد تا بتواند روی پنجرهٔ خروجی پیغام را نمایش بدهد:

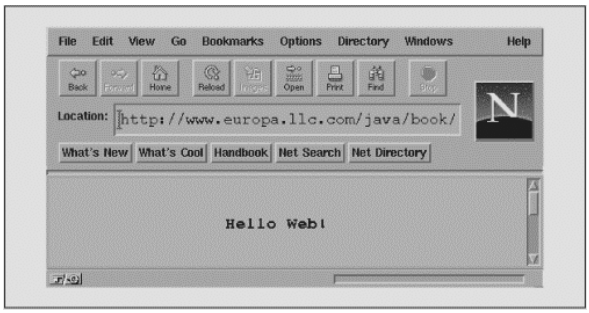
```
import java.awt.Graphics;
public class HelloWeb extends java.applet.Applet {
   public void paint( java.awt.Graphics gc ) {
      gc.drawString("Hello Web!", 125, 95 );
   }
}
```

برای اجرای یک اپلت ، لازم است صفحه وبی ایجاد کنید که اپلت را بارگذاری کند. برای ایجاد یک صفحه وب ، یک فایل جدید روی ویرایشگر معمولی باز کرده و پس از نوشتن یک صفحهٔ وب ساده همانند زیر ، آنرا با پسوند **html.** ذخیره کنید. سپس آنرا در محیط مرورگرتان باز کنید:

```
<html>
<head> </head>
 <body>
<applet code=HelloWeb width=300 height=200>
</applet>
</body>
</html>
```

در قطعه کد بالا ، پارامتر width ، طول پنجره و پارامتر height ، ارتفاع پنجرهٔ نمایش اپلت را مشخص می‌کند. ( در فصل دهم در مورد صفحات وب بیشتر خواهیم آموخت. ) خروجی اپلت بالا به صورت شکل (۳-۷) خواهد بود. به عنوان مثالی دیگر در شکل (٤-۷) ، یک اپلت با پنج عامل زیر در محیط مرورگر نشان داده شده است:

۱- فیلد ورود داده‌های متنی TextField  ۲- پانل Panel  ۳ - منوی انتخاب Choice menu
٤- صفحه ترسیم گرافیک Canvas  ۵- کلید فشاری Button

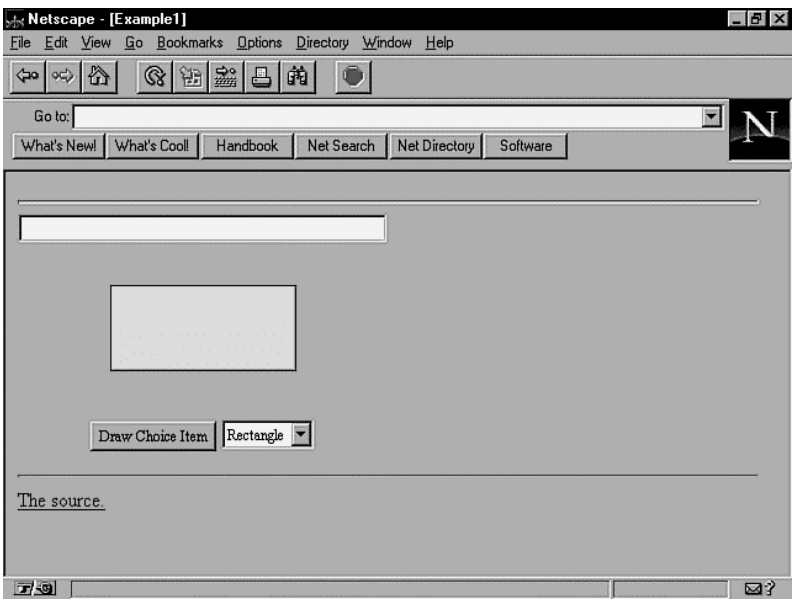



شکل (۳-۷) خروجی یک اپلت نمونه در محیط مرورگر

```
import java.awt.*;
import java.lang.*;
import java.io.*;
import java.applet.*;

// This program illustrates a simple applet with a TextField,
// Panel, Button, Choice menu, and Canvas.

public class Example1 extends Applet {
TextField tf;
DrawCanvas c;
Button drawBtn;
Choice ch;
// Add the Components to the screen..
.
public void init() {

// Set up display area...
resize(300,200);
setLayout(new BorderLayout());

// Add the components...
// Add the text at the top.
tf = new TextField();
add("North",tf);

// Add the custom Canvas to the center
c = new DrawCanvas(this);
add("Center",c);

// Create the panel with button and choices at the
bottom...
Panel p = new Panel();
drawBtn = new Button("Draw Choice Item");
p.add(drawBtn);
```

```
// Create the choice box and add the options...
ch = new Choice();
ch.addItem("Rectangle");
ch.addItem("Empty");
ch.addItem("Text");
p.add(ch);
add("South",p);
}

// Handle events that have occurred
public boolean handleEvent(Event evt) {
switch(evt.id) {
// This can be handled
case Event.ACTION_EVENT: {
if(evt.target instanceof Button) {
// Repaint canvas to use new choices...
c.repaint();
} // end if
return false;
}
default:
return false;
}
}
// Return the current choice to display...
public String getChoice() {
return ch.getSelectedItem();
}

// Return the text in the list box...
public String getTextString() {
return tf.getText();
}
}

// This is a custom canvas that is used for drawing
// text, a rectangle, or nothing...
class DrawCanvas extends Canvas {
Example1 e1app;
// Constructor - store the applet to get drawing
info...
public DrawCanvas(Example1 a) {
e1app = a;
}
// Draw the image per the choices in the applet...
public synchronized void paint (Graphics g) {
// Get the current size of the display area...
Dimension dm = size();
// Draw based on choice...
String s = e1app.getChoice();
```

```
// Calculate center coordinates...
int x,y,width,height;
x = dm.width/4;
y = dm.height / 4;
width = dm.width / 2;
height = dm.height / 2;
// Paint a rectangle in the center...
if (s.compareTo("Rectangle") == 0) {
// Draw the rectangle in the center with colors!
g.setColor(Color.blue);
g.drawRect(x,y,width,height);
g.setColor(Color.yellow);
g.fillRect(x + 1,y + 1,width - 2,height - 2);
} // end if
// Get the text in the applet and display in the
middle...
if (s.compareTo("Text") == 0) {
String displayText = e1app.getTextString();
g.setColor(Color.red);
g.drawString(displayText,x,y + (height/2));
}
}
}
```




شکل (٤-٧) خروجی یک اپلت نمونه با پنج عامل کنترل در محیط مرورگر

### ۴-۱۲) امکانات جاوا برای برنامه‌نویسی سوکت

java.net یک بسته از بسته‌های جاوا است که شامل کلاسهایی برای کار با شبکه ، سوکتها و URLهاست. در این بسته دو نوع کلاسِ سوکتِ استریم برای برنامه‌نویسی شبکه تعریف شده است:

- **کلاس Socket** : کلاسی جهت برقراری ارتباط و مبادلهٔ داده در سمت مشتری است.
- **کلاس ServerSocket** : کلاسی جهت تعریف ارتباط و مبادلهٔ داده در سمت سرویس‌دهنده است.

ابتدا کلاس Socket را مورد بررسی قرار می‌دهیم. در این کلاس چندین متود تعریف شده که مهمترین آنها در زیر معرفی می‌شوند:

**Socket(String host, int port)**
**Socket(InetAddress address, int port)**
**synchronized void close()**
**InputStream getInputStream()**
**OutputStream getOutputStream()**

متودهای اول و دوم در حقیقت متودهای **"سازندهٔ"**[۱] کلاس نوع Socket هستند که دارای دو تعریف مجزا بوده و می‌توانند به دو صورت استفاده شوند:

الف) **Socket(String host, int port)** : برای ایجاد کلاس سوکت از طریق این متود ، مستقیماً آدرس نام حوزهٔ یک ماشین و شمارهٔ پورت سرویس‌دهندهٔ مورد نظر در آرگومانهای آن مشخص می‌شود. اگر از این متود برای خلق یک سوکت استفاده شود ، قبل از بوجود آمدن آن ، نام حوزه بصورت خودکار توسط DNS ترجمه شده و آدرس IP آن باز خواهد گشت و در صورت موفقیت‌آمیز نبودن این عمل ، ادامهٔ کار ممکن نخواهد بود. مثال :

```
try {
   Socket sock = new Socket("cs.wustl.edu", 25);
}
catch ( UnknownHostException e ) {
   System.out.println("Can't find host.");
}
catch ( IOException e ) {
   System.out.println("Error connecting to host.");
}
```

[۱] Constructor

ب) **Socket(InetAddress address, int port)** : برای ایجاد کلاس سوکت از طریق این متـود ، آدرس IP یک ماشین و شمارهٔ پورت سرویس‌دهندهٔ مـورد نظـر در آرگومانـهای آن مشـخص می‌شود. آدرس IP به صورت یک رشته مثل "192.34.221.108" به متود ارسال می‌شود. مثال :

```
try {
 Socket sock = new Socket("128.252.120.1", 25);
}
catch ( UnknownHostException e ) {
   System.out.println("Can't find host.");
}
catch ( IOException e ) {
   System.out.println("Error connecting to host.");
}
```

ج) متود **close()** : این متود ضمن ختم ارتباط TCP بصورت دو طرفه ، سوکت را بسته و منابع تخصیص داده شده را آزاد خواهد کرد.

د ) متودهای **getInputStream()** و **getOutputStream()** بـرای آنسـت کـه بتـوان اسـتریمهای ورودی و خروجی نسبت داده شده به سوکت را بدست آورده و بتـوان از آن خوانـد یـا در آن نوشت.[1] به مثال زیر دقت کنید:

```
try
{
Socket server = new Socket("foo.bar.com", 1234);
InputStream in = server.getInputStream();
OutputStream out = server.getOutputStream();
// Write a byte
out.write(42);
// Say "Hello" (send newline delimited string)
PrintStream pout = new PrintStream( out );
pout.println("Hello!");
// Read a byte
Byte back = in.read();
// Read a newline delimited string
DataInputStream din = new DataInputStream( in );
String response = din.readLine();
server.close();
}
catch (IOException e ) { }
```

[1] دقت کنید که مفهوم استریم ورودی و خروجی ( Input Stream/Output Stream ) در زبانهای شیئ گرا را با مفهوم سوکتهای نوع استریم اشتباه نکنید. InputStream و OutputStream دو کلاس از کلاسـهای جاوا ( یـا C++ ) هستند.

در این مثال پس از ایجاد یک سوکت ، تلاش می‌شود تا با ماشینی با نام حوزهٔ foo.bar.com و شمارهٔ پورت ۱۲۳٤ ارتباط TCP برقرار شود. در صورت موفقیت‌آمیز بودن این عمل ، استریمهای ورودی و خروجی ایجاد شده برای سوکت ، بدست می‌آید (در استریمهای in و out ) تا بتوان روی آنها عملیاتِ ورودی / خروجی انجام داد. نهایتاً پس از ارسال و دریافت ، از طریق این استریمها سوکت بسته می‌شود.

حال به کلاس سوکتی که در سمت سرویس‌دهنده باید ایجاد شود و برخی متودهای آن ، دقت کنید :

**ServerSocket( int port)**
**ServerSocket(int port, int count)**
**synchronized void close()**
**Socket accept()**

متودهای اول و دوم به گونه‌ای که از ظاهر آنها پیداست ، متودهای **سازنده** هستند و پارامتر port ، آدرس (شمارهٔ) پورتی است که باید در سمت سرویس‌دهنده به سوکت مقید (bind) شود. در متود دوم پارامتر count زمان انتظار برای گوش دادن به پورت جهت برقراری ارتباط است. در ServerSocket ، بطور درونی و خودکار عمل گوش دادن به پورت (listen) ، انجام می‌شود. متود accept دقیقاً طبق مفهومی که در ابتدای فصل عنوان شد ، یکی از ارتباطات معلق را برای پردازش به برنامه هدایت می‌کند. مقدار برگشتی این متود ، مشخصهٔ یک شیئ سوکت است که می‌توان استریمهای ورودی/خروجی آنرا بدست آورده و روی آنها ارسال یا دریافت داشت. در مثال زیر که متناظر با برنامهٔ مثال قبلی است (یعنی در این دو مثال یکی سرویس‌دهنده و دیگری مشتری است) ، پس از ایجاد یک سوکت و مقید کردن شمارهٔ پورت ۱۲۳٤ ، یکی از ارتباطات معلق (در صورت وجود) پذیرفته شده و پس از عملیات ارسال و دریافت ، آن ارتباط بسته خواهد شد.

```
// Meanwhile, on foo.bar.com...
try {
   ServerSocket listener = new ServerSocket( 1234 );

   while ( !finished ) {
      Socket aClient = listener.accept();    // wait for connection

      InputStream in = aClient.getInputStream();
      OutputStream out = aClient.getOutputStream();

      // Read a byte
```

```
      Byte importantByte = in.read();

      // Read a string
      DataInputStream din = new DataInputStream( in );
      String request = din.readLine();

      // Write a byte
      out.write(43);

      // Say "Goodbye"
      PrintStream pout = new PrintStream( out );
      pout.println("Goodbye!");

      aClient.close();
    }

    listener.close();

}
catch (IOException e ) { }
```

از بین متودهای متنوعی که در کلاس Socket تعریف شده ، دو متود زیر در برخی از کاربردها بسیار مفیدند :

- **getport()** : این متود که متعلق به کلاس Socket است ، شمارهٔ پورت انتخاب شده برای سوکت را بر می‌گرداند.
- **getHostName()** : این متود نام ماشین (نام نمادین) متناظر با یک سوکت را بر می‌گرداند. این دو تابع زمانی مفید است که در سمت سرویس‌دهنده ، برنامه بخواهد با پذیرفتن یک ارتباط و دریافت شیئ Socket متناظر با آن ، هویت طرف مقابل ارتباط را تشخیص بدهد.

کلاسهای Socket و ServerSocket در جاوا بسیار ساده هستند و براحتی می‌توان آنها را مورد استفاده قرار داد. ( در مورد سوکتهای دیتاگرام فعلاً مطلبی را مطرح نمی‌کنیم. در صورت تمایل به مراجع فصل مراجعه نمایید. )

## ۱۳) مراجع این فصل

مجموعهٔ مراجع زیر می‌توانند برای دست آوردن جزییات دقیق و تحقیق جامع در مورد مفاهیم معرفی شده در این فصل مفید واقع شوند.


| |
|---|
| **Beej's Guide to Network Programming Using Internet Sockets, 1998. http://www.ects.csuchico.edu/~beej/guide/net** |
| **Unix Network Programming, volumes 1-2 by W. Richard Stevens. Prentice Hall.** |
| **Internetworking with TCP/IP, Commer D.E. ,Prentice-Hall, 1996.** |
| **Exploring Java, Patrick Niemeyer & Joshua Peck; 1-56592-184-271-9, 2nd Edition July 1997 (est.)** |
| **Java 1.2 Unleashed, Jamie Jaworski , Macmillan Computer Publishing,1998** |
| **Java By Example, Clayton Walnum Copyright© 1996 by Que® Corporation** |

## ۱) پروتکل TelNet

یکی از قابلیتهای اولیه یونیکس آن بود که امکان "ورود به سیستم"[۱] را از راه دور برای کاربران فراهم می‌کرد؛ یعنی یک کاربر می‌توانست با در اختیار داشتن یک "ترمینال"[۲] از هر مکانی و با استفاده از یک خط ارتباطی همانند خط تلفن با سیستم ارتباط برقرار کرده و به سیستم وارد شود و از آن سرویس بگیرد. یک کاربر مجاز در این سیستم ، ابتدا ارتباط فیزیکی ترمینال خود را با کامپیوتر مرکزی برقرار کرده و پس از وارد کردن شمارهٔ شناسایی و کلمهٔ عبور ، توسط سیستم یونیکس احراز هویت می‌شود؛ در صورت مجاز شناخته شدن ، یک پروسهٔ خاص برای سرویس‌دهی به او ایجاد خواهد شد. در حقیقت با این قابلیت ، یک کاربر در هر کجای دنیا ، در صورتی که بتواند ارتباط فیزیکی خود را با مرکز کامپیوتر برقرار کند ، قادر خواهد بود از سیستم سرویس بگیرد؛ با این قابلیت ، تفاوتی بین یک کاربر که در مقرّ کامپیوتر مرکزی نشسته و یک کاربر راه دور وجود ندارد.[۳]

با ارزان و سریع شدن سخت‌افزار و توسعهٔ خدمات اینترنت ، کامپیوترهای شخصی به خانه‌ها راه یافتند ولی هنوز کاربرانی وجود دارند که نیازمند آن هستند تا بجای استفاده از ترمینال ، از طریق کامپیوتر شخصی خود به یک سیستم راه دور وارد شوند. به عنوان مثال فرض کنید که شما یک کامپیوتر شخصی پنتیوم با سیستم عامل MS-Windows 9x در اختیار دارید ولی دانشگاه شما دارای یک سیستم مینی‌کامپیوتر SUN با سیستم عامل یونیکس است. شما برنامه‌های کاربردی خود را در محیط یونیکس نوشته‌اید و همانجا ذخیره کرده‌اید. حال به فرض اگر خواستید در منزل خود همانند کسی که در مرکز کامپیوتر نشسته است به محیط یونیکس وارد شده و برنامه‌های خود را ویرایش یا اجرا نمایید ، نیازمند یک ترمینال سازگار با یونیکس هستید ولی تنها چیزی که در اختیار شماست یک کامپیوتر شخصی است. در اینجا برنامهٔ Telnet راهگشاست.

برنامهٔ TelNet یک ترمینال مجازی و سازگار با ترمینالهای حقیقی از سیستم سرویس‌دهنده ، بر روی کامپیوتر شما شبیه‌سازی می‌کند و اجازه می‌دهد به سیستم یونیکس وارد شده و با آن محاوره نمایید. برنامهٔ TelNet فرامینی را که صادر می‌کنید به نحو مناسبی به سمت کامپیوتر راه

---

[۱] Remote Login

[۲] ترمینال ابزاری است که شامل یک صفحه نمایش ، یک صفحه کلید و یک ابزار ارتباطی -همانند یک مودم- جهت مخابرهٔ داده‌ها می‌باشد. ترمینال را نمی‌توان یک کامپیوتر مستقل به حساب آورد.

[۳] یک کاربر در محیط یونیکس ، به روش "اشتراک زمانی" از سیستم سرویس می‌گیرد و باید محدودیتهایی که سیستم عامل بر او تحمیل می‌کند را بپذیرد ، چراکه ترمینال ، یک کامپیوتر مستقل نیست و هیچ امکاناتی به غیر از محاوره -Conversation- با سیستم عامل در اختیار کاربر نمی‌گذارد.

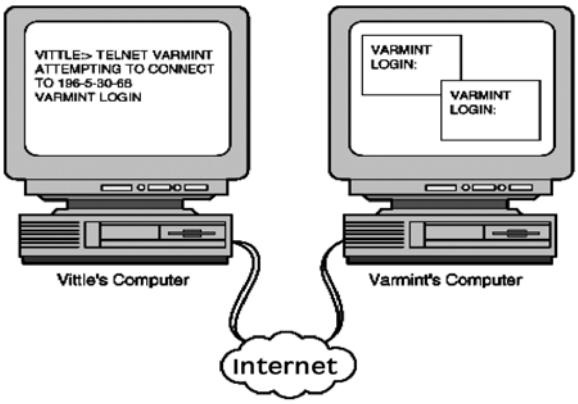
دور هدایت می‌کند و پس از تفسیر و اجرای فرمان صادره بر روی آن کامپیوتر ، نتیجه به برنامهٔ TelNet بر روی کامپیوتر شما باز خواهد گشت. بنابراین در یک تعریف ساده ، برنامهٔ TelNet موظف است بر روی ماشین کاربر ، مشخصه‌های ترمینال حقیقی سرویس‌دهنده را شبیه‌سازی نماید. به این ترمینال شبیه‌سازی شده اختصاراً NVT[1] گفته می‌شود.

در یک نگاه ساده برنامهٔ TelNet ، برنامهٔ ساده‌ای به نظر می‌رسد ، چرا که موظف است پس از برقراری یک "نشست"[2] فرمانهای کاربر را به سمت ماشین سرویس‌دهنده ارسال کرده و نتایج خروجی را نشان بدهد؛ ولی در مجموع برنامهٔ TelNet پیچیده‌تر از آنست که نشان می‌دهد ، چراکه موظف است با ترمینالهای متفاوت خود را تطبیق بدهد. به عنوان مثال فرض کنید یک کاربر از کامپیوتری با کدهای ASCII استفاده می‌کند در حالی که تمایل دارد به سیستمی وارد شود که استاندارد آن کدهای EBCDIC است؛ آگاهی از این موضوع و تبدیل این کدها به عهدهٔ برنامهٔ TelNet است. یا اگر کاربر یک برنامه به زبان C را بر روی سرویس‌دهنده اجرا کند ، نتایج خروجی منطبق با کنسول خروجی ماشین سرویس‌دهنده است نه ماشین خودش ، بنابراین تطبیق صفحه نمایش شبیه‌سازی شده ، به عهدهٔ برنامهٔ TelNet است.

مقصود از یک "نشست TelNet" برقراری موفق یک ارتباط TCP با پورت ۲۳ (یا یکی از پورتهای شناخته شده) از ماشین سرویس‌دهنده است به‌گونه‌ای که ماشین سرویس‌دهنده ضمن پذیرش این ارتباط و احراز هویت کاربر (در صورت لزوم) ، آمادهٔ پذیرش فرمانهای صادره از کاربر و اجرای آنها شود. در شکل (۱–۸) مراحل یک "نشست TelNet" به تصویر کشیده شده است. این نشست با اجرای برنامهٔ TelNet در خط فرمان آغاز می‌شود. در مثال زیر حروف پررنگ توسط کاربر نوشته شده و بقیه ، پیغامهای عمومی برنامهٔ TelNet هستند. در این مثال نام ماشین سرویس‌دهنده varmint و دارای سیستم عامل یونیکس و سخت‌افزار SUN است.

```
 telnet varmint
Trying 194.5.30.68 ...
Connected to varmint.
Escape character is '^]'.
SunOS UNIX (varmint)
login: anna
Password:******
varmint%
```

[1] Network Virtual Terminal
[2] TelNet Session




شکل (۱–۸) برقراری نشست Telnet

برنامهٔ TelNet در دو قسمت سازماندهی می‌شود:

- **پروسهٔ سرویس‌دهندهٔ TelNet** : این برنامه که بر روی کامپیوتر سرویس‌دهنده نصب و اجرا می‌شود ، موظف است تقاضاهای ورودی برای برقراری یک نشست TelNet را بپذیرد و پس از هماهنگی‌های لازم با برنامهٔ مشتری ، به او سرویس بدهد. این برنامه در محیط یونیکس به نام telnetd[۱] شناخته می‌شود.
- **پروسهٔ مشتری TelNet** : این برنامه که بر روی کامپیوتر کاربران نصب می‌شود و منطبق بر سخت‌افزار و سیستم عامل ماشین کاربر است وظیفه دارد تا مراحل برقراری یک نشست TelNet را برقرار کرده و یک ترمینال مجازی را به گونه‌ای شبیه‌سازی نماید که فرامین صادره از کاربر ، منطبق و سازگار با ماشین سرویس‌دهنده باشد. بطور عام این برنامه telnet نامیده شده است.[۲]

وقتی یک نشست TelNet برقرار شد کاربر می‌تواند یک فرمان را به سمت سرویس‌دهنده ارسال نماید. حال هر کلید یا فرمانی که بر روی کامپیوتر او فشار داده می‌شود ، باید از پروسه‌های گوناگونی عبور نماید تا تحویل برنامهٔ کاربردی شود. این روال بصورت زیر است:

---

[۱] Telnet Daemon

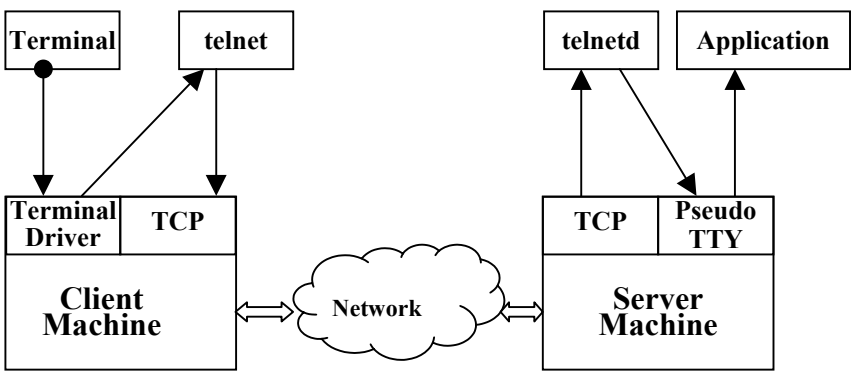
[۲] در محیط یونیکس برنامه **rlogin** نیز شرایط ورود به سیستم را از راه دور ، فراهم می‌نماید.

الف : ابتدا برنامهٔ TelNet بـر روی کـامپیوتر کـاربر ، آن کـاراکتر را تحویـل گرفتـه و پـس از پردازش لازم ( از لحاظ تغییر استاندارد کد) از آن یک بستهٔ TCP ساخته و از طریـق لایـه‌های زیرین آنرا به سمت ماشین مقصد به جریان می‌اندازد و نهایتاً در مقصد تحویل سـرویس‌دهندهٔ TelNet می‌شود.

ب : برنامهٔ TelNet بر روی کامپیوتر سرویس‌دهنده آنرا تحویل گرفته و در صورت لزوم پس از تبدیل کد و با کمک سیستم عامل ، در اختیار برنامهٔ کاربردی قرار می‌دهد.

ج : برنامهٔ کاربردی ضمن پردازش آن ، خروجی مناسب را تولید و به سمت برنامـهٔ TelNet در سمت کاربر هدایت می‌نماید.

در شکل (۲-۸) جریان عبور داده‌ها از بین پروسه‌های مختلف بیـن برنامـهٔ مشـتری تـا برنامـهٔ کاربردی در ماشین سرویس‌دهنده ، نشان داده شده است.




شکل (۲-۸) جریان عبور داده‌ها بین پروسه‌های مبدأ و مقصد در TelNet

حال فرض کنید که سیستم عامل در کامپیوتر سرویس‌دهنده ، مبتنی بر "پنجره"[۱] باشد بدین معنا که بجای صدور فرامین در خط فرمان ، کاربر در یک محیط گرافیکی ( شامل پنجره ، منـو ، آیکون و ... ) از سیستم سـرویس بگـیرد. در اینجـا اتصـال بـه سیسـتم از راه دور ، شـرایطِ پیچیده‌تری دارد. زیرا این محیط گرافیکی باید بر روی کامپیوتر کاربر شبیه‌سازی شود؛ ولی هر

[۱] Window Based OS

عملی که کاربر بر روی این محیط انجام می‌دهد باید توسط سرویس‌دهندهٔ راه دور پردازش شود. در چنین شرایطی ، حرکت موْس ، هرگونه کلیک یا فشار کلیدهای صفحه کلید باید سریعاً به سمت سرویس‌دهنده ارسال و پاسخ آن از سرویس‌دهنده برگردد.[۱] در این محیط حجم اطلاعاتی که بین ماشین محلی و ماشین راه دور مبادله می‌شود بسیار زیادتر از برنامه‌های مبتنی بر "خط فرمان" است. چرا که هر حرکت مکان‌نمای موْس باید به سرویس‌دهنده گزارش شود. ( برنامه‌های TelNet برای اتصال به سرویس‌دهنده‌های مبتنی بر پنجره وجود دارند ، همانند Motif یا X Windows . )

برای آنکه برنامهٔ TelNet را در خط فرمان اجرا نمایید ، ابتدا اسم برنامه و نام ماشین سرویس‌دهنده نوشته می‌شود. مثال:

telnet **leo.nmc.edu**

یا telnet **205.150.89.1**

telnet **varmint**

اگر از نامهای نمادین بجای آدرس IP استفاده می‌کنید باید این آدرسها قابل تحلیل و ترجمه به آدرس IP باشد.[۲]

```
merlin> telnet tpci_hpws4
Trying...Connected to tpci_hpws4.
Escape character is '^]'.
HP-UX tpci_hpws4 A.09.01 A 9000/720 (ttys2)
login: tparker
password: ********
tpci_hpws4-1 $  pwd
/u1/tparker
tpci_hpws4-2 $ cd docs
tpci_hpws4-3 $ pwd
/u1/tparker/docs
```

[۱] به عنوان مثال وقتی شما در این محیط بر روی یک منوی **Pull Down** کلیک می‌کنید هیچ پنجره‌ای ظاهر نمی‌شود مگر آنکه سرویس‌دهندهٔ راه دور اجازه بدهد.

[۲] اگر برنامهٔ **Telnet** را بدون مشخص کردن نام ماشین سرویس دهنده اجرا کنید ، این برنامه اجرا می‌شود ولی منتظر می‌ماند تا فرامین داخلی **Telnet** را اجرا نمایید. فرامین داخلی فرامینی هستند که به برنامهٔ **Telnet** مربوط می‌شوند و بر روی شبکه ارسال نخواهند شد.

```
tpci_hpws4-4 $ <Ctrl+d>
Connection closed by foreign host.
merlin>
```

وقتی برنامهٔ Telnet را برای ورود به یک سیستم راه دور بکار می‌گیرید ، پس از برقراری ارتباط ، از شما یک کد کاربری و یک کلمهٔ عبور خواسته می‌شود و در صورت تصدیق هویت ، علامت خط فرمان یونیکس[۱] روی خروجی ظاهر می‌شود. با دیدن این علامت هر فرمانی که روی خط فرمان صادر می‌نمایید به سمت سرویس‌دهنده ارسال خواهد شد. در مثال بالا پس از برقراری ارتباط ، پیغام زیر روی خروجی نشان داده شده است:

```
Escape character is '^]'.
HP-UX tpci_hpws4 A.09.01 A 9000/720 (ttys2)
```

این پیغام به این معناست که شما پس از برقراری نشست TelNet ، هر فرمانی که صادر کنید به سمت سرویس‌دهنده ارسال خواهد شد ولی اگر خواستید که فرامین کاربری Telnet را اجرا کنید می‌توانید با فشار دادن کلیدهای [+Ctrl به "حالت فرامین کاربری" بروید. در این حالت هر فرمانی که صادر می‌شود بجای ارسال به سرویس‌دهنده بصورت محلی و توسط برنامهٔ TelNet تحلیل خواهد شد. برای ختم یک نشست TelNet از کلیدهای Ctrl+D استفاده می‌شود.[۲]

در خط دوم از پیغام ، ضمن معرفی ماشین سرویس‌دهنده ، پارامترهای استاندارد ترمینال که بر روی آن توافق شده اعلان گردیده است. (ttys2)

بگونه‌ای که دیده می‌شود پس از برقراری نشست TelNet فرامین سیستم عامل یونیکس بر روی ماشین راه دور ، صادر و اجرا شده است.

برای برقراری ارتباط با یک سیستم عامل مبتنی بر پنجره :

اولاً باید سرویس‌دهندهٔ مورد نظر کاربر ، چنین سرویسی را فراهم آورده باشد.

ثانیاً برنامهٔ TelNet بر روی ماشین کاربر باید چنین قابلیتی را داشته باشد.

ثالثاً طرفین بر روی پارامترهای استاندارد توافق کنند تا بتوانند فرامین مربوط با پنجره را تحلیل کرده و خروجی لازم را نشان بدهد.

---

[۱] UNIX Prompt

[۲] **ختم یک نشست TelNet ، با استفاده از Ctrl+D ، در حقیقت معادل با عمل خروج از سیستم –logout– است.**

از TelNet می‌توان برای برقراری ارتباط با سرویس‌دهنده‌های دیگری مثل سرویس‌دهندهٔ HTTP ( پورت TCP شمارهٔ ۸۰ ) بهره برد. در این حالت پس از برقراری نشست می‌توان فرامین پروتکل HTTP را ارسال کرد. مثال:

```
telnet www.csc.com 80
Trying 18.23.0.23 …
Connected to www.csc.com.
Escape character is '^]'.
GET /client-server-computing/toc.html HTTP/1.0
```

در مثال فوق شمارهٔ پورت پروسهٔ مقصد که باید نشست با آن برقرار شود ، ۸۰ معرفی شده است ولی اگر در خط فرمان شمارهٔ پورت تعیین نشود بصورت پیش‌فرض پورت TCP شمارهٔ ۲۳ که متعلق به پروسهٔ سرویس‌دهندهٔ TelNet است در نظر گرفته خواهد شد. سطر آخر در مثال بالا توسط کاربر نوشته شده و یک فرمان معتبر برای سرویس‌دهندهٔ HTTP محسوب می‌شود.

## ۲) فرامین TelNet

فرامین در TelNet بر دو نوعند:

الف) فرامین داخلی : این فرامین دارای قالب استاندارد و جهانی هستند و بین سرویس‌دهندهٔ TelNet در ماشین راه دور و برنامهٔ مشتری مبادله می‌شوند و کاربر دخالتی در مبادلهٔ این فرامین نخواهد داشت بلکه فقط در صورت تمایل می‌تواند مبادلهٔ آنها را ببیند.

ب ) فرامین کاربری : این فرامین یکسری از دستورات کاربری در محیط TelNet هستند و با صدور آنها کاربر می‌تواند با برنامهٔ TelNet در ماشین خود "محاوره"[۱] داشته باشد. در مثال زیر فرامینی که در زمینهٔ خاکستری نشان داده شده‌اند فرامین کاربری در برنامهٔ TelNet هستند.

```
tpci_server-1> telnet
telnet> toggle options
Will show option processing.
telnet> open tpci_hpws4
Trying...
```

[۱] Converse

```
Connected to tpci_hpws4.
Escape character is '^]'.
SENT do SUPPRESS GO AHEAD
SENT will TERMINAL TYPE (don't reply)
SEND will NAWS (don't reply)
RCVD do 36 (reply)
sent won't 36 (don't reply)
RECD do TERMINAL TYPE (don't reply)
RCVD will SUPPRESS GO AHEAD (don't reply)
RCVD do NAWS (don't reply)
Sent suboption NAWS 0 80 (80) 0 37 (37)
Received suboption Terminal type - request to send.
RCVD will ECHO (reply)
SEND do ECHO (reply)
RCVD do ECHO (reply)
SENT won't ECHO (don't reply)
HP-UX tpci_hpws4 A.09.01 A 9000/720 (ttys2)
login: tparker
password: ********
tpci_hpws4-1 $
```

در این مثال ، فرمان کاربریِ toggle options باعث شده است که جریان مبادلهٔ فرامین داخلی بین سرویس‌دهندهٔ TelNet و برنامهٔ محلی ، به کاربر نشان داده شود. فرمان کاربری open نیز از برنامهٔ TelNet خواسته است که با یک ماشین خاص ارتباط برقرار کند.

با توجه به آنکه فرامین و داده‌ها در TelNet همگی روی یک کانال و بصورت رشته‌ای از کاراکترها ارسال می‌شوند لذا برای تمایز بین داده‌ها و فرامین ، یکسری کدهای فرمان تعریف

شده است. هر فرمان ابتدا با "کاراکترِ کد ۲۵۵"[1] شروع می‌شود و سپس کد فرمان بعد از آن قرار می‌گیرد و نهایتاً در صورت لزوم یکی از کدهای عمل اختیاری قرار می‌گیرد. پس بطور کلی فرامین داخلی در TelNet قالبی بصورت زیر دارند:

| IAC[2] | Command Code | Option Code |
|---|---|---|

مثال :

255 , 243 , 1

دقّت کنید در مثال بالا فرمان جمعاً سه بایت است که مقادیر عددی هر یک از کدها نوشته شده است؛ در ضمن کد عمل اختیاری می‌تواند حذف شود.

در جدول (٤-٨) برخی از کدهای فرمان که پس از کد ۲۵۵ قرار می‌گیرد ، تعریف شده‌اند. در جدول (٥-٨) نیز کدهای عمل اختیاری ارائه شده است.

شرح بعضی از کدهای فرمان در زیر آمده است:

- Will , Won't , Do , Don't : این چهار کد ، اصلیترین کدهای فرمان هستند که برای توافق و هماهنگی سرویس‌دهنده و برنامهٔ TelNet روی ماشین محلی کاربرد دارند. با این کدها یکی از طرفین ، گزینه‌ای را پیشنهاد می‌دهد و طرف دیگر می‌تواند آنرا پذیرفته یا رد کند.

**Will با کد ۲۵۱** : با این کد یکی از طرفین می‌تواند عملی را به طرف دیگر پیشنهاد بدهد. مثال :

| Character Code 255 | Character Code 251 | Character Code 1 |
|---|---|---|
| IAC | Will | Echo |

معادل نمادین :

با ارسال این سه کد از سرویس‌دهنده خواسته شده است که پس از دریافت هر کاراکتر آنرا برگشت بدهد.[3]

**Won't با کد ۲۵۲** : با این کد یکی از طرفین تقاضای عدم انجام یا لغو یک عمل را می‌نماید. مثال:

| Character Code 255 | Character Code 252 | Character Code 1 |
|---|---|---|
| IAC | Won't | Echo |

[1] Escape Code
[2] Interpret As Command
[3] معادل **Echo On**

با ارسال این سه کد از سرویس‌دهنده خواسته شده است که پس از دریافت هر کاراکتر آنرا برگشت ندهد.[۱]

Do : به معنای پذیرش تقاضای داده شده یا اعلام یک عمل انجام شده می‌باشد.[۲]

Don't : به معنای عدم پذیرش تقاضای داده شده یا اعلام لغو یک عمل می‌باشد.[۳]

- Interrupt Process (IP) : بسیاری از سیستمها به کاربر اجازه می‌دهند که کاربر بتواند برنامهٔ در حال اجرای خود را متوقف نماید. با ارسال این کد ، سرویس‌دهنده می‌تواند پروسهٔ در حال اجرای کاربر را متوقف نماید.
- Abort Output (AO) : فرض کنید کاربر برنامه‌ای را اجرا کرده که ضمن اجرا ، تولید خروجی می‌نماید. به عنوان مثال برنامه‌ای ضمن مرتب کردن رکوردهای درون یک فایل ، آنها را روی صفحه نمایش نشان می‌دهد. در این حالت کاربر می‌تواند با ارسال این کدها از سرویس‌دهنده بخواهد ، در حالی که اجرای برنامه را ادامه می‌دهد ، نتیجهٔ خروجی صفحه نمایش را برای او ارسال ننماید.
- Break (BRK) : ارسال این کد دقیقاً معادل آن است که شما در هنگام اجرای یک پروسه کلیدهای Ctrl+Break را فشاربدهید.
- Are You There (AYT) : فرض کنید که یک سرویس‌دهنده به دلایلی نظیر بار زیاد ، برای مدت طولانی سرویس‌دهی به کاربر را به تعویق بیاندازد. برنامهٔ TelNet با ارسال این کد می‌تواند مطمئن شود که آیا عدم پاسخگویی ناشی از بار زیاد است یا آنکه یک مشکل حادّ نظیر قطع ارتباط یا خرابی سرویس‌دهنده وجود دارد.
- Erase Line (EL) : با ارسال این کد از سرویس‌دهنده می‌خواهد که خط ارسال شدهٔ قبلی را از بافر ورودی خود حذف نماید.
- Erase Character (EC) : با ارسال این کد از سرویس‌دهنده می‌خواهد که کاراکتر ارسال شدهٔ قبلی را از بافر ورودی خود حذف نماید. ( همانند عملی که کد Backspace انجام می‌دهد.

اگر با دستور toggle option ، حالت مشاهدهٔ فرامین را فعال کنید ممکن است پیغامهایی نظیر مثال زیر را ببینید :

RCVD will ECHO

---

۱ معادل **Echo Off**
۲ معادل **Positive Acknowledgement**
۳ معادل **Negative Acknowledgement**

یعنی تقاضای فعال شدن حالت ECHO از طرف مقابل دریافت شد.

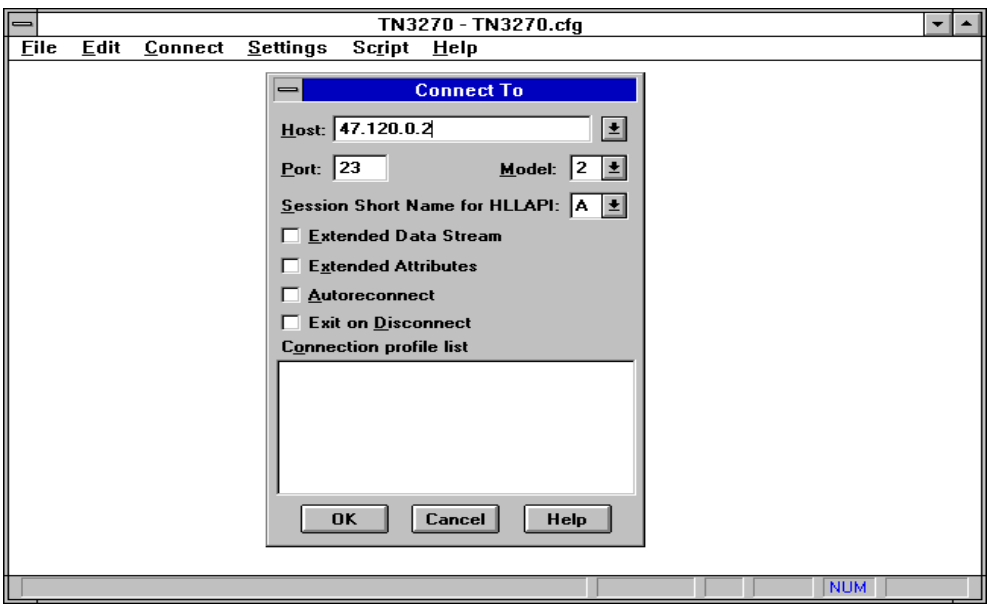
SEND do ECHO

یعنی تقاضای فعال شدن حالت ECHO ، پذیرفته و اعمال شد.

برای توضیح بیشتر در مورد عملکرد این کدها به مراجع معرفی شده در آخر این فصل مراجعه کنید.

### TN3270 (۳–۱

اغلب کامپیوترهای چندکاربرهٔ بزرگ[۱] از کدهای EBCDIC استفاده می‌نمایند در حالی که ریزکامپیوترها معمولاً از استاندارد ASCII بهره برده‌اند. بنابراین برای ورود به یک سیستم راه دور باید تبدیل کد انجام شود به همین دلیل برنامهٔ TelNet با قابلیت تغییر مجموعهٔ کد با نام TN3270 ارائه شد. در شکل (۳–۸) برنامهٔ TN3270 از مجموعهٔ نرم‌افزار NetManage ChameleonNFS نشان داده شده است که از طریق آن سعی شده است با یک ماشین با آدرس 47.120.0.2 و استاندارد EBCDIC ارتباط برقرار شود.




شکل (۳–۸) برنامهٔ TN3270 از مجموعهٔ نرم‌افزار NetManage ChameleonNFS

[۱] Multiuser Mainframe Computer

| نام کد | مقدار | توضیح |
|---|---|---|
| **Abort Output (AO)** | 245 | Runs process to completion but does not send the output |
| **Are you there (AYT)** | 246 | Queries the other end to ensure that an application is functioning |
| **Break (BRK)** | 243 | Sends a break instruction |
| **Data Mark** | 242 | Data portion of a Sync |
| **Do** | 253 | Asks for the other end to perform or an acknowledgment that the other end is to perform |
| **Don't** | 254 | Demands that the other end stop performing or confirms that the other end is no longer performing |
| **Erase Character (EC)** | 247 | Erases a character in the output stream |
| **Erase Line (EL)** | 248 | Erases a line in the output stream |
| **Go Ahead (GA)** | 249 | Indicates permission to proceed when using half-duplex (no echo) communications |
| **Interpret as Command (IAC)** | 255 | Interprets the following as a command |
| **Interrupt Process (IP)** | 244 | Interrupts, suspends, aborts, or terminates the process |
| **NOP** | 241 | No operation |
| **SB** | 250 | Subnegotiation of an option |
| **SE** | 240 | End of the subnegotiation |
| **Will** | 251 | Instructs the other end to begin performing or confirms that this end is now performing |
| **Won't** | 252 | Refuses to perform or rejects the other end performing |

جدول (٤-٨) برخی از کدهای فرمان در TelNet

| نام کد | توضیح |
|---|---|
| 0 | Binary transmission |
| 1 | Echo |
| 2 | Reconnection |
| 3 | Suppress Go Ahead (GA) |
| 4 | Approximate message size negotiation |
| 5 | Status |
| 6 | Timing mark |
| 7 | Remote controlled transmission and echo |
| 8 | Output line width |
| 9 | Output page length |
| 10 | Output carriage-return action |
| 11 | Output horizontal tab stop setting |
| 12 | Output horizontal tab stop action |
| 13 | Output form feed action |
| 14 | Output vertical tab stop setting |
| 15 | Output vertical tab stop action |
| 16 | Output line feed action |
| 17 | Extended ASCII characters |
| 18 | Logout |
| 19 | Bytes macro |
| 20 | Data entry terminal |
| 21 | SUPDUP |
| 22 | SUPDUP output |
| 23 | Send location |
| 24 | Terminal type |
| 25 | End of Record |
| 26 | TACACS user identification |
| 27 | Output marking |
| 28 | Terminal location number |
| 29 | 3270 regime |
| 30 | X.3 PAD (Packet assembly and disassembly) |
| 31 | Window size |

جدول (۵–۸) کدهای اعمال اختیاری

## ۳) پروتکل انتقال فایل ( FTP )

پروتکل انتقال فایل که از این به بعد آن را FTP می‌نامیم ابزاریست مطمئن برای انتقال فایل بین کامپیوترهائی که به شبکهٔ اینترنت متصل هستند. خدماتی که این پروتکل ارائه می‌کند عبارتند از:

- تهیهٔ لیستی از فایلهای موجود از سیستم فایل کامپیوتر راه دور
- حذف، تغییر نام و جابجا کردن فایلهای کامپیوتر راه دور
- جستجو در شاخه‌های (دایرکتوریهای) کامپیوتر راه دور
- ایجاد یاحذف شاخه روی کامپیوتر راه دور
- انتقال فایل از کامپیوتر راه دور به کامپیوتر میزبان[۱]
- انتقال فایل و ذخیرهٔ آن از کامپیوتر میزبان به کامپیوتر راه دور[۲]

قابلیتهائی که پروتکل FTP عرضه می‌کند می‌تواند برای سیستم سرویس دهنده بسیار خطرناک باشد چرا که بسادگی می‌توان فایلهای یک کامپیوتر راه دور را آلوده یا نابود کرد. فلذا در این پروتکل کاربران باید قبل از تقاضای هر سرویسی کلمهٔ عبور خود را وارد نمایند و سرویس دهنده پس از شناسائی کاربر، سطح دسترسی و عملیات مجاز برای کاربر را تعیین می‌کند و یک نشست[۳] FTP آغاز می‌شود.

FTP این قابلیت را ندارد که بتوان همانند پروتکل Telnet برنامه‌ای را بر روی ماشین راه دور اجرا کرد بلکه فقط روشی سریع، ساده و مطمئن برای خدمات فایل به کاربران راه دور محسوب می‌شود. حال باید ارتباط بین سرویس دهنده و سرویس گیرندهٔ FTP را تشریح نماییم:

در پروتکل FTP برای شروع یک "نشست" بین برنامه سرویس دهنده و برنامهٔ سرویس گیرنده باید دو ارتباط همزمان از نوع TCP برقرار شود. به هر یک از این ارتباطات در ادبیات پروتکل FTP، "کانا ل" گفته می‌شود. این دو کانال عبارتند از :

- کانا ل داده: یک ارتباط TCP با پورت شماره ۲۰ از سرویس دهنده که روی آن داده‌ها (مثلاً بلوکهای یک فایل) مبادله می‌شوند.
- کانال فرمان: یک ارتباط TCP با پورت شماره ۲۱ که روی آن فرامین لازم برای مدیریت فایلها رد و بدل می‌شوند.

---

۱ Downloading
۲ Uploading
۳ Session

دلیل لزوم برقراری دو کانال مجزا بین سرویس‌دهنده و سرویس‌گیرنده آن است که بتوان بدون قطع جریان داده‌ها فرامین را بطور همزمان مبادله کرد. بعنوان مثال در حین انتقال یک فایل می‌توان روی کانال فرمان، دستور لغو عمل انتقال یا تغییر مود انتقال را صادر کرد.
ذکر این نکته ضروری است که در پروتکل FTP همهٔ عملیات انتقال فایل در "پیش زمینه"[۱] انجام می‌شود بدین معنا که پروتکل FTP از سیستم Spooler یا صف برای انتقال فایلها استفاده نمی‌کند بلکه عملیات انتقال به صورت بلادرنگ انجام می‌گیرد. (سیستمهائی مثل مدیریت چاپ در "پس زمینه"[۲] عمل می‌کند یعنی وقتی پروسه‌ای تقاضای چاپ یک سند را می‌دهد سیستم عامل آنرا به صف می‌کند تا در موقع مناسب آن را چاپ نماید فلذا مشخص نیست از زمان صدور فرمان چاپ چه مدت طول بکشد تا سند چاپ شود چرا که اولویت با پروسه هائی است که در پیش زمینه اجرا میشوند.)

بگونه‌ای که اشاره شد سرویس دهندهٔ FTP بایستی دو پروسه همزمان ایجاد نماید که یکی وظیفهٔ مدیریت ارتباط روی کانال فرمان را به عهده داشته و اصطلاحا "مفسر پروتکل" یا پروسه PI[۳] نامیده می‌شود. وظیفهٔ پروسهٔ دیگر مدیریت انتقال داده‌ها است و به DTP[٤] یا "پروسه انتقال داده" معروف است. پروسهٔ PI همیشه به پورت شماره۲۱ گوش می‌دهد و پروسه DTP به پورت شماره ۲۰ مقید شده است.

### ۱-۳) روشهای برقراری یک نشست FTP

برقراری ارتباط بین سرویس‌دهنده و سرویس گیرندهٔ FTP با دو روش امکان‌پذیر است:

- روش معمولی یا Normal Mode
- روش غیرفعال یا Passive Mode

در روش معمولی برای برقراری یک نشست FTP مراحل زیر انجام می‌شود:

الف) در برنامه سمت سرویس گیرنده (برنامه سمت مشتری) ابتدا دو سوکت نوع TCP با شماره پورت تصادفی بالای ۱۰۲٤ ایجاد می‌شود.

ب) در مرحله دوم برنامهٔ سمت مشتری سعی می‌کند با استفاده از دستور ()connect ارتباط یکی از سوکتهای ایجاد شده را با پورت شماره ۲۱ از سرویس‌دهنده برقرار نماید. اگر این

---

[۱] Foreground
[۲] Background
[۳] Protocol Interpreter
[٤] Data Transfer Protocol

ارتباط برقرار شود در حقیقت کانال فرمان باز شده و پروسه PI آمادهٔ تفسیر فرامین صادره از سمت مشتری می‌باشد.

ج) برنامهٔ سمت مشتری با فرمان "PORT" به برنامهٔ سمت سرویس دهنده شمارهٔ پورت سوکت دوم را اعلام می‌نماید و منتظر می‌ماند. (در حقیقت برنامه مشتری روی سوکت دوم عمل ()listen انجام می‌دهد)

د) در ادامه برنامهٔ سرویس دهنده سعی می‌کند یک ارتباط با TCP با شماره پورت اعلام شده برقرار نماید. یکی از نکات عجیب در این پروتکل آنست که سرویس دهنده FTP موظف است اقدام به برقراری یک ارتباط TCP از طریق دستور ()connect با برنامه مشتری نماید در صورتی که معمولاً سرویس دهنده پذیرندهٔ ارتباط است نه شروع کنندهٔ ارتباط.

ه) برنامهٔ سمت مشتری ارتباط TCP شروع شده از سرویس دهنده را تصدیق کرده و یک نشست FTP آغاز می‌شود.

