فرض کنید امکان کابل‌کشی مستقیم بین هر جفت پریز، چه عمودی، افقی یا مورب وجود داشته باشد. برای اتصال این پریزها به هم، به چند متر کابل نیاز است اگر:

الف) پیکربندی مبتنی بر «ستاره» و با یک مسیریاب در وسط، مدنظر باشد.

ب) شبکهٔ محلی 802.3 مدنظر باشد.

۱۶. نرخ تغییرات سیگنال (یعنی Baud Rate) در استاندارد ده‌مگابیتی اترنت چقدر است؟

۱۷. سیگنال حاصل از کدینگ «منچستر» را برای رشته بیت 0001110101 نشان بدهید.

۱۸. سیگنال حاصل از کدینگ «منچستر تفاضلی» را برای رشته بیت مسئله بالا ترسیم نمائید. فرض کنید خط در هنگام شروع به ارسال در حالت LOW قرار داشته باشد.

۱۹. در یک شبکهٔ محلی 10Mbps مبتنی بر CSMA/CD (البته نه 801.3) با طول یک کیلومتر، سرعت انتشار سیگنال ۲۰۰ متر بر میکروثانیه است. طول فریمهای داده ۲۵۶ بیت شامل مجموعاً ۳۲ بیت سرآیند، کد کشف خطا و سربارهای دیگر است. اولین برش بیتی (Bit Slot) پس از ارسال موفق، برای گیرندهٔ فریم در نظر گرفته شده تا بتواند برای ارسال پیام ACK (تصدیق دریافت فریم) کانال را در اختیار بگیرد. نرخ ارسال مفید (بدون در نظر گرفتن سربار و با فرض عدم وجود تصادم) چقدر است؟

۲۰. دو ایستگاه CSMA/CD تلاش می‌کند تا یک فایل بزرگ (شامل چندین فریم) را ارسال نمایند. پس از ارسال هر فریم، آنها برای دسترسی به کانال طبق الگوریتم «عقب‌گرد نمایی» با یکدیگر رقابت می‌کنند. احتمال اینکه هر رقابت در k دور به اتمام برسد و همچنین تعداد متوسط دورها (rounds) در هر رقابت چقدر است؟

۲۱. ساختمانی را در نظر بگیرید که در آن شبکه‌ای CSMA/CD با سرعت 1Gbps و یک کیلومتر کابل، بدون هیچ تکرارکننده (Repeater) کار می‌کند. سرعت انتشار سیگنال روی کابل ۲۰۰۰۰۰ کیلومتر بر ثانیه است. مقدار حداقل طول هر فریم چقدر است؟

۲۲. طول یک بسته IP جهت ارسال بر روی اترنت، ۶۰ بایت (شامل کل سرآیندها) است. اگر از LLC استفاده نشده باشد آیا در فریم اترنت به عمل اضافه کردن داده‌های زائد (Padding) نیاز است و اگر نیاز است چند بایت؟

۲۳. طول فریمهای اترنت باید حداقل ۶۴ بایت باشد تا این اطمینان حاصل شود که در صورت تصادم در انتهای کابل، فرستنده کماکان در حال ارسال است.[تا بتواند قبل از خاتمهٔ فریم تصادم را کشف کند.] حداقل طول فریمها در «اترنت سریع»، ۶۴ بایت است در حالی که ارسال بیتها برروی خروجی ۱۰ برابر سریعتر شده است. چگونه ممکن بوده که بتوان طول حداقل فریمها را در همان ۶۴ بایت نگه داشت؟

۲۴. در برخی از کتابها طول حداکثر هر فریم اترنت ۱۵۱۸ بایت ذکر شده است. آیا اشتباهی در کار است؟ پاسخ خود را تشریح نمایید.

۲۵. در تشریح مشخصات 1000Base-SX بیان شده که سیگنال ساعت در فرکانس ۱۲۵۰ MHz کار می‌کند، در حالیکه در اترنت گیگابیت فرض بر آن است که نرخ تحویل داده‌ها 1Gbps است. آیا این سرعت بالاتر، برای ایجاد حاشیهٔ اطمینان بیشتر بوده است؟ اگر نه، چه نکته‌ای در کار است؟

۲۶. اترنت گیگابیت از عهدهٔ ارسال یا دریافت چند فریم در هر ثانیه بر می‌آید؟ به دقت بیندیشید و تمام حالات را در نظر بگیرید. راهنمایی: واقعیاتی که در شبکهٔ اترنت گیگابیتی وجود دارد را مدّ نظر قرار بدهید.

۲۷. دو شبکه را نام ببرید که اجازه می‌دهند فریمها بطور پشت سرهم و در یک بار ارسال انتقال یابند. به چه دلیل چنین خصوصیتی ارزشمند است؟

۲۸. در شکل ۴-۲۷ چهار ایستگاه A ، B ، C و D نشان داده شده‌اند. به نظر شما کدام یک از ایستگاه‌ها به A نزدیکتر هستند؟

۲۹. فرض کنید که در یک شبکه محلی بی‌سیم 802.11b [پس از در اختیار گرفتن کانال] تعدادی فریم پشت سر هم به طول ۶۴ بایت بر روی کانال با نرخ خطای 10^{-7} منتقل گردد. به طور متوسط چند فریم در هر ثانیه در اثر خطا آسیب خواهد دید؟

۳۰. یک شبکه 802.16 دارای کانالی با عرض باند 20MHz است. در هر ثانیه چند بیت برای یک ایستگاه قابل ارسال خواهد بود؟

۳۱. استاندارد IEEE 802.16 از چهار رده خدمات پشتیبانی می‌کند. کدامیک از این رده‌های خدمات، بهترین انتخاب برای ارسال تصاویر ویدیوئی **فشرده‌نشده** می‌باشد؟

۳۲. دو دلیل بیاورید که چرا برخی شبکه‌ها ممکن است از کدهای تصحیح خطا به جای فرآیند کشف خطا و ارسال مجدد استفاده نمایند.

۳۳. در شکل ۴-۳۵ می‌بینیم که یک ابزار مبتنی بر بلوتوث می‌تواند به طور همزمان درون دو پیکونت قرار داشته باشد. آیا دلیلی وجود دارد که یک ابزار نتواند بطور همزمان نقش گره اصلی (Master) را در هر دو پیکونت ایفا کنند؟

۳۴. شکل ۴-۲۵ ، چندین پروتکل لایهٔ فیزیکی را نشان می‌دهد. کدام یک از آنها به پروتکل لایهٔ فیزیکی در بلوتوث نزدیکتر است؟ کدام یک از آنها بیشترین اختلاف را دارد؟

۳۵. بلوتوث از دو نوع لینک بین گره اصلی (Master) و گره‌های پیرو (Slave) پشتیبانی می‌کند. این دو کدام است و هر کدام برای چه کاری در نظر گرفته شده‌اند؟

۳۶. فریمهای Beacon در گونه‌ای از شبکهٔ 802.11 که مبتنی بر روش «طیف گسترده با پرش فرکانسی» (Frequency Hopping Spread Spectrum) است زمانی به نام Dwell time را مشخص می‌کند. آیا به نظر شما در فریمهای مشابه Beacon در بلوتوث نیز زمانی به نام dwell time وجود دارد؟ پاسخ خود را شرح دهید.

۳۷. به شبکه‌های LAN متصل به هم که در شکل ۴-۴۱ نشان داده شده دقت کنید. فرض کنید ماشینهای میزبان a و b بر روی LAN 1 و c برروی LAN 2 و d برروی LAN 8 واقع هستند. در ابتدا جداول Hash پلها خالی است و «درخت پوشای» نشان داده شده در شکل ۴-۴۱-ب به کار گرفته شده است. نشان دهید که جداول پلهای مختلف در اثر هر یک از رخدادهای زیر که به ترتیب حروف الفبا اتفاق می‌افتند چگونه تغییر می‌کند.

الف) a برای d ارسال می‌کند.

ب) c برای a فریمی می‌فرستد.

ج) d برای c فریمی می‌فرستد.

د) d به LAN 6 نقل مکان می‌کند.

ه) d برای a فریمی می‌فرستد.

۳۸. یکی از تبعات استفاده از «درخت پوشا» برای هدایت فریمها در یک LAN گسترش‌یافته آن است که برخی از پلها در فرآیند هدایت فریمها نقشی ایفاء نخواهند کرد. در شکل ۴-۴۲ سه تا از اینگونه پلها را مشخص کنید. آیا دلیلی دارد که چنین پلهایی را در شبکه داشته باشیم، حتی وقتی در هدایت فریمها نقشی ایفا نمی‌کنند؟

۳۹. تصور کنید که یک سوئیچ دارای چند «کارت خطّ» (Line Card) است و هر یک از کارتها چهار خط

ورودی دارند. فرض کنید که بطور متناوب فریمهایی که به یکی از خطوط کارت وارد می‌شوند از خط دیگری بر روی همان کارت خارج می‌گردند؛ طراح سوئیچ چه راهکاری در مواجهه با این مسئله پیش رو دارد؟

۴۰. یک سوئیچ که برای به کارگیری در اترنت سریع طراحی شده، دارای یک Backplane است که می‌تواند 10Gbps را منتقل کند. در سنگین‌ترین بار این سوئیچ قادر به انتقال چند فریم در هر ثانیه است؟

۴۱. به شبکهٔ شکل ۴-۴۹-الف دقت کنید. اگر ماشین J به ناگاه «سفید» شود [یعنی به VLAN سفید بپیوندد] آیا تغییری در برچسبها لازم است؟ اگر این چنین است چگونه؟

۴۲. به طور مختصر تفاوت بین سوئیچهای نوع «ذخیره و هدایت» (Store & Forward) و سوئیچهای Cut Through را توضیح بدهید.

۴۳. سوئیچهای نوع «ذخیره و هدایت» در مقایسه با سوئیچهای Cut Through از دیدگاه آسیب رسیدن به فریمها برتر هستند. توضیح بدهید که این برتری از کجا ناشی می‌شود؟

۴۴. برای آنکه شبکه‌های VLAN شروع به کار کنند باید جداول پلها و سوئیچها تنظیم و پیکربندی شوند. اگر در شکل ۴-۴۹-الف به جای استفاده از شبکهٔ مبتنی بر کابل چندانتصالی، در VLAN از هاب استفاده شود آیا باز هم به تنظیم این جداول نیاز است؟ آیا هابها نیز نیاز به پیکربندی جدول دارند؟ چرا بله یا چرا نه؟

۴۵. در شکل ۴-۵۰، شبکهٔ موجود و قدیمی سمت راست به یک سوئیچ سازگار با VLAN متصل شده است؛ آیا امکان دارد در اینجا نیز از یک سوئیچ قدیمی استفاده کرد؟ اگر جواب منفی است چرا؟

۴۶. برنامه‌ای بنویسید تا رفتار پروتکل CSMA/CD را در شبکه اترنت (در شرایطی که پس از ارسال یک فریم، N ایستگاه آماده ارسال هستند) شبیه‌سازی کند. برنامهٔ شما باید زمانهایی را که هر ایستگاه موفق می‌شود ارسال فریم خود را آغاز کند گزارش نماید. فرض کنید که به ازای هر برش ۵۱/۲میکروثانیه‌ای یک تیک ساعت اتفاق می‌افتد و کشف تصادم و ارسال سیگنال نویز (دنبالهٔ Jam پس از کشف تصادم) جمعاً ۵۱/۲ میکروثانیه طول می‌کشد. فریمها می‌توانند بیشترین طول مجاز را داشته باشند.

۵ لایه شبکه

وظیفهٔ لایهٔ شبکه آن است که بسته‌های داده را به هر طریق از مبداء به مقصد برساند. هر بسته برای رسیدن به مقصد ممکن است از چندین مسیریاب (Router) در میانه راه گذر کند. این عملکرد به وضوح با عملکرد لایهٔ پیوند داده (که هدف آن انتقال فریمها از انتهای یک سیم به نقطهٔ دیگر آن است) تفاوت دارد. بنابراین لایهٔ شبکه تحتانی‌ترین لایه‌ای است که با انتقال «انتها به انتها» (End-To-End) سر و کار دارد.

لایهٔ شبکه برای نیل به اهداف خود، موظف است از توپولوژی «**زیرشبکه ارتباطی**»[1] (یا به عبارتی مجموعهٔ تمام مسیریابها) اطلاع داشته باشد و با استفاده از این دانش مسیری مناسب را برگزیند. همچنین باید مراقب باشد تا از تحمیل بار اضافی بر روی برخی از خطوط ارتباطی و مسیریابها (در حالی که برخی دیگر آزاد هستند) اجتناب نماید. نهایتاً وقتی که مبداء و مقصد یک بسته در دو شبکه با پروتکلهای متفاوت و ناسازگار قرار گرفته‌اند مشکلات جدیدی بروز می‌کند که حل آنها بر عهدهٔ لایهٔ شبکه گذاشته شده است. در این فصل به مطالعه و تشریح برخی از این موارد خواهیم پرداخت. تمرکز اصلی ما بر اینترنت و لایهٔ شبکهٔ آن یعنی IP است، هر چند شبکه‌های بی‌سیم را نیز خاطرنشان خواهیم نمود.

۵-۱ مسائل طراحی لایهٔ شبکه

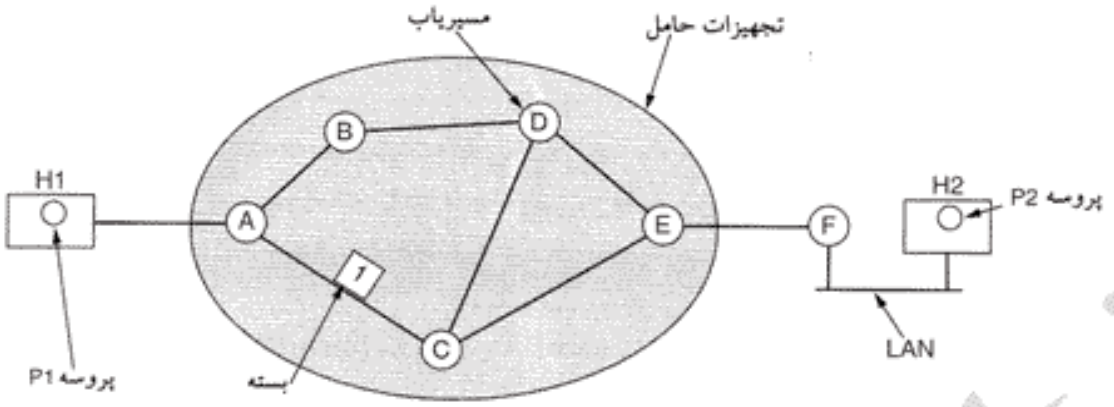
در بخشهای آتی مقدمه‌ای خواهیم داشت بر مسائلی که طراحان لایهٔ شبکه با آنها دست به گریبان خواهند بود. این مسائل شامل خدماتی است که به لایهٔ انتقال ارائه می‌گردد، یا به طراحی داخلی «زیرشبکه» (Subnet) مربوط می‌شود.

۵-۱-۱ هدایت (سوئیچینگ) بسته به روش «ذخیره و هدایت»

قبل از آنکه به تشریح جزئیات لایهٔ شبکه بپردازیم شاید مروری بر حیطه و زمینهٔ عملکرد پروتکلهای لایهٔ شبکه، خالی از لطف نباشد. الگوی کلی چنین محیطی در شکل ۵-۱ دیده می‌شود. مؤلفه‌های اصلی این سیستم عبارتند از: **تجهیزات حامل** (Carrier Equipments) که در این شکل درون بیضی سایه‌دار قرار گرفته‌اند و **تجهیزات مشتریان** (Customer's Equipments) که در خارج از بیضی قرار دارند. ماشین میزبان H1 از طریق یک خط اجاره‌ای، مستقیماً به یکی از مسیریابهای حامل یعنی A، متصل شده است. در مقابل H2 بر روی یک شبکهٔ LAN

---

۱. در این فصل به کرّات از واژهٔ «زیرشبکه» به معنای Communication Subnet (مجموعهٔ تمام مسیریابهای شبکه و خطوط ارتباطی بین آنها) بهره خواهیم گرفت. -م



شکل ۵-۱. محیطی که پروتکلهای لایهٔ شبکه در آن عمل می‌کنند.

قرار گرفته که دارای یک مسیریاب است و تحت فرمان مشتری عمل می‌کند. مسیریاب F، دارای یک خط اجاره‌ای با یکی از «تجهیزات حامل» است. ما در این شکل، مسیریاب F را خارج از بیضی در نظر گرفته‌ایم چرا که عضو بخش حامل [یعنی زیرشبکهٔ اصلی] نیست ولیکن ممکن است از لحاظ ساختار، نرم‌افزار و پروتکل تفاوتی با مسیریابهای حامل نداشته باشد. اینکه آیا چنین مسیریابی متعلق به زیرشبکه هست یا نه قابل بحث است ولیکن با اهدافی که در این فصل مدّنظر داریم مسیریابهای سمت مشتری را نیز به عنوان بخشی از زیرشبکه در نظر می‌گیریم چرا که آنها نیز همانند مسیریابهای حامل از الگوریتمهای مشابهی استفاده می‌کنند (و اهمّ کار ما نیز الگوریتمها هستند).

از این تجهیزات به نحو زیر استفاده می‌شود: یکی از ماشینهای میزبان که بسته‌ای را برای ارسال آماده دارد آن را برای نزدیکترین مسیریاب می‌فرستد، خواه این مسیریاب بر روی LAN خودش واقع باشد و خواه از طریق یک خط اجاره‌ای مستقیماً به زیرشبکهٔ حامل متصل شده باشد. بستهٔ ارسالی در آن مسیریاب موقتاً ذخیره می‌شود تا بطور کامل دریافت و کد کشف خطای آن، بررسی گردد؛ سپس به مسیریاب بعدی واقع بر مسیر، هدایت می‌شود و این روند آنقدر تکرار می‌شود تا بسته به ماشین مقصد رسیده و تحویل داده شود. این مکانیزم همان روش «**ذخیره و هدایت**» است که در فصول قبلی بدانها پرداختیم.

۵-۱-۲ خدمات ارائه شده برای لایهٔ انتقال

لایهٔ شبکه خدماتی را برای لایهٔ انتقال تدارک می‌بیند که توسط واسط میانی لایهٔ شبکه و انتقال[1] عرضه می‌شود. سؤال مهم آن است که لایهٔ شبکه چه نوع خدماتی را برای لایهٔ انتقال تدارک می‌بیند؟ خدمات لایهٔ شبکه با در نظر داشتن اهداف زیر طراحی شده‌اند:

۱. این خدمات بایستی مستقل از تکنولوژی بکار رفته در مسیریاب باشد.
۲. لایهٔ انتقال باید از درگیری با جزئیاتی مثل تعداد، نوع و توپولوژی مسیریابهای موجود دور نگه داشته شود.
۳. آدرسهای شبکه که در اختیار لایهٔ انتقال قرار می‌گیرند بایستی از ساختار متحدالشکل و استاندارد برخوردار باشند (خواه در شبکهٔ LAN و خواه در شبکهٔ WAN).

با در نظر داشتن این اهداف، طراحان لایهٔ شبکه در تدوین شرح مبسوط خدماتی که باید به لایهٔ انتقال عرضه شود آزادی عمل دارند. این آزادی عمل اغلب به مناقشات دیرینهٔ دو گروه با طرز فکر مخالف دامن می‌زند. مناقشه

---

۱. Network Layer/Transport Layer Interface

بر سر آن است که آیا لایهٔ شبکه باید خدمات اتصال‌گرا (Connection Oriented) ارائه کند یا خدمات بدون اتصال (Connectionless).

یکی از طرفین (به رهبری دست‌اندرکاران اینترنت) استدلال می‌کند که وظیفهٔ یک مسیریاب هدایت بسته‌هاست و نه چیز دیگر! از دیدگاه آنها (به پشتوانهٔ سی سال تجربهٔ واقعی با یک شبکه کامپیوتری حقیقی و در حال کار) زیرشبکه ذاتاً غیرقابل اعتماد است و چگونگی طراحی آن اهمیت ندارد. بدین ترتیب ماشینهای میزبان باید این حقیقت را بپذیرند که زیرساخت ارتباطی شبکهٔ آنها غیرقابل اعتماد است و خودشان موظفند عملیات نظارت بر خطاهای احتمالی (شامل کشف و تصحیح) و همچنین کنترل جریان (Flow Control) را بر عهده بگیرند.

این دیدگاه سریعاً بدانجا منتهی می‌شود که خدمات لایهٔ شبکه باید «**بدون اتصال**» بوده و فقط عملیات پایه و ابتدایی SEND PACKET و RECEIVE PACKET و تعداد کمی دیگر از همین عملیات را انجام بدهند. خصوصاً نباید عملیات مرتب‌سازی بسته‌ها [از لحاظ ترتیب ارسال] و کنترل جریان انجام شود چراکه ماشینهای میزبان در هر حال چنین کاری را انجام می‌دهند و انجام دوبارهٔ یک کار فایدهٔ چندانی نخواهد داشت. همچنین هر بسته باید مشخصات کامل آدرس مقصد را با خود داشته باشد چراکه بسته‌های ارسالی باید مستقل از یکدیگر و فارغ از آنکه مسیریاب قبلی کدام بوده، هدایت شوند.

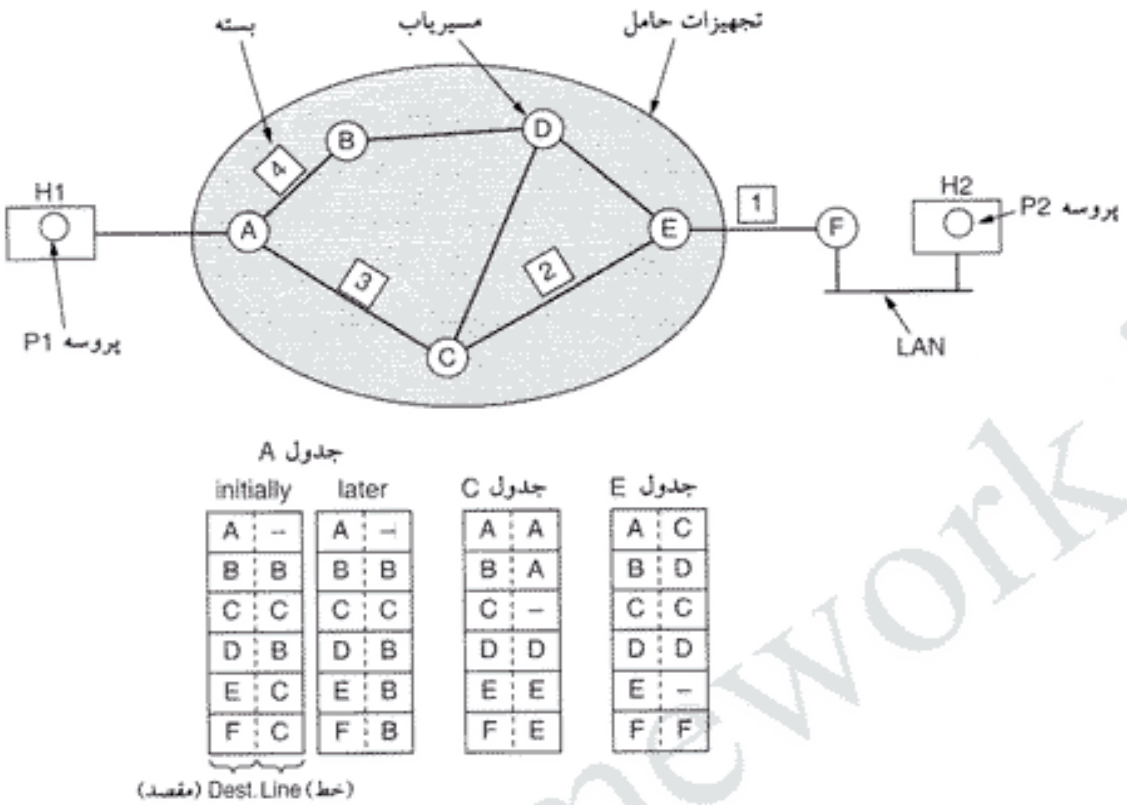
طرف مقابل این مناقشه (به رهبری شرکتهای مخابراتی) دلیل می‌آورد که زیرشبکه باید خدماتی اتصال‌گرا و قابل اعتماد ارائه کند. اینان نیز مدعیند که سابقهٔ درخشان و صد سالهٔ آنها در سیستمهای جهانی تلفن، دلیل عالی و رهنمای راه آنهاست. از دیدگاه این گروه، کیفیت خدمات (QoS) عمده‌ترین مسئله است و اگر زیرشبکه اتصال‌گرا نباشد دستیابی به کیفیت خدمات (بالاخص برای ترافیک داده‌های بی‌درنگ مثل صدا و تصویر) دشوار خواهد بود.

این دو طرف دعوا نمونه‌های خاص خود یعنی اینترنت و ATM را در دست داشتند: اینترنت خدمات لایهٔ شبکه را بدون اتصال عرضه می‌کند در حالی که ATM در لایهٔ شبکه خدماتی اتصال‌گرا ارائه می‌نماید. ولیکن اشاره به این نکته جالب است که هر چه تضمین کیفیت خدمات بیشتر و بیشتر اهمیت پیدا می‌کند، اینترنت نیز رشد می‌نماید. خصوصاً آنکه در راه کسب ویژگیهایی است که عموماً منتسب به خدمات اتصال‌گرا هستند؛ بعداً در این خصوص اشاراتی خواهیم داشت. در خلال مطالعهٔ شبکه‌های VLAN در فصل چهارم نمونه‌ای از این رشد و تکامل را ارائه کردیم.

۵-۱-۳ پیاده‌سازی خدمات بی‌اتصال

پس از نگاهی بر دو رده از خدماتی که لایهٔ شبکه می‌تواند به کاربران عرضه نماید وقت آن فرا رسیده که ببینیم در درون این لایه چه می‌گذرد. بسته به نوع خدمات ارائه شده، دو نوع معماری متفاوت ممکن خواهد بود. اگر خدمات بدون اتصال عرضه شود بسته‌ها بطور مجزا و مستقل به درون زیرشبکه تزریق و مستقل از یکدیگر هدایت و مسیریابی می‌شوند و هیچگونه تنظیمات قبلی نیاز نیست. در چنین حالتی عموماً به بسته‌ها «دیتاگرام» (Datagram) و به زیرشبکه نیز «**زیرشبکه دیتاگرام**» اطلاق می‌شود. اگر خدمات اتصال‌گرا عرضه شود قبل از ارسال هرگونه بستهٔ داده بایستی از قبل یک مسیر بین مسیریاب مبداء و مسیریاب مقصد تنظیم و ایجاد شود. چنین ارتباطی اصطلاحاً VC (مخفف Virtual Circuit) نامیده می‌شود (مترادف با ایجاد یک مدار فیزیکی در شبکهٔ تلفن) و به زیرشبکه نیز، «**زیرشبکهٔ مدار مجازی**» (Virtual Circuit Subnet) گفته می‌شود. در این بخش ابتدا به زیرشبکه‌های دیتاگرام می‌پردازیم و در بخش بعدی زیرشبکه‌های مدار مجازی را مطالعه خواهیم نمود.

حال بیائید به بررسی عملکرد زیرشبکه مبتنی بر دیتاگرام بپردازیم. فرض کنید پروسهٔ P1 در شکل ۵-۲ پیامی طولانی برای پروسهٔ P2 دارد. او این پیام را به لایهٔ انتقال سپرده و از او می‌خواهد که پیام را به پروسهٔ P2 بر روی



شکل ۵-۲. مسیریابی در یک زیرشبکهٔ دیناگرام.

ماشین میزبان H2 تحویل بدهد. کد نرم‌افزار لایهٔ انتقال که عموماً بخشی از سیستم عامل به حساب می‌آید اجرا شده و سرآیند لایهٔ انتقال در ابتدای پیام درج و نتیجه ، تحویل لایهٔ شبکه می‌شود که آن نیز یک پروسهٔ اجرایی (روال اجرایی) در سیستم عامل است.

فرض کنیم که پیام چهار برابر طولانی‌تر از اندازهٔ مجاز بسته‌های داده باشد؛ لذا لایهٔ شبکه مجبور است آنها را به چهار بستهٔ ۱ و ۲ و ۳ و ۴ بشکند و هر یک از آنها را بطور مجزا و از طریق یک پروتکل نقطه‌به‌نقطه (نظیر PPP)[1] برای مسیریاب A می‌فرستد. از اینجا به بعد ادامهٔ کار بر عهدهٔ زیرشبکهٔ حامل قرار می‌گیرد. هر مسیریاب دارای جدولی داخلی است که مشخص می‌کند برای رسیدن به تمام مسیریابهای مقصد در شبکه، بسته باید بر روی کدام خط خروجی [کدام مسیریاب بعدی] ارسال شود. هر یک از «درایه‌های» این جدول (Table Entry) شامل یک جفت آیتم اطلاعاتی است که یکی «مقصد» و دیگری خط خروجی برای رسیدن بدان مقصد را تعیین می‌نماید. برای هدایت بسته فقط می‌توان از خطوطی که مستقیماً به یک مسیریاب مجاور متصلند بهره گرفت. به عنوان مثال در شکل ۵-۲ ، مسیریاب A فقط دو خط خروجی (به مسیریابهای B و C) دارد و طبعاً هر یک از بسته‌های ورودی باید برای یکی از این دو مسیریاب ارسال گردد، حتی اگر مقصد نهایی به مسیریابی کاملاً متفاوت متصل باشد. جدول مسیریابی ابتدایی A با عنوان "Initialy" ، در شکل ۵-۲ مشخص شده است.

پس از ورود بسته‌های ۱ و ۲ و ۳ به مسیریاب A ، موقتاً در حافظه ذخیره می‌شوند (تا کدهای کشف خطای آنها بررسی شود). سپس براساس جدول مسیریابی A ، این بسته‌ها به سمت مسیریاب C هدایت می‌شوند. به همین ترتیب و در ادامهٔ طی مسیر، بستهٔ ۱ از E و نهایتاً F می‌گذرد. پس از آنکه بسته به F رسید درون فریم لایهٔ پیوند داده

---

۱. پروتکل PPP یک پروتکل برای خطوط نقطه‌به‌نقطه است که در لایهٔ پیوند داده‌ها عمل می‌کند یعنی یک فریم را بر روی کانالی واحد و غیر مشترک، از نقطه‌ای به نقطهٔ دیگر تحویل می‌دهد. به فصل سوم مراجعه کنید. ـم.

جاسازی (کپسوله) شده و از طریق شبکهٔ LAN برای H2 ارسال می‌گردد. بسته‌های ۲ و ۳ نیز به همین طریق مسیریابی و هدایت می‌شوند.

ولیکن (به عنوان مثال) برای بستهٔ ۴ اتفاق متفاوتی می‌افتد. وقتی این بسته به A می‌رسد اگرچه به مقصد F طی مسیر می‌کند ولیکن برخلاف قبل به سوی مسیریاب B ارسال می‌شود. به دلایل متعدد A تصمیم گرفته تا بستهٔ ۴ را از مسیری متفاوت با مسیر سه بستهٔ قبلی، به سمت F بفرستد. شاید این مسیریاب متوجه ازدیاد ترافیک و ازدحام بر روی مسیر ACE شده و جدول مسیریابی خود را به جدولی جدید (که در شکل با عنوان Later مشخص شده) اصلاح و بهنگام‌سازی کرده است. الگوریتمی که مدیریت این جداول را بر عهده دارد و در خصوص مسیریابی تصمیم می‌گیرد اصطلاحاً «**الگوریتم مسیریابی**» (Routing Algorithm) نامیده می‌شود. الگوریتمهای مسیریابی یکی از اصلیترین مفاد این فصل هستند.

۵-۱-٤ پیاده‌سازی خدمات اتصال‌گرا

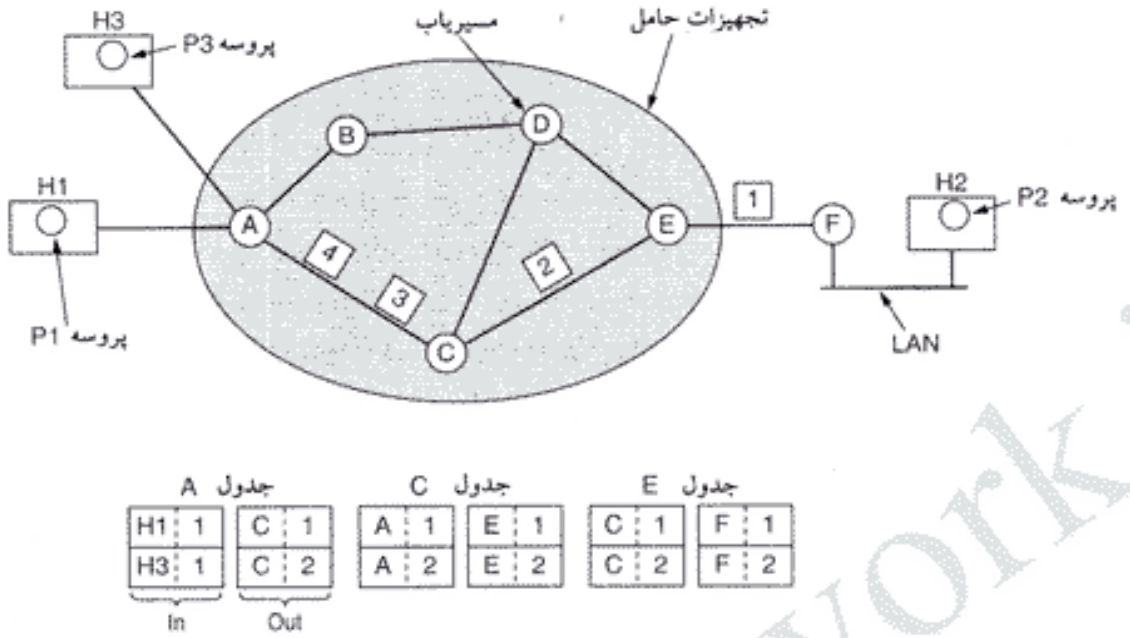
رای عرضهٔ خدمات اتصال‌گرا، به زیرشبکه‌ای مبتنی بر «مدار مجازی» (Virtual Circuit) نیازمندیم. حال ببینیم که چگونه کار می‌کند: ایدهٔ اصلی در مدار مجازی آن است که برخلاف مثال شکل ۵-۲ ، از انتخاب مسیرهای جدید برای هر بسته اجتناب شود. در عوض وقتی یک «**اتصال**» ایجاد شد، به عنوان بخشی از عملیات تنظیم اتصال (Connection Setup) ، مسیری بین ماشین مبداء و ماشین مقصد انتخاب می‌شود و مشخصات آن در جدول داخلی هر مسیریاب درج می‌گردد. دقیقاً مشابه با سیستم تلفن پس از برقراری اتصال، تمام بسته‌ها از طریق همین مسیر هدایت خواهند شد. پس از آنکه اتصال برقرارشده خاتمه یافت مدار مجازی متناظر نیز پایان یافته و حذف می‌شود. در خدمات اتصال‌گرا هر بسته با خود یک شمارهٔ شناسایی حمل می‌کند که مشخص می‌کند به کدام مدار مجازی تعلق دارد.[۱]

به عنوان مثال، به وضعیت نشان داده شده در شکل ۵-۳ توجه کنید. در اینجا ماشین میزبان H1 اتصال شماره ۱ را با ماشین میزبان H2 برقرار نموده است. مشخصهٔ این اتصال به عنوان اولین درایه (Entry) در جداول مسیریابی درج شده است. اولین سطر از جدول A بیان می‌کند که اگر بسته‌ای با شمارهٔ شناسایی ۱ از H1 دریافت شود باید با همین شمارهٔ شناسایی برای مسیریاب C ارسال گردد. به روش مشابه مسیریاب C مطابق با اولین درایهٔ خود، بسته را با شمارهٔ شناسایی ۱ به سوی مسیریاب E هدایت می‌نماید.

حال بررسی کنیم که اگر H3 نیز بخواهد اتصالی را با H2 برقرار کند چه اتفاقی می‌افتد. او نیز شماره شناسایی اتصال را ۱ در نظر می‌گیرد (چرا که این اتصال، اولین اتصالی است که او برقرار کرده است) و سپس از زیرشبکه می‌خواهد که برایش مداری مجازی ایجاد کند. انجام این تقاضا منجر به اضافه شدن سطر دوم به جداول است. دقت کنید که در اینجا یک تناقض وجود دارد و آن هم اینکه اگرچه A می‌تواند بسته‌هایی که با شمارهٔ شناسایی ۱ از H1 می‌رسند را از بسته‌هایی با همین شماره که از H3 می‌رسند تمیز بدهد ولی مسیریاب C قادر به چنین کاری نیست [چرا که هر دو با یک شماره و از A می‌رسند]. به همین دلیل A برای ترافیک خروجی که در قالب اتصال ۲ خارج می‌شوند شمارهٔ شناسایی متفاوتی را در نظر می‌گیرد. برای اجتناب از تناقضاتی از این دست، مسیریابها باید بتوانند شمارهٔ شناسایی اتصال هر بستهٔ خروجی را تغییر بدهند.

---

۱. اتصال را در ذهن خود با یک تماس تلفنی قیاس کنید. ابتدا شماره می‌گیرید، تماس شما برقرار می‌شود، تا زمان دلخواه گفتگو می‌کنید و نهایتاً تماس را قطع می‌نمایید. اتصال در زیرشبکهٔ مدار مجازی بمعنای هماهنگی قبلی بین مبداء، مقصد و مسیریابهای میانی تلقی می‌شود. به مسیری که با هماهنگی قبلی ایجاد می‌شود یک «مدار مجازی» می‌گویند و دارای یک شماره شناسایی است. به این شمارهٔ شناسایی اصطلاحاً «شناسهٔ مدار مجازی» یا «شناسهٔ اتصال» (Connection ID) اطلاق می‌شود. ـم



شکل ۵-۳. مسیریابی در یک زیرشبکهٔ مدار مجازی.

۵-۱-۵ مقایسهٔ زیرشبکه‌های مدار مجازی و دیتاگرام

هر یک از روشهای مدار مجازی و دیتاگرام حامیان و منتقدین خود را دارند. حال تلاش می‌کنیم مباحثات آنها را به طور خلاصه ارائه نماییم. محورهای اصلی این مباحثات در شکل ۵-۴ فهرست شده‌اند؛ هر چند ممکن است سفسطه‌کنندگان بتوانند برای هر یک از موارد این جدول، مثال نقضی پیدا کنند.

در هر زیرشبکه می‌توان اصولی را برای قیاس زیرشبکه‌های مدار مجازی و دیتاگرام مطرح کرد. یکی از آنها مسئله میزان حافظه مسیریاب در مقایسه با پهنای باند تلفاتی است. روش مدار مجازی اجازه می‌دهد که بسته‌ها به جای همراه داشتن آدرسهای کامل فقط شمارهٔ مدار مجازی را با خود داشته باشند. هر گاه بسته‌ها نسبتاً کوتاه باشند وجود آدرس کامل در هر بسته ممکن است حجم سربار زیادی را تحمیل کرده و بدین ترتیب بخشی از پهنای باند مفید هدر می‌رود. در عوض، هزینه‌ای که برای بکارگیری مدار مجازی پرداخت می‌شود فضای حافظه‌ای است که جدول مشخصات مدارات مجازی در درون مسیریاب، به خود اختصاص می‌دهد. بسته به قیمت پهنای باند کانالهای مخابراتی در مقایسه با حافظهٔ مسیریاب یکی از این دو ارزانتر تمام می‌شود.

یکی دیگر از مسائل قیاسی، «زمان تنظیم و ایجاد مدار مجازی» در مقایسه با زمان جستجو و تحلیل آدرسها است.[1] استفاده از مدار مجازی منوط به طی مراحل تنظیم مسیر است که فرآیندی زمان‌بر بوده و منابع زیرشبکه را مصرف می‌کند. با این وجود در زیرشبکهٔ مبتنی بر مدار مجازی، تشخیص آنکه چه کاری با یک بستهٔ داده باید انجام شود ساده است: مسیریاب از شمارهٔ شناسایی مدار مجازی به عنوان اندیس جدول مسیریابی استفاده می‌کند تا مسیری که بسته باید طی کند مشخص شود. در زیرشبکهٔ دیتاگرام برای جستجو و پیدا کردن درایهٔ متناظر با مقصد، به روالی پیچیده‌تر نیاز است.

مورد دیگر، فضای مورد نیاز جدول مسیریابی در حافظهٔ مسیریاب است. یک زیرشبکه دیتاگرام نیازمند آن است که به ازای هر مقصد در شبکه، یک درایه (Entry) در جدول مسیریابی خود داشته باشد، در حالی که در

---

۱. Setup Time versus Address Parsing Time

مورد	زیرشبکه دیتاگرام	زیرشبکه مدارمجازی
تنظیم مدار (Circuit Setup)	نیازی نیست	نیاز است.
آدرس‌دهی	هر بسته آدرس دقیق و کامل مبداء و مقصد را با خود حمل می‌کند.	هر بسته فقط یک شماره کوتاه مدار مجازی (VC) را با خود دارد.
اطلاعات وضعیت	مسیریاب نیازی به نگهداری اطلاعاتی در خصوص وضعیت هر اتصال ندارد.	به ازای هر مدار مجازی تمام مسیریابها باید اطلاعاتی در خصوص وضعیت آن نگاه دارند.
مسیریابی	هر بسته بطور مستقل مسیریابی می‌شود.	مسیر فقط یکبار و آنهم در هنگام تنظیم مدار مجازی انتخاب می‌شود و تمام بسته‌ها از همان مسیر حرکت می‌کنند.
تاثیر خرابی مسیریاب	بی‌تاثیر، مگر در مورد بسته‌هایی که در حین خرابی از بین رفته‌اند.	تمام مدارات مجازی که از مسیریاب خراب شده می‌گذشته‌اند قطع می‌شوند.
تضمین کیفیت خدمات	دشوار	اگر برای هر مدار مجازی منابع لازم از قبل تخصیص یابد، بسیار آسان است.
کنترل ازدحام	دشوار	اگر برای هر مدار مجازی منابع لازم از قبل تخصیص یابد، بسیار آسان است.

شکل ۵-۴. مقایسهٔ زیرشبکه‌های دیتاگرام و مدار مجازی.

زیرشبکهٔ مدار مجازی، فقط به ازای هر اتصال به یک درایه نیاز است. ولیکن این حُسن تا حدودی گمراه‌کننده و توخالی است چراکه بسته‌های تنظیم اتصال [که برای اولین بار ارسال می‌شوند] بهر نحو باید همانند روش دیتاگرام یکبار مسیریابی شوند و آنها نیز دارای آدرسهای کامل مقصد هستند. [یعنی در زیرشبکهٔ مدار مجازی هم به جدول مسیریابی و هم به جدول مدارات مجازی نیاز است. -م]

مدارهای مجازی دارای محاسنی در خصوص تضمین «کیفیت خدمات» (QoS) هستند و از بروز ازدحام (congestion) در زیرشبکه جلوگیری می‌کنند چراکه «منابع» (شامل بافر، پهنای باند و سیکلهای CPU) می‌توانند پیشاپیش و در هنگام ایجاد اتصال، رزرو شود. بعداً، به محض دریافت بسته‌ها، پهنای باند مورد نیاز و ظرفیت لازم در مسیریاب، مهیا و آماده است. در زیرشبکه‌های دیتاگرام، اجتناب از ازدحام دشوارتر است.

در «سیستمهای پردازش تراکنشی»[1]، (همانند وقتی که خرید با کارتهای اعتباری بررسی می‌شوند) مسئله تنظیم یک مدار مجازی و سپس حذف آن ممکن است کاربرد این روش را از اعتبار ساقط کند. [به عبارت بهتر در سیستمهایی که تعامل و مبادلهٔ داده با ماشینها، کوتاه و سریع است ایجاد یک مدار مجازی و ختم آن سربار زیادی به شبکه تحمیل می‌نماید. -م] اگر بخش اعظم ترافیک از این نوع باشد استفاده از مدار مجازی درون یک زیرشبکه چندان مناسب نیست. در طرف مقابل هرگاه حجم داده‌های ارسالی انبوه باشد، مدارهای مجازی دائم که به صورت دستی تنظیم می‌شوند و برای ماهها یا سالها تغییر نمی‌کنند می‌تواند سودمندتر باشد.

مدارهای مجازی مشکل «آسیب‌پذیری» دارند. اگر یک مسیریاب از کار بیفتد و حافظه خود را از دست بدهد (حتی در صورتی که چند ثانیه بعد به زیرشبکه برگردد)، قطعاً تمام مدارهای مجازی که از آن عبور می‌کنند، از دست خواهند رفت. در مقابل اگر یک مسیریاب مبتنی بر دیتاگرام از کار بیفتد تنها کاربرانی آسیب می‌بینند که

۱. Transaction Processing System

بسته‌های آنها هنگام بروز مشکل، درون حافظهٔ مسیریاب در صف منتظر بوده و بر حسب آنکه آیا دریافت بسته‌های داده قبلاً اعلام شده باشد یا نه، حتی ممکن است فقط بخشی از آنها با مشکل مواجه شوند. از دست رفتن یک خط مخابراتی برای مدارات مجازی که از آن خط استفاده کرده‌اند بحران‌زاست در حالی که اگر از روش دیتاگرام استفاده شده باشد این اشکال براحتی جبران خواهند شد. در روش دیتاگرام مسیریابها مجازند ترافیک را بر روی زیرشبکه توزیع و موازنه کنند لذا ممکن است در حین ارسال یک دنبالهٔ طولانی از بسته‌ها مسیرها عوض شوند.

۲-۵ الگوریتمهای مسیریابی

وظیفهٔ اصلی لایهٔ شبکه ، مسیریابی و هدایت بسته‌های داده از ماشین مبداء به ماشین مقصد است. در اکثر زیرشبکه‌ها، بسته‌ها برای طی مسیر خود بایستی چندین «**گام**» (HOP) راه بپیمایند.[1] الگوریتمی که این مسیرها را انتخاب می‌کنند و همچنین ساختمان دادهٔ مورد استفاده ، یکی از زمینه‌های مهم در طراحی لایهٔ شبکه محسوب می‌شود.

«**الگوریتمهای مسیریابی**» آن بخش از نرم‌افزار لایهٔ شبکه‌اند که مسئولیت دارند در خصوص خط خروجی که یک بسته ورودی باید بر روی آن ارسال شود، تصمیم‌گیری کنند. اگر در داخل زیرشبکه از روش دیتاگرام استفاده شده باشد این تصمیم‌گیری باید به ازای هر بستهٔ دریافتی از نو تکرار شود چراکه در هر لحظه ممکن است بهترین مسیر تغییر نماید. اگر زیرشبکه از روش مدار مجازی بهره گرفته باشد، تصمیم‌گیری در خصوص مسیرها فقط یکبار و آنهم در هنگام تنظیم و ایجاد هر مدار مجازی جدید صورت می‌گیرد و طبعاً بسته‌های داده، همگی از مسیری که قبلاً ایجاد شده هدایت می‌شوند. این حالت گاهی روش «**مسیریابی مبتنی بر نشست**» (Session Routing) نامیده می‌شود چرا که مسیر ایجاد شده در خلال نشست کاربر برقرار خواهد ماند؛ (مثلاً در حین نشست از راه دور با ترمینال ـLogin sessionـ یا نشست انتقال فایل).

گاهی تمایز قائل شدن بین فرآیند «مسیریابی» (که به تصمیم‌گیری در خصوص مسیرهای بهینه اطلاق می‌شود) و فرآیند «هدایت» (Forwarding) (آنچه که با ورود بسته اتفاق می‌افتد) مفید است: می‌توانید بدینگونه بیندیشید که هر مسیریاب دارای دو پروسه در درون خود است: یکی از آنها بسته‌ها را به محض ورود پردازش کرده و از طریق جدول مسیریابی، یک خط خروجی مناسب برای آنها انتخاب می‌نماید. این پروسه همان «عمل هدایت» است. پروسهٔ دیگر موظف به پر کردن و بهنگام‌سازی محتویات جدول مسیریابی است. این همان جایی که «الگوریتمهای مسیریابی» (Routing Algorithm) به میدان می‌آیند.

فارغ از آنکه آیا برای هر بسته ، مسیرها بطور مستقل و جداگانه انتخاب می‌شوند یا آنکه فقط یکبار و آن هم در حین ایجاد یک اتصال [مدار مجازی] تعیین می‌گردد، از یک الگوریتم مسیریابی ویژگیهای خاصّی انتظار می‌رود: (۱) **صحّت عملکرد** (correctness) (۲) **سادگی** (simplicity) (۳) **قابلیت تحمل** (Robustness) (۴) **پایداری** (stability) (۵) **مساوات** (Fairness) (۶) **بهینگی** (Optimality).

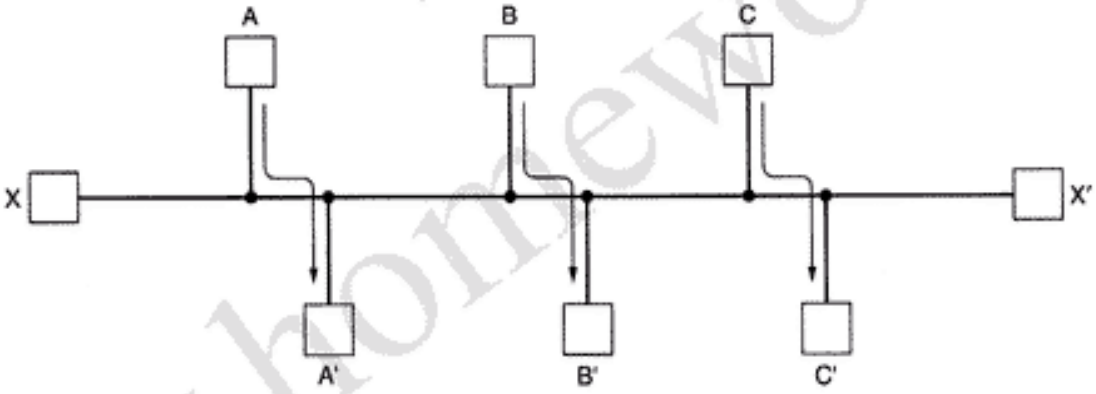
ویژگی «صحت» و «سادگی» نیازی به توضیح ندارد در حالی که نیاز به «قابلیت تحمل» در بدو امر چندان روشن به نظر نمی‌رسد. با ورود یک شبکهٔ عظیم به صحنهٔ عمل، انتظار آنست که برای سالها بدون بروز هیچگونه خرابی سیستم در سطح وسیع، به کار خود ادامه بدهد. در خلال این دورهٔ زمانی ممکن است انواع سخت‌افزارها و نرم‌افزارها از کار بیفتند. ماشینهای میزبان ، مسیریابها و خطوط ارتباطی به کرّات از کار می‌افتند و توپولوژی شبکه

---

۱. به گذر بسته از یک مسیریاب، گام یا Hop گفته می‌شود

چندین بار تغییر می‌کند. الگوریتم مسیریابی باید بتواند هر گونه تغییر در توپولوژی و ترافیک را بدون ایجاد وقفه در کار ماشینهای میزبان یا نیاز به راه‌اندازی مجدد شبکه (در صورت از کار افتادن برخی از مسیریابها)، اصلاح و مدیریت نماید.

یکی دیگر از اهداف و ویژگیهای الگوریتم مسیریابی «پایداری» است. برخی از الگوریتمهای مسیریابی هرگز به یک عملکرد ثابت و پایدار همگرا نمی‌شوند بدون آنکه مدت زمان در حال کار و اجرا بودن آنها اهمیت و تأثیری داشته باشد. ویژگیهای «مساوات» و «بهینگی» ممکن است واضح به نظر برسند و مطمئناً هیچ عقل سلیمی با آن مخالف نیست ولی گاهی در مرحلهٔ عمل با یکدیگر در تناقض و تضاد هستند. به عنوان یک مثال از این تناقض، به شکل ۵-۵ نگاه کنید. فرض نمائید که ترافیک بین A و A′، بین B و B′ و بین C و C′ بقدری است که لینک ارتباط افقی اشباع می‌شود. برای آنکه جریان کلی اطلاعات به حداکثر برسد باید ترافیک بین X و X′ قطع گردد؛ [تا بهینگی رعایت شده باشد]. متأسفانه X و X′ ممکن است متوجه چنین موضوعی نباشند ولیکن باید بین «کارآیی» و «مساوات» حالتی بینابین و متعادل مدّنظر قرار بگیرد. اگر ترافیک بین X و X′ قطع شود بهینگی بدست می‌آید ولی در عوض مساوات رعایت نخواهد شد!



شکل ۵-۵ تضاد بین «بهینگی» و مساوات».

قبل از آنکه بتوان بین «مساوات» و «بهینگی» تعادل ایجاد کرد باید ابتدا تصمیم بگیریم که در پی بهینه کردن چه چیزی هستیم! به حداقل رساندن متوسط تأخیر بسته‌ها نامزد واضحی برای بهینه شدن است ولی به حداکثر رساندن بازده مفید شبکه (Throughput) نیز نامزد دیگری است. به علاوه این دو هدف با یکدیگر در تناقض هستند چرا که مثلاً هر گاه یک سیستم صف‌بندی [همانند صفی که در هر مسیریاب ایجاد می‌شود] در ظرفیت حداکثر خود کار کند تأخیری طولانی را تحمیل خواهد کرد.[1] به عنوان یک حالت بینابین ، در برخی از شبکه‌ها سعی شده تا تعداد «گامهای مسیر» (که یک بسته باید طی کند) به حداقل برسد زیرا کاهش تعداد گامها باعث کاهش تأخیر و پهنای باند مصرفی خواهد شد که نهایتاً به بهبود بازده مفید شبکه می‌انجامد.

الگوریتمهای مسیریابی را می‌توان به دو ردهٔ کلی تقسیم‌بندی کرد: «غیروفقی» (nonadaptive) و «وفقی» (adaptive). روشهای غیروفقی تصمیم‌گیری خود در خصوص مسیریابی را بر مبنای اندازه‌گیری و تخمین هوشمند ترافیک و توپولوژی فعلی شبکه، قرار نداده‌اند. در عوض برای انتخاب مسیری بین I و J (برای هر I و J مفروض)، یک مسیر از قبل و به صورت offline محاسبه شده و در هنگام راه‌اندازی شبکه درون مسیریابها بارگذاری می‌شود. به این روال اغلب «مسیریابی ایستا» (Static Routing) گفته می‌شود.