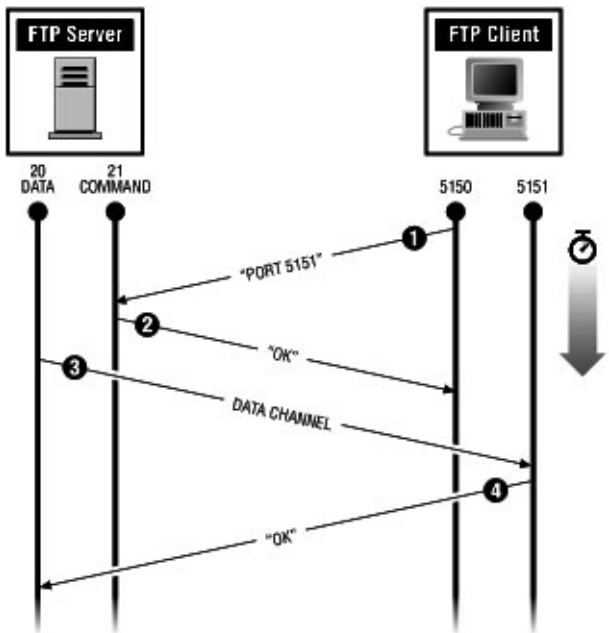
در شکل (٦–٨) با یک مثال مراحل فوق به تصویر کشیده شده است. ابتدا برنامهٔ سمت مشتری دو سوکت باز کرده و شمارهٔ پورتهای ۵۱۵۰ و ۵۱۵۱ را به آنها نسبت داده است. (با دستور ()bind ) سپس از طریق سوکت اول یک ارتباط TCP با پورت ۲۱ از سرویس‌دهنده برقرار کرده و پس از برقراری ارتباط با ارسال فرمان "PORT 5151" شمارهٔ پورت سوکت دوم خود را اعلام کرده است. سمت سرویس دهنده ضمن تصدیق پذیرش یک نشست بلافاصله اقدام به برقراری یک ارتباط TCP بین پورت شمارهٔ ۲۰ خودش و پورت شمارهٔ ۵۱۵۱ از سرویس گیرنده نموده است. با تصدیق این ارتباط نشست FTP آغاز می‌شود.

حال باید روش "غیرفعال" را در برقراری یک نشست FTP بررسی نمائیم:

الف) در برنامهٔ سمت مشتری ابتدا دو سوکت نوع TCP با شماره پورت تصادفی بالای ١٠٢٤ ایجاد می‌شود.

ب) برنامهٔ سمت مشتری سعی می‌کند ارتباط TCP یکی از سوکتهای ایجاد شده را با پورت شمارهٔ ۲۱ از سرویس دهنده برقرار نماید. با برقراری این ارتباط کانال فرمان باز شده و پروسه PI آمادهٔ تفسیر فرامین صادره از سمت مشتری خواهد شد.

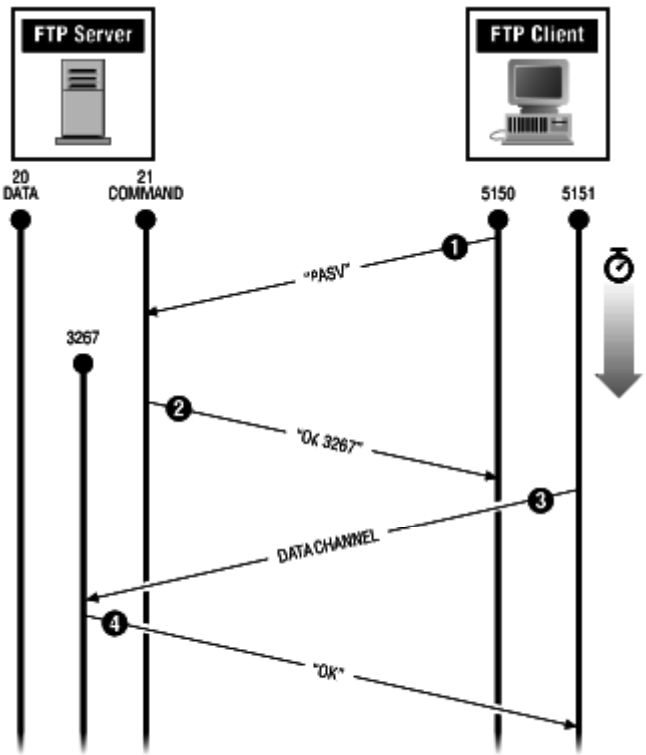
ج) برنامهٔ سمت مشتری با فرمان PASV به برنامهٔ سمت سرویس دهنده اعلام می‌کند که خواستار یک نشست از نوع غیر فعال است.




شکل (٦–٨) مثالی از یک نشست FTP به روش معمولی

د) برنامهٔ سمت سرویس دهنده یک سوکت با شماره پورت تصادفی (بالای ١٠٢٤) ایجاد کرده و شمارهٔ آنرا به برنامهٔ مشتری اعلام می‌نماید.
ه) برنامهٔ سمت مشتری ارتباط سوکت دوم خود را با شماره پورت اعلام شده برقرار کرده پس از تصدیق ارتباط نشست FTP آغاز می‌شود.

در شکل (٧–٨) مثالی از برقراری نشست غیرفعال تصویر شده است: ابتدا برنامهٔ سمت مشتری دو سوکت باز کرده و شماره پورتهای ٥١٥٠ و ٥١٥١ را به آنها نسبت داده است. سپس از طریق سوکت اول یک ارتباط TCP با پورت ٢١ از سرویس دهنده برقرار شده و با ارسال فرمان "PASV" منتظر تصدیق و اعلام شمارهٔ پورت از سرویس دهنده می‌ماند. در این مثال سرویس دهنده ضمن اعلام شماره پورت ٣٢٦٧ گشوده شدن کانال فرمان را تصدیق می‌کند.




شکل (۷–۸) مثالی از یک نشست FTP به روش غیرفعال

مجدداً برنامهٔ سمت مشتری ارتباط سوکت دوم خود را با شماره پورت ۳۲٦۷ از سرویس دهنده برقرار نموده و در صورت تصدیق ارتباط ، نشست FTP آغاز می‌شود.

در دنیای یونیکس برنامه سمت سرویس دهنده بنام ftpd[1] و برنامه سمت مشتری بنام ftp مشهور است. دقت کنید که برخی از سرویس دهنده‌ها از روش غیر فعال حمایت نمی‌کنند و فقط روش معمولی را می‌پذیرند.

در برنامهٔ ftp دو نوع فرمان تعریف شده که گاهاً با هم اشتباه می‌شوند. نوع اول فرامینی هستند که روی کانال فرمان یعنی بین سرویس دهنده و سرویس گیرنده رد و بدل می‌شود. این

[1] FTP Daemon

فرامین که در بخش بعدی معرفی می‌شوند فرامین داخلی نام دارند. نوع دوم فرامینی هستند که بین کاربر و برنامه ftp تعریف شده و کاربر برای بکارگیری برنامه ftp باید آنها را بداند. (همانند فرامین DOS)

## ۴) فرامین داخلی[۱] FTP

اگر به مراحل برقراری نشست FTP دقت کنید متوجه خواهید شد که در هر دو روش ابتدا کانال فرمان برقرار می‌شود و تا پایان نشست این کانال برقرار خواهد ماند. در این بخش بررسی می‌کنیم که روی این کانال چه فرامینی مبادله می‌شود.

فرامین داخلی ftp تماماً حالت متنی داشته و حداکثر چهار حرفی هستند. برخی از فرامین دارای پارامتر هستند که این پارامترها نیز بصورت متنی پس از یک فاصلهٔ خالی در ادامهٔ فرمان قرار می‌گیرند. خاتمهٔ فرمان با کدهای \r[۲] و \n[۳] مشخص شده است. مثلاً:

"PASV\r\n"

اولین مزیت بهره‌گیری از فرامین متنی آنست که کاربر می‌تواند سلسلهٔ فرامین را دیده و به آسانی بفهمد و این مزیت به یک کاربر حرفه‌ای تا حد زیادی به درک روند عملیات و اشکالزدایی کمک خواهد کرد. دومین مزیت آنست که یک متخصص می‌تواند براحتی (بدون نیاز به برنامهٔ ftp و با ابزاری مثل Telnet یا برنامه نویسی سوکت) مستقیماً با سرویس دهنده ارتباط برقرار کند.

مجموعه فرامین ftp با توضیح مختصر در جدول (۸-۸) آمده است. بگونه‌ای که مشاهده می‌شود این فرامین برای سه دسته عملیات در نظر گرفته شده‌اند:

- فرآیند برقراری و ختم یک نشست ftp (مثل فرمان PASV ، PORT ، QUIT)
- ارسال کلمهٔ عبور جهت تعیین جواز و سطح سرویس‌دهی
- فرامین مربوط به انتقال فایل، فهرست شاخه‌ها و عملیات مدیریت فایلها

در پاسخ به هر فرمان صادره یک کد سه رقمی برمی‌گردد تا وضعیت اجرای فرمان را مشخص نماید. برنامهٔ سمت مشتری با پردازش این کد می‌تواند در مورد عمل بعدی خود تصمیم بگیرید. (تصمیماتی مثل لغو فرمان، تلاش مجدد، صدور فرمان جایگزین یا ختم نشست)

---

۱ FTP Internal Commands
۲ Carriage Return
۳ New Line

| فرمان | عملکرد فرمان ( بسیاری از فرامین با پارامتری که در جلو فرمان درج شده معنا می‌یابند ) |
|---|---|
| ABOR | تقاضای لغو و ناتمام رها کردن فرمان قبلی |
| ACCT | اعلام مشخصه کاربری |
| ALLO | تقاضای تخصیص حافظه و فضا برای عمل بعدی |
| APPE | تقاضای ضمیمه شدن داده های ارسالی به یک فایل موجود |
| CDUP | تقاضای تغییر زیرشاخه جاری به شاخه پدری آن ( معادل دستور cd.. در dos ) |
| CWD | تقاضای تغییر شاخه جاری به شاخه ای که نامش در جلو فرمان درج شده است |
| DELE | تقاضای حذف فایلی که نامش در جلو فرمان درج شده است |
| HELP | تقاضای راهنمایی و اخذ اطلاعات مفید در مورد فرامین |
| LIST | تقاضای انتقال فهرست شاخه ها |
| MKD | تقاضای ایجاد یک زیرشاخه در شاخه فعلی |
| MODE | تنظیم روش انتقال ( متنی و دودویی ) |
| NLST | تقاضای فهرست گیری از شاخه فعلی |
| NOOP | هیچ عملی انجام نمیدهد |
| PASS | ارسال کلمه عبور کاربر |
| PASV | تقاضا برای برقراری یک نشست غیر فعال |
| PORT | اعلام شماره پورت TCP برای برقراری کانال داده |
| PWD | تقاضای اعلام فهرست فایلهای شاخه جاری |
| QUIT | تقاضای ختم نشست FTP |
| REIN | تقاضای ختم نشست فعلی و برقراری یک نشست جدید |
| REST | تقاضای انتقال مجدد داده های فایل از ابتدای آن |
| RETR | تقاضای دریافت نسخه ای از یک فایل |
| RMD | تقاضای حذف یک شاخه |
| RNFR | نام قدیم یک فایل یا مسیری که باید تغییر نام بدهد |

| فرمان | عملکرد فرمان ( بسیاری از فرامین با پارامتری که در جلو فرمان درج شده معنا می‌یابند ) |
|---|---|
| RNTO | نام جدید یک فایل یا مسیری که باید تغییر نام بدهد |
| SITE | تقاضای مشخصات سرویسی که سرویس دهنده ارائه میکند |
| STAT | تقاضای وضعیت فعلی سرویس دهنده |
| STOR | تقاضای پذیرش و ذخیره داده های یک فایل از سرویس دهنده |
| STOU | تقاضای پذیرش و ذخیره داده های یک فایل از سرویس دهنده تحت نامی متفاوت |
| STRU | تقاضای تعیین ساختار یک فایل |
| SYST | تقاضای تعیین نوع سیستم عامل سرویس دهنده |
| TYPE | تعیین نوع داده ها |
| USER | اعلام کد کاربری به سرویس دهنده |

جدول (۸-۸) فرامین داخلی پروتکل FTP

هریک از ارقام صدگان، دهگان و یکان کد بازگشتی معنای خاصی دارد. بعنوان مثال اگر رقم صدگان کد ۱ یا ۲ یا ۳ باشد فرمان ارسالی موفقیت‌آمیز بوده ولی اگر ٤ یا ٥ باشد بمعنای بروز خطا تلقی می‌شود.

در جدول (۹-۸) و (۱۰-۸) معانی متفاوت رقم صدگان و دهگان کد سه رقمی بازگشتی توسط سرویس دهنده را معرفی شده است. معانی رقم سوم (یکان) از کد بازگشتی در اینجا نیامده زیرا تعداد آنها زیاد و معانی آنها در سیستمهای متفاوت با هم فرق می‌کنند.

| مقدار رقم | معنا |
|---|---|
| ۱ | فرمان درخواستی شروع به اجرا شد. منتظر کدهای برگشتی بعدی باشید. |
| ۲ | فرمان کاملا اجرا شد. ارسال فرمان بعدی مجاز است. |
| ۳ | فرمان شناخته شد ولی فعلا به دلیل کمبود اطلاعات لازم برای اجرا معلق مانده است |
| ٤ | فرمان پذیرفته نشد ولی صدور مجدد آن بلامانع می‌باشد. |
| ٥ | فرمان پذیرفته نشد وصدور مجدد آن بی فایده خواهد بود. |

جدول (۹-۸) معانی رقم اول از کد بازگشتی در پروتکل FTP

| مقدار رقم | معنای رقم<br>( معنای این رقم با در نظر گرفتن معنای رقم صدگان کد بازگشتی تکمیل میشود) |
|---|---|
| ۰ | فرمان صادره بی معنی بوده و تشخیص داده نشده است |
| ۱ | به معنای آنکه کد بازگشتی کدی است در پاسخ به تقاضای اطلاعات |
| ۲ | به معنای آنکه کد بازگشتی کدی است در ارتباط با مدیریت نشست |
| ۳ | به معنای آنکه کد بازگشتی کدی است در ارتباط با احراز هویت کاربر |
| ٤ | بلا استفاده |
| ٥ | به معنای آنکه کد بازگشتی کدی است در ارتباط با وضعیت سرویس دهنده |

جدول (۱۰-۸) معانی رقم دوم از کد بازگشتی در پروتکل FTP

در پروتکل FTP ، انتقال فایل می‌تواند در چند حالت متفاوت انجام شود. اکثر سیستمهای عامل (مثل یونیکس) فقط دارای دو روش انتقال هستند: روش متنی[۱] و روش دودوئی[۲]
هر نوع فایل اعم از فایلهای اجرائی، صدا، تصویر و حتی فایلهای متنی را می‌توان به حالت دودوئی انتقال داد و سرعت انتقال در این روش رضایت‌بخش است. با حالت متنی فقط فایلهای متن (با کدها ASCII) قابل مبادله هستند. فایلهای متنی مجموعه‌ای از کاراکترهای ASCII بهمراه برخی از کدهای کنترلی همانند "\n" ، "\t" هستند که می‌توان با دستورات معمولی از این نوع فایلها خواند و در آن نوشت. معمولاً تمام سرویس دهنده‌ها حالت پیش‌فرض یک نشست را حالت دودوئی فرض می‌کنند. کاربر باید قبل از انتقال فایل اطمینان حاصل کند که از یک مود انتقال صحیح استفاده می‌نماید چرا که مثلاً انتقال یک فایل تصویر در مود متنی نتیجهٔ مخرب خواهد داشت.

## ۵) فرامین کاربری برنامه FTP

بغیراز فرامین داخلی پروتکل FTP که روی کانال فرمان مبادله می‌شوند یکسری فرمان کاربری وجود دارد که کاربر برای استفاده از ftp باید از آنها اطلاع داشته باشد. برنامه‌های ftp در سمت کاربر معمولاً بر دو نوع هستند:
الف) برنامه‌هائی که در خط فرمان اجرا می‌شوند و کاربر مجبوراست همانند محیط DOS یا یونیکس فرامین استفاده از برنامه ftp را از حفظ باشد.

---

[۱] Text Mode
[۲] Binary Mode

ب) برنامه‌هائی که دارای ظاهر گرافیکی هستند (برنامه های GUI) و کاربر از طریق آیکون یا منوها فرمان مورد نظر خود را اجرا می‌نماید.

برای بکارگیری برنامه ftp در خط فرمان لازم است ضمن اجرای برنامه، نام ماشینی که سرویس ftp می‌دهد، بعنوان پارامتر ورودی برنامه مشخص شود:

**ftp** rtfm.mit.edu

یا مثلاً **ftp** tpci_hpws4

نامی که در جلوی اسم برنامهٔ ftp نوشته می‌شود باید قابل تحلیل[۱] و ترجمه به یک آدرس IP باشد. اگر آدرس IP ماشین سرویس دهنده را می‌دانید می‌توانید مستقیماً آدرس IP را در جلوی نام برنامهٔ ftp وارد نمائید.

مثال: **ftp** 205.150.89.5

وقتیکه ارتباط بین برنامهٔ کاربر و سرویس‌دهندهٔ ftp برقرار شد باید اعتبار کاربر برای سرویس دهنده مشخص شود تا امکانات لازم را در اختیار او قرار بدهد. در حقیقت وقتی بعنوان یک کاربر معتبر شناخته شدید به شما اجازهٔ ورود به سیستم ftp داده خواهد شد؛ به این عمل اصطلاحاً Login گفته می‌شود. پروسه‌ای که اعتبار کاربران را از لحاظ مجوز ورود و سطح دسترسی، کنترل می‌نماید "پروسه ورود"[۲] نامیده می‌شود. برای اجازه ورود به سیستم کاربر به یک کلمهٔ شناسائی و رمز عبور احتیاج دارد که این دو را مدیر شبکه در اختیار او می‌گذارد. البته بسیاری از سرویس دهنده‌های ftp در دنیا موجودند که اجازه دسترسی آزادانه به فایلها را (فقط برای خواندن و دریافت فایلها) را به کاربران می‌دهند. به اینگونه سرویس دهنده‌ها "ftp بی‌نام"[۳] گفته می‌شود و شما می‌توانید بجای کلمه شناسائی کلمهٔ anonymous یا کلمه guest را تایپ کنید.

در قطعه کد زیر مثالی از مراحل یک نشست ftp برای برقراری ارتباط با سرویس دهنده‌ای بنام tpci_hpws4 ارائه شده است: ( خطوط پر رنگ از طرف سرویس دهنده صادر شده و در ابتدای خط شماره کد بازگشتی مشخص می‌باشد)

```
ftp tpci_hpws4
Connected to tpci_hpws4.
220 tpci_hpws4 FTP server
Name (tpci_hpws4:tparker):
```

---

[۱] Resolvable
[۲] Login Process
[۳] Anonymous FTP

**331 Password required for tparker.**

Password: *******

**230 User tparker logged in.**

Remote system type is UNIX.

Using binary mode to transfer files.

برخلاف برنامهٔ Telnet بعد از برقراری ارتباط با ماشین سرویس‌دهندهٔ FTP شما واقعاً روی ماشین راه دور نیستند و عملا با پردازنده و سیستم عامل خود کار می‌کنید. از دیدگاه کاربر، فرآیندی که برنامهٔ ftp برای برقراری یک نشست دنبال می‌کند به شرح ذیل است:

الف) عمل Login : شامل تعیین کلمهٔ شناسائی و رمز عبور و تائید آنها توسط سرویس دهنده

ب) Define Directory : تعیین شاخه‌ای که پس از برقراری ارتباط بصورت پیش فرض برای شروع عملیات کاربر در نظر گرفته می‌شود.

ج) Define file Transfer Mode : تعیین روش انتقال (حالت متنی یا دودوئی)

د) Start data transfer : امکان صدور فرمان را برای کاربر فراهم می‌آورد.

ه) Stop data transfer : ارتباط را خاتمه می‌دهد.

این مراحل چندگانه به ترتیب برای هر نشست انجام می‌شوند.

برای بهره‌گیری کاربر از خدمات سیستم فایل در پروتکل ftp فرامینی تعریف شده‌اند که در جدول (۱۱–۸) معرفی شده‌اند.

| فرامین کاربری | معنای فرمان |
|---|---|
| Ascii | تنظیم حالت انتقال فایل به حالت متنی |
| Binary | تنظیم حالت انتقال فایل به حالت دودویی |
| Cd | تغییر شاخه جاری به شاخه جدید بر روی سرویس دهنده |
| Close | ختم نشست |
| Del | حذف یک فایل از روی سرویس دهنده |
| Dir | فهرست گیری از شاخه جاری سرویس دهنده |
| Get | تقاضای انتقال یک فایل از سرویس دهنده |
| Hash | هرگاه یک بلوک داده از یک فایل در حال انتقال سالم رسید علامت ویژه ای را نشان بدهد |

| | |
|---|---|
| Help | راهنمایی |
| Lcd | تقاضای تغییر شاخه جاری بر روی ماشین محلی کاربر |
| Mget | دریافت چندین فایل از روی سرویس دهنده |
| Mput | ارسال چندین فایل بر روی سرویس دهنده |
| Open | تقاضای برقراری یک نشست و وصل به یک سرویس دهنده |
| Put | ارسال یک فایل بر روی سرویس دهنده |
| Pwd | نمایش شاخه جاری از سرویس دهنده |
| Qoute | ارسال مستقیم یکی از فرامین داخلی |
| Quit | تقاضای ختم نشست |

جدول (۱۱-۸) فرامین کاربری پروتکل FTP

یک مثال از نشست ftp در زیر آمده است: به کدهائی که در پاسخ به هر فرمان از طرف سرویس‌دهنده صادر شده دقت کرده با جدول (۹–۸) و (۱۰–۸) مقایسه نمایند.

tpci_hpws1-1> ftp tpci_hpws4

Connected to tpci_hpws4.

**220 tpci_hpws4 FTP server (Version 1.7.109.2 Tue Jul 28 23:32:34 GMT 1992) ready.**

Name (tpci_hpws4:tparker):

**331 Password required for tparker.**

Password: *******

**230 User tparker logged in.**

Remote system type is UNIX.

Using binary mode to transfer files.

ftp> pwd

**257 "/u1/tparker" is current directory.**

ftp> get mandelfile1.gif

remote: mandelfile1.gif local: mandelfi.gif

**200 PORT command successful**

**150 Opening BINARY mode data connection for mandelfile1.gif**

**226 File transfer complete**

1192834 bytes sent in 0.89 seconds

ftp> <Ctrl+d>

tpci_hpws1-2>

در مثال بالا فایلی بنام mandelfile1.gif از یک ماشین سرویس‌دهنده با سیستم عامل یونیکس به ماشین میزبان با سیستم عامل DOS منتقل شده است. دقت کنید که چون در سیستم عامل DOS نام فایل حداکثر هشت کاراکتر است به همین دلیل اسم فایل بصورت خودکار کوتاه شده است. در اینجا از حالت دودوئی برای انتقال فایل استفاده شده که پیش فرض سیستم بوده و احتیاجی به تنظیم ندارد.

با اضافه کردن گزینه d- در هنگام فراخوانی و اجرای برنامه می‌توان عمل اشکال زدائی انجام داد. با این گزینه تمام فرامین داخلی که بین برنامه FTP و سرویس دهنده مبادله میشوند قابل مشاهده خواهند بود.در مثال بالا پیغامهایی که از طرف سرویس دهنده ارسال شده همگی با یک کد سه رقمی ظاهر شده‌اند و بقیه پیغامها از طرف برنامهٔ ftp می‌باشد.

در مثال سادهٔ زیر عملیات نشست ftp با امکان اشکال‌زدایی آورده شده است:

tpci_hpws1-1> ftp -d

ftp> open tpci_hpws4

Connected to tpci_hpws4.

**220 tpci_hpws4 FTP server Name (tpci_hpws4:tparker):**

---> USER tparker

**331 Password required for tparker.**

Password:

---> PASS qwerty5

**230 User tparker logged in.**

---> SYST

**215 UNIX Type: L8**

Remote system type is UNIX.

---> Type I

**200 Type set to I.**

Using binary mode to transfer files.

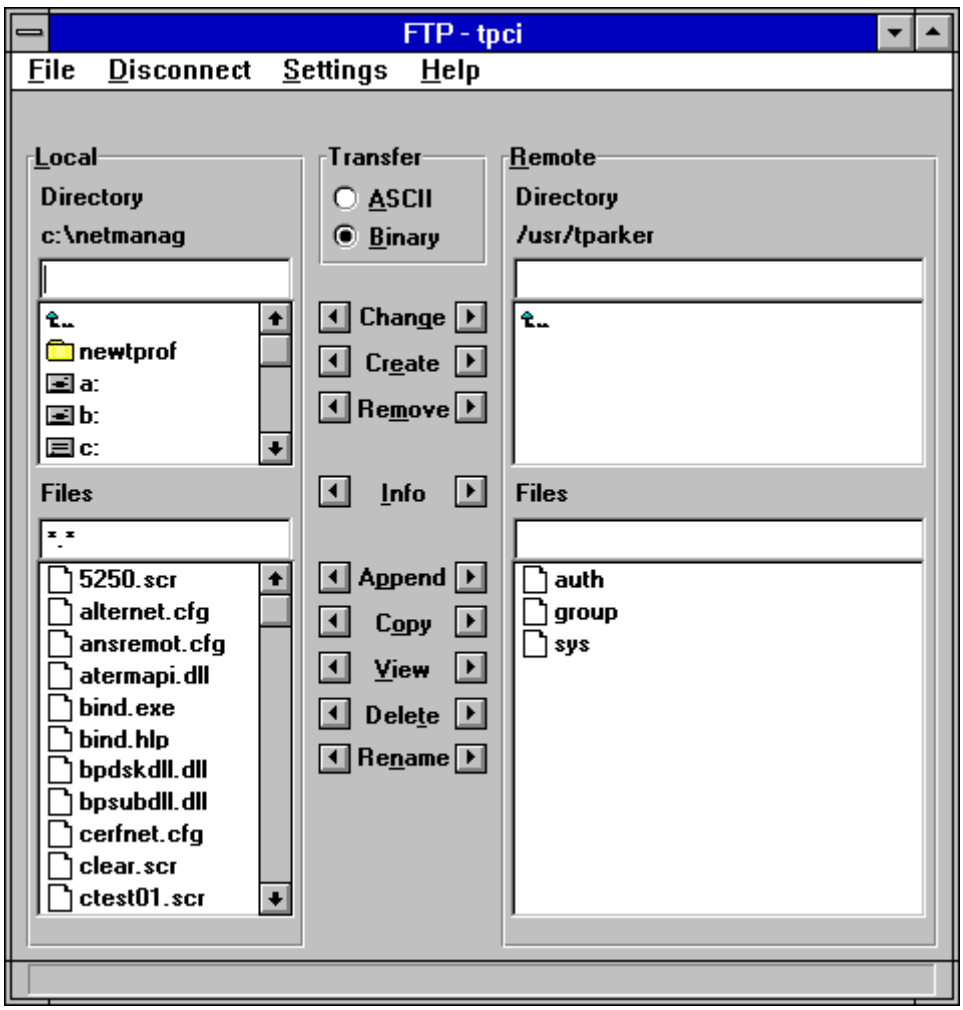
```
ftp> ls
---> PORT 47,80,10,28,4,175
200 PORT command successful.
---> TYPE A
200 Type set to A.
---> LIST
150 Opening ASCII mode data connection for /bin/ls.
total 4
-rw-r-----  1 tparker  tpci    2803  Apr 29 10:46 file1
-rw-rw-r--  1 tparker  tpci    1286  Apr 14 10:46 file5_draft
-rwxr-----  2 tparker  tpci   15635  Mar 14 23:23 test_comp_1
-rw-r-----  1 tparker  tpci      52  Apr 22 12:19 xyzzy
Transfer complete.
---> TYPE I
200 Type set to I.
ftp> <Ctrl+d>
tpci_hpws1-2>
```

دقت کنید که در مثال بالا هنگام دریافت لیست شاخه‌ها[1]، چگونه حالت باینری به متنی تبدیل شده و سپس دوباره به مود دودویی (مقدار پیش‌فرض سیستم) برمی‌گردد.

وقتیکه از یک سیستم عامل گرافیکی مثل MS-Windows استفاده می‌شود میتوان از یک ابزار مبتنی بر GUI[2] استفاده کرد. برای مثال نرم افزار NetManage's ChameleonNFS یا CuteFTP برنامه های کمکی FTP هستند. در شکل (۱۲–۸) نرم افزار FS client تحت Windows با یک سرویس دهنده FTP تحت یونیکس نشست برقرار کرده است. طرف Local ( سمت چپ ) پنجرهٔ مربوط به ماشین Windows را نشان میدهد و طرف Remote ( سمت راست ) پنجرهٔ مربوط به محتویات سیستم فایل جاری یونیکس را نشان می‌دهد. وقتی که از یک برنامه کمکی

[1] Directory listing
[2] Graphic User Interface

GUI مثل این برنامه استفاده می‌شود، می‌توان از ماوس و کلیدهای گوناگون برای انتقال فایل بین دو ماشین ، استفاده کرد.

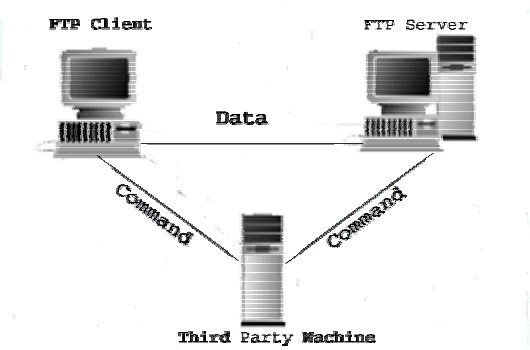



شکل (۱۲–۸) مثالی از یک نرم افزار مشتری FTP با ظاهر گرافیکی

## ۶) انتقال با واسطه در پروتکل FTP[۱]

FTP این امکان را فراهم می‌سازد که انتقال فایلها از طریق ماشین سومی که بین برنامه مشتری و سرویس دهنده قرار گرفته است، انجام شود. این رویه به عنوان "انتقال باواسطه" شناخته شده است و اجازه دسترسی به سیستم فایلِ ماشین سرویس دهنده و نظارت بر فرامین صادره از طریق این ماشین انجام می‌شود. استفاده از یک سیستم واسطه برای نظارت بر فرامین صادره از طرف کاربران و بررسی مجوزها و سطوح دسترسی هر کاربر باعث میشود که حجم پردازش روی سرویس دهنده FTP کاهش یافته ، سرعت انتقال افزایش داشته باشد. شکل (۱۳-۸) نمایش کلی یک انتقال باواسطه را همراه با کانالهای ایجاد شده در ماشین واسط، نشان می‌دهد.

در این روش هرگاه یک نشست برگزار شود، کانال فرمان حتما از طریق ماشین واسطه بین ماشین مشتری و ماشین سرویس دهنده برقرار میشود در حالیکه کانال داده مستقیماً بین دو ماشین برقرار می‌شود. وقتی یک فرمان صادر شود ، چون از کانال ماشین واسطه عبور میکند ، مجوزها بررسی می‌شود و در صورت مجاز بودن فرمان ، درخواست به سوی سرویس دهنده ارسال میشود. انتقال داده ها مستقیماً صورت می‌گیرد، چرا که مجوزها قبلا بررسی شده اند.




شکل (۱۳-۸) انتقال با واسطه در پروتکل FTP

[۱] Third Party Transfer

### ۱-۶) دسترسی بی‌نام به FTP

FTP برای توانا ساختن قابلیتهای انتقال فایل نیاز به کد کاربری و کلمه عبور کاربر دارد. روش بسیار راحت‌تری برای ایجاد دسترسی عمومی به یک مجموعه فایل یا دایرکتوری وجود دارد که FTP بی‌نام خوانده می‌شود. با FTP بی‌نام، دیگر نیازی به داشتن مجوز روی یک ماشین نیست. FTP این کار را معمولاً با فعال کردن حالت "ورود بی‌نام" توسط یک کلمه عبور به نام "مهمان یا guest" یا کلمه عبور پوچ (یک رشته خالی ) انجام می‌دهد. نشست زیر استفاده از یک سرویس دهنده FTP بی‌نام را نشان می‌دهد:

tpci_hpws4-1> ftp uofo.edu

Connected to uofo.edu.

**220 uofo.edu FTP server (Version 1.7.109.2 Tue Jul 28 23:32:34 GMT 1992) ready.**

Name (uofo:username): anonymous

**331 Guest login ok, send userID as password**.

Password: tparker

**230 Guest login ok, access restrictions apply**.

ftp> <Ctrl+d>

tpci_hpws4-2>

اگر سرویس دهنده FTP روی تواناسازی ورود بی‌نام تنظیم شده باشد، از شما یک کلمه عبور می‌خواهد و چون آنرا وارد نمیکنید بعد از یک اخطار در مورد محدودیتهای دسترسی به شما مجوز ورود داده می‌شود. اگر شما به فایلی روی سرویس دهنده نیاز داشته باشید می‌توانید آنرا انتقال بدهید. با عمومیت یافتن اینترنت، سایتهای FTP بی‌نام رشد فزاینده ای پیدا کرده اند و اهدافی نظیر نشر رایگان دانش یا تبلیغات دارند.

## ۷) سرویس دهنده های FTP

بسیاری از ماشینهای یونیکس به صورت پیش‌فرض به عنوان سرویس دهنده FTP عمل می‌کنند. برای ارائه امکانات سرویس دهنده FTP ، باید هنگام بوت شدن سیستم عامل ، دایمون[۱] ftpd اجرا شده باشد. این دایمون در سیستم عامل یونیکس معمولاً بوسیلهٔ پروسهٔ inted مدیریت می‌شود. هنگامیکه سیستم با استفاده از پروسه inetd بوت می‌شود ، به پورت

[۱] Daemon

فرمان TCP (کانال ۲۱) گوش می‌دهد تا درخواستهای رسیده برای برقراری ارتباط را ببیند، سپس دایمون ftpd را برای سرویس دادن به درخواست ، احیا می‌کند. اگر می‌خواهید مطمئن شوید که سیستم یونیکس یا لینوکس شما می‌تواند درخواستهای ftp را مدیریت کند باید مطمئن شوید که آیا برنامه ftpd ، وقتی که inetd آنرا فراخوانی می‌کند ، شروع می‌شود یا نه. در ضمن این کار را میتوان با بررسی فایل "تنظیمات پیکربندی"[1] inetd انجام داد. باید دید که آیا خطی مثل خط زیر در فایل تنظیمات inetd وجود دارد. اگر این خط وجود نداشته باشد باید آن را اضافه کنید:

ftp stream tcp nowait root /usr/etc/ftpd ftpd -l

در بسیاری از سیستم‌های یونیکس این خط در فایل تنظیمات پیکربندی inetd وجود دارد، گرچه ممکن است که به صورت توضیح[2] علامت خورده باشد که در این صورت باید علامتهای توضیح را بردارید. تنظیمات پیکربندی inetd در یونیکس یا لینوکس معمولا در فایل زیر ذخیره می‌شود:

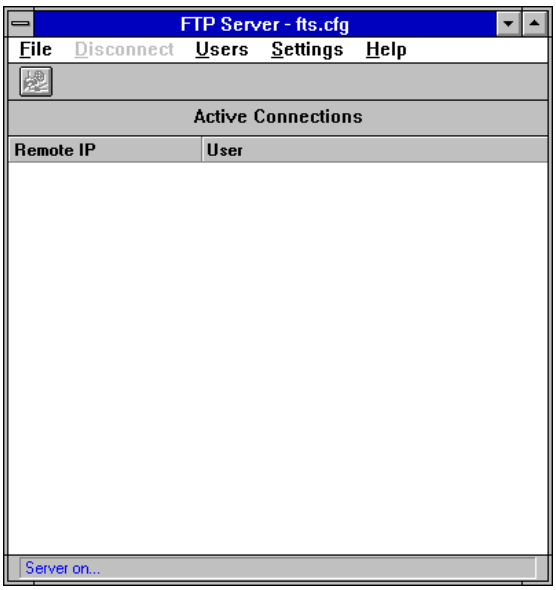
/etc/inetd.config یا /etc/inetd.conf

MS-Windows 95/98 فاقد یک برنامهٔ سرویس دهنده FTP به عنوان بخشی از نرم‌افزار توزیع شده خودشان هستند ، لذا در هنگام نیاز باید یک بستهٔ تجاری به آنها اضافه کرد. شکل (۸–۱٤) برنامهٔ Net Manage's ChameleonNFS را نشان می‌دهد که یک برنامهٔ سرویس دهنده FTP می‌باشد و می‌توان از آن، به عنوان ابزاری برای ارائه خدمات فایل در ماشینهای مبتنی بر سیستم عامل ویندوز 3.x استفاده کرد. برای فعال کردن نرم‌افزار Net Manage FTP Server باید ابتدا گزینه FTP Server Config در گروه برنامهٔ NetManage اجرا شود. در این صورت پنجره ای که در شکل (۸–۱۵) نشان داده شده، ظاهر می‌شود. پروسهٔ FTP Server اکنون فعال است و هرکس دیگری روی یک ماشین در شبکه ، اگر دارای مجوز دسترسی باشدمی‌تواند به ماشین شما متصل شود.

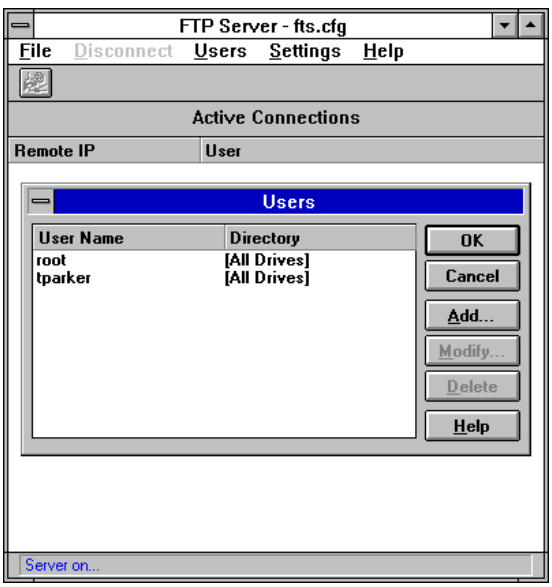
- دسترسی به سرویس FTP توسط تعریف لیست کاربران که در بستهٔ FTP Server قرار دارد کنترل می‌گردد و میتوان با انتخاب منوی User از Net Manage FTP Server کاربران را تعریف کرد. با این کار می‌توانید نام کاربرها را به سیستم خود اضافه کنید. اگر کاربری روی یک

[1] inetd Configuration File

[2] Comment




شکل (۱٤–۸) یک برنامهٔ سرویس دهنده FTP در محیط MS-Windows




شکل (۱۵–۸) یک برنامهٔ سرویس دهنده FTP در محیط MS-Windows

ماشین سعی کند که به نرم‌افزار FTP Server شما متصل شود، کد شناسائی و کلمه عبور آنها با نام و کلمهٔ عبوری که شما در این پنجره وارد کرده‌اید، تطبیق داده می‌شود. این به شما امکان می‌دهد که لیستی از کاربرهایی را که می‌توانند با سیستم شما مبادلهٔ فایل داشته باشند، تنظیم کنید. البته دسترسی به سیستم فایل تا زمانی امکان پذیر است که FTP Server در حال اجرا باشد.

اگر یک برنامهٔ FTP Server را اجرا می‌کنید بهترین کار آنست که برای هر کاربر یک شاخهٔ مجزا ایجاد کنید تا کاربر بتواند عملیات حذف ، اضافه یا تغییر فایلهایش را در شاخهٔ خودش انجام بدهد. از طرفی یک شاخهٔ عمومی[۱] ایجاد کنید بگونه ای که در اختیار همهٔ کاربران باشد وتمام فایلهایی که می‌خواهند وارد سیستم محلی شوند و یا از آن خارج گردند در آن قرار بگیرند. این کار موجب جلوگیری از دسترسی هر کاربر به فایلهای سایر کاربران و بالابردن ضریب امنیت سیستم خواهد شد. علاوه بر این امکان بررسی فایلهای وارده به شاخه عمومی از لحاظ آلوده بودن به ویروس وجود خواهد داشت.

اگر بخواهید یک سرویس بی‌نام یا مهمان برای کاربران روی شبکه تعریف کنید ، باید یک کاربر بدون کلمهٔ عبور یا یک کلمهٔ عبور ساده مثل مهمان (guest) تعریف شود. باید ناحیه‌ای که یک کاربر بی نام یا مهمان می‌تواند از آن استفاده کند به شدت محدود بوده و فقط اجازه دریافت فایل داشته باشد.

## ۸) پروتکل سادهٔ انتقال فایل : TFTP[۲]

پروتکل جزئی و ساده انتقال فایل که از این به بعد آنرا TFTP می‌نامیم یکی از ساده‌ترین پروتکلهای انتقال فایل است که امروزه از آن استفاده می‌شود. این پروتکل در دو زمینهٔ اصلی با FTP متفاوت است :

- این پروتکل نیاز به برقراری یک نشست و عملیات ورود[۳] به سیستم ندارد و بالطبع بدون برقراری یک نشست و انجام عملیات بررسی صلاحیت کاربر مشکلاتی نظیر دسترسی کاربران غیر مجاز محتمل خواهد بود.

TFTP از پروتکل UDP که یک پروتکل انتقال "بدون اتصال" است به جای TCP استفاده می‌کند و چون پروتکل UDP نظارتی بر ترتیب داده ها اعمال نمی‌کند بنابراین TFTP مجبور

---

[۱] Pubic
[۲] Trivial File Transfer Protocol
[۳] Login

است برای تضمین صحت و ترتیب داده ها الگوریتم‌هایی را به کـار بگـیرد.( TFTP شـمارهٔ پورت ٦٩ را به کار می‌برد.)

معمولاً برای انتقال فایل بین دو ماشین، در جایی که بشود از FTP استفاده کرد TFTP را بـه کار نمی‌برند چرا که در این پروتکل عملیاتی نظیر فهرست گیری از فایلها و شاخه هـا ، تغییـر شاخه جاری و احراز هویت کاربر امکان پذیر نیست ولی TFTP با تمام مشـکلاتش مزایـائی نسبت به FTP دارد. مثلا هنگام کار با ماشینهای بدون دیسک[۱] یا ایسـتگاههای کـاری[۲] TFTP کارآمدتر است. نوعاً TFTP برای بار کردن برنامه‌های کاربردی کوچک یا فونت روی ماشـینها به کار می‌رود. مهمترین کاربرد این پروتکل برای بوت کردن سیستمهائی است که بدون دیسک بوده و مجبورند از طریق ROM بوت شوند. در اینگونه موارد TFTP حتماً لازم است چـرا کـه ماشینهای بدون دیسک ، تا وقتی که سیستم عامل کاملا بار نشده باشـد، قـادر بـه اجـرای FTP نیستند. اندازه کوچک برنامه اجرایی TFTP و نیاز کم آن به حافظه باعث شده کـه بتـوان آن را در BOOTROM جا داد.

TFTP اجازه دسترسی به یک فایل را با در نظر گرفتن مشخصه[۳] آن فـایل اعمـال میکنـد. بـه عنوان مثال روی سیستم‌های یونیکس ، فـایلی کـه مشـخصه آن " خواندنـی/ نوشـتنی" اسـت می‌تواند توسط تمام کاربرها قابل دسترسی باشد. (اجازه خوانـدن و نوشـتن، هـر دو را داشـته باشد.) به خاطر این مقررات سهلگیرانه، بسیاری از مسـئولین شـبکه ، کنـترل بیشـتری را روی TFTP اعمال می‌کنند (یا کلاً استفاده از آن را منع می‌کنند.). در غـیر ایـن صـورت بـرای یـک کاربر آگاه بسیار ساده است که اقدام به دریافت فایلی کند که موجب خدشه دار شـدن امنیـت شود.

انتقال توسط TFTP به دلایل بسیاری می‌تواند با شکست مواجه شود و هر نوع خطـایی کـه هنگام عمل انتقال رخ بدهد، باعث شکست کامل انتقـال می‌شـود. TFTP برخـی از پیغام‌هـای اساسی خطا را پشتیبانی می‌کند ولی نمی‌تواند خطاهای ساده مثل کمبود منابعی نظیر حافظه یـا فضای ناکافی دیسک برای انتقال یک فـایل را رفـع و مدیریـت کنـد. در هنگـام بـروز چنیـن خطاهائی تمام مراحل انتقال باید از نو آغاز شود.

---

[۱] Diskless
[۲] Workstations
[۳] File Attribute

### ۸-۱) بسته‌های TFTP

بگونه ای که در بخش قبلی اشاره شد پروتکل TFTP از سوکتهای نوع دیتاگرام ( مبتنی بر UDP ) استفاده میکند و در ضمن کانالی مجزا برای ارسال فرمان وجود ندارد و چون فرامین و داده ها بطور همزمان ارسال می‌شوند بنابراین باید ساختاری برای بلوکهای داده تعریف شود تا طرفین ارتباط بتوانند داده ها را از فرامین و پیغامهای کنترلی تشخیص بدهند.

در پروتکل TFTP داده ها اعم از بلوکهای فایل ، فرامین یا پیغامهای کنترلی در قالب بلوکهائی از داده که ساختمان مشخصی دارند مبادله میشوند. هر بلوک داده که از ساختار تعریف شده تبعیت نکند بعنوان یک خطا تلقی شده و دور ریخته می‌شود.

پنج ساختار برای بلوکهای داده تعریف شده که در ادبیات این پروتکل به هر یک از آنها "بسته TFTP" گفته میشود. ساختار هر یک از این بسته ها به صورت زیر است:

**بسته RRQ : تقاضای دریافت یک فایل**

| Opcode (2 Byte) | File Name (String) | 0 | Mode (String) | 0 |
|---|---|---|---|---|

**بسته WRQ : تقاضای ارسال یک فایل**

| Opcode (2 Byte) | File Name (String) | 0 | Mode (String) | 0 |
|---|---|---|---|---|

**بسته Data : ارسال داده های یک فایل**

| Opcode (2 Byte) | Block Number (2 Byte) | Data (0~512 Byte) |
|---|---|---|

**بسته Ack : پیغام تصدیق و پذیرش**

| Opcode (2 Byte) | Block Number (2 Byte) |
|---|---|

**بسته Error : پیغام خطا**

| Opcode (2 Byte) | Block Number (2 Byte) | Error Message (String) | 0 |
|---|---|---|---|

همانگونه که از ساختار بسته ها مشخص است در تمام آنها دو بایت اول که Opcode نـام دارد نوع بسته را مشخص می‌کند. در این فیلد دو بایتی فقط یکی از مقادیر ۰ تا ۵ قرار می‌گیرد کـه معانی هر یک از آنها در جدول (۱۶-۸) مشخص شده است. حال باید ساختار برنامه سـرویس دهنده و مشتری و طریقه مبادله داده ها را بررسی نماییم:

| توضیح | Opcode | نوع بسته |
|---|---|---|
| **بسته Ack : پیغام تصدیق و پذیرش** | ٤ | Ack |
| **بسته Data : ارسال داده های یک فایل** | ۳ | Data |
| **بسته Error : پیغام خطا** | ۵ | Error |
| **بسته RRQ : تقاضای دریافت یک فایل** | ۱ | RRQ |
| **بسته WRQ : تقاضای ارسال یک فایل** | ۲ | WRQ |

جدول (۱۶-۸) انواع بسته های TFTP

الف) در برنامه سمت سرویس دهنده یک سوکت دیتاگرام باز شده و به آن شـماره پـورت ٦۹ نسبت داده می‌شود. (توسط تابع (bind) سپس برنامه به حالت دریافت ( با تـابع (recvfrom ) وارد شده و منتظر می‌ماند.

ب) در برنامه سمت مشتری سوکتی از نوع دیتاگرام باز میشود و سـپس یـک شـماره پـورت تصادفی به آن نسبت داده می‌شود. برنامه سمت مشتری در پروتکل TFTP از مواردی است کـه مجبور است به سوکت ایجاد شده آدرس پورت مقید کند چرا که در خـلال انتقـال یـک فـایل نباید شماره پورت عوض شود. پس از این کار تقاضای خود را در قـالب یکـی از بسـته هـای RRQ یا WRQ به سرویس دهنده ارسال مینماید.

ج) در صورتی که سرویس دهنده تقاضای رسیده را بپذیرد سـوکتی جدیـد بـاز کـرده و یـک شماره پورت تصادفی به آن نسبت می‌دهد. سپس از طریق سوکت جدیـد یـک بسـته Ack بـا مشخصات( Block No=0 و Opcode=4) به برنامه مشتری بر خواهد گرداند. سوکت جدیـد تـا پایان عملیات انتقال فایل باقی مانده و پس از آن بسته خواهد شد.

د) برنامه مشتری ، پس از دریافت بسته Ack ، اقدام به ارسال یا دریـافت بسـته هـای داده کـه دقیقا ۵۱۲ بایتی هستند می‌نماید. اگر بسته داده ای ارسال ( یا دریافت) شود که اندازه کمـتر از ۵۱۲ بایت داشته باشد به عنوان آخرین بلوک فایل تلقی شده و خاتمه ارتباط را اعـلام خواهـد کرد.

در ساختار بسته های RRQ یا WRQ باید فیلدهای زیر مقدار دهی شوند:

- فیلد Opcode : مقدار ۱ برای بسته RRQ ( تقاضای دریافت فایل )و مقدار ۲ برای بسته WRQ ( تقاضای ارسال فایل )
- فیلد File Name : نام فایلی که باید دریافت یا ارسال شود.
- فیلد Mode : در این فیلد یکی از سه رشته زیر می‌تواند قرار بگیرد.
  - 'NetASCII' : یعنی فایل متنی است و از کدهای ASCII استفاده کرده است.
  - 'Byte' : یعنی فایل بصورت دودوئی و در قالب رشته ای از بایتهای تفسیر نشده ارسال یا دریافت می‌شود.
  - 'Mail' : مشخص می‌کند که مقصد یک کاربر است نه یک فایل . در چنین حالتی به جای نام فایل آدرس پست الکترونیکی یک شخص ( به فرم user@xyz.com ) یا مشخصه کاربری او قرار می‌گیرد و قالب اطلاعات قطعاً متنی خواهد بود.

فرآیند ارتباط در TFTP توسط برنامه مشتری با فرستادن یک درخواست RRQ یا WRQ آغاز می‌شود. به عنوان بخشی از درخواست ، نام فایل و حالت انتقال مشخص می‌شود و شماره اولین بلوک داده ۱ خواهد بود. وقتی یکی از طرفین یک بلوک ۵۱۲ بایتی داده ارسال کرد یک زمان سنج را تنظیم کرده و منتظر بسته Ack می‌ماند. اگر در مهلت مقرر ( به طور معمول ۳ ثانیه ) بسته Ack نرسید بلوک داده مجددا ارسال می‌شود. اگر عمل تکرار یک بلوک چندین بار متوالی انجام شود ولی پاسخ Ack بر نگردد ( مثلا ۸ بار ) خاتمه ارتباط تلقی و سوکت ایجاد شده بسته خواهد شد و تمام مراحل باید از نو انجام شود.

هر بلوک داده یک شماره ترتیب دارد که از ۱ شروع شده و به ازای انتقال موفقیت آمیز آن یکی اضافه خواهد شد. طرفین هیچگاه بسته ای را خارج از ترتیب قبول نخواهند کرد و با این موضوع امکان هر گونه خطایی در مورد ترتیب بسته ها منتفی خواهد بود. به دلیل اینکه فرستنده یک بسته داده ، قبل از فرستادن بسته بعدی منتظر پاسخ Ack می‌ماند به پروتکل TFTP پروتکل فلیپ فلاپ گفته می‌شود.

فرآیند انتقال وقتی پایان می‌یابد که یک بستهٔ داده با طول کمتر از ۵۱۲ بایت دریافت شود یا خطائی اتفاق بیفتد. فقط یک مورد خطا وجود دارد که قابل جبران است و آن خطا در شماره پورت است (خطای شماره ۵ ) بدین معنا که برنامه مشتری از شماره پورتی که در ابتدا روی آن توافق شده است استفاده نکرده و شماره پورت دیگری را به کار برده است. برای روشنتر شدن چگونگی توافق روی شمارهٔ پورت ارائه یک مثال ، مناسب خواهد بود:

برنامه A به عنوان مشتری شماره پورت تصادفی ۳۵٤۰ را انتخاب و یکی از بسته های RRQ یا WRQ را برای سرویس دهنده به شماره پورت ٦۹ ارسال میکند. سرویس دهنده سوکتی ایجاد و با انتخاب یک شماره پورت تصادفی مثل ۱٤۵۲۳ برای برنامهٔ مشتری بسته Ack را با این شماره پورت ارسال میکند. برنامه A تا پایان عملیات انتقال ، پورت مقصد را ۱٤۵۲۳ تلقی میکند. پس بدین نحو روی شماره های ۳۵٤۰ از مشتری و ۱٤۵۲۳ توافق شده است. هرگاه برنامه مشتری برای ارسال از شماره پورت دیگری استفاده کند خطای شماره ۵ اتفاق می‌افتد که قابل جبران است یعنی میتواند عملیات را با اصلاح شماره پورت ادامه بدهد. در جدول (۸–۱۷) انواع خطاهائی که توسط سرویس دهنده و با ارسال بسته Error گزارش خواهد شد تعریف شده اند.

| شماره خطا | توضیح |
|---|---|
| ۰ | **نوع خطا تعریف نشده است. می‌توان به پیغام خطا در فیلد Error Message رجوع کرد.** |
| ۱ | **فایل درخواستی وجود ندارد** |
| ۲ | **اجازه دسترسی به فایل وجود ندارد** |
| ۳ | **ظرفیت دیسک پر شده یا آنکه در تخصیص فضا مشکل وجود دارد** |
| ٤ | **بسته ارسالی تعریف نشده و نا معتبر است** |
| ۵ | **شماره پورت استفاده شده توسط برنامه مشتری تعریف نشده است** |
| ٦ | **فایل از قبل وجود دارد و امکان نوشتن مجدد آن نیست.** |
| ۷ | **کاربر نام برده شده وجود خارجی ندارد** |

جدول (۸–۱۷) شماره و نوع خطاهای گزارش شده توسط سرویس دهنده TFTP

### ۸–۲) دستورات TFTP

مهمترین دستورات کاربری TFTP در جدول (۸–۱۸) نشان داده شده است. مجموعهٔ فرامین TFTP شبیه FTP است ولی از چند جنبهٔ مهم به لحاظ طبیعت بدون اتصال این پروتکل متفاوت است. قابل توجه‌ترین تفاوت در فرمان connect (اتصال) است که به سادگی به جای اینکه یک نشست را برقرار کند، آدرس ماشین سرویس دهنده را معین می‌کند.

در سیستم عامل یونیکس نام برنامه سرویس دهندهٔ TFTP ، دایمون tftpd میباشد.

در زیر مثالی از برقراری ارتباط با یک سرویس دهنده TFTP ارائه شده است :

```
tpci_hpws1-1> tftp
tftp> connect tpci_hpws4
tftp> trace
Packet tracing on.
tftp> binary
Binary mode on.
tftp> verbose
Verbose mode on.
tftp> status
Connected to tpci_hpws4.
Mode: octet Verbose: on Tracing: on
Rexmt-interval: 5 seconds, Max-timeout: 25 seconds
tftp> get /usr/rmaclean/docs/draft1
getting from tpci_hpws4:/usr/rmaclean/docs/draft1 to /tmp/draft1 [octet]
sent RRQ <file=/usr/rmaclean/docs/draft1, mode=octet>
received DATA <block1, 512 bytes>
send ACK <block=1>
received DATA <block2, 512 bytes>
send ACK <block=2>
received DATA <block3, 128 bytes>
send ACK <block=3>
Received 1152 bytes in 0.2 second 46080 bits/s]
tftp> quit
tpci_hpws1-2>
```

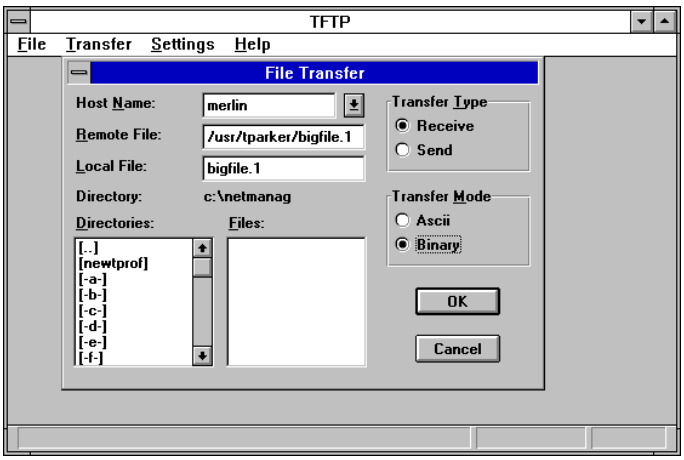
در فرآیند بالا می‌توانید ببینید که دستورات trace و verbose باعث میشود که در حین انتقال فایل تمامِ فعل و انفعالات بین سرویس دهنده و سرویس گیرنده و مبادله کدها بین دو ماشین روی خروجی نشان داده شود.

بگونه ای که در مثال مشخص شده است بعد از صدور دستور get برای تقاضای یک فایل هر بار که یک بلوک داده از فایل دریافت می‌شود در پاسخ به آن یک پیغام Ack ارسال میگردد تا سرویس دهنده ارسال بلوکها را ادامه بدهد.

سرویس دهنده TFTP روی تمام سیستم‌های یونیکس در دسترس میباشد ولی معمولا با احتیاط و اکراه نصب میشود چرا که این سرویس دهنده میتواند امنیت سیستم را به مخاطره بیندازد . شکل (۱۹-۸) برنامه کمکی TFTP از Chameleon NFS را نشان می‌دهد، که به شما اجازه می‌دهد نام یک یک ماشین راه دور ، نام فایل مورد نظر شما و نام فایلی که ارسال یا دریافت میشود و نوع انتقال را مشخص کرده تا انتقال فایل در "پس زمینه" و با استفاده از پروتکل UDP انجام شود.

| فرامین TFTP | توضیح فرمان |
|---|---|
| Binary | تعیین حالت انتقال فایل به صورت دودوئی |
| Connect | تعیین آدرس ماشین سرویس دهنده |
| Get | تقاضای دریافت یک فایل از سرویس دهنده |
| Put | تقاضای ذخیره فایل روی ماشین سرویس دهنده |
| Trace | فرامین پروتکل را روی خروجی نشان میدهد |
| Verbose | تمامی اطلاعات لازم را به کاربر نشان میدهد |

جدول (۱۸-۸) فرامین کاربری TFTP




شکل (۱۹-۸) نرم افزار TFTP در محیط MS_Windows

## ۹) مراجع این فصل

مجموعهٔ مراجع زیر می‌توانند برای دست آوردن جزییات دقیق و تحقیق جامع در مورد پروتکلهای معرفی شده در این فصل مفید واقع شوند.

### مراجع مفید در مبحث TelNet


| | |
|---|---|
| RFC1205 | "Telnet 5250 Interface," Chmielewski, P.; 1991 |
| RFC1198 | "FYI on the X Window System," Scheifler, R.W.; 1991 |
| RFC1184 | "Telnet Linemode Option," Borman, D.A., ed.; 1990 |
| RFC1091 | "Telnet Terminal-Type Option," VanBokkelen, J.; 1989 |
| RFC1080 | "Telnet Remote Flow Control Option," Hedrick, C.L.; 1988 |
| RFC1079 | "Telnet Terminal Speed Option," Hedrick, C.L.; 1988 |
| RFC1073 | "Telnet Window Size Option," Waitzman, D.; 1988 |
| RFC1053 | "Telnet X.3 PAD Option," Levy, S.; Jacobson, T.; 1988 |
| RFC1043 | "Telnet Data Entry Terminal Option: DODIIS Implementation," Yasuda, A.; Thompson, T.; 1988 |
| RFC1041 | "Telnet 3270 Regime Option," Rekhter, Y.; 1988 |
| RFC1013 | "X Window System Protocol, version 11: Alpha Update," Scheifler, R.W.; 1987 |
| RFC946 | "Telnet Terminal Location Number Option," Nedved, R.; 1985 |
| RFC933 | "Output Marking Telnet Option," Silverman, S.; 1985 |
| RFC885 | "Telnet End of Record Option," Postel, J.B.; 1983 |
| RFC861 | "Telnet Extended Options: List Option," Postel, J.B; Reynolds, J.K.; 1983 |

| | |
|---|---|
| RFC 860 | "Telnet Timing Mark Option," Postel, J.B.; Reynolds, J.K.; 1983 |
| RFC 859 | "Telnet Status Option," Postel, J.B.; Reynolds, J.K.; 1983 |
| RFC 858 | "Telnet Suppress Go Ahead Option," Postel, J.B,; Reynolds, J.K.; 1983 |
| RFC 857 | "Telnet Echo Option," Postel, J.B.; Reynolds, J.K.; 1983 |
| RFC 856 | "Telnet Binary Transmission," Postel, J.B.; Reynolds, J.K.; 1983 |
| RFC 855 | "Telnet Option Specifications," Postel, J.B.; Reynolds, J.K.; 1983 |
| RFC 854 | "Telnet Protocol Specification," Postel, J.B.; Reynolds, J.K.; 1983 |
| RFC 779 | "Telnet Send-Location Option," Killian, E.; 1981 |
| RFC 749 | "Telnet SUPDUP-Output Option," Greenberg, B.; 1978 |
| RFC 736 | "Telnet SUPDUP Option," Crispin, M.R; 1977 |
| RFC 732 | "Telnet Data Entry Terminal Option," Day, J.D.; 1977 |
| RFC 727 | "Telnet Logout Option," Crispin, M.R.; 1977 |
| RFC 726 | "Remote Controlled Transmission and Echoing Telnet Option," Postel, J.B.; Crocker, D.; 1977 |
| RFC 698 | "Telnet Extended ASCII Option," Mock, T.; 1975 |

## مراجع مفید در مبحث پروتکلهای انتقال فایل

| | |
|---|---|
| RFC 1094 | "NFS: Network File System Protocol Specification," Sun Microsystems, Inc.; 1989 |
| RFC 1068 | "Background File Transfer Program (BFTP)," DeSchon, A.L.; Braden, R. T.; 1988 |
| RFC 959 | "File Transfer Protocol," Postel, J.B.; Reynolds, J.K.; 1985 |
| RFC 949 | "FTP Unique-Named Store Command," Padlipsky, M.A.; 1985 |
| RFC 783 | "TFTP Protocol (Revision 2)," Sollins, K.R.; 1981 |
| RFC 775 | "Directory Oriented FTP Commands," Mankins, D.; Franklin, D.; Owen, A.D.; 1980 |

## ۱) مقدمه

پس از آشنایی مقدماتی با مبانی برنامه نویسی شبکه و مفهوم برنامه های سرویس دهنده / مشتری ، باید سرویس دهنده های استاندارد و مشهوری را که کاربرد فراگیر و جهانی دارند ، معرفی کرده و از دیدگاه مهندسی اینترنت به تحلیل پروتکلهای حاکم بر آنها بپردازیم. یکی از کاربردی‌ترین و عمومی‌ترین کاربردهای شبکهٔ جهانی اینترنت سیستم پست الکترونیکی است که حدود سه دهه قدمت دارد و در خلال این سالها متحول شده و پیشرفتهای چشمگیری داشته است . در این فصل سیستم پست الکترونیکی و پروتکلهای مرتبط را معرفی خواهیم کرد.

کلاً سیستم پست الکترونیکی در دو برنامهٔ مجزا سازماندهی می‌شود:

- **کارگزار کاربر** [۱]: امکان خواندن ، نوشتن و ارسال و دریافت نامه را برای کاربر فراهم می‌کند .
- **کارگزار انتقال پیام** [۲]: انتقال نامه ها را از مبداء به مقصد برعهده دارد.

کارگزار کاربر یک برنامه محلی است که محیطی را برای نوشتن ، خواندن ، ویرایش و نهایتاً تقاضای ارسال یا دریافت نامه فراهم می‌نماید. این برنامه‌ها به چند صورت عرضه شده‌اند :

**الف ) برنامه های مبتنی بر خط فرمان**[۳] : در این محیطها کاربر موظف است با دستوراتی که در خط فرمان اجرا میشوند اقدام به ارسال یا دریافت نامه هایش نماید. مثلاً کاربر در محیط یونیکس دستور mail را در خط فرمان صادر کرده و پیامی را پس از نوشتن ارسال می‌نماید. استفاده از این روش بعلت عدم راحتی کاربران منسوخ شده است.

**ب ) برنامه های مبتنی بر منوها**[۴] : در این برنامه ها کاربران برای انجام عملیات خود بجای استفاده از فرامین ، از منوهای چند گزینه‌ای استفاده می‌کنند . هر چند این محیطها بسیار ساده تر هستند ولی کمتر مورد استفاده قرار می‌گیرند.

**ج ) برنامه های گرافیکی** : در این گونه برنامه ها کاربر تمام فرامین و عملیات مورد نیاز را با استفاده از موس و فشاردادن روی گزینه‌ها و شکلک‌ها[۵] انجام داده و

---

[۱] User Agent
[۲] Message Transfer Agent
[۳] Command Based
[۴] Menu Based
[۵] Icons

محیط برنامه صد درصد گرافیکی و صفحه آرایی شده است لذا استفاده از آنها بسیار ساده و راحت است و امروزه بصورت فراگیر از آنها استفاده می‌شود. ( برنامه Outlook Express یا Eudroa از این دسته هستند)

کلاً هر سیستم پست الکترونیکی حداقل باید امکانات پنج‌گانه زیر را فراهم نماید:

- **امکان ایجاد و نگارش نامه** [۱] : این عملیات به منظور کمک به کاربران برای نوشتن و ویرایش نامه هایشان طراحی میشود و به آنها کمک می‌کند تا آدرسها و فیلدهای لازم را تنظیم نمایند .
- **امکان انتقال** [۲]: به پروتکل یا مجموعه عملیاتی که ارسال یک نامه را از مبداء به مقصد تضمین کند اطلاق میشود. دقّت کنید که اساس این عملیات بر خدمات لایه سوم از شبکه اینترنت بنیان نهاده شده است؛ این عملیات شامل برقراری یک ارتباطِ TCP ، ارسال اطلاعات ، تائید دریافت آن از طرف مقابل و سپس ختم ارتباط می‌باشد. این وظائف به صورت خودکار انجام می‌شود و کاربر هیچ اطلاعی از چگونگی آن ندارد.
- **امکان گزارش گیری** [۳]: به مجموعه عملیاتی اطلاق می‌شود که به شخص ارسال کنندهٔ نامه اطلاعاتی را در مورد سرنوشت نامه‌اش می‌دهد؛ آیا نامه پذیرفته شد یا آنکه بدلائلی از دست رفت؛ همچنین دلائل هر مشکل احتمالی گزارش می‌شود.
- **امکان نمایش** [٤]: به مجموعه عملیاتی اطلاق می‌شود که به کاربر این امکان را می‌دهد که نامه هایش را بخواند و اگر ضمائمی مانند صدا و تصویر به همراه آن است به نحوی ملاحظه و بررسی شود.
- **امکان تصمیم‌گیری** [۵]: به مجموعه عملیاتی گفته می‌شود که به کاربر این امکان را می‌دهد تا در مورد نامه هایش تصمیم بگیرد؛ نامه‌ای قبل از خواندن بصورت خودکار حذف شود ، نامه‌ای بعد از خواندن بصورت خودکار حذف شود؛ برخی از نامه ها هیچگاه حذف نشوند و برخی از نامه ها پس از دریافت بطور خودکار پاسخ داده شوند.

---

۱ Composition
۲ Transfer
۳ Repoting
٤ Displaying
۵ Dispositoin

امکانات پنج گانه‌ای که به آنها اشاره شد حداقل امکانات یک سیستم پست الکترونیکی هستند و امروزه امکانات سطح بالاتری از یک سیستم پست الکترونیکی انتظار می‌رود. بعنوان مثال یک شرکت می‌خواهد لیستی از مشتریان و همچنین لیستی از تولیدکنندگان را تهیه کرده و آنها را در یک فهرست پستی [۱] سازماندهی کند تا هر گاه نامه‌ای را به یکی از این فهرستها ارسال کرد سیستم پست الکترونیکی بطور خودکار یک نسخه از آن نامه را برای تمام اعضای آن فهرست ارسال نماید.

از امکانات مورد انتظار دیگر می‌توان به موارد زیر اشاره کرد:

- ارسال رونوشت یک نامه به دیگران که در نامه های الکترونیکی Cc [۲] نامیده می‌شود. یعنی فرستندهٔ یک نامه به غیر از آدرس فرد گیرندهٔ نامه ، آدرس اشخاص دیگری را که باید نسخه‌ای از آن نامه را دریافت کنند، درج می‌کند و سیستم پست الکترونیکی بصورت خودکار نسخه‌ای از آن نامه را برای گیرندگان رونوشت ارسال می‌کند.

- ارسال رونوشت یک نامه به دیگران بدون اطلاع دریافت کنندگان آن از آدرس سایر گیرندگان که اختصاراً Bcc [۳] نامیده می‌شود.
- پست الکترونیکی رمز شده برای ارسال نامه های محرمانه و سرّی

نکته مهمی که در مورد یک نامه الکترونیکی بایستی بدانید آنست که نامهٔ معمولی بصورت متنی نوشته می‌شود و اطلاعاتی که سرویس دهندهٔ پست الکترونیکی برای ارسال لازم دارد آنهم بصورت متنی به ابتدای نامه اضافه خواهد شد. بنابراین یک سیستم پست الکترونیکی باید بتواند بین مجموعه‌ای از رشته های متنی که بعنوان نامه دریافت می‌کند مشخصات سرآیند نامه را که شامل اطلاعات بسیار مهمی برای رساندن آن به مقصد است از بدنهٔ نامه که آنهم متنی است تمیز بدهد.

بنابراین یک نامهٔ الکترونیکی شامل دو قسمت است :

الف : اطلاعات سرآیند [۴] که شامل اطلاعاتی نظیر آدرس گیرنده ، آدرس فرستنده ، موضوع و... است.

ب : قسمت پیام که بدنهٔ نامه نامیده می‌شود.

---

۱ Mailing list
۲ Carbon copy
۳ Blind Carbon copy
۴ Header

قبل از هر کاری ابتدا استاندارد مربوط به قالب یک نامه الکترونیکی را بررسی می‌کنیم .

دقت کنید هر گونه خطایی در تنظیم قالب یک نامهٔ الکترونیکی ممکن است سرنوشت یک نامه را در رسیدن به مقصد تغییر بدهد، بنابراین شناخت کامل از قالب یک نامهٔ الکترونیکی بسیار مهم است. در ادامه استاندارد قدیمی پست الکترونیکی در مورد قالب نامه‌ها را بررسی کرده و پس از آن به روشهای جدیدتر خواهیم پرداخت.

## ۲) استاندارد RFC 822 : تبیین قالب یک نامه ساده الکترونیکی

در این استاندارد یک نامهٔ الکترونیکی بصورت زیر سازماندهی می‌شود:

- تعدادی فیلد مشخص و تعریف شده که برای عملیات انتقال نامه لازم است. این قسمت سرآیند نامه را تشکیل می‌دهد. ( این فیلدها متنی هستند)
- یک سطر خالی ( به عنوان مرز قسمت سرآیند و بدنهٔ نامه )
- بدنه پیام ( شامل متن اصلی نامه )

فیلدهائی که در سرآیند نامه قرار می‌گیرند (و بعضی الزامی و برخی دیگر اختیاری هستند ) در جدول (۱-۹) آمده است . دقت کنید که ترتیب کوچک و بزرگ بودن حروف هر فیلد برای سرویس دهنده های پست الکترونیکی مهم است و باید دقیقاً بگونه‌ای باشد که در جدول درج شده است؛ هر گونه تغییر در آن ممکن است منجر به بروز خطا شود.

| فیلد | شرح |
|---|---|
| To: | آدرس پست الکترونیکی گیرنده اصلی نامه |
| Cc: | آدرس پست الکترونیکی گیرنده یا گیرندگان ثانویه |
| Bcc: | آدرس پست الکترونیکی گیرنده یا گیرندگان ثانویه بدون اطلاع از آدرس یکدیگر |
| From: | آدرس پست الکترونیکی صاحب اصلی (نویسنده ) نامه |
| Sender: | آدرس پست الکترونیکی فرستنده اصلی نامه |
| Received: | خطی که توسط سیستمهای پست الکترونیکی در بین مسیر اضافه می‌شود. |
| Return-path: | مسیر برگشت نامه را تعریف می‌کند. |

جدول (۱-۹) فیلدهای سرآیند در استاندارد RFC 822

فیلدهای جدول (۱-۹) را جداگانه توضیح می‌دهیم. ذکر این نکته ضروری است که هر فیلد باید در یک سطر مجزا قرار بگیرد و نباید بین آنها سطر خالی وجود داشته باشد.

- فیلد **"To:"** : در جلوی این فیلد (البته پس از یک فاصلهٔ خالی ) آدرس شخص گیرندهٔ نامه قرارمی‌گیرد؛ مثلاً :

To: erlina@cs.uto.edu

همانگونه که در مبحث سیستم DNS اشاره شد ، قسمت سمت راستِ آدرس پست الکترونیکی (یعنی پس از علامت @) نام ماشین سرویس دهندهٔ پست الکترونیکی است که باید به آدرس IP ترجمه شود و سمت چپِ آن نام فردی قرار می‌گیرد که به عنوان یک مشترک بر روی آن ماشین تعریف شده و باید نامه را دریافت نماید.

- فیلد **"Cc:"** : درجلوی این فیلد آدرس پست الکترونیکی شخص دیگری درج می‌شود که باید رونوشتی از این نامه را دریافت نماید.
- فیلد **"Bcc:"** : این فیلد دقیقا همانند قبلی است با این تفاوت که گیرندگان نامه از این موضوع که شخص دیگری این نامه را دریافت کرده مطلع نخواهند شد چرا که در مقصد محتوای این فیلد نشان داده نمی‌شود. دقت کنید که هیچ تفاوتی بین نامه‌ای که هر یک از گیرندگان دریافت می‌کنند وجود نخواهد داشت یعنی اگر جای آدرسها در فیلد **To:** و فیلد **Bcc:** (یا **Cc:** ) عوض شوند اتفاق خاصی نخواهد افتاد.
- فیلد **"From:"** : در جلوی این فیلد آدرس پست الکترونیکی نویسنده و صاحبِ اصلی نامه درج می‌شود.
- فیلد **"Sender:"** : جلوی این فیلد آدرس پست الکترونیکی کسی که حقیقتاً نامه را ارسال کرده است قرار می‌گیرد. بعنوان مثال فرض کنید رئیس یک شرکت به منشی خود دستور می‌دهد نامه‌ای را برای یک موسسه نوشته و ارسال نماید. بنابراین باید در فیلد **From:** آدرس رئیس شرکت ودرجلوی فیلد **Sender:** آدرس پست الکترونیکی منشی شرکت قرار بگیرد. اگر نویسندهٔ اصلی نامه و ارسال کننده آن هر دو یکی باشند نیازی به فیلد **Sender:** نخواهد بود.
- فیلد **"Received:"** : گاهی بین سیستم پست الکترونیکی فرستندهٔ نامه و سیستم گیرندهٔ نامه ، سرویس دهنده های دیگری بعنوان واسطه های انتقال[1] وجود دارد. این

[1] Transfer Agent

نمایندگیها پس از دریافت نامه ، سطری حاوی فیلد **Received:** ، هویّت و آدرس خود ، ساعت و تاریخ دریافت پیام و هرگونه اطلاعاتی که می‌تواند برای ردّیابی مسیر مفید باشد به آن اضافه می‌نمایند. برای مثال فرض کنید یک دانشگاه دارای سیستم پست الکترونیکی است که تمام نامه ها را دریافت و بین سرویس دهنده های پست الکترونیکی در هر دانشکده توزیع می‌کند. سرویس دهندهٔ کل دانشگاه که با دنیای خارج در ارتباط است بعنوان واسطه انتقال شناخته می‌شود.

- فیلد **"Return-path:"** : آخرین واسطهٔ انتقال این فیلد را به نامه اضافه می‌نماید و مشخص می‌کند که پاسخ نامه چگونه باید به فرستندهٔ آن برگردد. از لحاظ تئوری این اطلاعات را می‌توان از فیلد **Received:** استخراج کرد.

به غیر از فیلدهائی که توضیح داده شد فیلدهای اضافی دیگری هم وجود دارد که توسط سیستم انتقال استفاده و پردازش نمی‌شود بلکه فقط برای اطلاع شخص خوانندهٔ نامه یا برنامه‌ای که کاربر برای خواندن نامه‌اش از آن استفاده می‌کند مفید خواهد بود و وجود و عدم وجود آنها مهم نیست . این فیلدها در جدول (۲-۹) معرفی شده‌اند که در ادامه آنها را معرفی می‌کنیم .

| فیلد سرآیند | شرح |
|---|---|
| Date: | **تاریخ و زمان ارسال پیام (نامه)** |
| Replay-To: | **آدرس پست الکترونیکی کسی که باید پاسخ این نامه را دریافت نماید.** |
| Message-Id: | **یک شماره منحصر بفرد برای آنکه بتوان بعداً به آن شماره استناد کرد.** |
| In-Replay-To: | **شماره نامه‌ای که این نامه در پاسخ به آن نامه می‌باشد.** |
| References: | **شماره های دیگری که این نامه با آنها مرتبط است.** |
| Keywords | **برخی از کلمات کلیدی از مضمون نامه که توسط نویسندهٔ نامه انتخاب می‌شود.** |
| Subjects | **موضوع نامه (خلاصه بسیار کوتاهی از نامه فقط در یک خط)** |

جدول (۲-۹) فیلدهای اختیاری در استاندارد RFC 822

- فیلد **"Date:"** : در جلوی این فیلد تاریخ و زمان ارسال نامه قرار می‌گیرد. معمولا تاریخ و زمان بر حسب ساعت گرینویچ[1] (GMT) درج می‌شود.
- فیلد **"Reply-To:"** : این فیلد زمانی بکار می‌رود که نویسنده اصلی نامه تمایلی به دریافت پاسخ آن نامه نداشته باشد بلکه بخواهد پاسخ نامهٔ ارسالی توسط شخص ثالثی دریافت شود. بعنوان مثال مدیر بازاریابی یک شرکت برای مشتریان خود نامه‌ای را تنظیم کرده و منشی او آنرا ارسال می‌نماید. پاسخ نامه های ارسالی باید به مدیر فروش شرکت ارجاع داده شود بنابراین در جلوی فیلد **Reply-To:** آدرس مدیر فروش درج می‌شود.
- فیلد **"Message-Id:"** : در جلوی این فیلد یک شمارهٔ منحصر به فرد و اختیاری قرار می‌گیردکه بتوان برای نامه های پیرو که بعداً ارسال می‌شود به آن استناد کرد.(دقیقاً مثل شماره نامه های اداری )
- فیلد **"In-Reply-To:"** در جلوی این فیلد شماره نامه‌ای قرار میگیرد که نامه فعلی در پاسخ به آن ارسال شده است .
- فیلد **"References:"** : در جلوی این فیلد شماره نامه های دیگری قرار میگیرد که نامه فعلی به موضوع آن نامه ها مرتبط است.
- فیلد**"Keywords:"** : برخی از کلمات کلیدی که با متن و موضوع نامه مرتبط است و برنامهٔ نامه‌خوان میتواند آنها را ملاک دسته بندی یا جستجو قرار بدهد.
- فیلد **"Subject:"** : یک کلمه یا جمله کوتاه که مضمون نامه را برای خواننده آن مشخص می‌نماید.

اگر چه فیلدهای فوق اختیاری هستند ولی اگر کمی به آنها دقّت کنید متوجه خواهید شد که در یک شرکت که ممکن است روزی صدها نامهٔ الکترونیکی دریافت کند این فیلدها چقدر می‌توانند مفید واقع شوند و یک نرم‌افزار نامه‌خوان برای دسته بندی ، جستجو ، تصمیم گیری برای سرنوشت نامه‌ها یا بایگانی آنها ، چگونه می‌تواند از آنها استفاده کند.

در استاندارد RFC 822 شخص کاربر می‌تواند فیلدهایی را برای استفاده خودش یا نرم‌افزاری که برای خواندن نامه هایش استفاده می‌کند تعریف نماید. تمام این فیلدها که همانند بقیهٔ فیلدها در قسمت سرآیند نامه درج می‌شوند بایستی حتما با دو حرف **X-** شروع شوند. این

---

[1] Greenwich Mean Time

فیلدها فقط برای یک نرم‌افزار خاصّ یا شخص کاربر مفید خواهد بود و سرویس دهنده‌های پستی که وظیفه انتقال و ذخیرهٔ نامه ها را بر عهده دارند آنها را نادیده می‌گیرند. مثلاً فرض کنید یک موسّسهٔ خوش ذوق بخواهد در یک فیلد از هر نامه‌ای که ارسال می‌کند (بطور خودکار و توسط نرم‌افزار ) یک بیت شعر یا ضرب المثل قرار بدهد. دراین حالت در قسمت سرآیند نامه ، فیلد زیر قابل تعریف است :

**X-Proverb:** ..... ضرب المثل .....

(البته باید نرم‌افزار نامه‌خوان شخص گیرنده بتواند این فیلد را تمیز داده و نشان بدهد)

یادآوری این نکته بسیار مهم خواهد بود که هر فیلد و رشته‌ای که جلوی آن نوشته می‌شود باید در یک سطر مجزا قرار بگیرد و بین سطرهای سرآیند سطر خالی نباشد چرا که سطر خالی به منزله خاتمه قسمت سرآیند نامه و شروع متن (بدنه) نامه تلقی می‌شود و نرم‌افزار نامه‌خوان درون بدنهٔ نامه را پردازش نخواهد کرد.

استاندارد RFC 822 تقریبا کامل و جامع است ولی فقط زمانی بکار می‌آید که هدف ما ارسال نامه هائی باشد که مطلقاً متنی و به زبان انگلیسی هستند. حال فرض کنید بخواهید نامه‌ای جهت تبریک سال نو به زبان فارسی بهمراه یک فایل تصویر بعنوان کارت پستال برای یک دوست ارسال نمایید یا بخواهید قطعهٔ کوتاهی از فیلم جشن تولد فرزندتان به نامه شما ضمیمه شود. در چنین مواردی استاندارد RFC 822 که مربوط به دو دهه قبل است جوابگو نخواهد بود. استاندارد جدید MIME ضمن پشتیبانی از RFC 822 این نواقص را برطرف کرده است .

## ۳) استاندارد MIME : سیستم نامه رسانی توسعه یافته در اینترنت

در این استاندارد ایدهٔ اصلی آن بوده است که بدون پشت پا زدن به استاندارد RFC 822 ( که در آن زمان بسیار محبوبیت داشت ) روشی ابداع شود تا بتوان فایلهای غیر اَسکی همانند فایلهای اجرایی ، صدا و تصویر بگونه‌ای در بدنهٔ نامه قرار گیرد که براساس سرویس‌دهنده‌های قدیمی قابل ارسال و دریافت باشد؛ در این صورت بدون نیاز به تغییر سرویس‌دهنده‌های قبلی ، فقط باید برنامه نامه‌خوان در سمت کاربر عوض شود که هزینه کمی را به کاربر تحمیل می‌کند.

استاندارد MIME پنج فیلد جدید در سرآیند نامه تعریف کرده است که این فیلدها در جدول (۳-۹) معرفی شده‌اند.

| توضیح | سرآیند |
|---|---|
| شمارهٔ نسخهٔ MIME | MIME-Version: |
| یک سطر که مضمون کلّی نامه را مشخص می‌نماید. | Content-Description: |
| یک مشخصه یا شماره منحصر به فرد | Content-Id: |
| طریقه کدگذاری محتوای نامه | Content-Transfer-Encoding: |
| نوع و محتوای نامه | Content-Type: |

جدول (۳-۹) سرآیندهای جدید در استاندارد MIME

• فیلد **"MIME-Version:"** : این فیلد به برنامه نامه‌خوان در سمت کاربر تفهیم می‌کند که این نامهٔ الکترونیکی با استاندارد MIME سازماندهی و ارسال شده است؛ در ضمن نسخهٔ استاندارد MIME را نیز مشخص می‌نماید. نامه هائی که این فیلد را در سرآیند نامه نداشته باشند (همانند نامه های قدیمی با استاندارد RFC 822 ) بصورت نامه‌ای تماماً متنی با کدهای اَسکی تلقی می‌شوند.

• فیلد **"Content-Description:"** : متنی که در جلوی این فیلد قرار می‌گیرد مضمون و محتوای نامه را مشخص می‌کند. گیرندهٔ نامه با استفاده از این فیلد می‌تواند تشخیص بدهد که آیا رمزگشائی و خواندن پیام ارزشمند است یا نه.

• فیلد **"Content-Id:"** : این فیلد که مشابه فیلد :Message–Id در استاندارد RFC 822 است شماره یا رشته‌ای است منحصر به فرد که می‌توان به عنوان شمارهٔ نامه در نامه های بعدی به آن استناد کرد.

• فیلد **"Content-Transfer-Encoding:"** : در جلوی این فیلد عبارتی قرار میگیرد که به برنامهٔ نامه‌خوان در سمت کاربر تفهیم می‌کند که چه قاعده‌ای را برای دیکود[۱] کردن بدنهٔ نامه بکار ببرد . بگونه‌ای که اشاره شد بر خلاف استاندارد RFC 822 در بدنهٔ نامه های مبتنی بر استاندارد MIME می‌تواند کدهای غیر اَسکی ، فایلهای صدا ، تصویر یا کلاً هر فایل دودویی قرار بگیرد. بنابراین در مقصد قبل از نمایش محتوای نامه ، باید قسمت بدنهٔ آن پردازش و دیکود شود. اگر بدون رمزگشائی ، نامه را نگاه کنید یکسری کاراکترهای نامفهوم خواهید دید .انواع کدگذاری[۲] در استاندارد MIME به شرح زیر است:

---

[۱] Decoding
[۲] Encoding

➢ کدهای ASCII ساده که باید کدهای بین صفر تا ۱۲۷ باشند. (یعنی حروف، علائم و کاراکترهای صرفاً انگلیسی) تنها محدودیتی که وجود دارد آنست که هر خط از متن نامه نبایستی از ۱۰۰۰ کاراکتر تجاوز کند.

➢ کدهای ASCII توسعه یافته[1] که می‌تواند تمام ۲۵۶ کاراکتر جدول اَسکی را شامل بشود. نکته‌ای که وجود دارد آنست که چون استاندارد معینی برای کاراکترهای بالای جدول اَسکی وجود ندارد لذا ارسال نامه هائی که از این کاراکترها استفاده می‌کنند ممکن است در مقصد با مشکل نمایش مواجه شوند. برای روشن شدن قضیه فرض کنید کسی نامه‌ای را بنویسد که در آن از حروف فارسی با کدهای بالای ۱۲۸ استفاده کرده باشد. حال وقتی این نامه ارسال شد ممکن است در مقصد بصورت یک متن کاملاً نامفهوم و مبهم دیده شود چرا که ماشین مقصد کدهای بالای ۱۲۸ در جدول اسکی را به صورت دیگری تفسیر کرده است. در اینجا هم حداکثر طول هر خط ۱۰۰۰ کاراکتر است.

➢ کد گذاری base64 : این روش که به آن ASCII Armor هم گفته می‌شود در مواقعی کاربرد دارد که بخواهید یک فایل دودویی (مثل یک فایل اجرائی یا فایل تصویر) را در بدنهٔ نامه جا سازی نمائید. در چنین مواقعی بهترین راه حل ممکن آن است که فایل به نحوی به کاراکترهای ASCII تبدیل شده و درون متن نامه قرار بگیرد . (برای روشن شدن قضیه فرض کنید نامه‌ای نوشته و به آن فایل تصویر ضمیمه می‌کنید؛ این فایل تصویر بصورت کدهای ASCII در متن نامه قرار می‌گیرد و شما می‌توانید آن کدها را توسط یک ویرایشگر ساده مثل Notepad ببینید)

روش تبدیل فایلهای باینری به کدهای ASCII بشرح زیر انجام می‌شود:

از درون فایل دودویی سه بایت سه بایت جدا می‌شود. (سه بایت مجموعاً ۲۴ بیت خواهد شد) این ۲۴ بیت به چهار قسمت شش بیتی تقسیم می‌گردد. هر قسمت شش بیتی مجموعا ۶۴ حالت دارد (از صفر تا ۶۳) که برای حالت صفر کاراکتر 'A' و برای ۱ کاراکتر 'B' و به همین ترتیب تا ۲۵ که 'Z'قرار می‌گیرد؛ برای ۲۶ تا ۵۱ به ترتیب 'a' تا'z' و برای ۵۲ تا ۶۱ به ترتیب کاراکترهای'0' تا '9' قرار می‌گیرد؛ برای ۶۲ و ۶۳ به ترتیب کاراکترهای '+' و '/' جایگزین می‌شود.

ممکن است تعداد بایتهای یک فایل دودویی ضریبی از ۳ نباشد. بنابراین در دسته آخر ممکن است فقط یک یا دو کاراکتر باقی مانده باشد. در این حالت اگر در گروه

[1] Extended ASCII

آخر یک کاراکتر باقی مانده باشد پس از تبدیل دو علامت == و اگر دو کاراکتر باقی مانده است یک علامت '=' قرار می‌گیرد. طرح یک مثال بسیار روشنگر خواهد بود: فرض کنید بخواهیم پنج بایت زیر را طبق روش فوق به کدهای ASCII Armor تبدیل نمائیم:

| 0x20 | 0x12 | 0x12 | 0xE1 | 0xA1 | - |
|---|---|---|---|---|---|
| 00100000 | 00010010 | 00010010 | 11100001 | 10100001 | 00xxxxxx |

| 001000 | 000001 | 001000 | 010010 | 111000 | 011010 | 000100 | xxxxxx |
|---|---|---|---|---|---|---|---|
| I | B | I | S | 4 | a | E | = |

با روش فوق هر فایل دودویی به حالت متنی تبدیل می‌شود. در مقصد برنامه نامه‌خوان وقتی گزینه زیر را در متن ببیند به راحتی آنرا به حالت اصلی برخواهد گرداند:

Content-Transfer-Encoding: base64

دقت کنید هر فایل که به روش فوق کد شود فقط شامل حروف کوچک و بزرگ انگلیسی ، کاراکترهای '0' تا '9' و علامتهای '+' و '/' و '=' خواهد بود و در مقصد هر کاراکتر که به غیر از کاراکترهای ذکر شده لابلای آن وجود داشته باشد به سادگی حذف خواهد شد؛ چون در استاندارد MIME هر سطر می‌تواند حداکثر هزار کاراکتر باشد لذا پس از تبدیل یک فایل دودویی به حالت ASCII Armor می‌توان در هر جای متن کد '\n' را اضافه کرد تا هر سطر زیر هزار کاراکتر باشد؛ این کدها در مقصد حذف خواهند شد. این نکته نیز قابل توجه است که طول یک فایل دودویی پس از تبدیل به حالت ASCII Armor حداقل با ضریب ۴/۳ افزایش می‌یابد.

➢ کد گذاری qouted-printable : استفاده از روش قبل برای تبدیل فایلهایی که تعداد کمی کاراکتر با کد بالای ۱۲۸ دارند راه مناسبی نیست چون طول فایل بیهوده افزایش می‌یابد. در این روش برای کاراکترهایی که کد آنها زیر ۱۲۸ است خود کاراکتر ولی برای آنهائی که کدشان بین ۱۲۸ تا ۲۵۵ است ابتدا علامت '=' و بعد دو کاراکتر معادل کد مبنای ۱٦ آن درج می‌شود. به عنوان مثال چهار بایت زیر به روش بالا کد شده‌اند:

| 01000110 | 10100111 | 11110011 | 01000001 |
|---|---|---|---|
| E | =A7 | =F3 | **A** |

روش کدگـذاری qouted-printable فقـط زمـانی مفیـد اسـت کـه نسـبت کاراکترهـای بـالای ۱۲۸ در متـن بسـیار کـم باشـد چـرا کـه هـر کـاراکتر بـالای ۱۲۸ ، پـس از تبدیـل بـا سـه کاراکتر جاینشین خواهـد شـد .

➢ کد گذاری تعریـف شـده توسـط کـاربر: اگـر هیچکـدام از روشـهای کـدگـذاری قبـل را نپسـندیدید می‌توانیـد خودتـان یـک سـبک کـدگـذاری ابـداع کنیـد بشـرطی کـه گـیرنده نامـه شـما قـادر باشـد متـن شـما را از حـالت کـدگـذاری خـارج نمـاید و بنـابراین بایسـتی نرم‌افزار نامه‌خوان یا نرم‌افزار واسـطه‌ای بـرای ایـن کـار طراحـی نمـایید.

• فیلـــد**"Content-Type:"** : آخریـــن فیلـــد ســـرآیند در اســـتاندارد MIME یکـــی از کـاربردی‌ترین فیلدهـا خواهـد بـود و مشـخصات محتـوای نامـه را تشـریح خواهـــد کـــرد. بعنـوان مثـال در قسـمت سـرآیند یـک نامـه ایـن فیلـد می‌توانـد بصـورت زیـر تنظیـم شـده باشـد:

Content-Type: Video/Mpeg

بـه معنـای آنکـه محتـوای بدنـه فـایلی اسـت ویدئوئـی بـا قـالب MPEG و بـالطبع نرم‌افـزار نامه‌خوان باید قـادر باشـد ضمـن اسـتخراج آن از متـن ، ابـزار نمـایش آنـرا هـم بـارگـذاری نمـاید.

حـال ببینیـم محتـوای نامـه الکـترونیکی چـه انواعـی را در برمـی‌گـیرد. انـواع محتویـات متن یک نامـهٔ الکـترونیکی بـا اسـتاندارد MIME در جـدول (٤-٩) مشـخص شـده اسـت .

➢ نوع **Text** : به معنای آنست کـه نامـه از نـوع متنـی اسـت. انـواع آن عبارتسـت از

- ♦ Text/plain : نامـه معمولـی بـا کاراکترهـای ASCII
- ♦ Text/Richtext : نامـه در قـالب یـک زبـان نشـانه گـذاری (همـانند HTML ) ارسـال شده است ، بنـابراین نرم‌افـزار نامـه‌خوان بایسـتی قبـل از نمـایش نامـه ، متـن را تفسـیر نماید. در حقیقت در ایـن نـوع نامـه در لابـلای محتـوای آن بـرچسـبهایی بـرای صفحـه آرایی و قالب بنـدی قـرار داده می‌شـود .

| نوع کلّی | نوع دقیق | شرح |
|---|---|---|
| Text | Plain | متن ساده معمولی |
| | Richtext | متن حاوی دستورات قالب بندی |
| Image | Gif | فایل تصویر با قالب GIF |
| | Jpeg | فایل تصویر با قالب JPEG |
| Audio | Basic | فایل صوتی با قالب snd |
| Video | Mpeg | فایل ویدئوئی با قالب MPEG |
| Application | Octet-stream | دنباله‌ای از بایتهای تفسیر نشده |
| | Postscript | متن تنظیم شده در پست اسکریپت |
| | Rfc822 | متن تنظیم شده در استاندارد RFC822 |
| Message | Partial | متن به منظور انتقال تکه تکه شده است. |
| | External-body | متن پیام باید از شبکه اینترنت بارگذاری شود. |
| | Mixed | متن دارای چندقسمت است که ترتیب مشخص دارد. |
| Multipart | Alternative | متن دارای چند قسمت با قالبهای متفاوت است. |
| | Parallel | قسمتهای مختلف متن باید همزمان ملاحظه شود. |
| | Digest | متن شامل چندقسمت است و هر قسمت از نوع RFC822 است. |

جدول (٤-٩) انواع محتویات متن در استاندارد MIME

- نوع **Image** : به معنای آن است که درون متن ، یک تصویر جاسازی شده است. نوع تصویر می‌تواند یکی از دو قالب زیر باشد:
  - Image/Gif : محتوای بدنه ، تصویری با قالب GIF می‌باشد.
  - Image/Jpeg : محتوای بدنه ، تصویری با قالب JPEG می‌باشد
- نوعِ **Audio** : متنِ نامه محتوی فایلی از نوع صداست. (با قالب snd )
- نوع **Video** : متنِ نامه محتوی فایلی از نوع ویدیو با قالب MPEG است.
- نوع **Application** : متنِ نامه محتوی فایلی کاربردی است و خودش به دو نوع دیگر تقسیم میشود:
  - Application/Octet-stream : مجموعه‌ای از بایتهای متوالی که می‌تواند یک فایل اجرایی یا هر فایل داده باشد.
  - Application/ postscript : یک فایل قابل چاپ با قالب پُست‌اسکریپت

- نوع **Message** : بدین معناست که درون بدنه یک پیام دیگر جاسازی شده است و بنابراین بایستی برای استخراج بقیه قسمتها عملیات اضافه تری انجام شود انواع آن به شرح ذیل است:
  - Message/Rfc822 : درون بدنهٔ نامه یک نامهٔ دیگر با استاندارد RFC 822 جاسازی شده است.
  - Message/Partial : بدین معناست که نامه اصلی چون بزرگ بوده به چندین قسمت شکسته شده است و برنامهٔ نامه‌خوان در سمت کاربر باید آنها را کنار هم قرار داده و بازیابی نماید. (در حقیقت نامه اصلی قطعه قطعه شده و باید هر قطعه شماره‌ای داشته باشد تا بتوان آنها را بازسازی کرد)
  - Message/External-body : با این نوع می‌توان به نرم‌افزار نامه‌خوان تفهیم کرد که باید یک فایل طولانی را با استفاده از پروتکلهای دیگر مثل FTP از روی شبکه استخراج نماید . با این روش بجای آنکه یک فایل حجیم صدا و تصویر را به نامه ضمیمه کرده و داده های آنرا در متن نامه قرار بدهیم می‌توان آدرس URL آنرا به خوانندهٔ نامه ارائه کرده تا در صورت تمایل آنرا روی سیستم خود بار نماید. نامه‌خوان باید امکان بارگذاری چنین فایلهایی را از روی شبکه داشته باشد.

- نوع Multipart : این گزینه شاید مهمترین نوع نامه‌های الکترونیکی به شمار بیاید چرا که با این گزینه می‌توان در بدنهٔ نامه تمام انواع متن ، صدا ، تصویر و فایلهای اجرائی را جاسازی کرد بگونه‌ای که ابتدا و انتهای هر بخش دقیقاً مشخص بوده و نامه‌خوان بتواند آنها را از هم تفکیک نماید . در انتهای این بخش یک مثال از این نوع را خواهیم داشت. انواع گزینهٔ Multipart به شرح ذیل است :
  - Multipart/Mixed : بدین معناست که نامه خودش شامل چندین نامه دیگر با استاندارد RFC 822 است .
  - Multipart/Alternative : بدین معناست که نامه دارای چندین قسمت متفاوت شامل متن ، صدا ، تصویر یا فایلهای دودویی می‌باشد. در این نوع نامه مرز هر قسمت از نامه با بقیه قسمتها دقیقاً مشخص می‌شود؛ در مثالی چگونگی آنرا تشریح خواهیم کرد.
  - Multipart/Parallel : این گزینه همانند قبلی است یعنی نامه شامل چندین بخش مختلف است ولی تفاوت عمده آن با گزینهٔ قبلی در آن است که نرم‌افزار

نامه‌خوان موظف است قسمتهای متفاوت را بطور همزمان دیکود و اجرا نماید. مثالی از این نوع نامه آنست که محتوای یک نامه شامل یک قطعه انیمیشن و یک قطعه صدا باشد و از نرم‌افزار نامه‌خوان انتظار داشته باشیم که قطعه صدا همزمان با نمایش انیمیشن به اجرا دربیاید.

- Multipart/Digest : این گزینه وقتی مورد استفاده قرار می‌گیرد که پیامها با یکدیگر ترکیب شده و پیام بزرگتری را ایجاد می‌کنند بعنوان مثال برخی از گروههای خبری در اینترنت پیامهای دسته‌ای از مشترکین را جمع آوری کرده و سپس آنرا بعنوان یک نامهٔ Multipart/Digest ارسال می‌نمایند.

```
From: erlinor@abc.com
To: carolyn@xyz.com
MIME-Version: 1.0
Message-Id: <0704760941 AA00747@abc.com>
Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
Subject: Earth orbits sun integral number of times

This is preamble. The user agent ignore it. Have a nice time.
--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/richtext

Happy birthday to you
Happy birthday dear <bold>Carolyn</bold>

--qwertyuiopasdfghjklzxcvbnm
Content-Type: message/external-body;
    Access-type="anon-ftp";
    Site="bicycle.abc.com";
    Directory="pub";
    Name="birthday.snd";

Content-Type: audio/basic
Content-Transfer-Encoding: Base64
--qwertyuiopasdfghjklzxcvbnm
داده های صدا در ادامهٔ نامه قرار می‌گیرد که در این مثال حذف شده است.
```

مثال (۵-۹) قالب یک نامه الکترونیکی با استاندارد MIME

در مثال (۵-۹) قالب یک نامه الکترونیکی بااستاندارد MIME آورده شده است. اگر به این مثال دقت شود در خط پنجم ، عبارت ذیل دیده می‌شود:

Content-Type: Multipart/alternative;boundary=qwertyuiopasdfghjklzxcvbnm

این فیلد دو موضوع را تبیین می‌کند:

اول آنکه نامه دارای چندین بخش مجزا و متوالی است و هر قسمت از آن ، نوع و محتوای متفاوتی دارد.

دوم آنکه نامه‌خوان موظف است ابتدا و انتهای هر قسمت را با رشتهٔ کاراکتری زیر جدا کرده و هر قسمت را مجزا پردازش و دیکود نماید:

--qwertyuiopasdfghjklzxcvbnm

رشته‌ای که بعد از دو کاراکتر -- آمده است همان رشته‌ای است که بعنوان متمایز کننده بخشها انتخاب شده است . دقت کنید که معمولاً این رشته طولانی و تصادفی است تا درون متن مشابه آن یافت نشود. دو کاراکتر -- نیز نشاندهندهٔ خطٌ تمایز می‌باشد و باید دقیقاً در ابتدای سطر ظاهر شود.

نکته نهائی آنکه فقط وقتی مجبور خواهید بود فیلدها و سرآیندهای معرفی شده را بکار ببرید که بخواهید ساختار نامه را توسط یک ویرایشگر ساده مثل Notepad ایجاد کنید یا خودتان نرم‌افزار نامه‌خوان بنویسید و گرنه این فیلدها بصورت خودکار توسط نرم‌افزار نامه‌خوان شما ایجاد خواهد شد.

## ۴) پروتکل ساده انتقال نامه های الکترونیکی: SMTP[1]

پس از تحلیل ساختار متنی یک نامهٔ الکترونیکی باید ببینیم که برای ارسال نامه یا دریافت آن ، از دیدگاه برنامهٔ سرویس دهنده و مشتری چه اتفاقاتی می‌افتد. مشهورترین سرویس دهندهٔ پست الکترونیکی ، SMTP نام دارد که روند عملیات آن بسیار ساده است :

ماشین مبداء (یعنی ماشینی که می‌خواهد نامهٔ نوشته وتنظیم شده‌ای را ارسال کند) با پورت شماره ۲۵ از ماشین مقصد که سرویس دهندهٔ STMP روی آن اجرا شده یک ارتباط TCP برقرار میکند. بنابراین براحتی می‌توانید در ذهن خود مجسم کنید که برنامهٔ سرویس دهنده یک برنامهٔ سوکت است که به پورت ۲۵ گوش می‌دهد (این

[1] Simple Mail Transfer Protocol

برنامه در محیط یونیکس به نام دایمون SMTP معروف است . دایمونها برنامه هایی هستند که در حالت انتظار می‌مانند و با یک سیگنال شروع به انجام عملیات خود می‌نمایند) این برنامه ارتباطات TCP به پورت ۲۵ را می‌پذیرد.