---

۱. در صف نگاه داشتن بسته‌ها می‌تواند به بازده مفید شبکه بیفزاید ولی تأخیر را نیز افزایش خواهد داد. رجوع کنید به نظریهٔ صف -م

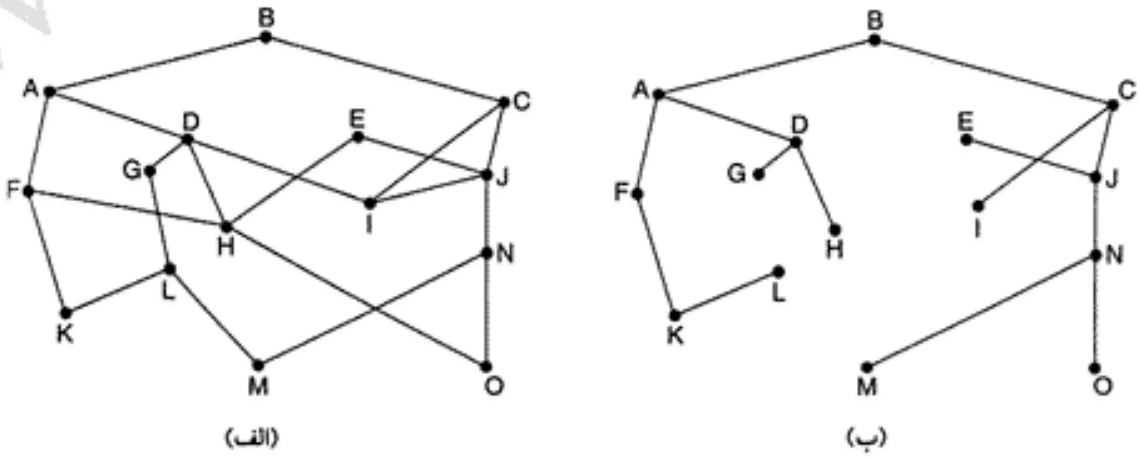
در مقابل ، «الگوریتمهای وفقی» تصمیم‌گیری در خصوص مسیریابی را برحسب تغییرات توپولوژی و عموماً ترافیک لحظه‌ای، بصورت هوشمند انجام می‌دهند. الگوریتمهای مسیریابی وفقی در نحوهٔ گردآوری «اطلاعات مسیر» با یکدیگر متفاوتند (مثلاً اینکه آیا این اطلاعات به صورت محلی گردآوری می‌شود یا از مسیریابها مجاور بدست می‌آید یا از همهٔ مسیریابها می‌رسد). الگوریتمهای وفقی همچنین در پارامترهایی نظیر: (۱) زمان بهنگام‌سازی اطلاعات مسیر (مثلاً بطور متناوب هر ΔT ثانیه بهنگام شوند یا فقط زمانیکه توپولوژی عوض شود) (۲) معیاری که برای بهینه‌سازی مورد استفاده قرار می‌گیرد (مثل فاصله ، تعداد گام (HOP) یا زمان تخمینی انتقال) با یکدیگر متفاوت هستند. در بخش بعدی گونه‌های مختلفی از الگوریتمهای مسیریابی اعم از پویا و ایستا را تشریح خواهیم کرد.

۵-۲-۱ اصل بهینگی

قبل از آنکه بر روی یک الگوریتم خاص متمرکز شویم، شاید پرداختن به این نکته مفید باشد که هر کسی می‌تواند بدون توجه به توپولوژی یا ترافیک شبکه یک ارزیابی و تعبیر کلی در خصوص مسیرهای بهینه ارائه بدهد. این تعبیر به نام «**اصل بهینگی**» مشهور است و بیان می‌کند که اگر مسیریاب J بر روی مسیر بهینهٔ بین مسیریاب I تا مسیریاب K واقع شده باشد بنابراین مسیر بهینه از J تا K نیز بر روی همان مسیر خواهد بود. برای بررسی این موضوع، آن قسمت از مسیر که بین I تا J قرار دارد را r_1 و مابقی مسیر را r_2 بنامید. اگر مسیری بهتر از r_2 بین J تا K وجود داشته باشد می‌توان آن را به r_1 افزود تا مسیر بهتری از I تا K ایجاد شود؛ این موضوع با اصل قضیه که بیان می‌کرد مسیر r_1r_2 بهینه است تناقض دارد.

به عنوان یک نتیجهٔ مستقیم از اصل بهینگی می‌توان اثبات کرد که مجموعهٔ مسیرهای بهینه بین تمام گره‌های مبداء و یک گره مقصد خاص ، درختی را تشکیل می‌دهد که ریشهٔ آن بر روی گره مقصد قرار دارد. به چنین درختی اصطلاحاً Sink Tree گفته می‌شود که شمای آن در شکل ۵-۶ به تصویر کشیده شده است. در این شکل معیار فاصله «تعداد گام» در نظر گرفته شده است. دقت کنید که درخت Sink Tree الزاماً یکتا و منحصر به فرد نیست و ممکن است درختی متفاوت با طول مسیرهای مشابه پیدا شود. هدف اصلی الگوریتمهای مسیریابی آن است که برای تمام مسیریابها درخت Sink Tree محاسبه شده و مورد استفاده قرار بگیرد.

بدیهی است که Sink Tree ، حداقل یک درخت است، بنابراین بهیچوجه دارای حلقه نیست و بدین ترتیب هر بسته پس از طی چند گام محدود و معین، تحویل مقصد خواهد شد. ولیکن در دنیای عمل همیشه کار بدین سادگی



شکل ۵-۶. (الف) یک زیرشبکه (ب) درخت Sink Tree برای مسیریاب B.

نخواهد بود. لینکها و مسیریابها می‌توانند به ناگاه از کار بیفتند و سپس مجدداً عملیات طبیعی خود را از سر بگیرند لذا مسیریابهای مختلف ممکن است توپولوژی شبکه را به گونهٔ متفاوتی ببینند.[1] همچنین ممکن است بدین نکتهٔ ظریف بیندیشیم که آیا هر مسیریاب مجبور است مستقلاً اطلاعات لازم در خصوص توپولوژی شبکه را جمع‌آوری کرده و Sink Tree مربوطه را خودش محاسبه نماید یا آنکه این اطلاعات به روشهای دیگری بدست می‌آیند. در بخشهای آتی مختصراً بدین موضوعات خواهیم پرداخت. بهر حال، «اصل بهینگی» و «Sink Tree» روشی برای محک زدن و ارزیابی الگوریتمهای مسیریابی دیگر هستند.

۲-۲-۵ مسیریابی مبتنی بر کوتاهترین مسیر

اجازه بدهید مطالعات خود پیرامون الگوریتم‌های مسیریابی را با پرداختن به روشی آغاز کنیم که بطور گسترده‌ای از آن استفاده می‌شود چراکه ساده و فهم جزئیات آن راحت است. ایدهٔ اصلی آن است که گراف زیرشبکه رابگونه‌ای تشکیل بدهیم که در آن هر گره از گراف یک مسیریاب را مشخص می‌کند و هر یک از «کمانها» (Arcs) مشخص‌کنندهٔ یک خط ارتباطی (که اغلب لینک نامیده می‌شوند) هستند. برای انتخاب بهترین مسیر بین یک زوج مسیریاب مفروض، یک الگوریتم خاصّ، «کوتاهترین مسیر» بین آن دو را در این گراف پیدا می‌کند.

مفهوم «کوتاهترین مسیر» نیاز به اندکی توضیح دارد: یکی از روشهای محاسبهٔ طول مسیر، شمارش تعداد «گام» است. با استناد به این معیار، مسیرهای ABC و ABE در شکل ۵-۷ طول معادلی دارند. یکی دیگر از معیارها، فاصلهٔ جغرافیایی گره‌ها از یکدیگر برحسب کیلومتر است که در این حالت ABC به وضوح از ABE طولانی‌تر می‌باشد (با فرض آنکه شکل به گونه‌ای ترسیم شده که مقیاسی از نقشهٔ واقعی را ارائه می‌دهد).

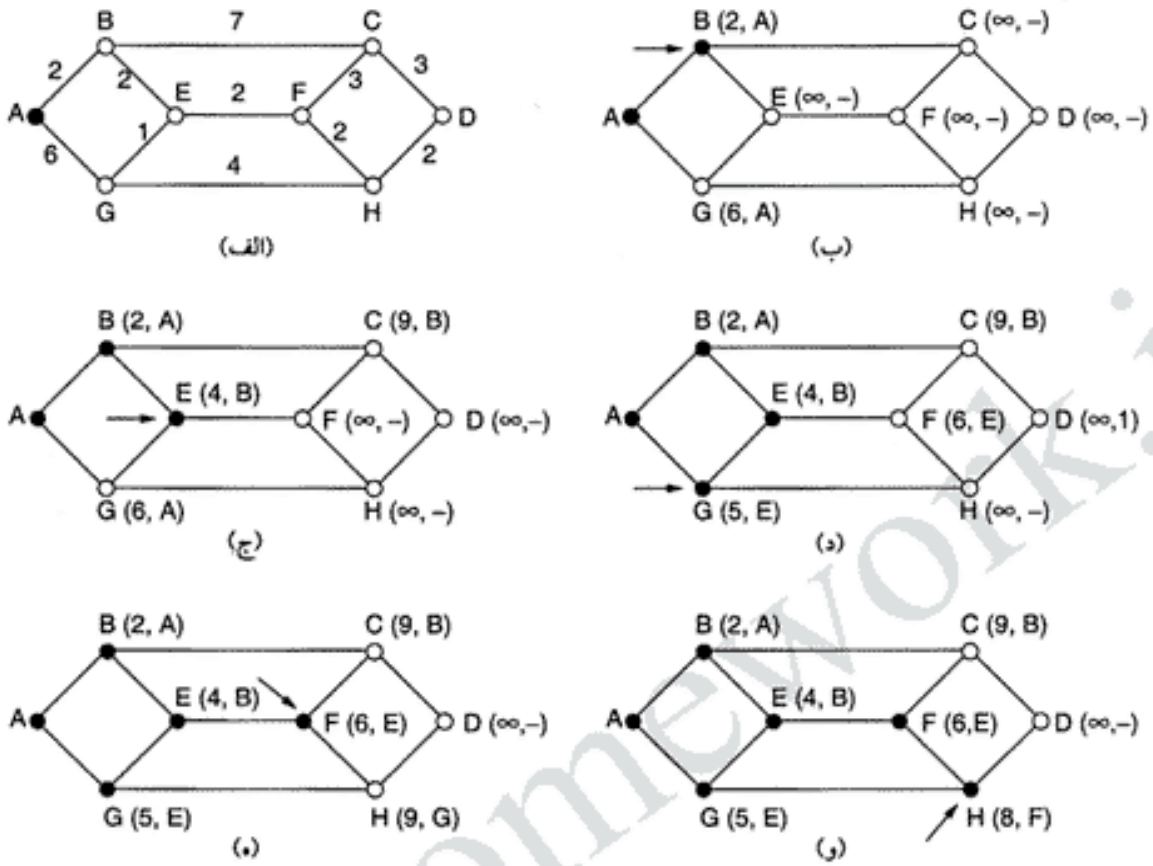
با این وجود، به غیر از معیارهای تعداد گام یا فاصله فیزیکی، معیارهای دیگری نیز وجود دارند. به عنوان مثال هر کمان از گراف می‌تواند برچسبی داشته باشد که مقدار متوسط تأخیر صفّ [صفّ درون حافظه مسیریاب] و تأخیر انتقال را تعیین کند؛ این برچسبها بطور متناوب و یا استفاده از برخی بسته‌های استاندارد محاسبه می‌شوند. در چنین گراف برچسب‌داری، کوتاهترین مسیر همانا سریعترین مسیر است (یعنی مسیر با حداقل تاخیر)، نه مسیری که کمترین تعداد کمان یا کوتاهترین فاصلهٔ جغرافیایی را دارد.

در حالت کلی، برچسب هر کمان می‌تواند برحسب تابعی از پارامترهای «فاصله»، «پهنای باند»، «میانگین ترافیک»، «هزینهٔ ارتباط»، «طول متوسط صف»، «تأخیر اندازه‌گیری‌شده» یا عوامل دیگر، محاسبه شود. با تغییر در «تابع وزن‌دهی» (Weighting Function)، این الگوریتم می‌تواند «کوتاهترین مسیر» را برحسب یکی از این معیارها یا ترکیبی از آنها بدست بیاورد.

تاکنون الگوریتمهای متعددی برای محاسبهٔ کوتاهترین مسیر بین دو گره از گراف، معرفی شده‌اند. یکی از آنها توسط «دایکسترا» (Dijkstra, 1959) ارائه شد. در الگوریتم دایکسترا، هر گره دارای برچسبی است که فاصلهٔ آن گره را تا یک گره مبداء برحسب بهترین مسیر، مشخص می‌کند. (در شکل ۵-۷ این برچسبها درون پرانتز درج شده‌اند.) در ابتدا هیچ مسیری مشخص نیست فلذا تمام گره‌ها در برچسب خود علامت بی‌نهایت دارند. در حین اجرای الگوریتم و پیدا شدن مسیرها این برچسبها تغییر می‌کنند تا مسیرهای بهتر را مشخص نمایند. هر برچسب می‌تواند یکی از دو حالت «موقت» (Tentative) یا «دائم» (Permanent) داشته باشد. در ابتدای کار تمام برچسبها در حالت موقتی علامت خورده‌اند. پس از آن که مشخص شد یک برچسب کوتاهترین مسیر ممکن را از مبداء تا آن گره مشخص کرده، حالت آن گره به حالت «دائم» تغییر کرده و از آن به بعد هیچگاه عوض نخواهد شد.

---

۱. اگر هر یک از مسیریابها، مسیرهای بهینه را متفاوت از دیگری تشخیص بدهند و بر اساس اطلاعات ناقص خود مسیر انتخاب کنند، گاهی حلقهٔ بی‌نهایت ایجاد می‌شود. -م



شکل ۵–۷. پنج مرحلهٔ نخست از روال محاسبهٔ کوتاهترین مسیر از A به D. فلشها «گره کار» را مشخص می‌کنند.

برای آن که ببینیم الگوریتم برچسب‌گذاری چگونه کار می‌کند، گراف «بدون جهت» و «وزن‌دار» شکل ۵–۷–الف را در نظر بگیرید که در آن وزنها فرضاً فاصله را مشخص کرده است. می‌خواهیم کوتاهترین مسیر از A به D را پیدا کنیم. کار را با تغییر علامت گره A به حالت «ثابت» (که به صورت یک دایرهٔ توپر مشخص شده) شروع می‌کنیم. سپس به نوبت هر یک از گره‌های مجاور گره A (که گره کار نام دارد) را آزمایش می‌کنیم و برچسبشان را برحسب فاصلهٔ آنها تا A، تغییر می‌دهیم. هرگاه برچسب یک گره را تغییر دادیم، در آن برچسب، نام گره‌ای که آزمایش بر مبنای آن صورت گرفته درج می‌شود تا بعداً بتوان مسیر نهایی را بدست آورد. [یعنی مثلاً وقتی گره‌های همسایهٔ A یعنی B و G را بررسی می‌نماییم و فاصلهٔ آنها تا A را در برچسبشان تغییر می‌دهیم، باید نام گره کار قبلی یعنی A در برچسب آنها درج شود. –م] پس از آزمایش گره‌های مجاور A، تمام گره‌هایی که به صورت موقتی علامت خورده‌اند [یعنی دایره‌های توخالی] را در «**کلّ گراف**» بررسی کرده و گره‌ای با کوچکترین برچسب را انتخاب و آن را به صورت «دائم» علامتگذاری می‌کنیم. (شکل ۵–۷–ب) این گره به عنوان «گره کار» جدید انتخاب می‌شود.

حال از گره B شروع کرده و تمام گره‌های مجاور آن را آزمایش می‌نماییم. اگر مجموع برچسب B و فاصلهٔ B تا گره مورد نظر از آنچه که درون برچسب B نوشته شده کمتر باشد برچسب آن گره تغییر می‌کند چراکه مسیری کوتاهتر را پیدا کرده‌ایم. [به عبارت دیگر وقتی گره‌های همسایهٔ B را بررسی می‌نماییم ـ یعنی E و C را ـ مجموع فاصلهٔ C تا B یعنی ۷ را با فاصله B تا A یعنی ۲ جمع می‌کنیم و چون ۹ از مقدار بی‌نهایت در برچسب C کمتر است برچسب آن به صورت (9,B) تغییر می‌کند یعنی تا رسیدن به گره A فاصله ۹ و گره قبلی B است.–م]

پس از آن که گره‌های مجاور «گره کار» بررسی شدند و در صورت لزوم برچسبهای موقت تغییر نمودند، برای پیدا کردن گره‌ای با علامت موقت و کمترین مقدار فاصله، کل گراف را جستجو می‌نماییم. این گره به حالت «دائم» تغییر وضعیت داده و گره کار جدید در مرحلهٔ بعد خواهد شد. شکل ۵-۷ پنج مرحلهٔ اول این الگوریتم را نشان می‌دهد.

برای آنکه ببینیم چرا این الگوریتم بدرستی کار می‌کند به شکل ۵-۷-ج دقت نمایید. در این مرحله، حالت گره E را به حالت «دائم» تغییر داده‌ایم. فرض کنید که مسیری کوتاهتر به غیر از ABE مثل AXYZE وجود داشته باشد. دو امکان وجود دارد: یا گره Z قبلاً به حالت «دائم» علامت خورده است یا حالت «موقت» دارد. اگر حالت دائم داشته باشد بنابراین نتیجه می‌گیریم که E قبلاً بررسی شده است (در همان مرحله‌ای که گره Z در آن مرحله علامت دائم خورده است) لذا مسیر AXYZE از چشم ما دور نمانده و قاعدتاً نمی‌توانسته کوتاهترین مسیر باشد.

اکنون حالتی را در نظر بگیرید که در آن Z هنوز برچسب موقت دارد. اگر برچسب Z بیشتر یا مساوی آنچه که در برچسب E درج شده، باشد که در این حالت مسیر AXYZE نمی‌تواند از ABE کوتاهتر باشد؛ یا آنکه برچسب Z کوچکتر از برچسب E است که در این حالت Z بجای E علامت دائم می‌خورد و E باید قبل از Z بررسی شود [یعنی از طریق E باید Z بررسی شود و مسیر AXYEZ خواهد بود. -م]

این الگوریتم در شکل ۵-۸ ارائه شده است. متغیرهای سراسری n و dist توصیف‌کنندهٔ گراف هستند و قبل از فراخوانی تابع shortest_path مقداردهی می‌شوند. تنها تفاوت بین این برنامه و الگوریتمی که در بالا تشریح شد آن است که در برنامهٔ شکل ۵-۸ محاسبهٔ کوتاهترین مسیر از آخر (یعنی از گره پایانی) شروع شده است. از آنجایی که کوتاهترین مسیر از t به s در یک گراف بی‌جهت هیچ تفاوتی با کوتاهترین مسیر از s به t ندارد لذا آنکه از کدام گره شروع کنیم اهمیتی ندارد. (مگر آن که کوتاهترین مسیرها بیش از یکی باشند و جستجوی معکوس مسیر دیگری را نتیجه بدهد.) دلیل آنکه جستجوی کوتاهترین مسیر برعکس انجام می‌گیرد آن است که برچسب هر گره با گره قبلی خود (نه گره بعدی) تغییر می‌کند و وقتی مسیر نهایی در متغیر خروجی یعنی path کپی می‌شود، مسیر معکوس خواهد شد [چون عمل کپی از آخر به اول انجام می‌گیرد]. چون از گره آخر به اول شروع کردیم و نهایتاً مسیر بطور معکوس کپی شد این دو، اثر هم را خنثی کرده و جواب نهایی بطور صحیح تولید خواهد شد. [جواب نهایی در متغیر path ، بهترین مسیر از گره مبداء به گره مقصد خواهد بود.]

۵-۲-۳ الگوریتم سیل‌آسا (Flooding)

یکی از الگوریتمهای ایستا در ارسال بسته‌ها، «**الگوریتم سیل‌آسا**» است که براساس آن هر بستهٔ ورودی به یک مسیریاب، بر روی تمام خطوط خروجی (به استثنای خطی که بسته از طریق آن دریافت شده) ارسال می‌شود. این فرآیند سیل‌آسا، به وضوح منجر به تولید تعداد بسیار زیادی بسته‌های تکراری (در حقیقت بی‌نهایت بسته) خواهد شد مگر آنکه برای خاتمهٔ آن، سنجشی صورت بگیرد. یکی از معیارهای سنجش، آنست که در سرآیند هر بسته یک «**شمارندهٔ گام**» (Hop Counter) داشته باشیم و در هر گام یک واحد از آن کم شود و هنگامی که مقدار این شمارنده به صفر رسید بسته حذف شود. حالت آرمانی آن است که مقدار اولیهٔ این شمارنده معادل طول مسیر بین مبداء و مقصد در نظر گرفته شود ولیکن اگر فرستنده نداند که طول مسیر چقدر است می‌تواند مقدار این شمارنده را به بدترین حالت یعنی بزرگترین طول مسیر در شبکه تنظیم نماید.[1]

راهکار دیگر برای خاتمه دادن به این فرآیند آن است که فهرست بسته‌هایی که قبلاً یکبار به صورت سیل‌آسا ارسال شده‌اند را نگاه داریم تا از ارسال مجدد آن اجتناب شود. برای انجام این کار، مسیریاب مبداء باید در هر بسته