پس از برقراری ارتباط و پذیرش آن توسط سرویس دهنده ، شروع کنندهٔ ارتباط (یعنی نرم‌افزار مشتری یا همان نامه‌خوان ) باید آنقدر صبر کند تا سرویس دهندهٔ مقصد با ارسال یک پیغام اعلام آمادگی نماید. روند اعلام آمادگی و بقیه مراحل مبادلهٔ نامه بصورت زیر است:

- سرویس دهنده با ارسال یک رشته متنی که معمولا بصورت زیر است به برنامه مبدا اعلام آمادگی می‌نماید:

  220 آدرس نام حوزه خود SMTP service ready

  مثال :

  220 xyz.com STMP service ready

- پس از اعلام آمادگی (کد 220 بمعنای اعلام آمادگی است ) برنامهٔ مبداء با ارسال یک رشته که حاوی کلمه HELO (مخفف کلمه سلام) و همچنین آدرس نام حوزه خودش می‌باشد هویت خود را برای سرویس دهنده آشکار می‌کند. مثال :

  HELO abc.com

- پس از آنکه سرویس دهنده هویت فرستندهٔ پیام را ارزیابی کرد در صورتی که تمایل به دریافت نامه داشته باشد با کد ۲۵۰ و رشته‌ای که در ادامهٔ آن می‌آید اعلام آمادگی می‌نماید. مثال:

  250 xyz.com says hello to abc.com

- سرویس دهنده صاحب نامه را بررسی کرده و در صورتی که منعی برای دریافت نامهٔ چنین شخصی وضع نشده باشد مجدداً با کد ۲۵۰ و رشته‌ای که در ادامه می‌آید اعلام آمادگی می‌کند. مثال:

  250 sender ok

- برنامه مبدا گیرندهٔ نامه را معرفی می‌کند. مثال:

  PCPT TO:<carolyn@xyz.com>

- بار دیگر سرویس دهنده ، گیرندهٔ نهایی نامه را ارزیابی کرده و بررسی می‌کند که آیا چنین شخصی (در مثال بالا Carolyn ) وجود دارد یا خیر . در صورتی که امکان

دریافت نامه وجود داشته باشد برای بار سوم با کد ۲۵۰ مطابق مثال زیر اعلام آمادگی می‌شود:

250 recipient ok

- برنامه مبداء اعلام می‌کند که برای ارسال داده ها که کلاً کاراکترهای اسکی با کد زیر 128 هستند آماده است؛ کلمه DATA بدون هیچ حرف اضافه به عنوان اعلام آمادگی برای ارسال است. مثال:

DATA

- سرویس دهنده ضمن اعلام آمادگی جهت دریافت داده ها به مبدا اعلام می‌کند که پس از آخرین سطرِ نامه یک خط که فقط شامل تک کاراکتر '•' است ارسال کند تا انتهای نامه مشخص باشد. مثال:

354 Send mail; end with "." on a line by itself

- مبداء ، نامه‌ای را که با استاندارد RFC822 یا MIME تنظیم شده است ، ارسال می‌کند؛ در انتهای نامه خطی که شامل تک حرف '•' است به معنای خاتمهٔ نامه ، ارسال می‌شود.
- سرویس دهنده دریافت موفقیت‌آمیز نامه را اعلام می‌کند؛ این اعلام بصورت زیر است:

250 message accepted

- مبدا با ارسال رشتهٔ QUIT اعلام خروج می‌کند. ( البته می‌تواند مجدداً از مرحله دوم شروع کرده و نامهٔ دیگری را ارسال کند.)

QUIT

- فرستنده ضمن تائید خروج و معرفی مجدّد خود اعلام می‌کند که ارتباط TCP را قطع خواهد کرد ، در این جا کار انتقال خاتمه یافته است. مثال:

221 xyz.com closing connection

به یک انتقال واقعی پیام در مثال (٦-٩) دقت نمائید. در این مثال نامهٔ الکترونیکی مثال (٥-٩) طبق روال فوق بین مبدا و مقصد مبادله شده است. در این مثال داده هائی که توسط مبداء ارسال شده با C: و آنهائی که توسط سرویس دهنده ارسال شده با S: نشان داده شده است . (C: , S: ارسال نمی‌شود)

سیستم پست الکترونیکیِ SMTP در مواردی بهبود داده شده که مشخّصات آن در RFC-1425 آمده است.

```
S: 220 xyz.com service ready
C: HELO abc.com
S: 250 xyz.com says hello to abc.com
C: MAIL FROM: <erlinor@abc.com>
S: 250 sender ok
C: RCPT TO: <carolyn@xyz.com>
S: 250 recepient ok
C: DATA
S: 354 Send mail; end with "." on a line by itself
C: From: erlinor@abc.com
C: To: carolyn@xyz.com
C: MIME-Version: 1.0
C: Message-Id: <0704760941 AA00747@abc.com>
C: Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
C: Subject: Earth orbits sun integral number of times
C:
C: This is preamble. The user agent ignore it. Have a nice time.
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: text/richtext
C:
C: Happy birthday to you
C: Happy birthday dear <bold>Carolyn</bold>
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: message/external-body;
C: Access-type="anon-ftp";
C: Site="bicycle.abc.com";
C: Directory="pub";
C: Name="birthday.snd";
C:
C: Content-Type: audio/basic
C: Content-Transfer-Encoding: Base64
C: --qwertyuiopasdfghjklzxcvbnm
C: .
S: 250 message accepted
C: QUIT
S: 221 xyz.com closing connection
```

مثال (٦–٩) مراحل انتقال یک نامهٔ الکترونیکی

## ۵) تحویل نهائی نامه[۱]

برای شروع این بخش شما را با یک سوال روبرو می‌کنیم : کاربری که بطور نامنظم و پراکنده به شبکه اینترنت متصل می‌شود چگونه می‌تواند روی سیستمش SMTP را نصب و شبانه روز سیستم خود را برای دریافت نامه های خود روشن و فعال نگه دارد؟

جواب این سوال ساده است کاربر باید وظیفهٔ دریافت نامه هایش را به یک کارگزار مطمئن که بصورت دائم فعال است و سیستمِ SMTP را فراهم کرده بسپارد. هر موقع که نامه‌ای برای او ارسال می‌شود کارگزارش آنرا دریافت و ذخیره می‌نماید. هنگامی که کاربر تمایل داشت سری به نامه های رسیده‌اش بزند و نامه‌ای را دریافت نماید بایستی با این کارگزار ارتباط برقرار نماید.

POP3 [۲] پروتکلی ساده برای دریافت نامه‌های الکترونیکی از سرویس‌دهندهٔ کارگزار شما است. این پروتکل مجموعه‌ای از فرامین برای برقراری اتصال ، قطع اتصال، دریافت پیام ها و حذف آنها می‌باشد. این پروتکل نیز همانند SMTP فرامین متنی دارد.

بدون آنکه بخواهیم وارد نکات ریز این پروتکل بشویم امکانات کلی آنرا معرفی می‌کنیم .

- **نصب فیلتر**: شما از سیستم پست الکترونیکی می‌خواهید که نامه های دریافتی از یک آدرسِ خاصّ را اصلا تحویل نگیرد یا نامه هائی که قسمت موضوع[۳] آن شامل کلمات کلیدی خاص می‌شود را حذف کند. یا مثلا نامه هائی را که کلمه‌ای خاص در آدرس فرستنده‌اش یافت می‌شود حذف نماید (این امکان برای رهائی از شر مزاحمت شرکتهای تبلیغاتی که پیاپی نامه ارسال می‌کنند بسیار مفید است )
- **ارسال نامه های رسیده به آدرسی دیگر**[۴] : فرض کنید که در شهر خود از کارگزاری صندوق پست الکترونیکی گرفته‌اید و برای مدتی به خارج از کشور سفر می‌کنید ، می‌توانید از سیستم پستی خود بخواهید که نامه های شما به آدرس پستی جدید ارسال شود. همچنین این امکان وجود دارد که یک نامه را بدون دخل و تصرف به آدرسی دیگر ارسال کنید.

---

۱ Final Delivery
۲ Post Office Protocol
۳ Subject
۴ Forwarding

- **Vacation Daemon** : فرض کنید می‌خواهید برای چند روزی جائی بروید که دسترسی به اینترنت ندارید. می‌توانید سیستم پستی را وادار کنید که ضمن دریافت نامه ها یک پیغام برای ارسال کنندگان نامه بفرستد. مثلاً یک شرکت که هر روز نامه‌های زیادی را دریافت می‌کند و ممکن است پاسخ دستی به آنها طولانی مدت شود ، می‌تواند از سیستم پستی بخواهد که بصورت خودکار برای فرستندگان نامه پیامی ارسال کرده و به پرسشهای متداول آنها پاسخ بدهد.

اطلاعات دقیق این پروتکل در RFC-1225 تشریح شده است.

## ۶) مراجع این فصل

مجموعهٔ مراجع زیر می‌توانند برای دست آوردن جزییات دقیق و تحقیق جامع در مورد مفاهیم معرفی شده در این فصل مفید واقع شوند.


| | |
|---|---|
| **RFC 1341** | **"MIME (Multipurpose Internet Mail Extensions) Mechanisms for Specifying and Describing the Format of Internet Message Bodies," Borenstein, N.; Freed, N.; 1992** |
| **RFC 1143** | **"Q Method of Implementing Telnet Option Negotiation," Bernstein, D.J.; 1990** |
| **RFC 1090** | **"SMTP on X.25," Ullmann, R.; 1989** |
| **RFC 1056** | **"PCMAIL: A Distributed Mail System for Personal Computers," Lambert, M.L.; 1988** |
| **RFC 974** | **"Mail Routing and the Domain System," Partridge, C.; 1986** |
| **RFC 822** | **"Standard for the Format of ARPA Internet Text Messages," Crocker, D.; 1982** |
| **RFC 821** | **"Simple Mail Transfer Protocol," Postel, J.B.; 1982** |

## ۱) مقدمه

برای سالیان متوالی سیستم پست الکترونیکی عمومی ترین ابزار کاربردی بر روی شبکه اینترنت به شمار می رفت تا آنکه در اوائل دهه نود میلادی تور جهان گستر[۱] متولد شد. گزافه نیست اگر تور جهان گستر (یا اختصاراً وب) را وسیعترین و پررونق‌ترین ابزار روی شبکه اینترنت تلقّی کنیم؛ بگونه‌ای که در نظر کاربران معمولی ، وب و اینترنت مفهومی معادل دارد. امروزه صحبت از تور جهان گستر (وب) ، صفحهٔ وب[۲] ، ابرمتن[۳] ، آدرسهای حوزه و پروتکل انتقال ابرمتن[٤] بر سر هر زبانی شنیده می‌شود و بقدری تب آن همه جا را فراگرفته که گاه مهندسین شبکه نیز مفاهیم این اصطلاحات را رها کرده و به کاربردهای آن دل بسته‌اند.

در این فصل پس از معرفی ساده و مفهومی هر یک از اصطلاحات فوق به سمت تعریف پروتکلهای حاکم بر تور جهان‌گستر پیش خواهیم رفت.

### ۱-۱) مفهوم تور جهان گستر

تور جهان‌گستر یا وب یک روش معماری یا بعبارتی یک نظام برای ذخیره‌سازی و دسترسی به مستندات به هم پیوندخورده‌ای است که روی هزاران هزار ماشین در کلّ جهان پراکنده و توزیع شده اند. هر یک از این مستندات پیوندخورده که شامل متن ، صدا و تصاویر گرافیکی و تصاویر متحرک می‌شود ، می‌تواند به یک سند دیگر در محلی متفاوت در جهان اشاره نماید. این روش برای دسترسی به اطلاعات توزیع شده در عرض چند سال چنان جاذبه‌ای ایجاد کرد که کل دنیا تحت تاثیر آن قرار گرفت ، بگونه‌ای که مردم عادی را نیز واداشت که به این پدیده نوظهور بعنوان ابزاری برای زندگی راحتتر نگاه کنند.

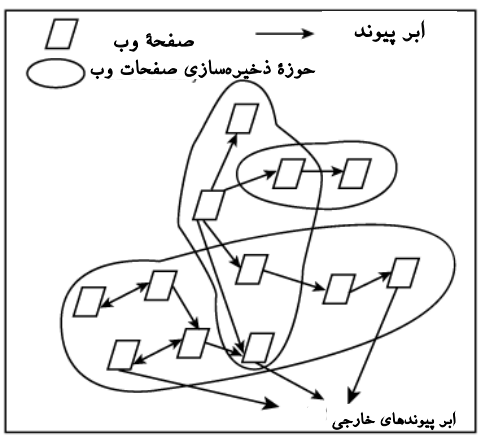
بزرگترین حسن وب، سادگی استفاده از آن است؛ کاربر با وارد کردن آدرس یک سایت که یک صفحه وب در آنجا ذخیره شده ، آنرا بر روی کامپیوترِ خود منتقل می‌کند ، آنرا نگاه و مطالعه کرده و اگر تمایل به دریافت اطلاعات بیشتری در مورد آیتمهائی که پررنگ[۵] هستند ، داشته باشد با ماوس خود روی آن کلیک می‌کند.

---

۱ Web
۲ Web page
۳ Hypertext
٤ Hype Text Transfer Protocol/URL : HTTP
۵ Highlight

(آیتمهای پر رنگ نقطهٔ پیوندِ[۱] صفحهٔ فعلی با یک صفحهٔ وبِ دیگر به حساب می‌آیند) صفحهٔ جدید هم مثل صفحهٔ قبلی روی کامپیوتر او بارَ شده و همین روند ادامه می‌یابد. ممکن است کاربر صفحه‌ای را از یک سایت در آمریکا بار کرده باشد. پس از مطالعهٔ آن به موضوعی علاقمند می‌شود که بصورت پر رنگ علامتگذاری شده است؛ روی آن کلیک می‌کند ، بدون آنکه بداند صفحه وب مورد نظر او روی سایتی مثلاً در اروپا قرار دارد. در حقیقت در صفحهٔ اوّلی که او نگاه می‌کند در ذیل عنوان پر رنگ یک آدرس دیگر (که از این بعد آنرا URL می‌نامیم) ، وجود داشته که ارتباط بین دو صفحه را برقرار کرده است. در شکل (۱-۱۰) شمای کلی ارتباطات بین صفحات وب به تصویر کشیده شده است.

حال شما تصور کنید اکنون که (ابتدای سال ۱۳۸۰) طبق آمار غیر رسمی حدود یک و نیم میلیارد صفحه وب در کلّ دنیا ایجاد و ذخیره شده و بین هر یک از آنها دهها پیوند برقرار است ، آیا این صفحات و ارتباطات آنها بصورت یک تار عنکبوت که روی کلّ جهان کشیده شده است مجسّم نمی‌شود؟ اصطلاح وِب (تار عنکبوت) از همین تصور نشأت گرفته است. به هر حال ما بدنبال وجه تسمیه و جنبه‌های ادبی قضیه نیستیم و باید مفاهیم فنی آنرا بررسی نمائیم.




شکل (۱-۱۰) شمای کلی ارتباطات بین صفحات وب

[۱] Link

شاید وب جالبترین جنبه استفاده از مفهوم برنامه‌های سرویس‌دهنده / مشتری در شبکه اینترنت باشد؛ بدینگونه که برنامهٔ سمت مشتری تقاضایی را برای دریافت یک صفحه وب یا یک فایل ، به سمت سرویس‌دهنده ارسال می‌کند. برنامهٔ سرویس‌دهنده در صورت امکان این تقاضا را اجابت کرده و داده‌های لازم را ارسال می‌نماید.

بد نیست قبل از بررسی برنامهٔ سرویس دهندهٔ وب[1] و برنامهٔ مشتری (مرورگر) تعریف URL را ارائه نمائیم.

### ۲-۱) مفهوم URL[2]

اشاره کردیم که یک صفحه وب می‌تواند شامل یک پیوند به صفحه وب دیگر درهر نقطه از دنیا باشد. هر پیوند در حقیقت آدرس دقیق یک فایل یا صفحهٔ وب محسوب می‌شود.

با توجه به ناهمگون بودن سیستمهای عامل و کامپیوترها در دنیا ، بعنوان یک نیاز بنیادی باید بتوان فایلها وپروسه‌ها را از لحاظ سبک نامگذاری و محل ذخیرهٔ آنها بر روی یک ماشین، هماهنگ و استاندارد کرد. یعنی باید یک روش آدرس دهی برای فایلها انتخاب شود به گونه ای که بتواند به سه سوال زیر برای هر فایلی در دنیا پاسخ بدهد:

- نام فایل چیست؟
- محل دقیق ذخیره شده فایل کجاست؟ (یعنی روی چه ماشینی و چه زیر شاخه‌ای قرار دارد؟)
- به چه روشی باید به فایل دسترسی داشت و طبق چه قاعده‌ای می‌توان آن فایل را انتقال داد؟

یک روش آدرس‌دهیِ استاندارد باید بتواند هر فایلی را بطور منحصر به فرد و یکتا در دنیا مشخص کند بنحوی که هیچ ابهامی در آن وجود نداشته باشد. URL روشی برای آدرس دهی منابع و فایلها در دنیاست که به هر سه سوال فوق بدون هیچ ابهامی پاسخ می‌دهد. بنابراین در ذهن خود URL را بعنوان یک آدرس استاندارد یا یک سَبکِ آدرس‌دهی تجسّم کنید.

---

[1] Web Server
[2] Uniform Resource Locator

آدرس URL که امروزه شما حتی روی پوسته یک بیسکوئیت آنرا می‌بینید شامل سه قسمت اساسی است :

- قسمت پروتکل که به آن قاعده انتقال[۱] هم گفته می‌شود.
- نام ماشینی که فایل روی آن قرار دارد. (نام حوزهٔ ماشین یا آدرس IP آن)
- شاخه و نام فایل

بعنوان مثال به آدرس URL زیر دقت کنید:

**http://www.ibm.com/researches/piii/paper.htm**

آدرس فوق از سه قسمت تشکیل شده است:

- فیلد پروتکل که در این مثال **http** است. این فیلد به برنامهٔ سمت مشتری که "مرورگر[۲]" نام دارد تفهیم می‌کند که برای بارکردن و انتقال فایل باید از کدام پروتکل و مجموعهٔ فرامین استفاده نماید. درحقیقت این فیلد زبان صحبت کردن مرورگر با سرویس‌دهنده را تعیین می‌کند. فیلد پروتکل با علامت //: از بقیه آدرس جدا می‌شود.
- نام حوزهٔ ماشینی که فایل مربوطه روی آن ذخیره شده است. در مثال فوق نام حوزهٔ ماشین www.ibm.com است. نام حوزه بین **//:** تا اوّلین **/** بعدی قرار می‌گیرد.
- نام شاخه و نام فایل: در این قسمت که دقیقاً بعد از نام حوزه شروع می‌شود و تا انتها ادامه می‌یابد ، نامِ شاخه ای که فایل درون آن قرار گرفته و همچنین نام فایل درج می‌شود. در مثال فوق نام فایل paper.htm است که بر روی شاخه /researches/piii/ قرار دارد.

دقت کنید که بر خلاف سیستم عامل DOS و MS-Windows شاخه‌ها با علامت \ از هم جدا نمی‌شوند بلکه مشابه سیستم عامل یونیکس با علامت / از یکدیگر تفکیک می‌شوند. بنابراین بصورت عمومی آدرس URL بصورت زیر تجزیه می‌شود:

**نام فایل/.../.../نام شاخه/ نام حوزهٔ ماشین // : نام پروتکل**

دو نکته بسیار مهم در مورد آدرس URL وجود دارد:

---

[۱] Transfer Protocol / Schema
[۲] Browser

اوّل آنکه اگر هنگام وارد کردن آدرس نام فایل ذکر نشود ، سرویس‌دهنده بصورت پیش‌فرض نام فایل مورد نظر را index.htm در نظر می‌گیرد. یعنی دو آدرس زیر دقیقاً معادلند:

http://www.sony.com/

http://www.sony.com/index.htm

با چنین فرضی آدرسها کوتاه می‌شوند. معمولاً به اولین صفحه ای که کاربر از یک سایت می‌بیند و با نام index.htm ذخیره شده است ، **"صفحه خانگی"**[1] گفته می‌شود. طراح صفحهٔ وب ، صفحه اصلی یک سایت را بگونه‌ای طراحی می‌کند که برای کاربر نقش یک کاتالوگِ فهرست‌دار را بازی کند؛ نام این فایل را index.htm می‌گذارد تا کاربر مجبور نباشد نام دقیق آنرا در آدرس URL وارد نماید و آدرس کوتاه باشد. پس از انجام مراحل بار شدن صفحهٔ خانگی با نام index.htm ، کاربر این صفحه را مطالعه کرده روی عناوین دلخواهش کلیک می‌کند. در بطن هر عنوان پررنگ می‌تواند یک URL مفصل و بزرگ باشد که اگر چه کاربر آنرا نمی‌بیند ولی وقتی روی آن کلیک می‌کند یک صفحهٔ دیگر بار می‌شود؛ بنابراین کاربر مجبور نیست تمام آدرسهای URL را بداند و فقط دانستن آدرس صفحهٔ خانگی برای دسترسی به تمام مستندات آن سایت کفایت می‌کند. این آدرسها توسط طراح صفحهٔ وب در ذیل هر عنوان ، تنظیم شده و یک پیوند به صفحهٔ دیگر محسوب می‌شود.

نکته دوم آنست که نام حوزه در آدرس URL می‌تواند با آدرس IP جایگزین شود؛ یعنی اگر آدرس 203.141.207.14 معادل با نام حوزهٔ www.sony.com باشد ، دو آدرس URL زیر معادلند:

http://www.sony.com/products.htm

http://203.141.207.14/products.htm

استفاده از آدرسهای IP بجای نام حوزه مرسوم نیست زیرا به راحتی به خاطر سپرده نمی‌شود ولی در مجموع امکان‌پذیر است و چون نیازی به ترجمهٔ نام حوزه وجود ندارد ، طبیعتاً اندکی سریعتر است.

توصیه مؤکّد آن است که سعی کنید تمامی حروف آدرس URL را حروف کوچک وارد کنید مگرآنکه صریحاً بصورت حروف بزرگ معرفی شده باشند.

---

[1] Home Page

دقت کنید که در آدرس‌دهی به سبک URL ، به غیر از نام و محل دقیق فایل ، روش دریافت فایل ذخیره شده نیز به مرورگر تفهیم می‌شود. بعنوان مثال اگر آدرس زیر را داشته باشید فایلی را بر روی کامپیوتر خودتان مشخص می‌کند:

file://utils/av/readme.htm

برخی از پروتکلها که در آدرس URL استفاده می‌شود در جدول (۲-۱۰) معرفی شده است که در آینده مهمترین آنها را تشریح خواهیم کرد.

| نام پروتکل | مورد استفاده | مثال |
|---|---|---|
| http | انتقال صفحات ابرمتن | http://www.ibm.com/ |
| ftp | انتقال فایل | ftp://ftp.cs.vu.nl.minx/readm |
| file | فایلهای محلی | file://user/prog.c |
| news | گروههای خبری | news://comp.os.minix |
| gopher | گوفر | gopher://gopher.tc.mn.edu |
| telnet | تِل نت | telenet://www.w3.org:80 |

جدول (۲-۱۰) نام پروتکلهای استاندارد در آدرس URL

## ۲) مقدمه ای بر سیستم وب

کلاً از دیدگاه مسائل فنی ، سیستمِ وِب در دو بخش سازماندهی می‌شود:

- برنامهٔ سمت سرویس‌دهندهٔ وب و برنامهٔ سمت مشتری وب[۱]
- پایگاه اطلاعاتیِ توزیع شده از صفحات ابرمتن و فایلهای داده مثل صدا ، تصویر و ...

صفحه وب چیزی نیست مگر یک فایل متنی بسیار ساده که با زبان علامت‌گذاری HTML تدوین می‌شوند و در بخشهای آتی روش ایجاد آنرا بوسیله یک ویرایشگر متنی (مثل Notepad) توضیح خواهیم داد.

[۱] Web Server/Web Browser

کاری که "مرورگر" بعنوان یک سرویس گیرندهٔ وب انجام می‌دهد آنست که تقاضای دریافت یکی از این صفحات یا فایلها را در قالب قراردادی استاندارد (به نام پروتکل HTTP) به سمت سرویس‌دهنده ارسال می‌کند. در سمت مقابل سرویس‌دهندهٔ وب این تقاضا را پردازش کرده و در صورت امکان ، فایل مورد نظر را برای مرورگر ارسال می‌کند. مرورگر پس از دریافتِ فایل ابرمتنی ، آنرا تفسیر کرده و بصورت صفحه آرایی شده روی خروجی نشان می‌دهد.

اگر فایل ابرمتنی در جائی به فایل صدا یا تصویر پیوند خورده باشد ، آنها نیز توسط مرورگر تقاضا شده و پس از دریافت در جای خود قرار می‌گیرند.

بنابراین سرویس‌دهندهٔ وب را باید یک برنامهٔ سوکت در نظر گرفت که فرامین مشتری‌ها را دریافت ، پردازش و در صورت امکان اجرا می‌کند. برنامه سمت مشتری نیز برنامه سوکتی است که تقاضاها را در قالب فرامین استاندارد ، برای سرویس‌دهندهٔ وب ارسال می‌کند؛ در ضمن وظیفه تفسیر و نمایش داده‌های دریافتی را نیز بر عهده دارد.

در ذهن خود دو مفهوم کاملاً مجزای زیر را از هم تفکیک کنید:

- **"پروتکل انتقال صفحات ابرمتن[۱]"** : این پروتکل زبان یا قراردادی برای صحبت کردن مشتری با سرویس‌دهنده وب (HTTP) است.
- **"زبان نشانه گذاری صفحات ابرمتن[۲]"** زبانی برای قالب‌بندی و صفحه‌آرائی اطلاعات متنی (HTML) است.

ممکن است خواننده نکته گیر به این قضیه اعتراض کند که سرویس دهندهٔ وب را با سرویس‌دهنده HTTP همسان گرفتیم در صورتیکه سرویس‌دهنده وب فراتر از سرویس دهندهٔ HTTP است. برای توجیه قضیه خاطر نشان می‌کنیم که فعلاً کاربردی‌ترین بخش از سرویس دهندهٔ وب همان سرویس دهندهٔ HTTP است.

معرفیِ زبان نشانه گذاری صفحات ابرمتنی (HTML) را به بخشهای بعد موکول کرده و فعلاً برنامه‌های سمت سرویس‌دهنده و سمت مشتری وب (مرورگر) را مورد بررسی قرار می‌دهیم تا پیوستگی بحث ما با موضوع برنامه نویسی سوکت از دست نرود.

---

[۱] Hyper Text Transfer Protocol
[۲] Hyper Text Markup Language

### ۱-۲) برنامه سمت سرویس‌دهنده وب

در سمت سرویس دهندهٔ وب ، پروسه‌ای وجود دارد که دائماً به پورت شماره 80 گوش می‌دهد و منتظر تقاضای برقراری ارتباط توسط مشتریان (برنامهٔ مرورگر) می‌باشد. دقت کنید که این برنامه از سوکتهای نوع استریم استفاده می‌کند و ارتباط از نوع TCP است؛ فرامین و داده‌هایی که بین سرویس‌دهنده و مرورگر مبادله می‌شوند تماماً متنی هستند.

بعد از آنکه ارتباط TCP بین برنامهٔ سرویس‌دهنده و برنامه مشتری برقرار شد برنامه مشتری حق دارد یک تقاضا بفرستد و این تقاضا باید در قالب استاندارد HTTP باشد. سرویس‌دهنده این تقاضا را دریافت و پردازش می‌کند و در صورت امکان این آن را اجرا می‌کند. مراحل بار شدن یک صفحه وب در قالب یک مثال ارائه می‌شود. به روند زیر توجه نمایید:

فرض کنید کاربر، URL زیر را در خط آدرس مرورگر خود (مثلاً در برنامه IE5.0 یا Netscape) وارد کرده و کلید ↵ را فشار داده باشد:

http://www.w3.org/hypertext/www/theproject.html

مرورگر با تحلیل آدرس URL متوجه می‌شود که باید تقاضای فایلی را طبق پروتکل HTTP به سمت سرویس‌دهنده بفرستد. مراحلی که اتفاق می‌افتد به شرح زیر خواهد بود :

**الف)** مرورگر آدرس URL را تحلیل کرده و قسمتهای پروتکل، آدرس نام حوزه ، شاخه و نام فایل را از آن استخراج می‌کند.

**ب)** مرورگر تقاضای ترجمهٔ آدرس نام حوزه را به DNS ارسال می‌نماید تا آدرس IP ماشین سرویس‌دهنده به دست آید. دراین مثال مرورگر تقاضای ترجمه نام www.w3.org را به DNS ارسال می‌کند. ( معادل فراخوانی تابع gethostbyname() در برنامه نویسی سوکت)

**ج)** DNS در پاسخ ، آدرس IP معادل با نام حوزه را بر می‌گرداند. فرض کنید در این مثال DNS آدرس IP را 18.23.0.23 برگردانده است.

**د)** مرورگر یک ارتباط TCP با آدرس 18.23.0.3 و پورت 80 برقرار می‌کند. ( معادل فراخوانی تابع connect() در برنامه نویسی سوکت)

**ه)** پس از برقراری ارتباط، رشته کاراکتری زیر که مجموعاً ۳۵ بایت است به سمت سرویس‌دهنده ارسال می‌شود:

"**GET /hypertex/www/theproject.html**"

GET یکی از فرامین استاندارد HTTP است که در ادامه توضیح خواهیم داد.

**و)** سرویس‌دهنده این رشته را دریافت و پس از پردازش آن ، فایل theproject.html را از شاخهٔ /hypertext/www/ استخراج کرده و برای مرورگر ارسال می‌نماید.

**ز)** مرورگر فایل را دریافت کرده و پس از خاتمهٔ دریافت ارتباط TCP را قطع می‌کند.

**ح)** مرورگر فایل ابرمتنی را تفسیر کرده و آنرا روی خروجی نمایش می‌دهد.

**ط)** اگر فایل ابرمتنی در جائی دارای تصویر یا صدا باشد به ازای تک‌تک آنها مراحل الف تا ح را تکرار نموده و آنها را به ترتیب دریافت می‌کند. دقّت کنید که درون فایلهای ابرمتنی داده‌های فایلهای صدا یا تصویر وجود ندارد بلکه فقط نام و محل قرار گرفتن فایل تصویر یا صدا درون آن درج شده است. ( این موضوع بر خلاف سیستم پست الکترونیکی است که داده‌های صدا و تصویر در ادامهٔ متن نامه قرار می‌گیرد.)

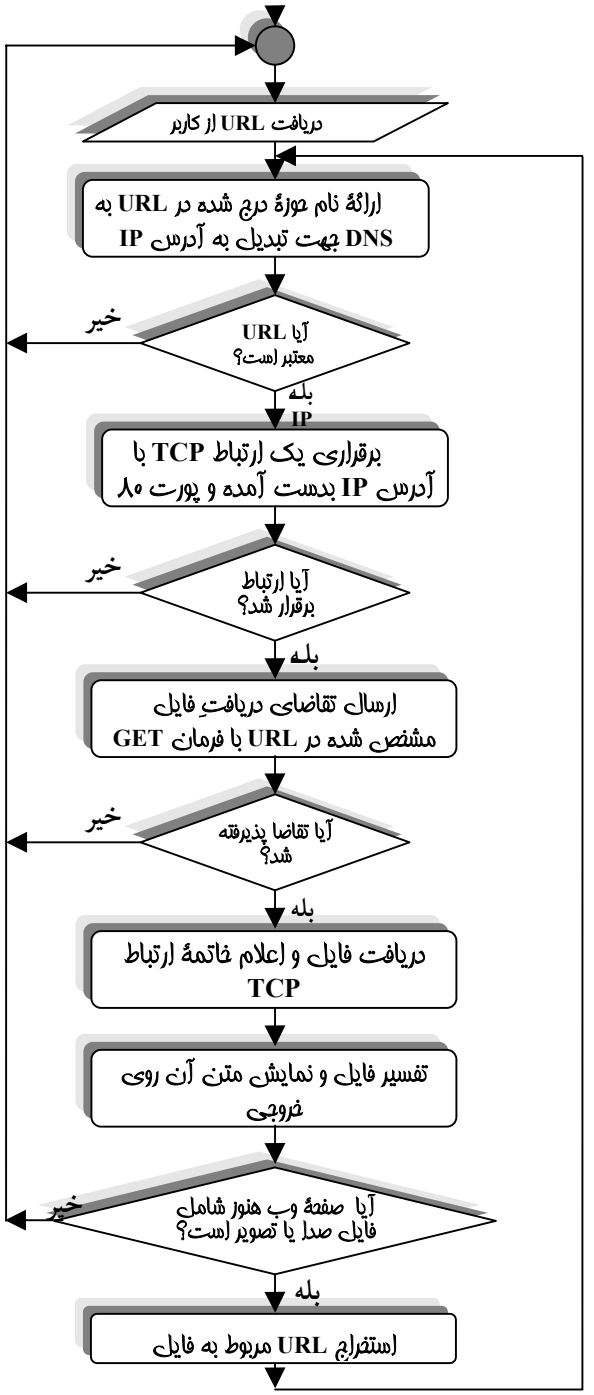
در شکل (۳-۱۰) فلوچارتِ عملیاتی که مرورگر برای دریافت یک صفحهٔ وب و ملحقات آن انجام می‌دهد ، به تصویر کشیده شده است. با تحلیل الگوریتم فوق ، روند عملیات برنامهٔ سمت سرویس‌دهندهٔ HTTP مشخص خواهد شد.

سرویس‌دهنده بایستی فرامین دریافتی که بصورت متنی و بر طبق پروتکل HTTP هستند را دریافت ، پردازش ، احراز هویت و اجرا نماید. در ادامه ، پروتکل انتقال صفحات ابرمتن را بررسی می‌نمائیم.

## ۳) پروتکل انتقال ابرمتن : HTTP

پروتکل انتقال ابرمتن مجموعه ای از فرامین استانداردی است که از سمت مشتری به سمت سرویس‌دهنده ارسال می‌شود. در حقیقت این پروتکل طریقهٔ صحبت کردن سرویس‌دهنده و مشتری را تبیین کرده است.

بدون مقدمه سراغ مجموعهٔ فرامین در جدول (٤-۱۰) می‌رویم؛ این فرامین در RFC-1945 **"متود"** نامیده شده است.

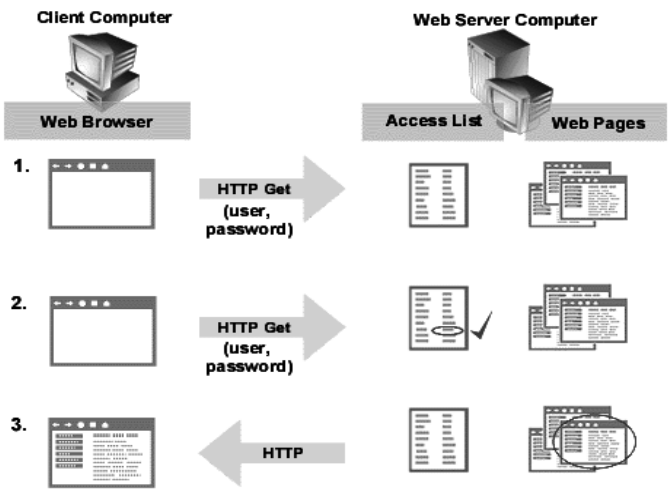



شکل (۳-۱۰) فلوچارت عملیات مرورگر برای دریافت یک صفحهٔ وب

| نام فرمان | توضیـــح |
|---|---|
| GET | تقاضا برای دریافت یک صفحه وب از سرویس‌دهنده |
| HEAD | تقاضا برای دریافت سرآیند[1] یک صفحه وب |
| PUT | تقاضا برای ذخیره کردن یک صفحه وب روی یک سرویس‌دهنده |
| POST | تقاضا برای ضمیمه کردن اطلاعاتی به یک منبع (مثل فایل یا صفحه وب) |
| DELETE | تقاضا برای حذف یک صفحه وب |
| LINK | تقاضای برقراری پیوند بین دو منبع موجود |
| UNLINK | تقاضای خاتمه پیوند دو منبع موجود |

جدول (٤-١٠) فرامین تعریف شده در پروتکل HTTP

- **متود GET** : مرورگر با ارسال این متود از سرویس‌دهنده تقاضا می‌کند که یک صفحه وب یا یک فایل دودویی مثل فایل تصویر یا صدا برایش ارسال شود. دقت کنید که فایلهای صدا یا تصویر درقالب استاندارد MIME ارسال خواهد شد؛ یعنی حتی فایلهای دودویی نیز بایستی طبق یکی از روشهای تبدیل در استاندارد MIME به حالت متنی تغییر شکل داده شود و سپس ارسال گردد. در جلوی متود GET نام فایل درخواستی درج می‌شود. در شکل (۵-۱۰) محاورهٔ مرورگر و سرویس‌دهندهٔ HTTP برای دریافت یک صفحهٔ وب را نشان می‌دهد.




شکل (۵-۱۰) محاورهٔ مرورگر و سرویس‌دهندهٔ HTTP برای دریافت یک صفحه

[1] Header

اگر بعد از نام فایل درخواستی گزینهٔ "If-Modified-Since:" که در ادامهٔ آن تاریخ و زمان درج شده ، اضافه شود سرویس‌دهنده را وادار می‌کند که فایل درخواستی را به شرطی ارسال نماید که آن فایل بعد از تاریخ و زمان مشخص شده ، تغییر کرده باشد. با این گزینه مرورگر می‌تواند درصورتی که قبلاً فایل را دریافت کرده و از آن تاریخ به بعد تغییری نداشته ، آنرا از روی ماشین محلی بار کند تا سرعت نمایش صفحات بیشتر شده و اطلاعات بیهوده مبادله نشود. مثال:

GET /hypertext/www/theproject.html HTTP/1.0 If-Modified- Since: Sat 29 Oct 1999

- **متود HEAD** : این متود از سرویس‌دهنده تقاضا می‌کند که فقط سرآیند صفحه وبی را که نام آن در جلوی متود درج شده ، ارسال نماید. این متود چند کاربرد دارد :

اوّل آنکه مشخصات صفحه وب ، شامل تاریخ آخرین تغییر ، عنوان صفحه ، نام تدوین کننده و صاحب اصلی آن و برخی از مشخصات اختیاری که در سرآیند صفحهٔ وب درج شده ، ارسال می‌شود و این اطلاعات میتواند برای مقاصدی همانند تهیه بانک اطلاعاتی از صفحات وب و طراحی "جستجوگرهای وب[1]" مفید واقع شود.

دوم آنکه می‌توان با این متود صحیح بودن یک URL و وجود یک صفحه وب را ارزیابی کرد.

- **متود PUT** : این متود عکس عمل GET است یعنی مرورگر تقاضا می‌کند که یک صفحه وب را بر روی ماشین سرویس‌دهنده ذخیره نماید. این متود را سرویس‌دهنده‌هائی حمایت می‌کنند که بخواهند صفحات برخی از کاربران را دریافت کرده ضمن ذخیره ، آنها را در اختیار دیگران قرار بدهند. فایلهایی که با این متود ارسال می‌شوند باید طبق استاندارد MIME سازماندهی‌شده باشند.

- **متود POST** : این متود از سرویس‌دهنده تقاضا می‌کند که داده‌هائی را به یک منبع موجود (مثل یک صفحه وب یا یک فایل) اضافه کند. این متود برای ایجاد "صفحات

---

[1] Search Engine

آزاد خبری" ، "تابلو اعلانات[1]" ، محیطهای نظر خواهی یا ارسال اطلاعات برای یک پروسهٔ تحت وِب همانند برنامه‌های CGI مورد استفاده قرار می‌گیرد.

- **متود DELETE** : این متود از سرویس‌دهنده تقاضا می‌کند که یک صفحهٔ وب با نام مشخص را از روی ماشین (ماشین سرویس‌دهنده ) حذف نماید.

دقت شود که بسیاری از سرویس‌دهنده‌ها به دلایل امنیتی از متودهای PUT ، POST و DELETE پشتیبانی نمی‌کنند.

- **متودهای LINK و UNLINK** : این دو متود اجازه می‌دهند که بین دو صفحه وب ( یا دو منبع) ارتباط و پیوند برقرار شده یا پیوند قبلی خاتمه داده شود؛ برای توضیح بیشتر در مورد این دو متود به RFC-1945 مراجعه نمائید.

وقتی تقاضا به سمت سرویس‌دهنده ارسال می‌شود چه پذیرفته شود و چه پذیرفته نشود ، پاسخی متنی دریافت می‌دارد که معمولاً بصورت زیر است :

| رشتهٔ متنی | شماره وضعیت | شمارهٔ نسخه‌پروتکل |
|---|---|---|

مثال :

HTTP/1.0 200 OK

یا HTTP/1.0 304 Not Modified

- **HTTP/1.0** : نسخه پروتکل را مشخص می‌کند.
- **شمارهٔ وضعیت** : شماره ای است سه رقمی که وضعیت اجرای فرمان ارسالی را مشخص می‌نماید. این شمارهٔ سه رقمی بر اساس رقم صدگان به پنج دسته تقسیم می‌شود:
  - 1xx : اطلاعاتی ( پاسخی جهت آگاهی بیشتر مشتری )
  - 2xx : عمل درخواستی موفقیت آمیز اجرا شده است.
  - 3xx : URL مورد تقاضا تغییر آدرس داشته است.
  - 4xx : در تقاضای ارسال شده از طرف مشتری خطایی وجود دارد.

[1] Bulletin Board

• 5xx : در سرویس‌دهنده خطایی داخلی رخ داده است.

در صورتی که که رقم صدگان ۳ ، ٤ یا ٥ باشد وضعیت فرمان ارسالی ناموفق بوده است.

♦ **رشتهٔ متنی** : متن کوتاهی که وضعیت اجرای فرمان را به زبان طبیعی توصیف می‌کند.

برخی از شماره‌های وضعیت در جدول جدول (٦-١٠) فهرست شده اند.

پس از ارسال اوّلین خطّ پاسخ توسط سرویس‌دهنده در خطوط بعدی نیز یک یا چند سطر اطلاعات اضافی برای مرورگر ارسال می‌شود که می‌تواند برای تفسیر داده‌ها بسیار مهم باشد. قالب برخی از این اطلاعات در جدول (٧-١٠) مشخص شده است. پس از این خطوط که **سرآیند** نامیده می‌شوند ، یک سطر خالی مرز داده‌های فایل را از سرآیند مشخص کرده و در ادامه داده‌های فایل ارسال می‌شوند.

| شماره | توصیف متنی | توضیح |
|---|---|---|
| 200 | OK | فرمان پذیرفته شد. |
| 204 | No content | چیزی وجود ندارد که به مشتری برگشت داده شود. |
| 301 | Moved permanently | URL درخواست شده برای همیشه تغییر کرده است. |
| 302 | Moved temporarily | URL درخواست شده موقتاً تغییر کرده است. |
| 304 | Not modified | عدم تغییر سند درخواستی پس از تاریخ اعلام شده |
| 400 | Bad request | فرمان صادره شناخته نشد. |
| 401 | Unauthorized | فرمان ارسالی به احراز هویت کاربر نیاز دارد. |
| 403 | Not permitted | فرمان صادره غیر مجاز است و اجرا نشد. |
| 404 | Not found | صفحه وب یا فایل درخواستی پیدا نشد. |
| 500 | Server error | بروز اشکال داخلی در سرویس‌دهنده |
| 502 | Bad gateway | برنامهٔ CGI پاسخ نمیدهد یا وجود ندارد. |
| 503 | Service Unavailable | سرویس‌دهنده به دلیل خاصی فعلاً سرویس نمی‌دهد. |

جدول (٦-١٠) معرفی مجموعه‌ای از پاسخهای سرویس دهندهٔ HTTP

| نوع اطلاعات | قالب اطلاعات |
|---|---|
| محل دقیق و مطلق منبع یا فایل درخواستی | **Location:** *absoluteURL* |
| نوع و نام سرویس دهندهٔ وب | **Server:** *product* |
| تاریخ و زمان آخرین تغییر منبع یا فایل | **Last-Modified:** *HTTP-date* |
| تاریخ و زمان منقضی شدن منبع یا فایل | **Expires:** *HTTP-date* |
| روش کد گذاری بدنهٔ پیام (استاندارد MIME ) | **Content-Encoding:** *encoding* |
| طول داده‌ها بر حسب بایت (استاندارد MIME ) | **Content-Length:** *length* |
| نوع سند ارسالی (استاندارد MIME ) | **Content-Type:** *type* |

جدول (۷-۱۰) معرفی مجموعه ای از پاسخهای سرویس دهندهٔ HTTP

بگونه ای که اشاره شد ، در پروتکل HTTP همانند پروتکل SMTP ، تمام اطلاعات در قالب کاراکترهای اَسکی مبادله می‌شوند و هر گاه فایل دودویی ( مثل صدا و تصویر ) مبادله شود ، باید طبق استاندارد MIME ( از طریق یکی از دو روش ASCII Armor یا qouted-printable–encoding ) به حالت متنی تبدیل شود.

برای آشنائی با چگونگی ارسال داده‌ها از طرف سرویس‌دهنده ، در شکل (۸-۱۰) پاسخی را که سرویس‌دهندهٔ HTTP به یک تقاضای دریافت صفحه وب برگردانده ، می‌بینیم. این تقاضا بصورت زیر فرستاده شده است :

**GET /example/hello.htm HTTP/1.0**

```
HTTP/1.0 200 OK
Server: Microsoft-IIS/4.0
Date: Friday, 9-OCT-1998 10:15:00 GMT
Last-Modified: Monday, 5-OCT-1998 21:35:30 GMT
Content-Type: text/html
Content-Length: 98

<html>
<head>
<title>Example of HTTP Response</title>
</head>
<body>
HELLO WORLD!
</body>
</html>
```

شکل (۸-۱۰) ساختار اطلاعات ارسالی از سرویس‌دهنده در پاسخ به تقاضای یک صفحهٔ وب

در شکل (۹-۱۰) مثالی دیگر از چگونگی مبادلهٔ فرمان و پاسخ بین برنامهٔ سرویس دهندهٔ HTTP و برنامهٔ مشتری برای دریافت یک صفحهٔ وب نشان داده شده است. بگونه ای که در این شکل مشاهده می‌کنید ابتدا سعی شده با ماشینی با نام حوزه www.csc.com و پورت ۸۰ ارتباط TCP برقرار شود؛ برای اینکار از برنامهٔ Telnet استفاده شده است. برای برقراری ارتباط ، برنامهٔ Telnet ابتدا سعی کرده که نام حوزه را به آدرس IP ترجمه کند؛ سپس اقدام به برقراری ارتباط کرده است. در ادامه کاربر توسط متود GET تقاضای یک صفحهٔ وب با نام toc.html را صادر کرده است. دقت شود که پس از ارسال رشته فوق در قالب یک خط مجزا ، باید یک خط خالی دیگر نیز ارسال شود. (یعنی دو عدد کاراکتر Carriage Return متوالی)

سرویس‌دهنده پس از دریافت فرمان فوق ابتدا پنج سطر حاوی اطلاعات متنی را بعنوان سرآیند آورده است و سپس با یک خط خالی قسمت سرآیند را خاتمه داده و در ادامه داده‌های فایل درخواستی ارسال را ارسال کرده است. معنای خطوط سرآیند به شرح ذیل می‌باشد:

- **HTTP/1.0 200 Document follows :** تقاضای مرورگر پذیرفته شده و صفحهٔ درخواستی در ادامه ضمیمه می‌باشد.
- **MIME-Version: 1.0 :** روش سازماندهی اطلاعات مبتنی بر استاندارد MIME نسخهٔ ۱ می‌باشد.
- **Server: CERN/3.0 :** نام برنامهٔ سرویس‌دهنده CERN و نسخهٔ آن ۱ است.
- **Content-Type: text/html :** محتوای داده‌ها متنی و از نوع HTML می‌باشد. این گزینه در استاندارد MIME تشریح شد.
- **Content-Length: 8247 :** داده‌های ارسال شده مجموعاً ۸۲٤۷ بایت می‌باشد.
- یک سطر خالی انتهای بخش سرآیند را مشخص می‌کند.

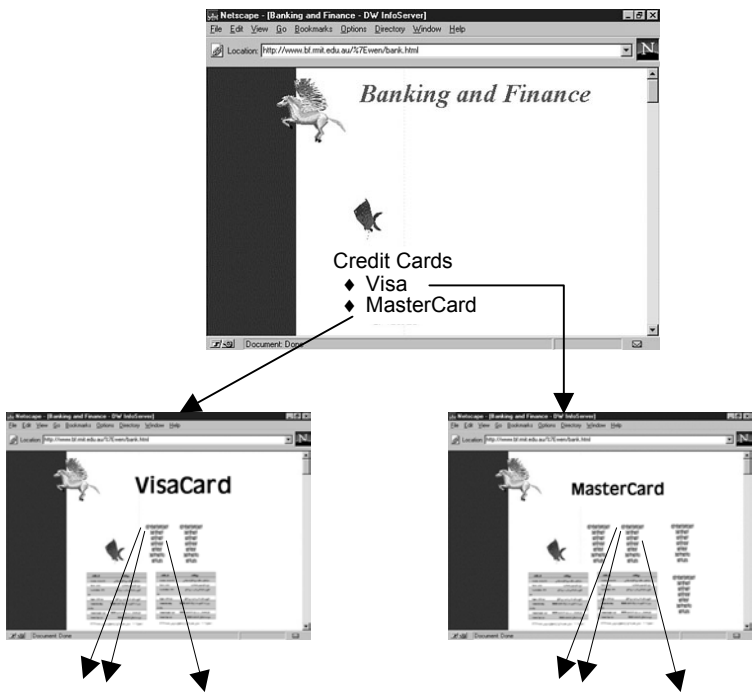
پس از معرفی **"زبان نشانه گذاری صفحات ابرمتن"** بازهم به سراغ پروتکل HTTP خواهیم آمد.

```
C: telnet www.csc.com 80
T: Trying 18.23.0.23 …
T: Connected to www.csc.com.
T: Escape character is '^]'.
C: GET /client-server-computing/toc.html HTTP/1.0
C:
S: HTTP/1.0 200 Document follows
S: MIME-Version: 1.0
S: Server: CERN/3.0
S: Content-Type: text/html
S: Content-Length: 8247
S:
S: <html>
S: <head>
S: <meta http-equiv="Content-Type"
S: content="text/html; charset=windows-1256">
S: <meta name="GENERATOR" content="Microsoft FrontPage Express 2.0">
S: <title>Client/Server Computing Table of Contents</title>
S: </head>

S: <body bgcolor="#FFFFFF" text="#000000" link="#0000FF" Vlink="#FF0000">
S:
S: <h1 align="center"><font color="#FF0000">Client/Server Computing </font>
S: <font color="#FF0000" size="4">Second Edition</font></h1>
S:
S: <ul>
S:    <li><ul>
S:          <li><a href="cscfm.htm#I8"><font size="3">Foreword</font></a></li>
S:          <li><a href="cscfm.htm#I9"><font size="3">Preface</font></a></li>
S:          <li><a href="cscfm.htm#I10"><font size="3">Acknowledgments</font></a></li>
S:          <li><a href="cscfm.htm#I12"><font size="3">Introduction</font></a></li>
S:    </ul> </li>
S:  <li><a href="cs01.htm#I1"><font size="3">— 1 —The  Business Opportunity</font></a></li>
S:  <li><a href="cs02.htm"><font size="3">— 2 —Advantages of  Client/Server Computing</font></a></li>
S:  <li><a href="cs03.htm"><font size="3">— 3 —Components of   Client/Server Applications—The Client</font></a></li>
S:  <li><a href="cs04.htm"><font size="3">— 4 —Components of  Client/Server Applications—The Server</font></a></li>
S:  <li><a href="cs05.htm"><font size="3">— 5 —Components of   Client/Server Applications—Connectivity</font></a></li>
S:  <li><a href="cs06.htm"><font size="3">— 6 —Client/Server  Systems Development—Software</font></a></li>
S:  <li><a href="cs07.htm"><font size="3">— 7 —Client/Server  Systems Development—Hardware</font></a></li>
S:  <li><a href="cs08.htm"><font size="3">— 8 —Client/Server  Systems Development—Service and Support</font></a></li>
S:  <li><a href="csc09.htm"><font size="3">— 9 —Client/Server  Systems Development—Training</font></a></li>
S:  <li><a href="csc10.htm"><font size="3">— 10 —The Future  of Client/Server Computing</font></a></li>
S:  <li><a href="cscxa.htm"><font size="3">— Appendix A —Case   Studies</font></a></li>
S:  <li><a href="cscxb.htm"><font size="3">— Appendix B —Apple/IBM  Joint Venture</font></a></li>
S:  <li><a href="cscxc.htm"><font size="3">— Appendix C —Electronic  Document Management standards</font></a></li>
S: </ul>
S: </body>
S: </html>
```

شکل (۹-۱۰) محاورهٔ سرویس‌دهنده و مشتری برای دریافت یک صفحهٔ وب

### ۴) زبان نشانه گذاری ابر متن : HTML

صفحات HTML ، متون غنی شده‌ای هستند که اطلاعات موجود در یک سند را بصورت صفحه‌آرایی شده و سازمان‌یافته ، در اختیار کاربر قرار می‌دهند. بزرگترین حسن این صفحات آنست که به کاربر این امکان را می‌دهند که به سادگی به صفحهٔ دیگری دسترسی پیدا کند؛ به گونه‌ای که می‌توان مجموعه‌ای از اطلاعات خام را بصورت سلسله‌مراتبی و سطح‌بندی شده در اختیار علاقمندان قرار داد. در شکل (۱۰-۱۰) یک صفحهٔ خانگی بسیار ساده از یک سایت نشان داده شده است. این صفحه دارای دو آیتم است که کاربر با کلیک کردن روی آن ، صفحهٔ دیگری را که مرتبط با موضوع دلخواهش می‌باشد ، دریافت می‌کند. آن صفحه نیز به صفحات مرتبط دیگری پیوند خورده است؛ این روند ممکن است تا چندین سطح ادامه داشته باشد.




شکل (۱۰-۱۰) یک صفحهٔ خانگی بسیار ساده از یک سایت در اینترنت

HTML (زبان نشانه‌گذاری ابرمتن) ، زبانی مانند پاسکال ، بیسیک یا C نیست بلکه روشی است که بواسطه آن می‌توان متون خالص و معمولی[۱] را صفحه‌آرایی کرده و عواملی مثل صدا ، تصویر ، پنجره ، منو و فهرستهای انتخاب را به متن اضافه کرد.

HTML حاوی فرامین صفحه‌آرایی است. این فرامین که در بطن متن اصلی قرار می‌گیرد ، "برچسب"[۲] نام دارد. برچسبهای درون متن توسط مرورگر تشخیص داده شده و پس از تفسیر ، نمایش متون را تحت تاثیر قرار می‌دهند. برچسبهای HTML با علامتهای < > از متن اصلی متمایز می‌شود و عبارتی که در بین آن قرار می‌گیرد ، توسط مرورگر از متن اصلی تشخیص داده خواهد شد. تاثیر عملی که با برچسب درون < ...> مشخص می‌شود ، با علامت < .../> لغو می‌گردد. به عنوان مثال متن سادهٔ زیر را در نظر بگیرید:

Internet Engineering

با استفاده از برچسبهای زیر می‌توان مرورگر را وادار کرد که متن را بصورت پررنگ و کج[۳] نشان بدهد:

<I><B>Internet Engineering</B></I>

مرورگر با تفسیر برچسبها ، متن را بصورت زیر نمایش می‌دهد:

***Internet Engineering***

کسی که با برچسبهای HTML آشنا باشد براحتی می‌تواند با یک ویرایشگر ساده ، متن خود را سازماندهی کند. ساختار کلی یک صفحهٔ HTML بصورت زیر است:

```
<HTML>
   <HEAD>
      <TITLE>
      عنوان سند در این ناحیه درج می‌شود.
      </TITLE>
   </HEAD>
   <BODY>
      تمام اطلاعات صفحهٔ وب در این ناحیه درج می‌شود.
   </BODY>
</HTML>
```

---

۱ Plain Text
۲ Tag
۳ Bold Italic

در ادامه با برخی از برچسبهای HTML آشنا می‌شویم. هدف از این بخش فقط آشنایی با کلیات HTML است و برای یادگیری اصولی آن باید به مراجع اصلی و کتب تفصیلی مراجعه کرد.

♦ یکی از اصلیترین برچسبها در سازماندهی متون ، برچسب پاراگراف است. مرورگر پاراگراف را بر اساس عرض پنجرهٔ نمایش ، تنظیم می‌کند و در صورت لزوم جملات را شکسته و به خطوط بعد می‌برد؛ به این کار پوشش خودکار گفته می‌شود. یک پارگراف با برچسب <P> آغاز می‌شود و با </P> خاتمه می‌یابد. به مثال زیر دقت کنید:

در اینجا خط بدلیل بزرگ بودن عرض صفحه ، شکسته نشده است.

در اینجا خط بدلیل کوچک بودن عرض صفحه شکسته شده است.

♦ برای ارائهٔ یک مطلب مهم که نظر مخاطب را به خود جلب کند از برچسب <EM> استفاده می‌شود؛ در این حالت متنی که تحت تاثیر این برچسب قرار می‌گیرد بصورت کج نشان داده می‌شود. برای تاکید بیشتر می‌توان از برچسب <STRONG> استفاده کرد تا متن را بصورت پررنگ نشان بدهد.

♦ برای آنکه یک سطر را قطع کرده و سطر بعدی از سر خط جدید آغاز شود از برچسب <BR> استفاده می‌شود. این برچسب می‌تواند عملیات نمایش متن را در سطر جدید بصورت زیر کنترل کند:

<BR CLEAR=LEFT> : خاتمهٔ سطر فعلی و شروع سطر بعدی از سمت چپ به راست

<BR CLEAR=RIGHT> : خاتمهٔ سطر فعلی و شروع سطر بعدی از سمت راست به چپ

<BR CLEAR=ALL> : خاتمهٔ سطر فعلی و شروع سطر بعدی در کل عرض پنجرهٔ نمایش

♦ **ابرپیوند**[1] : قبلاً اشاره شد که در یک صفحهٔ وب هر آیتم از متن می‌تواند به صفحهٔ دیگر اشاره کند. برای آنکه قطعه‌ای از متن بصورت "ابرپیوند" عمل کرده و کاربر با کلیک کردن روی آن صفحهٔ دیگری را دریافت نماید ، از برچسب <A HREF=.....> استفاده می‌شود. به مثال زیر دقت کنید :

<A HREF="introduc.html"> Introduction

[1] Hyperlink / Hyper Reference

این عبارت باعث می‌شود که کلمهٔ Introduction در متن بصورت پررنگ نشان داده شده و کاربر با کلیک روی آن ، مرورگر را وادار کند تا فایل introduc.html را بارگذاری[1] نموده و نمایش بدهد. در زیر یک ابرپیوند کامل را می‌بینید.

```
<A HREF="http://www.w3.org/hypertext/DataSource/www/Geo.html "> Geographical.html
```

♦ برای نمایش تصاویر گرافیکی در متن از برچسب <... IMG> استفاده می‌شود. تصاویر فشرده شدهٔ نوع GIF و JPEG قابل بارگذاری در یک صفحهٔ وب می‌باشد:

```
<IMG ALIGN = TOP SRC="filename.gif">
```

با این عبارت مرورگر تصویر مشخص شده را بر بالای خط فعلی متن قرار می‌دهد.

```
<IMG ALIGN = MIDDLE SRC="filename.gif">
<IMG ALIGN = BOTTOM SRC="filename.gif">
```

♦ **نمایش فهرستها** : در HTML سه نوع فهرست تعریف می‌شود:

• **فهرست بی‌شماره** : این نوع فهرست با برچسب <UL> و </UL> مشخص می‌شود و عناوین فهرست ، با برچسب <LI> و </LI> تفکیک می‌شود. بهترین توضیح مثال زیر است:

```
<UL>
<LI> عنوان ۱ </LI>
<LI> عنوان ۲ </LI>
<LI> عنوان ۳ </LI>
</UL>
```

نمایش این قسمت از متن بصورت زیر است :

- **عنوان ۱**
- **عنوان ۲**
- **عنوان ۳**

**فهرست شماره‌دار :** این نوع فهرست مشابه با فهرست قبلی است ، با این تفاوت که عناوین فهرست به ترتیب شماره‌گذاری می‌شود. ابتدای فهرست شماره‌دار با <OL> مشخص می‌شود و برچسب <LI> برای جداسازی عناوین از یکدیگر می‌باشد؛ پایان فهرست نیز با برچسب </OL> مشخص می‌گردد. به مثال زیر دقت کنید:

[1] Download

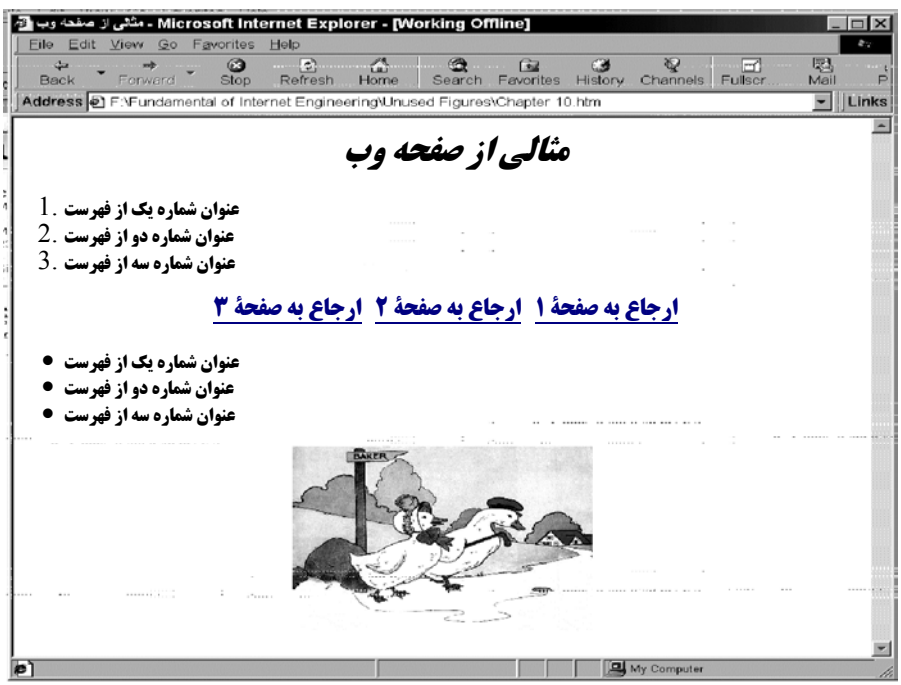
```
<OL>
<LI> عنوان ۱ </LI>
<LI> عنوان ۲ </LI>
<LI> عنوان ۳ </LI>
</OL>
```

نمایش این قسمت از متن بصورت زیر است :

1. عنوان ۱
2. عنوان ۲
3. عنوان ۳

♦ برای تعیین نوع و اندازهٔ قلم (فونت) از برچسب <font size="xxx" face="Font Name"> استفاده می‌شود که xxx اندازهٔ قلم و Font Name نام قلم مورد نظر را تعیین می‌کند.

برای آشنایی با برچسبهای فوق به شکل (۱۱-۱۰) دقت کنید. این شکل یک صفحهٔ HTML را در محیط مرورگر نشان می‌دهد؛ اصل فایل HTML آن در جدول (۱۲-۱۰) نشان داده شده است.




شکل (۱۱-۱۰) یک صفحهٔ HTML در محیط مرورگر

```
<html>

<head>
<title>اصول مهندسی اینترنت - مثالی از صفحهٔ وب</title>
</head>

<p align="center"><font size="6" face="Nazanin">
<em><strong>مثالی از صفحهٔ وب</strong></em></font></p>

<ol>
    <li>عنوان شماره یک از فهرست</li>
    <li>عنوان شماره دو از فهرست</li>
    <li>عنوان شماره سه از فهرست</li>
</ol>
</font></a><font size="5" face="Zar">
<p align="center">
<a href="http://www.malekian.com/page1.html"><strong>ارجاع به صفحهٔ ۱
</strong>
<a href="http://www.malekian.com/page1.html"><strong>ارجاع به صفحهٔ ۲
</strong>
<a href="http://www.malekian.com/page1.html"><strong>ارجاع به صفحهٔ ۳
</strong>

<ul>
    <li>عنوان شماره یک از فهرست</li>
    <li>عنوان شماره دو از فهرست</li>
    <li>عنوان شماره سه از فهرست</li>
</ul>

<p align="center">
<img src="file:///F:/ANIMALS/Bk.jpg" width="319" height="245">
</font></p>
</body>
</html>
```

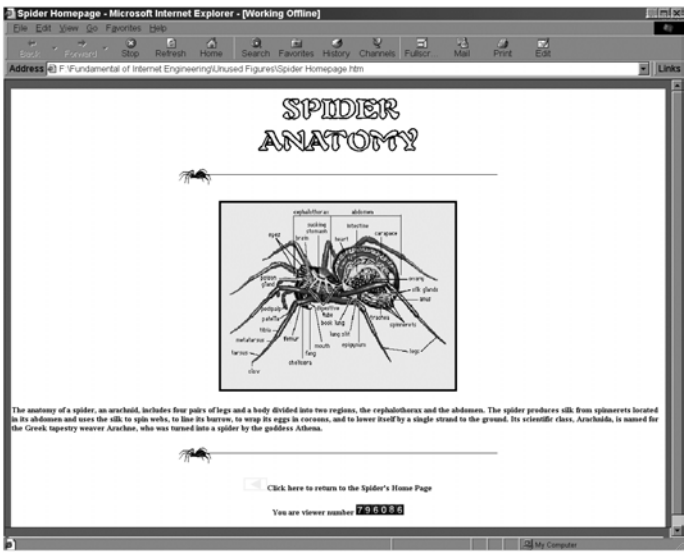
شکل (۱۲-۱۰) فایل HTML مربوط به صفحهٔ وب در شکل (۱۱-۱۰)

### ۱۴-۱) فرمهای ورود اطلاعات در HTML

آن دسته از صفحات وب که فقط ارائه کنندهٔ اطلاعات به کاربر هستند و هیچ کنش و واکنشی با کاربر ندارند ، اصطلاحاً "صفحات ایستا"[1] نامیده می‌شوند. کاربر با دریافت چنین صفحاتی ، در زمینهٔ موضوع دلخواه خود اطلاعاتی را از سرویس‌دهنده دریافت کرده و مطالعه

[1] Static Pages

می‌کند. در شکل (۱۳-۱۰) یک "صفحهٔ وب ایستا" دیده می‌شود. تمام اجزای صفحه به منظور مطالعهٔ کاربر ، آراسته شده است.




شکل (۱۳-۱۰) یک "صفحهٔ وب ایستا"

یکی از بزرگترین مشخصه‌های وب آنست که کاربر می‌تواند با یک صفحهٔ وب در تراکنش باشد؛ بدین معنا که کاربر قادر است در صفحهٔ وب اطلاعاتی را وارد نموده و آنرا به سمت سرویس‌دهنده ارسال کند. سرویس‌دهنده پس از دریافت اطلاعات و پردازش آن ، مجدداً در پاسخ ، اطلاعاتی را برمی‌گرداند. برای چنین کاربردهایی صفحهٔ وب باید شامل اجزایی جهت ورود اطلاعات باشد. به عنوان مثال فرض کنید یک شرکت مایل است در سایت وب خود ، ضمن معرفی محصولاتِ شرکت ، از مشتریان سفارش بگیرد. بنابراین طراح صفحهٔ وب باید نواحی ورود اطلاعات مشتری را در صفحهٔ وب تعریف کند. پس از آنکه کاربر اطلاعات خود را در این نواحی وارد کرد ، مرورگر را وادار می‌کند تا آنها را برای سرویس‌دهنده بفرستد. در شکل (۱۴-۱۰) یک صفحهٔ وب با قابلیت ورود و ارسال اطلاعات نشان داده شده است. در جدول (۱۵-۱۰) اصل فایل HTML آن ارائه شده است.


(۱۴-۱۰) یک صفحهٔ وب با قابلیت ورود و ارسال اطلاعات

```
 <html>

<head>
<title>نمونه</title>
</head>

<body background="../../SampleWebSite/Nabkgnd.jpg" bgcolor="#FFFFFF">

<p align="center"><font size="6" face="Tabassom"><strong>نمونه‌ای
از صفحهٔ وب با قابلیت ارسال اطلاعات برای سرویس‌دهنده</strong></font></p>

<form method="POST" ACTION="http://www.malekian.com/cgi-bin/proc1.pl">
<p align="right"><font face="Nazanin">
<input type="text"  size="108" name="Name"><strong> : نام</strong></font
<img src="../Figures/shock.gif"  width="52" height="56"></p>
<p dir="rtl">
<img  src="../Figures/shock.gif" width="52" height="56">
<strong>  : نام خانوادگی</strong>
<input type="text" size="140"   name="Family"></p>
<p dir="rtl"><strong>
```

ادامه صفحهٔ بعد

```
<img  src="Telephone.gif" width="50" height="51">شماره تلفن تماس:</strong>
<input type="text" size="79"  name="TelNo">
</p>
<p dir="rtl"><strong>آدرس دقیق پستی:</strong>
<input type="text" size="251" name="Addr1">
</p>
<input type="text"  size="294" name="Addr2">
</p>
<img  src="../Figures/dan.gif" align="left" hspace="0"  width="127" height="180"><strong>
<img  src="../ Figures/fl.gif" width="52" height="56">نام محصول مورد نظر: </strong>
<input type="text"  size="165" name="ProductNName">
</p>
<strong>
<img src="../ Figures/fl.gif" width="52" height="56">کد محصول مورد نظر:</strong>
<input type="text"  size="170" name=" ProductNo">
</p>
<p align="center">
<input type="submit" name="B1"  value="Submit/ارسال">
</p> <p>
<input type="reset" name="B2"   value="Reset/پاک کردن همهٔ فیلدها">
</p>
</form>
</body>
</html>
```

(۱۵-۱۰) اصل فایل HTML مربوط به صفحهٔ وبِ شکل (۱۴-۱۰)

در جدول (۱۵-۱۰) تمام برچسبهایی که در زمینهٔ خاکستری نشان داده شده‌اند ، مربوط به دریافت اطلاعات از کاربر هستند. این برچسبها را با در نظرگرفتن مثال فوق بررسی می‌کنیم:

♦ برای تعیین ناحیهٔ ورود اطلاعات در یک صفحهٔ وب ، از برچسب <... FORM> استفاده می‌شود. ناحیهٔ ورود اطلاعات با برچسب </FORM> خاتمه می‌یابد. هر صفحهٔ وب می‌تواند چند ناحیهٔ ورود اطلاعات داشته باشد؛ اطلاعاتِ هر ناحیه بطور یکجا برای سرویس‌دهنده ارسال خواهد شد. در هر ناحیه حداقل یک گزینهٔ **"تسلیم/ارسال"**[1] وجود دارد که کاربر پس از آنکه اطلاعات این ناحیه را تکمیل کرد ، توسط آن به مرورگر فرمان می‌دهد تا آنها را برای سرویس‌دهنده ارسال کند. اطلاعات ارسالی در قالب مشخص و استاندارد تحویل یک پروسهٔ خاص بر روی ماشین سرویس‌دهنده خواهد شد. نام و آدرس این پروسه درون برچسب <... FORM> مشخص شده است. این پروسه که اصطلاحاً **CGI** نامیده می‌شود ، توسط سرویس‌دهنده فراخوانی شده و اطلاعات ارسالی را دریافت می‌دارد. در ضمن این پروسه

[1] Submit

وظیفه دارد اطلاعات دریافتی را پردازش و در صورت لزوم ذخیره کند و پاسخهای مناسب را به کاربر برگرداند. (در ادامه CGI را بررسی خواهیم کرد.)

برچسب <... FORM> دارای ویژگیهای زیر است:

- ACTION : آدرس URL مربوط به محل قرار گرفتن برنامهٔ CGI که اطلاعات ارسالی را دریافت و پردازش خواهد کرد.
- METHOD : یکی از متودهای HTTP که توسط آن اطلاعات به سمت سرویس‌دهنده ارسال می‌شود. این متودها می‌تواند GET یا POST باشد. [ به جدول (٤-١٠) مراجعه کنید؛ تفاوت این متودها در بخشهای آتی تشریح خواهد شد.]
- ENCTYPE : نوع کدگذاری داده‌های ارسالی را مشخص می‌کند. (چون ارسال اطلاعات از مرورگر به سرویس‌دهنده از استاندارد MIME تبعیت می‌کند لذا در این قسمت نوع کدگذاری داده‌ها مشخص می‌شود-مثل base64-. اگر این خصوصیت معرفی نشود ، اطلاعات ارسالی ، متن معمولی فرض خواهد شد. )

♦ **برچسب <... INPUT>**

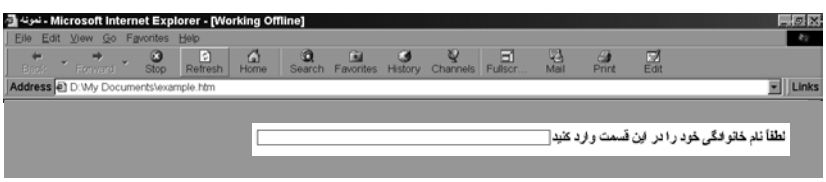
با استفاده از این برچسب می‌توان یکی از عوامل دریافت اطلاعات را در صفحهٔ وب تعریف کرد. ویژگیهای این برچسب عبارتند از:

- **TYPE** : در صفحهٔ وب می‌توان انواع اشیاء و عوامل ورود اطلاعات را تعریف کرد. با ویژگی TYPE ، می‌توان نوع عامل ورودی را تعیین کرد. انواع عوامل ورودی عبارتند از:

**TYPE=TEXT** : یک فضا برای ورود اطلاعات متنی فراهم می‌آورد. مثال:

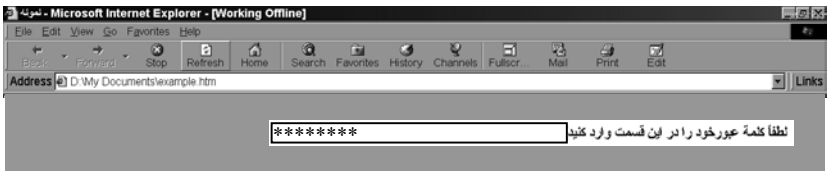
```
<p>
<input type="text" size="152" name="T1">
<strong>لطفاً نام خانوادگی خود را در این قسمت وارد کنید
</p >
```

نمایش این عامل ورودی در محیط مرورگر به صورت زیر است:



**TYPE=PASSWORD** : یک فضا برای ورود کلمهٔ عبور فراهم می‌آورد؛ با این تفاوت که در هنگام ورود اطلاعات ، محتوای آن دیده نخواهد شد. مثال:

```
<p>
<input type="password" size="152" name="T1">
لطفاً کلمهٔ عبور خود را در این قسمت وارد کنید<strong></p>
```




**TYPE=CHECKBOX** : یک لیست از گزینه‌ها را تعریف می‌کند تا کاربر بتواند به دلخواه ، هر کدام را انتخاب نماید. مثال:

```
<font face="Sina">
<p><input type="checkbox" checked name="C1" value="ON"><strong>گزینهٔ ۱</strong></p>
<p><input type="checkbox" name="C2" ><strong>گزینهٔ ۲</strong></p>
<p><input type="checkbox" name="C3" ><strong>گزینهٔ ۳</strong></p>
<p><input type="checkbox" name="C4" ><strong>گزینهٔ ۴</strong></p>
```


**TYPE=RADIO** : یک لیست از گزینه‌ها را تعریف می‌کند تا کاربر بتواند به دلخواه ، فقط یکی از آنها را انتخاب نماید. مثال:

```
<font face="Sina">
<p><input type="radio" checked name="R1" value="V1"><strong>گزینهٔ ۱</strong></p>
<p><input type="radio" name="R1" value="V2"><strong>گزینهٔ ۲</strong></p>
<p><input type="radio" name="R1" value="V3"><strong>گزینهٔ ۳</strong></p>
<p><input type="radio" name="R1" value="V4"><strong>گزینهٔ ۴</strong></p>
```


**TYPE=SUBMIT** : یک "کلید فشاری"[1] تعریف می‌کند تا کاربر به مرورگر فرمان بدهد که اطلاعات را به سمت سرویس‌دهنده ارسال نماید. مثال:

```
<p align="center">
<input type="submit" name="B1"  value="Submit/ارسال">
</p>
```

**TYPE=RESET** : یک "کلید فشاری" تعریف می‌کند تا کاربر به مرورگر فرمان بدهد که اطلاعات درون تمام عوامل ورودی اطلاعات را پاک نماید. مثال:

```
<p>
<input type="reset" name="B2"   value="Reset/پاک کردن همهٔ فیلدها">
</p>
```

در مثال زیر نقش کلید RESET مشخص شده است.


انتخاب نام برای تمام ورودیها بجز کلیدهای SUBMIT و RESET الزامی است زیرا وقتی مرورگر اطلاعات را برای سرویس‌دهنده ارسال می‌نماید ، نام فیلد مربوطه را هم به همراه

[1] Push Botton

مقدار آن ، ارسال خواهد کرد. برنامهٔ CGI ، وظیفه دارد نام فیلدها و مقادیر آنها را از هم تفکیک کند. در تمام اشیاء و عوامل ورودی ، گزینهٔ VALUE مقدار پیش‌فرض آنرا تعیین می‌کند. برای کلیدهای فشاری SUBMIT و RESET ، مقدار گزینهٔ VALUE عنوانی است که بر روی کلیدها ظاهر خواهد شد.

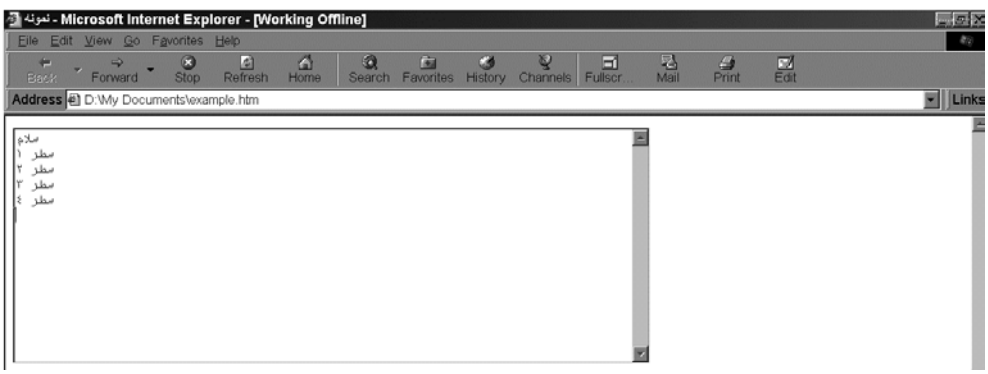
- **برچسب <... TEXTAREA>**

با استفاده از این برچسب می‌توان یک فضای چندخطی ورودی در صفحهٔ وب تعریف کرد. این فضا به کاربر اجازه خواهد داد تا متون طولانی خود را وارد کرده ، ارسال نماید. چنین فیلدی می‌تواند برای کاربردهایی نظیر ورود متن نامه یا نظرسنجی استفاده شود. برخی از ویژگیهای این برچسب عبارتند از:

- NAME : نام فیلد
- ROWS : تعداد سطرهای ورودی
- COLS : تعداد ستونهای ورودی
- DISABLED : ورودی را غیر فعال می‌کند.

مثال :

```
<p><textarea name="text1" rows="15" cols="80">
</textarea> </p>
```



### ۴-۲) قابلیتهای دیگر HTML

استاندارد HTML بطور مداوم رو به توسعه و پیشرفت است. پس از معرفی نسخهٔ سوم HTML در ۱۹۹۵ تقریباً هر سال ایده‌های جدیدی در این زمینه ارائه شده است. در نسخهٔ HTML 3.0 قابلیتهای زیر عرضه شده‌اند:

♦ **امکان رسم جدول** : جداول یکی از نیازهای اصلی صفحات وب تلقی می‌شوند در حالی که نسخه‌های قدیمی HTML ، این امکان را فراهم نکرده بودند. در نسخهٔ ۳ ، رسم جدول با برچسب <TABLE ...> امکان‌پذیر شد. قالب این برچسب بصورت زیر است:

<TABLE WIDTH="?" COLS="?" BORDER="?" FRAME="?" CELLSPACING="?" CELLPADDING="?">

WIDTH : عرض جدول  COLS : تعداد ستونها

BORDER : ضخامت حاشیهٔ اطراف جدول بر حسب پیکسل

FRAME : حاشیهٔ قابل رویت اطراف جدول

CELLSPACING : فضای خالی میان خانه‌های جدول

CELLPADDING : فضای خالی بین داده‌های درون جدول

پس از تعریف ساختار جدول سطر و ستونهای آن با برچسبهای زیر تعریف می‌شوند:

**برچسبهای <TR> و </TR>** : یک سطر از جدول را تعیین می‌کند.

**برچسبهای <TD> و </TD>** : محتوی هر یک از خانه‌های جدول توسط یک برچسب <TD> مشخص می‌شود.

به مثال زیر دقت کنید. در این مثال یک جدول ۳×۳ تعریف و به خانه‌های آن مقدار داده شده است.

```
<font size="6" face="Nazanin">
<table border="3" cellpadding="3" cellspacing="3">
  <tr>
    <td><p align="center"><strong>ستون ۱ سطر ۱ </strong></td>
    <td><p align="center"><strong>ستون ۲ سطر ۱ </strong></td>
    <td><p align="center"><strong>ستون ۳ سطر ۱ </strong></td>
  </tr>
<tr>
    <td><p align="center"><strong>ستون ۱ سطر ۲ </strong></td>
    <td><p align="center"><strong>ستون ۲ سطر ۲ </strong></td>
    <td><p align="center"><strong>ستون ۳ سطر ۲ </strong></td>
 </tr>
```

```
<tr>
    <td><p align="center"><strong>ستون ۱ سطر ۳ </strong></td>
    <td><p align="center"><strong>ستون ۲ سطر ۳ </strong></td>
    <td><p align="center"><strong>ستون ۳ سطر ۳ </strong></td>
  </tr>
</table>
```


♦ **جاسازی یک APPLET در صفحهٔ وب** : در یک صفحـهٔ وب بـا اسـتفاده از بـرچسـبهای <... APPLET> می‌توان اپلتهای جاوا را جاسازی کرد. اپلتها ، قابلیتهای حرفه‌ای شبیه مدیریـت نقشه‌های تصویری ، امکانات چندرسـانه‌ای و بازیـهای کـامپیوتری را بـه صفحـات وب ارائـه می‌کنند. مثال :

```
<APPLET CODE="FirstApplet" WIDTH=300 HEIGHT=200> </APPLET>
```

CODE : نام فایل اپلت
WIDTH : عرض پنجره‌ای که اپلت در آن اجرا می‌شود.
HEIGHT : ارتفاع پنجره‌ای که اپلت در آن اجرا می‌شود.

♦ **امکانات فرمول‌نویسی** : در نسخه‌های جدید HTML ارائهٔ فرمولهای ریاضی در یک صفحهٔ وب امکان‌پذیر شده است. این قابلیت معمولاً یکی از نیازهای اولیهٔ سایتهای علمی و تحقیقاتی تلقی می‌شود.

در خاتمهٔ این بخش بار دیگر تاکید می‌کنیم که هدف از معرفی HTML ، فقط آشنایی با اصـول آن بوده است و برای آشنایی تفصیلی با آن باید به مراجع آخر فصل مراجعه کنید.

### ۴-۳) مزیتهای HTML

HTML اولین زبان نشانه‌گذاری متن نبود و در دههٔ نود (دورهٔ تولد و رشد فراگیر HTML ) حتی ایدهٔ جدیدی هم محسوب نمی‌شد. رشد HTML ناشی از تواناییهای آن در برآورده کردن نیازهای شبکهٔ اینترنت ، در موارد زیر بوده است :

- **استقلال**[1] : HTML یک استاندارد مبتنی بر کدهای ASCII است و و هیچ وابستگی اجرایی به سخت‌افزار و نرم‌افزار ندارد. یک فایل HTML می‌تواند بین دو ماشین کاملاً متفاوت مبادله شود ، بدون آنکه هیچ نگرانی در مورد عدم سازگاری ماشینها وجود داشته باشد. هر ماشین با سخت‌افزار و نرم‌افزار خاص خود ، با استفاده از مرورگر سازگار با محیط خود ، یک فایل HTML را تفسیر کرده و نمایش خواهد داد. با توجه به تنوع ماشینها و سیستمهای عامل در شبکهٔ اینترنت ، این خصوصیت یکی از نیازهای بنیادی محسوب می‌شود.
- **سرعت و سادگی** : فایلهای HTML از نظر اندازه و حجم فایل ، کوچک هستند و برای پایین آوردن ترافیک شبکه ، ابزاری مناسب محسوب می‌شوند؛ در عین حال پیچیدگی خاصی ندارند و به راحتی می‌توان یک فایل HTML ایجاد و استفاده کرد. سادگی این استاندارد باعث شد که در زمان بسیار کوتاهی ابزارهای قدرتمند طراحی صفحات وب پدید آید و تولید صفحات وب در حد یک کار تجربی و بدون نیاز به اطلاعات فنی ، برای عموم آسان شود.
- **امکان دریافت اطلاعات از صفحهٔ وب** : صفحات وب ابزار بسیار ساده و مناسبی برای دریافت اطلاعات از کاربر و ارسال آن به سرویس‌دهنده هستند؛ کاری که اگر نیاز باشد مستقل از وب انجام شود ، به تخصص و زمان بسیار زیادی احتیاج دارد. این امکان باعث شده که محیط وب از حالت اطلاع‌رسانی صِرفْ درآمده و به یک ابزار مناسب و سریع جهت امور اقتصادی ، اداری ، تجاری تبدیل شود.
- **سازمان‌دهی سلسله‌مراتبی** : صفحات وب با استفاده از مفهوم ابرپیوند ، مستندات را بصورت سلسله‌مراتبی و دسته‌بندی‌شده به متقاضی عرضه می‌کنند. در این روش دسترسی به اطلاعات ، بسیار سریع و ساده خواهد شد.
- **پشتیبانی همگانی** : سادگی و جذابیتِ وب باعث شد که تمام توسعه‌دهندگان نرم‌افزار ، در سیستمهای خود از آن پشتیبانی کنند. امروزه سیستمی را نمی‌توان یافت که از وب پشتیبانی نکند یا مرورگر نداشته باشد. امروزه اکثر بانکهای اطلاعاتی قادرند در محیط وب نیز بکارگرفته شوند. یعنی کاربر از راه دور و با استفاده از مرورگر و استاندارد HTML با آن تراکنش داشته باشد.

---

[1] Platform Independence

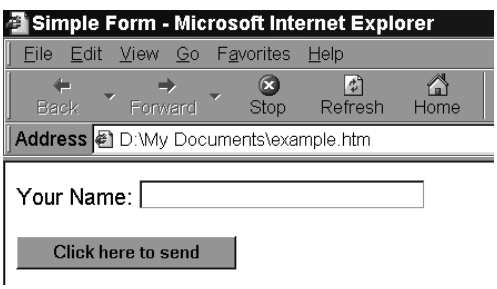
## ۵) برنامه‌های CGI[1]

فرض کنید که یک صفحهٔ وب ، اطلاعاتی را از کاربر دریافت کند. کاربر می‌تواند با فشار دادن کلید SUBMIT آنها را برای سرویس‌دهنده ارسال نماید. سرویس‌دهندهٔ HTTP فقط وظیفهٔ دریافت یا ارسال داده‌ها را بر عهده دارد و کاری در مورد پردازش آنها انجام نمی‌دهد. حال دو سوال زیر مطرح می‌شود :

- کدام برنامه بر روی ماشین سرویس‌دهنده داده‌های ارسالی از مرورگر را دریافت و پردازش می‌نماید؟
- مرورگر بر اساس چه الگویی داده‌ها را ارسال می‌کند و برنامهٔ پردازش‌کنندهٔ داده‌ها چگونه آنها را از سرویس‌دهندهٔ HTTP تحویل می‌گیرد؟

وقتی طراح صفحهٔ وب ، یک ناحیه را برای ورود اطلاعات با برچسب <FORM ...> تعریف می‌کند ، موظف است در درون این برچسب آدرس برنامهٔ تحویل‌گیرنده و پردازش‌کنندهٔ داده‌ها را دقیقاً مشخص نماید. به عنوان مثال به قطعه کد HTML زیر و نمایش آن در محیط مرورگر دقت کنید.

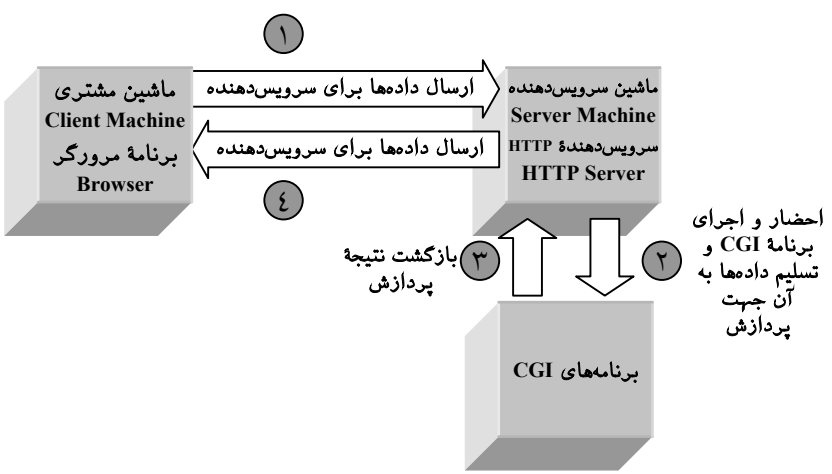
```
<HTML>
<HEAD><TITLE>Simple Form</TITLE></HEAD>
<BODY>
<FORM Method="POST" Action="http://www.abcdef.com/cgi-bin/prog.exe">
 Your Name:
<INPUT Name="user" SIZE="30"><P>
<INPUT Type=submit Value="Click here to send">
</FORM>
</BODY>
</HTML>
```




[1] Common Gateway Interface

در این قطعه کد ، نام و محل برنامهٔ تحویل گیرندهٔ داده‌ها با آدرس URL http://www.abcdef.com/cgi-bin/prog.exe مشخص شده است. وقتی مرورگر داده‌ها را برای سرویس‌دهندهٔ HTTP ارسال می‌کند ، سرویس‌دهنده برنامهٔ prog.exe را بارگذاری و اجرا کرده و داده‌ها را تحویل آن می‌دهد. این برنامه قادر است ضمن پردازش داده‌ها ، پاسخهای مناسب را در قالب استاندارد HTML تولید کرده و برای کاربر باز پس بفرستد. این برنامه اصطلاحاً CGI نامیده می‌شود. برنامه‌های CGI با نامهای "اسکریپتِ CGI"[1] یا "برنامهٔ کاربردی CGI"[2] هم معرفی شده‌اند.

CGI استانداردی است که چگونگی ارتباط برنامه‌های جانبی با سرویس‌دهندهٔ HTTP را تبیین می‌کند. پروتکل HTTP به تنهایی فقط قادر به ارسال و دریافت صفحات وب است و برنامه‌های CGI در کنار این پروتکل می‌توانند یک ارتباط دوطرفه با کاربر ایجاد نمایند؛ به گونه‌ای که کاربر می‌تواند از راه دور با این برنامه‌ها تراکنش داشته باشد. در حقیقت HTTP به عنوان یک پروتکل واسطِ انتقال داده ، بین کاربر و این برنامه‌ها انجام وظیفه می‌کند. شکل (١٦-١٠) این مفهوم را نشان می‌دهد.




شکل (١٦-١٠) تراکنش مرورگر و برنامهٔ CGI از طریق پروتکل HTTP

[1] CGI Script
[2] CGI Application

برنامه‌های CGI را می‌توان به زبانهای مختلفی نوشت و نیاز به ابزار خاصی ندارد. زبانهایی که امکان نوشتن برنامه‌های CGI را فراهم آورده‌اند ، عبارتند از:

- C
- C++
- Perl
- Tcl
- Visual Basic (Micosoft Windows)
- Shell Scripts (UNIX)
- Apple Scripts
- Delphi

برنامه‌های CGI ، برنامه‌های کاملاً معمولی هستند؛ تنها تفاوت آنها در دریافت داده‌ها از ورودی است. برنامه‌های معمولی داده‌های خود را از طریق صفحه‌کلید یا موْس دریافت می‌کنند ، درحالی که برنامه‌های CGI ورودیهای خود را از سرویس‌دهندهٔ HTTP می‌گیرند. خروجیهای این برنامه‌ها نیز به سرویس‌دهندهٔ HTTP ارسال می‌شود تا برای مشتری فرستاده شده و در محیط مرورگر نشان داده شود. برنامه‌های CGI معمولاً هیچگونه خروجی یا پنجره بر روی ماشین سرویس‌دهنده ندارند. (مگر در موارد خاص)

بطور معمول طراح یک صفحهٔ وب ، خودش برنامهٔ CGI متناظر با آن را برنامه‌نویسی می‌کند؛ زیرا نام فیلدها و اشیاء ورودی در یک صفحهٔ وب ، باید در برنامهٔ CGI متناظر با آن شناخته شده و تطابق داشته باشد و مرورگر محتوای هر یک از این فیلدها را همراه نام فیلد ، ارسال می‌نماید.

در ادامه باید الگوی ارسال داده‌ها را به یک برنامهٔ CGI بررسی کنیم.

### ۱-۵) الگوهای ارسال اطلاعات برای یک برنامهٔ CGI

به دو روش سرویس‌دهندهٔ HTTP یک برنامهٔ CGI را راه‌اندازی کرده و داده‌های ارسالی از مرورگر را تحویل آن می‌دهد. طراح صفحهٔ وب ، در برچسب <... FORM> ، ضمن معرفی کردن محل و نام برنامهٔ CGI ، روش تسلیم داده‌ها را نیز تعریف می‌نماید. این دو روش عبارتند از:

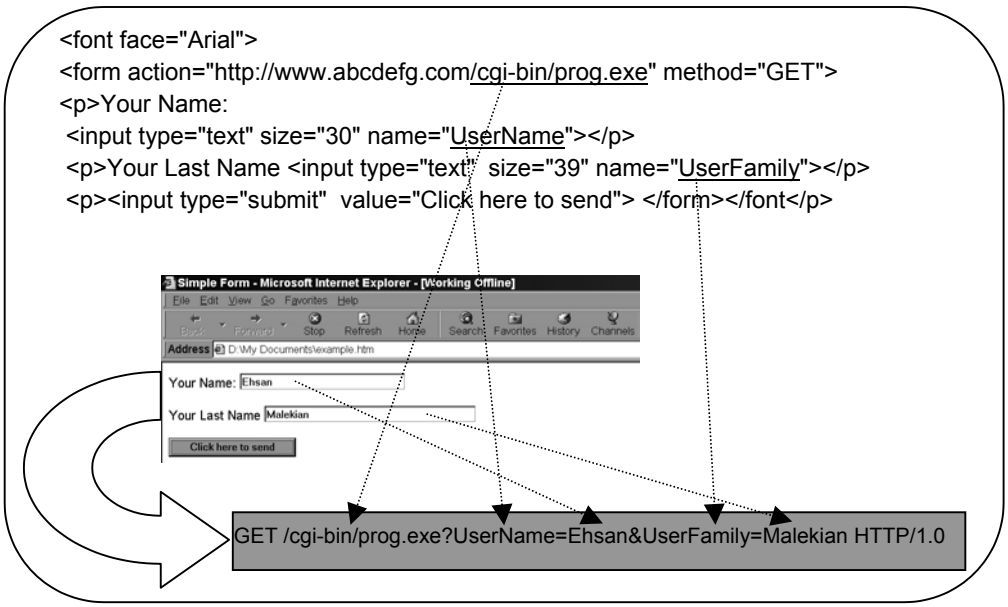
- استفاده از الگوی GET :

```
<FORM Method="GET" Action="http://www.abcdef.com/cgi-bin/prog.exe">
```

- استفاده از الگوی POST :

```
<FORM Method="POST" Action="http://www.abcdef.com/cgi-bin/prog.exe">
```

اگر از الگوی GET برای ارسال داده‌ها استفاده شود ، داده‌های جمع‌آوری شده از صفحهٔ وب به آدرس URL آن ضمیمه شده و به سمت سرویس‌دهنده ارسال خواهد شد. مثال زیر بسیار گویاست:




به گونه‌ای که نشان داده شده اگر از الگوی GET استفاده شود ، مرورگر پس از برقراری ارتباط TCP با سرویس‌دهندهٔ HTTP ، از فرمان معمولی GET استفاده می‌کند ، با این تفاوت که در ادامهٔ آدرس URL ، نام و مقادیر هر فیلد ضمیمه و ارسال می‌شود. وقتی سرویس‌دهندهٔ HTTP این رشته را دریافت می‌کند ، برنامهٔ مشخص شده را اجرا کرده و ادامهٔ رشته را ( بعد از کاراکتر ؟ ) در یک "متغیر محیطی"[1] به نام QUERY_STRING قرار می‌گیرد.[2] برنامهٔ CGI

[1] Environment Variable

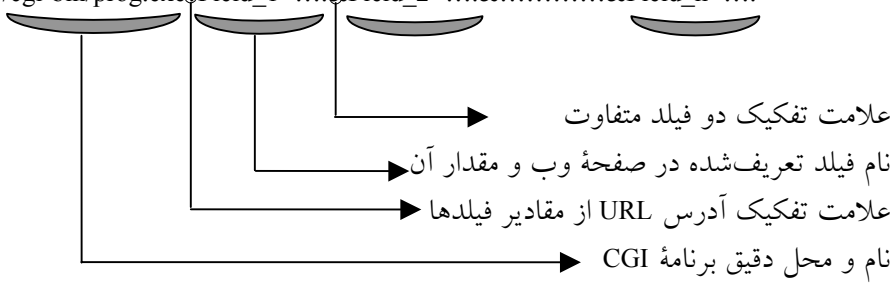
[2] **متغیر QUERY_STRING ، در برنامه نیاز به تعریف ندارد بلکه با استفاده از تابع getenv(...) ، ( در زبان C یا توابع معادل در دیگر زبانها ) می‌توان به آن دسترسی داشت.**

می‌تواند داده‌ها را از این متغیر استخراج ، تجزیه و تحلیل[1] و پردازش نماید. اگر بخواهیم این روش فراخوانی برنامهٔ CGI را با برنامه‌ها معمولی مقایسه کنیم ، همانند فراخوانی برنامه در خط فرمان بهمراه پارامترهای ورودی است؛ در این برنامه‌ها ، متغیرهای argc و argv[] پارامترهای خط فرمان را در اختیار برنامه‌نویس قرار می‌دهند.

الگوی ارسال داده‌های یک صفحهٔ وب بصورت زیر است:

فرض کنید عوامل و اشیاء یک صفحهٔ وب به نامهای فرضی Field_1 تا Field_n نامگذاری شده باشد. هرگاه کاربر کلید SUBMIT را فشار داد ، رشتهٔ زیر تولید شده و ارسال خواهد شد:

GET /cgi-bin/prog.exe?Field_1=….&Field_2=….&……&Field_n=….




رعایت قواعد زیر در تدوین رشتهٔ ارسالی ، لازم است:

- محل و نام برنامهٔ CGI با علامت **?** از بقیهٔ رشته جدا می‌شود.
- نام هر فیلد و مقدار فیلد با علامت = از هم تفکیک می‌شود.
- اگر قرار باشد چندین فیلد و مقادیر آن ارسال شود ، نام و مقدار هر فیلد با علامت **&** از هم تفکیک می‌شود.
- اگر در بین داده‌ها فاصلهٔ خالی (Blank) وجود داشته باشد ، باید با علامت + جایگزین شود.
- اگر در بین داده‌ها کاراکترهای ASCII با کد زیر ۳۳ یا یکی از علامتهای (+ = & ٪ ؟) وجود داشته باشد ، ابتدا علامت % و سپس کد هگزادسیمال آن بجای آن کاراکتر قرار می‌گیرد. مثلاً کاراکتر با کد ۳۲ بصورت 20% یا کاراکتر با کد ۳۰ بصورت 1E% جایگزین می‌شود.

مثال :

GET /cgi-bin/prog.exe?Name=Ali+Reza&Family=Ahmadi+Tehrani

[1] Parse

استفاده از الگوی GET زمانی مناسب خواهد بود که مجموع کل رشته‌ای که به سمت سرویس‌دهنده ارسال می‌شود زیر هزار کاراکتر باشد ، زیرا سرویس‌دهنده‌های HTTP رشته‌های با طول بیش از هزار کاراکتر را نخواهند پذیرفت.[1] اکثر برنامه‌های CGI از الگوی POST استفاده می‌کنند.

### ۲-۵) متغیرهای محیطی قابل استفاده در یک برنامهٔ CGI

به غیر از متغیر محیطی QUERY_STRING ، متغیرهای دیگری نیز هستند که برنامهٔ CGI می‌تواند در صورت لزوم از آنها استفاده کند. این متغیرهای محیطی در جدول (۱۷-۱۰) معرفی شده‌اند. برای دسترسی به این متغیرهای در یک زبان برنامه‌نویسی همانند C باید آدرس آنرا با استفاده از تابع سیستمی (**"نام متغیر محیطی"**)getenv بدست آورد. مثال :

```
p = getenv("CONTENT_LENGTH");
 if(p != NULL) {…}
else { …. }
```

| نام متغیر محیطی | محتوی |
|---|---|
| REQUEST_METHOD | الگوی ارسال داده‌ها ( یکی از دو مقدار GET یا POST ) |
| QUERY_STRING | رشتهٔ بعد از علامت ؟ ضمیمه شده به URL ( رشتهٔ حاوی نام و مقادیر فیلدها ) |
| CONTENT_LENGTH | طول داده‌های ارسالی به برنامهٔ CGI (بر حسب بایت) (اگر الگوی ارسال POST باشد.) |
| CONTENT_TYPE | نوع داده‌های ارسالی به برنامهٔ CGI (مشخصهٔ MIME) (اگر الگوی ارسال POST باشد.) |
| PATH_INFO | مسیر دقیق و محل قرار گرفتن برنامهٔ CGI (مشخص شده در URL ) |
| HTTP_AccEPT | نوعی از قالب داده‌ها که برنامهٔ مرورگر قادر است ، بپذیرد. (مشخصهٔ MIME) |
| GATEWAY_INTERFACE | نام و نسخهٔ CGI ( مثلاً CGI/1.1 ) |
| SERVER_PORT | شمارهٔ پورت سوریس‌دهندهٔ HTTP که تقاضا به آن ارسال شده است. (بطور معمول ۸۰) |
| SERVER_PROTOCOL | نام و نسخهٔ سرویس‌دهندهٔ وب ( مثلاً HTTP/1.0 ) |
| SERVER_SOFTWARE | نام و نسخهٔ نرم‌افزار نصب شده بعنوان سرویس‌دهندهٔ وب ( مثلاً IIS/4.0 ) |
| SERVER_NAME | نام ماشینی که سرویس‌دهندهٔ وب بر روی آن نصب شده ( مثلاً www.ibm.com ) |
| REMOTE_ADDR | آدرس IP ماشین سرویس‌دهندهٔ وب |

جدول (۱۷-۱۰) متغیرهای محیطی قابل استفاده در یک برنامهٔ CGI

[1] در برخی از سرویس‌دهنده‌ها ، حداکثر طول یک خط ۲۵۵ کاراکتر و پیش‌فرض آن ۸۰ است.

### ۳-۵) الگوی POST

برای آن دسته از صفحات وب که حجم نامشخصی از داده‌ها را برای سرویس‌دهنده ارسال می‌کند ، برنامهٔ CGI باید از الگوی POST استفاده کند. اگر برای فراخوانی برنامهٔ CGI از الگوی POST استفاده شود ، سرویس‌دهندهٔ HTTP ، داده‌ها را از طریق متغیر محیطی QUERY_STRING به برنامه نمی‌فرستد بلکه از طریق "ورودی استاندارد" ( یعنی همان مفهوم stdin در زبان C ) به برنامه هدایت می‌شود. در این حالت برنامهٔ CGI می‌تواند از دستورات معمولی خواندن از کنسول ورودی ( مثل تابع fscanf() یا fgetc() در زبان C یا cin در C++ ) برای دریافت داده‌ها از سرویس‌دهنده اقدام کند.