---

۱. در ادبیات شبکه به بزرگترین طول مسیر ، «قطر شبکه» گفته می‌شود.

```
#define MAX_NODES 1024              /* maximum number of nodes */
#define INFINITY 1000000000         /* a number larger than every maximum path */
int n, dist[MAX_NODES][MAX_NODES];          /* dist[i][j] is the distance from i to j */
void shortest_path(int s, int t, int path[])
{ struct state {            /* the path being worked on */
      int predecessor;          /* previous node */
      int length;             /* length from source to this node */
      enum {permanent, tentative} label; /* label state */
  } state[MAX_NODES];
  int i, k, min;
  struct state *p;
  for (p = &state[0]; p < &state[n]; p++) { /* initialize state */
      p->predecessor = -1;
      p->length = INFINITY;
      p->label = tentative;
  }
  state[t].length = 0;  state[t].label = permanent;
  k = t;            /* k is the initial working node */
  do {          /* Is there a better path from k? */
      for (i = 0; i < n; i++)           /* this graph has n nodes */
          if (dist[k][i] != 0 && state[i].label == tentative) {
              if (state[k].length + dist[k][i] < state[i].length) {
                  state[i].predecessor = k;
                  state[i].length = state[k].length + dist[k][i];
              }
          }

      /* Find the tentatively labeled node with the smallest label. */
      k = 0; min = INFINITY;
      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);

  /* Copy the path into the output array. */
  i = 0;  k = s;
  do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

شکل ۵-۸ الگوریتم دایجکسترا برای محاسبهٔ کوتاهترین مسیر در یک گراف.

که از یک ماشین میزبان دریافت می‌کند یک «شماره ترتیب» قرار بدهد. هر مسیریاب نیز باید فهرستی تشکیل بدهد و در آن شماره ترتیب بسته‌هایی را که قبلاً از یک مسیریاب دریافت کرده، درج کند. با ورود هر بسته ابتدا شمارهٔ ترتیب آن درون این فهرست جستجو می‌شود؛ اگر شمارهٔ آن در این فهرست وجود داشته باشد دیگر بهیچوجه (به روش سیل‌آسا) ارسال نخواهد شد (چون قبلاً یکبار ارسال شده است).

برای آنکه از رشد بی‌نهایت این فهرست اجتناب شود در فهرست به ازای هر مسیریاب مبداء، تنها یک شمارنده نگهداری می‌شود و مقدار این شمارنده یعنی k نشان می‌دهد که بسته‌های تا شمارهٔ k قبلاً دریافت شده‌اند. [یعنی در حقیقت فقط شمارهٔ ترتیب آخرین بستهٔ تولید شده توسط یک مسیریاب مشخص می‌شود. -م] وقتی بسته‌ای وارد یک مسیریاب می‌شود آزمون تکراری بودن آن بسیار ساده است [اگر شمارهٔ آن k یا کمتر از k بود تکراری است]؛ اگر بسته تکراری بود حذف می‌شود. بدین ترتیب به فهرست بسته‌های دریافتی با شمارهٔ کمتر از k نیازی نیست و فقط در اختیار داشتن آخرین شماره یعنی k کفایت می‌کند.

گونهٔ دیگری از روش سیل‌آسا که قابلیت اجرایی بیشتری دارد به نام **«روش سیل‌آسا بصورت انتخابی»** (Selective Flooding) مشهور است. در این الگوریتم، مسیریاب تمام بسته‌های دریافتی از یک خط را بر روی تمام خطوط خروجی ارسال نمی‌کند بلکه فقط آنها را بر روی خطوطی ارسال می‌کند که به صورت تخمینی در مسیر صحیحی قرار دارند. هدایت بسته‌هایی که قرار است به سمت غرب بروند به سمت شرق ، توجیه چندانی ندارد مگر آنکه توپولوژی زیرشبکه نوع عجیب و ویژه‌ای باشد و مسیریاب از آن به درستی آگاه باشد.

الگوریتم سیل‌آسا در اغلب کاربردها عملی نیست ولیکن در برخی از موارد خاص کاربردهایی دارد. به عنوان مثال در کاربردهای نظامی که گاهی باید برای تعداد زیادی از مسیریابها و در لحظه‌ای کوتاه، حجم زیادی اطلاعات ارسال شود، قدرت روش سیل‌آسا بسیار مطلوب خواهد بود. در پایگاه داده‌های توزیع‌شده (Distributed Database) گاهی لازم است که تمام پایگاههای داده بطور همزمان بهنگام‌سازی شوند؛ در چنین موردی نیز روش سیل‌آسا مفید خواهد بود. در شبکه‌های بی‌سیم تمام پیامهایی که توسط یک ایستگاه ارسال می‌شود توسط بقیهٔ ایستگاههایی که در برد رادیویی آن قرار گرفته‌اند قابل شنود است و در حقیقت نوعی از روش سیل‌آسا محسوب می‌شود و برخی از الگوریتمها از این ویژگی در راستای کار خود بهره می‌گیرند. چهارمین کاربرد روش سیل‌آسا، آنست که می‌توان از آن به عنوان معیاری برای مقایسه با الگوریتمهای دیگر مسیریابی بهره گرفت. در روش سیل‌آسا همیشه کوتاهترین مسیرها انتخاب می‌شود چرا که بسته‌ها از تمام مسیرهای ممکن به صورت موازی ارسال خواهد شد. طبعاً هیچ الگوریتم دیگری نمی‌تواند تأخیر کمتری نسبت به این روش داشته باشد (به شرط آنکه از سربار تحمیل شده توسط خود پروسهٔ سیل‌آسا، قابل چشمپوشی باشد.)

۵-۲-٤ مسیریابی بردار فاصله (Distance Vector Routing)

شبکه‌های کامپیوتری مدرن، عموماً بجای استفاده از روشهای ایستا که در قبل بدانها اشاره‌شد از روشهای پویا بهره می‌گیرند زیرا الگوریتمهای ایستا بار فعلی شبکه را در محاسبات مربوط به بهترین مسیرها دخالت نمی‌دهند. دو الگوریتم پویا یعنی **«مسیریابی بردار فاصله»** (DVR) و **«مسیریابی مبتنی بر حالت لینک»** (Link State Routing) از رایج‌ترین روشهای موجود هستند. در این بخش بطور خاص مروری بر الگوریتم نوع اول (یعنی بردار فاصله) خواهیم داشت و در بخش آتی به مطالعهٔ الگوریتم دوم می‌پردازیم.

الگوریتم «مسیریابی بردار فاصله» (DVR) بدین نحو کار می‌کند: هر مسیریاب جدولی را در خود نگه می‌دارد (به عبارتی یک بردار را) که در آن بهترین فاصله تا هر مسیریاب مقصد [یا به عبارتی کمترین هزینهٔ رسیدن به هر مسیریاب در زیرشبکه] و خطی که برای رسیدن به آن مقصد باید مورد استفاده قرار بگیرد، درج شده است. این جداول با مبادلهٔ اطلاعات بین مسیریابهای همسایه، بهنگام‌سازی می‌شود.

الگوریتمهای مسیریابی بردار فاصله گاهی با نامهای دیگری معرفی می‌شوند که اهم این اسامی عبارتند از «الگوریتم مسیریابی بلمن-فورد توزیع‌شده» (Distributed Bellman-Ford) و «الگوریتم فورد-فوکرسون» (Ford-Fulkerson) که به افتخار نام پژوهشگرانی است که آن را ابداع کرده‌اند. (Bellman, 1957; Ford and Fulkerson, 1962) این روش اولین الگوریتم مسیریابی در ARPANET بود و بعداً نیز با نام RIP (Routing Information Protocol) در اینترنت بکار گرفته شد.

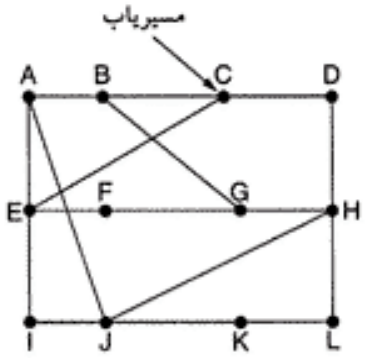
در «مسیریابی بردار فاصله» هر مسیریاب یک جدول مسیریابی دارد که به ازای هر مسیریاب موجود در زیرشبکه، یک «درایه» (Entry) در آن درج شده است. [کل جدول براساس آدرس هر مسیریاب، ایندکس شده است. -م] هر درایه دارای دو بخش است: (۱) خط خروجی مناسب برای رسیدن به مقصد مورد نظر (۲) تخمینی از زمان یا فاصلهٔ رسیدن بدان مقصد. معیار هزینه می‌تواند تعداد گام (Number of Hops)، تأخیر برحسب میلی‌ثانیه، تعداد کل بسته‌های به صف شده در آن مسیر یا چیزی شبیه به اینها باشد.

فرض بر آن است که هر مسیریاب «فاصلهٔ» خود تا هر یک از همسایه‌هایش را می‌داند. اگر معیار بکار رفته تعداد گام باشد فاصلهٔ هر مسیریاب از همسایه‌هایش دقیقاً ۱ است. اگر معیار، طول صفّ باشد [یعنی تعداد بسته‌هایی که برای ارسال بر روی یک خط خروجی منتظر هستند] مسیریاب می‌تواند بسادگی هر یک از صفها را بررسی نماید. اگر معیار، تأخیر باشد مسیریاب می‌تواند با ارسال بسته‌های خاصی به نام Echo و دریافت بستهٔ پاسخ (که گیرنده به آن «مُهر زمان» (Timestamp) زده و سریعاً بر می‌گرداند) این تأخیر را مستقیماً اندازه بگیرد.

به عنوان مثال فرض کنید از میزان تأخیر به عنوان معیار بهینگی استفاده شده باشد و مسیریاب تأخیر خود تا هر یک از همسایه‌ها را بداند. هر T میلی‌ثانیه یکبار، تمام مسیریابها برای همسایه‌های خود فهرستی از تأخیر تخمینی در رسیدن به هر یک از مسیریابهای مقصد را ارسال می‌نماید. در ضمن فهرست مشابهی را از همسایه‌های خود دریافت می‌دارد. تصور کنید که یکی از این جداول از طرف مسیریاب X دریافت شود و نماد X_i مشخص کنندهٔ میزان تأخیر برای رسیدن به مسیریاب i باشد. اگر این مسیریاب بداند که تأخیر خودش تا X معادل m میلی‌ثانیه است می‌تواند نتیجه بگیرد که با تأخیر X_i+m میلی‌ثانیه به هر مسیریاب i می‌رسد. با انجام این محاسبه برای تمام جداولی که از همسایه‌ها می‌رسند مسیریاب می‌تواند نتیجه بگیرد که کدام یک از مسیرها بهتر هستند و مقدار تخمینی تأخیر و همچنین خط متناظر با بهترین مسیر را در جدول جدید خود درج می‌نماید. بخاطر داشته که جدول قدیمی هر مسیریاب در محاسبات دخالت داده نمی‌شود.

فرآیند بهنگام‌سازی، در شکل ۵-۹ به تصویر کشیده شده است. قسمت «الف» از شکل، ساختار یک زیرشبکه را نشان می‌دهد. چهار ستون سمت چپ در بخش «ب» شکل، «بردار تأخیر» چهار همسایهٔ مسیریاب J است که در لحظهٔ بهنگام‌سازی دریافت شده است. مسیریاب A ادعا کرده که تا B دوازده میلی‌ثانیه تأخیر دارد؛ ۲۵ میلی‌ثانیه تاخیر تا C، ۴۰ میلی‌ثانیه تأخیر تا D و به همین ترتیب. فرض کنید مسیریاب J تأخیر خود را تا A و I و H و K به ترتیب مقادیر ۸، ۱۰، ۱۲ و ۶ اندازه‌گیری کرده یا تخمین زده باشد.

حال ببینیم مسیریاب J چگونه مسیرهای جدید برای رسیدن به مسیریاب G را محاسبه می‌کند. J می‌داند که برای رسیدن به A، ۸ میلی‌ثانیه تأخیر خواهد داشت و A نیز ادعا کرده که قادر است با تأخیر ۱۸ میلی‌ثانیه به G برسد، لذا J متوجه می‌شود که اگر بسته‌های خود را که به مقصد G روانه هستند برای A بفرستد کل تأخیر حدود ۲۶ میلی‌ثانیه [۸ + ۱۸] خواهد بود. به همین طریق، J تأخیر خود تا G از طریق I، H و K را به ترتیب ۴۱ (۱۰ + ۳۱)، ۱۸ (۱۲ + ۶) و ۳۷ (۶ + ۳۱) میلی‌ثانیه محاسبه می‌نماید. از بین این مقادیر، ۱۸ بهترین است لذا J در جدول مسیریابی خود، درایه‌ای برای G تشکیل می‌دهد و در آن درج می‌کند که تأخیر رسیدن به G، ۱۸ میلی‌ثانیه است و مسیر مربوطه از طریق H می‌گذرد. [یعنی از آن به بعد هر بسته‌ای که به J وارد شود و بخواهد به G برود به سمت



(الف)

تخمین تاخیر دیگر مسیریابها تا J

To	A	I	H	K	Line
A	0	24	20	21	8 A
B	12	36	31	28	20 A
C	25	18	19	36	28 I
D	40	27	8	24	20 H
E	14	7	30	22	17 I
F	23	20	19	40	30 I
G	18	31	6	31	18 H
H	17	20	0	19	12 H
I	21	0	14	22	10 I
J	9	11	7	10	0 –
K	24	22	22	0	6 K
L	29	33	9	9	15 K

تاخیرخط JA = 8
تاخیرخط JI = 10
تاخیرخط JH = 12
تاخیرخط JK = 6

بردارهای دریافتی از چهار همسایه J

جدول مسیریابی جدید در J

(ب)

شکل ۵-۹. (الف) یک زیرشبکه (ب) جداول دریافتی از A، I، H و K و جدول مسیریابی جدید در J .

مسیریاب H ارسال خواهد شد و تأخیر تخمینی کل مسیر، ۱۸ میلی‌ثانیه خواهد بود. -م]

همین محاسبه برای دیگر مسیریابهای مقصد نیز انجام می‌شود و نهایتاً جدول مسیریابی جدید (نشان داده شده در سمت راست شکل ۵-۹) بدست خواهد آمد.

مشکل شمارش تا بی‌نهایت (Count-To-Infinity Problem)

مسیریابی بردار فاصله از دیدگاه تئوری به درستی کار می‌کند ولیکن در عمل اشکالات جدی خواهد داشت: اگرچه نهایتاً به جواب صحیح همگرا خواهد شد ولی این همگرایی بسیار کند خواهد بود.[1] بالاخص، این الگوریتم به «خبرهای خوب» واکنش سریع نشان می‌دهد در حالی که «خبرهای بد» را به آهستگی منتقل می‌کند![2] یک مسیریاب را در نظر بگیرید که در جدول او تأخیر رسیدن به مقصد X، مقدار بسیار بالا، محاسبه و درج شده است. اگر در مرحلهٔ بعدی مبادلهٔ جداول [یعنی در لحظهٔ بهنگام‌سازی]، مسیریاب همسایهٔ او (مثلاً A)، هزینهٔ کمتری را برای رسیدن به X اعلام کند، او بلافاصله مسیر قبلی خود را تغییر داده و از آن به بعد مسیر ارسال ترافیک به X را از طریق A انتخاب خواهد کرد. در هر بار مبادلهٔ بردارهای هزینه، «خبرهای خوب» بسرعت پردازش می‌شوند.

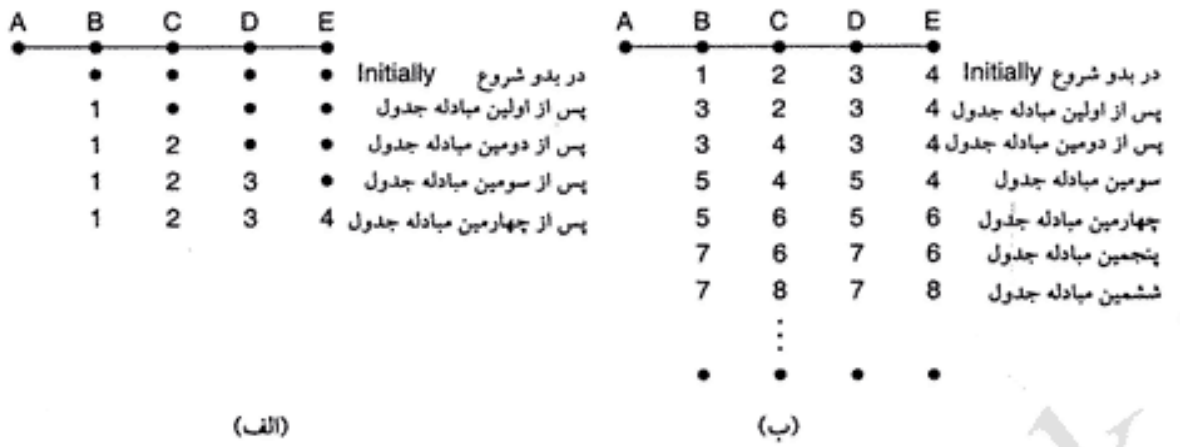
برای آنکه ببینیم چگونه اخبار خوب منتشر می‌شوند به زیرشبکهٔ پنج گره‌ای (خطی) شکل ۵-۱۰ دقت کنید که در آن معیار تأخیر «تعداد گام» است. فرض کنید A در همان ابتدا از کار افتاده و تمام مسیریابها از این موضوع آگاه هستند. به عبارت دیگر تمام مسیریابها تأخیر خود تا A را به مقدار بی‌نهایت تنظیم کرده‌اند.

وقتی A به کار می‌افتد دیگر مسیریابها با مبادلهٔ بردار [جدول هزینه] بلافاصله از این موضوع آگاه خواهند شد. برای سادگی فرض کرده‌ایم یک ناقوس بزرگ در جایی وجود دارد که بطور متناوب به صدا درمی‌آید تا همهٔ

---

۱. منظور از همگرایی رسیدن به یک جدول مسیریابی با مقادیر درست، واقعی و پایدار است. -م

۲. منظور از خبرهای خوب کاهش تأخیر در مسیرها، اضافه شدن لینک یا مسیریاب جدید و منظور از خبرهای بد خرابی یک خط، از کار افتادن یک مسیریاب یا افزایش تأخیر است. -م



شکل ۵-۱۰. مشکل «شمارش تا بی‌نهایت».

مسیریابها بطور همزمان عملیات مبادلهٔ بردارها (جداول) را شروع نمایند!! در اولین مبادله، B آگاه می‌شود که همسایهٔ سمت چپ او تأخیری معادل صفر به A دارد. B در جدول مسیریابی خود یک درایه (Entry) ایجاد و در آن تأخیر رسیدن به A را ۱ درج می‌نماید. هنوز بقیهٔ مسیریابها گمان می‌کنند که A غیرفعال است. تأخیر رسیدن به A در درایه‌های جدول مسیریابی تمام مسیریابها (در این لحظه) در سطر دوم از شکل ۵-۱۰-الف مشاهده می‌شود. در مبادلهٔ بعدی C متوجه می‌شود که B مسیری به A با طول ۱ دارد و به همین دلیل جدول خود را بهنگام می‌کند و در آن مسیری به A از طریق B با طول ۲ را مشخص می‌نماید ولی هنوز D و E از این خبر خوب آگاه نیستند. در یک زیرشبکه، اخبار خوب در هر بار مبادلهٔ جداول، یک گام به جلو منتشر می‌شود. اگر در یک زیرشبکه، طولانی‌ترین مسیر N گام باشد پس از N بار مبادله، همهٔ مسیریابها از اضافه شدن خطوط یا مسیریابهای جدید به زیرشبکه (خبر خوب)، آگاه خواهند شد.

حال به وضعیت شکل ۵-۱۰-ب نگاه کنید که در آن تمام خطوط و مسیریابها در ابتدا فعال هستند. فاصلهٔ مسیریابهای B، C، D و E تا A به ترتیب عبارتند از: ۱، ۲، ۳ و ۴. به ناگاه مسیریاب A از کار می‌افتد یا مثلاً خط بین A و B قطع می‌شود که هر دوی این حالات از دیدگاه B فرقی نمی‌کنند.

پس از اولین مبادلهٔ بسته، B از A چیزی نمی‌شنود. خوشبختانه C می‌گوید: «نگران نباش! من مسیری به A به طول ۲ دارم». B به درستی نمی‌داند که مسیری که C اعلام کرده از خود او (یعنی B) می‌گذرد. آنچه که B حدس زده آنست که شاید C ده خط دیگر دارد و می‌توان از طریق این خطوط با هزینهٔ ۲، به A رسید. در نتیجه، B گمان می‌کند که قادر است از طریق C با هزینهٔ ۳ به A برسد! D و E درایه‌های جدول مسیریابی خود را در اولین مبادله تغییر نمی‌دهند.

در دومین مبادله، C متوجه می‌شود که هر یک از همسایه‌های او ادعا کرده‌اند که مسیری به A با هزینهٔ ۳ دارند لذا یکی از آنها را انتخاب کرده و فاصلهٔ جدید رسیدن به A را به ۴ تنظیم می‌کند. (به گونه‌ای که در سطر سوم از شکل ۵-۱۰-ب نشان داده شده است.) مبادلات بعدی جداول، نتایج نشان داده شده در ادامهٔ شکل ۵-۱۰-ب را تولید خواهد کرد.

از این شکل باید روشن شده باشد که چرا خبرهای بد به کندی منتشر می‌شوند: همهٔ مسیریابها هزینهٔ جدید را یک واحد بیش از نتیجهٔ مینیمم‌گیری از هزینه‌های اعلام شده، در نظر می‌گیرند. [به عبارت دیگر هزینه‌های جدید براساس مقادیری محاسبه می‌شود که قدیمی و اشتباه است. -م] به تدریج مقدار فاصلهٔ درج شده در جدول تمام مسیریابها به سمت بی‌نهایت رشد می‌کند ولیکن تعداد دفعات مبادلهٔ جداول به عددی که برای مقدار «بی‌نهایت» در

نظر گرفته شده بستگی دارد. به همین دلیل بهترین کار آن است که مقدار بی‌نهایت، معادل طول بزرگترین مسیر در زیرشبکه به اضافهٔ ۱ ، در نظر گرفته شود. اگر معیار هزینه، «زمان تأخیر» باشد هیچ حد بالایی را نمی‌توان تعریف کرد مگر آنکه حد بالا را آنقدر زیاد فرض کنیم که با مسیری که بطور طبیعی تأخیر آن بالاست به عنوان مسیر از کار افتاده و خراب رفتار نشود. مشکل فوق‌الذکر به نام **«مشکل شمارش تا بی‌نهایت»** مشهور است. تلاشهایی برای حل این مشکل انجام گرفته (مثل روش split horizon with poisoned reverse که در RFC 1058 تشریح شده) ولی هیچکدام از آنها موفق عمل نکرده‌اند. اصل این مشکل از آنجایی ناشی شده که وقتی X به Y می‌گوید که مسیری به جایی دارد، Y بهیچوجه نمی‌تواند بفهمد که آیا خودش بر روی این مسیر قرار گرفته یا نه!

۵-۲-۵ مسیریابی حالت لینک (Link State Routing)

از «مسیریابی بردار فاصله» تا سال ۱۹۷۹ در ARPANET استفاده می‌شد و در همین ایام به «مسیریابی حالت لینک» (Link State) تغییر کرد. دو مشکل اساسی منجر به زوال آن شد: اول آنکه در این الگوریتم معیار تأخیر، طول صف[1] در نظر گرفته می‌شد و پهنای باند هر یک از خطوط در محاسبات انتخاب بهترین مسیر، دخالت داده نمی‌شد. در ابتدا تمام خطوط 56Kbps بودند لذا پهنای باند خط مورد مهمی نبود ولی پس از آنکه برخی از خطوط به سرعت 230 Kbps و برخی دیگر به 44 Mbps ارتقاء یافتند، بحساب نیاوردن پهنای باند، مشکلی عمده به حساب می‌آمد. البته این امکان وجود داشت که بتوان معیار تأخیر را به پهنای باند خط تغییر داد ولیکن مشکل دیگری ایجاد می‌شد و آن هم اینکه همگرایی الگوریتم بسیار طولانی می‌شد (بدلیل مشکل شمارش تا بی‌نهایت). به همین دلایل با الگوریتم کاملاً جدیدی که اکنون **«مسیریابی حالت لینک»** (LS) نامیده می‌شود، تعویض شد. امروزه گونه‌های متفاوتی از «مسیریابی حالت لینک» مورد استفاده قرار می‌گیرد.

ایدهٔ اصلی و زیربنای مسیریابی حالت لینک (LS) ، ساده است و می‌توان آن را در پنج بند بیان کرد. هر مسیریاب باید به ترتیب زیر عمل کند:

۱. همسایه‌های خود را شناسایی کرده و آدرسهای شبکهٔ آنها را بدست بیاورد.
۲. تأخیر یا هزینهٔ هر یک از همسایه‌های خود را اندازه‌گیری نماید.
۳. بسته‌ای بسازد و اطلاعاتی که از همسایه‌ها کسب کرده ، در آن جاسازی کند.
۴. این بسته را برای تمام مسیریابهای دیگر بفرستد.
۵. کوتاهترین مسیر رسیدن به دیگر مسیریابها را محاسبه نماید.

بدین ترتیب، توپولوژی کامل زیرشبکه و تمام تأخیرها (از طریق آزمایش) اندازه‌گیری شده و بین تمام مسیریابها توزیع می‌شود. برای پیدا کردن کوتاهترین مسیرها به تمام مسیریابهای زیرشبکه، می‌توان از «الگوریتم دایکسترا» بهره گرفت. در ادامه هر یک از این پنج مرحله را به تفصیل بررسی خواهیم نمود.

شناسایی همسایه‌ها

وقتی یک مسیریاب آغاز به کار می‌کند [بوت می‌شود] اولین وظیفهٔ او شناسایی همسایه‌های خودش است. این کار با ارسال یک بستهٔ خاص به نام **«بستهٔ سلام»** (HELLO Packet) بر روی تمام خطوط نقطه‌به‌نقطه انجام می‌شود. انتظار می‌رود که مسیریابهایی که در طرف مقابل هر خط هستند پاسخی برگردانده و خود را معرفی کنند. این نامها [یا به عبارت بهتر آدرس مسیریابها] باید جهانی، منحصر به فرد و یکتا باشند چراکه بعداً وقتی یک مسیریاب راه دور می‌شنود که سه مسیریاب همگی مثلاً به F متصلند، دانستن این موضوع که آیا هر سه F یکی هستند بسیار

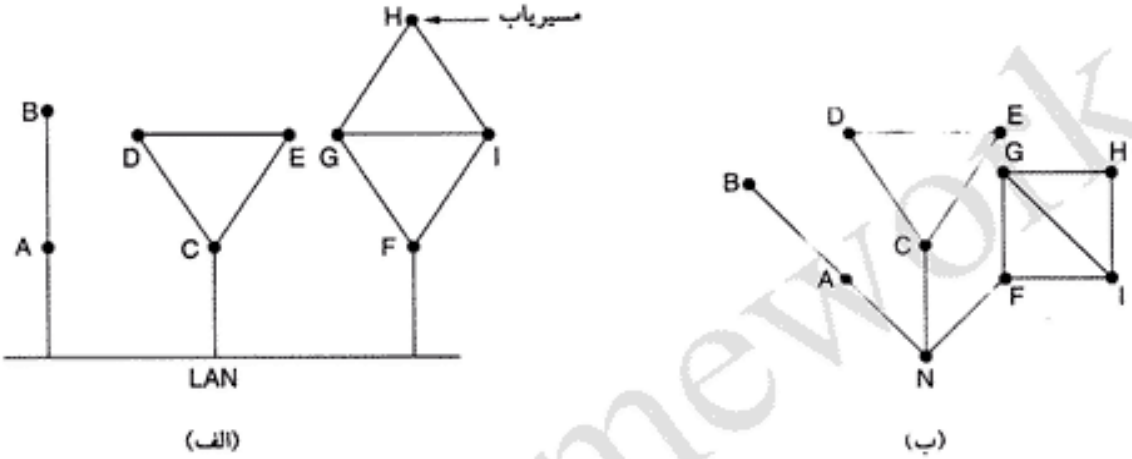
---

۱. تعداد بسته‌های به صف شدهٔ منتظر ارسال بر روی یکی از خطوط خروجی مسیریاب، به طول صف تعبیر می‌شود. -م

حیاتی است.

وقتی دو یا چند مسیریاب از طریق LAN به هم متصل شده باشند وضعیت اندکی پیچیده‌تر است. شکل ۵-۱۱-الف یک شبکهٔ LAN را نشان می‌دهد که سه مسیریاب A، C و F مستقیماً بدان متصل شده‌اند. بگونه‌ای که نشان داده شده این مسیریابها به یک یا چند مسیریاب دیگر نیز متصلند.

یک روش برای مدلسازی LAN آنست که همانند شکل ۵-۱۱-ب، آن را به عنوان یک گره در نظر بگیریم. در اینجا یک گره مصنوعی جدید به نام N معرفی کرده‌ایم که A و C و F بدانها متصل هستند. این واقعیت که رسیدن از A به C فقط از طریق LAN ممکن است با مسیر ANC بیان می‌شود.



شکل ۵-۱۱. (الف) نه مسیریاب و یک شبکهٔ LAN (ب) مدل گراف از شکل الف.

اندازه‌گیری هزینهٔ خط

«الگوریتم حالت لینک» نیازمند آن است که هر مسیریاب از تأخیر هر یک از همسایه‌های خود آگاه باشد (یا حداقل تخمینی معقول از تأخیر آنها داشته باشد.) راه مستقیم اندازه‌گیری این تأخیر آن است که یک بستهٔ خاص به نام Echo بر روی خط مورد نظر ارسال شده و طرف مقابل موظف به برگرداندن فوری آن بسته باشد. با اندازه‌گیری زمان رفت و برگشت این بسته و تقسیم آن بر ۲، مسیریاب فرستنده می‌تواند تخمینی معقول از تأخیر بدست بیاورد. برای بدست آوردن نتایج بهتر می‌توان این آزمون را چندین بار تکرار کرد و میانگین مقادیر اندازه‌گیری‌شده را در نظر گرفت. البته در این روش تلویحاً فرض بر آن گذاشته شده که تأخیرها متقارن[1] است در حالی که ممکن است همیشه اینگونه نباشد.

یک مسئلهٔ جالب آن است که آیا باید در اندازه‌گیری تأخیر، «میزان بار» (load) را نیز به حساب آورد؟ برای وارد کردن «میزان بار» در محاسبات، «تایمر سنجش زمان رفت‌وبرگشت» (Round-Trip Timer) باید زمانی آغاز به کار کند که بستهٔ Echo به **انتهای صف ارسال** وارد می‌شود. برای نادیده گرفتن میزان بار، تایمر زمانی شروع به اندازه‌گیری می‌کند که بستهٔ Echo به سر صف رسیده باشد.

می‌توان در خصوص هر دوی این روشها بحث کرد. در نظر گرفتن تأخیرات ناشی از ترافیک در محاسبات، بدین معنی است وقتی یک مسیریاب می‌خواهد از بین دو خط با پهنای باند مساوی یکی را انتخاب کند (در شرایطی که یکی از آنها برخلاف دیگری با بار سنگین مواجه است)، مسیریاب خطی را که فاقد بار است به عنوان

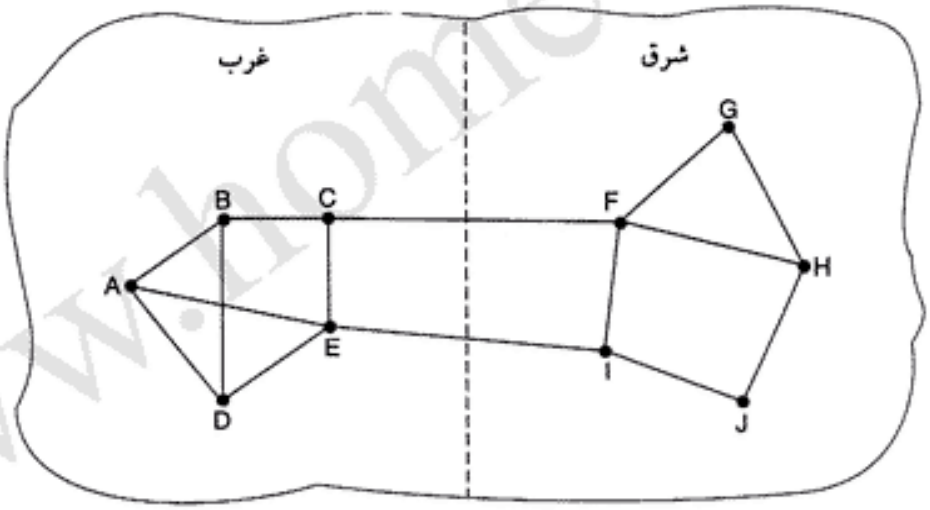
---

۱. متقارن بودن تاخیر بدین معناست که تاخیر رسیدن از A به B با تاخیر رسیدن از B به A یکسان است، در حالیکه در بسیاری از محیطها اینگونه نیست. -م

مسیر کوتاهتر در نظر می‌گیرد. چنین انتخابی، کارآیی بهتری را در پی خواهد داشت.

متأسفانه استدلالی بر علیه نظریهٔ «دخالت دادن میزان بار در محاسبهٔ تأخیر» وجود دارد: به زیرشبکه شکل ۵-۱۲ توجه کنید که در آن زیرشبکه به دو بخش شرقی و غربی تقسیم و توسط دو خط CF و EI بهم متصل شده‌اند.

فرض کنید که بیشتر حجم ترافیک بین شرق و غرب از طریق خط CF مبادله شود و در نتیجه این خط با بار سنگین و تأخیر بالایی مواجه است. در نظر گرفتن تأخیر انتظار (یعنی تأخیر صف) در محاسبات کوتاهترین مسیر باعث می‌شود که خط EI جلب توجه نماید. پس از آنکه جدول جدید مسیریابی تنظیم و اعمال شود خط EI به عنوان خط بهینه انتخاب شده و از آن به بعد بخش اعظم ترافیک شرق به غرب از طریق EI عبور خواهد کرد و طبعاً این خط با بار سنگین مواجه خواهد شد. طبعاً در بهنگام‌سازی بعدی، مجدداً خط CF به عنوان مسیر بهینه شناخته می‌شود. در نتیجه ممکن است جدول مسیریابی بطور نوسانی تغییر کرده و منجر به مسیریابی نامنظم شده و اشکالات بالقوهٔ فراوانی تولید کند. اگر از میزان بار چشمپوشی شده و فقط پهنای باند در نظر گرفته شود این مشکل رخ نخواهد داد. البته می‌توان بار را بر روی هر دو خط توزیع کرد ولیکن در این راه حل از مسیر بهینه، استفادهٔ کامل نخواهد شد. علیرغم این، برای اجتناب از بروز تغییرات نوسانی در انتخاب بهترین مسیر، شاید توزیع بار بر روی چند خط راهکاری معقول باشد (البته باید کسری از بار که بر روی هر خط توزیع می‌شود از قبل تعیین شده باشد).



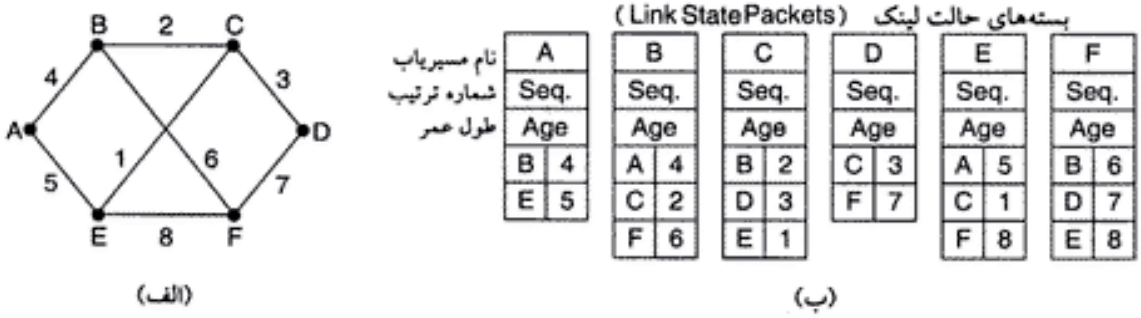
شکل ۵-۱۲. زیرشبکه‌ای که در آن دو بخش شرقی و غربی با دو خط به یکدیگر متصل شده‌اند.

ساخت بسته‌های حالت لینک (Building Link State Packet)

به محض آنکه اطلاعات لازم جهت مبادله گردآوری شد، گام بعدی هر مسیریاب، ساختن بسته‌ای است که این داده‌ها را در بر بگیرد.[1] در ابتدای هر بسته، هویت فرستندهٔ آن درج می‌شود، سپس فیلدهای «شماره ترتیب» و «طول عمر» (Age) بسته می‌آید (در مورد طول عمر بسته در ادامه توضیح خواهیم داد)؛ در آخر نیز فهرست همسایه‌ها و تأخیر رسیدن بدانها، مشخص می‌شود. در شکل ۵-۱۳-الف یک زیرشبکهٔ نمونه نشان داده شده که در آن، تأخیرها به صورت برچسب عددی، بر روی هر خط مشخص شده‌اند. بسته‌های LS ایجاد شده در هر یک

---

۱. «بسته‌های حالت لینک» را یک ساختمان دادهٔ استاندارد همانند یک استراکچر در زبان C در نظر بگیرید. از این به بعد این بسته‌ها را بسته‌های LS می‌نامیم. -م



شکل ۵-۱۳. (الف) یک زیرشبکه (ب) بسته‌های حالت لینک (LS) برای این زیر شبکه.

از شش مسیریاب ، در شکل ۵-۱۳-ب مشاهده می‌شود.

ساخت بسته‌های LS ساده است. مشکل‌ترین بخش قضیه تعیین زمانی است که مسیریابها باید به ساختن این بسته‌ها اقدام کنند. یک راه آن است که بطور متناوب و در فواصل زمانی مشخص ایجاد شوند. راه دیگر آن است که مسیریاب فقط زمانی مبادرت به ساخت این بسته ها کند که اتفاق خاصی رخ بدهد، مثلاً یکی از خطوط یا مسیریابهای همسایه از کار بیفتد یا مجدداً فعال شود یا ویژگیهای آن بطور محسوسی تغییر کند. [منظور از ویژگی، میزان تأخیر، بار، پهنای باند یا نظایر آنهاست. -م]

توزیع بسته‌های «حالت لینک»

ظریفترین بخش این الگوریتم، توزیع مطمئن بسته‌های LS است. به محض آنکه بسته‌ها، توزیع و در جداول اعمال شدند، آن مسیریاب که اول از همه این بسته‌ها را دریافت کند مسیرهای خود را تغییر می‌دهد [در حالیکه ممکن است بقیه هنوز چنین کاری نکرده باشند]. در نتیجه مسیریابهای متفاوت ممکن است برای لحظاتی از نسخه‌های متفاوتی از توپولوژی زیرشبکه استفاده کنند که این مسئله منجر به مشکلاتی در مسیریابی صحیح، بروز حلقهٔ بینهایت، از دسترس خارج شدن برخی از ماشینها و مشکلات دیگر شود.

در ابتدا به تشریح الگوریتم اساسی توزیع می‌پردازیم و در ادامه اصلاحاتی را بر روی آن انجام خواهیم داد. ایدهٔ اصلی آن است که برای توزیع بسته‌های LS از روش «سیل آسا» استفاده شود. برای آنکه کنترل این فرآیند را در دست داشته باشیم هر بسته دارای شمارهٔ ترتیب است؛ به ازای تولید هر بستهٔ جدید، به این شمارهٔ ترتیب یک واحد اضافه می‌شود. هرگاه مسیریابها بسته‌ای از این نوع را دریافت کنند زوج آیتم «آدرس مسیریاب مبداء , شمارهٔ ترتیب» آنرا در جایی ذخیره می‌کنند. وقتی یک بستهٔ LS جدید وارد شود ابتدا بررسی می‌شود که آیا این بسته قبلاً نیز دریافت شده است. اگر بسته جدید بود بر روی تمام خطوط خروجی (به استثنای خطی که از روی آن دریافت شده) ارسال می‌شود ولیکن اگر تکراری بود نادیده گرفته می‌شود. اگر بسته‌ای دریافت شود و شمارهٔ ترتیب آن از بزرگترین شماره‌ای که تاکنون دریافت شده، کوچکتر باشد به عنوان بستهٔ قدیمی تلقی و در نظر گرفته نخواهد شد چرا که مسیریاب، نسخهٔ جدیدتری از آن را در اختیار دارد.

این الگوریتم چند مشکل دارد ولی این مشکلات قابل مدیریت و کنترل هستند: اول آنکه اگر شمارهٔ ترتیب با افزایشهای متوالی، در جایی به صفر برگردد مشکلاتی بروز خواهد کرد. راه حل آن است که از یک شماره ترتیب ۳۲ بیتی استفاده شود؛ در این صورت اگر در هر ثانیه یک بستهٔ LS ارسال شود، ۱۳۷ سال طول می‌کشد تا این عدد به صفر برگردد لذا احتمال بروز چنین مشکلی قابل چشمپوشی است!

دوم آنکه اگر یکی از مسیریابها از کار بیفتد روند شمارهٔ ترتیب بسته‌های خود را از دست خواهد داد و اگر بعد از راه اندازی مجدّد، شمارهٔ ترتیب را از صفر شروع کند بسته‌های ارسالی او تکراری تلقی شده و

در نظر گرفته نمی‌شود.

سوم آنکه اگر شمارهٔ ترتیب به نحوی دچار خطا شود و مثلاً در اثر یک بیت خطا شمارهٔ ۴ به ۶۵۵۴۰ تبدیل شود از آن به بعد بسته‌های ۵ تا ۶۵۵۴۰ به عنوان بستهٔ قدیمی دور انداخته خواهد شد چرا که گمان می‌رود که شمارهٔ فعلی ۶۵۵۴۰ است.

راه حل تمام این مشکلات آن است که پس از شمارهٔ ترتیب، طول عمر بسته نیز درج گردد و با گذشت هر ثانیه یک واحد از آن کم شود. وقتی عمر بسته به صفر برسد اطلاعاتی که از آن مسیریاب دریافت شده باید حذف گردد. در یک روال طبیعی (مثلاً هر ده ثانیه یکبار)، بسته‌های جدید LS وارد مسیریاب شده و طبعاً اطلاعات قبلی بطور منظم و قبل از انقضای مهلت اعتبارشان تازه‌سازی می‌شوند؛ بدین ترتیب اعتبار اطلاعات مربوط به هر مسیریاب زمانی منقضی خواهد شد که آن مسیریاب از کار بیفتد (یا آنکه مثلاً در اثر هر گونه رخداد، متوالیاً شش بستهٔ LS از آن مسیریاب دریافت نشود). فیلد «طول عمر» (Age) در خلال فرآیند ارسال سیل‌آسا و توسط هر مسیریاب نیز یک واحد کاهش می‌یابد تا مطمئن شویم که هیچ بسته‌ای نمی‌تواند بطور نامحدودی زنده و در زیرشبکه سرگردان بماند. (هر گاه عمر بسته صفر شود، حذف خواهد شد.)

چند اصلاح در این الگوریتم، آنرا قدرتمندتر خواهد کرد. وقتی یک بستهٔ LS برای ارسال به روش سیل‌آسا وارد یک مسیریاب می‌شود بلافاصله در صف ارسال وارد نخواهد شد بلکه ابتدا به یک «فضای انتظار» وارد می‌شود تا برای مدتی کوتاه، منتظر بماند. اگر قبل از ارسال بستهٔ فعلی، بستهٔ مشابه دیگری از همان مبداء وارد شود، شمارهٔ ترتیب این دو بسته مقایسه شده و در صورت مساوی بودن، بستهٔ تکراری حذف خواهد شد و در صورت تکراری نبودن از بستهٔ قدیمی‌تر صرفنظر می‌شود. برای پیشگیری از خطاهای احتمالی بر روی خطوط مستقیم بین دو مسیریاب، دریافت تمام بسته‌های LS ورودی، تصدیق (Ack) خواهد شد.[1] وقتی یکی از خطوط خروجی مسیریاب آزاد گردد «فضای انتظار» به روش Round Robin (نوبت چرخشی) پویش می‌شود تا یک بستهٔ LS یا یک بستهٔ اعلام وصول (یعنی Ack) برای ارسال انتخاب گردد.

در جدول شکل ۵-۱۴ ساختمان دادهٔ مورد استفاده در مسیریاب B که برای زیرشبکهٔ شکل ۵-۱۳-الف ایجاد شده، دیده می‌شود. در این جدول، هر سطر متناظر با یک بستهٔ LS تازه وارد است که هنوز بطور کامل پردازش نشده است. در این جدول مبداء هر بسته، شمارهٔ ترتیب، طول عمر آن و مقداری داده ثبت می‌شود. بعلاوه به ازای هر یک از سه خط متصل به B (یعنی خطوط A و C و F) دو پرچم «ارسال» و «اعلام وصول»[2] در نظر گرفته شده است. «پرچم ارسال» بدین معناست که بسته باید بر روی خط مشخص شده، ارسال شود. «پرچم اعلام وصول» بدین معناست که دریافت این بسته باید بر روی خط مربوطه، به آگاهی طرف مقابل برسد.

از جدول شکل ۵-۱۴، مشخص است که یک بستهٔ LS مستقیماً از A دریافت شده و طبق آنچه که بیتهای پرچم نشان می‌دهند این بسته باید برای C و F ارسال شود و ضمناً وصول آن به A اعلام گردد. به طور مشابه بسته‌ای که از F آمده باید به A و C نیز ارسال شده و دریافت آن به F اعلام گردد.

با این حال شرایط بستهٔ سوم که از E می‌رسد متفاوت است. این بسته دو بار، یکبار از طریق EAB و یکبار از طریق EFB دریافت شده است. در نتیجه بنحوی که بیتهای پرچم نشان می‌دهند، این بسته باید برای C ارسال شود ولیکن فقط کافی است وصول آن به A و F اعلام گردد.

اگر بسته‌ای تکراری دریافت شود در حالی که نسخهٔ اصلی آن هنوز در بافر موجود است، بیتهای پرچم باید تغییر کنند. به عنوان مثال اگر بستهٔ متعلق به C قبل از آنکه مطابق با درایهٔ چهارم (4th Entry) برای A و F ارسال

---

۱. دریافت هر بستهٔ LS به مسیریاب همسایه (که آنرا فرستاده) اعلام خواهد شد نه مبداء آن. -م

۲. Send Flag and Achnowledgement Flag

Source (مبدا)	Seq.	Age	Send flags A	Send flags C	Send flags F	ACK flags A	ACK flags C	ACK flags F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

شکل ۵-۱۴. بافر بسته‌های LS برای مسیریاب B در شکل ۵-۱۳.

شود یکبار دیگر از طریق F دریافت گردد، شش بیت پرچم به 100011 تغییر خواهد کرد بدین معنا که باید وصول بسته به F اعلام شود ولی لازم نیست خود بسته برای F ارسال گردد.

محاسبهٔ مسیرهای جدید

به محض آنکه مسیریاب مجموعهٔ کامل بسته‌های LS را گردآوری کرد می‌تواند گراف کل زیرشبکه را تشکیل بدهد چرا که همهٔ لینکها و هزینهٔ آنها مشخص شده است. هر لینک در حقیقت دو بار معرفی شده است، یکبار در هر جهت. [چراکه مثلاً لینک بین A و B، یکبار توسط A و یکبار توسط B شناسایی و معرفی می‌شوند. -م]

حال می‌توان الگوریتم دایکسترا را به صورت محلی اجرا کرد و کوتاهترین مسیر ممکن به تمام نقاط مقصد در زیرشبکه را بدست آورد.[1] نتیجهٔ این الگوریتم می‌تواند در جدول مسیریابی درج شده و مسیریابها عملکرد طبیعی خود را از سر بگیرند.

برای زیرشبکه‌ای با n مسیریاب که هر کدام k همسایه دارند، حافظه لازم برای ذخیرهٔ داده‌های جدول مسیریابی، متناسب با nxk است که در زیرشبکه‌های بزرگ می‌تواند مشکل ایجاد کند. در ضمن زمان محاسبه ، نیز می‌تواند مسئله‌ساز باشد ولیکن علیرغم همهٔ اینها «الگوریتم مسیریابی حالت لینک» (Link State Routing) در محیطهای واقعی بخوبی کار می‌کند.

با این حال، هرگونه اشکال سخت‌افزاری یا نرم‌افزاری می‌تواند مشکلات بسیار جدی برای این الگوریتم به بار بیاورد. (هم‌چنین در عملکرد الگوریتم دیگر مسیریابها نیز مشکل ایجاد می‌کند.) به عنوان نمونه، اگر یک مسیریاب ادعّا کند که خطی در اختیار دارد در حالی که نداشته باشد یا فراموش کند خطی را که در اختیار دارد اعلام کند، گراف زیرشبکه صحیح نخواهد بود. همچنین اگر یک مسیریاب در ارسال بسته‌های LS مشکل بهم بزند یا آنها را در حین ارسال خراب کند مشکلاتی جدّی بروز خواهد کرد. سرانجام آنکه اگر حافظهٔ مسیریاب سرریز شود یا نتیجهٔ محاسبات مسیر اشتباه باشد رخدادهای ناگواری در مسیریابی بسته‌ها بوقوع می‌پیوندد. هر گاه رشد زیرشبکه به دهها، صدها یا حتی هزاران گره برسد احتمال از کار افتادن ناگهانی یک یا چند مسیریاب را نمی‌توان نادیده گرفت. تنها راه ممکن آن است که تلاش شود تا در هنگام بروز چنین رخدادهایی میزان آسیبها و مشکلات در حد محدودی کنترل شود. پرلمن (Perlman, 1988) این مشکلات و راه حلهای آنها را تشریح کرده است.

روش «مسیریابی حالت لینک» بطور گسترده‌ای در شبکه‌های واقعی بکار گرفته شده است لذا مختصری در

---

۱. اجرای الگوریتم دایکسترا به صورت محلی بدین معناست که هر یک از مسیریابها خودشان مستقل از دیگری آن را اجرا می‌کنند. -م

مورد چند پروتکل نمونه که از آن بهره گرفته‌اند خالی از لطف نیست. پروتکل OSPF که در اینترنت بسیار رایج است از همین الگوریتم استفاده می‌کند. در بخش ۵ـ۶ـ۴ ، OSPF را تشریح خواهیم کرد.

یکی دیگر از «پروتکلهای مبتنی بر حالت لینک»، پروتکل IS-IS[1] است که توسط DECnet طراحی شد و بعداً توسط ISO جهت بکارگیری در کنار پروتکل بی‌اتصال لایهٔ شبکهٔ خود یعنی CLNP ، مورد پذیرش و تأیید قرار گرفت. البته بعداً در آن تغییراتی ایجاد شد تا بتواند با پروتکلهای دیگری مثل IP نیز کار کند. پروتکل IS-IS در بخشهایی از ستون فقرات شبکهٔ اینترنت (مثلاً در ستون فقرات قدیمی متعلق به NSFNET) و در برخی از سیستمهای دیجیتال سلولی مثل CDPD بکار رفته است. شرکت Novell Netware نیز گونهٔ ساده‌تری از IS-IS را برای هدایت بسته‌های IPX خود بکار گرفته است.

IS-IS اساساً تصویری از توپولوژی مسیریاب را در زیرشبکه توزیع می‌کند و از طریق آن کوتاهترین مسیرها محاسبه می‌گردد. هر مسیریاب در هنگام اعلام اطلاعات مربوط به «حالت لینک» (LS) مشخص می‌کند که به چه آدرسهایی مستقیماً دسترسی دارد. (آدرسها مربوط به لایهٔ شبکه و جهانی هستند). این آدرسها می‌توانند IP ، IPX ، AppleTalk یا هر آدرس دیگر باشند. IS-IS همچنین می‌تواند بطور همزمان از چندین پروتکل لایهٔ شبکه پشتیبانی نماید.

بسیاری از نوآوریهایی که در IS-IS طراحی شده بود بعداً مورد پذیرش و استفادهٔ OSPF قرار گرفت. (OSPF چندین سال پس از IS-IS ابداع شد.) برخی از اینها عبارت بودند از روشی برای خاتمهٔ خودکار فرآیند ارسال بسته‌ها به روش سیل‌آسا، مفهوم «مسیریاب برگزیده» (Designated Router) در شبکهٔ LAN و روش محاسبه و پشتیبانی از تقسیم مسیر و همچنین تعریف معیارهای چندگانهٔ هزینه. در نتیجه اختلاف ناچیزی بین IS-IS و OSPF وجود دارد. مهمترین اختلاف این دو آن است که IS-IS به گونه‌ای بسته‌های LS را تعریف و کد کرده است که بسادگی و بطور همزمان می‌تواند اطلاعاتی در خصوص چندین پروتکل لایهٔ شبکه با خود حمل کند، خصوصیتی که OSPF فاقد آن است. این حسن در محیطهای بزرگ که با چندین پروتکل مختلف کار می‌کنند بسیار ارزشمند است.

۵ـ۲ـ۶ مسیریابی سلسله‌مراتبی (Hierachical Routing)

با رشد اندازهٔ شبکه، جداول مسیریابی به همان نسبت رشد می‌کنند. جداول مسیریابی رو به رشد، نه تنها حافظهٔ بیشتری مصرف می‌کنند بلکه به زمان CPU بیشتری برای پویش و جستجوی این جدول و همچنین پهنای باند زیادتری برای ارسال بسته‌های LS (بسته‌های گزارش وضعیت لینک) نیاز خواهد بود. ممکن است رشد شبکه بدان حد برسد که دیگر امکان نگهداری یک «درایه» (Entry) به ازای هر مسیریاب، در جدول مسیریابی وجود نداشته باشد و مسیریاب مجبور به مسیریابی سلسله‌مراتبی (همانند شبکهٔ تلفن) شود.

وقتی از مسیریابی سلسله‌مراتبی استفاده می‌شود، مسیریابها به تعدادی «منطقه» (Region) تقسیم می‌شوند؛ هر مسیریاب تمام جزییات منطقهٔ خود و مسیرهای دقیق رسیدن به هر مقصد در منطقهٔ خود را می‌داند ولی چیزی در خصوص ساختار داخلی مناطق دیگر نمی‌داند. وقتی شبکه‌های مختلف بهم متصل می‌شوند طبیعی است که هر یک از آنها را به عنوان مناطق مجزا در نظر بگیریم تا مسیریابهای درون یک منطقه از دانستن ساختار توپولوژی ناحیهٔ دیگر بی‌نیاز و فارغ باشند.

در شبکه‌های عظیم ممکن است سلسله‌مراتب دو سطحی کفایت نکند. ممکن است نیاز باشد که هر «منطقه» (Region) به تعدادی «دسته» (Cluster)، هر «دسته» به تعدادی «ناحیه» (Zone) و هر ناحیه به تعدادی «گروه»

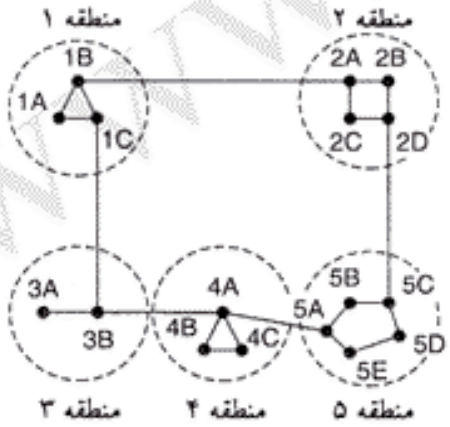
---

۱. Intermediate System-Intermediate System

(Group) تقسیم شود و به همین ترتیب ادامه یابد تا جایی که برای تقسیم‌بندی بیشتر اسمی باقی نماند!! به عنوان مثالی از سلسله‌مراتب چندسطحی، به چگونگی هدایت یک بسته از برکلی کالیفرنیا به مالیندی در کنیا دقت کنید. مسیریاب واقع در برکلی جزییات توپولوژی زیرشبکه خود در کالیفرنیا را می‌داند ولیکن تمام ترافیک متعلق به خارج از ایالت کالیفرنیا را به مسیریاب لوس آنجلس می‌فرستد. مسیریاب واقع در لوس آنجلس تنها قادر به مسیریابی ترافیک داده‌ها بین مسیریابهای داخل کشور است و هر گونه ترافیک خارجی را به نیویورک می‌فرستد. مسیریاب واقع در نیویورک به گونه‌ای برنامه‌ریزی شده که این بسته را به سمت مسیریابی که در کشور مقصد مسئول دریافت ترافیک خارجی است (مثلاً مسیریاب نایروبی) بفرستد. نهایتاً این بسته در کنیا به همین سیاق مسیر خود را ادامه می‌دهد تا به مالیندی برسد.

در شکل ۵-۱۵ یک نمونهٔ کمّی از مسیریابی سلسله‌مراتبی دو سطحی با پنج منطقه را ارائه کرده است. مطابق با شکل ۵-۱۵-ب جدول کامل مسیریابی در مسیریاب 1A دارای ۱۷ «درایه» است. وقتی مسیریابی به صورت سلسله‌مراتبی انجام می‌شود به ازای هر مسیریاب محلی یک درایه در جدول مسیریابی درج می‌شود ولیکن مناطق دیگر فقط در یک مسیریاب خلاصه می‌شوند لذا کل ترافیک مربوط به منطقهٔ ۲ از خط 1B-2A عبور خواهد کرد و مابقی ترافیک خارجی از طریق 1C-3B هدایت خواهد شد. مسیریابی سلسله‌مراتبی، تعداد درایه‌های جدول مسیریابی 1A را از ۱۷ به ۷ تا کاهش می‌دهد. [۳ تا برای مسیریابهای داخل منطقه و ۴ تا به ازای مناطق دیگر] هر چه نسبت تعداد مناطق به تعداد مسیریابهای درون هر منطقه افزایش یابد میزان صرفه‌جویی در فضای جدول بیشتر خواهد بود.

متأسفانه صرفه‌جویی در فضای حافظه رایگان بدست نمی‌آید. بهایی که باید برای آن پرداخت، افزایش طول مسیرها [یا به عبارتی عدم بهینگی کامل مسیرها] است. به عنوان مثال بهترین مسیر از 1A به 5C از منطقهٔ ۲ می‌گذرد در حالی که در مسیریابی سلسله‌مراتبی مسیر تمام ترافیک متعلق به ناحیهٔ ۵ از منطقهٔ ۳ تعیین شده چرا که برای



(الف)

جدول کامل مسیریابی 1A

(مقصد) Dest.	Line	Hops (گام)
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

(ب)

جدول مسیریابی سلسله‌مراتبی 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(ج)

شکل ۵-۱۵. مسیریابی سلسله‌مراتبی.

بیشتر مسیریابهای مقصد واقع در ناحیهٔ ۵، مسیر عبوری از منطقهٔ ۲ مناسبتر است.

سؤال قابل توجه آن است که وقتی یک شبکهٔ واحد، بشدت رشد می‌کند، سلسله‌مراتب باید چندسطحی باشد؟ به عنوان مثال زیرشبکه‌ای با ۷۲۰ مسیریاب را در نظر بگیرید. اگر سلسله مراتب وجود نداشته باشد هر مسیریاب در جدول مسیریابی خود به ۷۲۰ درایه نیاز خواهد داشت. اگر زیرشبکه به ۲۴ منطقه و ۳۰ مسیریاب در هر منطقه تقسیم شود، هر مسیریاب به ۳۰ درایه برای مسیریابهای محلی خود به علاوه ۲۳ درایه برای مناطق خارجی، نیاز دارد و بدین ترتیب جدول جمعاً ۵۳ درایه خواهد داشت. اگر سلسله مراتب، سه‌سطحی انتخاب شده و جمعاً هشت «دسته» (Cluster)، هر دسته شامل ۹ ناحیه و در هر ناحیه ۱۰ مسیریاب تعریف شده باشد، هر مسیریاب در جدول خود به ۱۰ درایه برای مسیریابهای محلی خود، ۸ درایه برای مسیریابی در نواحی داخل دستهٔ خودش و ۷ درایه برای دسته‌های خارجی (Distant Cluster) یعنی جمعاً ۲۵ درایه نیاز دارد. دو پژوهشگر بنامهای Kamoun و Kleinrock (۱۹۷۹) پی بردند که بهترین تعداد سطوح سلسله‌مراتب در زیرشبکه‌ای با N مسیریاب، معادل Ln(N) است و هر مسیریاب به جمعاً e.ln(N) عدد درایه نیاز خواهد داشت. همچنین نشان دادند که افزایش طول مؤثر مسیرها یا به عبارت دیگر کاهش بهینگی مسیرها، که از مسیریابی سلسله‌مراتبی منشاء می‌گیرد بقدر کافی کم و نتیجه معمولاً قابل قبول است.

۷-۲-۵ مسیریابی فراگیر (Broadcost Routing)

در برخی از کاربردها، ماشینهای میزبان نیازمند ارسال پیام به همهٔ ماشینهای شبکه یا تعدادی از آنها هستند. به عنوان مثال برای خدمات توزیع گزارشات هواشناسی، بهنگام‌سازی اطلاعات بازار سهام یا برنامه‌های زندهٔ رادیویی، راهکار مناسبتر آنست که داده‌ها را بصورت پخش فراگیر برای تمام ماشینها ارسال کرده و آنها را آزاد بگذاریم تا در صورت تمایل بسته‌های داده را دریافت کرده و بخوانند. «ارسال همزمان یک بسته به تمام ماشینهای مقصد، اصطلاحاً پخش فراگیر (Broadcasting) نامیده می‌شود و راهکارهای متنوعی برای اجرای آن پیشنهاد شده است.»

یکی از روشهای پخش فراگیر که به هیچ ویژگی خاصّی از زیرشبکه نیاز ندارد آن است که ماشین مبداء هر بستهٔ خود را بطور جداگانه برای یکایک ماشینهای مقصد بفرستد. این روش نه تنها پهنای باند را تلف خواهد کرد بلکه ماشین مبداء نیازمند آن است که فهرست کاملی از تمام ماشینهای مقصد در اختیار داشته باشد. اگرچه گاهی در عمل این روش تنها راه ممکن است ولیکن روش چندان مطلوبی نیست.

روش ارسال سیل‌آسا، نامزد دیگری برای پخش فراگیر محسوب می‌شود. هر چند روش سیل‌آسا، راهکار نامناسبی برای شبکه‌های نقطه‌به‌نقطه است ولیکن برای ارسال فراگیر می‌توان بر روی آن حساب باز کرد، خصوصاً وقتی که هیچیک از راهکارهایی که در ادامه معرفی می‌شوند عملی نباشد. استفاده از روش ارسال سیل‌آسا همان مشکلی را دارد که در الگوریتم مسیریابی نقطه‌به‌نقطه به آن اشاره کردیم: یعنی بسته‌های بسیار زیاد تولید می‌شود و پهنای باند بسیار وسیعی مصرف و تباه خواهد شد.

الگوریتم سوم «**مسیریابی چندمقصدی**» (Multidestination Routing) نام دارد. اگر از این روش استفاده شود هر بسته فهرستی از کلیهٔ مقصدهای مورد نظر یا «**نقشه‌ای بیتی**» (Bitmap) از این مقصدها با خود حمل می‌کند. هر گاه بسته‌ای به یک مسیریاب برسد، آن مسیریاب فهرست مقصدهای بسته را بررسی می‌کند تا مجموعهٔ خطوط خروجی منتهی به هر مقصد تعیین شوند. (هر خط خروجی که حداقل به یکی از مقاصد مورد نظر منتهی شود، انتخاب می‌گردد.) مسیریاب به ازای هر یک از خطوط خروجی تعیین شده، یک نسخهٔ جدید از آن بسته را تولید کرده و در آن بسته فقط آدرس مقصدهایی را قرار می‌دهد که مسیر آنها بر روی خط خروجی مربوطه است. [بقیهٔ آدرسهای مقصد حذف می‌شوند چرا که برای دیگر مقصدها خط خروجی جدا انتخاب و نسخه‌ای جداگانه تولید می‌شود. -م] در نتیجه مجموعهٔ آدرسهای مقصد هر بسته بر روی خطوط جداگانه تقسیم و توزیع می‌شود.

پس از چندین گام، هر بسته فقط یک آدرس مقصد با خود حمل می‌کند و با آن به عنوان یک بستهٔ معمولی رفتار می‌شود. «مسیریابی چندمقصدی» همانند بسته‌های مستقل و با آدرس جداگانه عمل می‌کنند، با این تفاوت که بجای ارسال چندین بسته بر روی یک مسیر مشابه، فقط یک بسته ارسال و بدین نحو فقط هزینهٔ ارسال یک بسته پرداخت می‌شود. [در این روش، هر بسته با رسیدن به یک مسیریاب به تعداد کافی تکثیر می‌شود. -م]

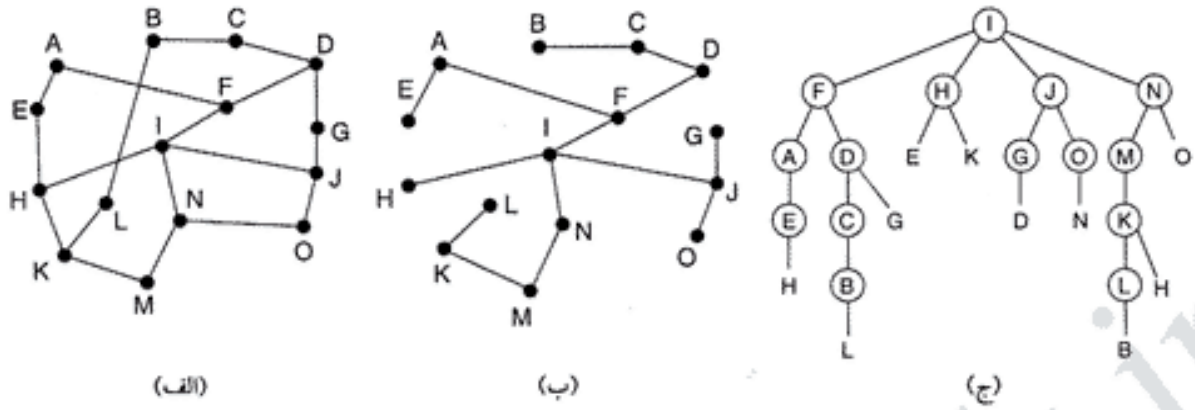
الگوریتم چهارم برای پخش فراگیر آن است که مسیریاب آغازگر این فرآیند، از درخت Sink Tree یا هر درخت پوشای مناسب (Spanning Tree) استفاده نماید. یک «**درخت پوشا**» زیر مجموعه‌ای از کلّ زیرشبکه است که تمام مسیریابها را در برمی‌گیرد ولیکن هیچ حلقه‌ای در آن نیست. اگر هر مسیریاب بداند که کدامیک از خطوط او در «درخت پوشا» قرار گرفته‌اند، می‌تواند یک بستهٔ فراگیر ورودی را فقط بر روی خطوطی در خروجی، بفرستد که بر روی این درخت پوشا هستند. (البته به استثنای خطی که بسته از آن دریافت شده است.) بهره‌وری پهنای باند در این روش بسیار عالی است و برای انجام کار، کمترین بستهٔ لازم تولید خواهد شد. تنها مشکل این روش آن است که هر مسیریاب برای بکارگیری این روش باید دانش و اطلاعات کافی در خصوص درختهای پوشا در زیرشبکه داشته باشد. برخی اوقات این اطلاعات در دسترس هستند (مثلاً در مسیریابی حالت لینک ـ LS ـ) ولی گاهی موجود نیستند (مثل مسیریابی بردار فاصله ـ DV ـ)

آخرین الگوریتم پخش فراگیر، آنست که حتی وقتی مسیریاب چیزی در خصوص درختهای پوشا نمی‌داند تلاش کند بصورت تقریبی ، رفتار الگوریتم درخت پوشا را از خود نشان بدهد! این ایده، «**هدایت بر روی مسیر معکوس**» (Reverse Path Forwarding) نامیده شده و به نظر بسیار ساده می‌رسد. وقتی یک بسته از نوع پخش فراگیر به یک مسیریاب وارد می‌شود آن مسیریاب ابتدا بررسی می‌کند که «آیا بسته از روی همان خطی وارد شده که بسته‌های معمولی برای ارسال به مبداء آن بسته، بر روی همان خط ارسال می‌شوند؟» اگر اینگونه باشد این احتمال بالا وجود دارد که بسته خودش در مسیر بهینه قرار دارد و طبعاً اولین نسخهٔ بسته‌ای است که او دریافت داشته است. در این حالت مسیریاب این بسته را بر روی تمام خطوط خود (به استثنای خطی که بسته از آن وارد شده) کپی و ارسال می‌کند. اگر بسته از روی خطی وارد شود که آن خط بر روی مسیر بهینه برای رسیدن بدان مبداء نیست بسته نادیده گرفته می‌شود چرا که احتمالاً تکراری است.

مثالی از روش «هدایت بر روی مسیر معکوس» در شکل ۵-۱۶ نشان داده شده است. بخش (الف) زیرشبکه را نشان می‌دهد و بخش (ب) درخت Sink Tree به مبداء مسیریاب I و بخش (ج) عملکرد الگوریتم «هدایت بر روی مسیر معکوس» را به تصویر کشیده است. در اولین گام، I بسته‌ای فراگیر برای F و H و J و N (یعنی گره‌های سطح دوی درخت) می‌فرستد. هر یک از این بسته‌ها از طریق خطّی دریافت می‌شوند که بر روی مسیری بهینه به I قرار دارند[1] (با فرض آنکه مسیرهای بهینه از F و H و J و N به مسیریاب I ، بر روی درخت Sink Tree قرار گرفته‌اند)؛ در شکل به دور نام هر مسیریاب که بر روی مسیری بهینه به I قرار گرفته یک دایره ترسیم شده است. در گام دوم، ۸ بسته تولید می‌شود (در هر مسیریاب که در گام اول بسته‌ای را دریافت کرده، ۲ بسته تولید می‌گردد) از این هشت بسته ۵ عدد به مسیریابهایی می‌رسند که بر روی مسیر بهینه به سمت I قرار دارند [یعنی A و D و G و O و M؛ سه بستهٔ دیگر چون بر روی مسیر بهینه به I نیستند پس از دریافت می‌شوند. -م]. از شش بسته‌ای که در گام سوم تولید شده فقط سه تای آنها از طریق خطی که بر روی مسیر بهینه به I قرار دارد دریافت شده‌اند (از طریق C و E و K) و بقیه تکراری تلقی می‌شوند. پس از پنج گام و تولید ۲۴ بسته، فرآیند ارسال فراگیر به پایان می‌رسد، درحالیکه اگر ارسال بسته‌ها دقیقاً از طریق درخت Sink Tree انجام شود، تعداد گامها ۴ و کلّ بسته‌های تولیدی

---

۱. بعبارت دیگر مسیریابهای F ، H ، J ، N همگی برای ارسال یک بستهٔ معمولی برای مبداء بسته یعنی I ، از همان خطی استفاده می‌کنند که بستهٔ پخشی از روی همان خط وارد شده است. -م



شکل ۵-۱۶. «هدایت بر روی مسیر معکوس» (الف) یک زیرشبکه (ب) یک درخت Sink Tree (ج) درختی که در روش هدایت بر روی مسیر معکوس ساخته می‌شود.

۱۴ عدد خواهد بود.

مزیت اساسی روش «هدایت بر روی مسیر معکوس» آن است که هم در حد معقولی کارآمد و هم پیاده‌سازی آن ساده است. در این روش هر مسیریاب نیازی به دانستن درختهای پوشا (Spanning Tree) ندارد و سربار ناشی از قرار دادن فهرست مقاصد چندگانه در بسته‌های فراگیر تحمیل نخواهد شد. همچنین این روش نیاز به مکانیزم خاصی جهت خاتمه دادن به پروسهٔ تولید نامحدود بسته‌ها ندارد در حالی که در روش ارسال سیل‌آسا به چنین مکانیزمی نیاز است (یعنی در این روش به تمهیداتی مثل درج شمارندهٔ گام در هر بسته یا اطلاع قبلی از قطر زیرشبکه یا ذخیرهٔ فهرست بسته‌هایی که تاکنون از یک مبداء منتشر شده‌اند، نیاز نیست.)

۸-۲-۵ مسیریابی چندپخشی (Multicast Routing)

در برخی از کاربردها نیاز است که پروسه‌های مجزا و پراکنده، در یک گروه با یکدیگر کار کنند؛ به عنوان نمونه می‌توان به گروهی از پروسه‌ها که یک «سیستم پایگاه توزیع‌شدهٔ اطلاعات»[1] را پیاده‌سازی کرده‌اند، اشاره نمود. در چنین شرایطی، بطور متناوب لازم می‌شود که یک پروسه برای دیگر پروسه‌های عضو گروه خود، پیام بفرستد. اگر این گروه کوچک باشد می‌توان پیام را بطور جداگانه برای یکایک اعضاء ارسال کرد ولیکن اگر گروه بزرگ باشد این راهکار بسیار پر هزینه است. برخی اوقات می‌توان از روش پخش فراگیر (Broadcasting) بهره گرفت اما روش پخش فراگیر برای رساندن پیامی به ۱۰۰۰ ماشین بر روی شبکه‌ای که یک میلیون گره دارد، بهیچوجه کارآمد نیست زیرا اکثر گیرندگان پیام هیچ تمایلی به دریافت این گونه پیامها ندارند (بدتر از همه آنکه دیگران علاقمند به دریافت چنین پیامهایی باشند ولی مجاز به دریافت آنها نباشند!) بنابراین به راهکاری نیازمندیم که بتوان پیامهایی را برای گروه کاملاً مشخصی ارسال کرد؛ گروهی که اگرچه از لحاظ تعداد، بزرگ به نظر می‌رسد ولی در مقایسه با کلّ شبکه بسیار کوچک است.

ارسال یک پیام به یک چنین گروهی، «**چندپخشی**» (Multicasting) و به الگوریتم مسیریابی آن «**مسیریابی چندپخشی**» (Multicast Routing) گفته می‌شود. در این بخش یکی از روشهای «مسیریابی چندپخشی» را مطالعه خواهیم نمود. برای کسب آگاهی بیشتر به مراجع زیر نگاهی بیندازید:

(Chu et al., 2000; Costa et al., 2001 ; Kasera et al., 2000; Madruga and Garcia-Luna-Aceves, 2001; Zhang and Ryu, 2001).

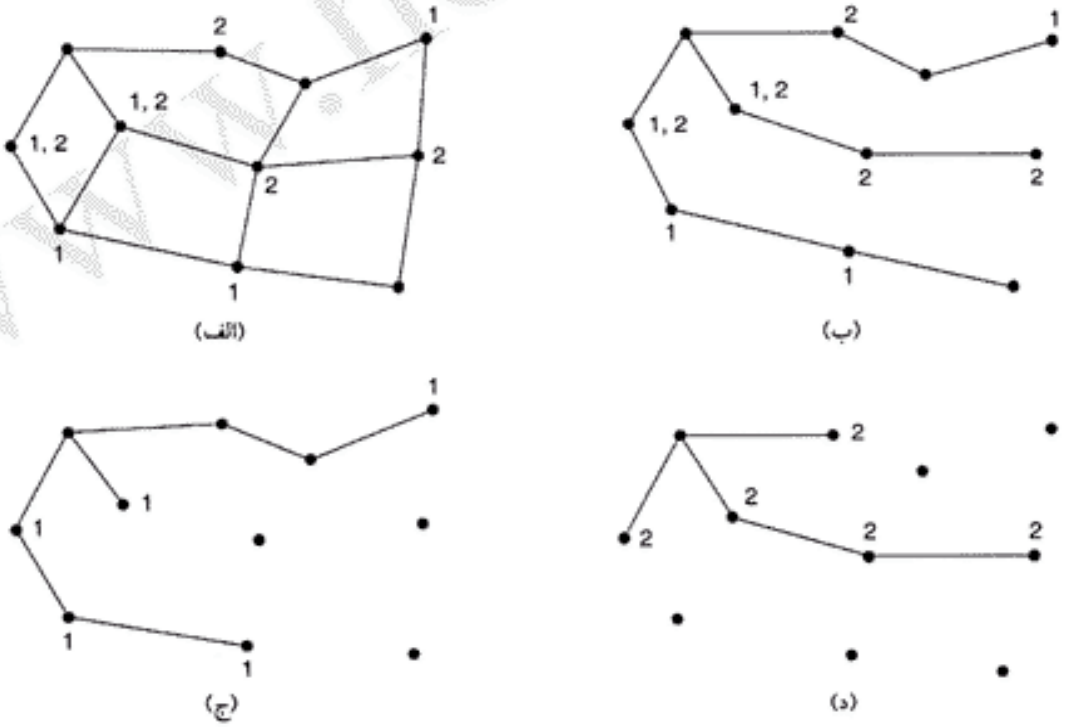
---

۱. Distributed Database System

مسیریابی چندپخشی به مدیریت گروه نیاز دارد؛ یعنی به روشهایی احتیاج است که بتوان گروه ایجاد و حذف کرد و پروسه‌ها اجازه داشته باشند به گروههایی بپیوندند یا از آنها جدا شوند. چگونگی انجام چنین کارهایی به الگوریتم مسیریابی ربطی ندارد. وقتی پروسه‌ای به گروهی می‌پیوندد باید ماشین میزبان خود را از این موضوع آگاه‌کند. آنچه اهمیت دارد آنست که مسیریاب می‌داند هر یک از ماشینهای میزبان او عضو چه گروههایی هستند. ماشینهای میزبان باید هر گونه تغییر عضویت خود در یک گروه را به اطلاع مسیریابهای خود برسانند یا آنکه مسیریابها باید خودشان بطور متناوب در این خصوص از ماشینهای میزبان سؤال کنند. به هر حال، مسیریابها از اینکه چه ماشینی عضو چه گروهی است باخبر می‌شوند. از آن به بعد، مسیریابها به همسایه‌های خود خبر می‌دهند و بدین ترتیب این اطلاعات در کلّ زیرشبکه منتشر خواهد شد.

برای انجام عملیات مسیریابی چندپخشی، هر مسیریاب یک «درخت پوشا» (Spanning Tree) که کلّ مسیرها را در بر می‌گیرد، ایجاد می‌کند. به عنوان مثال در شکل ۵-۱۷-الف، دو گروه ۱ و ۲ را تعریف کرده‌ایم. به نحوی که در این شکل ملاحظه می‌کنید برخی از مسیریابها به ماشینهای میزبانی متصلند که بعضاً عضو یک یا هر دوی این گروهها هستند. [در این شکل، برچسب ۱ یا ۲ بر روی هر مسیریاب، مبیّن آنست که ماشینهای میزبان متصل به آن مسیریاب عضو چه گروهی هستند. -م] در شکل ۵-۱۷-ب ، یک درخت پوشا برای مسیریاب سمت چپ نشان داده شده است.

وقتی پروسه‌ای ، یک بسته چندپخشی (Multicast Packet) را برای گروهی ارسال می‌کند اولین مسیریاب، درخت پوشای خود را بررسی کرده و آن را پیرایش (Prune) می‌کند، یعنی تمام خطوطی را که نهایتاً به ماشینهای عضو این گروه ختم نمی‌شوند، حذف می‌نماید. در این مثال شکل ۵-۱۷-ج درخت پوشا و پیرایش‌شدهٔ گروه ۱ را نشان می‌دهد. به روش مشابه شکل ۵-۱۷-د درخت پوشا و پیرایش‌شدهٔ گروه ۲ را به تصویر کشیده است. بسته‌های چندپخشی فقط از طریق درخت پوشای متناسب با گروه مقصد هدایت می‌شوند.



شکل ۵-۱۷. (الف) ساختار یک شبکه (ب) درخت پوشا برای مسیریاب سمت چپ (ج) درخت چندپخشی برای گروه ۱ (د) درخت چندپخشی برای گروه ۲.

روشهای گوناگونی برای پیرایش درخت پوشا وجود دارد. ساده‌ترین روش، زمانی قابل استفاده است که از روش «مسیریابی حالت لینک» (LS) بهره گرفته شده باشد و هر مسیریاب از توپولوژی کامل شبکه و از جمله عضویت ماشینهای میزبان در گروه‌ها آگاه باشد. در این حالت می‌توان با شروع از انتهای هر مسیر و حرکت به سمت ریشهٔ درخت و حذف تمام مسیریابهایی که در گروه مورد نظر عضو نیستند، درخت پوشا را پیرایش کرد.

در روش «مسیریابی بردار فاصله» (DV) می‌توان از راهکار متفاوتی برای پیرایش درخت پوشا بهره گرفت. الگوریتم اصلی همان روش «هدایت بر روی مسیر معکوس» است (Reverse Path Forwarding)، ولیکن هر گاه یک مسیریاب دارای هیچ ماشینی عضو یک گروه خاص، نبوده و همچنین به هیچ مسیریاب دیگری که علاقمند به دریافت بسته‌های آن گروه است، متصل نشده باشد با ارسال یک بستهٔ خاصّ به نام PRUNE به فرستنده اعلام می‌کند که بسته‌های متعلق بدان گروه خاص را برایش نفرستد. اگر یک مسیریاب از تمام خطوط ورودیش پیغام PRUNE دریافت کند و خودش نیز ماشینی که در آن گروه عضو است نداشته باشد، او نیز پیغام PRUNE را صادر می‌نماید. در چنین حالتی، زیرشبکه به صورت بازگشتی (Recursively) پیرایش می‌شود.[1]

اشکال بالقوهٔ این الگوریتم آن است که در مقیاس شبکه‌های بزرگ، ضعیف عمل می‌کند. فرض کنید که شبکه‌ای دارای n گروه و در هر گروه بطور متوسط m عضو وجود داشته باشد. به ازای هر گروه باید m «درخت پیرایش‌شدهٔ پوشا» ایجاد و ذخیره گردد که جمعاً معادل m×n درخت است. وقتی گروههای زیادی ایجاد شده باشد برای ذخیرهٔ این درختها، به حجم حافظهٔ قابل توجهی نیاز خواهد بود.

در طرحی دیگر، از «درختهای مبتنی بر هسته» (Core-Based Tree) بهره گرفته شده است. (Ballardie et al., 1993) در این طرح به ازای هر گروه فقط یک درخت پوشا محاسبه می‌شود، بگونه‌ای که «**ریشهٔ درخت**» (یا بعبارتی هسته) در نزدیکی مرکز آن گروه قرار می‌گیرد. برای ارسال یک پیام چندپخشی، ماشین میزبان آن را به سوی هسته می‌فرستد و هسته نیز عملیات پخش آن بسته را در گروه، از طریق درخت پوشا انجام می‌دهد. اگرچه این درخت برای تمام مسیریابهای مبداء کاملاً بهینه نیست ولی در عوض کاهش هزینهٔ ذخیره‌سازی m درخت به یک درخت در هر گروه، صرفه جویی بسیار قابل توجهی محسوب می‌شود.

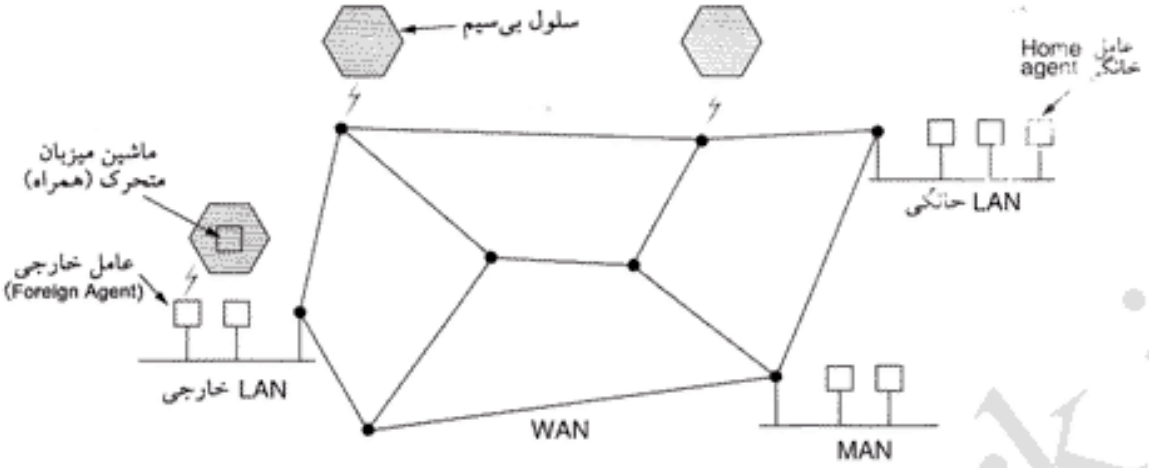
۵-۲-۹ مسیریابی برای ماشینهای متحرک

امروزه میلیونها نفر از مردم کامپیوترهای قابل حمل دارند و عموماً تمایل دارند در هر کجای دنیا که باشند نامه‌های الکترونیکی خود را بخوانند و به سیستم فایل همیشگی خود دسترسی داشته باشند. این ماشینهای متحرک مشکل جدیدی را بوجود می‌آورند: قبل از هدایت یک بسته به سوی ماشین متحرک، در ابتدا شبکه باید آن ماشین را پیدا کند. موضوع پیوستن ماشینهای میزبان متحرک (Mobile Hosts) به شبکه، موضوعی بسیار جدید و نو است ولیکن در این بخش به ارائهٔ برخی از راهکارها در این زمینه خواهیم پرداخت.

مدلی که طراحان شبکه عموماً برای ساختار ارتباطی جهان در نظر می‌گیرند در شکل ۵-۱۸ نشان داده شده است. در این شکل یک WAN با تعدادی مسیریاب و ماشین میزبان دیده می‌شود. به WAN تعدادی LAN ،

---

۱. بعبارت ساده‌تر در ابتدای کار هر بستهٔ چندپخشی بروش «هدایت بر روی مسیر معکوس» که در بخش قبل بررسی شد بر روی خطوط خروجی مسیریابها هدایت می‌شوند و تا اینجای کار هیچ تفاوتی با پخش فراگیر ندارد. به محض پخش اولین بسته بصورت فراگیر، تمام مسیریابهایی که انتظار دریافت چنین بسته‌ایی را نداشته‌اند با ارسال بستهٔ کنترلی Prune از همسایهٔ خود که این بسته را برایشان فرستاده خواهش می‌کنند این کار را تکرار نکند. وقتی یک مسیریاب خودش هیچ عضوی در این گروه ندارد و تمام همسایه‌هایش نیز با ارسال Prune از او خواسته‌اند که بسته‌های آن گروه را برایشان نفرستد، لزومی به دریافت بسته‌های آن گروه نمی‌بیند و او هم پیغام Prune به همسایهٔ قبلی خود می‌فرستد. بدین ترتیب از آخر مسیر به اول درخت پوشای فرضی اصلاح می‌شود. ـم



شکل ۵-۱۸. یک شبکهٔ WAN که شبکه‌های LAN، MAN و سلولهای بی‌سیم بدان متصل شده‌اند.

MAN و شبکهٔ بی‌سیم (از نوعی که در فصل ۲ مطالعه کردیم)، متصل شده است.

به ماشینهای میزبان که حرکت نمی‌کنند ماشینهای ثابت (Stationary) گفته می‌شود. این ماشینها از طریق سیمهای مسی یا فیبرهای نوری به شبکه متصل شده‌اند. برعکس، دو نوع دیگر ماشین میزبان، قابل رؤیت است: «ماشینهای میزبان مهاجر» (Migratory Hosts) ماشینهای ثابتی هستند که گاه‌به‌گاه از یک سایت ثابت به سایتی دیگر نقل مکان می‌کنند ولی فقط زمانی در شبکه قرار می‌گیرند که اتصال فیزیکی آنها برقرار شود. «ماشینهای میزبان سیّار» (Roaming Hosts) در حال حرکت هم کار می‌کنند و باید ارتباط خود را در حین حرکت حفظ کنند. ما این دو رده از ماشینهای میزبان را «ماشینهای میزبان متحرک» (Mobile Hosts) می‌نامیم: یعنی تمام ماشینها دور از محل استقرار همیشگی خود هستند و می‌خواهند به شبکه متصل شوند.

فرض بر آن است که تمام ماشینها دارای یک «**محل استقرار دائمی**» (Home Location) هستند که هیچگاه تغییر نمی‌کند. همچنین هر ماشین میزبان دارای آدرس ثابتی در محل استقرار دائم خود است که موقعیت این محل را مشخص می‌کند (شبیه به یک شماره تلفن در آمریکا مثل ۲۱۲-۵۵۵۱۲۱۲-۱، با کد کشوری ۱ و کد منهتن، ۲۱۲). هدف نهایی فرآیند مسیریابی در سیستمی با ماشینهای متحرک، آنست که بتوان بسته‌ها را به کمک آدرس دائم آنها به ماشینها رساند و ماشینهای میزبان (در هر کجا که باشند) بتوانند بسته‌های خود را تحویل بگیرند. ظریفترین بخش قضیه آنست که ابتدا باید این ماشینها را پیدا کرد.

در مدل شکل ۵-۱۸ کل دنیا از نظر جغرافیایی به چندین بخش کوچک تقسیم شده است. اجازه بدهید آنها را «ناحیه» بنامیم؛ «ناحیه»، خود یک شبکهٔ LAN یا یک «سلول بی‌سیم» (Wireless Cell) است. در هر ناحیه یک یا چند «**عامل خارجی**» (Foreign Agent) وجود دارد که در عمل پروسه‌هایی هستند که فهرست ماشینهای متحرک و میهمان آن ناحیه را پیگیری و مدیریت می‌کنند. بعلاوه هر ناحیه یک «**عامل خانگی**» (Home Agent) دارد که فهرست ماشینهایی را نگهداری می‌کند که محل استقرار دائمی آنها همین ناحیه است ولی فعلاً در نواحی دیگر به سر می‌برند.

وقتی ماشین جدیدی وارد یک ناحیه می‌شود (از طریق اتصال فیزیکی یا شبکه مثلاً با اتصال کابل آن به LAN و یا با پرسه زدن در درون سلول بی‌سیم)، باید خودش را در «عامل خارجی» ثبت‌نام نماید. روال ثبت نام عموماً به شکل زیر انجام می‌شود:

۱. هر «**عامل خارجی**» بطور متناوب یک بستهٔ فراگیر پخش کرده و حضور خود و آدرسش را به اطلاع همه

می‌رساند. یک ماشین متحرک تازه وارد باید منتظر چنین پیامهایی بماند ولیکن اگر در اسرع وقت، چنین پیامی نرسد ماشین متحرک می‌تواند بسته‌ای فراگیر برای همه بفرستد با این مضمون که: «آیا هیچ عامل خارجی در اینجا هست؟!»

۲. ماشین متحرک در عامل خارجی ثبت‌نام می‌کند، آدرس محل استقرار دائم خود، آدرس فیزیکی (یعنی آدرس لایهٔ پیوند داده‌ها) فعلی خود و برخی اطلاعات امنیتی (جهت احراز هویت) را ارائه می‌دهد.

۳. عامل خارجی با عامل خانگی آن ماشین متحرک تماس گرفته و می‌گوید: یکی از ماشینهای شما در ناحیهٔ ما به سر می‌برد! در این پیام که از سوی «عامل خارجی» به سوی «عامل خانگی» ارسال می‌شود آدرس شبکهٔ عامل خارجی [آدرس خودش] مشخص می‌شود. این پیام همچنین شامل برخی اطلاعات امنیتی است تا عامل خانگی را متقاعد کند که ماشین متحرک واقعاً در ناحیهٔ اوست.

۴. عامل خانگی اطلاعات امنیتی ارائه شده را که محتوی «مهر زمان» (Timestamp) نیز هست بررسی می‌کند تا برایش ثابت شود که این پیام در خلال همین چند ثانیهٔ اخیر تولید شده است. اگر متقاعد شود به «عامل خارجی» اجازهٔ ادامهٔ کار را می‌دهد.

۵. وقتی «عامل خارجی» پیام تأیید «عامل داخلی» را دریافت کند یک درایه (Entry) در جدول خود ایجاد کرده و به ماشین متحرک اعلام می‌کند که ثبت‌نام شده است.

آرمانی آن است که وقتی یک ماشین میزبان ناحیه‌ای را ترک می‌کند به همان ترتیب اعلام کند تا عضویت آن لغو شود، ولی بسیاری از کاربران به محض آنکه کارشان تمام می‌شود بلافاصله کامپیوتر خود را خاموش می‌کنند و مهلتی برای اعلام ترک ناحیه، باقی نمی‌ماند!

وقتی بسته‌ای برای یک ماشین متحرک ارسال می‌شود ابتدا آن بسته به سوی شبکهٔ LAN خانگی او [یعنی محل استقرار دائم او] مسیریابی و هدایت می‌شود زیرا در حقیقت بسته به آدرس دائم او ارسال شده است. (مرحلهٔ ۱ از شکل ۵-۱۹) در این شکل فرستنده‌ای در شهری از سیاتل آمریکا می‌خواهد بسته‌ای را برای یک ماشین میزبان متحرک در نیویورک بفرستد. بسته ابتدا به آدرس شبکهٔ LAN خانگی او در نیویورک ارسال می‌شود و در آنجا توسط «عامل خانگی» تحویل گرفته می‌شود. عامل خانگی در حافظهٔ خود محل استقرار جدید (و موقت) ماشین متحرک را جستجو کرده و آدرس عامل خارجی که آن ماشین را در پوشش خود گرفته (مثلاً در لوس آنجلس) پیدا می‌کند.

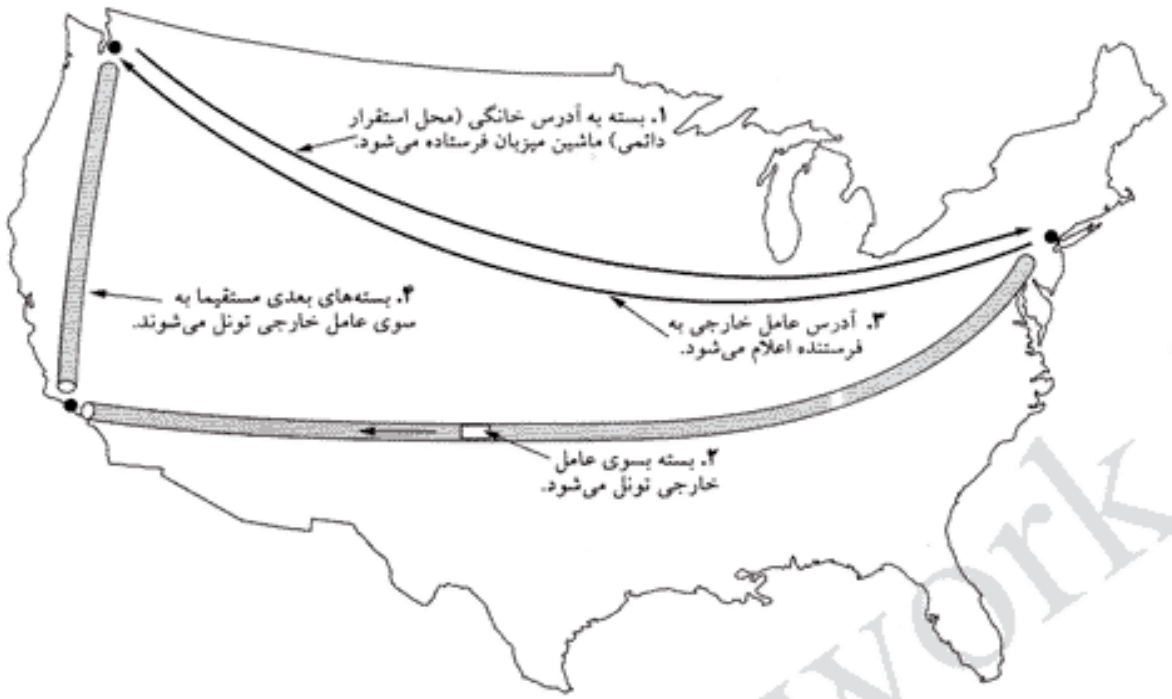
در این لحظه، عامل خانگی دو کار انجام می‌دهد: اول آن که این بسته را در درون فیلد داده از یک بستهٔ بیرونی دیگر جاسازی کرده و آن را برای عامل خارجی می‌فرستد. (مرحلهٔ ۲ از شکل ۵-۱۹) به این مکانیزم «ایجاد تونل» (Tunneling) گفته می‌شود و بعداً به تفصیل در مورد آن صحبت خواهیم کرد. پس از دریافت بستهٔ جاسازی‌شده توسط عامل خارجی، بستهٔ اصلی از درون فیلد دادهٔ آن جدا شده و به ماشین متحرک تحویل می‌شود.

دوم آنکه عامل خانگی به فرستندهٔ بسته اعلام می‌کند که از این به بعد بسته‌های خود را با جاسازی درون یک بستهٔ دیگر (که به صراحت آدرس «عامل خارجی» در آن درج شده) مستقیماً به عامل خارجی بفرستد. (مرحلهٔ ۳) از آن به بعد بدون در نظر گرفتن موقعیت دائمی ماشین متحرک، بسته‌ها مستقیماً به سوی عامل خارجی هدایت شده و تحویل ماشین متحرک می‌شود.[۱]

روشهایی که تاکنون معرفی شده‌اند از چندین جهت با هم تفاوت دارند: اولین مورد آن که چه مقدار از وظایف

---

۱. در فرآیند تونل یک بستهٔ کامل در درون یک بستهٔ دیگر جاسازی می‌شود: آدرس بستهٔ بیرونی مربوط به «عامل خارجی» است و آدرس بستهٔ درونی هویت صاحب اصلی بسته یعنی ماشین متحرک را مشخص می‌کند. -م



شکل ۵-۱۹. مسیریابی بسته‌ها بسوی ماشینهای متحرک.

این پروتکل توسط مسیریابها و چه مقدار توسط ماشینهای میزبان انجام می‌شود و آن بخش از وظایف که در ماشین میزبان باید انجام شود در چه لایه‌ای تعریف شده است. دوم آن که در برخی از روشها، مسیریابهای واقع بر روی مسیر قادرند «نگاشتهای آدرس» [یعنی تغییر آدرس ماشین متحرک به آدرس جدید] را ثبت کرده و بدون هیچ دخالت بیرونی در میانهٔ راه جلوی ترافیک ارسالی به آدرس قبلی را گرفته و آن را به سمت آدرس جدید تغییر مسیر بدهند.

سوم آن که در برخی دیگر از این روشها به هر ماشین متحرک که میهمان شبکه‌ای جدید شده یک آدرس موقت ولیکن منحصر به فرد داده می‌شود در حالی که در برخی دیگر این آدرس موقت مربوط به آن عامل خارجی می‌باشد که هدایت ترافیک تمام میهمانان خود را بر عهده گرفته است.

تفاوت چهارم این روشها آنست که وقتی بسته‌ها برای رسیدن به یک ماشین دیگر آدرس‌دهی شده‌اند چگونه تحویل ماشین دیگری می‌شوند. [بدین معنا که بسته‌هایی که به سوی آدرس قبلی یک ماشین روانه شده‌اند به چه نحو برای هدایت به جایی دیگر تغییر آدرس می‌دهند. -م] گزینهٔ اول آنست که در هر بسته، آدرس مقصد عوض شود و بستهٔ تغییر یافته از نو ارسال شود. گزینهٔ دوم آنست که کل بسته به همراه آدرس دائم (خانگی) و با کلیهٔ مشخصات [به صورت دست نخورده]، در درون فیلد داده از یک بستهٔ دیگر جاسازی (کپسوله) شود و بستهٔ جدید به آدرس موقت ماشین ارسال گردد.

تفاوت آخر آنکه روشهای مختلف از دیدگاه تمهیدات امنیتی با یکدیگر فرق می‌کنند. عموماً وقتی یک ماشین میزبان یا مسیریاب، پیامی با مضمون این مثال دریافت می‌کند: «لطفاً از هم اکنون به بعد تمام نامه‌های استفانی را برای من بفرست!» دو سؤال مطرح می‌شود: این پیام واقعاً متعلق به کیست (با چه کسی صحبت می‌شود) و آیا چنین کاری اصولاً به صلاح است؛ در مراجع زیر چندین پروتکل در خصوص ماشینهای متحرک تشریح و مقایسه شده‌اند:

(Hac and Guo, 2000; Perkins, 1998a; Snoeren and Balakrishnan, 2000; Solomon, 1998; Wang and Chen, 2001)

۱۰-۲-۵ مسیریابی در شبکه‌های ویژه (Routing in Ad Hoc Networks)

تاکنون روشهای مسیریابی در محیطهایی را که ماشینهای میزبان متحرکند ولی مسیریابها ثابت هستند، بررسی کرده‌ایم. یک حالت استثنایی آن است که مسیریابها نیز خودشان متحرک باشند! از چنین محیطهایی می‌توان به موارد زیر اشاره کرد:

۱. وسایل نقلیهٔ نظامی در صحنهٔ نبرد بدون دسترسی به هیچگونه زیرساخت ارتباطی

۲. ناوگان دریایی بر روی دریا

۳. نیروی امداد در شرایط اضطراری مثل زلزله که کلیهٔ زیرساختها را نابود کرده است

۴. گردهمایی افراد با کامپیوتر کیفی، در ناحیه‌ای بدون شبکهٔ بی‌سیم 802.11

در تمام این موارد (و نظایر آن) هر گره شامل یک مسیریاب و یک ماشین میزبان است که اغلب هر دوی آنها بر روی یک ماشین قرار می‌گیرند. شبکه‌ای از این گره‌ها که در کنار هم قرار می‌گیرند اصطلاحاً «**شبکه‌های ویژه**» یا MANET[1] نامیده می‌شود. اجازه بدهید مختصراً این شبکه‌ها را بررسی نماییم. برای آگاهی بیشتر می‌توان به (Perkins, 2001) مراجعه کرد.

آنچه که «شبکه‌های ویژه» را از «شبکه‌های سیمی» متمایز می‌کند آن است که تمام قواعد طبیعی در خصوص توپولوژی ثابت، همسایه‌های شناخته شدهٔ ثابت، تناظر دائم بین موقعیت فیزیکی و آدرس IP ماشینها و مواردی از این قبیل در خصوص «شبکه‌های ویژه» صادق نیست. میسریابها در رفت و آمد هستند و ممکن است در هر لحظه از محل جدیدی سر در آورند! در یک شبکهٔ سیمی هر گاه یک مسیریاب، مسیری معتبر به برخی از نقاط مقصد در شبکه داشته باشد آن مسیر تا ابد معتبر خواهد بود (مگر آن که در جایی از سیستم خرابی بوجود بیاید). در یک شبکهٔ ویژه، توپولوژی شبکه به طور دائم در تغییر است لذا اعتبار مسیرها و بهینگی آنها به ناگاه و بی‌هیچ هشدار قبلی تغییر می‌کند. آشکار است که با چنین وضعیتی، مسیریابی در شبکه‌های ویژه کاملاً متفاوت از شبکه‌های ثابت می‌باشد.

گونه‌های متعددی از الگوریتمهای مسیریابی برای شبکه‌های ویژه پیشنهاد شده است. یکی از جالبترین آنها الگوریتم مسیریابی AODV[2] (Perkins and Royer, 1999) است. این الگوریتم گونه‌ای از «الگوریتم بردار فاصلهٔ بلمن-فورد» محسوب می‌شود که برای کار در محیطهای متحرک تطبیق داده شده و در آن پهنای باند محدود و عمر کم باطری ماشینها در این محیط [در محاسبات مربوط به مسیرهای بهینه] در نظر گرفته شده است. یکی دیگر از ویژگیهای نامتعارف این روش آن است که الگوریتم «برحسب تقاضا» (On-Demand) عمل می‌کند بدین معنا که مسیر رسیدن به برخی از نقاط مقصد، فقط وقتی تعیین می‌شود که کسی بخواهد بسته‌ای را بدان مقصد بفرستد. اجازه بدهید ببینیم قضیه از چه قرار است.

کشف مسیر

یک شبکهٔ ویژه را در هر لحظه از زمان می‌توان با گرافی از گره‌ها توصیف کرد. (گره‌ها عبارتند از مسیریابها + ماشینهای میزبان) اگر دو ماشین بتوانند از طریق سیستم رادیویی خود مستقیماً با یکدیگر ارتباط برقرار کنند می‌گوییم این دو گره (Node) بهم متصل هستند. (یعنی در گراف شبکه، بین این دو گره یک کمان وجود دارد). از آنجایی که امکان دارد یکی از این دو ماشین، فرستندهٔ پرقدرت‌تری نسبت به دیگری داشته باشد لذا این امکان وجود دارد که ارتباط A به B برقرار باشد ولی ارتباط B با A میسّر نباشد؛ ولیکن برای سادگی فرض را بر آن می‌گذاریم که تمام ارتباطات، دو طرفه و متقارن هستند. همچنین باید بدین نکته اشاره کرد که هر گاه دو گره در برد

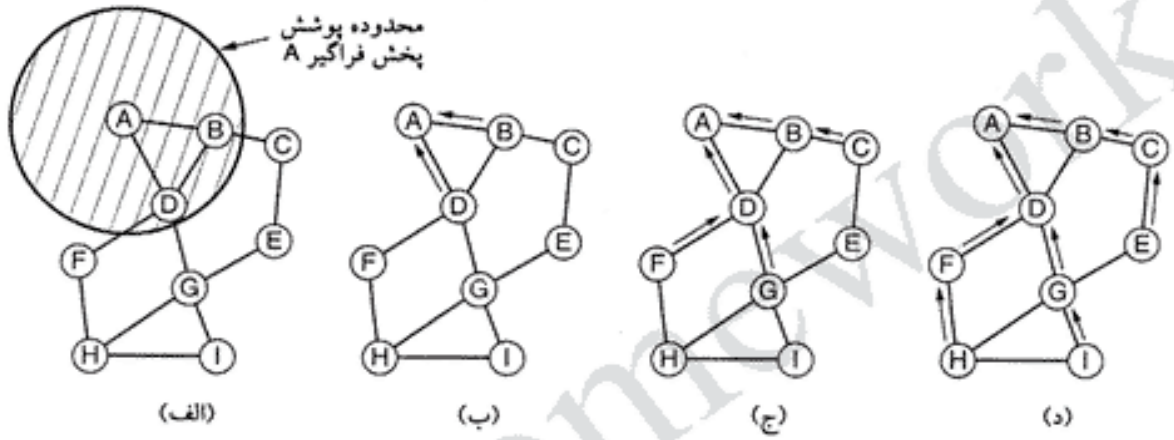
---

۱. Mobile Ad Hoc Network

۲. Ad Hoc On-Demand Distance Vector

رادیویی یکدیگر باشند تضمینی وجود ندارد که ارتباط آنها برقرار باشد؛ ممکن است بین آنها ساختمان، تپه یا موانع دیگری وجود داشته باشد که جلوی ارتباط آنها را بگیرد. [هر چند در برد یکدیگر واقعند.]

برای توصیف الگوریتم، شبکهٔ ویژهٔ شکل ۵-۲۰ را در نظر بگیرید که در آن یک پروسه در گره A می‌خواهد بسته‌ای را برای گره I بفرستد. در الگوریتم AODV، هر گره دارای جدولی است که کلید این جدول، آدرس مقصد است[1] و هر یک از رکوردهای این جدول، اطلاعاتی در خصوص مقصد و آنکه برای رساندن بسته‌ای به آن مقصد باید بسته را به کدامیک از همسایه‌های آن فرستاد، در خود نگاهداری می‌کنند. فرض کنید که A در جدول خود جستجو کرده و هیچ درایه‌ای متناظر با I در آن نمی‌یابد. حال بایستی مسیری به I کشف کند. همین ویژگی که مسیرها فقط در هنگام لزوم کشف می‌شوند به الگوریتم، ویژگی On-Demand یعنی «برحسب تقاضا» داده است.



شکل ۵-۲۰. (الف) محدودهٔ پوشش پخش فراگیر A (ب) پس از آنکه B و D پخش فراگیر A را دریافت کردند. (ج) پس از آنکه C و F و G پخش فراگیر A را دریافت کردند. (د) پس از آنکه E و H و I پخش فراگیر A را دریافت کردند. گره‌های سایه‌دار دریافت کنندگان جدید هر مرحله محسوب می‌شوند. فلشها مسیر معکوس (مسیر برگشت) را مشخص می‌کنند.

برای پیدا کردن موقعیت I، گره A یک بستهٔ خاص به نام ROUTE REQUEST (تقاضای مسیر) ساخته و آن را به صورت فراگیر منتشر می‌کند. به گونه‌ای که در شکل ۵-۲۰-الف مشهود است، این بسته به B و D می‌رسد. در حقیقت دلیل آنکه در این گراف، B و D به A متصل شده‌اند آنست که قادرند سیگنال مخابراتی A را دریافت کنند. به عنوان مثال بین گره F و A در گراف هیچ کمانی ترسیم نشده است چراکه F نمی‌تواند سیگنال رادیویی A را دریافت کند. بنابراین ارتباط مستقیمی بین F و A وجود ندارد.

قالب بستهٔ ROUTE REQUEST در شکل ۵-۲۱ نشان داده شده است. این بسته شامل «آدرس مبداء» و «آدرس مقصد» است و مشخص می‌کند که چه کسی در جستجوی چه کسی است. (عموماً از آدرسهای IP آنها استفاده می‌شود) این بسته همچنین حاوی یک «**شناسهٔ تقاضا**» (Request ID) است. این شناسه در حقیقت یک شمارندهٔ محلی است که در هر گره وجود دارد و هرگاه که یک بستهٔ ROUTE REQEST منتشر گردید یک واحد به آن اضافه می‌شود. ترکیب «آدرس مبداء» و فیلد «شناسهٔ تقاضا»، هویت بستهٔ ROUTE REQUEST را به صورت یکتا و منحصر بفرد تعیین کرده و بدین ترتیب گره‌ها قادرند بسته‌هایی که احتمالاً به صورت تکراری دریافت می‌شوند را تشخیص داده و حذف کنند.

هر گره به غیر از «شمارندهٔ شناسهٔ تقاضا» یک شمارندهٔ دیگر هم دارد که هرگاه بستهٔ ROUTE REQUEST

---

۱. یعنی جستجوی رکوردها در این جدول براساس آدرس مقصد انجام می‌شود. -م

Source address	Request ID	Destination address	Source sequence #	Dest. sequence #	Hop count

شکل ۵-۲۱. قالب بستهٔ ROUTE REQUEST.

ارسال شود (یا پاسخی برای بستهٔ ROUTE REQUEST دیگران دریافت گردد) یک واحد بدان اضافه می‌گردد. عملکرد این شمارنده تا حدودی شبیه به یک ساعت است و کاربرد آن تشخیص مسیر جدید از مسیر قدیمی است. [یعنی وقتی دو بستهٔ یکسان در مورد مشخصات یک مسیر دریافت می‌شود این شمارنده مشخص می‌کند که کدام جدیدتر از دیگری است] فیلد چهارم از شکل ۵-۲۱ مقدار همین شمارنده را برای گره مبداء (که در این مثال A است) مشخص می‌کند. [در این مثال A در جستجوی I است لذا در بستهٔ ROUTE REQUEST شمارهٔ ترتیب تقاضای خود را درج کرده است.] فیلد پنجم مشخص کنندهٔ آخرین و جدیدترین شمارهٔ بسته‌ای است که A در مورد I دریافت کرده است. (و صفر است اگر تاکنون در مورد I چیزی دریافت نکرده باشد.) آخرین فیلد یعنی «**شمارندهٔ گام**» (Hop Counter) مشخص می‌کند که بسته تاکنون چند گام را طی کرده است. مقدار اولیهٔ این فیلد صفر است و به ازای عبور از هر گره یک واحد به آن اضافه می‌شود.

وقتی بستهٔ ROUTE REQUEST به یک گره می‌رسد (در این مثال به B و D) طبق مراحل زیر پردازش می‌شود:

۱. ابتدا جفت مشخصهٔ (آدرس مبداء، شناسهٔ تقاضا) در یک جدول محلی (که سوابق دریافت چنین بسته‌هایی را نگاهداری می‌کند) جستجو می‌شود تا مشخص گردد که آیا این بسته قبلاً نیز دریافت و پردازش شده است؟ اگر تکراری بود بسته حذف شده و پردازش آن در همین جا خاتمه می‌یابد. اگر تکراری نبود این زوج مشخصهٔ بسته را در جدول سوابق وارد می‌کند تا در آینده نیز بسته‌ای مشابه با این بسته پردازش نشود. سپس فرآیند پردازش ادامه می‌یابد.

۲. سپس گیرنده در جدول مسیریابی خود، آدرس مقصد را جستجو می‌کند. اگر یک مسیر جدید و «تازه» به این مقصد پیدا شود یک بستهٔ ROUTE RERLY برای مبداء برگردانده می‌شود تا مبداء نیز از چگونگی رسیدن به مقصد مورد نظر آگاه شود. (اغلب، پاسخ بدین نحو است: از طریق من عمل کن!) «تازه» بودن مسیر بدین معنی است که فیلد «شمارهٔ ترتیب مقصد» ذخیره شده در جدول مسیریابی باید بزرگتر یا مساوی همین فیلد در بستهٔ ROUTE REQUEST باشد. اگر کمتر بود بدین معنی تلقی می‌شود که مشخصات ذخیره شده در حافظه، قدیمی‌تر از مسیری است که خود مبداء برای رسیدن به مقصد در اختیار دارد[1]، در این صورت مرحلهٔ سوم به اجرا در می‌آید.

۳. از آنجایی که گیرنده هیچ مسیر جدیدی بدان مقصد نمی‌شناسد، به مقدار فیلد «شمارندهٔ گام» (Hop Count) یک واحد اضافه کرده و بستهٔ ROUTE REQUEST را مجدداً پیرامون خود منتشر می‌نماید. البته داده‌های درون بسته را استخراج کرده و آن را به عنوان یک درایهٔ جدید در جدول «**مسیرهای معکوس**» ذخیره می‌کند.[2] این اطلاعات بدان جهت مفید است که می‌توان مسیرهای معکوس ایجاد کرد و از

---

۱. کاربرد فیلد شمارهٔ ترتیب مقصد (Destination Sequence #) را در ذهن خود اینگونه فرض کنید که مثلاً A به همسایه‌های خود اعلام می‌کند که من خودم اطلاعاتی در خصوص مسیر رسیدن به I دارم که شمارهٔ ترتیب آن (مثلاً) Destination Sequence #=۱۲۳۴ است. اگر شما اطلاعات جدیدتری دارید لطفاً برای من بفرستید! -م

۲. یعنی اگرچه مبداء به دنبال یافتن یک مقصد خاص است ولیکن در بستهٔ تقاضا حداقل خود را معرفی کرده است فلذا مسیریابهای گیرندهٔ این تقاضا، لااقل می‌توانند از حضور این مبداء و مسیر رسیدن به آن کسب آگاهی کنند؛ به چنین مسیری «مسیر معکوس» گفته می‌شود. -م

طریق آن، در آینده پاسخ این تقاضا را به مبداء برگرداند. فلشهایی که بر روی شکل ۵-۲۰ دیده می‌شوند چگونگی ساخته شدن مسیرهای معکوس را نشان می‌دهند. به محض ایجاد یک مسیر معکوس و جدید، یک زمان‌سنج (تایمر) برای آن تنظیم می‌شود. اگر مهلت این زمان‌سنج منقضی شود و پاسخی به بستهٔ ROUTE REQUEST برنگردد، مسیر ایجاد شده حذف خواهد شد.

هیچیک از گره‌های B و D نمی‌دانند که I کجاست لذا این دو نیز ضمن ساختن مسیر معکوس جهت بازگشت پاسخ به A، با تغییر فیلد Hop Count به ۱، آنرا مجدداً منتشر می‌کنند.(به شکل ۵-۲۰ دقت کنید.) انتشار مجدّد بسته توسط B، به C و D می‌رسد. C نیز درایه‌ای در جدول مسیرهای معکوس خود ایجاد و آن را از نو منتشر می‌کند. در مقابل، D آنرا به عنوان بسته‌ای تکراری حذف می‌کند. [چون قبلاً از طریق A دریافت کرده است.] به همین طریق بستهٔ منتشره توسط D در B تکراری تشخیص داده شده و حذف می‌گردد. ولیکن به گونه‌ای که در شکل ۵-۲۰-ج دیده می‌شود بستهٔ منتشره توسط D در F و G پذیرفته و ذخیره می‌شود.

پس از آن که E و H و I نیز بستهٔ منتشر شده را دریافت کردند، عاقبت پیغام ROUTE REQUEST به مقصد خود می‌رسد (یعنی به خود I یا به گره‌ای که از موقعیت I خبر دقیق دارد می‌رسد). (شکل ۵-۲۰-د را ببینید.) اگرچه ما کلّ فرآیند انتشار را در سه مرحلهٔ جدا نشان داده‌ایم ولیکن انتشار بسته‌ها از گره‌های مختلف به صورت هماهنگ و همزمان انجام نمی‌شود.

گره I در پاسخ به تقاضای ورودی، یک بستهٔ ROUTE REPLY مطابق با شکل ۵-۲۲ ایجاد می‌کند. فیلدهای «**آدرس مبداء**»، «**آدرس مقصد**» مستقیماً از بستهٔ تقاضا استخراج و در بستهٔ پاسخ کپی می‌شوند ولیکن «**شمارهٔ ترتیب مقصد**» (Destination Sequence Number) برگرفته از مقدار شمارنده‌ای است که درون حافظهٔ I قرار دارد. فیلد «شمارندهٔ گام» نیز به صفر تنظیم می‌شود. فیلد «طول عمر» (Lifetime) مشخص می‌کند که مشخصات مسیر اعلام شده تا چه زمانی معتبر است. این بسته صرفاً برای گره‌ای ارسال می‌شود که بستهٔ تقاضا (یعنی ROUTE REQUEST) از طریق او دریافت شده است (در این مثال گره G). این بسته، مسیر معکوس خود به D و نهایتاً به A را طی می‌کند. در هر گره به مقدار فیلد شمارندهٔ پیام یک واحد اضافه می‌شود تا هر گره که آن را می‌بیند بفهمد که فاصله‌اش تا گره مقصد (در این مثال I) چقدر است.