هر برنامهٔ CGI ممکن است بخواهد برای اطلاع کاربر نتیجه‌ای را به مرورگر برگرداند. واضح است که خروجی یک برنامهٔ CGI باید روی مرورگر کاربر نشان داده شود و اینگونه برنامه‌ها بطور معمول خروجی خاصی بر روی سرویس‌دهنده ندارند. برای برگرداندن اطلاعات به مرورگری که برای یک برنامهٔ CGI داده ارسال کرده است ، از "خروجی استاندارد" ( یعنی مفهوم stdout در زبان C ) استفاده می‌شود. یعنی برای نمایش خروجی‌ها در برنامهٔ CGI می‌توان از دستورات معمولی نوشتن روی کنسول خروجی ( مثل تابع printf() در زبان C یا cout در C++ ) استفاده کرد.

در حقیقت سرویس‌دهندهٔ HTTP هنگامی که یک برنامهٔ CGI را فراخوانی می‌کند ، مسیر کنسول ورودی و خروجی استاندارد ( stdin و stdout ) را به سمت خودش برمی‌گرداند.[۱]

در هنگام نوشتن بر روی کنسول خروجی (یعنی زمانی که پیغامی جهت نمایش برای مرورگر ارسال می‌شود) باید به دو نکتهٔ اساسی زیر دقت شود:

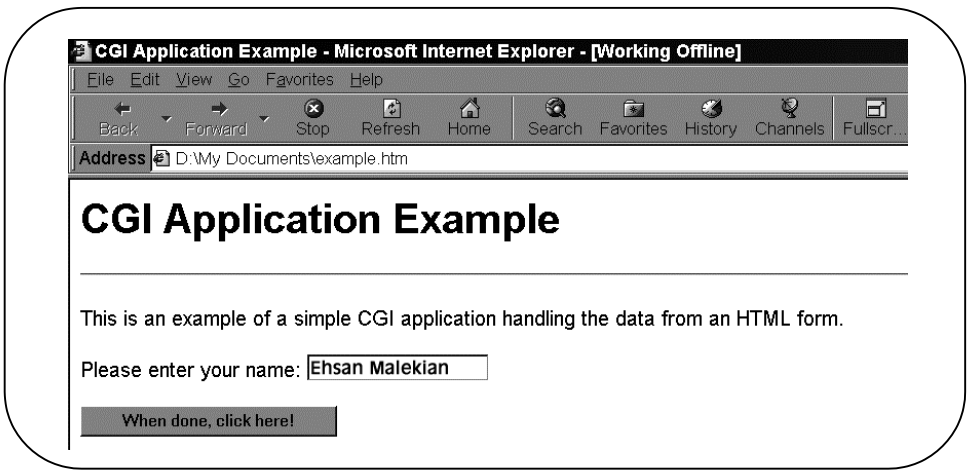
- با توجه به آنکه هر چیزی که با دستورات معمولی مثل printf() بر روی خروجی نوشته می‌شود ، بر روی مرورگر نشان داده خواهد شد ، لذا پیغامها باید در قالب یک فایل HTML نوشته شوند.
- هر خط ارسالی برای مرورگر باید با دو کاراکتر \n\n خاتمه یابد.

برای آشنایی با روش برنامه‌نویسی CGI ارائهٔ یک مثال بسیار راهگشا خواهد بود. به صفحهٔ وب شکل (۱۸-۱۰) و فایل HTML آن دقت کنید.

---

[۱] **برای درک این مفهوم تکنیک Piping در یونیکس و مفهوم کنسول ( Consol I/O ) را به یاد بیاورید.**

```
<HTML>
<HEAD>
<TITLE>CGI Application Example</TITLE>
</HEAD>
<BODY>
<H1>CGI Application Example</H1>
<hr>
This is an example of a simple CGI application
handling the data from an HTML form.
<BR>
<FORM ACTION="http://www.hqz.com/scripts/cgisamp.exe" METHOD="Post">
Please enter your name: <INPUT NAME="name" TYPE="text"><p>
<input type=submit value="When done, click here!">
</FORM>
</BODY>
</HTML>
```




شکل (۱۸-۱۰) یک صفحهٔ وب فرضی برای فراخوانی برنامهٔ CGI

وقتی کاربر "کلید ارسال" را فشار می‌دهد ، سرویس‌دهنده برنامه‌ای به نام cgisamp.exe را فراخوانی کرده و ورودی را به کنسول stdin هدایت می‌نماید. در ضمن متغیرهای محیطی جدول (۱۷-۱۰) را نیز تنظیم می‌کند. ( به غیر از QUERY_STRING ) برنامهٔ cgisamp.exe را که به زبان C نوشته شده ، بررسی می‌نماییم. در این برنامه پنج تابع تعریف شده است:

- تابع void strcvrt(char *cStr, char cOld, char cNew) : رشتهٔ cStr را جستجو کرده و هرگاه کاراکتر cOld را در آن بیابد به کاراکتر cNew تبدیل می‌کند.
- تابع static int TwoHex2Int(char *pC) : اگر در یک رشتهٔ ورودی ، کدهایی باشند که با علامت % و معادل هگزادسیمال آن مشخص شده باشند آنرا به کد اصلی بر می‌گرداند.

- تابع void urlDecode(char *p) : این تابع تمام رشتهٔ دادهٔ ارسالی از مرورگر را بررسی و به حالت اصلی تبدیل می‌کند.
- تابع void StoreField(char *f, char *Item) : این تابع زوج "نام فیلد/مقدار" را از رشتهٔ ورودی استخراج می‌کند.
- تابع اصلی برنامه main() : این برنامه داده‌های ارسالی توسط مرورگر را از stdin می‌خواند و بر اساس ورودی (نام کاربر) ، یک خروجی ساده طبق شکل (۱۹-۱۰) برای آن تولید و و از طریق stdout برای مرورگر ارسال می‌کند.

```
<HEAD><TITLE>Submitted OK</TITLE></HEAD>
<BODY><h2>The information you supplied has been accepted.
<br>سپاسگزار از لطف شما<br>Ehsan Malekian</h2>
<h3><A href="http://www.hqz.com/cgisamp.htm">
[Return]</a></h3></BODY>
```


شکل (۱۹-۱۰) خروجی تولید شده توسط برنامهٔ CGI

در ادامه اصل برنامهٔ CGI که به زبان C نوشته شده ، ارائه شده است. این برنامه با صفحهٔ وب شکل (۱۸-۱۰) تراکنش داشته و یک خروجی مطابق با شکل (۱۹-۱۰) تولید و ارسال می‌نماید. این برنامه با نام cgisamp.exe بر روی سرویس‌دهنده نصب می‌شود. این برنامه را در جدول (۲۰-۱۰) ملاحظه می‌کنید. اگر به دستورات printf() دقت کنید هیچ تفاوتی با برنامه‌نویسی معمولی زبان C ندارند؛ با این تفاوت که پیغامهای ارسالی روی خروجی ، متن معمولی نیستند بلکه با برچسبهای HTML غنی شده‌اند تا بر روی مرورگر نشان داده شوند.

```
/**************************************************************************
* File: cgisamp.c
*
* Use: CGI Example Script.
*
* Notes: Assumes it is invoked from a form and that REQUEST_METHOD is POST.
* Ensure that you compile this script as a console mode app.
*
* This script is a modified version of the script that comes with EMWAC
* HTTPS.
*
* Date: 8/21/95
* Christopher L. T. Brown clbrown@netcom.com
*
**************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <io.h>

char InputBuffer[4096];
static char * field;
static char * name;

/* Convert all cOld characters   */
/* in cStr into cNew characters. */
void strcvrt(char *cStr, char cOld, char cNew)
{
    int i = 0;

    while(cStr[i])
    {
        if(cStr[i] == cOld)
            cStr[i] = cNew;
        i++;
    }
}

/* The string starts with two hex */
/* characters.  Return an integer */
/* formed from them.              */
static int TwoHex2Int(char *pC)
{
    int Hi, Lo, Result;

    Hi = pC[0];
    if('0' <= Hi && Hi <= '9')
        Hi -= '0';
    else if('a' <= Hi && Hi <= 'f')
        Hi -= ('a' – 10);
    else if('A' <= Hi && Hi <= 'F')
        Hi -= ('A' – 10);

    Lo = pC[1];
    if('0' <= Lo && Lo <= '9')
        Lo -= '0';
    else if('a' <= Lo && Lo <= 'f')
```

```
            Lo -= ('a' – 10);
        else if('A' <= Lo && Lo <= 'F')
            Lo -= ('A' – 10);

        Result = Lo + 16 * Hi;
        return(Result);
    }

    /* Decode the given string in-place */
    /* by expanding %XX escapes.       */
    void urlDecode(char *p)
    {
        char *pD = p;

        while(*p)
        {
            if (*p == '%')      /* Escape: next 2 chars are hex          */
            {                   /* representation of the actual character.*/
                p++;
                if(isxdigit(p[0]) && isxdigit(p[1]))
                {
                    *pD++ = (char)TwoHex2Int(p);
                    p += 2;
                }
            }
            else
                *pD++ = *p++;
        }
        *pD = '\0';
    }

    /* Parse out and store field=value items. */
    /* Don't use strtok!                      */
    void StoreField(char *f, char *Item)
    {
        char *p;

        p = strchr(Item, '=');
        *p++ = '\0';
        urlDecode(Item);
        urlDecode(p);
        strcvrt(p, '\n', ' ');
        strcvrt(p, '+', ' ');       /* Get rid of those nasty +'s */
        field = f;                  /* Hold on to the field just in case. */
        name = p;                   /* Hold on to the name to print*/
    }

    int main(void)
    {
        int ContentLength, x, i;
        char *p,
            *pRequestMethod,
            *URL,
            *f;

        /* Turn buffering off for stdin.*/
        setvbuf(stdin, NULL, _IONBF, 0);

        /* Tell the client what we're going to send */
```

```
        printf("Content-type: text/html\n\n");

        /* What method were we invoked through? */
        pRequestMethod = getenv("REQUEST_METHOD");

        /* Get the data from the client     */
        if(strcmp(pRequestMethod,"POST") == 0)
        {
            /* according to the requested method.*/
            /* Read in the data from the client. */
            p = getenv("CONTENT_LENGTH");
            if(p != NULL)
                ContentLength = atoi(p);
            else
                ContentLength = 0;
            if(ContentLength > sizeof(InputBuffer) –1)
                ContentLength = sizeof(InputBuffer) –1;

            i = 0;
            while(i < ContentLength)
            {
                x = fgetc(stdin);
                if(x == EOF)
                    break;
                InputBuffer[i++] = x;
            }
            InputBuffer[i] = '\0';
            ContentLength = i;

            p = getenv("CONTENT_TYPE");
            if(p == NULL)
                return(0);

            if(strcmp(p, "application/x-www-form-urlencoded") == 0)
             {
                p = strtok(InputBuffer, "&");      /* Parse the data */
                while(p != NULL)
                {
                    StoreField(f, p);
                    p = strtok(NULL, "&");
                }
             }
         }

        URL = getenv("HTTP_REFERER");           /* What url called me.*/
        printf("<HEAD><TITLE>Submitted OK</TITLE></HEAD>\n");
        printf("<BODY><h2>The information you supplied has been accepted.");
        printf("<br> سپاسگزار از لطف شما<br>%s</h2>\n", name);
        printf("<h3><A href=\"%s\">[Return]</a></h3></BODY>\n", URL);

        return(0);
    }
```

جدول (۲۰–۱۰) اصل برنامهٔ CGI

## ۶) مفهوم حقیقت مجازی[۱] -VR-

**"حقیقت مجازی"** (که به اختصار VR گفته می شود ) در یک مفهوم عام ، به صورت زیر تعریف می‌شود :

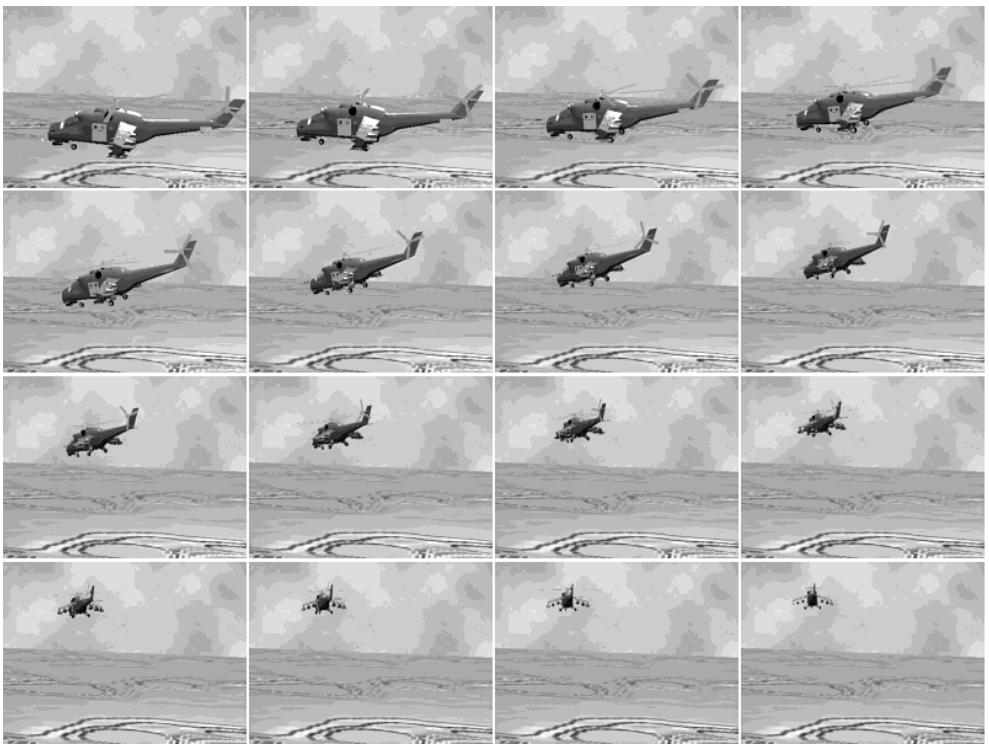
" شبیه‌سازی فضای سه‌بعدی توسط تکنیکهای نرم‌افزاری روی فضائی که ذاتاً دو بعدی است؛ بگونه‌ای که کاربر می‌تواند به غیر از مشاهدهٔ این فضا ، در آن سیر کرده و در محیط تغییراتی را نیز ایجاد نماید."

مفهومی که ارائه شد ذاتاً دارای ابهام است؛ زیرا هر کس می‌تواند از آن استنتاج شخصی خود را داشته باشد. برای روشنتر شدن قضیه ، صفحه شطرنجی را در نظر بگیرید که از روبرو به آن می‌نگرید. در محیط VR شما می‌توانید این صفحه را طبق تمایلتان بچرخانید و از زوایای مختلف به آن نگاه کنید؛ در عین حال اشیاء صفحه (مهره‌ها) را جابجا کنید (با استفاده از صفحه کلید ، موس ، دسته فرمان[۲]) و در نهایت نتیجه را روی صفحه ببینید. چنین کاری با تصاویر دو بعدی گرفته شده از طبیعت امکان پذیر نیست ، زیرا شما ملزم به دیدن تصویر از زاویه‌ای هستید که دوربین آن را ثبت کرده است. قابلیت تغییر اشیاء در محیط VR به این دلیل است که هر شیی در این محیط مشخصات و مختصات فضای سه‌بعدی خود را دقیقاً حفظ می‌کند؛ ولی چون صفحه نمایش دو بعدی است ، بعد سوم با استفاده از روشهای پرسپکتیو روی فضای دو بعدی نگاشته می‌شود.[۳] چون اشیاء در محیط VR مختصات فضایی (x,y,z) خود را حفظ می‌کنند ، بنابراین با تغییر زاویه دید یا چرخانیدن صفحه ، دور یا نزدیک شدن به اشیاء ، عمل نگاشت بُعد z روی صفحه x-y از نو انجام می‌شود؛ همانند عملی که در محیطهای واقعی برای چشم اتفاق می‌افتد. – شکل (۲۱–۱۰) – اگر هنوز در مفهوم حقیقت مجازی (VR ) شک دارید یا به یقین نرسیده‌اید تا انتهای فصل صبر کنید .

ممکن است حقیقت مجازی در ذهن شما محیط های چندرسانه‌ای[۴] را تداعی کند و این چندان عجیب نیست چرا که هر دوی این محیطها شامل صدا و تصویر هستند و هر دوی آنها تقریباً با هم رشد کرده‌اند و پیدایش آنها در دنیای "تکنولوژی اطلاعات"[۵]سبب ایجاد تحولات شگرف شده است؛ ولیکن تفاوتهای بنیادی بین محیطهای حقیقت مجازی ( VR ) و محیطهای

---

۱ Virtual Reality
۲ Joystick
۳ Mapping
۴ Multimedia
۵ Information Technology



شکل (۲۱-۱۰) تعریف یک شیی در محیط VR و نمایش آن از زوایای مختلف

چندرسانه‌ای وجود دارد. این تفاوت‌ها را می‌توان در موارد کلی زیر خلاصه کرد:

- محیط‌های چندرسانه‌ای با داده‌هائی سروکار دارند که از قبل جمع آوری و برنامه‌ریزی شده‌اند. اینها در حقیقت یکسری اطلاعات دسته‌بندی و ذخیره شده هستند که به صورت متوالی و با یک روال از قبل مشخص در یک فضای دو بعدی به نمایش درمی‌آیند. فیلم ، تصاویر متحرک (به همراه صدا) و یا هر چیزی در دنیای چندرسانه‌ای ، یکسری تصاویر دو بعدی ذخیره شده هستند که طبق قاعده خاص و هماهنگ( و شاید مقداری پیش‌پردازش ) روی صفحه نمایش ظاهر می‌شوند. برخلاف محیط های چندرسانه ای ، اشیاء در محیط VR ، طبیعتشان سه‌بعدی است و علاوه بر مختصات طول و عرض ، پارامتر ارتفاع را هم دارند.
- در محیط‌های چندرسانه‌ای کاربر نمی‌تواند آنچه را که وجود دارد ، تغییر بدهد یا چیزی به آن اضافه نماید. مثلاً وقتی که یک فیلم یا انیمیشن را نگاه می‌کنید ، نمی‌توانید زاویه دیدتان را تغییر بدهید و یا صفحه نمایش را بگونه ای بچرخانید که از پشت به تصاویر نگاه کنید.

برخلاف محیط‌های چندرسانه ای ، محیط‌های حقیقت مجازی کاملاً محاوره‌ای است و قابلیت سازگاری و تغییر دائم دارد و کاربر می‌تواند اشیاء محیط را جابجا کند ، یا آنها را از زوایای مختلف نگاه کند و در یک کلام می‌تواند خود را در یک محیط واقعی احساس کند.

در تکنولوژی VR، کاربر می‌تواند با استفاده از کلاههای مخصوصی بنام HMD[1] یا عینکهای ویژه ، خود را دقیقاً در محیط VR احساس نماید؛ در آن بگردد و با آن در تعامل باشد .

### ۱-۶) VRML[2] : زبان مدلسازی حقیقت مجازی

VRML زبانی است که توسط آن ، هر شیی در محیط VR با علائم مخصوص و گرامر خاص یک زبان نشانه‌گذاری[3] بصورت استاندارد ، مدل می‌شود. یعنی VRML قالب استاندارد فایلی است که درون آن اطلاعات مربوط به اشیاء محیط VR ، بصورت متنی تعریف می‌شود.

برای نمایش یک فایل متنی VRML (که در آن مجموعه ای از اشیاء تعریف شده است ) دو مرحله لازم است :

- خوانده شدن فایل متنی VRML ، استخراج تک تک اشیاء و پارامترهای آن و تبدیل آن به یک "ساختمان دادهٔ" مناسب برای تفسیر و نمایش.
  این قسمت از کار را برنامهٔ مرورگر بر عهده دارد.
- تحویل دادن ساختمان دادهٔ تشکیل شده برای اشیاء به برنامه‌ای که آنرا پردازش و تفسیر کرده ونهایتاً نمایش می‌دهد. به این برنامه **"موتور تفسیر"**[4] گفته می‌شود.

تمام جنبه‌های دنیای مجازی و فعل و انفعالاتی که کاربر می‌تواند با این میحط داشته باشد توسط استاندارد VRML تعریف می‌شود. فایلهای VRML را می‌توان با فایلهای HTML مقایسه کرد زیرا هر دو فایلهای متنی و استانداردی هستند که توسط مرورگر خوانده و برای نمایش ، تفسیر می‌شوند.

زبان VRML به این دلیل طراحی شد تا بتوان با فایلهای بسیار کم حجم ، صحنه‌های سه‌بعدی طراحی شده در محیط VR را مستند کرده و آنها را روی شبکهٔ

---

[1] Head Mounted Display
[2] Virtual Reality Modeling Language
[3] Markup Language
[4] Rendering Engine

اینترنت (که فعلاً از محدودیت پهنای باند رنج می‌برد) ارسال کرد. در حقیقت VRML مشخصات لازم جهت تفسیر و نمایش اشیاء هر صفحهٔ VR را مدلسازی کرده و این فایل بر روی شبکه اینترنت انتقال می‌یابد و پس از دریافت آن توسط ماشین کاربر ، تفسیر شده و نمایش داده خواهد شد.

فلسفهٔ وجودی VRML و HTML از یک منبع سرچشمه گرفته است؛ زیرا بعنوان مثال بارشدن یک تصویر دوبعدی کامل از یک صحنه بر روی ماشین کاربر اینترنت که از خط تلفن بعنوان خط ارتباط استفاده می‌کند ، ۲۰ دقیقه طول می‌کشد. در حالی که اگر همان صحنه را توسط VRML مدلسازی نمائیم ، بسته به پیچیدگی اشیاء آن در چند ثانیه بار می‌شود.[1] البته باید این نکته را خاطرنشان کرد که بنابر طبیعت اشیاء در محیط VR ، فایلهای VRML از فایلهای HTML بزرگتر هستند و حجم پردازشی که در ماشین کاربر باید برای تفسیر و نمایش یک فایل VRML انجام شود ، هزاران برابر حجم پردازش برای نمایش یک فایل HTML است. بنابراین کاربر برای آنکه بتواند در حین بار شدن فایل VRML ، اشیاء را در محیط VR بصورت بلادرنگ[2] ببیند ، نیاز به سخت افزار و مرورگر بسیار سریع دارد.[3]

به هر حال VRML هنوز به طور کامل رشد نکرده و حتی به عقیده برخی فعلاً شکست خورده است ولی شاید با گذشت زمان کاربردهای زیبای آن فراگیر شود.[4] بعنوان مثال شاید در آینده شما بتوانید برای خرید یک خانه از طریق اینترنت ، به سایت یک بنگاه معاملاتِ ملکی وصل شوید ، خانه‌ای را انتخاب کنید و با بارکردن فایل VRML آن ، درب خانهٔ مورد نظرتان را باز کنید ، وارد آن شوید ، در اتاقها بچرخید و زوایای مختلف آنرا نگاه کنید و حتی چشم انداز بیرون را تماشا کنید.

---

[1] **به عنوان مثال یک صفحهٔ شطرنج با تمام مهره‌های آن و عملیاتی که کاربر می‌تواند بر روی آن انجام بدهد ، توسط نرم افزار Caligari Pioneer حدود ۷۵ کیلوبایت است که انتقال آن به مرورگر کاربری که با مودم 33.6kbps به اینترنت متصل شده است ، زیر بیست ثانیه طول می‌کشد.**

[2] Real Time

[3] **یکی از این مرورگرها که فایلهای VRML را تفسیر می‌کند ، Live3D نام دارد.**

[4] **دلایل عدم پیشرفت VRML را می‌توان نیاز به تخصص و هنر کافی برای طراحی اشیاء سه‌بعدی و همچنین سخت‌افزار گرانتر عنوان کرد. طراحی یک صحنهٔ VR ، صدها برابر طراحی صفحات HTML ، نیاز به هزینه و وقت دارد که اکثریت نمی‌توانند چنین هزینه‌ای را متقبل شوند.**

### ۶-۲) اصول VRML

در بخش قبل اشاره شد که VRML فایلی است متنی و تبدیل آن به تصویر ، شامل چند مرحله است که در این قمست آنرا بیشتر تشریح خواهیم کرد:

پس از بار شدن یک فایل VRML ، مرورگر آنرا پویش کرده و مجموعه‌ای از ساختمان داده‌ها را در حافظه تولید می‌کند. ( "ساختمان داده" مجموعه‌ای از اطلاعات سازمان‌یافته در حافظهٔ کامپیوتر است که برای هر شیی تشکیل می‌شود.)

پس از تشکیل ساختمان داده‌های لازم ، مرورگر "موتور تفسیر" را برای نمایش اشیاء فرا می‌خواند. موتور تفسیر دو وظیفه برعهده دارد:

- محاسبات و پردازش لازم
- رسم تصاویر

موتور تفسیر را می‌توان پروسه‌ای در نظر گرفت که کارش رسم تصاویر محیط VR با استفاده از ساختمان دادهٔ حاصل از پویش فایل VRML می‌باشد.

همانگونه که اشاره شد گاه حجم محاسبات لازم برای تشکیل و ترسیم تصاویر محیط VR بسیار زیاد است ، بهمین دلیل برای بالا بردن سرعت بخشی از این پردازش به سخت‌افزار کارتهای گرافیکی ویژه محول می‌شود.

موتورهای تفسیر غالباً از مفاهیم شیی‌گرائی[۱] و ذخیرهٔ بُرداری[۲] برای ساختمان داده‌های مربوط به هر شیئ در محیط VR استفاده می‌کنند. برای مشخص شدن مفهوم ذخیره‌سازی بُرداری ، دستگاه مختصات قائم را با یک مرکز فرضی در فضای سه بعدی داشته در نظر بگیرید. به هر نقطه که در فضای سه‌بعدی با مختصات x-y-z بصورت یکتا مشخص می‌شود ، بُردار می‌گوییم. یک پاره خط در محیط VR می‌تواند فقط با دو بردار $(x_0,y_0,z_0)$ و $(x_1,y_1,z_1)$ مشخص شود. برای رسم یک چند ضلعی در فضا به بردارهای مربوط به هر رأس نیاز است. اشیاء شناخته‌شده‌ای مثل استوانه ، کره و مخروط با استفاه از روابط ریاضیشان ، در دستگاه مختصات سه‌بعدی قابل ترسیم هستند. بعنوان مثال برای رسم یک کره در فضای سه بعدی ، دو پارامتر شعاع کره و بُردار مرکز آن نیاز است . موتور تفسیر براساس رابطه ریاضی زیر نقاط سطح کره را محاسبه و ترسیم می‌نماید:

$$(x-x_0)^2+(y-y_0)^2+(z-z_0)^2=R^2$$

---

۱ Object Oriented
۲ Vector

هر شیی خـواه سـاده خـواه پیچیـده می‌توانـد براسـاس اشـیاء سـاده‌تر یـا مجموعـه‌ای از بردارها توصیف شود. بـرای آنکـه اشـیاء بتواننـد تغییـر کننـد می‌تـوان عملیـات زیـر را روی آنها تعریف کـرد :

- **عمـل مقیـاس (Scaling)** : ایـن عمـل بـا یـک ضریـب شـیئی را در جـهت محورهـای مختصات منقبض یا منبسط می‌کند. عمـل مقیـاس ، بـا رابطـهٔ ریـاضی زیـر مـدل می‌شـود:

$$(x, y, z) \xrightarrow{Scaling} (\alpha x, \beta y, \gamma z)$$

اگر $\alpha$ ، $\beta$ یا $\gamma$ بزرگتر از ۱ باشند ، شیئ منبسط و اگر کوچکتر از ۱ باشد منقبض می‌شود.

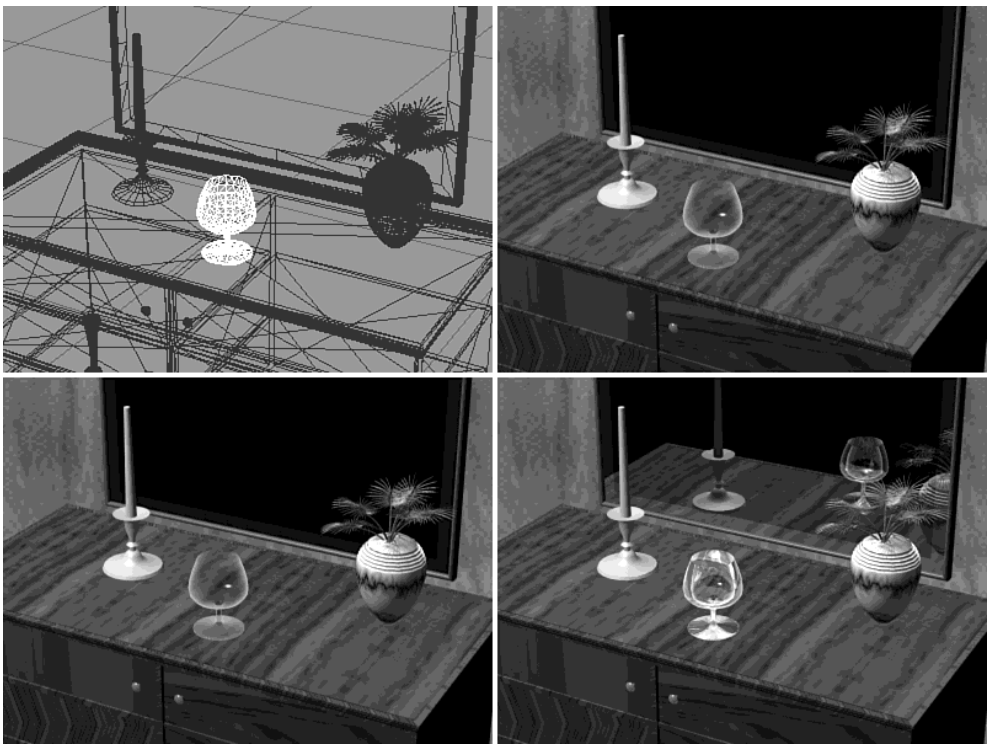
- **عمل چرخش (Rotation)** : چرخـش یـک شـیئ حـول یـک بـردار
- **عمل انتقال (Translation)** : انتقال یـک شـیئ از یـک مکـان بـه مکـان جدیـد.

حرکـات یـا تغییـرات پیچیـدهٔ یـک شـیئ ، می‌توانـد بـر اسـاس ترکیبـات سـه عمـل اساسی بـالا توصیـف شـود. ایـن عملیـات بـر اسـاس محاسـبات ضـرب ماتریسـی (بعبـارت عام‌تر ، محاسبات جـبر خطـی ) توسـط CPU انجـام می‌شـود.

عملیاتی که ذکر شد بصـورت مجـزا بـر روی تـک تـک اشـیاء یـک صحنـه قـابل انجـام است ، ولی نکتهٔ اساسی اثـرات و تعـاملاتی اسـت کـه اشـیاء یـک صحنـه بـر روی یکدیگـر خواهند داشت. مثلاً با نگاه از روبرو بـه کتـابی کـه در جلـو یـک مـداد قـرار گرفتـه اسـت ، چون مداد پشـت کتـاب پنـهان شـده ، نبـاید از ایـن زاویـه دیـده شـود؛ ولیکـن اگـر کـاربر صحنه را بچرخاند ، مداد دیـده خواهـد شـد. یـک لیـوان یـا یـک جسـم شـفاف روی یـک صحنه ، باید اشیاء پشت خـود را نشـان دهـد. یـک آینـه بـاید تصویـر اشـیاء روبـروی خـود را نشـان بدهـد. در شـکل (۲۲-۱۰) تعـامل اشـیاء بـا یکدیگـر در محیـط VR نشـان داده شده اسـت.

موتور تفسیر برای ترسیم هر شـیئ بـه پردازشـهای زیـر نیـاز دارد:

- بررسی موقعیت یک شیئ نسـبت بـه اشـیاء دیگـر
- محاسـبهٔ درخشـندگـی شـیئ : موتـور تفسـیر بـاید شـدت درخشـندگـی هـر نقطـه در تصویـر و تغییـرات در مقـدار روشـنائی یـک شـیئ را در اثـر مجـاورت بـا اشـیاء دیگـر ، محاسبه نمـاید.
- محاسـبه سـطوح قـابل رؤیـت : در هنگـام ترسـیم شـیئ بـاید فقـط سـطوحی نشـان داده شوند که از زاویهٔ دیـدِ فعلـی ، قـابل رؤیـت هسـتند. محاسـبهٔ سـطوح قـابل رؤیـت برعـهدهٔ موتور تفسیر اسـت .



شکل (۲۲-۱۰) تعامل اشیاء با یکدیگر در محیط VR

- محاسبهٔ پرسپکتیو شیئ : با توجه به آنکه صفحه نمایش تخت و دو بعدی است ولی تصاویر توصیف شده در فایل VRML ذاتاً سه بعدی هستند ، موتور تفسیر باید به نحوی مختصهٔz از هر شیئ را روی صفحهٔ دو بعدی x-y بنگارد ، تا جسم دوبعدی توسط چشم ، سه‌بعدی تجسم شود .

محاسبات ذکر شده فقط شمّه‌ای از عملیات لازم بر روی اشیاء است. پردازشهای متنوع دیگری هم وجود دارند تا محیط VR را طبیعی‌تر جلوه دهند. البته این محاسبات بسیار وقتگیر هستند.

### ۶-۳) ساختار یک فایل VRML

فایلهای VRML در حقیقت مجموعه‌ای از اشیاء تعریف شده به زبان VRML (در محیط گرافیک سه بعدی) هستند و همانند فایلهای HTM ذاتاً فایلهای متنی سادهٔ ASCII محسوب می‌شوند که براحتی در یک ویرایشگر متن ، قابل ایجاد و ویرایش

هستند. فایلهای VRML پس از ایجاد ، معمولاً با پسوند wrl. ذخیره می‌شوند. (پسوندهای vrml. یا vrm. هم استفاده شده است .)

وقتی مرورگر یک فایل را از سرویس دهندهٔ وب بار می‌کند ، ابتدا نوع فایل و چگونگی تفسیر آنرا تعیین می‌کند. انتقال این فایل دقیقاً مبتنی بر پروتکل HTTP بوده و هیچ تفاوتی با انتقال فایلهای HTML ندارد. تنها تفاوت در نوع مرورگری است که موظف به تفسیر اینگونه فایلهاست. نسخه‌های جدید مرورگر Netscape بصورت داخلی قادرند فایلهای VRML را پویش کرده و تفسیر نمایند ولی مرورگرهای فعلی مایکروسافت (مثل IE5.0) نیاز به "برنامه‌های اتصالی"[1] دارند.

یک فایل VRML ازچهار قسمت تشکیل شده است :

- **خطِ سرآیندِ فایل** : اولین خط در هر فایل VRML ، با علامت # شروع می‌شود و مشخصاتِ فایل را تعیین می‌کند. این خط قالبی همانند مثال زیر دارد:

#VRML V1.0 ascii

یعنی محتوای فایل از نوع VRML و با کدهای استاندارد ASCII است و مطابق با نسخهٔ 1.0 از زبان VRML تدوین شده است. یا :

#VRML V2.0 uttf2

محتوای فایل از نوع VRML و با کدهای ASCII (شامل کدهای بالاتر از ۱۲۸) است و مطابق با نسخهٔ 2.0 از زبان VRML تدوین شده است.

- **خطوطِ توضیح** : به غیر از خطِ اول از فایل که با علامت # شروع می‌شود ، هر خطی که در متن فایل با این علامت شروع شود به عنوان خط توضیح تلقی شده و نادیده انگاشته می‌شود.

- **گره‌ها**[2] : یک فایل VRML از عناصری تشکیل شده است که به آن "گره" گفته می‌شود. گره می‌تواند معرف یک شیئ ساده یا یک شیئ پیچیدهٔ سه‌بعدی باشد. یک گره در فایل VRML بصورت زیر نمایش داده می‌شود:

**NodeType { Fields }**

---

[1] Plug-ins
[2] Nodes

➢ **فیلدها** : در تعریف یک گره به صورت بالا ، مقادیری که بین {...} قرار می‌گیرد ، توصیف آن گره است و چگونگی تفسیر یا نمایش یک شیئ را تعیین می‌نماید. به مثال زیر دقت نمایید:

| | |
|---|---|
| خط سرآیند | #VRML V1.0 ascii |
| خط توضیح | #This file defines a simple red sphere |
| یک گره (شیئ کلی صحنه ) | Separator { |
| یک گره (تعریف رنگ پوستهٔ اشیاء ) | Material { diffuseColor 1 0 0 } #The color red |
| یک گره (تعریف کره‌ای با شعاع ۱ در مرکز) | Sphere { } |
| پایان تعریف گره (پایان تعریف شیئ صحنه ) | } |

اگر به دنیای واقعی اطرافتان نگاه کنید ، مجموعه‌ای از اشیاء سه‌بعدی با مشخصات خاص خود را می‌بینید. این اشیاء معمولاً ساختاری پیچیده دارند و از اشیاء ساده و کوچکتر تشکیل شده‌اند. مثلاً یک میز را به عنوان شیئی مستقل در فضای سه‌بُعدی در نظر بگیرید. این شیئ را می‌توان به پنج شیئ کوچکتر تقسیم کرد ( ٤ پایه و یک سطح ). اگر میز را **"شیئ پدر"** در نظر بگیریم ، اجزای کوچکتر آن **"فرزند"**[۱] تلقی می‌شوند. اشیاء فرزند می‌توانند خصوصیات پدر را به ارث ببرند یا آنکه خصوصیاتی متمایز از پدر خود داشته باشند. مثلاً پایهٔ میز به عنوان فرزند می‌تواند رنگ خود را از شیئ پدر به ارث ببرد. اشیاء فرزند می‌توانند خودشان دارای اجزای کوچکتری به عنوان فرزند باشند. هرگونه تغییر در موقعیت شیئ پدر باید تمام فرزندان را تحت تاثیر قرار بدهد.

VRML دقیقاً مبتنی بر مفهوم شیئ‌گرایی[۲] و ارث‌بری[۳] است. برای شروع ، به تعریف بدن انسان در VRML دقت کنید:

```
#VRML V1.0 ascii
# A simple humanoid body made up of cylinders and spheres
# Built by hand with a text editor!
```

[۱] Child
[۲] Object Oriented Concept
[۳] Inheritance

```
DEF Body Separator {
    DEF Pelvis Separator {
      DEF Chest Separator {
        DEF LeftUpperArm Separator {
          DEF LowerArm Separator {
          #تعریف اجزا در اینجا می‌آید
          }
        }
        DEF RightUpperArm Separator {
          DEF LowerArm Separator {
          #تعریف اجزا در اینجا می‌آید
          }
        }
        DEF Head Separator {
          DEF Skull Separator {
          #تعریف اجزای جمجمه در اینجا می‌آید
          }
          DEF LeftEye Separator {
          #تعریف اجزای چشم چپ در اینجا می‌آید
          }
          DEF RightEye Separator {
          #تعریف اجزای چشم راست در اینجا می‌آید
          }
          DEF Nose Separator {
          #تعریف اجزای بینی در اینجا می‌آید
          }
        }
        DEF LeftThigh Separator {
          DEF Calf Separator {
          #تعریف اجزای ساق پای چپ در اینجا می‌آید
          }
        }
        DEF RightThigh Separator {
          DEF Calf Separator {
          #تعریف اجزای ساق پای راست در اینجا می‌آید
          }
        } } }
```

در این مثال اگر شیئ "بدن" را به عنوان **"پدر"** تلقی کنیم ، اشیایی مثل دست ، سر ، بازو و ... فرزندان آن محسوب می‌شوند. هر شیئ فرزند مثل **"دست"** خود شامل اشیاء فرزند مثل انگشتان ، بازو و ساعد است. هر گونه تغییر و تبدیلِ شیئ پدر (بدن) باید به اشیاء فرزند نیز اعمال شود. به عنوان مثال حرکت بدن به سمت جلو یا چرخش آن باید منجر به حرکت یا چرخش یکایک اجزای آن نیز بشود. ( در مثال بالا تعریف جزئیات هر شیئ فرزند نیامده است.)

قالب یک شیئ در VRML با گره **Separator** تعریف می‌شود. در تعریف گره‌ها با Separator مشخصه‌های زیر می‌تواند روی نحوهٔ نمایش اشیاء در صحنهٔ VR تاثیر بگذارد:

**Separator** {

*Transformations Definition*

*Surface properties*

*Shapes*

*Children*

}

اشیاء فرزند نیز دقیقاً همانند پدر خود با ساختار Separator تعریف می‌شوند و هرگاه مشخصاتی مثل تبدیل یا رنگ سطح آن تعریف نشود ، آنرا از پدر خود به ارث خواهند برد.

هر فایل VRML دارای یک شیئ کلی به عنوان پدر تمام اشیاء موجود در صحنهٔ VR می‌باشد و هر شیئ دیگر در درون آن تعریف می‌شود. به عبارت دیگر این شیئ ، **ظرفی**[۱] برای بقیهٔ اشیاء محسوب می‌شود. به یک مثال ساده دقت کنید :

```
Separator {
Material { diffuseColor 0 1 0 }
Transform { translation 2 5 7 }
Cone { }
Material { diffuseColor 0 1 0 }
Transform { translation 2 5 7 }
Sphere { }
}
```

[۱] Container

در این مثال در شیئ پدر که شامل کل صحنهٔ VR می‌باشد ، ابتدا رنگ سطح اشیاء ، سبز (R=0,G=1,B=0 ) نظر گرفته می‌شود.[1] ( با گره { }Material ) . سپس یک عمل انتقال به اندازهٔ (x=2,y=5,z=7) تعریف می‌شود. تمام اشیاء بعدی نسبت به موقعیت جدید ترسیم خواهند شد؛ ( با گره { } Transform ). در ادامه ، یک مخروط با مشخصات پیش‌فرض تعریف شده است؛ ( با گره { }Cone ). بعد از آن رنگ سطح اشیاء ، به قرمز (R=1,G=0,B=0 ) تبدیل شده و یک عمل انتقال جدید تعریف می‌شود. تمام اشیاء بعدی نسبت به موقعیت جدید ترسیم می‌شوند ، سپس یک کُره با مشخصات پیش‌فرض تعریف شده است؛ ( با گره { }Sphere ).

گره { } Material اثر قبلی خود را **باطل**[2] می‌کند ، در حالی که عمل انتقال بعدی توسط گره { }Transform ، نسبت به انتقال قبلی محاسبه می‌شود؛ یعنی کُرهٔ قرمزرنگ به مرکز (x=8+2,y=9+5,z=17) نمایش خواهد یافت. (x=10,y=14,z=17)

هرگاه یک شیئ درون شیئی دیگر تعریف شود ، فرزند آن محسوب می‌شود در حالی که اگر بیرون از آن تعریف گردد ، مستقل از دیگر فرزندان خواهد بود. مثال :

| | |
|---|---|
| شیئ پدر | Separator { |
| شیئ فرزند | Separator { |
| شیئ فرزندِ فرزند ( نوه ) | Separator { |
| | } |
| | } |
| | } |

| | |
|---|---|
| شیئ پدر | Separator { |
| شیئ فرزند | Separator { |
| | } |
| شیئ فرزند | Separator { |
| | } |
| | } |

[1] **رنگ در VRML با سه مولفهٔ RED/GREEN/BLUE تعریف می‌شود. هر یک از مقادیر RGB عددی اعشاری بین صفر و یک هستند.**

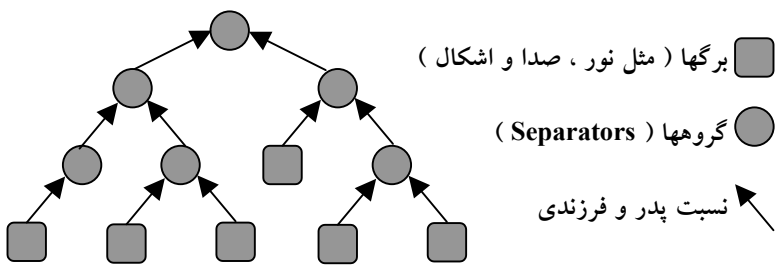
[2] Override

در VRML لازم نیست تمام اشیاء دارای ظاهر گرافیکی یا هندسی باشند بلکه یک شیئ ممکن است فقط برای آن بوجود بیاید که دو شیئ دیگر را به عنوان اشیاء فرزند خود ، در یک گروه قرار بدهد.

بنیان ساختمان دادهٔ اشیاء در VRML درخت معکوس است که سلسله‌مراتبی از گره‌ها را تعریف می‌کند. در شکل (۲۳-۱۰) این ساختار سلسله‌مراتبی به تصویر کشیده شده است. دو نوع اصلی شیئ وجود دارد :

- **برگ (Leaf)**
- **گروه (Group)**

هر گروه از گره‌ها می‌تواند شامل برگ و گروههای دیگری باشد. برگها معمولاً مطابق با ترتیب اشیایی است که شما آنها را در فضای سه‌بعدی می‌بینید. ( نظیر اشکال ، صدا ، نور و ... )




شکل (۲۳-۱۰) ساختار سلسله‌مراتبی اشیاء در محیط VR

تعریف اشیاء در VRML تحت تاثیر مستقیم تجربهٔ شما از جهان مجازی است؛ هر چیز دیدنی یا شنیدنی مثل یک میز ، صندلی یا صدای باران بوسیلهٔ گره نشان داده می‌شود. صدای یک ساعت بوسیلهٔ **گره صدا**[1] ایجاد می‌شود و در یک منظرهٔ دیدنی ، یک یا چند گره مولد مجازی نور ، صحنه را نورپردازی می‌کند. شما وقتی در دنیای VRML هستید نمی‌توانید گره **گروه** را ببینید ، اما آنها وجود دارند و بر موقعیت و نحوهٔ مشاهدهٔ گره‌های برگ در یک درخت تاثیر می‌گذارند. توجه کنید که در VRML 2.0 فرزندان همسطح معمولاً بی‌ارتباط با یکدیگر هستند. در این نسخه از زبان VRML حدود ٥٤ نوع متفاوت از گره‌ها وجود دارد. البته امکان تعریف

[1] Sound Node

گره‌های جدید به صورت **الگو**[1] وجود دارد که هر یک از این گره‌ها برای مشخص کردن یک شیئ خاص است که برای ایجاد یک صحنه در چند جا باید ظاهر شود.

هر گره دارای مجموعه‌ای از فیلدهای مقدار است؛ برای مثال یک گره از نور ، فیلدی دارد که شدت نور را مشخص می‌کند و اگر ارزش آن فیلد را تغییر بدهید ، شدت روشنایی نور صحنه و اشیایی که در معرض آن هستند ، تغییر خواهد کرد.

### ۴-۶) پیاده‌سازی گره‌ها در VRML

بار دیگر ساختار یک گره را بررسی می‌کنیم :

**Separator** {

*Transformations Definition*

*Surface properties*

*Shapes*

*Children*

}

اگر در یک گره هیچ **شکل**[2] **هندسی** تعریف نشده باشد ، هیچگونه نمود ظاهری نخواهد داشت. در ادامه فیلدهای یک گره را بررسی می‌کنیم:

#### ۱-۴-۶) تبدیلات

در یک گره فیلدِ {..} Transform ، مجموعهٔ تبدیلاتی را تعریف می‌کند که بر یک شیئ و فرزندان آن اعمال می‌شود. این مجموعهٔ تبدیلات می‌تواند بصورت زیر تعریف شود:

```
Transform  {
        Center  x  y z
        scalingFactor  x  y  z
        scaleOrientation x y z  a
        rotation  x  y  z  a
        translation  x  y  z
}
```

---

[1] Prototype
[2] Shape

هر فیلد بر روی خط مجزا قرار می‌گیرد و رعایت ترتیب نوشتن آنها مهم نیست زیرا اجرای گرهٔ Transform توسط "موتور تفسیر" به ترتیب زیر خواهد بود :

۱. مقیاس (Scaling)

۲. چرخش (Rotation)

۳. انتقال (Translation)

قالب بالا تمام تبدیلات را در برمی‌گیرد و هر گاه به یک یا تعدادی از آنها نیاز نبود ، فیلد مربوطه حذف می‌شود. مثال :

```
Transform {                          Transform {
        Center 3 1 2                         translation 4 0 3
        rotation 4 0 0 0.31          }
        }
```

- **Center** : نقطه‌ای را در فضای سه‌بُعدی تعریف می‌کند که عملیاتی نظیر چرخش اشیاء حول آن انجام خواهد شد.
- **scalingFactor** : ضرائب انقباض یا انبساط اشیاء را در راستای سه محور مختصات تعریف می‌نماید.
- **scaleOrientation** : با این تبدیل ابتدا محور مختصات به اندازهٔ a حول مرکز پیش‌فرض می‌چرخد و سپس در سه محور مختصات ، با ضرائب $x\ y\ z$ منقبض یا منبسط می‌شود.
- **rotation** : این تبدیل اشیاء را به اندازهٔ a رادیان حول بردار $(x\ y\ z)$ می‌چرخاند.

مثال زیر یک چرخش ۴۵ درجه‌ای حول محورy و سپس یک انتقال به اندازهٔ نیم واحد[۱] (۰/۵ متر) ، روی محور x تعریف می‌کند؛ ( دقت کنید زاویه بطور تقریبی بر حسب رادیان نوشته شده است ) :

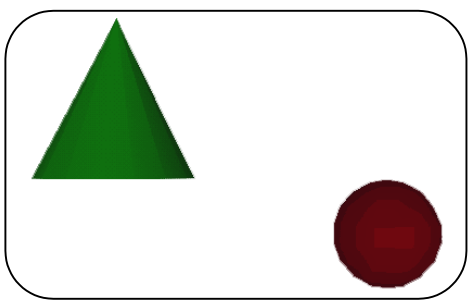
```
Transform {
        rotation 0 1 0 0.78541
        translation 0.5 0 0

}
```

در مثالهای بَعد نمایش واقعی از اشیائی را که تحت تاثیر "گره تبدیل" قرار گرفته‌اند ، ملاحظه می‌کنید:

---

[۱] معمولاً واحد طول در VRML ، متر است که برای نمایش در یک پنجره نرمالیزه می‌شود.

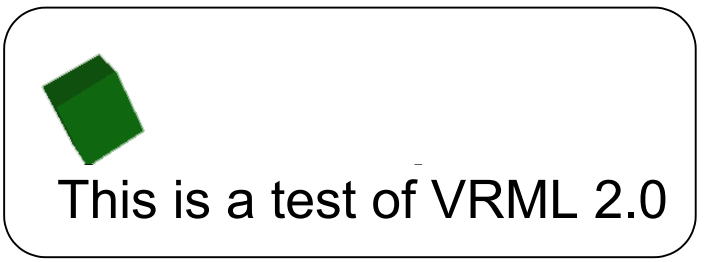
```
#VRML V1.0 ascii
Separator {
        Separator {
          Material { diffuseColor 0 1 0 }
          Transform { translation 2 5 7
                      rotation 1 0 0 0.78
                    }
        Cone { }
        }
        Separator {
          Material { diffuseColor 1 0 0 }
          Transform { translation 6 3 2
                    }
          Sphere { }
}
}
```



شکل (۲۴-۱۰) تاثیر تبدیلات بر روی اشیاء در محیط VR

```
#VRML V1.0 ascii
Separator {
        Separator {
          Material { diffuseColor 0 1 0 }
          Transform { translation 15 15 19 }
          Transform { rotation 2 2 2 0.78 }
          Cube {
                width 10
                 height 10
                 depth 10 }
```

```
    }
    Separator {
      AsciiText {
            String [" This is a test of VRML 2.0"]
             }
} }
```




شکل (۲۵-۱۰) تاثیر تبدیلات بر روی اشیاء در محیط VR

**۲-۴-۶) اشکال هندسی در VRML**

در VRML براحتی می‌توان اشکال شناخته شدهٔ هندسی را تعریف کرد. برخی از این اشیاء را معرفی می‌کنیم:

- **کره** : ایجاد کره بسیار ساده است و فقط نیاز به تعریف یک فیلد دارد:

```
Sphere {
        radius r                    شعاع کره بر حسب متر
        }
```

کره به مرکز پیش‌فرض ، ترسیم خواهد شد و اگر شعاع کره مشخص نشود مقدار آن ۱ فرض خواهد شد.

- **مخروط** :

```
Cone {
        bottomRadius r              شعاع قاعدهٔ مخروط بر حسب متر
        height h                    ارتفاع مخروط بر حسب متر
        }
```

اگر هر یک از فیلدها تعریف نشود ، مقادیر پیش‌فرض آنها به شرح ذیل است:

شعاع قاعده : ۱ متر  ارتفاع مخروط : ۲ متر  مرکز ترسیم : مرکز پیش‌فرض

➢ **استوانه :**

```
Cylinder {
        bottomRadius  r          شعاع قاعدهٔ استوانه بر حسب متر
        height  h                ارتفاع استوانه بر حسب متر
        }
```

➢ **مکعب :**

```
Cube {
        width w                  طول مکعب بر حسب متر
        height h                 ارتفاع مکعب بر حسب متر
        depth  d                 عمق مکعب بر حسب متر
    }
```

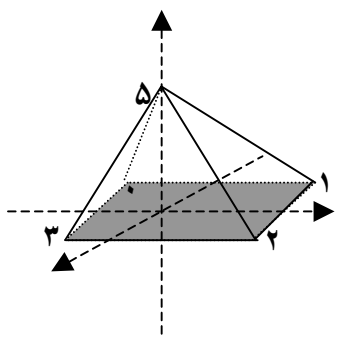
➢ **متن :**

```
AsciiText {
        string  s                رشتهٔ متنی مورد نظر برای نمایش
        spacing sp               ضریب فاصلهٔ عمودی سطرهای متن از هم
        justification  j         موقعیت نمایش متن ( وسط ، چپ یا راست )
    }
```

➢ **اشکال هندسی مرکب** : گره‌های هندسی که تا اینجا بررسی شد اشکال بسیار ساده‌ای هستند که بوسیلهٔ روابط ریاضیشان توسط موتور تفسیر ، رسم می‌شوند ولی برای تولید اشکال سه‌بعدی پیچیده کافی نیستند. VRML گره‌های زیر را برای ایجاد اشکال دلخواه ارائه کرده است:

♦ **Coordinate3** : اولین گام در خلق یک شکل سه‌بعدی هندسی ( بدون انحنا ) ، تعیین مختصات تمام رئوس آن می‌باشد. این کار با گره Coordinate3 انجام می‌شود. به عنوان مثال فرض کنید که می‌خواهیم یک هرم بوجود بیاوریم. ابتدا باید مختصات پنج راس آن تعیین شود:

```
Coordinate3  {
Point  [
-1  0 –1,
1  0 –1,
1  0 1,
-1  0 –1,
0  1 0
]
}
```



♦ **IndexFaceSet** : در دومین گام باید وجوه[1] جسم ، تعریف شوند. اینکار با تعیین رئوسی که یک سطح از جسم را تشکیل می‌دهند ، توسط گره IndexFaceSet انجام می‌شود. ساختار این گره بصورت زیر است :

```
IndexFaceSet {
        CoordIndex [
                V0,V1,V3,...,Vn,-1,
                V0,V1,V3,...,Vn,-1,
                V0,V1,V3,...,Vn,-1,
                V0,V1,V3,...,Vn,-1,
                ]
}
```

این گره با توجه به گره Coordinate3 ، **"اندیس گره‌هایی"** که یکی از وجوه شکل را مشخص می‌کنند ، تعیین می‌نماید. برای هرم بالا اندیس گره‌ها بصورت زیر خواهد بود :

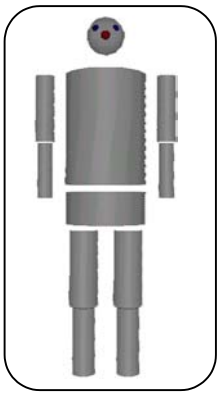
```
IndexFaceSet {
        CoordIndex [
                0,4,1,-1,
                1,4,2,-1,
                2,4,3,-1,
                3,4,0,-1,
                0,1,2,3,-1
                ]
}
```

دقت کنید که ۱- علامت ختم یکی از وجوه و شروع وجه بعدی از شکل هندسی است. مثلاً قاعدهٔ هرم بالا ، در برگیرندهٔ رئوس با اندیس ( ۰ ، ۱ ، ۲ ، ۳ ) است و ۱- پایان رئوس این وجه را اعلام می‌نماید. مجموعهٔ اندیسها در درون علائم [ ] تعریف می‌شوند و هر یک از اندیسها با علامت **و** از هم جدا می‌شوند.

برای روشنتر شدن تعریف اشیاء فوق در فایل VRML ، در ادامه با مثالی سعی شده است تا از طریق اشکال هندسی ساده مثل استوانه و کره ، ساختار نمادین زیر از بدن یک انسان مدلسازی شود:

---

[1] Face



```
#VRML  V1.0  ascii
# A simple humanoid body made up of cylinders and spheres
# Built by hand with a text editor!

DEF  Body  Separator  {
        DEF  Pelvis  Separator  {
        Cylinder { radius 0.2 height 0.15 }
            DEF  Chest  Separator {
            Transform { translation 0 0.350  0 }
            Cylinder { radius 0.2 height 0.50 }
               DEF  LeftUpperArm  Separator {
               Transform { translation -0.275 0.10 0 }
               Cylinder { radius 0.05 height 0.30 }
                  DEF LowerArm Separator {
                  Transform { translation 0 -0.280 0 }
                  Cylinder { radius 0.04 height 0.25 }
          }
            }
             DEF  RightUpperArm  Separator  {
             Transform { translation 0.275 0.10 0 }
             Cylinder { radius 0.05 height 0.30 }
                  DEF LowerArm Separator {
                  Transform { translation 0 -0.280 0 }
                  Cylinder { radius 0.04 height 0.250 }
                  }
             }
            DEF  Head Separator {
             Transform { translation 0 0.425 0 }
                DEF Skull Separator {
                Sphere { radius 0.1 }
                }
```

```
                DEF LeftEye Separator {
                Transform { translation -0.04 0.025 0.0707 }
                 aterial   { diffuseColor  0  0  1  }
                Sphere { radius 0.02 }
                 }
                DEF RightEye Separator {
                Transform { translation 0.04 0.025 0.0707 }
               Material  { diffuseColor 0 0 1 }
               Sphere { radius 0.02 }
                }
          DEF Nose Separator {
            Transform { translation 0 -0.01 0.1 }
               Material   { diffuseColor  1  0  0  }
           Sphere { radius 0.02 }
        }
           }
      }
        DEF LeftThigh Separator {
         Transform { translation -0.1 -0.275 0 }
         Cylinder { radius 0.075 height 0.35 }
              DEF Calf Separator {
              Transform { translation 0 -0.33 0 }
              Cylinder { radius 0.06 height 0.30 }
           }
         }
        DEF RightThigh Separator {
         Transform { translation 0.1 -0.275 0 }
         Cylinder { radius 0.075 height 0.35 }
            DEF Calf Separator {
            Transform { translation 0 -0.33 0 }
            Cylinder { radius 0.06 height 0.30 }
          }
          }
          }
}
```

### ۳-۴-۶) نور و صدا و رنگ در VRML

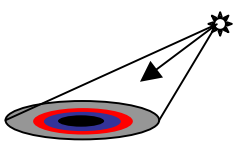
VRML از سه نوع منبع نور برای نورپردازی صحنه‌ها پشتیبانی می‌کند. منابع نور مجموعه‌ای از فیلدهای اصلی نظیر "شدت نور" ، فیلدی جهت روشن یا خاموش کردن منبع نور و فیلدِ رنگ نور دارند که انعطاف لازم را جهت نورپردازی ماهرانهٔ صحنهٔ VR ، در اختیار طراح گذاشته‌اند. برای زیبایی هرچه بیشتر ، باید منابع نور با دقت و ظرافت به صحنه اضافه شود.

- **نورافکن نقطه‌ای**[1] : این منبع نور ، پرتوهای نوری را در تمام جهات به صحنهٔ VR می‌تاباند ( همانند یک لامپ معلق و آویزان ) :

```
PointLight {
On TRUE / FALSE          روشن یا خاموش بودن منبع نور
intensity x              شدت منبع نور (عددی بین صفر –خاموش– و ۱/۰ –روشن–)
color  r g b             رنگ منبع نور بر حسب سه مولفهٔ قرمز/سبز/آبی
location x y z           مختصات موقعیت منبع نور
}
```

- **نورافکن صحنه**[2] : این منبع نور ، پرتوهای نوری را در حالت مخروطی شکل به صحنهٔ VR می‌تاباند تا بخشی از صحنه بطور خاص نورپردازی شود :

```
SpotLight {
On TRUE / FALSE          روشن یا خاموش بودن منبع نور
intensity x              شدت منبع نور (عددی بین صفر –خاموش– و ۱/۰ –روشن–)
color  r g b             رنگ منبع نور بر حسب سه مولفهٔ قرمز/سبز/آبی
location x y z           مختصات موقعیت منبع نور
direction x y z          بردار جهت نور
dropOffRate  dor         میزان تفرق نور از مرکز به سمت لبه‌های قاعدهٔ مخروط
cutOffAngle  cof         زاویه‌ای که تحت آن زاویه ، نور به صحنه تابانیده می‌شود.
```



```
}
```

- **نورافکن مستقیم**[3] : این منبع نور ، پرتوهای نوری را به موازات یک بردار به صحنهٔ VR می‌تاباند :

```
DirectionalLight {
On TRUE / FALSE          روشن یا خاموش بودن منبع نور
intensity x              شدت منبع نور (عددی بین صفر –خاموش– و ۱/۰ –روشن–)
color  r g b             رنگ منبع نور بر حسب سه مولفهٔ قرمز/سبز/آبی
direction x y z          بردار جهت نور
}
```

---

[1] Point Light
[2] Spot Light
[3] Directionl

- برای رنگ‌آمیزی سطوح اشیاء در VRML می‌توان از گره {} Material استفاده کرد:

```
Material {
        ambientColor  r  g  b
        diffuseColor   r  g  b
        specularColor r  g  b
        emissiveColor r  g  b
        shininess  s
        transparency t
        }
```

با گرهٔ Material علاوه بر رنگِ سطوح ، می‌توان ویژگیهای درخشندگی ، شفافیت و خصوصیات دیگر سطح اشکال را تعریف کرد. در گرهٔ Material می‌توان از فیلد emissiveColor برای نمایش اشیاء مشتعل ، فیلد specularColor برای اشیاء با درخشش متالیک (فلزگونه) و از فیلد ambientColor برای اجسامی که باید کسری از نور صحنه را مثل یک جسم صیقلی منعکس کنند ، استفاده کرد.

- Fog : گره‌ای برای کدر و مات کردن رنگ صحنه در محیط VR است.
- Background : با این گره می‌توان پس‌زمینهٔ صحنه را طراحی و رنگ‌آمیزی کرد. مانند طراحی پس‌زمینهٔ آسمان ، زمین ، جنگل و ...

یکی از قابلیتهای مهمی که به VRML 2.0 اضافه شد ، پشتیبانی از منابع صدا بود که با دو گرهٔ Audioclip و Sound در اختیار طراح قرار گرفت. گره Sound شبیه نورافکن صحنه عمل می‌کند یعنی صدا با استفاده از فیلد شدتِ صدا تحت کنترل طراح می‌باشد. این گره دارای فیلدهای مکان ، بردار جهت و شدتِ صدا می‌باشد.

گرهٔ Audioclip URL محل قرار گرفتن فایل صدا را در شبکهٔ اینترنت مشخص می‌کند. ( در VRML 2.0 فایلهای MIDI یا فایلهای WAV پشتیبانی می‌شوند.)

هدف از این فصل آشنایی با اصول کلی WEB ، HTTP ، HTML و VRML بوده است؛ مباحث و جزئیات هر یک از مقولات بررسی شده بسیار وسیع و پرحجم هستند و برای آشنایی دقیق و کاربردی باید به مراجع آنها مراجعه کرد.

## ۷ ) مراجع این فصل

مجموعهٔ مراجع زیر می‌توانند برای دست آوردن جزییات دقیق و تحقیق جامع در مورد مفاهیم معرفی شده در این فصل مفید واقع شوند.


| | |
|---|---|
| "Computer Networks" , Andrew S.Tanenbaum, Third Edition, Prentice-Hall, 1996. | |
| "Internet and Intranet Engineering", Daniel Minoli , McGraw-Hill, NewYork 1997. | |
| "Internetworking with TCP/IP", Commer D.E. Prentice-Hall, 1996. | |
| "Special Edition Using VRML",Atephen Matsuba , Bernie Roehl, 1998. | |
| "CGI Programming Unleashed , Eugene Eric Kim , 1996 by Sams.net Publishing. | |
| "HTML 3.2 and CGI", Professional Reference Edition,UNLEASHED , John December and Mark Ginsburg, Sams.net Publishing. | |
| "CGI Manual of Style" , Robert McDaniel, ISBN: 1-56276-397-0. | |
| "Teach yourself VRML 2.0 in 21 days", Chris Arrin Bruse Campbell, 1997. | |
| RFC1945 | Hypertext Transfer Protocol -- HTTP/1.0. T. Berners-Lee, R. Fielding & H. Frystyk. May 1996 |
| RFC2068 | Hypertext Transfer Protocol -- HTTP/1.1. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee. January 1997 |
| Microsoft Internet URL : http://www.microsoft.com/internet | |
| The VRML Repository URL : http://www.sdsc.edu/vrml/ | |
| The HTML Guru URL : http://members.aol.com/htmlguru/index.html | |
| Tim Berners-Lee's Style Guide URL : http://www.w3.org/hypertext/WWW/Provider/Style/Overview.html | |
| The Home of the WWW Consortium URL : http://www.w3.org/ | |
| The World Wide Web FAQ URL : http://www.boutell.com/faq/ | |
| The 24-Hour HTML Café URL : http://www.mcp.com/sams/books/235-8/ | |
| The Developer's JumpStation URL : http://oneworld.wa.com/htmldev/devpage/dev-page.html | |

## ۱) مقدمه

تا یکی دو دهه قبل شبکه های کامپیوتری معمولاً در دو محیط وجود خارجی داشت:

- محیطهای نظامی که طبق آئین نامه های حفاظتیِ ویژه بصورت فیزیکی حراست می‌شد و چون سایتهای ارتباطی خودشان هم در محیط حفاظت شدهٔ نظامی مستقر بود و هیچ ارتباط مستقیم با دنیای خارج نداشتند ، لذا دغدغه کمتری برای حفظ اسرار و اطلاعات وجود داشت. (نمونه بارز این شبکه ARPANET در وزارت دفاع آمریکا بود)
- محیطهای علمی و دانشگاهی که برای مبادله دستاوردهای تحقیقی و دسترسی به اطلاعات علمی از شبکه استفاده می‌کردند و معمولاً بر روی چنین شبکه هائی اطلاعاتی مبادله می‌شد که آشکار شدن آنها لطمه چندانی به کسی وارد نمی‌کرد. (اداراتی هم که اطلاعات محرمانه و سری داشتند معمولاً از کامپیوترهای Mainframe استفاده می‌کردند که هم مدیریت و حراست ساده تری نیاز دارد و هم کنترل کاربران آن بصورت فیزیکی ساده است)

با گسترش روز افزون شبکه های بهم پیوسته و ازدیاد حجم اطلاعات مورد مبادله و متکی شدن قسمت زیادی از امور روزمره به شبکه های کامپیوتری و ایجاد شبکه های جهانی چالش بزرگی برای صاحبان اطلاعات پدید آمده است . امروزه سرقت دانشی که برای آن هزینه و وقت ، صرف شده یکی از خطرات بالقوه شبکه‌های کامپیوتری به شمار می‌آید .

در جهان امروز با محوّل شدن امور اداری و مالی به شبکه های کامپیوتری زنگ خطر برای تمام مردم به صدا در آمده است و بر خلاف گذشته که خطراتی نظیر دزدی و راهزنی معمولاً توسط افراد کم سواد و ولگرد متوجه مردم بود امروزه این خطر توسط افرادی تحمیل می‌شود که با هوش و باسوادند (حتی باهوش تر از افراد معمولی) و قدرت نفوذ و ضربه به شبکه را دارند . معمولاً هدف افرادی که به شبکه های کامپیوتری نفوذ یا حمله می‌کنند یکی از موارد زیر است :

- تفریح یا اندازه گیری ضریب توانائی فردی یا کنجکاوی ( معمولاً دانشجویان! )
- دزدیدن دانشی که برای تهیه آن بایستی صرف هزینه کرد. ( راهزنان دانش )
- انتقام جوئی و ضربه زدن به رقیب
- آزار رسانی و کسب شهرت از طریق مردم آزاری ( بیماران روانی )

- جاسوسی و کسب اطلاع از وضعیت نظامی و سیاسی یک کشور یا منطقه
- رقابت ناسالم در عرصه تجارت و اقتصاد
- جابجا کردن مستقیم پول و اعتبار از حسابهای بانکی و دزدیدن شماره کارتهای اعتبار
- کسب اخبار جهت اعمال خرابکاری و موذیانه (توسط تروریستها)

بهر حال امروزه امنیت ملی و اقتدار سیاسی و اقتصادی به طرز پیچیده ای به امنیت اطلاعات گره خورده و نه تنها دولتها بلکه تک تک افراد را نیز تهدید می‌کند. برای ختم مقدمه از شما سوال می‌کنیم که چه حالی به شما دست می‌دهد وقتی متوجه شوید که شماره حساب بانکی یا کارت اعتباریتان توسط فرد ناشناسی فاش شده و انبوهی هزینه روی دست شما گذاشته است؟ پس بعنوان یک فرد مطلع از خطراتی که یک شبکه کامپیوتری را تهدید می‌کند این فصل را دنبال کنید.

## ۱-۱) سرویسهای امنیتی در شبکه ها

تهدیدهای بالقوه برای امنیت شبکه های کامپیوتری بصورت عمده عبارتند از:

- فاش شدن غیر مجاز اطلاعات در نتیجهٔ استراق سمع داده‌ها یا پیامهای در حال مبادله روی شبکه
- قطع ارتباط و اختلال در شبکه به واسطه یک اقدام خرابکارانه
- تغییر و دستکاری غیر مجاز اطلاعات یا یک پیغام ارسال شده

بایستی با مفاهیم اصطلاحات زیر بعنوان سرویسهای امنیتی آشنا باشید:

**الف) محرمانه ماندن اطلاعات**[۱]: دلایل متعددی برای یک سازمان یا حتی یک فرد عادی وجود دارد که بخواهد اطلاعات خود را محرمانه نگه دارد.

**ب) احراز هویت**[۲]: پیش از آنکه محتوای یک پیام یا اطلاعات اهمیت داشته باشد باید مطمئن شوید که پیام حقیقتاً از کسی که تصور می‌کنید رسیده است و کسی قصد فریب و گمراه کردن (یا آزار) شما را ندارد .

---

[۱] Confidentality
[۲] Authentication

**ج) سلامت داده‌ها**[۱]: یعنی دست نخوردگی و عدم تغییر پیام و اطمینان از آنکه داده‌ها با اطلاعات مخرب مثل یک ویروس کامپیوتری آلوده نشده اند.

**د) کنترل دسترسی**[۲]: یعنی مایلید دسترسی افرادی را که مجاز نیستند ، کنترل کنید و قدرت منع افرادی را که از دیدگاه شما قابل اعتماد به شمار نمی‌آیند از دسترسی به شبکه داشته باشید.

**ه) در دسترس بودن**[۳]: با این تفاصیل ، باید تمام امکانات شبکه بدون دردسر و زحمت در اختیار آنهائی که مجاز به استفاده از شبکه هستند ، باشد و در ضمن هیچکس نتواند در دسترسی به شبکه اختلال ایجاد کند .

زمانی که یکی از سرویسهای امنیتی پنج گانه فوق نقض شود می‌گوئیم به سیستم حمله شده است . معمولاً یک شبکه کامپیوتری درمعرض چهار نوع حمله قرار دارد:

- **حمله از نوع وقفه**[۴]: بدین معنا که حمله کننده باعث شود شبکه مختل شده و مبادله اطلاعات امکان پذیر نباشد.
- **حمله از نوع استراق سمع** [۵]: بدین معنا که حمله کننده به نحوی توانسته اطلاعات در حال تبادل روی شبکه را گوش داده و بهره برداری نماید .
- **حمله از نوع دستکاری داده‌ها**[۶]: یعنی حمله کننده توانسته به نحوی اطلاعاتی که روی شبکه مبادله می‌شود را تغییر دهد یعنی داده‌هایی که در مقصد دریافت میشود متفاوت با آنچیزی باشد که از مبداء آن ارسال شده است .
- **حمله از نوع افزودن اطلاعات**[۷]: یعنی حمله کننده اطلاعاتی را که در حال تبادل روی شبکه است تغییر نمی‌دهد بلکه اطلاعات دیگری را که می‌تواند مخرب یا بنیانگذار حملات بعدی باشد ، به اطلاعات اضافه می‌نماید (مثل ویروسها)

به حمله‌ای که هنگام شروع با بروز اختلال در شبکه علنی می‌شود و در کار ارسال یا دریافت مشکل ایجاد می‌کند "حمله فعال" می‌گویند. بر عکس حمله‌ای که شبکه را

---

۱ Integrity
۲ Access Control
۳ Availability
۴ Interruption
۵ Interception
۶ Modification
۷ Fabrication

با اختلال مواجه نمی‌کند و ظاهراً مشکلی در کار ارسال و دریافت بوجود نمی‌آورد "حمله غیر فعال"[۱] نامیده می‌شود و از خطرناکترین انواع حمله به شبکه به شمار می‌رود.

در ادامه این فصل دو راه کلی برای حراست و حفظ امنیت اطلاعات در یک شبکه کامپیوتری معرفی می‌شود:

- حراست و حفاظت داده‌ها و شبکه از طریق نظارت بر اطلاعات و دسترسیها به کمک سیستمی که "دیوار آتش"[۲] نامیده می‌شود.
- رمزگذاری اطلاعات به گونه ای که حتی اگر کسی آنها را دریافت کرد نتواند محتوای آنرا بفهمد و و از آن بهره برداری کند .

برای تمایز دو مورد فوق مثال عامیانه زیر بد نیست:

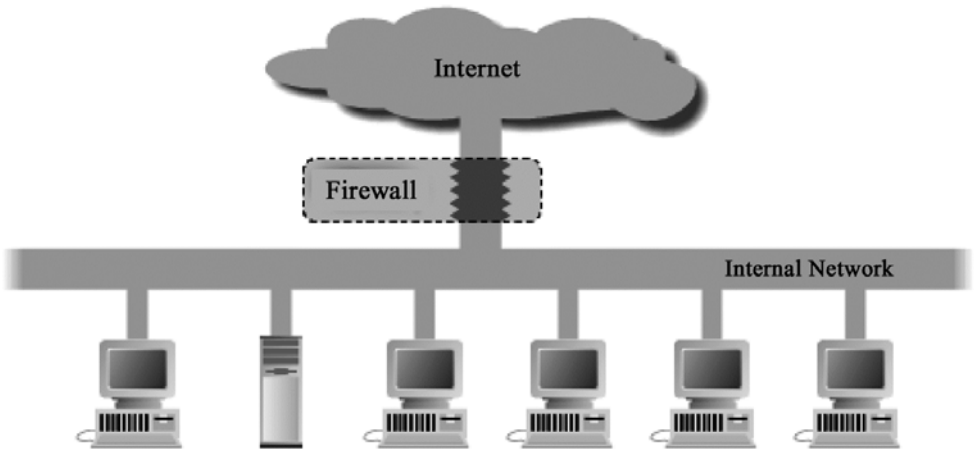
چون احتمال سرقت همیشه وجود دارد اولاً شما قفلهای مطمئن و دزدگیر برای منزل خود نصب می‌کنید و احتمالاً نگهبانی می‌گمارید تا ورود و خروج افراد را نظارت کند (کاری که دیوار آتش انجام می‌دهد) ثانیاً چون باز هم احتمال نفوذ می‌دهید لوازم قیمتی و وجوه نقد را در گوشه ای مخفی می‌کنید تا حتی در صورت ورود سارق موفق به پیدا کردن و بهره برداری از آن نشود . با تمام این کارها باز هم اطمینان صددرصد وجود ندارد چرا که هر کاری از یک انسان باهوش بر می‌آید.

## ۲) دیوار آتش

دیوار آتش سیستمی است که در بین کاربران یک شبکه محلی و شبکه بیرونی (مثلاً اینترنت) قرار می‌گیرد و ضمن نظارت بر دسترسیها ، در تمام سطوح ورود و خروج اطلاعات را تحت نظر دارد. مدلی ساده برای یک سیستم دیوار آتش درشکل (۱-۱۱) ارائه شده است . در این ساختار هر سازمان یا نهادی که بخواهد ورود و خروج اطلاعات شبکه را کنترل کند موظف است تمام ارتباطات مستقیم شبکه داخلی خود را با دنیای خارج قطع کرده و هرگونه ارتباط خارجی از طریق یک دروازه که در شکل (۱-۱۱) نشان داده شده ، انجام شود.

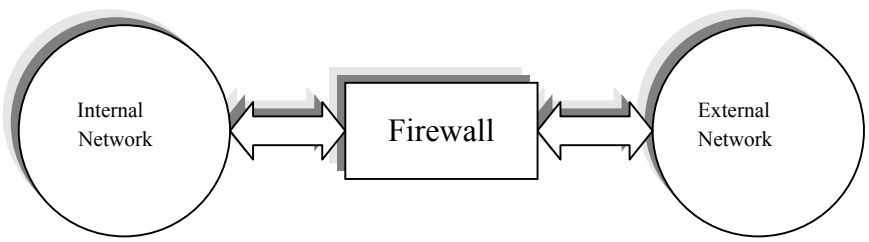
---

[۱] Passive
[۲] Firewall




شکل (۱-۱۱) نمودار کلی بکارگیری یک دیوار آتش

قبل از آنکه اجزای یک دیوار آتش را تحلیل کنیم باید عملکرد کلی و مشکلات استفاده از یک دیوار آتش را بررسی کنیم.
در شکل (۲-۱۱) مدل ساده تر شده یک دیوار آتش را در نظر بگیرید:




شکل (۲-۱۱) نمایی ساده از یک دیوار آتش

بسته‌های IP قبل از مسیریابی روی شبکه اینترنت ابتدا وارد دیوار آتش می‌شوند و منتظر می‌مانند تا طبق معیارهای حفاظتی و امنیتی پردازش شوند . پس از پردازش و تحلیل بسته سه حالت ممکن است اتفاق بیفتد:
**الف)** اجازه عبور بسته صادر شود. (Accept Mode)

**ب)** بسته حذف گردد. (Blocking Mode)
**ج)** بسته حذف شده و پاسخ مناسب به مبداء آن بسته داده شود. (Response Mode)
(به غیر از پیغام حذف بسته می‌توان عملیاتی نظیر اخطار ، رد گیری ، جلوگیری از ادامه استفاده از شبکه و توبیخ هم در نظر گرفت)
در حقیقت دیوار آتش محلی است برای ایست و بازرسی بسته‌های اطلاعاتی به گونه ای که بسته‌ها بر اساس تابعی از قواعد امنیتی و حفاظتی ، پردازش شده و برای آنها مجوز عبور یا عدم عبور صادر شود .
اگر P مجموعه ای از بسته‌های ورودی به سیستم دیوار آتش در نظر گرفته شود و S مجموعه ای متناهی از قواعد امنیتی باشد داریم:

$$X=F(P,S)$$

F تابع عملکرد دیوار آتش و X نتیجه تحلیل بسته ( شامل سه حالت Accept,Blocking,Response ) خواهد بود.

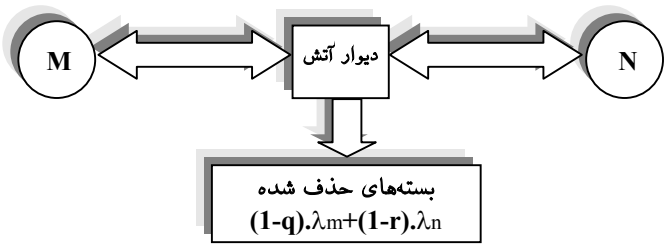
همانطوریکه همه جا عملیات ایست و بازرسی وقتگیر و اعصاب خرد کن است دیوار آتش هم بعنوان یک گلوگاه[۱] می‌تواند منجر به بالارفتن ترافیک ، تاخیر، ازدحام[۲] و نهایتاً بن بست در شبکه شود . (بن بست زمانی است که بسته‌ها آنقدر در حافظه دیوار آتش معطل می‌شوند تا طول عمرشان تمام شده و فرستنده اقدام به ارسال مجدّد آنها کرده و این کار بطور متناوب تکرار شود) به همین دلیل دیوار آتش نیاز به طراحی صحیح و دقیق دارد تا از حالت گلوگاهی خارج شود. ( تاخیر در دیوار آتش مجموعاً اجتناب ناپذیر است فقط بایستی بگونه ای باشد که بحران ایجاد نکند. )
اگر از دیدگاه نظریه صف[۳] به یک دیوار آتش نگاه کنیم می‌توان تخمینی از تاخیر تحمیل شده به هر بسته را بدست آورد. معمولاً تابع توزیعِ تولید بسته‌ها را در شبکه های اطلاعاتی پوآسون در نظر می‌گیرند.
در شکل زیر فرض کنید $\lambda_n$ متوسط انتقال بستهٔ IP در واحد زمان از شبکه N به دیوار آتش و $\lambda_m$ متوسط انتقال بسته در واحد زمان از شبکه M باشد . q را احتمال عبور بسته $P_M$ و r را احتمال عبور بسته $P_N$ فرض کنید؛ طبق شکل (۳-۱۱) داریم:

**متوسط بسته‌های حذف شده** $= (1-q).\lambda_m+(1-r).\lambda_n$

---

۱ Bottleneck
۲ Congestion
۳ Queuing Theory

**متوسط انتقال بسته از دیوار آتش به M = $r.\lambda_n$**
**متوسط انتقال بسته از دیوار اتش به N = $q.\lambda_m$**




شکل (۳-۱۱) دیوار آتش از دیدگاه نظریه صف

طبق نظریه صف اگر دیوار آتش بخواهد از نقش گلوگاهی خود بکاهد بایستی بگونه ای طراحی شود که نسبت متوسط خروجی بسته‌ها از دیوار آتش ( $\mu$ ) به ورودی بسته‌ها ( یعنی نسبت $\mu/\lambda$ ) تا حدّ امکان زیاد باشد که این کار منوط به افزایش سرعت پردازش ، داشتن حافظه کافی برای ذخیره بسته‌های پردازش نشده و هر چه سریعتر کردن تابع تصمیم گیری می‌باشد. مشکل زمانی حاد می‌شود که دیوار آتش مجبور باشد برای تصمیم گیری و اجازه عبور تعدادی از بسته‌ها را نگه دارد تا تصمیم گیری بر اساس مجموعه ای از بسته‌ها انجام شود . این موضوع در ادامه آشکار خواهد شد.

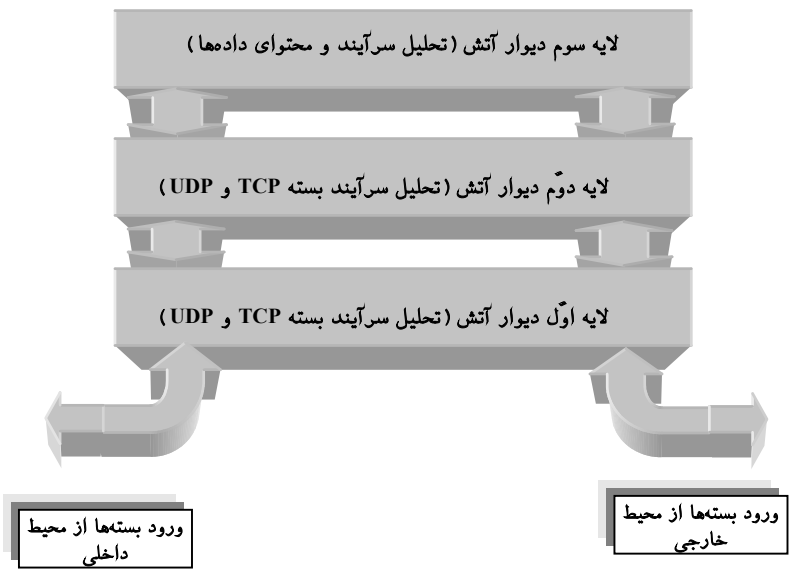
## ۳) مبانی طراحی دیوار آتش

از آنجائی که معماری شبکه بصورت لایه به لایه است ، در مدل TCP/IP برای انتقال یک واحد اطلاعات از لایه چهارم بر روی شبکه باید تمام لایه ها را بگذراند و و هر لایه برای انجام وظیفهٔ خود تعدادی فیلد مشخص به ابتدای بستهٔ اطلاعاتی اضافه کرده و آنرا تحویل لایه زیرین می‌دهد. قسمت اعظم کار یک دیوار آتش تحلیل فیلدهای اضافه شده در هر لایه و سرآیند هر بسته می‌باشد . در بسته ای که وارد دیوار آتش می‌شود به تعداد لایه ها (٤ لایه) سرآیند متفاوت وجود خواهد داشت . معمولاً سرآیند لایه اوّل (لایه فیزیکی یا Network Interface در شبکه اینترنت) اهمیت چندانی ندارد چرا که محتوای این فیلدها فقط روی کانال فیزیکی از شبکه محلی معنا دارند و در گذر از هر شبکه یا مسیر یاب این فیلدها عوض

خواهند شد. بیشترین اهمیت در سرآیندی است که در لایه های دوم ، سوم و چهارم به یک واحد از اطلاعات اضافه خواهد شد:

- در لایه شبکه دیوار آتش فیلدهای بسته IP را پردازش و تحلیل می‌کند.
- در لایه انتقال دیوار آتش فیلدهای بسته‌های TCP یا UDP را پردازش و تحلیل می‌کند .
- در لایه کاربرد دیوار آتش فیلدهای سرآیند و همچنین محتوای خود داده‌ها را بررسی می‌کند . ( مثلا سرآیند و محتوای یک نامه الکترونیکی یا یک صفحه وب می‌تواند مورد بررسی قرار گیرد.)

با توجه به لایه لایه بودن معماری شبکه لاجرم یک دیوار آتش نیز لایه به لایه خواهد بود به شکل (٤-١١) دقت کنید:




شکل (٤-١١) لایه بندی ساختار یک دیوار آتش

اگر یک بسته در یکی از لایه های دیواره آتش شرایط عبور را احراز نکند همانجا حذف شده و به لایه های بالاتر ارجاع داده نمی‌شود بلکه این امکان وجود دارد که

آن بسته جهت پیگیریهای امنیتی نظیر ثبت عمل و ردگیری به سیستمی جانبی تحویل داده شود.

سیاست امنیتی یک شبکه مجموعه ای متناهی از قواعد امنیتی است که بنابر ماهیتشان در یکی از سه لایه دیوار آتش تعریف می‌شوند ، بعنوان مثال:

- قواعد تعیین آدرسهای ممنوع در اولین لایه از دیوار آتش
- قواعد بستن برخی از سرویسها مثل Telnet یا FTP در لایه دوم
- قواعد تحلیل سرآیند متن یک نامه الکترونیکی یا صفحه وب در لایه سوم

### ۱-۳) لایه اول دیوار آتش

لایه اول در دیوار آتش بر اساس تحلیل بسته IP و فیلدهای سرآیند این بسته کار می‌کند و در این بسته فیلدهای زیر قابل نظارت و بررسی هستند :

- **آدرس مبداء**: برخی از ماشینهای داخل یا خارج شبکه با آدرس IP خاص "حق ارسال" بسته نداشته باشند و بسته‌های آنها به محض ورود به دیوار آتش حذف شود.
- **آدرس مقصد**: برخی از ماشینهای داخل یا خارج شبکه با آدرس IP خاص "حق دریافت" بسته نداشته باشند و بسته‌های آنها به محض ورود به دیوار آتش حذف شود .
- (آدرسهای IP غیر مجاز توسط مسئول دیوار آتش تعریف می‌شود)
- **شماره شناسایی یک دیتاگرام**[1] : بسته‌هائی که متعلق به یک دیتاگرام خاص هستند حذف شوند.
- **شماره پروتکل**: بسته‌هایی که متعلق به پروتکل خاصی در لایه بالاتر هستند می‌توانند حذف شود. یعنی بررسی اینکه بسته متعلق به چه پروتکلی در لایه بالاتر است و آیا برای تحویل به آن پروتکل مجاز است یا نه .
- **زمان حیات بسته**: بسته‌هائی که بیش از تعداد مشخصی مسیریاب را طی کرده اند مشکوک هستند و باید حذف شوند.

[1] Identifier & Fragment offset

- بقیه فیلدها بنابر صلاحدید و قواعد امنیتی مسئول دیوار آتش قابل بررسی هستند.

مهمترین خصوصیت لایهٔ اول از دیوار آتش آنست که در این لایه بسته‌ها بطور مجزا و مستقل از هم بررسی می‌شوند و هیچ نیازی به نگه داشتن بسته‌های قبلی یا بعدی یک بسته نیست. بهمین دلیل ساده ترین و سریعترین تصمیم گیری در این لایه انجام می‌شود. امروزه برخی از مسیریابها با امکان لایه اوّل دیوار آتش به بازار عرضه می‌شوند یعنی به غیر از مسیریابی ، وظیفهٔ لایه اوّل یکَ دیوار آتش را هم انجام می‌دهند که به آنها "مسیریابهای فیلتر کنندهٔ بسته"[1] گفته می‌شود. بنابراین مسیریاب قبل از اقدام به مسیریابی ، بر اساس جدولی بسته‌های IP را غربال می‌کند و تنظیم این جدول بر اساس نظر مسئول شبکه و برخی از قواعد امنیتی انجام می‌گیرد.

با توجه به سریع بودن این لایه هر چه درصد قواعد امنیتی در این لایه دقیقتر و سختگیرانه تر باشد حجم پردازش در لایه های بالاتر کمتر و در عین حال احتمال نفوذ پایینتر خواهد بود ولی در مجموع بخاطر تنوع میلیاردی آدرسهای IP نفوذ از این لایه با آدرسهای جعلی یا قرضی امکان پذیر خواهد بود و این ضعف در لایه های بالاتر بایستی جبران شود.

### ۲-۳) لایه دوم دیوار آتش

در این لایه از فیلدهای سرآیند لایه انتقال برای تحلیل بسته استفاده می‌شود. عمومی‌ترین فیلدهای این بسته عبارتند از :

- **شماره پورت پروسه مبداء و شماره پورت پروسه مقصد**: با توجه به آنکه پورتهای استاندارد شناخته شده هستند ممکن است مسئول یک دیوار آتش بخواهد سرویس ftp (انتقال فایل) فقط در محیط شبکه محلی امکان پذیر باشد و برای تمام ماشینهای خارجی این سرویس وجود نداشته باشد بنابراین دیوار آتش می‌تواند بسته‌های TCP با شماره پورت ۲۰ و ۲۱ (مربوط به ftp) که قصد ورود یا خروج از شبکه را دارند ،حذف کند. یکی دیگر از سرویسهای خطرناک که ممکن است مورد

[1] Pocket Filtering Router

سوء استفاده قرار گیرد Telnet است که می‌توان براحتی پورت ۲۳ را مسدود کرد یعنی بسته‌هائی که شماره پورت مقصدشان ۲۳ است حذف شوند.

- **فیلد شماره ترتیب[1] و فیلد Acknowledgment:** این دو فیلد نیز بنا بر قواعد تعریف شده توسط مسئول شبکه قابل استفاده هستند.

از مهمترین خصوصیات این لایه آنست که تمام تقاضاهای برقراری ارتباط TCP بایستی از این لایه بگذرد و چون در ارتباط TCP ، تا مراحل **"دست تکانی سه گانه اش"** به اتمام نرسد انتقال داده امکان پذیر نیست لذا قبل از هر گونه مبادله داده دیوار آتش می‌تواند مانع برقراری هر ارتباط غیر مجاز شود . یعنی دیوار آتش می‌تواند تقاضاهای برقراری ارتباط TCP را قبل از ارائه به ماشین مقصد بررسی نماید و در صورت غیر قابل اعتماد بودن ، مانع از برقراری ارتباط شود. دیوار آتش در این لایه نیاز به جدولی از شماره پورتهای غیر مجاز دارد.

### ۳-۳) لایه سوم دیوار آتش

در این لایه حفاظت بر اساس نوع سرویس و برنامه کاربردی انجام می‌شود. یعنی با در نظر گرفتن پروتکل در لایه چهارم به تحلیل داده‌ها می‌پردازد. تعداد سرآیند ها در این لایه بسته به نوع سرویس بسیار متنوع و فراوان است. بنابراین در لایه سومِ دیوار آتش برای هر سرویس مجزا (مثل سرویس پست الکترونیکی، سرویس ftp، سرویس وب و ...) باید یک سلسله پردازش و قواعد امنیتی مجزا تعریف شود و به همین دلیل حجم و پیچیدگی پردازش در لایه سوم زیاد است . توصیه موکّد آنست که تمام سرویسهای غیرضروری و شماره پورتهایی که مورد استفاده نیستند در لایه دوم مسدود شوند تا کار در لایه سوم کمتر باشد.

بعنوان مثال فرض کنید موسسه ای نظامی سرویس پست الکترونیکی خود را دائر کرده ولی نگران فاش شدن برخی اطلاعات محرمانه است . در این حالت دیوار آتش در لایه سوّم می‌تواند کمک کند تا برخی از آدرسهای پست الکترونیکی مسدود شود ، در عین حال می‌تواند در متون نامه های رمز نشده دنبال برخی از کلمات کلیدی

---

[1] Sequence Number

حسّاس بگردد و متون رمزگـذاری شـده را در صورتـی کـه موفـق بـه رمـزگشـائی آن نشـود حذف نمـاید.

بعنوان مثالی دیگـر یـک مرکـز فرهنگـی علاقمنـد اسـت قبـل از تحویـل صفحـه وب بـه یـک کـاربر ، درون آنـرا از لحـاظ وجـود برخـی از کلمـات کلیـدی بررســـی کنـــد و اگـــر کلماتی که با معیارهـــای فرهنگـی مطـابقت نـدارد درون متـن صفحـه یـافت شـد آن صفحـه را حذف نمـاید.

## ۴) اجزای جانبی یک دیوار آتش

دیوار آتـش یـک سیسـتم امنیتـی اسـت کـه سیاسـتهای مسـئول شـبکه را پیـاده و اعمـال می‌کنـد. بنـابراین دیـوار آتـش بایسـتی از طریـق یـک ورودی سـهل و راحـت قواعـــد را از مسـئول شـبکه دریـافت نمـاید و همـواره فعالیتـهای موجـــود روی شـــبکه را بـــه مســـئول شـبکه گـزارش بدهـد . بـهمین دلیـل معمـولاً یـک سیسـتم دیـوار آتـش دارای اجـزاء ذیـل است:

### ۴-۱) واسط محاوره ای و ساده ورودی/خروجی

برای تبادل اطلاعات و ســهولت در تنظیـم قواعـد امنیتـی و ارائـه گـزارش، نیـاز بـه یـک واسط کـاربر[۱] کـه سـاده و در عیـن حـال کـارآمد باشـد وجـود دارد . معمـولاً واسـط کـاربر مستقل از سیستم دیوار آتــش اسـت تـا حجـم پـردازش اضـافی روی سیسـتم تحمیـل نکنـد یعنی معمولاً دیوار آتـش دارای دسـتگاهی بـه عنـوان صفحـه نمـایش نیسـت بلکـه از طریـق وصل یک ابزار جانبی مثــل یـک ترمینـال سـاده یـا یـک کـامپیوتر شـخصی معمولـی فرمـان می‌گیرد یا گــزارش می‌دهـد.

### ۴-۲) سیستم ثبت[۲]

برای بـالاتر بـردن ضریـب امنیـت و اطمینـان در شـبکه ، دیـوار آتـش بـاید بتوانـد حتـی در مواقعیکـه اجـازه خـروج یـا ورود یـک بسـته بـه شـبکه صـــادر می‌شـــود اطلاعـــاتی در

---

[۱] User Interface
[۲] Logger

مورد آن بسته ذخیره کند تا در صورت هر گونه حمله یا نفوذ بتوان مسئله را پیگیری کرد. در یک دیوار آتش عملی که ثبت کننده می‌تواند انجام بدهد آنست که مبداء و مقصد بسته‌های خروجی و ورودی ، شماره پورتهای مبداء و مقصد ، سرآیند یا حتی محتوای پیام در لایه کاربرد را ( برای تمام مبادلات خارج از شبکهٔ محلی) ذخیره کند و لحظه به لحظه مبادلهٔ اطلاعات تمام کاربران و حتی مسئول شبکه را در فایلی درج نماید . این اطلاعات می‌تواند بعنوان سندی بر علیه فرد خاطی استفاده شود یا به یافتن کسی که در خارج از شبکه مشغول اخلال گری است کمک نماید .

### ۳-۴) سیستم هشدار دهنده

در صورت بروز هر گونه مشکل یا انتقالی مشکوک ، دیوار آتش می‌تواند مسئول شبکه را مطلع نماید و در در صورت لزوم کسب تکلیف کند . اعمال مشکوک در هر سه لایه تعریف میشود : مثل تقاضای ارتباط با آدرسهای IP مشکوک ، آدرسهای پورت مشکوک ، اطلاعات مشکوک در لایه کاربرد (صفحات وب یا نامه های مشکوک).

ارتباط مشکوک را می‌توان مصداق ارتباطاتی دانست که بی هدف یا مکرر در طی روز برقرار می‌شود یا آنکه اطلاعات ارسالی مفهوم یا مضمون خاصّی نداشته باشد یا آنکه رمز شده باشد. در این حالت سیستم دیوار آتش ضمن کسب تکلیف می‌تواند یک آدرس مشکوک را به عنوان آدرس غیر مجاز مسدود کند.

## ۵) راه حل نهائی

با تمام نظارتی که بر تردد بسته‌های اطلاعاتی حین ورود یا خروج از شبکه می‌شود باز هم می‌توان زیرکانه از مرز دیوار آتش عبور کرد و بهترین حفاظت برای جلوگیری از فاش شدن اطلاعات محرمانه به دنیای خارج، نابود کردن خط ارتباطی شبکه به دنیای خارج است! چرا که می‌توان اطلاعات سری را رمز و فشرده کرد و آنرا بعنوان بیت آخر از نقاط تصویر یک گل رز بعنوان کارت پستال تبریک سال نو ارسال نمود. در حقیقت سیستم دیوار آتش فقط یک ابزار محدود کننده است و اطمینان صد در صد ندارد.

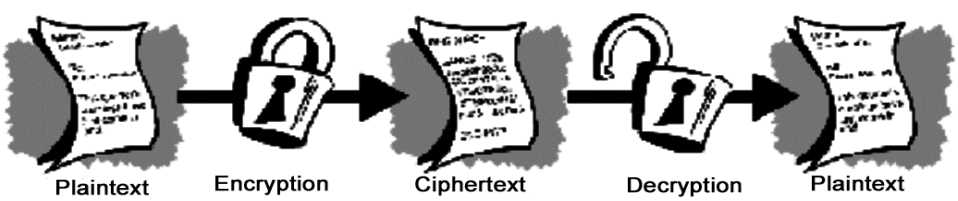
## ۶) رمزنگاری

زمانیکه ژولیوس سزار پیامهائی را برای فرمانده ارتش خود در جنگ می‌فرستاد از بیم کشته شدن یا خیانت پیک ، در تمام متنِ نامهٔ خود هرحرف را با حرفی که سه تا بعد از آن قرار گرفته بود عوض می‌کرد (مثلاً بجای A حرف D و بجای B حرف E را قرار می‌داد) تا متن حالت معنی دار خود را از دست بدهد. تنها کسی می‌توانست از مفهوم متن چیزی بفهمد که به رمز آن (یعنی Shift by 3 ) آگاهی داشت.

داده‌هائی که به راحتی قابل فهم هستند و هیچ نکته و ابهام خاصی در درک آنها وجود ندارد، متن ساده یا متن آشکار نامیده می‌شوند. روشی که باعث می‌شود متن ساده حالت قابل درک و فهم خود را از دست بدهد "رمزنگاری"[1] نامیده می‌شود. معمولاً در دنیای شبکه های کامپیوتری رمزنگاری سلسله ای از عملیات ریاضی است که مجموعه ای از اطلاعات خالص و قابل فهم را به مجموعه ای از اطلاعات غیرقابل فهم ، بی معنا و بلا استفاده تبدیل می‌کند. به گونه ای که فقط گیرنده اصلی آن قادر باشد آنرا از حالت رمز خارج و از آن بهره برداری نماید. ( یعنی کلید رمز را در اختیار داشته باشد)

علم رمزنگاری [2] با اصول ریاضی به رمز درآوردن اطلاعات و خارج کردن آنها از حالت رمز سر و کار دارد. در مقابلِ علم رمزنگاری ، علم تحلیل رمز[3] قرار دارد که روشهای تجزیه و شکستن رمز اطلاعات (بدون نیاز به کلید) و کشف کلید رمز را مورد بحث قرار می‌دهد.

به شکل (۵-۱۱) دقت کنید. در این شکل سیمای کلی یک سیستم رمزنگاری و رمزگشائی به تصویر کشیده شده است .




شکل (۵-۱۱) سیمای کلّی سیستم رمزنگاری و رمزگشائی

---

[1] Encryptioin
[2] Cryptography
[3] Cryptoanalysis

الگوریتم یا روشی که بر اساس آن متن رمز می‌شود باید بگونه ای قابل برگشت (وارون پذیر) باشد تا بتوان به متن اصلی دست پیدا کرد.

در ادامه چند روش رمزنگاری را معرفی می‌نمائیم:

### ۱-۶) روشهای جانشینی[1]

روش جانشینی قدیمی‌ترین نوع رمزنگاری است که اولین بار سزار آن را بکار برده است. در این روش هر حرف از جدول حروف الفبا به حرفی دیگر تبدیل می‌شود . بعنوان مثال در رمز سزار هر حرف به حرف سوم بعد از خودش تبدیل می‌شد که با این روش کلمه "حمله"[2] بصورت زیر در می‌آمد:

Attack متن اصلی

Dwwdfn متن رمز شده

این روش بعداً بهبود داده شد و بجای آنکه تمام حروف بطور منظم و با قاعده به یکدیگر تبدیل شوند جدول حروف الفبا طبق یک قاعده نامشخص که "جدول رمز" نامیده می‌شد به هم تبدیل می‌شدند. بعنوان مثال اگر نامه یا متن تماماً حروف کوچک در نظر بگیریم جدول رمز می‌تواند بصورت زیر باشد:

| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | Y | z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Q | W | E | R | T | Y | U | I | O | P | A | S | D | F | G | H | J | K | L | Z | X | C | V | B | N | M |

طبق این جدول که گیرنده پیام بایستی از آن آگاهی داشته باشد کلمه attack به کلمه QZZQEA تبدیل می‌شود. شاید یک مبتدی احساس کند که این روش امروزه مفید خواهد بود چرا که جدول رمز دارای !26 (معادل $٤*١٠^{٢٦}$) حالت متفاوت خواهد بود و امتحان تمام این حالات مختلف برای یافتن جدول رمز کاری مشکل است ، در حالی که چنین نیست و این نوع رمزنگاری برای متون معمولی در کسری از ثانیه و بدون کلید رمز شکسته خواهد شد! نقطه ضعف این روش در مشخصات

[1] Substitution
[2] Attack

آماری هر حرف در یک زبان می‌باشد. بعنوان مثال درزبان انگلیسی حرف e در متن بیش از همه حروف تکرار می‌شود . ترتیب فراوانیِ نسبی برای شش حرف پرتکرار در متون انگلیسی بصورت زیر است:

e > t > o > a > n > i

اولین اقدام در رمزشکنی ( رمزشکنی همان رمزگشائی است بدون در اختیار داشتن کلید یا جدول رمز) آنست که متن رمز شده تحلیل آماری شود و هر کاراکتری که بیش از همه در آن تکرار شده باشد معادل e ، حرف پرتکرار بعدی معادل t قرار بگیرد و روند به همین ترتیب ادامه یابد. البته ممکن است برخی از حروف اشتباه سنجیده شوند ولی می‌توانند در مراحل بعدی اصلاح شود.