Source address	Destination address	Destination sequence #	Hop count	Lifetime

شکل ۵-۲۲. قالب بستهٔ ROUTE REPLY.

هر گره میانی این بسته را در راه بازگشت، بررسی می‌کند. اگر یکی از سه شرط زیر برقرار باشد، اطلاعاتی در خصوص مسیر رسیدن به I، در جدول مسیریابی هر گره میانی ذخیره خواهد شد:

۱. اگر هیچ مسیر شناخته شده‌ای به I نداشته باشد.
۲. اگر شمارهٔ ترتیب I [یعنی شماره‌ای که I در فیلد Destination Seq.No. گذاشته است] بزرگتر از شماره‌ای باشد که در جدول مسیریابی درج شده است.
۳. اگر شماره ترتیب یکسان باشد ولی مسیر جدید کوتاهتر باشد. [کوتاه بودن از فیلد شمارندهٔ گام مشخص می‌شود.]

در این روش تمام گره‌هایی که بر روی مسیر معکوس قرار دارند، مجاناً از مسیر رسیدن به I آگاه می‌شوند (یکی از محاسن جانبی کشف مسیر توسط A، آگاهی گره‌های میانی است). گره‌هایی که بستهٔ ROUTE REQUEST را

در مسیر رفت دریافت کرده‌اند ولیکن در مسیر معکوس نیستند (در این مثال B و C و E و F و H) پس از منقضی شدن زمان‌سنج (تایمر)، مسیر معکوس به A را حذف می‌کنند.

در شبکه‌های عظیم، این الگوریتم بسته‌های فراگیر بسیار زیادی را تولید می‌کند، حتی اگر گره مقصد در نزدیکی مبداء باشد. برای کاهش تولید بسته‌ها در فرآیند انتشار، می‌توان بدین نحو عمل کرد: فیلد TTL (Time To Live) در بستهٔ IP، توسط فرستندهٔ آن به مقداری نزدیک به «قطر»[1] شبکه تنظیم شده و به ازای عبور از هر گره، یک واحد از آن کسر گردد؛ هر گاه به صفر رسید، بسته به جای انتشار مجدد حذف شود.

فرآیند کشف مسیر را می‌توان بدین نحو اصلاح کرد: برای یافتن موقعیت مقصد، فرستنده، بستهٔ ROUTE REQUEST را با تنظیم فیلد TTL به ۱ ارسال می‌کند. اگر در یک مدت زمان معقول پاسخی برنگشت بستهٔ بعدی را با TTL معادل ۲ ارسال می‌نماید. همین روال با مقادیر ۳ و ۴ و ۵ و... ادامه می‌یابد تا بالاخره پاسخی برگردد. در این روش، جستجو ابتدا به صورت محلی و در پیرامون گره شروع شده و در هر مرحله محدودهٔ جستجو گسترده‌تر می‌شود.

نگهداری مسیر (Route Maintenance)

از آنجایی که گره‌ها می‌توانند جابجا شده یا کلاً خاموش شوند لذا توپولوژی شبکه گاه‌به‌گاه تغییر می‌کند. به عنوان مثال در شکل ۵-۲۰ اگر G به ناگاه خاموش شود، A نخواهد فهمید مسیری که برای رسیدن به I در اختیار داشته (یعنی مسیر ADGI) دیگر برقرار نیست. این الگوریتم باید بتواند به نحوی این مسئله را حل و فصل کند.

هر گره در شبکه بطور متناوب «پیغام سلام» (Hello Message) منتشر می‌کند. انتظار می‌رود که همسایه‌ها به این پیام پاسخ بدهند. اگر پاسخی باز نگشت، منتشرکنندهٔ پیام آگاه می‌شود که همسایهٔ او از بُردش خارج شده و دیگر ارتباط آنها برقرار نیست. به روش مشابه اگر بسته‌ای معمولی برای همسایهٔ خود بفرستد و پاسخی نگیرد متوجه می‌شود که آن همسایه در دسترس نیست.

از این اطلاعات می‌توان برای پاک کردن مسیرهایی که دیگر کار نمی‌کنند بهره گرفت. هر گره مثل N برای یکایک گره‌های مقصد که همسایه‌هایش در خلال ΔT ثانیهٔ گذشته بسته‌ای را از طریق او بدان مقصد ارسال کرده‌اند، فهرستی را نگه می‌دارد. این فهرستها اصطلاحاً «**همسایه‌های فعال** N» نامیده می‌شوند. بدین منظور گره N دارای جدولی است که کلید آن، آدرس مقصد گره‌های شبکه است؛ همچنین در این فهرست، گره بعدی برای رسیدن بدان مقصد، تعداد گام برای رسیدن به آن مقصد، «آخرین شمارهٔ ترتیب» و «فهرست همسایه‌های فعال» درج شده است. به عنوان نمونه جدول مسیریابی گره D برای توپولوژی مثال قبلی، چیزی شبیه به شکل ۵-۲۳-الف است.

وقتی یکی از همسایه‌های N از دسترس خارج می‌شود، گره N جدول مسیریابی خود را بررسی می‌کند تا ببیند کدامیک از گره‌های شبکه در مسیرهای خود از گره حذف شده استفاده می‌کرده‌اند. این موضوع به همسایه‌های فعال اطلاع داده می‌شود که: تمام مسیرهای آنها از طریق N نامعتبر است (چرا که N بسته‌های آنها از طریق گره حذف شده ارسال می‌شده است)؛ همسایه‌های فعال نیز به همسایه‌های فعال خود خبر می‌دهند و مکرراً این کار انجام می‌شود تا تمام مسیرهایی که وابسته به گره تازه از دست رفته بوده‌اند از کلّ جداول مسیریابی حذف شوند.

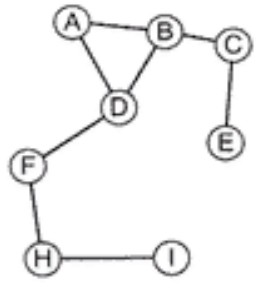
به عنوان مثالی از روش «نگهداری مسیر»، مثال قبلیمان را مدنظر قرار بدهید ولیکن فرض کنید که G به ناگاه خاموش شده است. توپولوژی تغییریافتهٔ شبکه در شکل ۵-۲۳-ب نشان داده شده است؛ وقتی D متوجه می‌شود که G از میان رفته است به جدول مسیریابی خود مراجعه می‌کند و می‌بیند که G بر روی مسیری قرار داشته که به E،

---

۱. قطر شبکه به معنای طول بزرگترین مسیر در شبکه است. -م

Dest.	Next hop	Distance	Active neighbors	Other fields
A	A	1	F, G	
B	B	1	F, G	
C	B	2	F	
E	G	2		
F	F	1	A, B	
G	G	1	A, B	
H	F	2	A, B	
I	G	2	A, B	

(الف)



(ب)

شکل ۵-۲۳. (الف) جدول مسیریابی D قبل از آنکه G از کار بیفتد. (ب) گراف پس از حذف G از شبکه.

G و I ختم می‌شده است. اجتماع مجموعهٔ همسایه‌های فعال این سه گره مقصد عبارت است از {B و A}. به عبارت دیگر A و B در برخی از مسیرهای خود به G متکی بوده‌اند لذا باید به این مسیریابها اطلاع داد که G دیگر کار نمی‌کند. D با ارسال بسته‌های خاصی این موضوع را بدانها اطلاع داده تا آنها نیز جدول مسیریابی خود را اصلاح نمایند. خود D نیز درایه‌های متناظر با E، G و I را از جدول مسیریابی خود حذف می‌کند.[۱]

شاید از توضیحاتی که در خصوص الگوریتم AODV ارائه کردیم مشخص نشده باشد که بنیانی‌ترین تفاوت بین این الگوریتم و الگوریتم **بلمن-فورد** آن است که گره‌ها بطور متناوب کلّ جدول مسیریابی خود را منتشر نمی‌کنند، بلکه فقط بخش کوچکی از آنرا و آنهم بر حسب تقاضا، ارسال می‌نمایند. این ویژگی، در پهنای باند مصرفی و طول عمر باطری گره‌های متحرک صرفه جویی می‌کند.

AODV همچنین قادر به پخش داده‌های فراگیر (Broadcast) و مسیریابی چندپخشی (Multicast) است. برای آگاهی بیشتر به مرجع (Perkins and Royer,2001) مراجعه نمائید. مسیریابی در شبکه‌های ویژه، یک موضوع پژوهشی داغ و جدید است و اخیراً در این خصوص مقالاتی تألیف و منتشر شده است. برخی از آنها را می‌توان در مراجع ذیل یافت:

Chen et al., 2002; Hu and Johnson, 2001; Li et al., 2001; Raju and Garcia-Luna-Aceves, 2001; Ramanathan and Redi, 2002; Royer and Toh, 1999; Spohn and Garcia-Luna-Aceves, 2001; Tseng et al., 2001; and Zadeh et al., 2002).

۱۱-۲-۵ جستجوی گره در شبکه‌های همتا به همتا (Peer-to-Peer)

یکی از پدیده‌های نسبتاً جدید، شبکه‌های همتا به همتا است که در آن تعداد کثیری از افراد برای به اشتراک گذاشتن منابع مشترک خود، مستقیماً با یکدیگر در تماسند (این افراد عموماً از طریق یک اتصال ثابت و باسیم به اینترنت متصل شده‌اند). اولین کاربرد بسیار گسترده از تکنولوژی «همتا به همتا» به یکی از بحث‌برانگیزترین ماجراهای «گناه جمعی» بدل شد: ۵۰ میلیون از کاربران Napster می‌توانستند فایلهای موزیک و آواز را که حق امتیاز آنها در

---

۱. توضیحی برای رفع ابهام از مفهوم همسایه‌های فعال خالی از لطف نیست. به سطر آخر از جدول ۵-۲۳-ب دقت کنید. این جدول فرضاً متعلق به D است و آخرین رکورد آن بیان می‌کند که برای رسیدن به گره I باید بسته‌ها به G فرستاده شوند و تا رسیدن به I فقط دو گام باقیمانده است؛ در فیلد «همسایه‌های فعال» نام A و B درج شده است یعنی: A و B برای ارسال بسته‌هایشان به سوی I آنها را به من -D- می‌دهند. لذا در صورت خرابی G باید به این دو همسایه اطلاع داد تا آنها نیز از این موضوع آگاه شده و جداول خود را اصلاح کنند. -م

تملک دیگران بود بدون اجازه از صاحبین آنها با یکدیگر رد و بدل کنند؛ نهایتاً Napster پس از مناقشات فراوان به حکم دادگاه تعطیل شد.[1] بهر تقدیر، تکنولوژی همتا به همتا کاربردهای قانونی بسیار جالبی دارد. این تکنولوژی با مسائل و مشکلات مسیریابی مواجه است؛ ولیکن این مسائل با آنچه که تاکنون مطالعه کرده‌ایم کاملاً یکسان نیست، لذا مروری اجمالی بر آن خالی از لطف نخواهد بود.

آنچه که سیستمهای «همتا به همتا» را جذّاب کرده آنست که سیستمی کاملاً توزیع‌شده ایجاد می‌کند یعنی تمام گره‌ها متقارن هستند و هیچگونه کنترل مرکزی یا سلسله‌مراتب خاصی بر آن حاکم نیست. در یک سیستم رایج همتا به همتا، هر یک از کاربران دارای مقداری اطلاعات هستند که ممکن است برای کاربران دیگر جالب باشد. این اطلاعات می‌تواند نرم‌افزارهای رایگان و عمومی، موزیک، عکس و نظایر آنها باشد. هر گاه تعداد کاربران زیاد باشد، یکدیگر را نمی‌شناسند و نمی‌دانند آنچه راکه در جستجوی آن هستند در کجا پیدا کنند. یک راه حلّ آن است که یک پایگاه دادهٔ مرکزی و بسیار بزرگ از فهرست آنها داشته باشیم ولی ممکن است به دلایلی امکان‌پذیر نباشد (مثلاً هیچکس حاضر به میزبانی و نگهداری آن نشود) بنابراین مسئله اساسی آن است که در غیاب یک پایگاه مرکزی یا یک فهرست مرکزی، یک کاربر چگونه می‌تواند گره‌ای راکه اطلاعات مورد نیاز او را در اختیار دارد، پیدا کند.

اجازه بدهید فرض را بر آن بگذاریم که هر کاربر یک یا چند آیتم داده مثل موزیک، عکس، برنامه، فایل و نظائر آنها در اختیار دارد و کاربران دیگر علاقمند به خواندن آنها هستند. هر آیتم توسط یک رشتهٔ ASCII نامگذاری شده است. کاربران احتمالی فقط همین رشتهٔ ASCII را می‌دانند و می‌خواهند بدانند آیا فردی یا افرادی آن را در اختیار دارند و اگر دارند آدرس IP آنان چیست؟

به عنوان مثال به پایگاه توزیع‌شدهٔ مردم‌شناسی (شجره‌نامهٔ مردم) دقت کنید. هر کسی که به مردم‌شناسی علاقمند است فرضاً در خصوص اجداد و بستگان خود اطلاعاتی مثل عکس، صدا یا حتی قطعات ویدیویی در اختیار دارد. ممکن است جدّ اعلای بسیاری از افراد یکی باشد و امکان دارد اطلاعات مربوط به آن در چندین «گره» وجود داشته باشد. نام هر رکورد عموماً نام فرد موردنظر (در شکل و قالب مشخص) است. حال در جایی یک نفر که علاقمند به تحقیق در مورد شجره‌نامهٔ خود بوده متوجه می‌شود که نام جدّ اعلای او در آرشیو یک گره خاصّ وجود دارد و ساعت جیبی و طلائی خود را برای برادرزاده‌اش به ارث گذاشته است!! حال او می‌خواهد بداند که نام این برادرزاده چیست و آیا در جایی دیگر رکوردی در این رابطه وجود دارد. بدون وجود یک پایگاه اطلاعات مرکزی، چگونه می‌توان کسی که چنین رکوردی را در اختیار دارد، پیدا کرد؟

برای حل این مسئله الگوریتمهای گوناگونی پیشنهاد شده است. الگوریتمی که بررسی می‌کنیم «**الگوریتم Chord**» نام دارد. (Dabek et al., 2001; Stoica et al., 2001) شرح سادهٔ عملکرد آن بدینگونه است: سیستم Chord از مجموعهٔ n کاربر شرکت‌کننده تشکیل شده است. هر یک از این کاربران ممکن است رکوردهایی را در خود ذخیره کرده و همچنین آمادگی داشته باشند بخشی از فهرست (ایندکس) سیستم را برای استفادهٔ دیگر کاربران نگهداری و عرضه کنند. هر یک از کاربران دارای یک آدرس IP هستند که این آدرس را می‌توان توسط یک «**تابع درهم‌سازی**» (Hash Function) به یک آدرس m بیتی تبدیل کرد. در سیستم Chord از تابع درهم‌سازی SHA-1 استفاده می‌شود؛ SHA-1 در مبحث رمزنگاری کاربرد دارد و در فصل هشتم نگاهی بدان خواهیم انداخت. فعلاً به همین اندازه بسنده می‌کنیم که این تابع یک رشته از بایتها با طول دلخواه را گرفته و براساس آن یک عدد ۱۶۰ بیتی

---

۱. در حقیقت کاری که Napster انجام می‌داد این بود که مشترکین خود راکه فایلهای موزیک بر روی کامپیوتر خودداشتند بهم معرفی می‌کرد تا بتوانند به صورت بی‌واسطه و رو در رو (یعنی همان همتابه همتا) فایلهای خود را ردّ و بدل کنند. اینکار خشم دست‌اندرکاران صنعت موسیقی را برانگیخت! -م-

بشدّت تصادفی تولید می‌کند. بنابراین می‌توانیم یک آدرس IP را به عددی ۱۶۰ بیتی و یکتا تبدیل کنیم که به این عدد اصطلاحاً «**شناسهٔ گره**» (Node Identifier) گفته می‌شود.

از دیدگاه مفهومی تجسّم کنید که تمام 2^{160} شمارهٔ شناسهٔ گره‌ها به صورت صعودی پیرامون یک دایرهٔ بزرگ چیده شده‌اند. برخی از این شماره‌ها مربوط به گره‌های شرکت‌کننده (کاربران فعّال) هستند ولی بیشتر آنها پوچند. در شکل ۵-۲۴-الف ما دایرهٔ شماره‌ها را برای m=۵ نشان داده‌ایم. (فعلاً به کمانهای میانی کاری نداشته باشید). m=۵ بدین معناست که شماره‌ها پنج بیتی هستند و شماره‌های از ۰ تا ۳۱ را در بر می‌گیرند. در این مثال گره‌های ۱، ۴، ۷، ۱۲، ۱۵، ۲۰ و ۲۷ مربوط به گره‌های واقعی بوده و در شکل به صورت سایه‌دار نشان داده شده‌اند. مابقی شماره‌ها وجود خارجی ندارند. (شماره‌های پوچ)

تابع *successor(k)* را بدین مضمون تعریف می‌کنیم که اولین «شناسهٔ گره» مربوط به اولین کاربر واقعی که پس از عدد k قرار گرفته را بر می‌گرداند. به عنوان مثال *successor*(6)=7، *successor*(8)=12 و *successor*(22)=27.

اسامی رکوردها (مثل نام فایلهای آواز و نظائر آن) نیز توسط تابع درهم‌سازی (یعنی SHA-1) به یک رشته ۱۶۰ بیتی تبدیل می‌شوند که این رشته «**کلید**» (Key) نامیده می‌شود. لذا برای تبدیل یک نام (یعنی نام ASCII هر رکورد) به «**کلید**» متناظر با آن، رابطهٔ *Key=hash(name)* را تعریف کرده‌ایم. برای این کار فقط کافی است پروسیجر محلی *hash* را فراخوانی نماییم. اگر کسی که یک رکورد مفید با نام name در اختیار دارد، بخواهد آن را در دسترس عموم قرار بدهد ابتدا یک شاخص دوتایی شامل (IP Address,name) ساخته و برای ذخیره کردن این شاخص بر روی یک گره مناسب، تابع *successor*(*hash*(name)) را فراخوانی می‌نماید. اگرچه ممکن است چندین رکورد با همین نام در گره‌های متفاوت موجود باشد ولیکن شاخص دوتایی آنها در یک گره ذخیره می‌شود.[1] بدین طریق، ایندکس رکوردها (شاخص دوتایی هر رکورد) به صورت تصادفی بر روی گره‌ها توزیع می‌شوند. برای آن که تحمل خرابی (Fault Tolerance) این سیستم بالا باشد می‌توان از تعداد p تابع مختلف hash استفاده کرد تا هر شاخص دوتایی در p گره مختلف ذخیره شود ولیکن فعلاً به این موضوع نخواهیم پرداخت.

اگر بعداً کاربری بخواهد آیتمی با نام *name* را جستجو نماید ابتدا آن را توسط تابع hash درهم‌سازی می‌کند تا کلید آن بدست آید. سپس تابع *successor*(key) را فراخوانی می‌کند تا آدرس IP گره‌ای که فهرست شاخصها را ذخیره کرده، پیدا شود. اگرچه اولین مرحله ساده است ولی مرحلهٔ دوم چندان ساده نیست. برای آن که بتوان آدرس IP گره‌ای را که متناظر با یک کلید خاص است پیدا کرد، هر گره باید چندین «**ساختمان دادهٔ نظارتی**»[2] در خود ذخیره نماید. یکی از این ساختمانهای داده، آدرس IP گره‌ای است که در دایرهٔ «شناسهٔ گره‌ها»، پس از خود گره قرار گرفته است. به عنوان مثال در شکل ۵-۲۴، گره بعدی ۴، گره ۷ است و گره بعدی ۷، گره ۱۲ است.

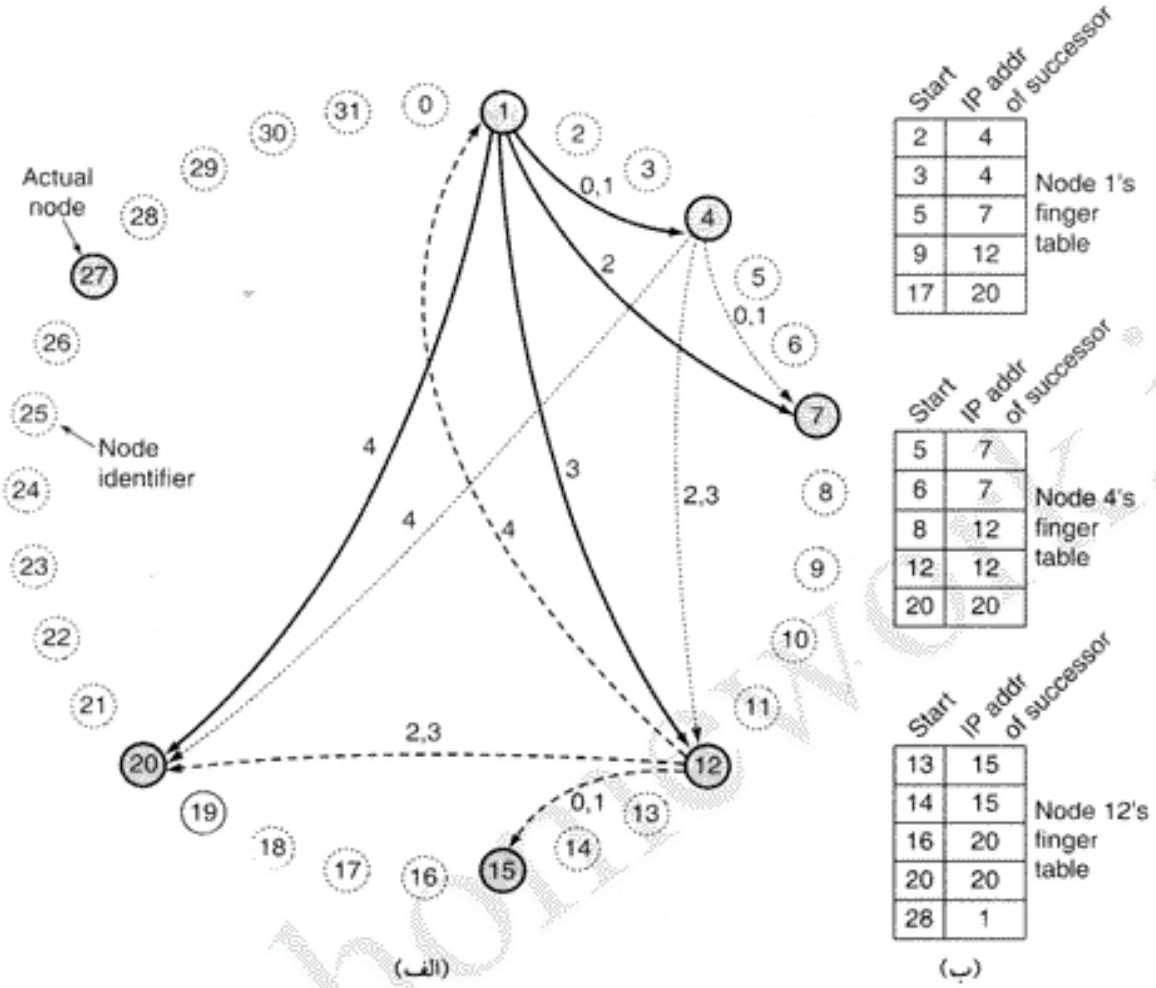
حال مراحل جستجو به ترتیب زیر ادامه می‌یابد: گره متقاضی بسته‌ای را برای گره بعدی خود ارسال کرده و ضمن اعلام آدرس IP خود، کلید مورد جستجو را مشخص می‌نماید. این بسته حول این دایره منتشر می‌شود تا آن که گره مورد جستجو پیدا شود. هر گره بررسی می‌کند که آیا اطلاعات مورد جستجو و منطبق با کلید را در اختیار دارد یا خیر. اگر داشته باشد آن اطلاعات را مستقیماً به آدرس IP متقاضی ارسال می‌نماید و در غیر اینصورت بستهٔ تقاضا را برای بعدی خود در دایره می‌فرستد.

اولین بهینه‌سازی این سیستم آن است که هر گره آدرس IP گره قبلی و بعدی خود را داشته باشد تا تقاضاها بتوانند در دو جهت ساعتگرد و پادساعتگرد (بسته به آن که کدام مسیر کوتاهتر است)، ارسال شوند. به عنوان مثال

---

۱. در حقیقت شناسنامهٔ هر رکورد مثل فایلهای صدا، تصویر یا برنامه، که شامل نام آن و آدرس IP ماشین نگهدارندهٔ آنست، بر روی ماشینهای دیگری قرار می‌گیرد. -م

۲. Administrative Data Structure



شکل ۵-۲۴. (الف) مجموعه‌ای از ۳۲ شناسهٔ گره که پیرامون یک دایره مرتب شده‌اند. گره‌های خاکستری رنگ متناظر با یک ماشین واقعی هستند. فلشها اشاره گرهای جدول Finger را نشان می‌دهند. برچسبهای هر فلش اندیسهای جدول Finger هستند. (ب) مثالی از جداول Finger

در شکل ۵-۲۴، گره ۷ می‌تواند برای یافتن گره ۱۰ در جهت ساعتگرد اقدام کند در حالی که برای یافتن گره ۳ می‌تواند به صورت پادساعتگرد عمل نماید.

حتی با وجود دو جهت برای جستجو، در سیستمهای عظیم «همتا به همتا» جستجوی خطی (Linear Search) بسیار ناکارآمد و کُند عمل می‌کند چرا که در هر جستجو بطور متوسط باید n/2 گره بررسی شود. برای بالا بردن سرعت جستجو، هر گره دارای جدولی است که در سیستم Chord اصطلاحاً Finger Table نامیده شده است. جدول Finger حاوی m درایه (Entry) است که از صفر تا m-1 شماره‌گذاری می‌شود و هر یک از آنها به آدرس گره‌های واقعی اشاره می‌کنند. هر یک از این درایه‌ها دارای دو فیلد هستند: فیلد start و آدرس IP گره‌ای که شناسهٔ آن معادل *successor*(start) است. در شکل ۵-۲۴-ب، جدول Finger برای سه گره نمونه نشان داده شده است. مقادیر فیلدهای مربوط به درایهٔ i در گره k عبارتست از:

$$start = k + 2^{i} \quad (\text{modulo } 2^{m})$$

IP address of *successor*(*start*[i])

دقت کنید که هر گره، آدرس IP تعداد نسبتاً کمی از گره‌ها را در خود ذخیره می‌کند و اغلب این گره‌ها دارای شمارهٔ

شناسهٔ نزدیک بهم هستند. [یعنی در هر گره آدرس IP گره‌هایی نگهداری می‌شوند که بر روی دایرهٔ شناسه‌ها، تقریباً در کنار آن گره باشند. -م]

با استفاده از جدول Finger ، جستجوی کلید Key در گره k به ترتیب زیر انجام می‌شود: اگر مقدار key بین k و *successor*(k) باشد، گره‌ای که اطلاعاتی در خصوص key در اختیار دارد همان *successor*(k) است و جستجو خاتمه می‌یابد. در غیر اینصورت در جدول Finger جستجو می‌شود تا درایه‌ای که در آن، مقدار فیلد start ، به *predecessor*(k) [یعنی شناسهٔ گره قبلی k در دایره] نزدیکتر است پیدا شود. این تقاضا مستقیماً به آدرس IP گره فوق‌الذکر ارسال می‌شود تا ادامهٔ جستجو در آنجا پیگیری شود. (آدرس IP گره مذکور از جدول Figer بدست می‌آید.) از آنجایی شناسهٔ این گره نزدیک به key و در عین حال کمتر از آن است لذا این شانس خوب وجود دارد که با تعداد کمی پرس‌وجوی دیگر نتیجهٔ مورد نظر بدست آید. در حقیقت چون که در هر جستجو فاصلهٔ باقیمانده تا هدف مورد نظر نصف می‌شود می‌توان نشان داد که متوسط تعداد جستجوها $log_2 n$ است.

به عنوان اولین مثال فرض کنید در گره ۱ ، کلید key=3 را جستجو می‌کنیم. از آنجایی که گره ۱ می‌داند که ۳ بین خودش و گره بعدی او یعنی ۴ واقع شده، لذا گره مورد نظر همان ۴ است و جستجو خاتمه یافته و آدرس IP گره ۴ بدست می‌آید.

در مثال دوم، فرض کنید در جستجوی کلید key=14 بر روی گره ۱ هستیم. چون که ۱۴ بین ۱ و ۴ نیست، از جدول Finger کمک گرفته می‌شود. نزدیکترین عدد قبل از ۱۴ ، ۹ است بنابراین، تقاضا به سوی آدرس IP گره حقیقی پس از ۹ که در جدول ۱۲ تعیین شده است هدایت می‌شود. گره ۱۲ می‌بیند که گره ۱۴ بین خودش و گره بعدیش یعنی ۱۵ قرار گرفته ، لذا آدرس IP گره ۱۵ را برمی‌گرداند.

به عنوان مثال سوم، فرض کنید بر روی گره ۱ به دنبال کلید key=16 هستیم. مجدداً این تقاضا برای گره ۱۲ ارسال می‌شود ولی در اینجا گره ۱۲ پاسخ این تقاضا را نمی‌داند و به همین دلیل نزدیکترین شمارهٔ قبل از ۱۶ یعنی ۱۴ را یافته و از طریق جدول خود آدرس IP گره ۱۵ را بدست می‌آورد؛ سپس تقاضا برای آن گره ارسال می‌شود. گره ۱۵ می‌بیند که عدد ۱۶ بین خودش و گره بعدی یعنی ۲۰ قرار دارد لذا آدرس IP گره ۲۰ را به سوال‌کننده بر می‌گرداند و به همین ترتیب کار ادامه می‌یابد تا به گره ۱ برسد.

از آنجایی که گره‌ها بطور متوالی به شبکه می‌پیوندند یا از آن جدا می‌شوند فلذا Chord نیازمند روشی است که بتواند این مسئله را نیز حل و فصل کند. فرض را بر آن می‌گذاریم که وقتی این سیستم آغاز به کار کرده تعداد گره‌ها آنقدر کم بوده که توانسته‌اند مستقیماً با یکدیگر مبادلهٔ اطلاعات کرده و اولین دایره و جداول Finger را بسازند. پس از آن ، به یک روال خودکار نیاز است: وقتی گره جدید r می‌خواهد به این سیستم بپیوندد باید با یکی از گره‌های موجود تماس برقرار کرده و از او بخواهد که آدرس IP گره *successor*(r) را برایش پیدا کند. پس از پیدا شدن، گره جدید از *successor*(r) می‌خواهد که گره قبلی خود خود را معرفی کند. در آخر، گره جدید از این دو گره می‌خواهد که او را به عنوان گره r در دایره وارد کند. به عنوان مثال اگر در شکل ۵-۲۴ گره ۲۴ بخواهد به سیستم بپیوندد از یکی از گره‌های فعال تقاضا می‌کند که *successor*(24) را برایش پیدا نماید که در این جا ۲۷ است. سپس از ۲۷ در مورد گره ماقبل او یعنی ۲۰ سؤال می‌کند. پس از آن که حضور خود را به این دو گره (یعنی ۲۰ و ۲۷) اعلام کرد گره ۲۰ گره ۲۴ را به عنوان گره بعدی خود و گره ۲۷ آن را به عنوان گره قبلی خود به رسمیت می‌شناسند. مضاف بر این، گره ۲۷ آن دسته از درایه‌هایی را که کلیدشان در محدودهٔ ۲۱ تا ۲۴ است به گره ۲۴ تقدیم می‌دارد. در این لحظه گره ۲۴ بطور کامل به جمع پیوسته است. ولیکن در این لحظه بسیاری از جداول Finger غلط هستند. برای اصلاح این جداول، هر گره یک پروسه را که در پس‌زمینه اجرا می‌شود ، اجرا می‌کند تا جداول مربوطه (با فراخوانی متناوب تابع *successor*) از نو محاسبه و اصلاح شوند. هر گاه در خلال عملیات تازه‌سازی، یکی از این

تقاضاها به گره جدیدی بر بخورد درایهٔ مربوط به آن نیز در جدول Finger بهنگام می‌شود.

وقتی یک گره به صورت مسالمت‌آمیز سیستم را ترک می‌کند کلیدهای خود را به گره بعدی خود در حلقه تسلیم می‌کند تا به او اطلاع بدهد که در آستانهٔ خروج از سیستم است و آن گره بتواند پیوند خود را با گره ماقبل از گره در آستانهٔ خروج، برقرار نماید. وقتی گره‌ای به ناگاه از کار بیفتد مشکل پیش می‌آید چراکه گره ماقبل از گره خراب شده، هیچ گره معتبر و فعال بعدی ندارد. برای کاهش اثر این مشکل، هر گره نه تنها آدرس مستقیم گره بعدی خود را نگه می‌دارد بلکه آدرس مستقیم s گره بعد از خود را نیز دارد تا حتی اگر s-1 گره متوالی از کار بیفتد باز هم بتوان دایره را اصلاح کرد.

از سیستم Chord برای ایجاد سیستم توزیع شدهٔ فایل (Dabek et al., 2001) و چند کاربرد دیگر استفاده شده و تحقیقات بر روی آن کماکان ادامه دارد. یک سیستم همتابه همتای دیگر به نام Pastry و کاربردهای آن در مرجع (Rowstron and Druschel, 2001) تشریح شده است. سیستم سومی به نام Freenet نیز معرفی شده که مستندات آن در (Clark et al., 2002) در دسترس عموم قرار دارد. چهارمین سیستم از این نوع نیز در مرجع (Ratnasamy,2001) تشریح شده است.

۵-۳ الگوریتمهای کنترل ازدحام

وقتی به بخشی از زیرشبکه، تعداد بسیار زیادی بسته تحویل شود کارآیی آن کاهش می‌یابد. بدین وضعیت «**ازدحام**» (Congestion) گفته می‌شود. شکل ۵-۲۵ ، علائم بروز این وضعیت را نشان می‌دهد. هر گاه تعداد بسته‌هایی که توسط ماشینهای میزبان به زیرشبکه سرازیر می‌شوند متناسب با ظرفیت حمل زیرشبکه باشد تمام این بسته‌ها تحویل مقصدشان خواهند شد (به استثنای آنهایی که در اثر خطای انتقال آسیب می‌بینند) و تعداد بسته‌های تحویلی متناسب با تعداد بسته‌های ارسالی است. ولیکن به محض افزایش بی‌رویهٔ ترافیک، مسیریابها قادر نیستند از عهدهٔ آن بر آمده و بسته‌ها شروع به از دست رفتن می‌کنند. این مسئله حادّتر نیز می‌شود و در ترافیک بسیار بالا کارآیی بطور کامل سقوط کرده و هیچ بسته‌ای تحویل مقصد نخواهد شد!

شکل ۵-۲۵. وقتی ترافیک تحویلی به شبکه بیش از اندازه باشد ازدحام بوجود می‌آید و کارآیی بشدت افت می‌کند.

چندین عامل می‌تواند به بروز ازدحام بینجامد. اگر به ناگاه دنباله‌ای از بسته‌ها بر روی سه یا چهار خط ورودی دریافت شده و خط خروجی همهٔ آنها یکی باشد صف تشکیل خواهد شد و اگر فضای حافظهٔ کافی برای نگهداری تمام آنها وجود نداشته باشد، بسته‌ها از بین می‌روند. شاید اضافه کردن حافظه مفید به نظر برسد ولیکن «ناگل» (Nagle,1987) بدین نتیجه رسید که حتی اگر حافظهٔ مسیریاب نامحدود باشد، وضعیت ازدحام نه تنها بهتر

نمی‌شود بلکه بدتر هم خواهد شد زیرا بسته‌ها، زمانی به سر صفّ می‌رسند که مهلت رسیدنشان به مقصد تمام شده و نسخه‌های تکراری آن ارسال شده‌اند. تمام این بسته‌های تکراری نیز برحسب وظیفه‌شناسی به مسیریاب بعدی هدایت شده و در تمام مسیر رسیدن به مقصد، «بار» افزایش می‌یابد. [بعبارت دیگر، «ازدحام» قابلیت انتشار در کل یک مسیر دارد.]

پردازنده‌های کُند نیز می‌توانند عامل بروز ازدحام باشند. اگر پردازندهٔ اصلی مسیریاب در انجام وظائف محوله به خود (مثل عملیات صف‌بندی، بهنگام سازی جداول مسیریابی و نظائر آن) کُند عمل نماید، صف ایجاد می‌شود، حتی وقتی که ظرفیت خطوط خروجی بیش از حدّ مورد نیاز است.

به دلیل مشابه، خطوط با پهنای باند کم نیز می‌توانند منجر به بروز ازدحام شوند. ارتقاء پهنای باند خطوط بدون تغییر در پردازنده‌ها یا بالعکس اغلب فایدهٔ چندانی ندارد و فقط جای گلوگاه و منشاء مشکل را تغییر می‌دهد. همچنین ارتقاء کامل یک بخش کوچک [مثل تغییر یک مسیریاب و ارتقاء ظرفیت خطوط آن] بدون تغییر در کلّ سیستم، فقط محل بروز مشکل و گلوگاه را جابجا می‌کند و در بهبود کارآیی کل سیستم تأثیر چشمگیری نخواهد داشت. مشکل اساسی یک سیستم، عدم تطابق و تناسب بخشهای مختلف آنست. این مشکل تا زمانی که کلیهٔ مؤلفه‌های سیستم متعادل نشوند باقی خواهد ماند.

باید به صراحت تفاوت بین «**کنترل ازدحام**»[1] و «**کنترل جریان**»[2] و همچنین ارتباط ظریف این دو مکانیزم را تبیین کنیم. «کنترل ازدحام» مکانیزمهایی است «جهت ایجاد اطمینان از این که زیرشبکه قادر به حمل ترافیک عرضه شده به آن هست». این مشکل یک مورد همگانی و سراسری است و از رفتار و عملکرد تمام ماشینهای میزبان، کلّیه مسیریابها، عملیات پردازشی «ذخیره و هدایت» (Store & Forward) درون مسیریابها یا هر عامل دیگری که ظرفیت حمل زیرشبکه را کاهش بدهد، ناشی می‌شود.

در مقابل، «**کنترل جریان**» به ترافیک نقطه‌به‌نقطه بین یک فرستنده و گیرندهٔ مفروض مربوط می‌شود و وظیفهٔ اصلی آن ایجاد اطمینان از این موضوع است که یک فرستندهٔ سریع نمی‌تواند متوالیاً داده‌ها را با سرعتی بیش از توانایی دریافت گیرنده، ارسال نماید. در کنترل جریان، گزارشات و فیدبکهای مستقیمی از گیرنده به فرستنده ارسال می‌شود تا به فرستندهٔ طرف مقابل تفهیم کند که کارها را چگونه انجام بدهد.

برای درک اختلاف بین این دو مفهوم، یک شبکهٔ فیبرنوری با ظرفیت 1000 Gagabits/sec را در نظر بگیرید که در آن یک اَبَرکامپیوتر سعی می‌کند با سرعت یک گیگابیت بر ثانیه فایلی را برای یک کامپیوتر شخصی، ارسال کند! اگرچه هیچگونه ازدحامی رخ نخواهد داد (خود شبکه مشکلی ندارد) ولیکن برای آن که بتوان به نحوی اَبَرکامپیوتر را وادار کرد که هرازگاهی متوقف شود و اجازهٔ نفس کشیدن به کامپیوتر شخصی بدهد به مکانیزمهای کنترل جریان نیاز است.

در سمت مقابل، یک شبکهٔ «ذخیره و هدایت» با خطوط 1Mbps و هزار کامپیوتر بزرگ را در نظر بگیرید که در آن نیمی از کامپیوترها سعی می‌کنند با سرعت 100Kbps برای نیم دیگر، فایل ارسال کنند! در اینجا مشکل تحت فشار قرار گرفتن گیرندهٔ کُند توسط فرستندهٔ سریع مطرح نیست بلکه مسئله آنست که ترافیک تحمیل شده به شبکه از میزان ظرفیت و توانایی آن بیشتر است.

دلیل آن که مفهوم کنترل ازدحام، اغلب با مفهوم کنترل جریان اشتباه می‌شود آن است که در برخی از الگوریتمهای کنترل ازدحام، هنگام بروز مشکل برای شبکه، پیغامهایی را برای ماشینهای مبداء ارسال کرده و به آنها تفهیم می‌کند که باید سرعت خود را پایین بیاورند. بنابراین یک ماشین میزبان ممکن است به دو دلیل پیغام

---

۱. Congestion Control ۲. Flow Control

Slowdown (آهسته‌تر ارسال کن) را دریافت کند: اول آن که گیرنده نتواند با همان سرعتی که فرستنده ارسال می‌کند داده‌ها را دریافت نماید. دوم آن که شبکه با ازدحام مواجه شود و از عهدهٔ ارسال داده‌ها بر نیاید. بعداً باز هم به این موضوع بر می‌گردیم.

مطالعات خود پیرامون کنترل ازدحام را با نگاهی به مدل و روشهای عمومی برخورد با آن آغاز می‌کنیم. سپس به راهکارهای گسترده‌ای خواهیم پرداخت که سعی می‌کنند در همان ابتدا از بروز این مشکل پیشگیری کنند. همچنین الگوریتمهای پویایی را مرور می‌کنیم که پس از بروز مشکل ازدحام، سعی در حلّ و فصل آن می‌کنند.

۵-۳-۱ اصول کلّی در کنترل جریان

بسیاری از مشکلات و مسائل سیستمهای پیچیده مثل شبکه‌های کامپیوتری را می‌توان از دیدگاه نظریهٔ کنترل بررسی کرد. در این روش تمام راه حلها به دو گروه تقسیم می‌شوند: «حلقه باز» و «حلقه بسته»[1]. راه‌حلهای «حلقه باز» سعی می‌کنند یک مسئله را با طراحی خوب حل کرده و از همان ابتدا اطمینان بدهند که مشکلی رخ نخواهد داد. وقتی این سیستم شروع به کار و انجام فعالیت نمود هیچگونه نظارت یا تصحیح عملکرد ممکن نیست.

تمهیداتی که برای کنترل ازدحام به سَبک «حلقه باز» پیش‌بینی شده عبارتند از: تصمیم‌گیری در خصوص زمان پذیرش ترافیک جدید ، تصمیم‌گیری در خصوص زمان حذف بسته‌ها ، انتخاب بسته‌هایی که باید حذف شوند و تصمیم‌گیری در خصوص زمان‌بندی صحیح در شبکه؛ حقیقت مشترک در تمام این موارد آنست که تصمیم‌گیری آنها مبتنی بر شرایط جاری شبکه نیست.

در مقابل، راه حلهای «حلقه بسته» مبتنی بر مفهوم حلقه‌های فیدبک هستند. اینگونه راهکارهای کنترل ازدحام، سه بخش را در بر می‌گیرند:

۱. نظارت بر سیستم به منظور تشخیص آنکه در کجا و چه وقت ازدحام رخ داده است.

۲. تحویل این اطلاعات به محلی که بایستی واکنش نشان بدهد.

۳. تنظیم عملکرد سیستم برای رفع مشکل

برای نظارت و تشخیص بروز ازدحام در زیرشبکه می‌توان معیارهای گوناگونی را بکار گرفت. مهمترین آنها عبارتند: درصد بسته‌هایی که به دلیل فقدان فضای کافی بافر حذف می‌شوند، متوسط طول صف، تعداد بسته‌هایی که مهلت ارسال آنها منقضی شده و از نو ارسال گردیده‌اند، متوسط تأخیر بسته و انحراف معیار[2] تأخیر بسته . در تمام این معیارها رشد اعداد نمایانگر افزایش ازدحام است.

دومین مرحله از «حلقه فیدبک» آن است که اطلاعاتی در خصوص ازدحام، از محل تشخیص و بروز آن به محلی که می‌تواند کاری برای حلّ آن انجام بدهد، انتقال یابد. بدیهی‌ترین روش آن است که مسیریاب کشف‌کنندهٔ ازدحام بسته‌ای خاصّ برای ماشین یا ماشینهای مبداء بفرستد و مشکل را به آنها اعلام نماید. البته این بسته‌های اضافی خودشان به بار شبکه می‌افزایند، آن هم دقیقاً زمانی که شبکه ظرفیت هیچ بار اضافی ندارد یعنی دقیقاً حین ازدحام در زیرشبکه!

با این وجود روشهای دیگری نیز امکان‌پذیر است. به عنوان مثال می‌توان یک بیت یا یک فیلد در هر بسته کنار گذاشت تا هر گاه ازدحام از یک حد آستانه فراتر رفت، مسیریاب آنرا پر کند. هر گاه مسیریاب تشخیص بدهد که ازدحام رخ داده در تمام بسته‌های خروجی، این فیلد را پر می‌کند تا به همسایه‌های خود در این خصوص هشدار بدهد.

راهکار دیگر آن است که ماشینهای میزبان یا مسیریابها بطور متناوب بسته‌های «**آزمون**» تولید و ارسال کرده و

---

۱. Open loop , Close loop ۲. Standard deviation

مستقیماً در خصوص ازدحام کسب آگاهی کنند. بکمک این اطلاعات می‌توان بسته‌ها را بنحوی مسیریابی کرد که از ناحیهٔ بروز مشکل، ردّ نشوند. به مثابهٔ ایستگاههای کنترل ترافیک که هلی‌کوپترهای آنها بر روی شهرها به پرواز درمی‌آیند تا به شنودگان در حال حرکت هشدار بدهند تا بسوی نقاط بحران، حرکت نکنند.

در تمام روشهای مبتنی بر فیدبک، انتظار می‌رود که آگاهی از ازدحام موجب شود ماشینهای میزبان برای کاهش ازدحام از خود واکنش مناسبی نشان بدهند. برای آنکه این روشها به درستی کار کنند بایستی مقیاس زمان به دقت تنظیم شده باشد. اگر وقتی دو بستهٔ پشت سر هم می‌رسند مسیریاب فریاد بزند «ایست» و به محض آنکه فقط برای ۲۰ میکروثانیه آزاد می‌شود فریاد بزند: «حرکت» ، سیستم بشدّت نوسانی عمل می‌کند و هیچگاه همگرا و پایدار نخواهد شد. برعکس اگر قبل از اعلام هر چیزی برای اطمینان به مدت ۳۰ دقیقه صبر کند، مکانیزم کنترل ازدحام آنقدر کُند واکنش نشان می‌دهد که عملاً هیچ سودی در دنیای واقعی نخواهد داشت. برای آنکه عملکرد خوبی انتظار داشته باشیم به حالت میانه‌ای نیازمندیم ولیکن بدست آوردن «**ثابت زمانی سیستم**» (Time Constant) مسئلهٔ چندان ساده‌ای نیست.

الگوریتمهای متعددی برای کنترل ازدحام معرفی شده‌اند. برای آنکه بتوان این الگوریتمها را به روش قابل فهمی سازماندهی کرد، Yang و Reddy (۱۹۹۵) الگوریتمهای کنترل ازدحام را طبقه‌بندی کرده‌اند. آنها به همان ترتیبی که در بالا تشریح شد تمام الگوریتمها را به دو ردهٔ «حلقه باز» و «حلقه بسته» تقسیم‌بندی نمودند. مضاف بر این، الگوریتمهای «حلقه باز» را برحسب اینکه در ماشینهای مبداء عمل می‌کنند یا در ماشینهای مقصد به دو رده تقسیم کردند. الگوریتمهای «حلقه بسته» نیز به دو ردهٔ فرعی تقسیم شده‌اند: «الگوریتمهای مبتنی بر فیدبک مستقیم» و «الگوریتمهای مبتنی بر فیدبک ضمنی». در الگوریتمهای مبتنی بر فیدبک مستقیم، برای هشدار دادن به مبداء، بسته‌های فیدبک دقیقاً از محل بروز ازدحام برگردانده می‌شوند. در الگوریتمهای مبتنی بر فیدبک ضمنی، مبداء با استناد به برخی از مشاهدات و علائم محلّی (همانند زمان لازم برای برگشت بسته‌های ACK)بروز ازدحام را حدس می‌زند.

بروز ازدحام بدین معناست که «بار شبکه» موقتاً از «منابع» موجود در برخی از بخشهای سیستم بیشتر شده و شبکه از عهدهٔ این بار بر نمی‌آید. [منظور از منابع پهنای باند، توان پردازشی CPU ، حافظهٔ مسیریاب و نظایر آنهاست. -م] دو راه حلّ به ذهن متبادر می‌شود: افزایش منابع یا کاهش بار. به عنوان مثال زیرشبکه می‌تواند برای افزایش پهنای باند بین دو نقطه، موقتاً از یک خطّ تلفن (Dialup) کمکی بهره بگیرد. در سیستمهای ماهواره‌ای، افزایش توان فرستنده اغلب پهنای باند را افزایش می‌دهد. در ضمن اگر بجای استفادهٔ دائمی از بهترین مسیر، بخشی از بار بر روی مسیرهای دیگر تقسیم شود پهنای باند شبکه بطور مؤثری افزایش می‌یابد. نهایتاً آنکه می‌توان مسیریابهای یدکی را که فقط به عنوان پشتیبان در شبکه نصب شده‌اند (برای آنکه سیستم تحمل خرابی داشته باشد) به خدمت گرفت تا در هنگام بروز ازدحام، ظرفیت زیرشبکه افزایش یابد.

علیرغم این راهکارها، بسیاری از اوقات افزایش ظرفیت زیرشبکه ممکن نیست یا آنکه این ظرفیت تا آخرین حدّ افزایش یافته است. تنها راه باقیمانده برای رفع ازدحام کاهش بار است. چندین روش برای کاهش بار وجود دارد که از آن جمله می‌توان به این موارد اشاره کرد: (۱) اجتناب از سرویس دادن به برخی از کاربران (۲) کاهش سطح سرویس‌دهی به برخی یا تمام کاربران (۳) مجبور کردن کاربران به زمان‌بندی تقاضاهای خود به روشی قابل پیش‌بینی.[۱]

---

۱. از آنجایی که برخی از ماشینها ترافیک انفجاری و غیرقابل پیش‌بینی تولید می‌کنند لذا می‌توان آنها را وادار کرد تا بکمک مکانیزمهای خاصّ (همانند مکانیزم بافرینگ یا مکانیزمهای شکل‌دهی به ترافیک) حجم ترافیک خود را متعادل و قابل پیش‌بینی کنند. -م

برخی از این روشها که به اختصار آنها بررسی خواهیم کرد در زیرشبکه‌های نوع «مدار مجازی» به بهترین نحو قابل اعمال هستند. در زیرشبکه‌های مدار مجازی، این روشها در لایهٔ شبکه پیاده می‌شوند. در زیرشبکه‌های دیتاگرام علیرغم اصراری که بر تفکیک وظائف لایه‌ها وجود دارد بخشی از این روشها باید در اتصالات لایهٔ انتقال بکار گرفته شوند. در این فصل به کاربرد این روشها در لایهٔ شبکه می‌پردازیم و در فصل بعدی به عملیات مدیریت ازدحام در لایهٔ انتقال، نگاهی خواهیم انداخت.

۵-۳-۲ سیاستهای پیشگیری از ازدحام

اجازه بدهید مطالعهٔ روشهای کنترل ازدحام را با بررسی سیستمهای «حلقه باز» شروع کنیم. اینگونه سیستمها به گونه‌ای طراحی می‌شوند تا بجای آنکه بگذارند ازدحام اتفاق بیفتد و بعداً واکنش نشان بدهند، در همان ابتدا ازدحام را به حداقل برسانند. این روشها برای رسیدن بدین هدف، از سیاستهای خاصّ و مناسب در سطوح مختلف، بهره می‌گیرند. در شکل ۵-۲۶ سیاستهای مختلفی که در لایه پیوند داده، لایهٔ شبکه و لایهٔ انتقال، می‌تواند بر پدیدهٔ ازدحام تأثیر بگذارد، فهرست شده‌اند. (Jain,1990)

لایه	سیاستها
انتقال	• سیاستهای ارسال مجدد • سیاستهای ذخیره بسته‌هایی که خارج از ترتیب می‌رسند. • سیاستهای تصدیق وصول بسته‌ها (Ack) • سیاستهای کنترل جریان • تعیین زمان انقضای مهلت تایمرها
شبکه	• مدار مجازی در مقابل روش دیتاگرام در زیرشبکه • مکانیزمهای صف‌بندی بسته‌ها و روشهای متنوع سرویس‌دهی • سیاستهای حذف بسته • الگوریتم مسیریابی • مدیریت طول عمر بسته‌ها
پیوند داده	• سیاستهای ارسال مجدد • سیاستهای ذخیره بسته‌هایی که خارج از ترتیب می‌رسند. • سیاستهای تصدیق وصول بسته‌ها (ack) • سیاستهای کنترل جریان

شکل ۵-۲۶. سیاستهایی که بر پدیدهٔ ازدحام تاثیر می‌گذارند.

اجازه بدهید از لایهٔ پیوند داده شروع کرده و سپس به سمت بالا حرکت کنیم. سیاستهای ارسال مجدد (Retransmission Policy) با مسائلی از این قبیل سر و کار دارد: (۱) با چه سرعتی مهلت فرستنده (برای دریافت Ack یک بسته) خاتمه می‌یابد؟ (۲) در اثر انقضای مهلت چه چیزی ارسال می‌شود؟

یک فرستندهٔ عجول که مهلت آن به سرعت خاتمه می‌یابد و مبتنی بر پروتکل Go Back n تمام بسته‌های ارسالی را از نو می‌فرستد، بار بسیار سنگینتری را نسبت به فرستنده‌ای که از روش Selective Repeat استفاده کرده به زیرشبکه تحمیل می‌کند. یکی دیگر از مسائل که ارتباط تنگاتنگی با این موضوع دارد **سیاستهای بافرینگ** است. اگر گیرنده، بسته‌هایی را که خارج از ترتیب می‌رسند دور بیندازد تمام آنها باید از نو ارسال شوند و بار اضافی به زیرشبکه تحمیل خواهد شد. از دیدگاه کنترل ازدحام، روش «تکرار انتخابی» (Selective Repeat) به وضوح بهتر از روش Go Back n عمل می‌کند.

سیاست تصدیق دریافت بسته‌ها (Acknowledgement Policy) نیز بر روی ازدحام تأثیر می‌گذارد. اگر هر

بسته فوراً اعلام وصول شود، بسته‌های اعلام وصول (Ack) ترافیک زائد و اضافی تحمیل خواهند کرد. در عوض اگر برای صرفه‌جویی، از روش Piggybacking استفاده شود و اعلام وصول داده‌ها از طریق ترافیک معکوس انجام شود ممکن است مهلت فرستندهٔ داده‌ها متوالیاً منقضی شود و تکرار ارسال رخ بدهد که آن هم منجر به تحمیل بار اضافه خواهد شد. استفاده از یک روش محکم و سختگیرانه در الگوی کنترل جریان (مثلاً داشتن پنجرهٔ کوچک) می‌تواند نرخ ارسال داده را کاهش داده و به کاهش ازدحام کمک کند.

در لایهٔ شبکه، انتخاب بین روش «مدار مجازی» یا «دیتاگرام» تأثیر مستقیمی بر پدیدهٔ ازدحام دارد چراکه بسیاری از الگوریتمهای کنترل ازدحام فقط در زیرشبکه‌های مدار مجازی قابل‌اعمال هستند. صفّ‌بندی بسته‌ها و سیاستهای سرویس‌دهی نیز در کنترل ازدحام موثرند. این سیاستها بدین موضوع مرتبطند که آیا مسیریاب به ازای هر خط ورودی یک صفّ دارد، به ازای هر خط خروجی یک صفّ دارد یا هر دو صفّ را تشکیل می‌دهد. همچنین ترتیب پردازش بسته‌ها (مثلاً نوبت‌بندی چرخشی ـRound Robinـ یا روش مبتنی بر اولویت‌بندی) جزو سیاستها صف‌بندی محسوب می‌شود. «سیاست حذف بسته‌ها» قاعده‌ای است که براساس آن تعیین می‌شود که وقتی فضایی برای نگهداری بسته‌ها نیست کدامیک از بسته‌ها را باید حذف کرد. اتخاذ یک سیاست مناسب می‌تواند به کاهش ازدحام کمک کند و سیاستهای غلط شرایط ازدحام را بدتر خواهد کرد.

یک الگوریتم مسیریابی خوب می‌تواند با توزیع مناسب ترافیک بین خطوط مختلف به پیشگیری از ازدحام کمک کند در حالی که یک الگوریتم بد می‌تواند با ارسال ترافیک بسیار زیاد بر روی یک خط که با ازدحام مواجه شده شرایط را بدتر کند. مدیریت طول عمر بسته (Packet Lifetime) با این مسئله سر و کار دارد که چه مدت طول می‌کشد تا بسته [در صورت نرسیدن به مقصد] حذف شود. اگر این زمان بیش از حد طولانی باشد بسته‌های گمشده و سرگردان ممکن است تا قبل از موعد حذف شدن، شبکه را با انباشتگی و ازدحام مواجه کنند، در حالی که اگر این زمان کوتاه در نظر گرفته شود ممکن است بسته‌ها قبل از آنکه فرصت رسیدن به مقصد را داشته باشند حذف شوند و به تکرار ارسال بینجامد.

در لایهٔ انتقال همان موارد و سیاستهایی که در لایهٔ پیوند داده وجود داشت دنبال می‌شود ولی مضاف بر آنها روش تعیین بازه‌های زمانی انقضای مهلت (Timeout Interval) پیچیده‌تر است چراکه زمان عبور بسته‌ها از میان یک شبکه [با تعدادی مسیریاب و کانالهای متعدد] در مقایسه با عبور بسته‌ها از یک سیم واحد، چندان قابل پیش‌بینی نیست. اگر بازه‌های انقضای مهلت بیش از اندازه کوتاه باشد بسته‌های بی‌مصرف زیادی ارسال خواهد شد. اگر این زمان بیش از اندازه طولانی باشد، ازدحام کاهش می‌یابد ولیکن وقتی بسته‌ای به هر دلیل از دست برود زمان پاسخ [یعنی تأخیر ارسال مجدّد] مشکل‌آفرین خواهد شد.

۳-۳-۵ کنترل ازدحام در زیرشبکه‌های مدار مجازی

روشهای کنترل ازدحام که در بالا معرفی شدند اساساً «حلقه باز» هستند یعنی تلاش می‌کنند در همان ابتدا و قبل از بروز، از ازدحام پیشگیری کنند. در این بخش راهکارهایی را بررسی خواهیم کرد که در زیرشبکه‌های مدار مجازی به صورت پویا ازدحام را کنترل می‌کنند. در دو بخش آتی نیز نگاهی به تکنیکهای قابل استفاده در هر نوع زیرشبکه خواهیم انداخت.

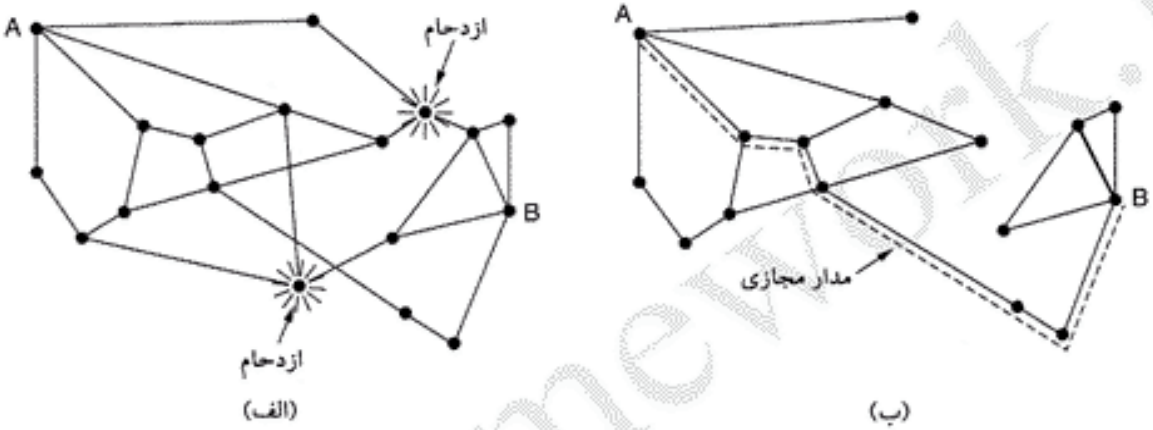
یکی از تکنیکهایی که بطور گسترده‌ای از آن در جهت پیشگیری از وخیم شدن مشکل ازدحام (که از قبل شروع شده) استفاده می‌شود روش «**کنترل پذیرش**»[1] (Admission Control) نام دارد. ایدهٔ این روش ساده است: هر گاه ازدحام گزارش شود تا رفع کامل مشکل، هیچگونه تقاضای تنظیم مدار مجازی پذیرفته نخواهد شد. بدین

---

۱. بمعنای محدود کردن پذیرش ارتباط و امتناع از ایجاد مدار مجازی جدید

ترتیب هر گونه تلاش برای ایجاد اتصال در لایهٔ انتقال با شکست مواجه خواهد شد چراکه پذیرش افراد جدید وضع ازدحام را وخیم‌تر خواهد کرد. هر چند این روش خام و ناشیانه است ولی در عوض ساده و سهل‌الاجراست. در سیستمهای تلفن نیز هر گاه یک سوئیچ با بار بسیار زیاد مواجه شود، با قطع بوق آزاد (Dial Tone)، پذیرشهای جدید را محدود می‌نماید.

راهکار دیگر آن است که اجازهٔ تنظیم مدار مجازی جدید داده شود ولیکن مسیرهای انتخابی برای مدارات مجازی جدید در خارج از نواحی بحران‌زده انتخاب گردد. به عنوان مثال زیرشبکه شکل ۵-۲۷-الف را در نظر بگیرید که در آن دو مسیریاب با ازدحام مواجه شده‌اند.



شکل ۵-۲۷. (الف) یک زیرشبکهٔ مواجه با ازدحام (ب) ترسیم مجدد گراف زیرشبکه پس از حذف مناطق درگیر ازدحام. مدار مجازی مناسب بین A و B نشان داده شده است.

فرض نمایید که ماشین میزبان متصل به مسیریاب A بخواهد یک «اتصال» با ماشین متصل به B برقرار نماید. طبیعتاً این اتصال از یکی از مسیریابهای دارای مشکل عبور خواهد کرد. برای اجتناب از این وضعیت می‌توانیم زیرشبکه را مطابق با شکل ۵-۲۷-ب از نو ترسیم کرده و مسیریابهای دچار ازدحام و تمام خطوط آنها را حذف نماییم. خط نقطه‌چین، مسیر ممکن برای ایجاد مدار مجازی بین A و B را نشان می‌دهد. در این مسیر، مسیریابها و خطوط دچار ازدحام وجود ندارد.

استراتژی دیگر در زیرشبکه‌های مدار مجازی، آنست که در حین تنظیم یک مدار مجازی، بین ماشین میزبان و زیرشبکه توافقاتی صورت بگیرد. در این توافقات، عموماً حجم و شکل ترافیک، کیفیت مورد نیاز خدمات و پارامترهای دیگر مشخص می‌شوند. به منظور عمل به مفاد این قرارداد، زیرشبکه در هنگام تنظیم مسیر، منابع مورد نیاز را در طول مسیر، از قبل کنار می‌گذارد. این منابع شامل فضای بافر، فضای جدول در هر مسیریاب و پهنای باند خطوط است. در این روش احتمال بروز ازدحام در مدارات مجازی نادر است چرا که تمام منابع مورد نیاز از قبل رزرو و موجود بودن آن تضمین می‌شود.

این گونه رزروسازی می‌تواند به صورت یک روال عملیاتی استاندارد همیشه انجام شود و یا فقط در شرایطی اتخاذ شود که شبکه با ازدحام مواجه شده است. اشکال استفادهٔ همیشگی از این روش آن است که می‌تواند منابع را هدر بدهد. به عنوان مثال اگر شش مدار مجازی که هر یک به پهنای باند 1Mbps نیازمندند همگی از یک خط فیزیکی 6Mbps عبور کرده باشند مسیریاب مجبور به علامتگذاری آن به عنوان خط پر خواهد بود در حالی که به ندرت اتفاق می‌افتد که هر شش مدار مجازی بطور همزمان مورد استفاده قرار بگیرند. نتیجتاً در شرایط معمولی هزینه‌ای که برای کنترل ازدحام پرداخت می‌شود پهنای باند بلااستفاده (تلفاتی) است.

۵-۳-٤ کنترل ازدحام در زیرشبکه‌های دیتاگرام

حال اجازه بدهید به راهکارهایی بپردازیم که می‌توان از آنها در زیرشبکه‌های دیتاگرام (و همچنین زیرشبکه‌های مدار مجازی) بهره گرفت. هر مسیریاب می‌تواند براحتی بر میزان بهره‌وری (Utilization) خطوط خروجی و دیگر منابع خودش نظارت داشته باشد. به عنوان مثال مسیریاب می‌تواند به هر یک از خطوط خود یک متغیر اعشاری u که مقداری بین 0.0 تا 1.0 دارد نسبت بدهد. مقدار این متغیر میزان بهره‌وری خط را منعکس می‌کند. برای آنکه بتوان تخمین دقیقی از u داشت می‌توان بطور متناوب و در لحظات خاصّ از میزان بهره‌وری خطّ[1] نمونه‌برداری کرد و با فرض آنکه این مقدار f محاسبه شده باشد (f نیز بین صفر و یک است)، مقدار u را طبق رابطهٔ زیر بهنگام سازی نمود:

$$u_{new} = \propto.u_{old}+(1-\propto).F$$

$\propto$ یک ثابت است که تعیین می‌کند مسیریاب با چه سرعتی وضعیت گذشته را فراموش خواهد کرد.

هر گاه u از یک مقدار آستانه (Threshold) [یا به عبارتی از حدّ مجاز] تجاوز کند آن خط خروجی در وضعیت «هشدار» وارد می‌شود و در این صورت باید عملیاتی برای خروج از این وضعیت انجام گیرد. این عملیات می‌تواند یکی از گزینه‌های زیر باشد که در ادامه آنها را تشریح خواهیم کرد.

بیت هشدار (The Warning Bit)

در معماری قدیم شبکه‌های DECNET، «وضعیت هشدار» با تنظیم یک بیت خاص در سرآیند بسته‌ها، اعلام می‌شد. در شبکهٔ Frame Relay نیز همینگونه است. وقتی بسته‌ای به مقصد خود می‌رسید، لایهٔ انتقال همان بیت را در درون بستهٔ Ack کپی کرده و آن را برای مبداء پس می‌فرستاد و بدین نحو ماشین مبداء ترافیک خود را تقلیل می‌داد.

مادامیکه مسیریاب در وضعیت هشدار قرار داشت، تنظیم این بیت نیز ادامه می‌یافت بدین معنا که ماشین مبداء در بسته‌های بعدی Ack بازهم این بیت را دریافت می‌کرد. مبداء نیز با شمارش بسته‌های Ack حساب می‌کرد که در چه کسری از این بسته‌ها بیت هشدار وجود دارد و براساس آن نرخ انتقال خود را تنظیم می‌نمود. تا وقتی که دریافت این بیتها ادامه می‌یافت مبداء نیز به کاهش نرخ ارسال خود ادامه می‌داد. پس از کاهش نرخ ارسال تا حدّ نهایی، مجدداً نرخ ارسال شروع به افزایش می‌کرد [در صورت خروج از وضعیت هشدار]. دقت کنید که چون هر مسیریاب واقع بر روی مسیر می‌توانست «بیت هشدار» را به ۱ تنظیم کند لذا فقط زمانی ترافیک افزایش می‌یافت که هیچ یک از مسیریابها مشکلی نداشتند.

بسته‌های دعوت به آرامش (Choke Packet)

الگوریتم کنترل ازدحام قبلی نسبتاً زیرکانه است چراکه با یک بیان غیرمستقیم و تلویحی به مبداء تفهیم می‌کند که باید از سرعت خود بکاهد. چرا این کار مستقیماً انجام نشود؟ برای این کار مسیریاب یک بسته به نام **«دعوت به آرامش»** (Choke Packet) به ماشین مبداء بر گردانده و در آن آدرس مقصد بسته را نیز درج می‌کند [تا ماشین مبداء آگاه شود بسته‌های ارسالی او در راه رسیدن به کدام مقصد در مسیر پر ازدحام قرار گرفته‌اند و بداند که نرخ ارسال برای چه مقصدی را باید کاهش بدهد چراکه یک ماشین ممکن است بطور همزمان برای چند ماشین بسته ارسال کند -م]. البته بستهٔ اصلی، علامتگذاری شده و به راه خود ادامه می‌دهد (فقط یک بیت خاص در سرآیند

---

۱. نسبت داده‌های ارسالی بر روی خطّ به ظرفیت کلّ خط را بهره‌وری آن خطّ در نظر بگیرید. اندازه‌گیری این نسبت برای هر خط چندان دشوار نیست و برخی از سخت‌افزارهای شبکه امکان این اندازه‌گیری را در اختیار می‌گذارند. -م

بسته تنظیم می‌شود). این علامتگذاری از آن جهت انجام می‌شود که بسته‌های «دعوت به آرامش» بیشتری در طول مسیر تولید نشود؛ سپس بستهٔ اصلی بروش معمول به سوی مقصد خود هدایت می‌شود.

هر گاه ماشین مبداء، بستهٔ دعوت به آرامش دریافت کند، ملزم به کاهش ترافیک ارسالی بدان مقصد خاصّ (تا X درصد) می‌باشد. از آنجایی که ممکن است بسته‌های دیگری که قبلاً به همان مقصد روانه شده‌اند در همان مسیر حرکت کرده باشند و آنها نیز منجر به تولید بسته‌های «دعوت به آرامش» دیگری شده باشند لذا ماشین مبداء پس از دریافت اولین بستهٔ «دعوت به آرامش» به مدت زمان ثابت و مشخصی بسته‌هایی از این نوع را نادیده می‌گیرد. پس از طی این مدت، ماشین میزبان مجدداً به اندازهٔ زمان مشخصی گوش می‌دهد که ببیند آیا «بستهٔ دعوت به آرامش» دیگری دریافت می‌شود؟ اگر چنین بسته‌ای دریافت شد، خط کماکان در وضعیت ازدحام است فلذا باز هم جریان داده‌های خود را کاهش می‌دهد و مجدداً برای مدتی بسته‌های دعوت به آرامش را نادیده می‌گیرد. برعکس اگر در خلال مدت گوش دادن هیچ بستهٔ دعوت به آرامش دریافت نشد، ماشین میزبان می‌تواند جریان داده‌های خود را افزایش بدهد. در این پروتکل «فیدبک ضمنی» کمک می‌کند تا بتوان قبل از آنکه جریان بسته‌ها بطور کلّ مسدود شود از ازدحام پیشگیری کرد مگر آنکه مشکلی جدی رخ بدهد.

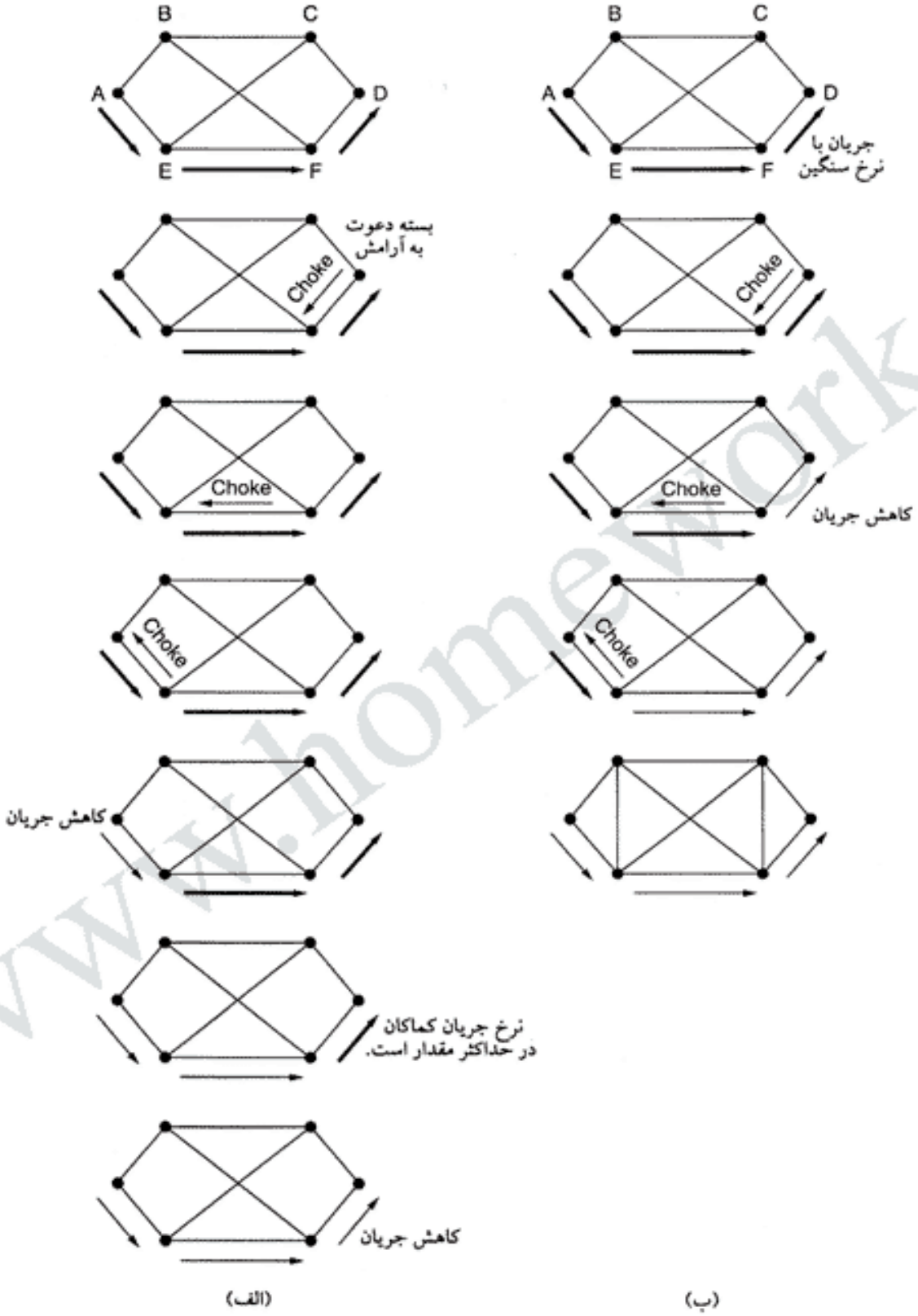
ماشینهای میزبان می‌توانند با تنظیم پارامترها و تغییر در سیاستهای کنترل ازدحام مثل اندازهٔ پنجره، ترافیک ارسالی خود را کاهش بدهند. عموماً دریافت یک بستهٔ «دعوت به آرامش» باعث پنجاه درصد کاهش در نرخ قبلی ارسال می‌شود. بستهٔ بعدی، ترافیک را به ۲۵ درصد تقلیل می‌دهد و به همین ترتیب. [به عبارت دیگر دریافت بستهٔ دعوت به آرامش مقدار فعلی نرخ ترافیک را عموماً نصف می‌کند. -م] افزایش نرخ ارسال معمولاً ضریب رشد کمتری دارد تا از بروز مجدد و پیاپی ازدحام پیشگیری شود. [یعنی مثلاً اگر دریافت بستهٔ دعوت به آرامش نرخ ارسال را نصف می‌کند عدم دریافت مجدد آن در یک دورهٔ زمانی مشخص، باعث دو برابر شدن نرخ ارسال نخواهد شد. -م]

چندین گونه از این الگوریتم برای کنترل ازدحام ارائه شده است. مثلاً در یکی از آنها، مسیریاب چندین سطح آستانه (Threshold) در نظر می‌گیرد. بسته به آنکه وضعیت از کدامیک از سطوح تجاوز کرده باشد بسته‌های دعوت به آرامش می‌توانند در برگیرندهٔ یکی از سه هشدار «وضعیت احتیاط»، «وضعیت حادّ» و «بحران قطعی» باشند.

در گونهٔ دیگری از این الگوریتم برای صدور علائم هشداردهنده، بجای استفاده از معیار بهره‌وری خط از معیار طول صف یا میزان استفاده از فضای بافر بهره گرفته می‌شود. البته می‌توان از یک رابطهٔ وزن‌دهی نمایی (Exponential Weighting) استفاده کرد و براساس تمامی این معیارها یک مقدار تلفیقی برای میزان بهره‌وری (u) تعریف نمود.

بسته‌های دعوت به آرامش گام‌به‌گام (Hop-by-Hop Choke Packet)

در سرعتهای بالا یا در مسیرهای طولانی، ارسال بسته‌های «دعوت به آرامش» به ماشین مبداء، کارآیی خوبی نخواهد داشت چراکه واکنش این روش بسیار کُند است. به عنوان مثال فرض کنید که ماشینی در سان‌فرانسیسکو (مثلاً مسیریاب A در شکل ۵-۲۸) ترافیکی از بسته‌ها را با سرعت ۱۵۵ مگابیت در ثانیه برای ماشینی در نیویورک (مثلاً مسیریاب D در شکل ۵-۲۸) ارسال کند. اگر بافر ماشین در نیویورک، رو به پرشدن بگذارد حدوداً سی میلی‌ثانیه طول می‌کشد تا بسته‌های دعوت به آرامش به سان‌فرانسیسکو برگردد و از مبداء بخواهد کُندتر ارسال کند. در شکل ۵-۲۸-الف انتشار و بازگشت «بستهٔ دعوت به آرامش» در مراحل دوم، سوم و چهارم نشان داده شده است. در خلال این سی میلی‌ثانیه تاخیر، ۴/۶ مگابیت اطلاعات دیگر فرستاده خواهد شد. حتی اگر پس از دریافت



شکل ۵-۲۸. (الف) یک بستهٔ دعوت به آرامش که فقط مبداء را تحت تاثیر قرار می‌دهد. (ب) یک بستهٔ دعوت به آرامش که در هر گام، بر روی مسیریابها واقع بر مسیر تاثیر می‌گذارد.

این هشدار ماشین واقع در سانفرانسیسکو فوراً ارسال خود را بطور کلّ قطع نماید، ۴/۶ مگابیت اطلاعات بر روی خط به سوی ماشین روانه هستند و باید به نحوی تکلیف آنها مشخص شود. فقط در مرحلهٔ هفتم از شکل ۵-۲۸-الف مسیریاب واقع در نیویورک متوجه کاهش ترافیک خواهد شد.

راهکار دیگر آن است که بستهٔ دعوت به آرامش در هر گام از مسیری که می‌پیماید تأثیری بر عملکرد مسیریابهای میانی بگذارد. (به شکل ۵-۲۸-ب دقت کنید). در اینجا به محض آنکه بستهٔ دعوت به آرامش به مسیریاب F می‌رسد، F ملزم به کاهش ترافیک ارسالی به D خواهد بود. انجام این کار منوط به آن است که F فضای بافر بیشتری را برای جریان بسته‌ها اختصاص بدهد چراکه ماشین مبداء با سرعت کامل در حال ارسال است و F در این میان می‌تواند کمکی موقت و سریع به D بنماید [با ایجاد توقف مصنوعی]؛ این کار را می‌توانید همانند آگهی‌های تجاری فرض کنید که شما را موقتاً از سردرد ناشی از دیدن یک برنامهٔ طولانی نجات می‌دهند!! در گام بعدی بستهٔ دعوت به آرامش به E می‌رسد و به او نیز تفهیم می‌شود که ترافیک خود به F را کاهش بدهد. این عمل نیز مستلزم وجود فضای بافر بیشتری در E است ولی F را موقتاً یاری می‌دهد. نهایتاً بستهٔ دعوت به آرامش به A می‌رسد و جریان بطور واقعی و از مبداء آن کاهش می‌یابد.

تأثیر نهایی این روش گام‌به‌گام (Hop-by-Hop) آن است که در لحظه بروز ازدحام به سرعت چاره‌ای اندیشیده می‌شود ولی این راهکار به بهای افزایش فضای بافر ارسال، تمام خواهد شد. در این روش می‌توان بدون از دست رفتن هیچ بسته‌ای، مانع از افزایش ازدحام شد. این نظریه و نتایج شبیه‌سازی آن در مرجع (Mishra and Kanakia, 1992) تشریح شده است.

۵-۳-۵ دور ریختن بار (Load Shedding)

هر گاه هیچیک از روشهای فوق مشکل ازدحام را رفع نکند، مسیریابها می‌توانند آخرین تیر ترکش خود را بیازمایند: «**دور ریختن بار**»! عمل دور ریختن بار، آخرین و بدترین روشی است که به مسیریاب اجازه می‌دهد هرگاه با سیل بسته‌هایی که نمی‌تواند از عهدهٔ هدایت آنها برآید، مواجه شود آنها را دور بریزد. (این اصطلاح از ادبیات شرکتهای تولید انرژی برق گرفته شده که در آنجا نیز مثلاً در یک روز گرم تابستان که مصرف برق از میزان تولید بیشتر می‌شود، برق بخشی از مناطق عمداً قطع می‌شود تا کل مناطق با مشکل قطع برق مواجه نشوند!)

یک مسیریاب که غرق در بسته‌های اطلاعاتی شده می‌تواند تصادفاً برخی از آنها را انتخاب و حذف کند ولی از این بهتر هم می‌تواند عمل کند. اینکه کدام بسته باید حذف شود به نوع برنامهٔ کاربردی که آنرا تولید کرده بستگی دارد. در انتقال فایل بسته‌های قدیمی ارزشمندتر از بسته‌های جدید هستند چراکه حذف بستهٔ ۶ و نگه داشتن بسته‌های ۷ تا ۱۰ باعث ایجاد یک شکاف در میان داده‌های گیرنده شده و ممکن است فرستنده مجبور شود بسته‌های ۶ تا ۱۰ را نیز از نو ارسال کند (اگر گیرنده بطور طبیعی بسته‌های خارج از ترتیب را حذف کند.) در یک فایل ۱۲ بسته‌ای، حذف بستهٔ ۶ ممکن است منجر به تکرار ارسال بسته‌های ۶ تا ۱۰ شود در حالی که حذف بستهٔ ۱۰ ممکن است فقط به تکرار بسته‌های ۱۰ تا ۱۲ نیاز داشته باشد. در طرف مقابل برای کاربردهای چند رسانه‌ای (مثل ارسال صدا یا تصویر)، بسته‌های جدید مهمتر از بسته‌های قدیمیند.

برای آن که سطح هوشمندی الگوریتم حذف بسته‌ها از این هم فراتر برود به همکاری فرستندهٔ بسته‌ها نیاز است. در بسیاری از برنامه‌های کاربردی، برخی از بسته‌ها بسیار مهمتر از بقیه هستند. به عنوان مثال در برخی الگوریتمهای ارسال ویدیوی فشرده شده، در لحظات خاصی یک فریم تصویر کامل ارسال می‌شود و پس از آن فریمهای تصویر بعدی کامل نیستند بلکه فقط تفاوت این فریمها با آخرین فریم کامل محاسبه و پس از فشرده‌سازی ارسال می‌شوند. در چنین حالتی حذف بسته‌ای که بخشی از فریمهای فرعی محسوب می‌شود ارجح‌تر از حذف بسته‌ای است که به فریم اصلی و کامل تعلق دارد. به عنوان مثالی دیگر، انتقال یک سند

(Document) حاوی تصویر و متن ASCII را در نظر بگیرید. از دست دادن یک خط از نقاط تصویر در برخی از عکسها، کم ضررتر از دست دادن یک خط از متن است.

برای پیاده‌سازی یک سیاست هوشمند برای حذف بسته‌ها، برنامه‌های کاربردی باید ردهٔ اولویت مورد نظر را در بسته‌های خود مشخص کنند تا میزان اهمیت آنها مشخص گردد. اگر چنین کاری انجام شده باشد زمانی که مسیریاب مجبور به حذف بسته‌ها می‌شود می‌تواند از بسته‌های با اولویت پایین آغاز کند و بعداً به سراغ رده‌های بالاتر برود. البته فقط در موارد خاصّ می‌توان بسته‌هایی را در بالاترین ردهٔ اولویت علامتگذاری کرد.

البته چون هر ماشین در ارسال بسته‌ها با هر اولویتی، آزادی عمل دارد لذا باید به نحوی در کاربر انگیزه ایجاد کرد تا بسته‌هایش را با اولویت پایین بفرستد؛ **پول** می‌تواند این انگیزه را ایجاد کند؛ اگر برای حمل بسته‌های با اولویت بالا هزینهٔ بیشتری گرفته شود کاربران سعی می‌کنند بی‌مورد از اولویتهای بالا استفاده نکنند. البته فرستنده ممکن است اجازهٔ ارسال بسته‌های با اولویت بالا را در شرایط بار سبک به کاربر بدهد ولیکن با افزایش بار حذف خواهند شد. این موضوع کاربران را تشویق می‌کند از ارسال بی‌مورد بسته‌های با اولویت بالا صرفنظر کنند.

گزینهٔ دیگر آنست که به ماشینهای میزبان اجازه بدهیم گاهی بیش از حد توافق شده در هنگام ایجاد مدار مجازی، ارسال داشته باشند ولیکن مشروط بدانکه ترافیک اضافی اولویت پایینی داشته باشد و به محض آشکار شدن علائم ازدحام حذف شود. چنین روشی ایدهٔ بدی نیست چراکه از منابع آزاد سیستم استفادهٔ مفید می‌شود: به ماشینهای میزبان اجازه داده‌ایم مادامی که کس دیگر به منابع آزاد نیاز نداشته باشد از آن استفاده کنند ولیکن در شرایط دشوار [بار بالا] حقی برای کسی ایجاد نکرده‌ایم.

تشخیص زودهنگام (Random Early Detection)

بر هر کسی روشن است که رفع ازدحام به محض کشف علائم آن بسیار کارآمدتر از آن است که بگذاریم اوضاع را بهم بریزد و سپس به رفع آن اقدام کنیم. چنین تجربه‌ای بدین ایده منتهی می‌شود که قبل از اشباع شدن کل فضای بافر، مسیریاب به حذف برخی از بسته‌ها اقدام کند. الگوریتم رایجی که در این خصوص کاربرد دارد اصطلاحاً الگوریتم RED (Random Early Detection) نامیده می‌شود. در برخی از پروتکلهای لایهٔ انتقال (مثل TCP) اگر بسته‌ها در حین انتقال از بین بروند مبداء ، نرخ ارسال بسته‌هایش را کاهش می‌دهد. استدلالی که در پشت این منطق نهفته است آن است که TCP در اصل برای شبکه‌های سیمی طراحی شده و از آنجایی که شبکه‌های سیمی بسیار قابل‌اعتماد و کم خطا هستند لذا دلیل از دست رفتن بسته‌ها اغلب ناشی از پر شدن بافر است تا خطاهای انتقال. از این حقیقت می‌توان برای کمک به کاهش ازدحام بهره گرفت.

دلیل آنکه اجازه می‌دهیم مسیریاب قبل از آنکه وضعیت وخیم شود بسته‌ها را حذف کند آن است که بتوان قبل از دیر شدن کاری انجام داد. برای تعیین زمان شروع حذف بسته‌ها، هر مسیریاب میانگین طول صفهای خود را نگه می‌دارد. هر گاه طول متوسط صف بر روی برخی از خطوط، از حد مجاز تجاوز کرد آن خط با ازدحام مواجه شده و عملیات حذف شروع می‌شود.

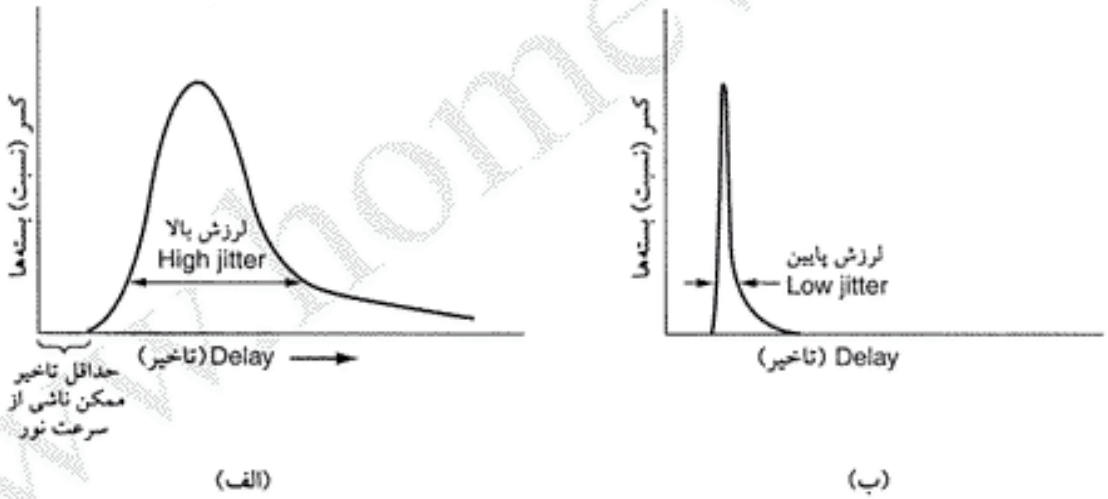
از آنجا که مسیریاب احتمالاً نمی‌تواند بفهمد کدام مبداء باعث مشکل بیشتری شده لذا این ایده که یکی از بسته‌ها را تصادفاً از صف دچار ازدحام بیرون بکشد، می‌تواند ایدهٔ خوبی باشد.

مسیریاب چگونه می‌تواند بروز مشکل را به مبداء بسته اعلام کند؟ یک روش آن است که یک «بستهٔ دعوت به آرامش» برای مبداء بسته ارسال شود. اشکال این روش در آن است که بار بیشتری را بر روی شبکه‌ای که از قبل دچار ازدحام شده، تحمیل می‌کند. راهکار دیگر آن است که بستهٔ انتخاب شده را حذف کرده و هیچ گزارشی هم اعلام نکند. در این صورت مبداء بسته از عدم برگشت بستهٔ اعلام وصول (Ack) متوجه حذف بسته شده و اقدام

لازم را صورت می‌دهد. از آنجایی که مبداء بسته می‌داند که از بین رفتن بسته‌ها ناشی از ازدحام و حذف بسته‌هاست لذا بجای آنکه سخت‌تر تلاش کند، سرعت خود را کاهش می‌دهد. این «فیدبک ضمنی» صرفاً زمانی کار می‌کند که مبداء بسته وقتی متوجه از دست رفتن آن شود سرعت خود را کاهش بدهد. در شبکه‌های بی‌سیم که از بین رفتن بسته‌ها بیشتر ناشی از نویز لینک رادیویی است نمی‌توان از این روش استفاده کرد.

۶-۳-۵ کنترل لرزش (Jitter Control)

برای کاربردهایی مثل ارسال جریان داده‌های صدا و تصویر (ویدئو)، این که آیا ۲۰ میلی‌ثانیه طول می‌کشد تا بسته‌ای تحویل مقصد شود یا ۳۰ میلی‌ثانیه، موضوع مهمی نیست بلکه مهم آنست که این زمان انتقال ثابت باشد. میزان تغییر در زمان رسیدن بسته‌ها (یا به عبارتی انحراف معیار زمان تحویل بسته‌ها) اصطلاحاً **«لرزش»** (Jitter) نامیده می‌شود. وقتی «میزان لرزش» بالاست یعنی مثلاً وقتی که برخی از بسته‌ها در ۲۰ میلی‌ثانیه و برخی دیگردر ۳۰ میلی‌ثانیه به مقصد می‌رسند، کیفیت ارسال صدا و تصویر ویدیوئی را کاهش می‌دهد. نمودار «لرزش» در شکل ۵-۲۹ به تصویر کشیده شده است. در این مثال، هر گاه ۹۹ درصد از بسته‌ها تأخیری بین ۲۴/۵ تا ۲۵/۵ میلی‌ثانیه داشته باشند، برای ارسال صدا و تصویر احتمالاً مناسب و قابل قبول خواهد بود. البته باید بازهٔ تغییرات تأخیر، قابل تحقق و امکان‌پذیر باشد. در ضمن باید تأخیر ناشی از انتقال سیگنال با سرعت نور و همچنین حداقل تأخیر عبور از مسیریابها را به حساب آورد ولیکن می‌توان برخی از تأخیرهای کوچک و غیرقابل پیش‌بینی را نادیده گرفت.



شکل ۵-۲۹. (الف) لرزش زیاد (ب) لرزش کم.

«لرزش» را می‌توان با محاسبهٔ میانگین زمان انتقال در هر گام از کلّ مسیر، محدود کرد: وقتی بسته‌ای به یک مسیریاب می‌رسد آن مسیریاب بررسی می‌کند تا ببیند آن بسته به چه مقدار از برنامهٔ زمانی خود جلوتر یا عقب‌تر است. این اطلاعات در درون هر بسته ذخیره شده و در هر گام بهنگام می‌شود. اگر بسته از برنامهٔ زمانی خود جلو باشد آنقدر معطل می‌شود تا به برنامهٔ زمانی خود برگردد ولیکن اگر از برنامه عقب افتاده باشد مسیریاب سعی می‌کند آنرا به سرعت بر روی خروجی بفرستد.

در حقیقت الگوریتمی که تعیین می‌کند کدامیک از بسته‌های منتظر ارسال، برای خروج از یک خط اولویت دارد می‌تواند بسته‌ای را انتخاب کند که بیشتر از بقیه، از برنامهٔ زمانی خود عقب‌افتاده است. بدین ترتیب بسته‌هایی که از برنامهٔ زمانی خود جلوتر هستند آهسته‌تر و بسته‌هایی که از برنامهٔ زمانی خود عقب افتاده‌اند سریعتر هدایت می‌شوند و در هر دو حالت میزان لرزش کاهش خواهد یافت.

در برخی از کاربردها مثل ارسال «ویدئو برحسب تقاضا» (Video on-Demand)، خود گیرنده می‌تواند با بافر

کردن بسته‌ها و استخراج آنها از بافر جهت نمایش [در زمان مناسب] میزان لرزش را کاهش بدهد (بجای آنکه این کار به صورت بی‌درنگ به شبکه محول شود). ولیکن در برخی دیگر از کاربردها خصوصاً زمانی که نیاز به محاورهٔ بی‌درنگ بین مردم باشد (مثل تلفن اینترنتی یا کنفرانسهای ویدیویی از راه دور)، تأخیر ناشی از بافر کردن داده‌ها ، قابل قبول نخواهد بود.

روشهای کنترل ازدحام، حوزه‌ای فعّال برای پژوهش و تحقیق است. پژوهشهای تازه و دستاوردهایی که در این زمینه حاصل شده در مرجع (Gevros et al., 2001) جمع‌بندی و ارائه شده است.

۵-۴ کیفیت خدمات (Quality of Service)

تکنیکهایی که در بخشهای قبلی مرور کردیم به منظور کاهش ازدحام و بهبود کارآیی شبکه طراحی شده بودند. با این حال با رشد شبکه‌های چندرسانه‌ای ، این ارزیابی‌های خاصّ و تمهیدات موردی کفایت نمی‌کند و به تلاشی جدی در تضمین کیفیت خدمات شبکه و طراحی پروتکل ویژه نیاز است. در بخشهای آتی نیز مطالعات خود را در خصوص کارآیی شبکه، ادامه خواهیم داد با این تفاوت که تمرکز ویژه‌ای بر روی روشهای «تضمین کیفیت خدمات» متناسب با نیاز برنامه‌های کاربرد ، خواهیم داشت. ولی در همین ابتدا باید متذکر شویم که بسیاری از این ایده‌ها در حال تغییر هستند و عوض خواهند شد.

۵-۴-۱ نیازها

به دنباله‌ای از بسته‌ها که از مبداء به سوی مقصد روانه می‌شوند اصطلاحاً «**جریان**» (Flow) گفته می‌شود. درشبکه‌های اتصال‌گرا تمام بسته‌هایی که به یک «جریان» تعلق دارند، مسیر مشابهی را طی می‌کنند در حالی که در شبکه‌های بدون‌اتصال (Connectionless) بسته‌ها ممکن است از مسیرهای متفاوتی حرکت کنند. نیازهای مطلوب برای هر «جریان» را می‌توان با چهار پارامتر ابتدایی مشخص کرد: (۱) **قابلیت اطمینان** (۲) **تأخیر** (۳) **لرزش** (Jitter) (۴) **پهنای باند** . مجموعهٔ این چهار پارامتر، «کیفیت خدمات» یا به اختصار QoS مورد نیاز یک «جریان» را تعیین می‌کند. در شکل ۵-۳۰ برخی از کاربردهای رایج و سطح نیازمندیهای آنها فهرست شده‌اند.

نوع کاربرد	قابلیت اطمینان	تاخیر	لرزش	پهنای باند
پست الکترونیکی	بالا	پایین	پایین	پایین
انتقال فایل	بالا	پایین	پایین	متوسط
دسترسی به وب	بالا	متوسط	پایین	متوسط
ورود به سیستم از راه دور	بالا	متوسط	متوسط	پایین
دریافت صوت برحسب تقاضا	پایین	پایین	بالا	متوسط
دریافت ویدیو برحسب تقاضا	پایین	پایین	بالا	بالا
تلفن اینترنتی	پایین	بالا	بالا	پایین
ویدیو کنفرانس	پایین	بالا	بالا	بالا

شکل ۵-۳۰. سطح نیازمندی برنامه‌های کاربردی به کیفیت خدمات .

چهار کاربرد اول نیاز شدیدی به «قابلیت اطمینان» (Reliability) دارند یعنی هیچ یک از بیتها نباید اشتباه تحویل شوند. [به عبارت دیگر حتی یک بیت خطا در کل جریان داده‌ها قابل قبول نیست.] با اضافه کردن کدهای کشف خطا به هر بسته و بررسی آنها در مقصد می‌توان به این هدف نائل شد. اگر بسته‌ای در حین انتقال آسیب دید، وصول آن اعلام و تصدیق نمی‌شود و طبعاً باید از نو ارسال شود. این استراتژی قابلیت اطمینان بالایی برای بسته‌ها

فراهم می‌کند. چهار کاربرد آخر (صدا / ویدیو) می‌توانند اندکی خطا را تحمل کنند لذاکدهای کشف خطا برای هر بسته محاسبه و بررسی نخواهد شد.

کاربردهای انتقال فایل، شامل پست الکترونیکی یا دریافت تصاویر، حساسیت چندانی به تأخیر ندارند. حتی اگر تمام بسته‌ها به طور یکنواخت تا چند ثانیه هم تأخیر داشته باشند هیچ مشکل حادی پدید نمی‌آید. کاربردهای محاوره‌ای (Interactive) همانند جستجو در وب یا ورود به سیستم از راه دور (Remote Login) حساسیت بیشتری به تأخیر دارند.

کاربردهای بی‌درنگ مثل تلفن یا کنفرانس از راه دور حساسیت شدیدی به تأخیر دارند. اگر کلمات ادا شده در یک تماس تلفنی همگی با 2.000 ثانیه تأخیر برسند. کاربران چنین تماسی را نامناسب و غیرقابل قبول خواهند دانست. برعکس، اجرای فایلهای صدا یا ویدیو که بر روی یک سرویس‌دهنده ذخیره شده به تأخیر پایین نیاز ندارند.[۱]

سه کاربرد اول حساسیت چندانی به رسیدن بسته‌ها با فاصلهٔ زمانی نامشخص (نسبت به یکدیگر) ندارند؛ به عبارت بهتر لرزش بالا (High Jitter) مشکل چندانی برای این کاربردها ایجاد نمی‌کند. کاربرد چهارم یعنی ورود به سیستم از راه دور (Remote Login) تا حدودی به لرزش حساستر هستند چراکه در اثر لرزش زیاد ممکن است کاراکترها به صورت ناگهانی و نامتعارف بر روی صفحه نمایش ظاهر شوند. تصاویر ویدیویی و خصوصاً صدا شدیداً به «لرزش» حساس هستند. همانگونه که اشاره شد کاربری که در حال تماشای یک نمایش ویدیویی غیرزنده از روی شبکه است و فریمهای تصویر، همگی با 2.000 ثانیه تأخیر می‌رسند مشکلی پدید نخواهد آمد در حالی که اگر زمان انتقال بین ۱ تا ۲ ثانیه متغیر باشد، مشکلات جدی ایجاد خواهد شد. برای «صدا» لرزش چند میلی‌ثانیه‌ای نیز بوضوح شنیده خواهد شد.

در آخر، کاربردها نیاز متفاوتی به پهنای باند دارند: پست الکترونیکی و Remote Login به پهنای باند زیادی نیاز ندارند در حالی که ارسال ویدیو در تمام اَشکال [فشرده شده یا غیرفشرده] پهنای باند بسیار زیادی را می‌طلبد.

شبکه‌های ATM ، «جریان» (Flow) را براساس QoS مورد نیاز در چهار ردهٔ وسیع دسته‌بندی کرده است:

۱. ارسال با نرخ ثابت (مثل تلفن از طریق شبکه)
۲. ارسال بی‌درنگ با نرخ متغیر (مثل کنفرانس ویدیویی فشرده شده)
۳. ارسال غیربی‌درنگ با نرخ متغیر (مثل تماشای نمایش غیرزنده از طریق اینترنت)
۴. ارسال با هر نرخ ممکن (برای انتقال فایل) (Available Bit Rate)

این رده‌ها در شبکه‌های دیگر و اهداف دیگر نیز مفید و راهگشا هستند. ردهٔ «ارسال با نرخ ثابت» تلاشی است در جهت شبیه‌سازی یک اتصال سیمی با پهنای باند ثابت و تاخیر یکنواخت.

«ارسال با نرخ متغیر» زمانی مفید خواهد بود که تصاویر ویدیویی، فشرده شده باشند و ضریب فشرده سازی فریمهای تصویر با هم فرق کند. بنابراین ارسال یک فریم تصویر با جزئیات و ظرافت زیاد نیاز به ارسال تعداد بیت زیادی دارد در حالیکه ارسال یک فریم تصویر که از یک دیوار سفید گرفته شده ممکن است تا حد بسیار بالایی فشرده شود. ردهٔ «ارسال با هر نرخ ممکن» برای کاربردهایی مثل پست الکترونیکی کاربرد دارد که به تأخیر یا لرزش حساس نیستند.

۵-۴-۲ راهکارهای دستیابی به کیفیت خوب خدمات

تا اینجا چیزهایی را در خصوص نیازهای QoS آموخته‌ایم؛ سؤال این است که چگونه به آنها برسیم؟ در همین جا

---

۱. چراکه مشاهدهٔ یک فیلم یا شنیدن یک موسیقی غیرزنده با چند ثانیه تأخیر دائم مشکل حادی ایجاد نمی‌کند. -م

باید اذعان داشت که هیچ راهکار واحد و جادویی برای نیل به این اهداف وجود ندارد. به عبارت دیگر روشی واحد که تمام نیازهای QoS را به صورت بهینه برآورده نماید موجود نیست بلکه راهکارهای گوناگونی تعریف شده که در آنها برای ارائه راه حلهای عملی، ترکیبی از چندین روش مختلف بکار گرفته شده است. حال به بررسی برخی از این ر اهکارها که طراحان سیستم برای تأمین کیفیت خدمات از آنها بهره گرفته‌اند می‌پردازیم.

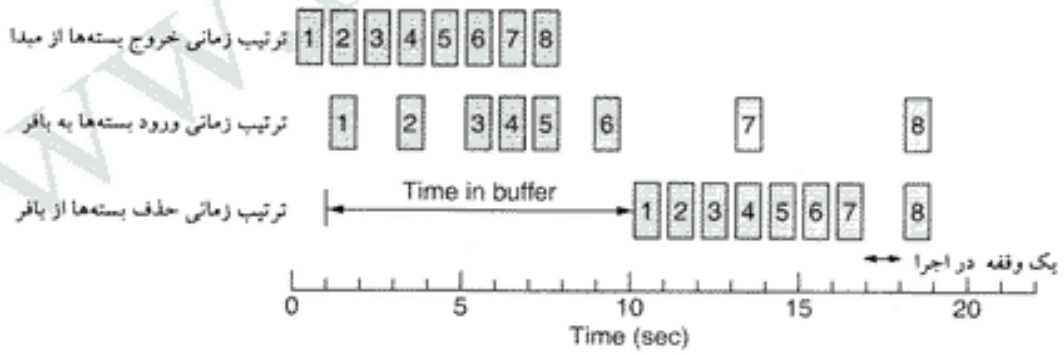
تمهیدات کافی

یک راه حل ساده آن است که برای مسیریاب ظرفیت پردازش، فضای بافر و پهنای باند زیادتری تدارک دیده شود تا بسته‌ها بتوانند بسادگی و بسرعت از آنها بگذرند. اشکال این راه حل آن است که گران تمام می‌شود! به مرور زمان، طراحان ایده‌های بهتری در خصوص پیش‌بینی میزان منابع مورد نیاز مطرح می‌کنند؛ در آن زمان شاید بتوان از این روش در عمل استفاده کرد. از این دیدگاه سیستم تلفن، سیستمی با تمهیدات کافی است و به ندرت گوشی تلفن را برمی‌دارید و فوراً بوق آزاد آنرا نمی‌شنوید زیرا ظرفیت این سیستم، بیش از حد معمولی تقاضا در نظر گرفته شده است.

بافرکردن

«جریان» داده‌ها می‌تواند در طرف گیرنده و قبل از تحویل به پروسهٔ اصلی، بافر شود. بافر کردن جریان تأثیری در قابلیت اطمینان ندارد، تأخیر را افزایش می‌دهد ولیکن در عوض میزان لرزش را متعادل‌تر می‌کند. برای «صدا و تصویر برحسب تقاضا»[1]، لرزش یک مشکل اساسی است و این روش کمک زیادی به حل آن می‌کند.

تفاوت بین لرزش زیاد و لرزش کم را در شکل ۵-۲۹ مشاهده کردیم. در شکل ۵-۳۱ دنباله‌ای از بسته‌ها را می‌بینید که با «لرزش» قابل توجهی تحویل می‌شوند. [فرض شده این بسته‌ها حاوی صدا یا تصویر هستند.] بستهٔ اول در لحظهٔ t=0 ارسال و در لحظهٔ t=1sec تحویل ماشین گیرنده شده است. بستهٔ دوم با تأخیر بیشتری مواجه شده و ۲ ثانیه در راه بوده تا برسد. به محض آنکه این بسته‌ها از راه می‌رسند در ماشین گیرنده بافر می‌شوند.



شکل ۵-۳۱. یکنواخت کردن استریم خروجی بکمک بافرسازی بسته‌ها.

در t=10sec اجرای آنها (Playback) شروع می‌شود. در این لحظه بسته‌های ۱ تا ۶ بافر شده‌اند فلذا می‌توان در فواصل زمانی مشخص و یکنواخت آنها را از بافر استخراج کرد و اجرای متعادل و مناسبی را انتظار داشت. متأسفانه لحظه‌ای که نوبت به اجرای بسته ۸ فرا می‌رسد به دلیل تأخیر نامتعارف، در بافر موجود نیست و اجرای آن باید تا زمان رسیدن آن متوقف شود که این مسئله یک توقف نامطلوب در اجرای موزیک یا فیلم ایجاد می‌کند. این مشکل را می‌توان با ایجاد تأخیر بیشتر در زمان شروع اجرا کاهش داد ولی در عوض به حجم بافر بزرگتری نیاز

---

۱. Video/Audio On-Damand

است. سایتهای وب تجاری که دارای صدا یا تصویر هستند از برنامه‌هایی برای اجرای صدا و تصویر استفاده می‌کنند که قبل از شروع اجرا، داده‌ها را تا ۱۰ ثانیه بافر می‌کنند.

شکل‌دهی ترافیک (Traffic Shaping)

در مثال بالا ماشین مبداء، بسته‌ها را در فواصل زمانی یکنواخت بیرون می‌فرستد، در حالی که مواردی وجود دارد که بسته‌ها بطور نامنظم تولید و ارسال می‌شوند و این مسئله ممکن است منجر به بروز ازدحام در شبکه شود. خروجی غیریکنواخت عموماً وقتی است که سرویس‌دهندهٔ مربوطه، بطور همزمان درگیر ارسال چندین «جریان ویدیویی یا صدا» است و در ضمن امکان عملیات دیگری مثل «جلو و عقب رفتن سریع» (Fast Forward/Rewind) در اجرای صدا و تصویر) و همچنین احراز هویت کاربران و نظایر آنرا نیز فراهم آورده است. از طرفی در برخی از کاربردها مثل کنفرانس از راه دور، بافر کردن داده‌ها بطور کلّ منطقی نیست.[1] با این حال اگر بتوان کاری انجام داد تا سرویس‌دهنده (و کلاً ماشینهای میزبان) مجبور به ارسال با نرخ یکنواخت باشند، کیفیت خدمات (QoS) بهتر خواهد بود. در اینجا به بررسی روشی با عنوان «**شکل‌دهی به ترافیک**» می‌پردازیم که بجای آنکه بر ماشین گیرنده تمرکز داشته باشد، ترافیک تولیدی در سمت سرویس دهنده را متعادل و یکنواخت می‌کند.

«شکل‌دهی به ترافیک» در خصوص منظم کردن نرخ متوسط (و کاهش حالت انفجارگونه) تولید داده‌های ارسالی است. «پروتکل پنجرهٔ لغزان» (Sliding Window Protocol) که قبلاً مطالعه کردیم فقط میزان داده‌هایی را که بطور همزمان ارسال می‌شوند، محدود می‌کند ولی متوسط نرخ ارسال آنها را محدود نمی‌کند. در روش شکل‌دهی به ترافیک، وقتی یک «اتصال» تنظیم می‌شود، کاربر و زیرشبکه (یا به عبارتی مشتری و حامل) بر روی الگوی خاصی از ترافیک (شکل ارسال ترافیک) برای آن مدار توافق می‌کنند. گاهی به این توافق، اصطلاحاً «توافق بر روی سطح خدمات» (Service Level Agreement) گفته می‌شود. مادامیکه مشتری بر طبق این توافق عمل کند و بسته‌ها را براساس قرارداد مورد توافق ارسال نماید، حامل (یعنی زیرشبکه) نیز متعهد است که بسته‌ها را بطور منظم تحویل بدهد. این عمل ازدحام را کاهش داده و اجازه می‌دهد زیرشبکهٔ حامل بتواند برای عمل به تعهدات خود به فعالیت طبیعی ادامه بدهد. چنین توافقی برای عملیاتی مثل انتقال فایل چندان مهم نیست بلکه برای داده‌های بی‌درنگ مثل ارسال صدا و تصویر که شدیداً به کیفیت خدمات نیازمند است، اهمیت حیاتی دارد.

در نتیجه برای عملیات شکل‌دهی به جریان، مشتری به حامل می‌گوید: «الگوی ارسال من بدین نحو است؛ آیا از عهدهٔ آن بر می‌آیی؟» اگر زیرشبکه حامل این الگو را پذیرفت، مورد دیگری به میان خواهد آمد: حامل چگونه متوجه می‌شود که آیا مشتری از مفاد توافق پیروی می‌کند؟ و اگر پیروی نکرد چه عکس‌العملی باید نشان بدهد؟

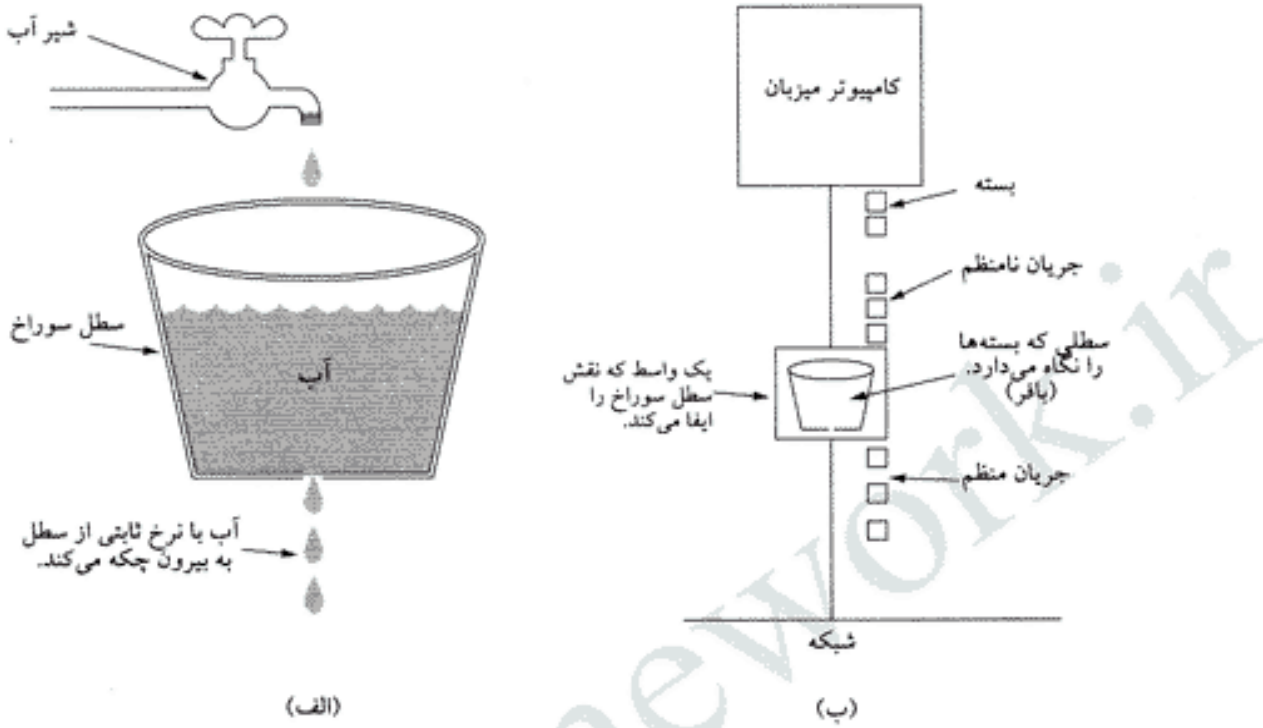
نظارت بر جریان ترافیک اصطلاحاً «**اِعمال سیاست بر ترافیک**» (Traffic Policing) نامیده می‌شود. توافق در خصوص شکل ترافیک و نظارتهای بعدی، در زیرشبکه‌های مبتنی بر مدار مجازی ساده‌تر از زیرشبکه‌های مبتنی بر دیتاگرام است. با این حال حتی در زیرشبکه‌های دیتاگرام، از همین ایده‌ها می‌توان برای اتصالات ایجاد شده در لایهٔ انتقال استفاده کرد.

الگوریتم سطل سوراخ (Leaky Bucket Algorithm)

سطحی را در نظر بگیرید که همانند شکل ۵-۳۲-الف رخنهٔ کوچکی در کف آن وجود دارد. بدون توجه به نرخ ورود آب به این سطل، جریان نشتی از آن ثابت است: مادامیکه آب در سطل وجود دارد نرخ خروجی معادل ρ و

---

۱. بافر کردن مثلاً ۱۰ ثانیه از یک کنفرانس ویدیویی آنرا از حالت زنده خارج خواهد کرد و بهیچوجه مطلوب کاربران نیست. -م



شکل ۵-۳۲. (الف) یک سطل سوراخ‌دار آب (ب) یک سطل سوراخ‌دار «بسته».

وقتی سطل خالی است معادل صفر است. همچنین اگر آب اضافی به سطل پر وارد شود، از اطراف آن بیرون ریخته و هدر می‌رود. (یعنی در ظرفی دیگری که زیر این سوراخ قرار گرفته وارد نخواهد شد.)

به نحوی که در شکل ۵-۳۲-ب نشان داده شده از همین ایده می‌توان برای بسته‌ها نیز استفاده کرد. بطور مفهومی هر ماشین میزبان از طریق یک کارت واسط به شبکه متصل شده که این کارت واسط دارای سطلی سوراخ (یعنی یک صفّ داخلی بی‌نهایت) است. اگر بسته‌ای بخواهد در حالی که صفّ پر است وارد آن شود، حذف خواهد شد. به عبارت دیگر هرگاه یک یا چند پروسه در ماشین میزبان سعی در ارسال بسته‌ای نمایند در حالی که تعداد بسته‌های صفّ به حداکثر ممکن رسیده باشد، بستهٔ جدید بدون هیچ تشریفاتی حذف خواهد شد. این عملیات می‌تواند توسط سخت‌افزار واسط انجام شود یا آنکه توسط سیستم عامل ماشین میزبان شبیه‌سازی شود. این الگوریتم برای اولین بار توسط Turner (۱۹۸۶) پیشنهاد و الگوریتم سطل سوراخ نام گرفت. در حقیقت این روش چیزی نیست مگر یک سیستم صف‌بندی با یک سرویس‌دهندهٔ واحد که سرویس دهی به عناصر منتظر در صفّ با نرخ ثابتی انجام می‌شود.