دومین نکته آنست که در زبانی مثل انگلیسی حروف کنار هم وابستگی آماری بهم دارند مثلاً در ۹۹/۹ درصد مواقع در سمت راست حرف q حرف u قرار گرفته (qu) یا در کنار حرف t معمولاً (البته با احتمال پائین تر) h قرار گرفته است. یعنی بمحض کشف حرف q رمز u هم کشف می‌شود و اگر t کشف شد کشف h ساده تر خواهد شد. ترتیب فراوانیِ نسبی برای پنج "دوحرفی"[1] پرتکرار در متون انگلیسی بصورت زیر است:

th > in > er > re > an

سومین نکته نیز به سه حرفی ها بر می‌گردد. مثلاً در زبان انگلیسی سه حرفی های ion, and, the, ing به کرّات استفاده می‌شوند و می‌توانند ملاک رمزشکنی قرار بگیرند.

چهارمین نکته برای رمزشکنی مراجعه به فرهنگ لغات یک زبان است که بر اساس پیدا شدن چند حرف از یک کلمه رمز بقیه حروف آن نیز آشکار میشود.

به دلائل فوق روش رمزگذاری جانشینی کارآئی مناسبی ندارد و براحتی رمز آن بدست خواهد آمد.

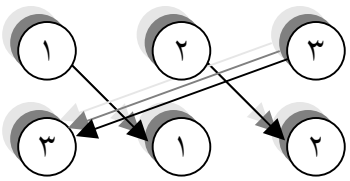
### ۲-۶) رمزنگاری جایگشتی[2]

در رمزگذاری جانشینی محل قرار گرفتن و ترتیب حروف کلمات در یک متن بهم نمی‌خورد بلکه طبق یک جدول رمز جایگزین می‌شد. در روش رمزنگاری جایگشتی آرایش و ترتیب کلمات به هم می‌خورد. بعنوان یک مثالِ بسیار ساده فرض

---

[1] Digram
[2] Permutation

کنید تمام حروف یک متن اصلی را سه تا سه تا جدا کرده و طبق قاعده زیر ترتیب آن را بهم بریزیم:




کلمه اصلی: **the**
کلمه رمز: **eth**

برای رمزگشائی، گیرندهٔ پیام باید کلید جایگشت را که در مثال ما (۲و۱و۳) است بداند.

معمولا برای راحتی در به خاطر سپردن کلید رمز ، یک کلید متنی انتخاب میشود و سپس جایگشت بر اساس ترتیب حروف کلمهٔ رمز انجام می‌شود. برای وضوح این روش به مثال زیر دقت کنید:

متن اصلی: please-trasnfer-one-million-dollors-to-my-swiss-bank-account-six-two-two

کلمه رمز: **MEGABUCK**

تمام کلمات متن اصلی بصورت دسته های هشت تائی جدا شده و تماماً زیر هم نوشته می‌شود: ( علامتِ – را فاصلهٔ خالی در نظر بگیرید )

| کلمه رمز | *M* | *E* | *G* | *A* | *B* | *U* | *C* | *K* |
|---|---|---|---|---|---|---|---|---|
| ترتیب حروف کلمه رمز | ۷ | ٤ | ٥ | ۱ | ۲ | ۸ | ۳ | ٦ |
| ۱ | p | l | e | a | s | e | - | t |
| ۲ | r | a | n | s | f | e | r | - |
| ۳ | o | n | e | - | m | i | l | l |
| ٤ | i | o | n | - | d | o | l | l |
| ٥ | a | r | s | - | t | o | - | m |
| ٦ | y | - | s | w | i | s | s | - |
| ۷ | b | a | n | k | - | a | c | c |
| ۸ | o | u | n | t | - | s | i | x |
| ۹ | t | w | o | - | t | w | o | - |

حال بر اساس ترتیب الفبایی هر حرف در کلمه رمز، ستونها بصورت پشت سر هم نوشته می‌شوند . یعنی ابتدا ستون مربوط به حرف A ، سپس B ، E و ...

پس رمز بصورت زیر در می‌آید:

**"as---wkt-sfmdti---rll-sciolanor-auwenenssnnot-llm-cx-proiayboteeioosasw"**

بنابراین برای بازیابیِ اصل پیام درمقصد ، گیرنده باید کلید رمز (یا حداقل ترتیب جایگشت) را بداند.

این روش رمز هم قابل شکستن است چرا که اگر چه ترتیب حروف بهم ریخته است ولی در متن رمز شده تمام حروف هر یک از کلمات وجود دارند. بعنوان مثال تک تک حروف dollars یا swiss bank را می‌توان در متن رمز شده پیدا کرد لذا با بررسی تمام حالات ممکن که کلمه dollars را به صورت پراکنده در متن در می‌آورد می‌توان رمز را بدست آورد. البته حجم پردازش مورد نیاز بیشتر خواهد بود ولی بهر حال این نوع رمزگذاری براحتی قابل شکستن می‌باشد و در دنیای امروز چندان قابل اعتماد نیست.

## ۷) استانداردهای نوین رمزگذاری

در اوائل دهه هفتاد دولت فدرال آمریکا و شرکت آی.بی.ام (IBM) مشترکاً روشی را برای رمزنگاری داده‌ها ایجاد کردند که بعنوان استانداردی برای نگهداری اسناد محرمانه دولتی مورد استفاده قرار گرفت. این استاندارد که DES[1] نام گرفت امروزه محبوبیت خود را از دست داده است .

الگوریتم روش رمزنگاری DES در شکل (٦-١١) نشان داده شده است که در ادامه کلیّت آنرا توضیح می‌دهیم:

ورودی رمزنگار یک رشته ٦٤ بیتی است، بنابراین متنی که باید رمز شود بایستی در گروههای هشت کاراکتری دسته بندی شوند . اولین عملی که بر روی رشته ورودی ٦٤ بیتی انجام می‌شود جابجا کردن محل بیتهای رشتهٔ ٦٤ بیتی طبق جدول صفحهٔ بعد است . به این عمل جایگشت مقدماتی[2] گفته می‌شود :

[1] Data Encryption Standard
[2] Initial permutation

| جدول جایگشت مقدماتی | | | | | | | |
|---|---|---|---|---|---|---|---|
| 58 | 50 | 42 | 34 | 26 | 18 | 10 | 2 |
| 60 | 52 | 44 | 36 | 28 | 20 | 12 | 4 |
| 62 | 54 | 46 | 38 | 30 | 22 | 14 | 6 |
| 64 | 56 | 48 | 40 | 32 | 24 | 16 | 8 |
| 57 | 49 | 41 | 33 | 25 | 17 | 9 | 1 |
| 59 | 51 | 43 | 35 | 27 | 19 | 11 | 3 |
| 61 | 53 | 45 | 37 | 29 | 21 | 13 | 5 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 | 7 |

در جدول بالا پس از عمل جایگشت, بیت اول به موقعیت بیت پنجاه وهشتم و بیت دوم به بیت پنجاهم از رشته جدید منتقل شده و بهمین ترتیب ادامه می‌یابد تا رشته ٦٤ بیتی جدید بدست آید.

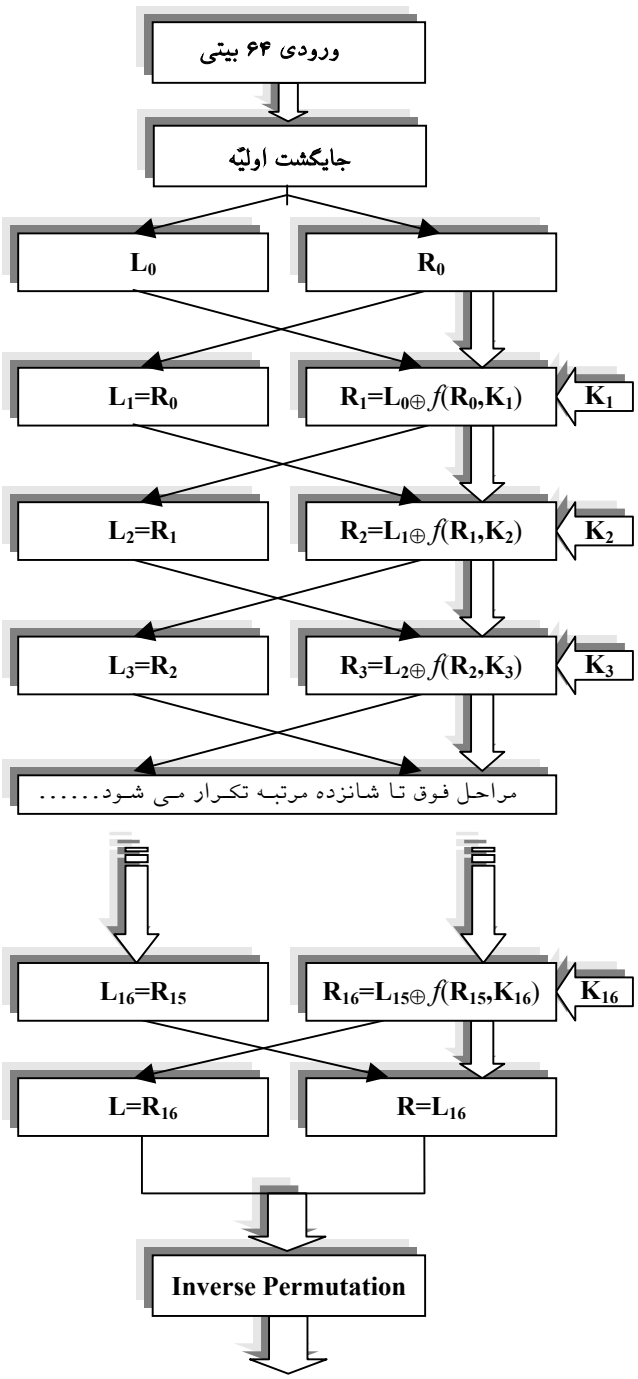
در مرحله بعدی رشته ٦٤ بیتی جدید از وسط به دو قسمت ٣٢ بیتی چپ و راست تقسیم می‌شود. ٣٢ بیت سمت چپ را $L_0$ و ٣٢ بیت سمت راست را $R_0$ می‌نامیم. (به شکل (٦-١١) نگاه کنید)

سپس در ١٦ مرحلهٔ پیاپی اعمال زیر انجام می‌شود:

در هر مرحله ٣٢ بیت سمت راست مستقیماً به سمت چپ منتقل شده و ٣٢ بیت سمت چپ طبق رابطه زیر به یک رشته بیت جدید تبدیل و به سمت راست منتقل خواهد شد.

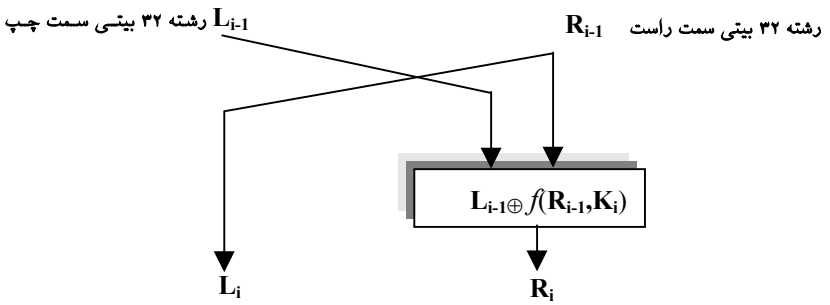
$$\mathbf{L_{i-1} \oplus f(R_{i-1}, K_i)}$$

$L_{i-1}$ رشته سی و دو بیتی سمت چپ از مرحله قبل می‌باشد . علامت $\oplus$ بمعنای XOR و $f$ تابع خاصی است که آنرا به صورت مجزا توضیح خواهیم داد $\mathbf{R_{i-1}}$ رشته سی و دو بیتی سمت راست از مرحله قبل و **Ki** کلید رمز در هر مرحله است. (پس مجموعاً ١٦ کلید مختلف وجود دارد. )




شکل (٦-١١) الگوریتم روش رمزنگاری DES

نمودار زیر یک مرحلـه از ١٦ مرحلـه عملیـات را نشـان می‌دهـد:




ایـن عملیـات ١٦ مرحلـه اجـرا می‌شـود و پـس از مرحلـه آخـر جـای $L_i$ و $R_i$ عـوض خواهد شـد.

حال عکـس عمـل جایگشـتی کـه در ابتـدا انجـام شـده بـود صـورت مـی‌گـیرد تـا بیتـها سرجایشان برگردند این کار طبـق جـدول زیـر انجـام می‌شـود:

| جدول جایگشت معکوس | | | | | | | |
|---|---|---|---|---|---|---|---|
| 40 | 8 | 48 | 16 | 56 | 24 | 64 | 32 |
| 39 | 7 | 47 | 15 | 55 | 23 | 63 | 31 |
| 38 | 6 | 46 | 14 | 54 | 22 | 62 | 30 |
| 37 | 5 | 45 | 13 | 53 | 21 | 61 | 29 |
| 36 | 4 | 44 | 12 | 52 | 20 | 60 | 28 |
| 35 | 3 | 43 | 11 | 51 | 19 | 59 | 27 |
| 34 | 2 | 42 | 10 | 50 | 18 | 58 | 26 |
| 33 | 1 | 41 | 9 | 49 | 17 | 57 | 25 |

پس از ایـن عمـل ، ٦٤ بیـت جدیـد معـادل هشـت کـاراکتر رمـز شـده خواهـد بـود کـه می‌توان آنها را بجای متـن اصلـی ارسـال کـرد.

حال بایستی جزئیات تابع $f$ را کـه اصـل عمـل رمزنگـاری اسـت تعییـن کنیـم:

در تابع $f$ کـه بـه صـورت یـک بلـوک پیـاده سـازی می‌شـود ابـتدا رشـته ۳۲ بیتـی Ri که از مرحله قبل بدست آمده بر طبـق جـدول زیـر بـه یـک رشـته ٤٨ بیتـی تبدیـل می‌شـود. بنابراین بعضی از بیتـها در رشـته جدیـد تکـراری هسـتند .

| جدول بسط ۳۲ به ٤٨ | | | | | |
|---|---|---|---|---|---|
| 32 | 1 | 2 | 3 | 4 | 5 |
| 4 | 5 | 6 | 7 | 8 | 9 |
| 8 | 9 | 10 | 11 | 12 | 13 |
| 12 | 13 | 14 | 15 | 16 | 17 |
| 16 | 17 | 18 | 19 | 20 | 21 |
| 20 | 21 | 22 | 23 | 24 | 25 |
| 24 | 25 | 26 | 27 | 28 | 29 |
| 28 | 29 | 30 | 31 | 32 | 1 |

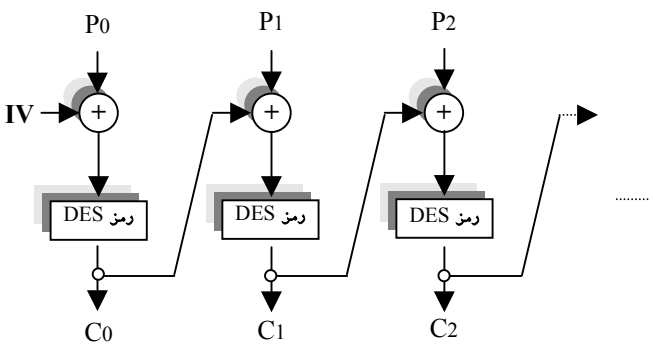
پـس از تبدیـل $R_i$ بـه رشـته ٤٨ بیتـی عمـل XOR روی آن بـا کلیـد ٤٨ بیتـی$K_i$ انجـام می‌شود. نتیجهٔ عمل یـک رشـتهٔ ٤٨ بیتـی اسـت و بایسـتی بـه ۳۲ بیتـی تبدیـل شـود . بـرای اینکار ٤٨ بیت به هشت مجموعـه ٦ بیتـی تبدیـل شـده و هـر کـدام از شـش بیتـی هـا طبـق جداولی به یک چـهار بیتـی جدیـد نگاشـته می‌شـود ( در مجمـوع ۸ جـدول ) . بـرای کـامل شـدن عمـل تـابع $f$ رشـته ۳۲ بیتـی جدیـد طبـق جـدول زیـر جایگشـت مجـددی خواهـد داشـت.

| جایگشت ۳۲ بیتی در تابع $f$ | | | |
|---|---|---|---|
| 16 | 7 | 20 | 21 |
| 29 | 12 | 28 | 17 |
| 1 | 15 | 23 | 26 |
| 5 | 18 | 31 | 10 |
| 2 | 8 | 24 | 14 |
| 32 | 27 | 3 | 9 |
| 19 | 13 | 30 | 6 |
| 22 | 11 | 4 | 25 |

در سیستم DES فقط یک کلید ۵۶ بیتی وجود دارد که تمام ۱۶ کلید مورد نیاز در هر مرحله با جایگشتهای متفاوت از همان کلید ۵۶ بیتی استخراج خواهد شد . بنابراین کاربر برای رمزگشایی فقط باید یک کلید در اختیار داشته باشد و آنهم همان کلیدی است که برای رمزنگاری به کار رفته است .

برای رمزگشائی از سیستم DES دقیقاً مراحل قبلی تکرار می‌شود با این تفاوت که کلید $K_1$ برای رمزگشائی ، کلید $K_{16}$ در مرحله رمزنگاری خواهد بود، کلید $K_2$ همان $K_{15}$ است و به همین ترتیب . در حقیقت برای رمزگشائی کافی است ۱۶ کلید بصورت معکوس به سیستم اعمال شوند.

نکته دیگری که در مورد سیستم DES قابل توجه است آنست که چون رشتهٔ رمز شده بصورت هشت کاراکتری رمز می‌شود ، تکرار بلوکهای رمز می‌تواند به رمزشکنها برای حمله به سیستم DES کمک نماید . بهمین دلیل در سیستم DES قبل از آنکه یک بلوک رمز شود ابتدا با بلوک رمز شدهٔ قبلی خود XOR می‌شود و سپس این ۸ کاراکتر مجدداً رمز خواهد شد. به شکل (۷-۱۱) دقت کنید:




شکل (۷-۱۱) عملیات بین بلوکهای داده در سیستم رمزنگاری DES

بلوک اول با یک رشته ٦٤ بیتی اولیه بنام IV[1] (بردار اولیّه) XOR می‌شود . نتیجه این بلوک کد رمز ٦٤ بیتی است . همین کد رمز برای رمز کردن بلوک بعدی بکار می‌آید ، بدینصورت که بلوک رمز نشدهٔ Pi با بلوک رمز شدهٔ قبلی Ci-1 ابتدا XOR شده و متن جدید مجدّداً رمز خواهد شد.

[1] Initialization Vector

این الگوی رمزنگاری بعنوان استانداردی برای اسناد حسّاس فدرال آمریکا پذیرفته شد تا آنکه در سال ۱۹۷۷ یکی از محققین دانشگاه استانفورد با هزینه ای معادل ۲۰ میلیون دلار ماشینی طراحی کرد که در عرض ۲۴ ساعت می‌توانست رمز DES را بشکند. بعد از آن ایده های جدیدی برای رمزنگاری مطرح شد که DES را در سیستمهای عملی کنار زد.

نکته دیگر آنست که چون کلید رمزنگاری و رمزگشائی هر دو یکی است لذا باید از کلید شدیداً حفاظت شود . در مدلهای جدید کلید رمزنگاری را همه می‌دانند ولی کلید رمزگشائی سرّی است .

## ۸) رمزگذاری کلید عمومی[1]

در هر یک از الگوهای رمزنگاری که مورد بحث قرار گرفتند لازم است که فرستنده پیام و گیرنده پیام کلید رمز را بدانند . وقتی فرستندهٔ پیام از کلیدی برای رمزنگاری استفاده می‌کند و گیرندگان هم از همان کلید برای رمزگشایی بهره می‌برند ، افشا شدن کلید رمز توسط یکی از گیرندگان پیام ، امنیت را به خطر می‌اندازد. در الگوهای جدیدِ رمزگذاری، برای حل مشکل از دو کلید متفاوت استفاده می‌شود . یک کلید برای رمز کردن پیام و کلید دیگر برای رمزگشائی آن . با کلیدِ مخصوص رمزنگاری نمی‌توان رمزگشائی پیام را انجام داد. بنابراین رمزکنندهٔ پیام خودش کلیدی دارد که حتی معتمدین و گیرندگان پیام هم آنرا لازم ندارند چرا که فقط برای رمزنگاری بکار می‌آید و افشا شدن آن هم لطمه ای به کسی نمی‌زند چرا که با آن کلید نمی‌توان متون رمز شده را برگرداند و پیدا کردن کلید رمزگشائی از روی کلید رمزنگاری کار ساده ای نیست. (هنوز امکان پذیر نشده است)

در سال ۱۹۷۸ سه نفر بنامهای ری وست ، شامیر و اَدلمن روشی را برای پیاده سازی "رمزنگاری کلید عمومی" با یک جفت کلید ابداع کردند . این روش که چگونگی آن در زیر تشریح شده است بنام روش RSA (مخفف اسامی آنها) مشهور است و بطرز فزاینده ای از آن استفاده می‌شود:

[1] Public Key Cryptography

روش کار فوق العاده ساده است: دو عدد صحیح (e,n) برای رمزگذاری انتخاب می‌شوند؛ متنی که باید رمز شود به بلوکهایی تقسیم می‌شود . مثلاً کل متن پیام به K تا بلوک تقسیم شده و هر بلوک به نحوی به یک عدد صحیح تبدیل می‌شود . مثلاً کدهای اَسکی هر حرف پشت سر هم قرار می‌گیرند. برای آنکه همین ابتدا بحث را پیچیده نکنیم فرض کنید بخواهیم رشته "M="IDESOFMARCH را رمز کنیم . برای سادگی این رشته را به بلوکهای ۲ کاراکتری تقسیم کرده و سپس هر بلوک را به یک عدد صحیح تبدیل می‌نماییم:

| | | | | | | |
|---|---|---|---|---|---|---|
| **رشته اصلی بلوکهای ۲ کاراکتری** → | ID | ES | OF | MA | RC | HX |
| **تبدیل رشته به شش بلوک** → | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ | $M_6$ |
| **تبدیل بلوک به عدد صحیح** → | 0803 | 0418 | 1405 | 1200 | 1702 | 0723 |
| **بلوکهای جدید عددی** → | $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ | $P_6$ |

روش تبدیل در مثال بالا این بوده که برای A عدد ۰۰ ، B عدد ۰۱ ، ... و Z عدد 25 در نظر گرفته شده و در هر بلوک عدد متناظر با هر کاراکتر پشت سر هم قرار می‌گیرد تا کد بلوک را بسازد . شما می‌توانستید کد اَسکی آن یا هر قاعدهٔ دیگری را به کار ببرید.

در مرحله بعدی جفت عدد صحیح (17,2773) معادل (e,n) برای رمزگذاری بلوکها با استفاده از روش زیر انتخاب می‌شوند:

$$Ci=(P_i)^e \bmod n$$

بلوکهای عددی پس از آنکه به توان e رسید ، باقیماندهٔ تقسیم آن بر n محاسبه می‌شود و بلوکهای Ci بدست می‌آید. بلوکهای Ci کدهای رمز هستند و بجای متن اصلی ارسال می‌شوند . پس در مثال فوق داریم:

| Pi | 0803 | 0418 | 1405 | 1200 | 1702 | 0723 |
|---|---|---|---|---|---|---|
| $Ci=(P_i)^e \bmod n$ | 0779 | 1983 | 2641 | 1444 | 0052 | 0802 |

قبل از آنکه روش رمزگشائی را تشریح کنیم الگوی رمزنگاری RSA را بصورت جمع بندی شده ارائه می‌دهیم:

**الف)** رشته ای که باید رمز شود ، به بلوکهای K کاراکتری تبدیل می‌شود.

**ب)** هر بلوک طبق قاعدهٔ دلخواه بـه یـک عـدد صحیـح تبدیـل می‌شـود. ($P_i$)
**ج)** با جفت عدد صحیح (e,n) بـرای تمـام بلوکـها اعـداد جدیـدی طبـق رابطـه زیـر بدسـت می‌آیـد:

$$C_i=(P_i)^e \bmod n$$

**د)** کدهای $C_i$ ، بجای کـد اصلـی ارسـال می‌شـود.

نکته اساسی در این الگو آنست کـه بـرای رمـزگشـائی کدهـا بـاید عـددی مثـل d پیـدا شـود که در رابطه زیر صـدق کنـد:

$$(x^{e.d}) \bmod n=x$$

با چنین عددی خواهیـم داشـت:

$$P_i =(C_i^d) \bmod n$$

یعنی مشـابه عمـل رمزنگـاری مجـدداً کدهـای رمـز بـه تـوان d رسـیده ، باقیمـاندهٔ آن بـر n محاسبه خواهد شد. کدهای حـاصل دقیقـاً همـان کدهـای اولیـه هسـتند.
به کلید (e,n) کـه بـا آن متـن رمـز می‌شـود "کلیـد عمومـی"[1] و بـه کلیـد (d,n) کـه بـا آن متن از رمز خارج می‌شـود "کلیـد خصوصـی" [2] اطـلاق می‌شـود.

قبل از آنکـه مثـالی دیگـر ارائـه بدهیـم اجـازه بدهیـد روش انتخـاب و معیارهـای e , d را که توسط ابداع کنندگان ایـن روش پیشـنهاد شـده اسـت ، معرفـی کنیـم:
**الف)** دو عـدد اوّل دلخـواه (ولـی بـزرگ) p , q انتخـاب کنیـد. (بـرای کاربردهـای عملـی اگـر ایـن اعـداد صـد رقمـی باشـند اطمینـان بخـش خواهـد بـود - یعنـی از مرتبـه $10^{100}$ باشـد-)
**ب)** عدد n , z را طبق دو رابطـه زیـر محاسـبه نمـاید:

$$n = p \times q$$
$$z = (p-1)(q-1)$$

**ج)** عـدد d را بگونـه ای انتخـاب کنیـد کـه نسـبت بـه z اول باشـد یعنـی هیـچ عـامل مشترکی که هر دو بر آن بخـش پذیـر باشـند نداشـته باشـد.
**د)** براساس d عدد e را بگونه ای انتخاب کنیـد تـا رابطـه زیـر برقـرار باشـد:

$$(e \times d) \bmod z = 1$$

---

[1]- Public key
[2]- Private key

نکاتی که در رمزنگاری باید رعایت شود آنست که کدهای $P_i$ که به هر بلوک نسبت می‌دهیم باید $P_i < n$ ? 0 باشد بنابراین اگر بلوکها را بصورت رشته های k بیتی مدل می‌کنید باید شرط $2^k < n$ برقرار باشد.

برای یک مثال آموزشی فرض کنید بخواهیم رشته "SUZANNE" را رمز نمائیم. برای راحتی کار مجبوریم کلیدها را بسیار کوچک بگیریم ولی دقت داشته باشید در عمل اینطور نیست:

**الف)** دو عدد اول p=3 و q=11 را انتخاب می‌کنیم.

**ب)** عدد n=33 و z=20 بدست می‌آیند.

**ج)** عدد 7 که نسبت به z اول است را برای d انتخاب می‌نمائیم.

**د)** باید عدد e بگونه ای پیدا شود که رابطهٔ $(7 \times e) \bmod 20 = 1$ برقرار باشد این عدد را 3 انتخاب کرده ایم. ( عدد ۲۳ هم قابل قبول است )

پس داریم :

(3,33)=(e,n) کلید عمومی

(7,33)=(d,n) کلید خصوصی

برای آشنایی با مراحل کار به شکل (۸-۱۱) دقت نمائید. بدلیل آنکه n عدد کوچکی است و باید $P_i<33$ باشد، مجبوریم بلوکها را یک کاراکتری فرض کرده و به A عدد ۱ ، به B عدد ۲ نسبت داده و بهمین ترتیب کاراکترها را به عدد صحیح تبدیل نمائیم.

| سمبولهای متن | عدد $P_i$ | محاسبهٔ $P^3$ | $P^3 \bmod 33$ | محاسبهٔ $C^7$ | $C^7 \bmod 33$ |
|---|---|---|---|---|---|
| S | 19 | 6859 | 28 | 13492928512 | **19** |
| U | 21 | 9261 | 21 | 1801088541 | **21** |
| Z | 26 | 17576 | 20 | 1280000000 | **26** |
| A | 01 | 1 | 1 | 1 | **1** |
| N | 14 | 2744 | 5 | 78125 | **14** |
| N | 14 | 2744 | 5 | 78125 | **14** |
| E | 05 | 125 | 26 | 8031810176 | **5** |

**رمزنگاری** **رمزگشایی**

شکل (۸-۱۱) مثالی از رمزنگاری و رمزگشائی RSA

همانگونه که اشاره شد در عمل p و q صد رقمی انتخاب می‌شوند. (یعنی $(q \approx 10^{100}, P \approx 10^{100})$ بنابراین مقدار n از مرتبه $10^{200}$ (دویست رقمی ) خواهد بود. سؤال آنست که عدد صحیح مربوط به بلوک های $P_i$ که باید از n کوچکتر باشند چند بیتی خواهند بود؟

$$n < 10^{200} \text{ و } (10^{200} \approx 2^{664}) \Rightarrow n < 2^{664}$$

پس هر بلوک متن بایستی حداکثر 664 بیت یا معادل 83 کاراکتر هشت بیتی باشد.

ممکن است تاکنون ذهن شما مشغول این نکته شده باشد که چگونه می‌توان اعداد با این عظمت را به توان رساند. نکته ظریفی که وجود دارد آنست که برای محاسبهٔ $P^e \bmod n$ لازم نیست که اول P به تعداد e بار در خودش ضرب شود و بعد باقیمانده آن بر n بدست آید بلکه می‌توان پس از انجام یکبار عمل ضرب ، پیمانهٔ n ( mod n ) آن هم محاسبه شود تا نتیجهٔ محاسبه کاهش مقدار داشته باشد. برای روشن شدن قضیه به الگوی زیر دقت کنید:

$$7^3 \bmod 5 = ((7 \bmod 5) * 7^2) \bmod 5 = (2 * 7^2) \bmod 5 = ((2*7 \bmod 5) * 7) \bmod 5 = ((4*7) \bmod 5 = 3$$

فرض کنید بخواهیم A را به توان E برسانیم و بسط E در مبنای دودوئی بصورت زیر باشد:

$$E = (e_{k-1}, ....., e_0)_2 = \sum_{i=0}^{k-1} e_i 2^i$$

پس داریم :

$$A^E = A^{\sum_{i=0}^{k-1} e_i 2^i} = A^{2^{k-1}.ek-1} \times ..... \times A^{2^{e1}} \times A^{e0}$$

بنابراین برای رساندن A به توان E می‌توان براساس بسط باینری عدد E عمل کرد. این بسط دودویی مشکل رشد بی نهایت حاصل را حل نخواهد کرد. مشکل زمانی حل خواهد شد که ما بخواهیم $A^E \bmod n$ را محاسبه کنیم که دراین حالت باید پس از هر بار به توان رساندن ، باقیماندهٔ حاصل را بر n محاسبه کنیم تا نتیجه به زیر n کاهش یابد سپس ادامه می‌دهیم تا اعداد رشد بی نهایت نکند.

الگوریتم زیر حاصل $P^E \bmod n$ را محاسبه می‌کند و در Y بر می‌گرداند:

**الف )** نمایش دودوئی E بصورت $E = e_{k-1}e_{k-2}.....e_0$ مشخص می‌شود.

**ب )** $y \leftarrow 1$

**ج)** از شمارنده k-1 تا صفر بصورت شمارش معکوس دو عمل زیر تکرار می‌شود (i شمارنده است )

- y=(y*y) mod n

- اگر $e_i$ مساوی 1 است آنگاه $y=(y*P) \bmod n$

**د)** نتیجه در y قرار دارد.

اگر دقت کافی داشته باشید الگوریتم فوق با مثال قبلی ( $7^3 \bmod 5$ ) معادل خواهد بود. بنابراین مشکل حادّی در عملیات محاسبه کدهای رمز RSA و همچنین رمزگشائی آن وجود ندارد

به یاد داشته باشید که کلید رمزگذاری (e,n) یک کلید عمومی است و دلایلی بر سرّی و محرمانه ماندن آن وجود ندارد در حالی که کلید رمزگشائی (d,n) کلید اختصاصی است و باید سرّی باشد. برای شکستن رمز RSA باید مقدار d را از (e,n) به دست آورد و برای بدست آوردن d ابتدا باید n را به عوامل اول [1] تجزیه کرد تا بتوان p ، q و z و نهایتاً d را بدست آورد. با توجه به آنکه n معمولا دویست رقمی است با کامپیوترهای معمولی برای تجزیه چنین عددی چهار میلیون سال طول خواهد کشید!

به جدول (۹-۱۱) نگاه کنید فرض کنید کامپیوتری هر عمل را در یک میکروثانیه انجام بدهد این جدول زمان تجزیه یک عدد را به عوامل اوّل بر حسب تعداد ارقامِ عدد مشخص کرده است.

| زمان محاسبه | تعداد ارقام |
|---|---|
| ٤ ساعت | ٥٠ |
| ١٠٤ روز | ٧٥ |
| ٧٤ سال | ١٠٠ |
| چهار میلیون سال | ٢٠٠ |
| $٥*١٠^{١٥}$ سال | ٣٠٠ |
| $٤*١٠^{٢٥}$ سال | ٥٠٠ |

جدول (۹-۱۱) زمان لازم برای تجزیه یک عدد به عوامل اوّل

[1] Prime factors

گرچه تحقیق بر روی تجزیهٔ اعداد به عوامل اول ادامه دارد ولی هیچ الگوریتم کارآمدتری که بتواند زمانهای جدول فوق را کاهش بدهد پیدا نشده است و بهمین دلیل بطور فراگیر از آن استفاده می‌شود.

## ۹) احراز هوّیت[1]

"احراز هوّیت" در شبکه های کامپیوتری بدین معناست که یک سرویس دهنده بتواند تشخیص بدهد کسی که تقاضائی را روی آن سیستم دارد شخص مجازی است یا یک شیّاد است . بعنوان مثال فرض کنید A می‌خواهد سعی کند از حسابش در بانکی مقداری پول را به حسابی دیگر منتقل نماید؛ سرویس دهندهٔ بانک باید مطمئن شود کسی که ادعا می‌کند شخص A است حقیقی است یا یک شیّاد است که خود را بجای A جا زده است چرا که با استفاده از تکنیکهائی می‌توان اطلاعاتی را که در قالب اسم رمز و کلمه عبور روی شبکه منتقل می‌شود دزدید و از آن استفاده کرد. تکنیکهای بهتری برای احراز هویت مبتنی بر اصول رمزنگاری با کلید عمومی وخصوصی قابل پیاده سازی است.

در مدل اوّل برای احراز هوّیت روشی را معرفی می‌کنیم که مبتنی بر اصول رمزنگاری و کلید رمز است . در این روش کلید رمزنگاری و رمزگشایی مشترک است:

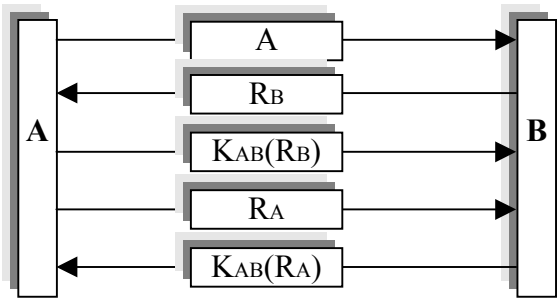
فرض کنید دو برنامه A , B می‌خواهند قبل از تبادل هر گونه اطلاعاتی هوّیت یکدیگر را تصدیق کرده و بعد انتقال داده‌ها را انجام بدهند. (A برنامهٔ سرویس دهنده و B برنامهٔ مشتری است) A مشخصه ای مثل شماره شناسائی یا کلمه عبور برای معرفی خود به B دارد که احتمال فاش شدن آن هم وجود دارد چرا که روی شبکه مبادله می‌شود و ممکن است استراق سمع شود.

حال به روند زیر دقت کنید:

فرض کنید A بعنوان مشتری ، شروع کنندهٔ ارتباط و B بعنوان سرویس دهنده پذیرنده ارتباط است . بعد از برقراریِ ارتباط و قبل از مبادله هر گونه داده ، عملیات

[1] Authentication

زیر مطابق شکل (۱۰-۱۱) صورت میگیرد. (در این روش A و B بر روی کلید رمز مشترکی توافق کرده اند که نام آنرا $K_{AB}$ فرض کرده ایم و کاملاً سرّی است)




شکل (۱۰-۱۱) احراز هوّیت با کلید مشترک

**الف)** A با ارسال مشخصه شناسائی ( ID ) ، خود را به B معرفی می‌کند.

**ب )** B در پاسخ ، یک عدد تصادفی بزرگ مثلاً (صدرقمی) تولید کرده و برای A می‌فرستد. این عدد تصادفی در شکل با $R_B$ مشخص شده است.

**ج)** A با کلید مشترک عدد دریافتیِ $R_B$ را رمز کرده و برای B پس می‌فرستد. عدد رمز شده $K_{AB}(R_B)$ نامیده شده است.

**د)** B پس از دریافتِ عدد رمز شده آنرا با کلید مشترک رمزگشائی کرده و پس از مقایسه با عدد اصلی یعنی $R_B$ می‌تواند مطمئن شود که هوّیت A محرز است و کسی قصد فریب ندارد.

**ه)** چون A هم تمایل دارد هویت طرف مقابل خود را تشخیص بدهد بنابراین او هم یک عدد تصادفی بزرگ تولید کرده و برای B می‌فرستد. B با کلید مشترک آنرا رمز کرده پس می‌فرستد؛ A هم با رمزگشائی و مقایسهٔ آن هوّیت B را تشخیص می‌دهد.

نکته مهم در روش های احراز هویت ، آنست که بر خلاف شماره شناسایی و کلمه عبور ، کلید رمز روی کانالهای ارتباطی ارسال نمی‌شود و توسط خود کاربر حفظ می‌شود تا مسئله افشای کلید از طریق استراق سمع ممکن نباشد؛ به روش فوق "احراز هویت دو مرحله ای"[1] گفته می‌شود.

[1] 2-Way Authentication

روش مشابه دیگری برای احراز هوّیت با استفاده از کلید عمومی [۱] وجود دارد که مبتنی بر روش رمزنگاری RSA است. برای تشریح این روش مجدداً فرض کنید A بعنوان برنامهٔ مشتری و B بعنوان سرویس دهنده تمایل دارند قبل از هر گونه مبادله اطلاعات ، هویت همدیگر را تصدیق نمایند. در این روش هریک از طرفین یک کلید سرّی و یک کلید عمومی دارند. A و B کلید عمومی همدیگر را می‌دانند ولی هرگز از کلید سرّی یکدیگر مطلع نیستند. ثابت شده که این روش صد تا هزار بار از روش قبلی سریعتر است. مراحل احراز هوّیت بصورت زیر است:

**الف )** A بعنوان شروع کننده ، شمارهٔ شناسائی خود و همچنین یک عدد تصادفی بزرگ $R_A$ را با استفاده از کلید عمومیِ B به روش RSA رمز کرده و برای B می‌فرستد.

**ب)** سرویس دهندهٔ B که دارنده کلید سرّی خودش است آنرا رمزگشائی کرده و ضمن استخراج مشخصهٔ طرف مقابل و عدد $R_A$ ، خودش عدد تصادفی $R_B$ را تولید کرده و بهمراه یک کلید سرّی دیگر (یعنی جمعاً سه آیتم $K_S$ , $R_B$ , $R_A$ ) با استفاده از کلید عمومیِ A رمز کرده و برای A پس می‌فرستد. به کلید Ks که بعنوان کلید رمزنگاری جدید برای ادامه ارتباط مورد استفاده قرار می‌گیرد "کلید جلسه" [۲] گفته می‌شود.

**ج)** وقتی A اطلاعات رمز شدهٔ قبلی را از B دریافت و با استفاده از کلید سرّی خود رمزگشایی کرد ، اولا $R_A$ را دارد که با مقایسهٔ آن متوجه می‌شود طرف مقابل همانی است که باید باشد . ثانیا کلید جدید Ks را دارد که رمزنگاری اطلاعات در مراحل بعد باید با آن کلید باشد ، ثانیا $R_B$ را استخراج کرده است که باید با استفاده از کلید Ks مجدّداً آنرا رمز کرده برای A پس بفرستد. سرویس دهندهٔ B با دریافت آن و رمزگشائی هوّیت A را احراز می‌کند.

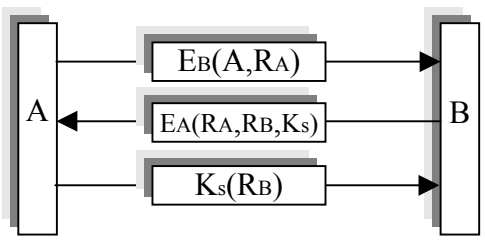
احراز هوّیت بر این مبناست که اگر B دروغگو باشد پس کلید سرّی لازم برای رمزگشایی اطلاعات ارسالی در مرحلهٔ الف را ندارد و اصلاً نمی‌تواند بفهمد چه کسی تقاضا داده و مراحل احراز هویّت ادامه نخواهد یافت. اگر A دروغگو باشد پس کلید سرّی لازم برای رمزگشایی اطلاعات در مرحلهٔ ب را نداشته و قادر به استخراج

---

[۱] Public key
[۲] Session key

"کلید جلسه" نبوده و بنابراین در اجرای مرحلهٔ سوم از احراز هویّت موفّق نخواهد شد.

در مرحلهٔ ۱ و ۲ از شکل (۱۱-۱۱) عملیات رمزنگاری با استفاده از کلید عمومی که علنی است انجام می‌شود ولی در مرحلهٔ ۳ رمزنگاری با کلید جدیدی انجام می‌شود که طرفین از آن اطلاع دارند. (یعنی کلید Ks یا همان کلید جلسه)
مراحل سه گانه این روش در شکل (۱۱-۱۱) مشخص شده است.




شکل (۱۱-۱۱) احراز هویت با استفاده از کلید عمومی و روش RSA

## ۱۰) امضاهای دیجیتالی [1]

یکی دیگر از منافع روش رمزگذاری RSA امضای دیجیتالی است . امضای دیجیتالی همانند امضای معمولی ابزاری است که برای رسمیت بخشیدن به یک پیام یا نامه استفاده می‌شود. با استفاده از امضاهای دیجیتالی :

- گیرندهٔ یک پیام بوسیله آن می‌تواند هوّیت فرستندهٔ آنرا تصدیق نماید.
- فرستندهٔ پیام نمی‌تواند محتوای پیام ارسالی اش را انکار کند.
- گیرندهٔ پیام نمی‌تواند پیامهای جعلی بسازد و همچنین دیگران قادر به جعل پیام نیستند.

[1] Digital Signature

امروزه با محوّل شدن امور مالی همانند حسابهای بانکی و کارتهای اعتباری به شبکه ها و ناامنی شبکه در مبادلهٔ اسناد و همچنین ناکارآمدی امضاهای دست نوشته روشی برای رسمیت بخشیدن به پیامها (یا فرامین ) در شبکه وضع شده که حتی می‌تواند در دادگاه مورد استناد قرار گیرد. (شاید بتوان امضاهای دست نوشته را جعل کرد ولی در مورد امضاهای دیجیتالی هنوز امکان پذیر نشده است)

فرض کنید به یک سرویس دهندهٔ بانکی متصل شده و فرمان می‌دهید تا مقداری پول از حسابتان به حساب دیگری واریز شود. در اینجا فقط تشخیص و احراز هوّیت کافی نیست بلکه باید همانند امضاء روشی وجود داشته باشد که شما نتوانید در آینده اقامه دعوا کرده و عنوان کنید هیچگاه چنین تقاضایی نکرده اید.

روشهای متفاوتی برای پیاده سازی امضاهای دیجیتالی وجود دارد که دو مورد از آنها را معرفی می‌کنیم:

### ۱-۱۰) امضا با کلید سرّی [1]

در این روش که باید با کمک دولت یا انجمن حقوقدانان یا بانکها پیاده سازی شود ، مرکزی وجود دارد که معتمد همه است و قانون از آن حمایت می‌نماید. هر شخص با مراجعه حضوری به آن مرکز و تنظیم تعهّدنامه های لازم برروی یک کلید سرّی توافق می‌کند. (این مرکز را در ذهن خود محضر رسمی گواهی امضاء فرض کنید) بنابراین فقط شخص و مرکز مورد نظر آن کلید رمز را می‌دانند. حال فرض کنید که شخص A بخواهد سندی را در قالب یک پیام متنی امضا کرده برای B بفرستد (مثلاً سند تقاضای جابجایی پول از حساب بانکی ) A دارای کلید رمز $K_A$ است و بنابراین آیتمهای زیر را با کلید خودش رمز کرده و به همراه شماره شناسایی خود به سرویس دهندهٔ مرکز گواهی امضاء که فعلا آنرا BB می‌نامیم ارسال می‌نماید. آیتمهائی که رمز می‌شوند عبارتند از :

B : مشخصه شناسائی گیرنده نهائی پیام

$R_A$ : یک عدد تصادفی بزرگ

[1] Secret key signature

t : زمان دقیق صدور پیام (تاریخ+زمان) معمولا زمان بر حسب گرینویچ –GMT– است .

P: متن پیام

حال مرکز گواهی امضای دیجیتالی که کلید سری A را در اختیار دارد متن و آیتم ها را رمزگشائی کرده و در صورتی که اینکار موفقیت آمیز انجام شد عملا هویت A تائید شده است چرا که هیچکس غیر از این دو کلید رمز را نمی‌داند.

در ادامهٔ این روند ، مرکز گواهی امضاء ضمن ثبت این تقاضا ، آیتمهای زیر را با کلید مشترک توافق شده بین خودش و B ، که کاملاً سرّی است ، برای B ارسال می‌نماید؛ این کلید را $K_B$ فرض نمایید.

A : مشخصه فرستنده پیام

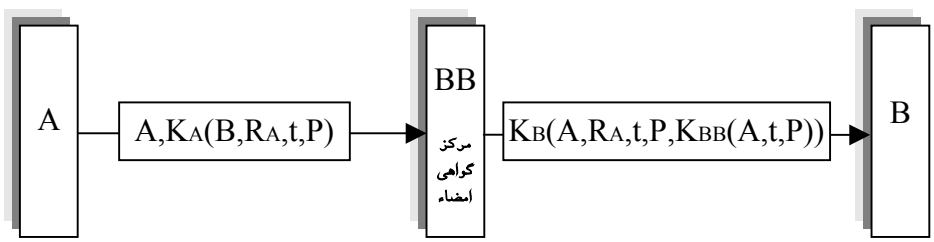
$R_A$ : عدد ارسالی از A

t : زمان دقیق صدور پیام (تاریخ + زمان)

P : متن اصلی پیام ارسالی از A

$K_{BB}(A,t,p)$ : آیتمهای رمز شدهٔ P,t,A که با کلیدی کاملاً سرّی رمز شده اند. این کلید را که فقط و فقط مرکز گواهی امضاء در اختیار دارد $K_{BB}$ فرض نمایید.

پس از رمزگشایی پیام در B تمام آیتمها برای استناد قانونی ذخیره شده و می‌توان به محتوای پیام یا تقاضا عمل کرد. شمای کلّی روش امضای دیجیتالی با کلید سرّی در شکل (۱۲–۱۱) نشان داده شده است.




شکل (۱۲–۱۱) امضای دیجیتالی با کلید سرّی

اگر زمانی A منکر ارسال پیام P شود و ادعا کند پیام ساختگی است ، B می‌تواند متن رمز شده $K_{BB}(A,t,P)$ را به همراه متن اصلی پیام و $R_A$ به دادگاه ارائه کند. کلید $K_{BB}$ در اختیار مرکز گواهی امضاء است که مورد اعتماد دادگاه می‌باشد. مرکز گواهی امضاء متن رمز شدهٔ $K_{BB}(A,t,P)$ را رمزگشائی کرده و با اصل پیام مورد دعوا مطابقت می‌دهد و اگر مطابق بود B تبرئه می‌شود!

### ۲-۱۰) امضای دیجیتالی با کلید عمومی[1]

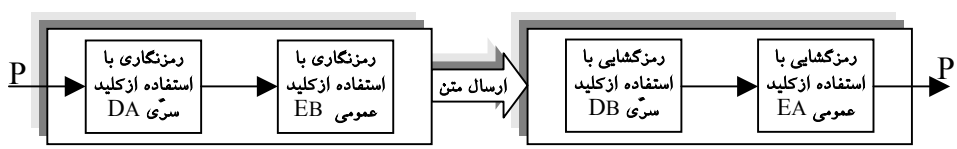
در این بخش روش ساده تری را که نیاز به مرکز واسطه ندارد و از رمزنگاری RSA استفاده میکند ، معرفی میکنیم. ساختار کلّیِ این روش در شکل (۱۳-۱۱) نشان داده شده است.

فرض کنید A و B می‌خواهند با هم ارتباط رسمی داشته باشند. A در رمزنگاری RSA دو کلید برای خود تعریف می‌کند: کلید $D_A$ که سرّی و خصوصی است و کلید $E_A$ که عمومی است. $E_A$ را B می‌داند ولی $D_A$ را فقط خودش خبر دارد.

B هم برای خود دو کلید تعریف می‌کند کلید $D_B$ بعنوان کلید سرّی و کلید $E_B$ که عمومی است. $E_B$ را A هم می‌داند چون کلیدی عمومی است ولی $D_B$ را فقط خودش خبر دارد.

وقتی A می‌خواهد متنی را برای B بفرستد اول آن را با کلید خصوصی اش یعنی $D_A$ رمز می‌کند تا متن رمز شده $D_A(P)$ بدست آید. متن رمز شده جدید را مجدداً با کلید عمومی $E_B$ رمز کرده و نتیجه را برای B می‌فرستد.

در B ابتدا متن دریافتی با کلید خصوصی یعنی $D_B$ از رمز درآمده و مجدداً با کلید عمومی $E_A$ رمزگشائی می‌شود تا متن اصلی بدست آید.




شکل (۱۳-۱۱) روش امضای دیجیتالی با کلید عمومی

[1] Public Key Signature

اصول کار این روش بر دو مورد زیر استوار است:

- اگر B جعلی و دروغین باشد هر چند می‌تواند کلید عمومی EA را داشته باشد ولی بهیچوجه کلید خصوصی B را نداشته و قادر به رمزگشایی متن نخواهد بود.
- اگر A جعلی و دروغین باشد چون کلید خصوصی A را ندارد بنابراین نخواهد توانست متن را رمز کند و اگر از کلید جعلی استفاده کند قابل بازیابی و رمزگشائی نخواهد بود.

در مجموع استفاده از امضاهای دیجیتالی به طرز فزاینده ای در حال رواج یافتن است ولی هنوز بسیاری از کشورها قوانینی در حمایت از آن وضع نکرده اند و استناد قانونی به چنین امضاهایی هنوز با مشکلاتی روبرو است.

## ۱۱) مراجع این فصل

مجموعهٔ مراجع زیر می‌توانند برای دست آوردن جزییات دقیق و تحقیق جامع در مورد مفاهیم معرفی شده در این فصل مفید واقع شوند.


| | |
|---|---|
| "Computer Networks" , Andrew S.Tanenbaum, Third Edition, Prentice-Hall, 1996. | |
| RFC1244 | "Site Security Handbook" |
| RFC1115 | "Privacy Enhancement for Internet Electronic Mail: Part III—Algorithms, Modes, and Identifiers [Draft]," Linn, J.; 1989 |
| RFC1114 | "Privacy Enhancement for Internet Electronic Mail: Part II—Certificate-Based Key Management [Draft]," Kent, S.T.; Linn, J.; 1989 |
| RFC1113 | "Privacy Enhancement for Internet Electronic Mail: Part I—Message Encipherment and Authentication Procedures [Draft]," Linn, J.; 1989 |
| RFC1108 | "Security Options for the Internet Protocol," 1991 |