ماشین میزبان اجازه دارد در هر **تیک ساعت** یک بسته بر روی شبکه قرار بدهد. این کار نیز می‌تواند توسط کارت واسط شبکه یا سیستم عامل ماشین انجام و مدیریت شود. این مکانیزم جریانی نامنظم و بی‌قاعده از بسته‌های تولید شده توسط پروسه‌های کاربری ماشین را به جریانی منظم بر روی شبکه تبدیل کرده و بدین ترتیب حالت انفجارگونهٔ ترافیک، معتدلتر شده و احتمال ازدحام کاهش چشمگیری می‌یابد.

اگر بسته‌ها دارای اندازه‌ای ثابت و یکسان باشند (مثل سلولهای ATM) می‌توان طبق توضیح بالا از این الگوریتم بهره گرفت: «**یک سلول در هر تیک ساعت**». ولیکن وقتی اندازهٔ بسته‌ها متغیر باشد، بهتر آن است که در هر تیک ساعت به جای یک بسته، تعداد بایت ثابتی ارسال شود. بدین ترتیب اگر قرار باشد در هر تیک ساعت ۱۰۲۴ بایت ارسال شود، می‌توان یک بستهٔ ۱۰۲۴ بایتی یا دو بستهٔ ۵۱۲ بایتی یا چهار بستهٔ ۲۵۶ بایتی و به همین

ترتیب، ارسال کرد. اگر تعداد بایتهای باقیمانده ناچیز باشد، بستهٔ بعدی باید تا تیک ساعت بعدی منتظر بماند.

پیاده‌سازی الگوریتم سطل سوراخ (نسخهٔ اصلی آن) ساده است. سطل سوراخ متشکل از یک صف متناهی (محدود) است. هر گاه بسته‌ای دریافت شود و فضای کافی در صف وجود داشته باشد به آخر صف ملحق می‌شود و در غیر این صورت حذف می‌شود. در هر تیک ساعت یک بسته ارسال می‌شود (مگر آنکه صف خالی باشد).

«الگوریتم سطل سوراخ مبتنی بر شمارش بایت» نیز تقریباً به همین روش پیاده‌سازی می‌شود. در هر تیک ساعت شمارندهٔ بایت، به مقدار n تنظیم می‌شود. اگر بستهٔ سر صف تعداد بایت کمتری از مقدار فعلی شمارنده داشته باشد، ارسال شده و به تعداد بایتهای بسته از شمارنده کم می‌شود. مادامیکه مقدار شمارنده از بسته‌های موجود در سر صف بیشتر است می‌توان این بسته‌ها را ارسال کرد. وقتی مقدار شمارنده از اندازهٔ بستهٔ بعدی در سر صف کمتر شود، عمل ارسال تا تیک بعدی متوقف می‌شود تا مقدار شمارنده بایت مجدداً به مقدار n تنظیم شده و جریان بسته‌ها ادامه یابد.

به عنوان مثالی از الگوریتم سطل سوراخ، کامپیوتری را در نظر بگیرید که می‌تواند داده‌هایی با سرعت ۲۵ میلیون بایت در ثانیه (معادل 200 Mbps) تولید کند و شبکه نیز با همین سرعت کار می‌کند، ولیکن مسیریاب فقط می‌تواند چنین نرخ داده‌ای را در یک فاصله زمانی محدود بپذیرد (تا زمانی که بافرهایش پر شود). در زمانهای طولانی بهتر آن است که نرخ ارسال از ۲ میلیون بایت در ثانیه تجاوز نکند.[1] حال فرض کنید هر ۴۰ میلی‌ثانیه، داده‌ها در یک حالت انفجارگونه و بصورت توده‌های یک میلیون بایتی ارسال شوند. [یعنی هر ۴۰ میلی‌ثانیه یک تودهٔ یک میلیون بایتی با حداکثر سرعت یعنی ۲۵ میلیون بایت در ثانیه تولید و ارسال می‌شوند.] برای آنکه نرخ متوسط ارسال به 2MByte/Sec کاهش یابد، باید از «سطلی سوراخ» با ρ=2MByte/sec و فضای بافر 1MByte بهره گرفته شود. این بدین معناست که توده‌های داده‌ای که انفجارگونه ارسال می‌شوند با طول حداکثر 1MByte قابل دریافت و مدیریت هستند و چیزی از دست نخواهد رفت. این سطل نیز در طول ۵۰۰ میلی‌ثانیه تخلیه می‌شود و اینکه داده‌ها با چه سرعتی وارد می‌شوند اهمیتی ندارد.

در شکل ۵-۳۳-الف ورود داده‌ها به سطل سوراخ در خلال ۴۰ میلی‌ثانیه و با نرخ 25 MByte/Sec صورت گرفته است [معادل ۱ مگابایت داده و متناسب با ظرفیت سطل]. در شکل ۵-۳۳-ب می‌بینیم که این مقدار از داده در خلال ۵۰۰ میلی‌ثانیه و با نرخ 2MByte/sec تخلیه و ارسال شده است.

الگوریتم سطل نشانه‌دار (The Token Bucket Algorithm)

الگوریتم سطل سوراخ الگوی خروجی سختگیرانه و با نرخ میانگین و ثابتی را تحمیل می‌کند و آنکه ترافیک تا چه اندازه انفجاری است برایش مهم نیست.[2] در بسیاری از کاربردها وقتی توده‌ای انفجاری از داده‌ها می‌رسد بهتر آن است که اجازه بدهیم سرعت تا حدّی افزایش یابد؛ بنابراین به الگوریتمی احتیاج داریم که قابلیت انعطاف بیشتری داشته باشد و ترجیحاً هیچ داده‌ای از دست نرود. یکی از این الگوریتمها **«الگوریتم سطل نشانه‌دار»** (Token Bucket Algorithm) است. در این الگوریتم، سطل سوراخ (یا بعبارتی بافر موجود در کارت شبکه یا

---

۱. یعنی اگرچه ظرفیت ارسال شبکه 25 Mbyte/Sec است ولی مسیریاب بدلیل محدودیت سرعت، فقط از عهدهٔ پردازش 2Mbyte/Sec برمی‌آید و در غیر اینصورت با ازدحام مواجه می‌شود. اگر فرستنده در لحظاتی با نرخ بیشتری ارسال کرد در عوض باید برای لحظاتی متوقف شود. -م

۲. یادآوری می‌کنیم که انفجاری بودن ترافیک بدین معناست که فقط در لحظات کوتاهی حجم انبوهی اطلاعات تولید و ارسال می‌شود و در بخش زیادی از زمان ارسال داده‌ها ناچیز است. فاکتور انفجاری بودن ترافیک را «حاصل تقسیم ماکزیمم ترافیک بر میانگین ترافیک» در نظر بگیرید. -م

سیستم عامل)، نشانه‌ای (اصطلاحاً یک **توکن**[1]) را نگه می‌دارد که این توکن در هر ΔT ثانیه یکبار تولید می‌شود. در شکل ۵-۳۴-الف ، سطلی را مشاهده می‌کنید که در آن سه نشانه (توکن) وجود دارد و همچنین پنج بسته منتظر ارسال هستند. هر گاه بسته‌ای بخواهد ارسال شود باید ابتدا یک نشانه بدست آورده و آن را از بین ببرد. در شکل ۵-۳۴-ب می‌بینیم که سه بسته از مجموع پنج بسته جواز خروج گرفته و ارسال شده‌اند در حالی که دو بستهٔ باقیمانده در انتظار تولید دو نشانهٔ دیگر به سر می‌برند.

نوع و مکانیزم شکل‌دهی به ترافیک در «الگوریتم سطل نشانه‌دار» نسبت به «الگوریتم سطل سوراخ» متفاوت است. الگوریتم سطل سوراخ به ماشینهای بیکار اجازه نمی‌دهد که وقتی بیکار هستند سهمیهٔ ارسال خود را نگه داشته و از این سهمیهٔ به یکباره جهت ارسال انفجارگونهٔ بسته‌ها استفاده کنند. در طرف مقابل، الگوریتم سطل نشانه‌دار اجازه می‌دهد که ماشین سهمیهٔ خود را تا حداکثر حجم سطل (n بسته) پس‌انداز کنند. این ویژگی بدین معناست که توده‌ای انفجاری از بسته‌ها را (تا حداکثر n بسته) می‌توان یکجا فرستاد. [البته به شرط پس‌انداز تعداد n نشانه]. این ویژگی اجازه می‌دهد که خروجی بتواند تا حدودی حالت انفجارگونه داشته باشد و هنگامی که ورود بسته‌ها انفجاری است، پاسخ سریعتری داده شود.

تفاوت دیگر این دو الگوریتم آن است که الگوریتم سطل نشانه‌دار در هنگامی که سطل پر شود نشانه‌ها را دور می‌اندازد (یا به عبارتی ظرفیت ارسال را) در حالی که هرگز بسته‌ها را حذف نمی‌کند. در مقابل، الگوریتم سطل سوراخ به محض پر شدن سطل، بسته‌ها را حذف می‌نماید.

در اینجا یک تغییر کوچک ممکن است و آن اینکه هر نشانه (توکن) مجوز ارسال k بایت تلقی شود نه ارسال یک بسته. در این حالت یک بسته را فقط زمانی می‌توان ارسال کرد که به تعداد کافی نشانه (معادل با اندازهٔ بسته) پس‌انداز شده باشد. نشانه‌هایی که سهمیهٔ ارسال آنها (برحسب بایت) کوچکتر از اندازهٔ بستهٔ فعلی است برای استفادهٔ بعدی نگه داشته می‌شود.

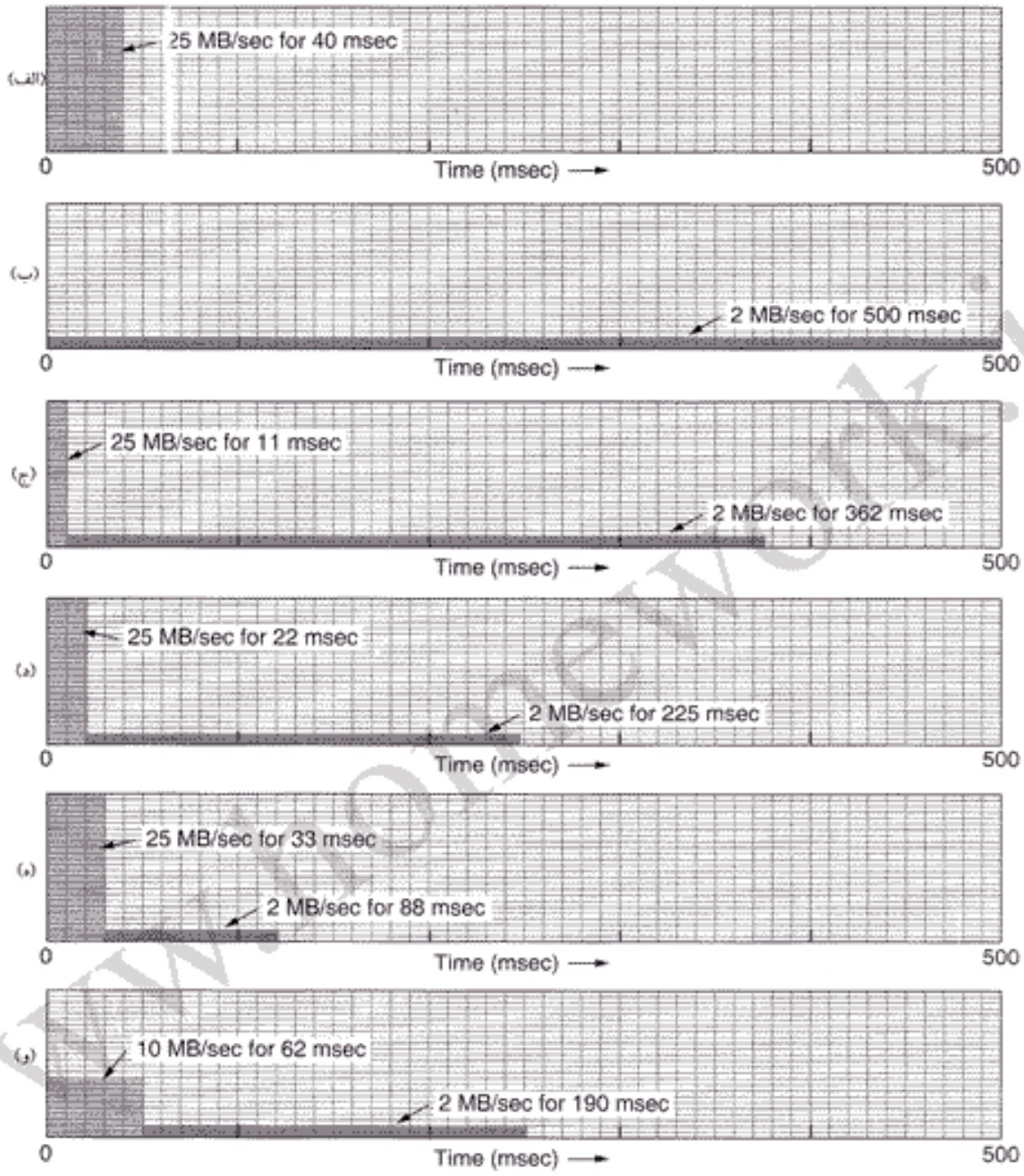
الگوریتم سطل سوراخ و الگوریتم سطل نشانه‌دار را می‌توان به همان نحوی که برای منظم کردن ترافیک خروجی ماشینهای میزبان بکار می‌رود برای متعادل کردن ترافیک بین مسیریابها نیز استفاده کرد. با این حال یک تفاوت بدیهی بین این دو کاربرد وجود دارد و آن هم اینکه الگوریتم سطل نشانه‌دار می‌تواند یک ماشین میزبان را وادار به توقف ارسال نماید، در حالی که وادار کردن یک مسیریاب به توقف (در حالی که بافرهای ورودی آن پر است) ، می‌تواند به از دست رفتن داده‌ها بینجامد.

پیاده‌سازی الگوریتم سطل نشانه‌دار، به سادگی تعریف یک متغیر است تا این متغیر تعداد نشانه‌ها (توکنها) را بشمارد. این شمارنده در هر ΔT ثانیه یک واحد افزایش داده شده و با ارسال یک بسته، کاهش می‌یابد. وقتی این شمارنده به صفر برسد هیچ بسته‌ای را نمی‌توان ارسال کرد. در گونهٔ دیگر این الگوریتم یعنی روش شمارش بایت، شمارنده هر ΔT ثانیه k واحد (k بایت) افزایش یافته و با ارسال بسته به اندازهٔ طول آن از شمارنده کسر می‌شود.

اصولاً کاری که الگوریتم سطل نشانه‌دار انجام می‌دهد آن است که اجازهٔ ارسال انفجارگونه بسته‌ها را صادر می‌کند ولیکن با سقف حداکثر و تنظیم‌شده‌ای که قابل کنترل و اداره باشد. برای مثال به شکل ۵-۳۳-ج دقت کنید. در اینجا یک سطل نشانه دار با ظرفیت ۲۵۰ کیلوبایت در اختیار داریم. نشانه‌ها (توکنها) متناسب با نرخ ۲ مگابایت بر ثانیه تولید می‌شوند. [یعنی نشانه‌ها با نرخی تولید می‌شوند که فقط اجازهٔ ارسال ۲ مگابایت اطلاعات در ثانیه را صادر می‌کنند. -م] فرض کنید هنگامی که یک تودهٔ انفجارگونهٔ ۱ مگابایتی تولید می‌شود، سطل پر است. سطل ، قادر است بمدّت حدود یازده میلی‌ثانیه و با سرعت 25MByte/sec ، داده‌های خود را خالی کند. از آن به بعد

---

۱. نشانه یا «توکن» را در اینجا جواز یا سهمیهٔ ارسال تعداد مشخصی بایت در نظر بگیرید. -م

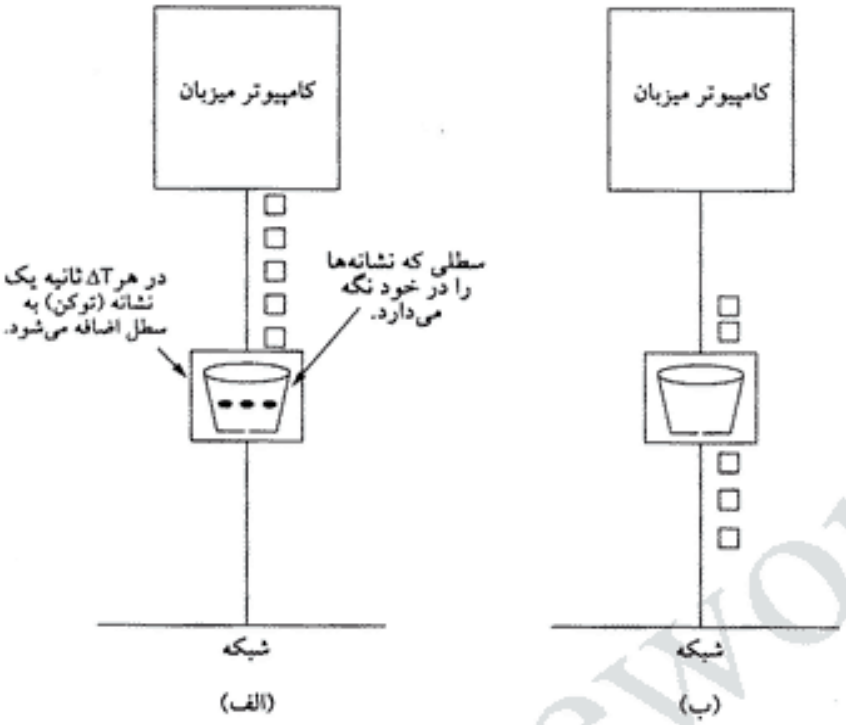


شکل ۵-۳۳. (الف) ورودی به یک سطل سوراخ‌دار. (ب) خروجی از سطل سوراخ‌دار. خروجی از یک سطل نشانه‌دار با ظرفیت: (ج) 250 KByte (د) 500 KByte (ه) 750 KByte. (و) خروجی یک سطل نشانه‌دار با ظرفیت 500KByte به یک سطل سوراخ‌دار با نرخ خروجی 10-Mbps اعمال شده است.

مجبور است تا وقتی که تودهٔ انفجاری داده ارسال نشده سرعت خروجی را بر روی 2MB/sec ثابت نگه دارد.[1] محاسبهٔ طول زمانی که می‌توان بصورت انفجاری و با نرخ حداکثر، تودهٔ تولید شده را ارسال کرد، اندکی پیچیده است. یعنی سقف حداکثر زمان ارسال انفجارگونه، معادل تقسیم «سهمیه موجود در سطل» بر «نرخ ارسال»

---

۱. یادآوری می‌کنیم که در الگوریتم سطل نشانه‌دار اگر بسته‌ها بیش از ظرفیت (سهمیهٔ) موجود در سطل تولید شوند، حذف نخواهند شد.



شکل ۵-۳۴. الگوریتم سطل نشانه‌دار. (الف) قبل از ورود بسته (ب) بعد از خروج بسته.

نیست چراکه در خلال ارسال این توده از بسته‌ها، نشانه‌های جدیدی تولید می‌شوند. اگر طول زمان ارسال انفجاری بسته‌ها را S بنامیم، ظرفیت موجود در سطل نشانه‌دار C بایت، نرخ تولید نشانه ρ بایت بر ثانیه و حداکثر نرخ خروجی M bytes/sec باشد، حداکثر نرخ ارسال انفجارگونهٔ خروجی $C+\rho.S$ بایت است.[1] همچنین می‌دانیم که اگر طول زمان ارسال، S ثانیه و حداکثر نرخ ارسال M بایت بر ثانیه باشد، تعداد بایتی که در این زمان ارسال می‌شود M.S بایت خواهد بود. بنابراین داریم:

$$C + \rho.S = M.S$$

با حل این معادله بدست می‌آوریم:

$$S = \frac{C}{M - \rho}$$

در مثال فوق با پارامترهای C=250 KByte ، ρ=2 MByte/sec و M=25 MByte/sec حداکثر زمان ارسال انفجارگونهٔ بسته‌ها یازده میلی‌ثانیه بدست خواهد آمد. شکلهای ۵-۳۳-د و ۵-۳۳-ه سطل نشانه‌دار را برای ظرفیت ۵۰۰کیلوبایت و ۷۵۰کیلوبایت نشان می‌دهد.

مشکل بالقوهٔ الگوریتم سطل نشانه‌دار آن است که کماکان اجازه می‌دهد توده‌ای از بسته‌ها به صورت انفجارگونه تولید و ارسال شود (حتی اگر طول زمان تولید به دقت و براساس انتخاب صحیح ρ و M تنظیم شده باشد). اغلب مطلوب آن است که نرخ حداکثر تولید بسته‌ها کاهش یابد ولی بدون آنکه بخواهیم به مقدار کم و ثابت

---

۱. یادآوری می‌کنیم که اگر سهمیهٔ موجود در سطل C بایت باشد، می‌توان C بایت از داده‌ها را با نرخ حداکثر و بصورت انفجاری ارسال کرد و پس از آن باید نرخ ارسال را متناسب با نرخ تولید نشانه کاهش داد. در خلال ارسال انفجاری این C بایت که بمدت S ثانیه طول می‌کشد تعدادی نشانهٔ جدید (سهمیهٔ جدید) تولید می‌شود و اجازه می‌دهد حجم ارسال انفجاری اندکی افزایش یابد. این حجم معادل $C+\rho.S$ بایت است: C بایت سهمیهٔ قبلی و $\rho.S$ بایت سهمیهٔ جدید تولید شده در زمان ارسال. -م.

الگوریتم سطل سوراخ بسنده نماییم.

یک روش برای متعادل کردن ترافیک آن است که بعد از «سطل نشانه‌دار» یک «سطل سوراخ» قرار داده شود. نرخ خروجی سطل سوراخ باید بیشتر از نرخ خروجی سطل نشانه‌دار (یعنی مقدار ρ) انتخاب شود ولی این مقدار باید از حداکثر نرخ مجاز شبکه کمتر باشد. شکل ۵-۳۳-و خروجی یک سطل نشانه‌دار با ظرفیت ۵۰۰کیلوبایت که پس از آن یک سطل سوراخ 10MByte/sec قرار داده شده را نشان می‌دهد.

نظارت بر این روشها ممکن است اندکی پیچیده باشد. اصولاً شبکه باید این الگوریتم را شبیه‌سازی کند و مطمئن گردد که بسته یا بایتهای بیشتر از حد مجاز و مشخص ارسال نمی‌شود. در هر حال این ابزارها روشهایی برای شکل‌دهی به ترافیک شبکه به گونه‌ای قابلیت مدیریت است تا بتوان نیازهای QoS (کیفیت خدمات) را برآورده نمود.

رزرو منابع (Resource Reservation)

توانایی شکل‌دهی و تنظیم ترافیک ارسالی، تمهید خوبی برای تضمین «کیفیت خدمات» (QoS) محسوب می‌شود ولیکن استفاده از این روشها زمانی کارآمد خواهد بود که تمام بسته‌ها از مسیر یکسانی عبور کنند. پراکندگی تصادفی بسته‌ها بر روی مسیرهای متفاوت، تضمین هر چیزی را بسیار دشوار می‌کند. بنابراین برای تامین کیفیت خدمات باید بین مبداء و مقصد چیزی شبیه به یک مدار مجازی ایجاد و تنظیم شود و تمام بسته‌های یک «جریان» از این مسیر حرکت کنند.

هر گاه برای جریان داده‌ها، مسیر ویژه داشته باشیم می‌توان منابع لازم را در طول این مسیر، رزرو کرده و موجود بودن ظرفیت مورد نیاز را تضمین کرد. سه نوع متفاوت از منابع را می‌توان از قبل رزرو کرد:

۱. پهنای باند

۲. فضای بافر

۳. سیکلهای CPU [ظرفیت پردازش مورد نیاز]

رزرو پهنای باند آشکارترین مورد است: اگر یک جریان به 1Mbps پهنای باند نیاز داشته باشد و ظرفیت خط خروجی 2Mbps باشد تلاش برای هدایت سه جریان همزمان بر روی این خط عملی نیست. لذا رزرو پهنای باند به معنای آن نیست که می‌توان یک خط خروجی را بیش از ظرفیت آن رزرو کرد.

دومین منبع که اغلب با محدودیت مواجه است، فضای بافر می‌باشد. وقتی یک بسته دریافت می‌شود معمولاً توسط سخت‌افزار در حافظهٔ کارت شبکه ذخیره می‌گردد. بعداً نرم‌افزار مسیریاب باید آن را به یک بافر در فضای RAM اصلی منتقل کرده و پس از پردازش و تعیین مسیر، آنرا در صف مربوط به خط خروجی منتخب، وارد نماید. اگر هیچ بافری موجود نباشد، بستهٔ دریافتی حذف خواهد شد چرا که فضایی برای ذخیرهٔ آن موجود نیست. برای تضمین کیفیت خوب خدمات، باید برای هر «جریان» مشخص مقداری بافر کنار گذاشت تا لازم نباشد بسته‌های آن جریان برای بدست آوردن فضای بافر با دیگر بسته‌ها رقابت کنند. در این صورت به محض نیاز به بافر، فضای لازم در اختیار خواهد بود (البته تا حدّ معقول و مشخص).

در آخر، «سیکلهای CPU» نیز منبع کمیابی است: پردازش یک بسته به اندازهٔ معینی وقت مسیریاب را می‌گیرد لذا هر مسیریاب قادر است در یک ثانیه، تعداد مشخص و محدودی از بسته‌ها را پردازش کند. برای آنکه بتوان مطمئن شد که بسته‌ها در روالی منظم و به موقع پردازش می‌شوند باید مراقب بود که بار CPU بیش از حدّ زیاد نشود.

در نگاه اول شاید به نظر برسد که اگر پردازش هر بسته مثلاً یک میکروثانیه طول بکشد، پس مسیریاب

می‌تواند یک میلیون بسته را در هر ثانیه پردازش و هدایت نماید. چنین نتیجه‌گیری غلط است زیرا به دلیل تغییرات آماری بار [یعنی ثابت نبودن میزان بار]، پردازنده در دوره‌هایی از زمان بیکار می‌ماند. اگر CPU موظف باشد که در هر سیکل، کار پردازش را شروع و تکمیل کند، حتی از دست رفتن چند سیکل (ناشی از بیکاریهای مقطعی) انباشته شده و قابل جبران نخواهد بود.

با این حال حتی وقتی میزان بار کمتر از ظرفیت تئوریک مسیریاب باشد باز هم ممکن است صفّ تشکیل و تأخیر ایجاد شود. وضعیتی را در نظر بگیرید که در آن بسته‌ها بطور کاملاً تصادفی و با نرخ متوسط λ بسته بر ثانیه وارد می‌شوند. زمان مورد نیاز برای پردازش هر بسته نیز تصادفی است و بطور متوسط μ بسته در هر ثانیه پردازش می‌شود. با این فرض که تابع توزیع دریافت بسته‌ها و همچنین پردازش آنها «پوآسون» باشد می‌توان به کمک «نظریهٔ صف» (Queuing Theory) ثابت کرد که متوسط تأخیری که هر بسته با آن مواجه خواهد شد (یعنی T) معادل است با:

$$T = \frac{1}{\mu} \times \frac{1}{1-\lambda/\mu} = \frac{1}{\mu} \times \frac{1}{1-\rho}$$

که در آن ρ معادل است با λ/μ و از آن به عنوان «میزان بهره‌وری CPU» (CPU Utilization) یاد می‌شود. فاکتور $1/\mu$ ، زمان لازم برای پردازش یک بسته است. (هنگامی که رقابت و صف وجود ندارد). فاکتور $1/(1-\rho)$ میزان کاهش سرعت پردازش، در اثر تشکیل صفّ (رقابت) تلقی می‌شود. به عنوان مثال اگر λ =950000 Packet/sec و μ = 1,000,000 Packet/sec باشد میزان بهره‌وری CPU یعنی ρ معادل با ۰/۹۵ است و متوسط تأخیر بسته‌ها بجای ۱ میکروثانیه، ۲۰ میکروثانیه خواهد بود. این زمان در برگیرندهٔ زمان انتظار در صف و زمان سرویس دهی [زمان پردازش] بسته است. اگر مثلاً سی مسیریاب در مسیر جریان بسته‌ها قرار گرفته باشد زمان تأخیر انتظار صفّ در کل مسیر بتنهایی معادل با ۶۰۰ میکروثانیه است.

کنترل پذیرش (Admission Control)

حال در مرحله‌ای هستیم که ترافیک ورودی از یک «جریان» (Flow) خاصّ به خوبی شکل و نظم داده شده و بسته‌ها از یک مسیر واحد حرکت می‌کنند و پیشاپیش ظرفیت مورد نیاز در طول مسیر، پیش‌بینی و رزرو شده است. با چنین فرضی، هر گاه جریانی از بسته‌ها به یک مسیریاب تسلیم شود براساس ظرفیت موجود خود و سطح تعهداتی که در خصوص دیگر جریانها پذیرفته، باید در خصوص قبول یا ردّ آن تصمیم بگیرد.

تصمیم‌گیری در خصوص پذیرش یا ردّ یک «جریان»، یک مقایسه ساده بین میزان پهنای‌باند، بافر و سیکلهای CPU موردنیاز و ظرفیت باقیمانده در مسیریاب نیست. این تصمیم‌گیری اندکی پیچیده‌تر از این مقایسهٔ ساده است. اگرچه برخی از برنامه‌های کاربردی ممکن است میزان نیاز خود به پهنای باند را بدانند ولیکن آگاهی کمی از بافر یا حجم پردازش مورد نیاز بسته‌ها در مسیریابهای واقع بر روی مسیر دارند لذا حداقل می‌توان گفت که به روشی متفاوت برای توصیف و ارزیابی جریان نیاز است. در ثانی، برخی از کاربردها در مقابل برآورده نشدن انتظاراتشان ، تحمل بیشتری از خود نشان می‌دهند. در آخر آنکه ممکن است برخی از کاربردها بخواهند که در خصوص «پارامترهای جریان» چانه‌زنی و مذاکره کنند در حالی که امکان دارد برخی دیگر از کاربردها چنین خصوصیتی نداشته باشند. به عنوان مثال یک برنامهٔ نمایش‌دهندهٔ فیلم که معمولاً ۳۰ فریم تصویر در ثانیه را نمایش می‌دهد ممکن است بخواهد در صورت عدم وجود پهنای باند کافی برای این حجم از داده، تعداد فریمهای تصویر را تا 25 Frame/sec کاهش بدهد. به همین ترتیب ممکن است تعداد نقاط تصویر در هر فریم (Pixel Per Frame)، پهنای باند صدا و دیگر ویژگیها نیز قابل تغییر باشد.

چونکه برای رسیدن به توافق نهایی در خصوص تامین نیازهای یک «جریان»، باید مولفه‌های متعددی در

مذاکرات شرکت داشته باشند (اعمّ از فرستنده، گیرنده و تمام مسیریابهای واقع بر روی مسیر)، لذا هر «جریان» باید برحسب پارامترهای مشخصی بدقت توصیف شود تا بتوان بر روی این پارامترها مذاکره و توافق کرد. مجموعهٔ چنین پارامترهایی اصطلاحاً «**مشخصات توصیفی جریان**» (Flow Specification) نامیده می‌شود. بدین ترتیب یک فرستنده (مثل سرویس‌دهندهٔ ویدیو) مشخصات توصیفی جریان را به صورت پارامترهای پیشنهادی و مورد نظر خود تعریف می‌نماید. این پارامترهای پیشنهادی در طول مسیر منتشر می‌شود و هر مسیریاب واقع بر مسیر آنها را بررسی کرده و در صورت نیاز در آنها تغییراتی ایجاد می‌کند. این تغییرات فقط کاهشی است نه افزایشی (یعنی مثلاً نرخ مورد نظر ارسال داده‌ها را کاهش می‌دهد نه افزایش). وقتی این پارامترها به طرف مقابل برسد، به اجرا گذاشته می‌شوند.

در مثال شکل ۵-۳۵، نمونه‌ای از پارامترهای توصیف جریان، مبتنی بر RFC 2210 و RFC 2211 نشان داده شده است. در اینجا پنج پارامتر وجود دارد که اولین آنها پارامتر Token Bucket Rate (نرخ تولید نشانه در الگوریتم سطل نشانه‌دار) است و تعداد بایتهایی را مشخص می‌کند که در هر ثانیه به سطل وارد می‌شوند. در حقیقت این پارامتر حداکثر نرخ ارسال مُجاز و قابل قبولی است که فرستنده می‌تواند ارسال نماید و میانگین ارسال در یک زمان طولانی تلقی می‌شود.

پارامترهای توصیف جریان	واحد
Token bucket rate	Bytes/sec
Token bucket size	Bytes
Peak data rate	Bytes/sec
Minimum packet size	Bytes
Maximum packet size	Bytes

شکل ۵-۳۵. مثالی از توصیف جریان.

پارامتر دوم، حجم سطل را برحسب بایت مشخص می‌کند. به عنوان مثال اگر پارامتر Token Bucket Rate یک مگابیت در ثانیه و حجم سطل 500KByte باشد حتی در صورت توقف ارسال، این سطل می‌تواند تا چهار ثانیه پر شود و پس از آن سرریز خواهد شد. [سطلی با گنجایش ۵۰۰کیلوبایت معادل ۴ مگابیت، می‌تواند جریانی از داده‌ها با نرخ 1Mbps را بمدت ۴ ثانیه بافر کند. -م]

پارامتر سوم یا Peak Data Rate (حداکثر نرخ ارسال) بیانگر حداکثر نرخ قابل تحمل حتی در بازه‌های زمانی کوتاه است. فرستنده هیچگاه نباید از این نرخ تجاوز نماید.

دو پارامتر آخر مقدار حداقل و حداکثر طول بسته‌ها را مشخص می‌نماید. (این طول شامل سرآیند بستهٔ لایهٔ شبکه و لایهٔ انتقال است.) حداقل طول بسته‌ها نیز اهمیت دارد چراکه پردازش هر بسته (حتی اگر کوچک باشد) مدت زمان ثابتی طول می‌کشد. یک مسیریاب ممکن است آمادگی پذیرش و هدایت ۱۰,۰۰۰ بستهٔ یک کیلوبایتی در هر ثانیه را داشته باشد ولیکن آمادگی پذیرش و هدایت ۱۰۰,۰۰۰ بستهٔ ۵۰بایتی در ثانیه را نداشته باشد، هر چند که مورد دوم از لحاظ حجم داده کمتر از مورد اول است. طول حداکثر هر بسته نیز به دلیل محدودیتهای درونی شبکه اهمیت دارد و نباید از این مقدار حداکثر تجاوز شود. به عنوان مثال اگر بخشی از مسیر از درون شبکهٔ اترنت بگذرد، حداکثر طول بسته نباید بیش از ۱۵۰۰ بایت باشد [به دلیل معماری سخت‌افزار اترنت].

یک سؤال جالب آن است که یک مسیریاب با استفاده از مشخصات توصیفی جریان، چگونه می‌تواند منابع مورد نیاز را رزرو نماید. نگاشت «مشخصات توصیفی جریان» به میزان منابع مورد نیاز، در مقولهٔ پیاده‌سازی

می‌گنجد و استانداردسازی نشده است. فرض کنید که یک مسیریاب قادر به پردازش صد هزار بسته در ثانیه باشد. اگر «جریان» پیشنهادی با نرخ 1MB/sec و مقدار حداقل و حداکثر طول بسته‌ها ۵۱۲ بایت باشد، مسیریاب بدین نتیجه می‌رسد که از این جریان در هر ثانیه ۲۰۴۸ بسته دریافت خواهد کرد و بدین منظور ۲ درصد از CPU خود را برای پردازش این بسته‌ها در نظر می‌گیرد. ترجیحاً این مقدار باید بیشتر هم باشد تا از تأخیرات ناشی از ایجاد صفهای طولانی احتراز شود. اگر سیاستهای یک مسیریاب بر این اساس باشد که هیچگاه بیش از ۵۰ درصد از وقت CPU خود را اختصاص ندهد (یعنی تلویحاً تأخیرات را دو برابر فرض کرده است) و قبلاً ۴۹ درصد از این وقت رزرو شده باشد نمی‌تواند یک «جریان» با مشخصات فوق را بپذیرد [چراکه به دو درصد از CPU احتیاج دارد در حالی که فقط یک درصد باقیمانده است]. برای منابع دیگر [مثل حافظه و پهنای باند] نیز محاسبات و پیش‌بینی‌های مشابه فوق انجام می‌شود.

توصیف دقیقتر «جریان»، برای مسیریابها مفیدتر خواهد بود. اگر در توصیف یک «جریان» بیان شود که پارامتر Token Bucket Rate (نرخ تولید نشانه در الگوریتم سطل) معادل 5MB/sec است ولیکن طول بسته‌ها بین ۵۰ تا ۱۵۰۰ بایت متغیر اعلام شود در این توصیف نادقیق، نرخ ارسال بسته‌ها بین ۳۵۰۰ تا ۱۰۵,۰۰۰ بسته در ثانیه متغیر خواهد بود و ممکن است مسیریاب از عدد ۱۰۵۰۰۰ نگران شده و چنین جریانی را نپذیرد در حالی که اگر مقدار حداقل طول بسته ۱۰۰۰ بایت پیشنهاد شود احتمالاً این جریان پذیرفته خواهد شد.

مسیریابی نسبی

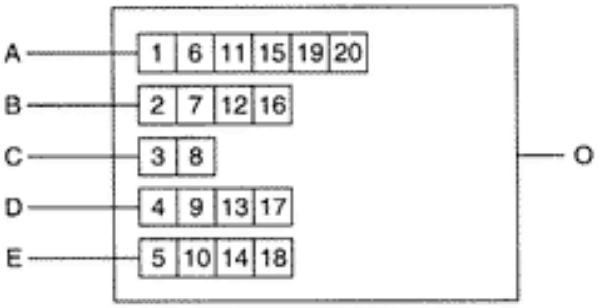
در اغلب الگوریتمهای مسیریابی سعی بر آن است که بهترین مسیر به هر مقصد پیدا شود و پس از آن تمام ترافیک به یک مقصد از مسیر بهینه ارسال خواهد شد. راهکار دیگر برای ارائه کیفیت بهتر خدمات آنست که ترافیک بسته‌های ارسالی برای یک مقصد، از چندین مسیر هدایت و ارسال شود. از آنجایی که مسیریابها دید جامعی از ترافیک سرتاسر شبکه ندارند لذا تنها راه ممکن برای توزیع ترافیک بر روی چندین مسیر، استفاده از اطلاعات موجود و محلی است. یک روش ساده آن است که ترافیک بطور مساوی یا به تناسب ظرفیت هر یک از خطوط خروجی مسیریاب، بر روی آنها توزیع شود. با این حال الگوریتمهای پیچیده‌تری در این خصوص وجود دارد. (Nelakuditi and Zhang, 2002)

زمان بندی بسته‌ها

هر گاه یک مسیریاب هدایت چندین «جریان» را بر عهده داشته باشد این خطر وجود دارد که یک «جریان» از حدود و ظرفیت مجاز خود تجاوز نماید و در نتیجه جریانهای دیگر را با کمبود منابع (starvation) مواجه سازد. اگر پردازش بسته‌ها به ترتیب ورودشان انجام گیرد باعث می‌شود که یک فرستندهٔ متجاوز بتواند بیشتر ظرفیت مسیریابهایی را که بر روی خط سیر بسته‌های او هستند اشغال کرده و کیفیت خدمات دیگران کاهش یابد. برای خنثی کردن چنین تلاشی، الگوریتمهایی جهت زمان‌بندی بسته‌ها پیشنهاد شده است. (Bhatti and Crowcroft, 2000)

یکی از اولین روشها، الگوریتم «صف‌بندی بی‌طرفانه» (Fair Queuing) است. (Nagle, 1987) جوهرهٔ این الگوریتم آنست که مسیریابها باید برای هر خط خروجی و به ازای هر «جریان» که از آن خط خروجی می‌گذرد، صفهای جداگانه‌ای تشکیل بدهند. هر گاه خطی بیکار شود، مسیریاب صفّها را به ترتیب پویش کرده و از سر هر صفّ یکی را بر می‌دارد. بدین ترتیب، در شرایطی که n ماشین میزبان برای یک خط خروجی رقابت می‌کنند، از هر n بستهٔ ارسالی بر روی خط یک بسته به هر ماشین میزبان تعلق می‌گیرد. افزایش نرخ ارسال بسته‌ها، در نسبت سهم هر ماشین تغییری ایجاد نخواهد کرد.

البته در همین ابتدا، الگوریتم فوق دارای یک مشکل است: آن ماشینهای میزبان که طول بسته‌هایشان بزرگ است نسبت به ماشینهایی که بسته‌های کوچک تولید می‌کنند، سهم بیشتری از پهنای باند را بخود اختصاص خواهند داد. پژوهشگری به نام Demer و همکاران او (۱۹۹۰) پیشنهادی جهت بهبود این الگوریتم ارائه دادند. پیشنهاد آن بود که بجای ارسال یک بسته از هر صفّ به صورت نوبت چرخشی (Round Robin)، سهم ارسال هر صفّ بر حسب بایت باشد. در نتیجه صفّها بطور متوالی و بایت به بایت پویش شده و بسته‌های هر صفّ برحسب زمان خاتمهٔ ارسالشان مرتب شده و بهمان ترتیب ارسال می‌شوند. یعنی به جای آنکه از هر صف فقط یک بسته انتخاب شود تعدادی بایت و متناسب با سهم زمانی هر صف ارسال می‌شود. این الگوریتم در شکل ۵-۳۶ به تصویر کشیده شده است.



(الف)

بسته	زمان خاتمه ارسال
C	8
B	16
D	17
E	18
A	20

(ب)

شکل ۵-۳۶. (الف) مسیریابی که در آن پنج بسته برای خروج از خطّ O به صفّ شده‌اند. (ب) زمان خاتمهٔ ارسال این پنج بسته.

در شکل ۵-۳۶-الف بسته‌هایی به طول ۲ تا ۶ بایت می‌بینیم. در هر تیک ساعت (مجازی) اولین بایت از بستهٔ دریافتی از خط A ارسال می‌شود. در تیک بعدی اولین بایت از بستهٔ دریافتی از خط B ارسال می‌گردد و کار به همین ترتیب ادامه می‌یابد. اولین بسته‌ای که ارسال آن پس از هشت تیک ساعت خاتمه خواهد یافت بستهٔ C است.[1] در شکل ۵-۳۶-ب فهرست مرتب شدهٔ بسته‌ها به ترتیب ارسال مشخص شده است. هرگاه بستهٔ جدیدی دریافت نشود بسته‌ها به ترتیب فوق‌الذکر (از C تا A) ارسال خواهد شد.

یک اشکال این الگوریتم آن است که به تمام ماشینهای میزبان، اولویت یکسانی می‌دهد. در بسیاری از محیطها مطلوبتر آن است که به سرویس‌دهنده‌های ویدیو (Video Server) اولویت بیشتری نسبت به یک سرویس‌دهندهٔ معمولی فایل داده شود و در هر تیک ساعت، سهم آن دو یا چند بایت باشد. این الگوریتم اصلاح شده به نام «الگوریتم صف‌بندی بی‌طرفانهٔ وزن‌دار» (Weighted Fair Queuing) مشهور است و کاربرد گسترده‌ای دارد. گاهی اوقات وزن هر صف معادل با تعداد «جریان» (Flow) منشعب از یک ماشین در نظر گرفته می‌شود و بدین ترتیب هر پروسهٔ مولد جریان، پهنای باند یکسانی دریافت می‌دارد.[2] روش پیاده‌سازی مؤثر این الگوریتم در مرجع (Shreedhar & Varghese, 1995) تشریح شده است. امروزه در عمل، هدایت بسته‌ها توسط سخت‌افزار مسیریاب یا سوئیچ انجام می‌شود و الگوریتمهای فوق برروی سخت‌افزاری پیاده‌سازی شده‌اند. (Elhanany et al., 2001)

---

۱. دقت کنید که شماره‌هایی که درون مربعهای هر بسته از شکل ۵-۳۶ نوشته شده شمارهٔ ترتیب ارسال تلقی می‌شود و هر مربع شماره‌دار صرفاً معادل یک بایت است. -م

۲. به عبارت دیگر به جای آنکه به ماشینهای میزبان پهنای باند مساوی داده شود به پروسه‌های هر ماشین پهنای باند یکسان داده می‌شود.-م

۵-۴-۳ خدمات مجتمع (Integrated Services)

در خلال سالهای ۱۹۹۵ تا ۱۹۹۷، تلاش IETF بر آن بود که برای انتقال داده‌های مالتی‌مدیا (Multimedia Streaming) معماری مناسبی ابداع کند. نتیجهٔ کار چندین RFC به شماره‌های ۲۲۰۵ تا ۲۲۱۰ بود. این پروژه با نام کلّی «**الگوریتمهای مبتنی بر جریان**» (Flow-based algorithms) یا «**خدمات مجتمع**» (Integrated Services) شناخته می‌شود و کاربردهای چندپخشی (Multicast) و تک‌پخشی (Unicast) را در بر می‌گیرد. به عنوان یک نمونه از کاربردهای تک‌پخشی، کاربری را در نظر بگیرید که قطعه‌ای ویدیو را از یک سایت تماشا می‌کند. به عنوان مثالی از کاربردهای چندپخشی، ایستگاههای پخش تلویزیون دیجیتال را در نظر بگیرید که برنامه‌های خود را در قالب جریانی از بسته‌های IP به گیرندگان بی‌شمار و پراکندهٔ خود ارسال می‌دارند. در ادامه بر روی کاربردهای چندپخشی متمرکز خواهیم شد چراکه کاربردهای تک‌پخشی حالت خاص چندپخشی هستند.

در اغلب کاربردهای چندپخشی، گروههای مختلف می‌توانند به صورت پویا عضویت خود را تغییر بدهند: مثلاً افرادی را مجسم کنید که در یک کنفرانس ویدیویی وارد می‌شوند و پس از مدتی حوصلهٔ آنها سر می‌رود و به یک کانال پخش آواز می‌پیوندند! در چنین شرایطی روش رزرو پهنای باند به خوبی کار نخواهد کرد چرا که هر فرستنده باید دائماً ورود و خروج اعضای خود را پیگیری نماید. برای سیستمی که مثلاً برای پخش تلویزیونی طراحی شده و میلیونها مشترک دارد این روشها هرگز کار نخواهد کرد!

RSVP[1]: پروتکل رزرو منابع

اصلیترین پروتکل پیشنهاد شده توسط IETF برای ارائه خدمات مجتمع، RSVP نامیده می‌شود. این پروتکل در RFC 2205 تشریح شده است و برای رزرو کردن پهنای باند بکار می‌آید. پروتکلهای دیگری که در RFC 2206 تا RFC 2210 تشریح شده‌اند چگونگی ارسال داده‌ها را توصیف می‌کنند. RSVP اجازه می‌دهد که چندین فرستنده بتواند برای چندین گروه از گیرندگان خود داده بفرستند و همچنین امکان آن را فراهم کرده که گیرندگان بتوانند کانال مورد نظر خود را آزادانه عوض کنند. در عین حال پروتکل RSVP، استفاده از پهنای باند را بهینه‌سازی کرده و از بروز ازدحام جلوگیری می‌کند.

در ساده‌ترین حالت، این پروتکل از روش «مسیریابی چندپخشی مبتنی بر درخت پوشا»[2] که قبلاً تشریح شد، بهره می‌گیرد. به هر گروه یک آدرس یکتا انتساب داده می‌شود و برای ارسال یک بسته به گروه خاصّ، آدرس آن گروه در بسته قرار می‌گیرد. سپس توسط الگوریتم استاندارد مسیریابی چندپخشی، یک درخت پوشا که تمام اعضای آن گروه را در بر می‌گیرد، ایجاد می‌گردد. الگوریتم مسیریابی چندپخشی جزو استاندارد RSVP محسوب نمی‌شود و تنها تفاوت آن با الگوریتم معمولی مسیریابی چندپخشی آنست که بطور متناوب، مقداری اطلاعات اضافی برای هر گروه ارسال می‌شود تا مسیریابهای واقع بر روی درخت ، آنها را در ساختمان دادهٔ خاصّی در حافظهٔ خود ذخیره نمایند.

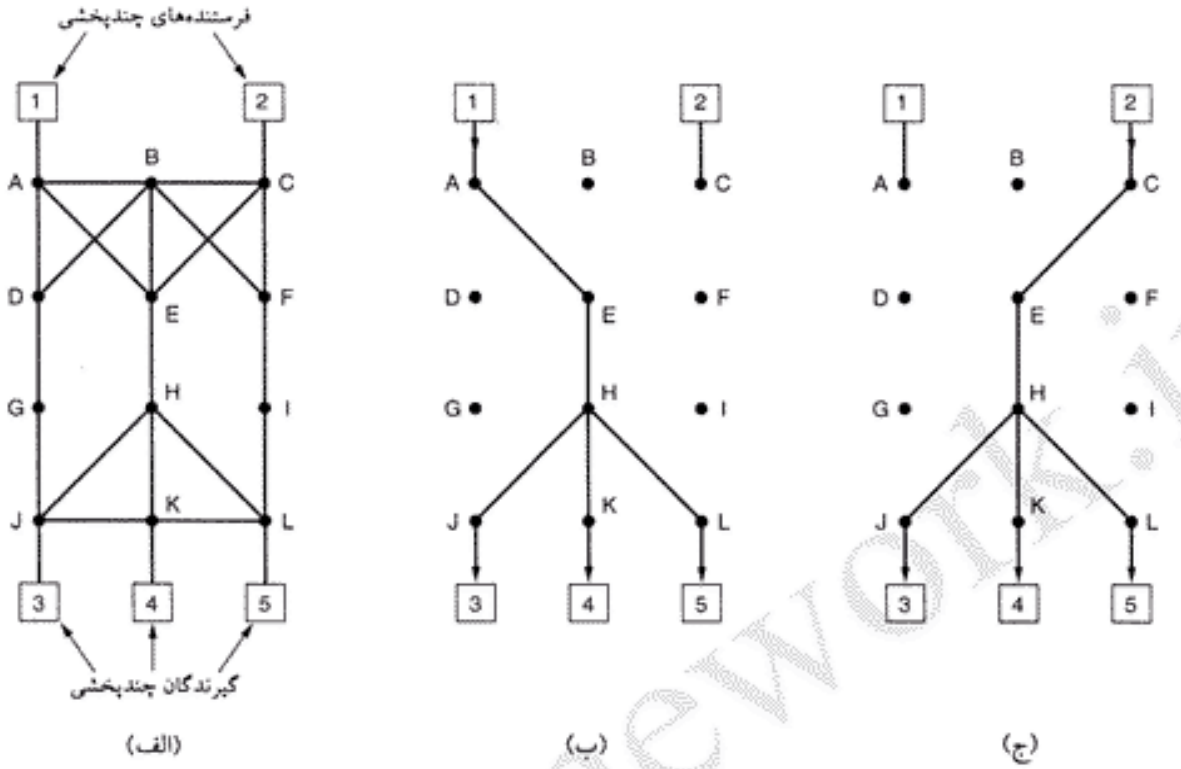
به عنوان مثال شبکهٔ شکل ۵-۳۷-الف را در نظر بگیرید. ماشینهای میزبان ۱ و ۲ فرستنده‌های چندپخشی و ماشینهای ۳ و ۴ و ۵ گیرندگان چندپخشی هستند. در این مثال فرستنده‌ها و گیرنده‌ها کاملاً از هم جدا (Disjoint) هستند ولی در حالت کلی ممکن است مجموعهٔ ماشینهای فرستنده و گیرنده عضو مشترک هم داشته باشند. درختهای چندپخشی برای فرستندهٔ شماره ۱ و شماره ۲ در شکلهای ۵-۳۷-ب و ۵-۳۷-ج نشان داده شده‌اند.

برای دریافت بهتر و جلوگیری از ازدحام، هر یک از گیرندگان یک گروه می‌توانند پیامی برای رزرو پهنای باند

---

۱. Resource Reservation Protocol

۲. Spanning Tree Multicast Routing (به بخش ۵-۲-۸ مراجعه کنید.)



شکل ۵-۳۷. (الف) ساختار یک شبکه (ب) «درخت پوشای چندپخشی» برای ماشین میزبان ۱ (ج) «درخت پوشای چندپخشی» برای ماشین میزبان ۲.

ارسال کنند تا از طریق درخت به فرستنده برسد. این پیام به کمک «الگوریتم هدایت در مسیر معکوس»[1] که قبلاً تشریح شد، در درخت انتشار می‌یابد. در هر گام، مسیریاب به این پیام توجه کرده و پهنای باند لازم را رزرو می‌نماید. اگر پهنای باند کافی موجود نباشد، «پیغام شکست» برخواهد گشت. اگر این پیغام موفقیت‌آمیز به مبداء آن برگردد، پهنای باند لازم در کل مسیر رزرو شده است.

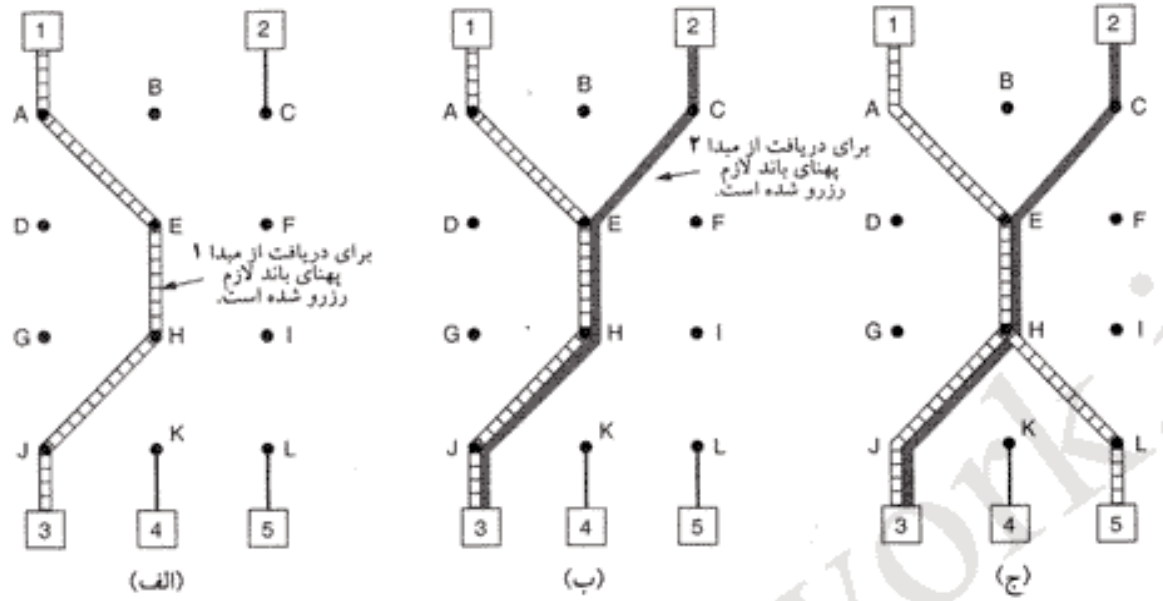
مثالی از چگونگی رزروسازی در شکل ۵-۳۸-الف نشان داده شده است. در اینجا ماشین میزبان ۳ تقاضای برقراری یک کانال با ماشین ۱ را داده است. به محض برقراری چنین کانالی، بسته‌ها می‌توانند بدون مسئلهٔ ازدحام از ۱ به ۳ جاری شوند. حال ببینیم اگر ماشین ۳ بعداً بخواهد کانالی دیگر با ماشین میزبان ۲ برقرار کند چه اتفاقی می‌افتد (زیرا مثلاً می‌خواسته دو کانال تلویزیونی را بطور همزمان تماشا کند.) مسیر دوم مطابق شکل ۵-۳۸-ب رزرو می‌شود. دقت کنید که از ماشین گیرندهٔ ۳ تا مسیریاب E به دو کانال مستقل نیاز است چراکه قرار است دو «استریم ویدیویی مستقل»[2] ارسال شود.

در شکل ۵-۳۸-ج، ماشین میزبان ۵ تصمیم می‌گیرد برنامه‌ای را که توسط ماشین ۱ پخش می‌شود تماشا کند و او نیز رزرواسیون لازم را انجام می‌دهد. ابتدا پهنای باند لازم تا H رزرو می‌شود. از H به بعد، مسیریابها متوجه می‌شوند که از قبل کانالی با ماشین ۱ دارند و از داده‌های آن تغذیه می‌شوند، لذا نیازی به رزرو نیست. دقت کنید که ممکن است ماشینهای ۳ و ۵ به پهنای باند متفاوتی نیاز داشته باشند (چرا که مثلاً ماشین ۳ تصاویر تلویزیونی را سیاه و سفید نگاه می‌کند و نیازی به اطلاعات رنگی ندارد). فلذا ظرفیت رزرو شده باید آنقدر زیاد باشد تا بتواند از گیرنده‌ای را که به بیشترین مقدار پهنای باند احتیاج دارد، برآورده سازد.

---

۱ Reverse Path Forwarding (به بخش ۵-۲-۷ مراجعه کنید.)

۲ Independent Stream



شکل ۵-۳۸. (الف) ماشین میزبان ۳ تقاضای کانالی به ماشین ۱ می‌کند. (ب) ماشین میزبان ۳ تقاضای کانالی دیگر به ماشین ۲ می‌کند. (ج) ماشین میزبان ۵ تقاضای کانالی به ماشین ۱ می‌کند.

در RSVP به هر گیرنده آزادی عمل داده شده تا بتواند با یکبار عملیات رزرو، کانالهایی با بیش از یک فرستنده، ایجاد نماید. همچنین می‌تواند مشخص کند که آیا انتخاب او برای دوره‌ای از زمان ثابت است یا آنکه حق تغییر مبداء ارسال (فرستنده) را برای خود محفوظ نگاه داشته است. مسیریاب می‌تواند به کمک این اطلاعات برآورد بهینه و مناسبی از پهنای باند داشته باشد.

دلیل اتخاذ این استراتژی در محیطهای کاملاً پویا و در حال تغییر آنست که پهنای باند رزرو شده از انتخاب مبداء (فرستنده) مستقل و مجزاست: به محض آنکه یک گیرنده که قبلاً پهنای باند لازم را برای دریافت از یک مبداء خاص رزرو کرده، بخواهد مبداء دریافت خود را تغییر بدهد بخشی از مسیر قبلی برای مبداء جدید نیز معتبر است و نیازی به تخصیص پهنای باند ندارد. در ضمن اگر ماشین ۲ در حال ارسال همزمان چندین جریان ویدیویی (Video Streams) باشد، ماشینی مثل ۳ می‌تواند بدون نیاز به هیچگونه رزرواسیون مجدد و تغییر، از بین این کانالهای ویدیویی یکی را انتخاب نماید زیرا مسیریابها به آنچه که گیرنده تماشا می‌کند دقت و اعتنایی نمی‌کنند.[1]

۵-۴-۴ خدمات متمایز (Differentiated Services)

«الگوریتمهای مبتنی بر جریان»[2] قابلیت عرضهٔ کیفیت خوب خدمات به یک یا چند جریان را دارند زیرا در طول مسیر هر منبعی را که نیاز است از قبل رزرو می‌کنند. ولی این روشها یک اشکال دارند: در این الگوریتمها نیاز است که برای هر جریان (Flow)، پیشاپیش تنظیمات لازم انجام شود در حالی که در مقیاس کلان یعنی وقتی که هزاران یا میلیونها «جریان» وجود دارد قابلیت اجرایی خود را از دست می‌دهند. از طرفی در هر مسیریاب «وضعیت» هر

---

۱. بعبارت روشنتر وقتی یک گیرنده مثل A کانالی با پهنای باند معین با فرستندهٔ B رزرو می‌کند و B بطور همزمان مثلاً ده استریم ویدیویی پخش می‌کند، گیرندهٔ A در انتخاب یکی از این ده استریم آزادی عمل دارد و نیازی نیست که به مسیریابها در این خصوص اطلاع داده شود چراکه آنها پهنای باند لازم را رزرو کرده‌اند و اصراری ندارند که بدانند فرستنده چه چیزی برای گیرنده می‌فرستد. -م

۲. Flow-Based Algorithms

جریان بطور جداگانه نگهداری می‌شود و عملکرد این الگوریتم‌ها در مقابل خرابی یک مسیریاب آسیب‌پذیر خواهد بود. نهایتاً آنکه برای تنظیم و ایجاد «جریان» باید تبادل اطلاعات پیچیده‌ای بین مسیریابها انجام گیرد. در نتیجه RSVP یا الگوریتمهای مشابه آن، بسیار کم پیاده‌سازی عملی شده‌اند.

به همین دلایل، IETF راهکاری ساده‌تر برای تأمین کیفیت خدمات (QoS) ابداع کرد؛ روشی که بدون نیاز به هیچ تنظیمات قبلی یا تعیین کلّ مسیر، می‌تواند به صورت محلی و مجزا در هر مسیریاب پیاده‌سازی شود. این راهکار اصطلاحاً «**روش مبتنی بر کلاس**» (Class-Based) برای تضمین کیفیت خدمات نامیده می‌شود (در مقابل روشهای مبتنی بر جریان). IETF یک معماری مناسب به نام «**خدمات متمایز**» برای آن طراحی و استانداردسازی کرده است که در مستندات RFC به شماره‌های ۲۴۷۴ و ۲۴۷۵ و مستندات دیگر تشریح شده است. در ادامه به تشریح این روش می‌پردازیم.

«خدمات متمایز» (که به اختصار DS گفته می‌شود) می‌تواند توسط مجموعه‌ای از مسیریابها که در یک «حوزهٔ مدیریتی واحد» (Administrative Domain) قرار می‌گیرند (مثلاً یک ISP یا شرکت مخابرات)، عرضه شود. مدیریت مسئول شبکه، مجموعه‌ای از کلاسهای متفاوت خدمات و متناظر با آن، قواعد هدایت بسته‌ها (Forwarding Rules) را تعریف می‌کند.

اگر یک مشتری برای دریافت خدمات نوع DS تقاضای ورود به شبکه را بدهد، بسته‌های ارسالی او در ورود به حوزه، فیلد «نوع خدمات» (Type of Service) را با خود حمل می‌کنند تا به برخی از آنها خدمات بهتری (**مثل خدمات ویژه**[1]) ارائه شود. ممکن است لازم باشد ترافیک تعریف شده در هر کلاس از شکل خاصی پیروی نماید (مثلاً باید از الگوریتم سطل سوراخ با نرخ خروجی مشخص تبعیت کند). متصدی شبکه با گرایشات اقتصادی و تجاری ممکن است برای انتقال «بسته‌های ویژه» (Premium Packets) هزینهٔ اضافی بگیرد یا مثلاً به ازای بهای اشتراک ثابت و ماهانه، تعداد N بستهٔ ویژه از کاربر پذیرفته و هدایت شود. دقت کنید که این الگو نیاز به تنظیمات قبلی، رزروسازی منبع و نیازی به اتلاف وقت برای مذاکره بین طرفین نهایی در هر «جریان» ندارد. به همین دلیل پیاده‌سازی خدمات DS بسیار آسان است.

«خدمات مبتنی بر کلاس» در صنایع دیگر نیز وجود دارد. به عنوان مثال، شرکتهای تحویل محموله‌های پستی (Package Delivery) نیز سه نوع خدمات عرضه می‌کنند: شبانه روزی، دو روزه یا سه روزه. یا مثلاً خطوط هواپیمایی «خدمات کلاس برتر» (First Class)، «کلاس تجاری» (Business Class) و «کلاس معمولی» عرضه می‌کنند. در قطارهای دورپیما نیز خدمات در کلاسهای متفاوتی عرضه می‌شود و حتی قطارهای زیرزمینی (مترو) نیز در دو کلاس مختلف خدمات ارائه می‌کنند. برای بسته‌های حاوی اطلاعات، کلاسهای متفاوت خدمات برحسب میزان «تأخیر»، «لرزش» (Jitter)، احتمال حذف بسته در صورت بروز ازدحام و امکاناتی نظیر همینها تعیین می‌شود.

برای آنکه تفاوت بین «کیفیت خدمات مبتنی بر جریان» و «کیفیت خدمات مبتنی بر کلاس» روشنتر شود نمونه‌ای مثل «تلفن اینترنتی» را مدنظر قرار بدهید. در روش مبتنی بر جریان، هر تماس تلفنی منابع خاصّ خود و تضمینهای لازم را شبکه اخذ می‌کند. در روش مبتنی بر کلاس تمام تماسهای تلفنی همگی از منابع رزرو شده‌ای که برای «کلاس تلفنی» تهیه دیده شده، استفاده می‌کنند. این منابع در اختیار بسته هایی که در کلاس انتقال فایل یا کلاسهای دیگر هستند، قرار نمی‌گیرد و صرفاً برای «کلاس تلفنی» پیش‌بینی شده است ولی اینگونه هم نیست که برای هر تماس تلفنی منابع اختصاصی و مجزا در نظر گرفته شود.

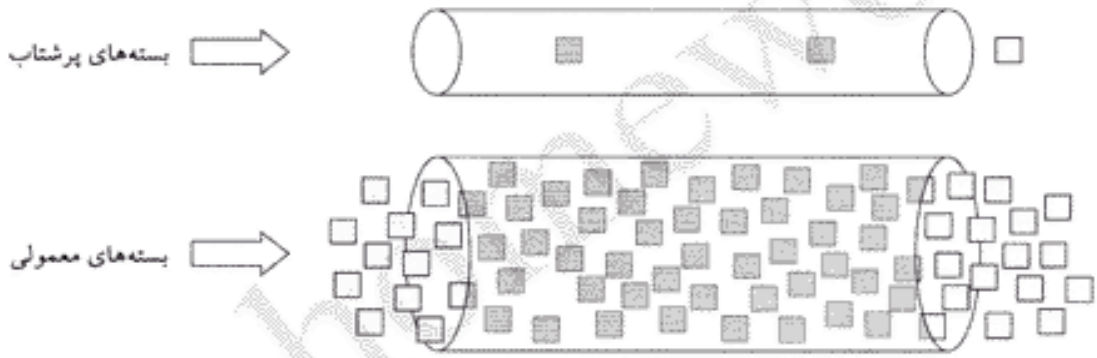
---

۱. Premium Service

هدایت پُرشتاب (Expedited Forwarding)

انتخاب کلاس خدمات بر عهدهٔ کارفرمای شبکه است ولیکن از آنجایی که بسته‌ها از چندین زیرشبکه مجزا (با کارفرمای مستقل) عبور می‌کنند [و ممکن است کلاس خدمات هر زیرشبکه متفاوت و سلیقه‌ای باشد]، IETF در حال کار بر روی تعریفی واحد برای کلاسهای خدمات است به گونه‌ای که مستقل از نوع شبکه باشد. ساده‌ترین کلاس، کلاس «هدایت پر شتاب» است که با آن شروع می‌کنیم. این کلاس در RFC 3246 تشریح شده است.

ایده‌ای که در پشت روش «هدایت پر شتاب» نهفته است ساده به نظر می‌رسد. خدمات در دو کلاس قابل ارائه است: «معمولی» و «پر شتاب» (Regular & Expedited) بخش اعظم ترافیک شبکه از نوع معمولی هستند در حالی که فقط کسر کوچکی از آن نیاز به «خدمات پُرشتاب» دارند. بسته‌های پُرشتاب باید بگونه‌ای در زیرشبکه حرکت کنند که گویی هیچ بستهٔ دیگری وجود ندارد. توصیفی نمادین از مفهوم سیستم «دو تونلی» (Two-Tube) در شکل ۵-۳۹ ارائه شده است. به خاطر داشته باشید که کماکان یک خط فیزیکی در اختیار است؛ دو لولهٔ منطقی که در شکل نشان داده شده فقط نماد رزرو بخشی از پهنای باند هستند نه آنکه یک خط فیزیکی دیگر هم وجود داشته باشد.



شکل ۵-۳۹. بسته‌های پرشتاب با شبکه‌ای بدون ترافیک مواجه می‌شوند.

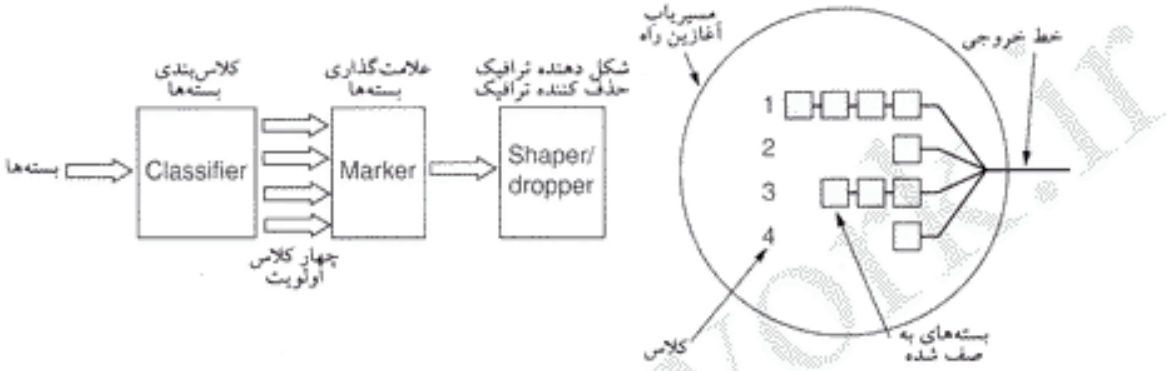
یکی از روشهای پیاده سازی این استراتژی آن است که مسیریابها به نحوی برنامه‌ریزی شوند که برای هر یک از خطوط خروجی خود دو صف مجزا تشکیل بدهند: یکی برای بسته‌های پر شتاب و دیگری برای بسته‌های معمولی. بسته‌های دریافتی برحسب نوع آنها به یکی از این صفها وارد می‌شوند. برای زمان‌بندی بسته‌ها می‌توان از روشی مثل «روش صف‌بندی وزن‌دار» (Weighted Queuing) بهره گرفت. مثلاً اگر ده درصد از بسته‌ها از نوع پر شتاب و مابقی از نوع معمولی باشند، تخصیص ۲۰ درصد از پهنای باند برای ترافیک پر شتاب و هشتاد درصد باقیمانده برای ترافیک معمولی مناسب خواهد بود. با این کار پهنای باند اختصاص داده شده به ترافیک پر شتاب دو برابر مقدار مورد نیاز آن است و بدین ترتیب تأخیر پائینی خواهد داشت. برای اجرای چنین راهکاری می‌توان به ازای ارسال ۴ بستهٔ معمولی یک بستهٔ پر شتاب ارسال کرد (البته با فرض آن که اندازهٔ بسته‌ها توزیعی مشابه و یکنواخت داشته باشد.) انتظار می‌رود در این روش حتی در صورت سنگین بودن بار زیرشبکه، بسته‌های پر شتاب زیرشبکه را بی‌بار و خلوت ببینند.

هدایت تضمین شده (Assured Forwarding)

برای مدیریت انواع کلاسهای خدمات، روشی دقیقتر به نام «هدایت تضمین شده» ارائه و در RFC 2597 تشریح شده است. در این روش چهار کلاس اولویت تعریف شده و هر کلاس منابع خاص خود را در اختیار دارد. علاوه بر این، سه احتمال برای حذف بسته در اثر بروز ازدحام تعریف شده است: احتمال پایین، متوسط و زیاد. مجموع

ترکیبات مختلف این دو عامل، دوازده کلاس خدمات متفاوت ایجاد می‌کند.

در شکل ۵-۴۰ یک روش برای پردازش بسته‌ها به منظور هدایت تضمینی آنها، نشان داده شده است. در مرحلهٔ اول بسته‌ها برحسب کلاس اولویتشان در یکی از چهار کلاس، رده‌بندی می‌شوند. این مرحله می‌تواند در ماشین میزبان فرستندهٔ بسته‌ها انجام شود (به نحوی که در شکل نشان داده شده است) یا آنکه در اولین مسیریاب (مدخل ورودی به زیرشبکه) انجام شود.



شکل ۵-۴۰. پیاده‌سازی مکانیزم هدایت تضمین شده برای یک «جریان داده».

در مرحلهٔ ۲ بسته‌ها بر حسب کلاسشان علامتگذاری می‌شوند. بدین منظور در سرآیند هر بسته به فیلد خاصی نیاز است. خوشبختانه یک فیلد هشت بیتی به نام Type of Service (نوع خدمات) در بستهٔ IP وجود دارد که در آینده آن را به اختصار بررسی خواهیم کرد. در RFC 2597 شش بیت از این هشت بیت برای تعیین کلاس بسته‌ها تعریف شده و دو بیت باقیمانده برای استفاده‌هایی که از قبل داشته یا استفاده در آینده، رها شده‌اند.

در مرحلهٔ سوم بسته‌ها از یک «فیلتر شکل دهنده / حذف کننده» (Shaper / Dropper Filter) عبور کرده و برای آنکه ترافیک بسته‌های هر یک از چهار کلاس شکل قابل قبولی داشته باشند به برخی از آنها تأخیر مصنوعی تحمیل می‌شود (مثلاً به کمک الگوریتم سطل سوراخ یا سطل نشانه‌دار). در صورتی که تعداد بسته‌ها در هر کلاس، از حدّ مجاز بیشتر شده باشد در این مرحله برخی از آنها حذف می‌گردند. (روشهای دقیقتری نیز برای حذف بسته‌ها وجود دارد که از فیدبک بهره می‌گیرند.)

در این مثال هر سه مرحلهٔ فوق‌الذکر توسط ماشین فرستنده انجام شده و جریان خروجی بسته‌ها به اولین مسیریاب ارسال می‌شود. بدیهی است که این مراحل می‌تواند توسط یک نرم‌افزار خاص شبکه یا سیستم عامل هر ماشین انجام شود تا نیازی به تغییر در برنامه‌های کاربردی موجود نباشد.

۵-۴-۵ سوئیچ برچسب[۱] و MPLS

در خلال زمانی که IETF بر روی موضوع «خدمات مجتمع» کار می‌کرد، چندین تولیدکنندهٔ محصولات مسیریابی نیز بر روی روشهای بهتر هدایت بسته‌ها متمرکز شده بودند. کار آنها بر این محور بود که در ابتدای هر بسته یک «برچسب» (Label) اضافه شود و بجای آنکه مسیریابی و هدایت بسته‌ها مبتنی بر آدرس مقصد باشد براساس این «برچسب» انجام شود. با استفاده از این «برچسب» به عنوان یک اندیس در جدول داخلی هر مسیریاب، خط خروجی صحیح و مناسب برای هر بسته پیدا می‌شود. بکمک این روش، مسیریابی بسته‌ها به سرعت انجام شده و

۱. (Label Switching)

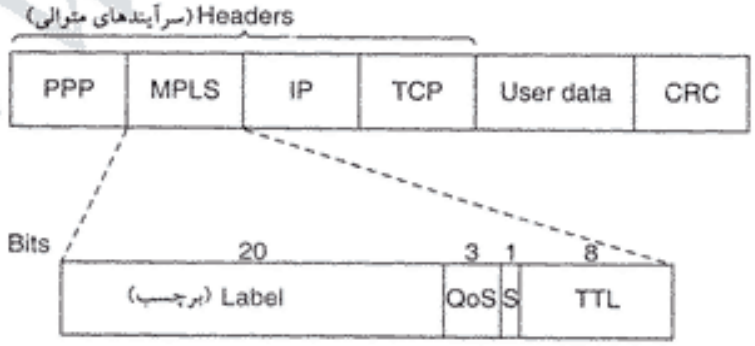
منابع مورد نیاز در طول مسیر رزرو خواهد شد.

البته برچسب‌گذاری بر روی هر «جریان» شباهت عجیبی به مدارهای مجازی پیدا می‌کند. در شبکه‌های ATM، X.25 و Frame Relay یا هر زیرشبکه مدار مجازی دیگر نیز یک «برچسب» (یا به عبارتی یک شناسهٔ مدار مجازی[1]) در هر بسته قرار داده می‌شود و با استفاده از آن به عنوان یک اندیس برای درایه‌های جدول[2]، مسیر مناسب بدست می‌آید. علیرغم آنکه بسیار از افراد در جامعهٔ اینترنت از شبکه‌های اتصال‌گرا بشدت گریزان هستند، به نظر می‌رسد که این ایده با هدف مسیریابی سریع و تأمین کیفیت خدمات (QoS) بار دیگر به صحنه برگشته است. ولیکن بین روشی که در اینترنت برای تعیین مسیر بکار می‌رود و روشی که در شبکه‌های مدار مجازی اعمال می‌شود تفاوتهای بنیادی وجود دارد و تکنیک برچسب گذاری بسته‌ها با روش سنتی سوئیچینگ متفاوت است.

ایدهٔ جدید سوئیچینگ با نامهای متنوعی مثل **«سوئیچینگ برچسب»**[3] یا **«سوئیچینگ علامت»**[4] شناخته می‌شود. در نهایت IETF آن را تحت نام MPLS[5] استاندارد کرد. ما نیز در ادامه از نام MPLS استفاده می‌کنیم. این استاندارد در RFC 3031 و چندین RFC دیگر تشریح شده است.

مضاف بر این، برخی افراد بین «مسیریابی» و «سوئیچینگ» فرق می‌گذارند. مسیریابی فرآیند جستجو در جدول مسیریابی به دنبال آدرس مقصد هر بسته و پیدا کردن خط مناسب برای آن است. برعکس در فرآیند سوئیچینگ از برچسب هر بسته به عنوان یک اندیس در جدول مسیریابی استفاده می‌شود و با استفاده از این اندیس بلافاصله خط خروجی پیدا می‌شود، بدون آن که نیازی به جستجو باشد. البته این تعاریف و تعابیر جهان‌شمول و همگانی نیستند.

اولین مسئله آنست که این برچسب در کجا قرار داده شود. از آنجایی که بسته‌های IP برای شبکه‌های مدار مجازی طراحی نشده بودند، طبعاً هیچ فیلدی در سرآیند بستهٔ IP برای درج شماره‌های مدار مجازی وجود ندارد. به همین دلیل سرآیند جدید MPLS، باید در جلوی سرآیند هر بستهٔ IP قرار بگیرد. در خطوط مستقیم بین هر دو مسیریاب که مبتنی بر «فریمینگ PPP» کار می‌کنند ترتیب سرآیندها طبق شکل ۵-۴۱ عبارتند از: سرآیند PPP، سرآیند MPLS، سرآیند IP و نهایتاً سرآیند TCP. در واقع باید MPLS را در لایه ۲/۵ فرض کرد!!!



شکل ۵-۴۱. ارسال یک قطعهٔ TCP (TCP Segment) با استفاده از IP، MPLS، و PPP.

سرآیند عمومی MPLS (MPLS Header) چهار فیلد دارد که مهمترین آنها فیلد Label (فیلد برچسب) است که در آن یک اندیس درج می‌شود. فیلد QoS، کلاس خدمات را مشخص می‌کند. فیلد S بدان منظور تعریف شده که در شبکه‌های سلسله‌مراتبی چندین سرآیند MPLS متوالیاً به بسته اضافه گردد. (این موضوع در زیر تشریح

---

۱. Virtual Circuit Identifier ۲. Table Entries ۳. Label Switching
۴. Tag Switching ۵. Multi-Protocol Label Switching

شده است.) فیلد TTL زمان حیات بسته را مشخص می‌کند و به ازای هر گام یک واحد از آن کم می‌گردد؛ هر گاه مقدار این فیلد به صفر برسد، بسته حذف می‌شود. این ویژگی بدان منظور مفید است که از حلقهٔ بی‌نهایت که در اثر ناپایداری (عدم همگرایی) جدول مسیریابی بروز می‌کند، اجتناب شود.

از آنجایی که سرآیند MPLS بخشی از بستهٔ لایهٔ شبکه یا فریم لایهٔ پیوند داده‌ها محسوب نمی‌شود لذا MPLS تا حد زیادی مستقل از هر دو لایه است. از بین تمام محاسن دیگر، دستاورد ویژگی «استقلال از دیگر لایه‌ها» آنست که می‌توان سوئیچهای MPLS را به گونه‌ای ساخت که بتواند هم بسته‌های IP و هم سلولهای ATM را برحسب مورد، هدایت کند. این ویژگی همانی است که براساس آن کلمهٔ Multiprotocol در ابتدای نام MPLS ظاهر شده است.

وقتی یک بسته یا سلول غنی‌شده با سرآیند MPLS در یک مسیریاب MPLS دریافت می‌شود از برچسب آن به عنوان اندیسی در جدول داخلی مسیریاب استفاده شده و خط خروجی متناسب با آن تعیین می‌شود و قبل از خروج بسته از آن خط، برچسب جدیدی در فیلد مربوطه درج می‌گردد. تغییر در برچسبها در تمام زیرشبکه‌های مدار مجازی معمول و متعارف است چرا که برچسبها در هر مسیریاب معنای محلی دارند و دو مسیریاب متفاوت ممکن است بسته‌های نامربوط را با برچسبی یکسان برای مسیریاب دیگر بفرستند چراکه این بسته‌ها همگی در بخشی از مسیر مشترکند.[1] به همین دلیل در هر گام برچسبهای بسته قبل از انتقال بر روی خط خروجی به برچسب جدید و معتبر در مسیریاب بعدی نگاشته می‌شود. این مکانیزم را در شکل ۵-۳ مشاهده کردیم. MPLS نیز از روش مشابهی بهره گرفته است.

یکی از تفاوتهای MPLS با شبکه‌های مدار مجازی، «میزان تجمیع» (Aggregation Level) و ضرفه‌جویی در تعداد درایه‌های جدول[2] مسیریابی است. در MPLS این امکان وجود دارد که هر «جریان» در زیرشبکه، دارای مجموعهٔ برچسبهای خاص خود باشد ولی این قابلیت مهم نیز وجود دارد که گروهی از جریانها که همگی به یک مسیریاب خاص یا یک LAN ختم می‌شوند با برچسب یکسان و واحدی مشخص شوند. [بدین ترتیب تعداد درایه‌های جدول مسیریابی کاهش یافته و اصطلاحاً عمل تجمیع یا Aggregation انجام می‌شود.] گروهی از جریانها که با یک برچسب واحد مشخص می‌شوند اصطلاحاً به یک FEC[3] مشابه متعلق هستند. [یعنی همهٔ آنها به یک طریق هدایت می‌شوند.] کلاس FEC نه تنها مقصد همهٔ بسته‌ها را مشخص می‌کند، بلکه کلاس خدمات مورد نیاز آنها را نیز تعیین می‌نماید (از دیدگاه انواع خدمات متمایز که در بخش ۵-۴-۴ بدان اشاره شد). طبعاً فرآیند هدایت تمام بسته‌های یک FEC یکسان خواهد بود.

در مسیریابی متعارف به روش مدار مجازی، این قابلیت که بتوان چندین مسیر مجزا با نقاط پایانی متفاوت را با «شناسهٔ مدار مجازی» واحد مشخص کرد وجود ندارد چرا که در این صورت راهی برای مشخص کردن مقصد نهایی بسته‌ها وجود نخواهد داشت در حالی که در MPLS بسته‌ها [به غیر از سرآیند ۴ بایتی MPLS] آدرس واقعی ماشین مقصد را نیز با خود حمل می‌کنند و بدین ترتیب در انتهای مسیری که با برچسب مشخص شده می‌توان سرآیند حاوی برچسب را حذف کرد و هدایت بسته‌ها به روش معمول و مبتنی بر آدرس لایهٔ شبکه [مثل آدرس IP] ادامه یابد.

یکی از تفاوتهای بنیادی بین MPLS و شبکه‌های مدار مجازی در چگونگی تشکیل جداول مسیریابی است.

---

۱. بعبارت دیگر برچسب هر بسته هویت آنرا مشخص نمی‌کند بلکه مسیر خروج آن از مسیریاب فعلی را مشخص می‌کند لذا طبیعی است که بسته‌های خروجی از یک مسیریاب که هیچ ربطی بهم ندارند ولی لااقل گام بعدی مسیر آنها یکی است (یعنی از خط مشابهی وارد مسیریاب بعدی و از خط مشابهی، از آن خارج می‌شوند) دارای برچسب یکسانی باشند. -م

۲. Table Entries ۳. Forwarding Equivalence Class

در شبکه‌های مدار مجازی وقتی یک کاربر بخواهد یک «اتصال» ایجاد کند، توسط لایهٔ شبکه یک بستهٔ خاصّ جهت تنظیم مسیر به زیرشبکه روانه می‌شود تا ضمن ایجاد یک مسیر درایه‌های لازم در جداول مسیریابی درج شود. MPLS بدین نحو عمل نمی‌کند چرا که در آن عمداً هیچ مرحله‌ای برای تنظیم اتصال پیش‌بینی نشده است. (زیرا در غیر این صورت نرم‌افزارهای موجود اینترنت دچار شکاف و ناسازگاری می‌شد.) در عوض برای تنظیم و ایجاد درایه‌های جدول مسیریابی از دو راهکار جدید استفاده شده است.

در راهکار اول که «**روش متکی به داده**» (Data driven) نامیده می‌شود هرگاه بسته‌ای در اولین مسیریاب دریافت شود، آن مسیریاب با مسیریاب واقع بر روی مسیر جریان، تماس گرفته و از او می‌خواهد که یک برچسب برای این جریان ایجاد نماید. این فرآیند به صورت بازگشتی (Recursive) ادامه می‌یابد تا مجموعهٔ برچسبها ایجاد شوند. در واقع این روش را می‌توان ایجاد «مدار مجازی برحسب تقاضا»[1] فرض کرد.

پروتکلهایی که عمل برچسب‌دهی را انجام می‌دهند مراقب هستند تا از بروز حلقه اجتناب شود. برای این کار از تکنیکی به نام «ریسمانهای رنگی» (Colored Thread) بهره گرفته می‌شود. انتشار معکوس یک FEC را می‌توان با یک «ریسمان رنگی و یکتا» [تمثیلی از یک مسیر در شبکه] در زیرشبکه مقایسه کرد. اگر مسیریاب رنگی را مشاهده کند که خودش نیز به همان رنگ است متوجه می‌شود که در انتخاب مسیر، حلقه ایجاد شده و برای رفع آن اقدام می‌کند.[2] روش «برچسب دهی متکی به داده» (Data driven) در شبکه‌هایی کاربرد دارد که زیر ساخت انتقال آنها ATM است. (همانند بیشتر سیستمهای تلفن)

راهکار دیگر برای برچسب دهی به جریان داده‌ها، در شبکه‌هایی کاربرد دارد که زیربنای آنها ATM نیست. این روش اصطلاحاً «**روش متکی به کنترل**» (Control Driven) نامیده می‌شود و گونه‌های متنوعی از آن وجود دارد. یکی از این گونه‌ها به ترتیب ذیل عمل می‌کند: وقتی یک مسیریاب راه‌اندازی (بوت) می‌شود ابتدا بررسی می‌کند که در انتهای چه مسیرهایی قرار دارد (یعنی مثلاً چه ماشینهایی بر روی LAN متصل به او قرار دارند). سپس برای تمام آنها یک یا چند FEC [شناسهٔ یک گروه یا کلاس معادل] تولید کرده و ضمن تخصیص یک برچسب به هر یک از این گروهها، آنها را به همسایه‌های خود اطلاع می‌دهد. آنها نیز به ترتیب برچسبها را در جدول مسیریابی خود وارد کرده و با تعیین برچسبی جدید [متناظر با هر برچسب قبلی] آنها را به همسایه‌های خود اطلاع می‌دهند تا آنکه تمام مسیریابها از مسیرهای جدید آگاه شوند. در حین ایجاد مسیر می‌توان منابع لازم را نیز برای تضمین کیفیت خدمات رزرو کرد.

MPLS می‌تواند بطور همزمان در چندین سطح عمل کند. در بالاترین سطح ، هر زیرشبکه حامل را می‌توان یک نوع Metarouter فرض کرد که بین هر مبداء و مقصد مسیری وجود دارد که از این **متاروترها** می‌گذرد؛ در این مسیر از MPLS استفاده می‌شود.[3] با این حال در درون یک زیرشبکه حامل نیز می‌توان از MPLS بهره گرفت و بدین ترتیب مسیریابهای داخلی نیز برچسب دومی به هر بسته می‌افزایند و برچسب گذاری سطح دوم پدید می‌آید. در حقیقت یک بسته می‌تواند دنباله‌ای از برچسبهای MPLS را به همراه داشته باشد. بیت S در شکل ۵-۴۱ مسیریاب را آگاه می‌کند که آیا برچسبهای دیگری هم وجود دارد. در آخرین برچسب، بیت S یک است؛ در حالیکه در بقیهٔ برچسبها بیت S صفر می‌باشد. در عمل می‌توان از این قابلیت برای پیاده‌سازی VPN یا تونلهای بازگشتی

---

۱. On-Demand Vitual Circuit

۲. به عبارت بهتر اگر یک مسیر بین دو نقطه را در قالب یک ریسمان رنگی مجسم کنیم هر بسته‌ای که با رنگ همان مسیر دو بار دریافت شود نشان می‌دهد که بسته در یک حلقه قرار گرفته است وگرنه باید تا رسیدن به مقصد راه خود را ادامه بدهد. -م

۳. یعنی مسیر بین مبداء و مقصد از زیرشبکه‌های متفاوتی می‌گذرند که هر یک از این زیرشبکه‌های حامل یک مسیریاب واحد به نام «متاروتر» فرض شده است. -م

(Recursive Tunnel) بهره گرفت.

اگرچه ایدهٔ بنیادی MPLS ساده است ولیکن جزئیات آن بی‌نهایت پیچیده است و تنوع و بهینه‌سازیهای گسترده‌ای دارد، لذا ما بیش از این به موضوع فوق نخواهیم پرداخت. برای آگاهی بیشتر از مراجع ذیل استفاده کنید:

Davie and Rekhter, 2000; Lin et al., 2002; Pepelnjak and Guichard, 2001; Wang, 2001.

۵.۵ بهم‌بندی شبکه‌ها (Internetworking)

تا اینجا تلویحاً فرض کرده‌ایم که تنها یک شبکهٔ واحد و همگن وجود دارد که تمام ماشینهای چنین شبکه‌ای، در تمام لایه‌ها از پروتکل مشابهی بهره گرفته‌اند. متأسفانه چنین فرضی خیلی خوش‌باورانه است. شبکه‌ها اعم از LAN، MAN و WAN، انواع بسیار گوناگونی دارند. در هر لایه نیز از پروتکلهای متعددی استفاده می‌شود. در بخشهای آتی مواردی را موشکافی خواهیم کرد که در وصل دو یا چند شبکه و تشکیل «اینترنت»[1] (internet) با آن مواجه خواهیم بود.

مناقشات گسترده‌ای پیرامون این سؤال وجود دارد که آیا کثرت بسیار زیاد انواع شبکه‌ها در حال حاضر، وضعیتی گذرا و موقتی است و به محض آنکه عموم مردم به شگفتی‌های یک شبکهٔ خاص (مثلاً شبکهٔ مورد نظر شما!) پی بردند این کثرت به وحدت می‌رسد یا آنکه تکثر انواع امری اجتناب‌ناپذیر و همیشگی در جهان است و باقی خواهد ماند. وجود شبکه‌های متفاوت مستلزم داشتن پروتکلهای متفاوت است.

اعتقاد ما بر آن است که به دلایل ذیل گونه‌های متفاوتی از شبکه‌ها (و به تبع آن پروتکلهای متفاوت) تا ابد وجود خواهد داشت: اول آنکه شبکه‌های بسیار متنوعی در محیطهای متفاوتی نصب شده‌اند: تقریباً تمام کامپیوترهای شخصی بر مبنای TCP/IP کار می‌کنند. بسیاری از مؤسسات تجاری دارای کامپیوترهای بزرگ (Mainframe) با معماری شبکهٔ SNA (متعلق به شرکت IBM) هستند. تعداد قابل توجهی از شرکتهای مخابرات تلفنی، شبکه‌های ATM را به خدمت گرفته‌اند. در برخی از شبکه‌های محلی هنوز از پروتکل NCP/IPX (متعلق به شرکت Novell) یا AppleTalk (متعلق به شرکت Apple) بهره می‌گیرند. و از همه گذشته شبکه‌های بی‌سیم با پروتکلهای متنوعی در حال ظهور هستند. این رویه برای سالها ادامه خواهد داشت چرا که تکنولوژیهای جدید ظهور می‌کند، اشکالات و ناکارآمدیهای گذشته آشکار می‌شود؛ در عین حال با عرضهٔ تکنولوژی جدید نمی‌توان یک شبه مشتریان را وادار کرد سیستمی جدید را بپذیرند و سیستمهای قدیمی خود را دور بریزند.

دلیل دوم آنکه کامپیوترها و شبکه‌ها روز به روز ارزانتر می‌شوند و تصمیم‌گیری در خصوص انتخاب و پیاده‌سازی شبکه‌ها به رده‌های پائینی یک سازمان محول می‌شود. بسیاری از شرکتها این سیاست را پیش گرفته‌اند که خریدهای بالای یک میلیون دلار باید توسط مدیر ارشد آن شرکت تأئید شود و خریدهای بالای صد هزار دلار توسط مدیران میانی مُجاز است در حالی که خریدهای زیر صد هزار دلار توسط مدیران هر واحد بدون اجازهٔ مدیران مافوق انجام می‌گیرد. این رویه به آنجا منتهی می‌شود که مثلاً واحد فنی مهندسی در یک سازمان، ایستگاههایی مبتنی بر یونیکس و پروتکل TCP/IP پیاده کند و واحد فروش، کامپیوترهای Mac با پروتکل AppleTalk را به خدمت بگیرد.

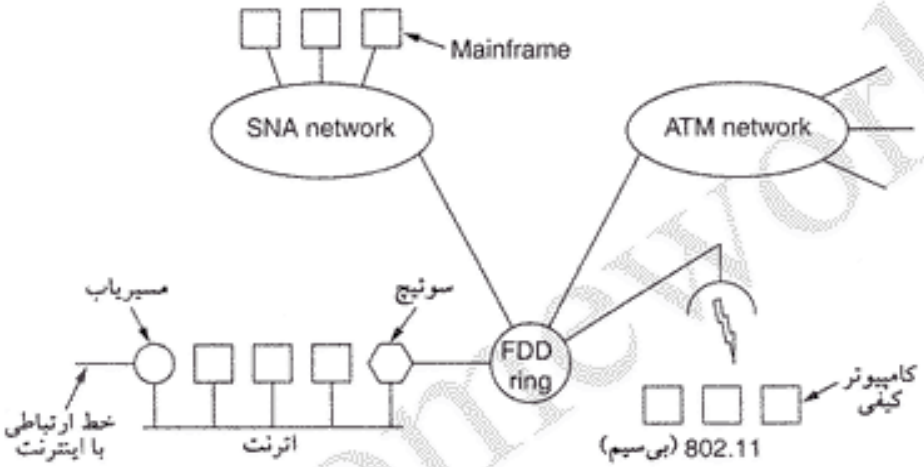
دلیل سوم آنکه شبکه‌های مختلف (مثل ATM و بی‌سیم) تکنولوژی شدیداً متفاوتی دارند لذا دور از انتظار نیست که وقتی سخت‌افزار جدیدی خلق می‌شود نرم‌افزار جدیدی نیز برای به خدمت‌گیری آن ایجاد شود. به

---

۱. وقتی کلمهٔ internet تماماً با حروف کوچک نوشته می‌شود به شبکهٔ عظیم و جهانی اینترنت اشاره نمی‌کند بلکه مراد از آن وصل چند شبکهٔ کوچک و بزرگ و یکپارچه‌سازی آنهاست؛ مخفف internetwork

عنوان مثال امروزه منازلی که در آنها کامپیوتر وجود دارد همانند ادارات در ده سال قبل است: مملو از کامپیوترهایی که ارتباطی با هم ندارند! در آینده ممکن است تلفن، تلویزیون و دستگاههای الکترونیکی خانگی نیز با یکدیگر شبکه شوند و بتوان از راه دور آنها را کنترل کرد. این تکنولوژی جدید بی‌شک شبکه‌های جدید و پروتکلهای جدیدی را به صحنه خواهد آورد.

به عنوان مثالی از چگونگی اتصال شبکه‌های متفاوت به یکدیگر به شکل ۵-۴۲ دقت کنید. در این شکل شبکهٔ یکپارچه‌ای را می‌بینیم که اجزای آن در چندین موقعیت فیزیکی پراکنده هستند و از طریق یک شبکهٔ گستردهٔ ATM بهم متصل شده‌اند. در یکی از مکانها یک شبکهٔ فیبرنوری FDDI به عنوان ستون فقرات نصب شده است تا ارتباط یک شبکهٔ اترنت، یک شبکهٔ بی‌سیم 802.11 و یک شبکه متمرکز SNA را با یکدیگر برقرار کند.



شکل ۵-۴۲. مجموعه‌ای از شبکه‌های بهم متصل.

هدف از اتصال تمام این شبکه‌ها آن است که کاربران هر یک از آنها بتوانند با یکدیگر مبادله اطلاعات داشته باشند یا آنکه هر کاربر بتواند به اطلاعات مورد نظر خود در هر نقطه از شبکه دسترسی پیدا کند. رسیدن بدین هدف متضمن آن است که بسته‌ها، بین این شبکه‌ها رد و بدل شوند. از آنجایی که شبکه‌ها اختلافات بنیانی با یکدیگر دارند، رساندن بسته‌ها از یک شبکه به شبکهٔ دیگر به نحوی که در ادامه خواهیم دید، چندان هم ساده نیست.

۵-۵-۱ شبکه‌ها از چه دیدگاهی متفاوتند؟

شبکه‌ها در موارد مختلفی با یکدیگر تفاوت ذاتی دارند. برخی از این تفاوتها مثل تکنیکهای مدولاسیون یا قالب فریم، مربوط به لایهٔ فیزیکی یا لایهٔ پیوند داده‌ها است. اینگونه تفاوتها مدنظر ما نیستند. در مقابل، در شکل ۵-۴۳ برخی از تفاوتهایی را که در لایهٔ شبکه بروز می‌کنند، فهرست نموده‌ایم. تشریح این تفاوتهاست که نشان می‌دهد بهم بستن شبکه‌ها دشوارتر از کار کردن در یک شبکهٔ واحد و همگون است.

وقتی بستهٔ ارسالی از یک مبداء در یکی از شبکه‌ها، مجبور باشد برای رسیدن به شبکهٔ مقصد از یک یا چند شبکهٔ خارجی عبور کند (که این شبکه‌ها نیز ممکن است با شبکهٔ مبداء اختلاف بنیادی داشته باشند)، در مرز ارتباطی بین دو شبکه مشکلات عدیده‌ای رخ می‌دهد. اولین مورد آن است که وقتی بسته‌هایی از یک شبکهٔ «اتصال‌گرا» مجبور به عبور از یک شبکهٔ «بدون اتصال» باشند (که احتمالاً ترتیب بسته‌ها را بهم می‌ریزد) این مشکل بروز می‌کند که فرستندهٔ بسته‌ها انتظار چنین رخدادی را ندارد و گیرنده نیز کاری نمی‌تواند انجام بدهد. [چرا که فرض فرستنده و گیرنده آن است که بسته‌ها به ترتیب ارسال و به ترتیب نیز دریافت می‌شود. -م]

غالباً بین دو شبکه به تبدیل پروتکل نیاز است و اگر عملکرد مورد نیاز برآورده نشود این تبدیل دشواریهایی را در

مورد اختلاف	برخی از رویکردهای ممکن
نوع سرویس ارائه شده	سرویسهای اتصال‌گرا در مقابل سرویسهای بدون اتصال
انواع پروتکل	IP, IPX, SNA, ATM, MPLS, AppleTalk, ...
الگوی آدرس‌دهی	روش مسطح (Flat) مثلا در استانداردهای ۸۰۲ در مقابل روش سلسله‌مراتبی در IP
چندپخشی	در برخی از شبکه‌ها از آن پشتیبانی می‌شود و در برخی نمی‌شود.
اندازه بسته	هر شبکه برای خودش یک سقف حداکثر برای طول بسته تعریف کرده است.
کیفیت خدمات (QoS)	در برخی از شبکه‌ها از آن پشتیبانی می‌شود (آنهم در رده‌های متفاوت) و در برخی نمی‌شود.
مدیریت خطا	تحویل مطمئن و به ترتیب در مقابل تحویل غیرقابل اطمینان و خارج از ترتیب
کنترل جریان	پنجره لغزان، کنترل نرخ ارسال یا حتی بدون مکانیزم کنترل جریان
کنترل ازدحام	اعمال الگوریتم سطل سوراخ، الگوریتم سطل نشانه‌دار، بسته‌های دعوت به آرامش و نظائر آن
امنیت	قوانین امنیتی، اعمال روشهای رمزنگاری و نظائر آن
پارامترها	مقادیر مختلف زمان انقضای مهلت تایمرها، پارامترهای متفاوت توصیف جریان و نظائر آن
حسابرسی و دریافت هزینه	بر حسب زمان اتصال، برحسب تعداد بسته، بایت یا حتی هیچکدام

شکل ۵-۴۳. موارد بی‌شمار اختلاف شبکه‌ها.

پی خواهد داشت. همچنین اغلب به تبدیل و نگاشت آدرسها نیاز است که متضمن وجود گونه‌ای از یک «سیستم فهرست» (Directory System) خواهد بود. از طرفی عبور یک بستهٔ چندپخشی (Multicast) از شبکه‌ای که نمی‌تواند از مسیریابی چندپخشی پشتیبانی کند مستلزم تولید بسته‌های جداگانه برای یکایک ماشینهای مقصد است.

تفاوت در مقدار حداکثر طول هر بستهٔ داده در شبکه‌های مختلف، می‌تواند یک معضل اساسی باشد: چگونه می‌توان یک بستهٔ ۸۰۰۰ بایتی را از شبکه‌ای عبور داد که حداکثر طول بسته‌های آن ۱۵۰۰ بایت است؟ تفاوت در کیفیت خدمات دو شبکه (QoS) نیز موردی است که برای تحویل بسته‌های بی‌درنگ از طریق شبکه‌ای که بی‌درنگ بودن ارسال را تضمین نمی‌کند، به یک معضل جدی تبدیل می‌شود.

کنترل خطا، کنترل جریان و کنترل ازدحام نیز در شبکه‌های مختلف، تفاوت دارد. اگر مبداء و مقصد، هر دو انتظار داشته باشند که تمام بسته‌ها به ترتیب و بدون خطا تحویل شوند ولی یک شبکهٔ میانی به محض احساس بروز ازدحام برخی از بسته‌ها را حذف نماید، بسیاری از برنامه‌های کاربردی درهم خواهند شکست. همچنین اگر بسته‌ها مدتی را بی‌هدف سرگردان شوند و به ناگاه راه خود را پیدا کرده و تحویل داده شوند و گیرنده انتظار چنین رفتاری را نداشته و از عهدهٔ دریافت این بسته‌ها بر نیاید معضلی جدی پدیدار می‌شود. همچنین مکانیزمهای تضمین امنیت، تنظیم پارامترها، قواعد دریافت هزینه (حسابرسی) و حتی قوانین ملی متفاوت، می‌تواند منجر به بروز مشکلاتی شود.

۵-۵-۲ چگونگی اتصال شبکه‌ها به یکدیگر

به گونه‌ای که در فصل چهارم بررسی کردیم شبکه‌ها را می‌توان به کمک ابزارهای متفاوتی به یکدیگر متصل کرد. اجازه بدهید اجمالاً به آن مفاد بپردازیم: در لایهٔ فیزیکی شبکه‌ها را می‌توان توسط «تکرارکننده» (Repeater) یا هاب به یکدیگر متصل کرد. این دو ابزار فقط بیتها را از یک شبکه به شبکه‌ای از همان نوع منتقل می‌نمایند. اینها اغلب ابزارهای آنالوگ هستند و هیچ درکی در خصوص پروتکلهای دیجیتال [پروتکلهای لایهٔ بالاتر] ندارند و فقط سیگنالهای دریافتی را «باز تولید» (Regenerate) می‌نمایند.

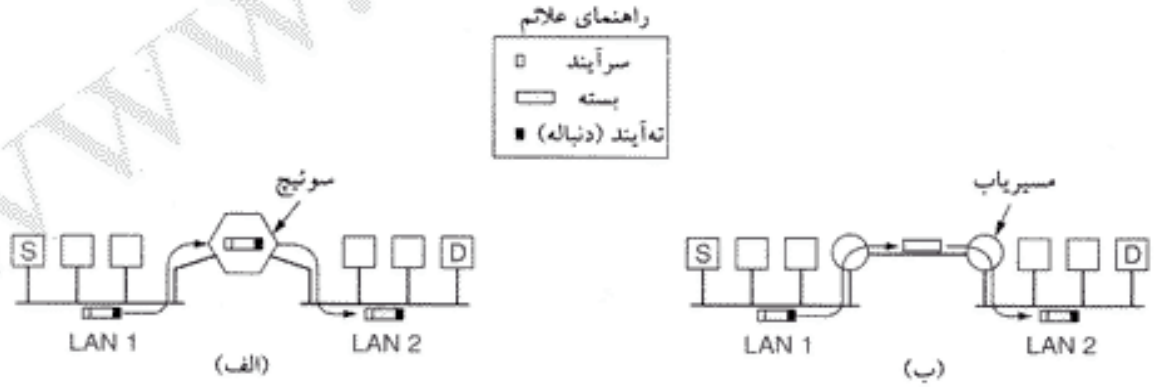
در یک لایهٔ بالاتر به پلها و سوئیچها بر می‌خوریم که در لایهٔ پیوند داده‌ها کار می‌کنند. این دستگاهها فریمها را می‌پذیرند، آدرسهای MAC را بررسی می‌کنند و آنها را به شبکه‌ای دیگر هدایت می‌نمایند. در ضمن اگر لازم باشد تبدیل پروتکل نیز انجام می‌دهند مثلاً اترنت را به FDDI یا 802.11 تبدیل می‌کنند.

در لایهٔ شبکه، مسیریابها را داریم که می‌توانند دو شبکه را به یکدیگر متصل کنند. اگر دو شبکه، لایه‌های شبکهٔ متفاوتی داشته باشند، مسیریاب ممکن است بتواند قالب بسته‌ها را به یکدیگر ترجمه نماید اگرچه امروزه ترجمه و تبدیل بسته‌ها به یکدیگر، به ندرت انجام می‌شود. یک مسیریاب را که بتواند با چندین پروتکل مختلف کار کند، اصطلاحاً «مسیریاب چندپروتکلی» (Multiprotocol Router) می‌نامند.

در لایهٔ انتقال به «دروازه‌های انتقال» (Transport Gateway) می‌رسیم که می‌توانند واسطهٔ بین دو اتصال در لایهٔ انتقال شوند. به عنوان مثال یک «دروازهٔ انتقال» می‌تواند این امکان را فراهم کند که جریان بسته‌ها بین یک شبکهٔ TCP و یک شبکهٔ SNA (که پروتکل لایهٔ انتقال آنها متفاوت است)، مبادله شود. این دروازه، یک «اتصال TCP» (TCP connection) را به یک «اتصال SNA» می‌چسباند.

در آخر به لایهٔ کاربرد و «دروازهٔ کاربرد» (Application Gateway) می‌رسیم که این دروازه محتوای پیامها را بهم ترجمه می‌نماید. به عنوان مثال یک دروازهٔ بین سیستم پست الکترونیکی در اینترنت (مثل سیستم RFC822) و سیستم پست الکترونیکی X.400، باید بتواند محتوای پیام نامه‌های الکترونیکی را تجزیه و تحلیل کرده و فیلدهای مختلف سرآیند آنرا به یکدیگر تبدیل نماید.

در این فصل به موضوع بهم‌بندی شبکه‌ها در لایهٔ شبکه خواهیم پرداخت. برای آنکه ببینید هدایت اطلاعات در لایهٔ پیوند داده‌ها چه تفاوتی با هدایت در لایهٔ شبکه دارد، شکل ۵-۴۴ را در نظر بگیرید. در شکل ۵-۴۴-الف ماشین مبداء یعنی S می‌خواهد بسته‌ای را برای ماشین مقصد D بفرستد. این دو ماشین بر روی دو شبکهٔ اترنت جدا که از طریق سوئیچ بهم متصل شده‌اند، واقع هستند. ماشین S بسته‌ای را در درون یک فریم جاسازی کرده و آن را به خروجی می‌فرستد. این فریم به سوئیچ رسیده و با بررسی آدرس MAC مشخص می‌شود که باید به LAN2 برود. سوئیچ، این فریم را از LAN1 برداشته و به LAN2 روانه می‌کند.



شکل ۵-۴۴. (الف) دو شبکهٔ اترنت که از طریق سوئیچ بهم متصل شده‌اند. (ب) دو شبکهٔ اترنت که از طریق مسیریاب بهم متصل شده‌اند.

حال همین وضعیت را در شرایطی در نظر بگیرید که این دو شبکهٔ اترنت به جای سوئیچ از طریق یک جفت مسیریاب به یکدیگر متصل شده‌اند. ارتباط بین مسیریابها از طریق یک خط نقطه‌به‌نقطه برقرار شده است؛ این خط می‌تواند یک خط استیجاری با هزاران کیلومتر طول باشد. در اینجا فریم توسط مسیریاب اول دریافت شده و بستهٔ جاسازی شده در درون آن، از فیلد دادهٔ فریم استخراج می‌شود. مسیریاب، آدرس درون بسته را (مثلاً آدرس IP

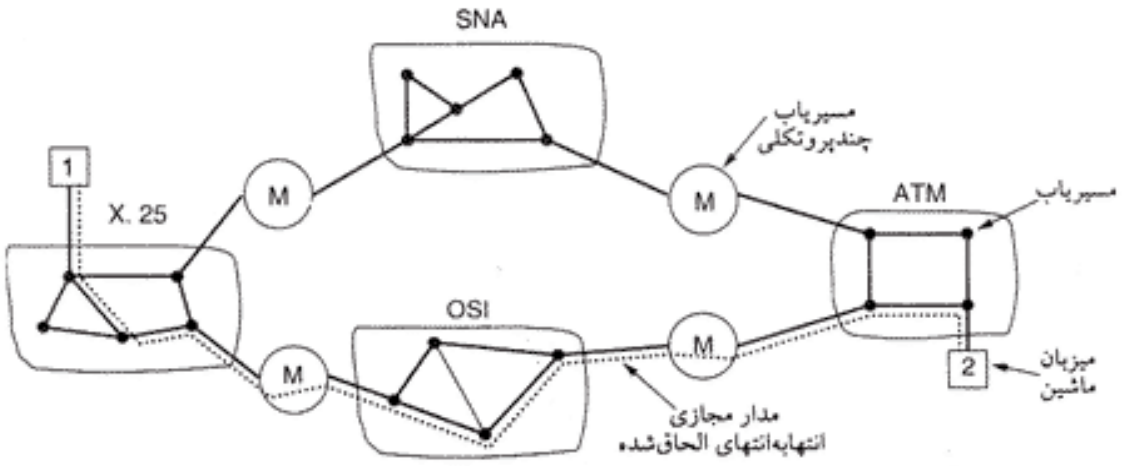
را) بررسی کرده و آنرا در درون جدول مسیریابی خود جستجو می‌نماید؛ سپس براساس این آدرس به نتیجه می‌رسد که باید بسته را به مسیریاب راه دور بفرستد و احتمالاً برای این کار مجبور خواهد شد که مبتنی بر پروتکل این خط ، آنرا در فریم جدید و متفاوتی جاسازی نماید. در مسیریاب مقابل این بسته مجدداً درون فیلد داده از یک فریم اترنت جاسازی شده و بر روی LAN2 ارسال می‌شود.

تفاوت بنیادی بین حالتی که سوئیچ (یا پل) در میان است با وقتی که مسیریاب اتصال شبکه را برقرار کرده ، آنست که در سوئیچ یا پل کلّ یک فریم براساس آدرس MAC آن هدایت و منتقل می‌شود در حالیکه در یک مسیریاب، یک بسته از درون فریم استخراج شده و سپس از آدرس درون بسته برای تصمیم‌گیری در خصوص محل ارسال آن استفاده می‌شود. سوئیچها مجبور نیستند که برای هدایت بسته‌ها درکی از پروتکل لایهٔ شبکه داشته باشند در حالیکه مسیریابها اینگونه‌اند.

۵-۵-۳ مدارات مجازی الحاق‌شده (Concatenated Virtual Circuit)

دو روش برای بهم بندی شبکه‌ها ممکن است: روش الحاق زیرشبکه‌های مدار مجازی به صورت اتصال‌گرا و روش الحاق دیتاگرام. به نوبت این دو روش را بررسی خواهیم کرد ولی در ابتدا به ذکر نکته‌ای می‌پردازیم. در گذشته اکثر شبکه‌های عمومی اتصال‌گرا بودند (و شبکه‌هایی مثل Frame Relay، SNA، 802.16 و ATM هنوز هم اینگونه‌اند). با رشد و مقبولیت سریع اینترنت، شبکهٔ دیتاگرام مُد روز شد ولیکن این تصور که شبکه‌های دیتاگرام ابدی هستند، اشتباه است. در دنیای شبکه‌ها تنها چیزی که جاوید و ابدی می‌ماند «تغییر» است. با رشد شبکه‌های چند رسانه‌ای، احتمالاً اتصال‌گرایی با یکی از اَشکال خود مجدداً به صحنه برمی‌گردد چراکه در شبکه‌های اتصال‌گرا راحتتر می‌توان کیفیت خدمات را تضمین نمود. لذا در ابتدا شبکه‌های اتصال‌گرا را مورد بحث و بررسی قرار می‌دهیم.

در مدل الحاق شبکه‌های مدار مجازی که در شکل ۵-۴۵ نشان داده شده، ایجاد «اتصال» با یک ماشین در شبکه‌ای دوردست، بروشی مشابه با روش معمولی انجام می‌گیرد: زیرشبکه می‌بیند که مقصد در شبکه‌ای دیگر واقع شده لذا یک مدار مجازی با آن مسیریاب که به شبکهٔ مقصد نزدیکتر است، ایجاد می‌کند. از آن مسیریاب نیز یک مدار مجازی با یک «دروازهٔ خارجی» ایجاد می‌شود. (دروازهٔ خارجی یا External Router، یک مسیریاب چند پروتکلی است.) آن «دروازه» نیز مشخصات این مدار مجازی را در جدول خود درج کرده و کار را با ایجاد یک مدار مجازی جدید با مسیریاب زیرشبکهٔ دیگر ادامه می‌دهد. این فرآیند ادامه می‌یابد تا آنکه مدار مجازی به ماشین مقصد ختم شود.



شکل ۵-۴۵. بهم‌بندی شبکه‌ها به کمک مدارات مجازی الحاق‌شده.

هرگاه یک بستهٔ داده، در مسیری جریان یابد، هر یک از دروازه‌های میانی این بسته را رله می‌کنند و در صورت نیاز قالب بسته را تبدیل نموده و شماره‌های مدار مجازی را تغییر می‌دهند. بدیهی است که تمام بسته‌های داده باید به یک ترتیب از دروازه‌ها بگذرند؛ نتیجتاً ترتیب بسته‌های متعلق به یک جریان هرگز به هم نخواهد ریخت.

ویژگی بنیادی این راهکار آنست که دنباله‌ای از مدارات مجازی بین ماشین مبداء و مقصد (از طریق دروازه‌های میانی) ایجاد و تنظیم می‌شود. هر دروازه جدولی را در خود نگاهداری می‌کند که این جدول فهرست مدارات مجازی را که از آن دروازه می‌گذرند و همچنین طریقهٔ مسیریابی و شماره‌های جدید مدار مجازی را تعیین کرده است.

این ساختار زمانی به بهترین نحو کار می‌کند که تمام شبکه‌های میانی از ویژگیهای مشابهی برخوردار باشند. به عنوان مثال اگر تمام آنها تحویل مطمئن بسته‌های لایهٔ شبکه را تضمین کرده باشند، آنگاه جریان بین مبداء و مقصد، مطمئن و قابل اعتماد خواهد بود (به استثنای وقتی که یکی از مسیریابهای میانی از کار بیفتد.) به دلیل مشابه اگر هیچیک از شبکه‌های میانی تحویل مطمئن بسته‌ها را تضمین نکرده باشند، الحاق مدارات مجازی نیز نامطمئن خواهد بود. ولیکن اگر ماشین مبداء بر روی شبکه‌ای باشد که تحویل مطمئن بسته‌ها را تضمین کرده در حالی که یکی از شبکه‌های میانی استعداد از بین بردن بسته‌ای را داشته باشد، الحاق مدارات مجازی منجر به نامطمئن شدن کل مدار مجازی شده و طبیعت خدمات پیش‌بینی شده مثل تحویل مطمئن و حفظ ترتیب بسته‌ها تغییر می‌کند.

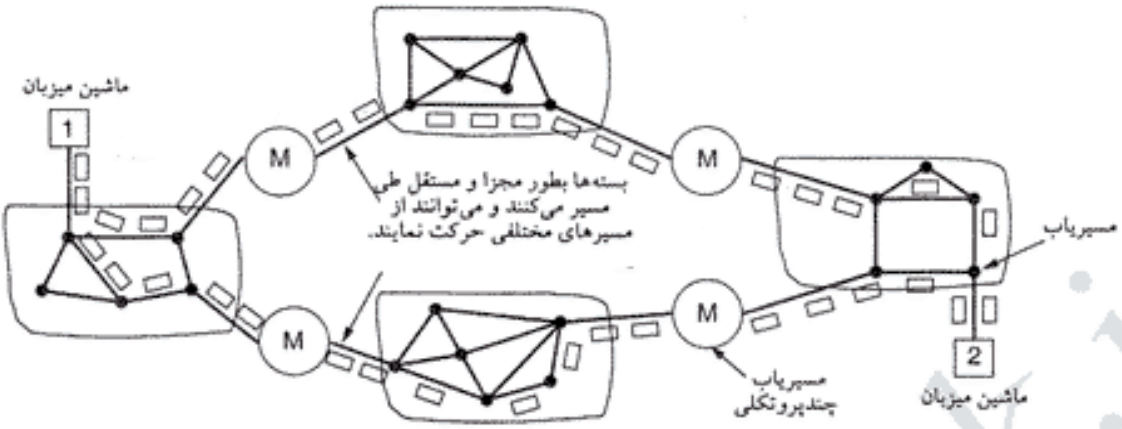
الحاق مدارات مجازی در لایهٔ انتقال نیز رایج است. بویژه این امکان وجود دارد که یک «خط انتقال بیت» (Bit Pipe) بین شبکه‌ای مثل SNA که به یک دروازه منتهی می‌شود و اتصالی TCP با دروازهٔ دیگر دارد، ایجاد نمود. بدین ترتیب می‌توان یک مدار مجازی «انتها به انتها» (End To End) ایجاد کرد در حالی که چندین شبکه با پروتکلهای مختلف در میانهٔ راه قرار گرفته‌اند.

۵-۵-٤ بهم‌بندی شبکه‌های بدون‌اتصال (Connectionless Internetworking)

مدلی دیگر از بهم‌بندی شبکه‌ها، مدل دیتاگرام است؛ (به شکل ۵-۴۶ دقت کنید). در این مدل تنها خدمتی که لایهٔ شبکه به لایهٔ انتقال ارائه می‌دهد آن است که بسته‌های دیتاگرام را بر روی زیرشبکه تزریق کند؛ زیرشبکه نیز حداکثر تلاش خود را در جهت تحویل آن به عمل می‌آورد. در این مدل هیچگونه نشانی از مدار مجازی در سطح لایهٔ شبکه وجود ندارد و فقط شبکه‌ها به هم متصل و ملحق می‌شوند. در این مدل نیازی نیست که بسته‌های متعلق به یک اتصال [تولید شده توسط یک ماشین] به ترتیب از دروازه‌های یکسانی بگذرند. در شکل ۵-۴۶ دیتاگرامهای ارسالی توسط ماشین ۱ که به سوی ماشین ۲ روانه شده‌اند از مسیرهای متفاوتی در شبکه عبور کرده‌اند. تصمیم‌گیری در خصوص مسیر هر بسته، بطور جداگانه انجام می‌شود و این تصمیم‌گیری بستگی به ترافیک لحظهٔ ارسال بسته دارد. در این استراتژی از چندین مسیر بهره گرفته می‌شود و طبعاً پهنای باند بیشتری در مقایسه با مدل الحاق شبکه‌های مدار مجازی حاصل خواهد شد. ولی در مقابل تضمینی در به ترتیب رسیدن بسته‌ها به مقصد وجود ندارد، بلکه فقط فرض بر تحویل بسته‌هاست نه حفظ ترتیب آنها.

مدل شکل ۵-۴۶ به همین سادگی که بنظر می‌رسد نیست. اولین مورد اشکال آنکه، اگر هر یک از شبکه‌های میانی، پروتکل لایهٔ شبکهٔ خاص خودشان را داشته باشد، انتقال بسته از یک شبکه به شبکه‌ای دیگر ممکن نخواهد بود. شاید کسی تصور کند که یک مسیریاب چند پروتکلی قادر به ترجمهٔ قالب بسته‌ها به یکدیگر است در حالی که این تصور زمانی درست است که قالب بسته‌ها نزدیک به یکدیگر بوده و فیلدهای اطلاعاتی هر بسته مشابه باشند و در غیر این صورت چنین تبدیلی ناقص بوده و محکوم به شکست است. به همین دلیل به ندرت تلاش می‌شود چنین تبدیلی انجام گیرد.

دومین مشکل جدی، مسئله آدرس‌دهی است. یک حالت ساده را مدنظر قرار بدهید: یک ماشین بر روی شبکه



شکل ۵-۴۶. بهم‌بندی شبکه‌های بدون‌اتصال.

اینترنت تلاش می‌کند یک بستهٔ IP برای ماشینی بر روی یک شبکهٔ SNA (متصل به اینترنت) بفرستد. آدرسهای IP و SNA متفاوت از هم هستند. آدرسهای IP و SNA باید در هر دو جهت به یکدیگر نگاشته و ترجمه شوند. مضاف بر این، در شبکه‌های متفاوت مفهوم «چیزهایی که قابل آدرس‌دهی هستند» فرق دارد. در IP ماشینهای میزبان (یا در حقیقت کارتهای واسط شبکه) دارای آدرس هستند. در SNA هر «**موجودیت**» (Entity) مثل هر ابزار سخت‌افزاری می‌تواند آدرس داشته باشد. در بهترین حالت هر دروازه باید دارای یک پایگاه اطلاعاتی بوده و بتواند آدرسها را بهم بنگارد (تبدیل کند) ولی همین کار منشاء بروز مشکلات جدی است.

یک نظریهٔ دیگر آن است که یک بستهٔ جهانی و استاندارد طراحی شود و تمام مسیریابها آنرا به رسمیت بشناسند. این راهکار در حقیقت همین IP است که بسته‌های آن به گونه‌ای طراحی شده که هر شبکه‌ای قادر به حمل و هدایت آنهاست. البته ممکن است به نظر برسد IPv4 (پروتکل فعلی اینترنت) نهایتاً تمام ساختارها و پروتکلهای دیگر را از صحنه خارج می‌کند و IPv6 (پروتکل آیندهٔ اینترنت) نیز راه به جایی نمی‌برد و هیچ پروتکل جدیدی ابداع نمی‌شود! ولی تاریخ نشان داده که اینگونه نیست. جلب موافقت عموم افراد برای پذیرش یک قالب واحد بسیار دشوار است چرا که شرکتهای مختلف مصالح و سود خود را در آن می‌بینند که قالب و ساختار اختصاصی و تحت کنترل خود را داشته باشند.

حال بیایید به اختصار مروری بر دو روش شبکه‌بندی داشته باشیم. مدل الحاق شبکه به روش مدار مجازی همان مزایایی را دارد که مدار مجازی در یک زیرشبکه واحد خواهد داشت: یعنی پیشاپیش می‌توان بافرها را از قبل رزرو کرد، ترتیب بسته‌ها حفظ می‌شود، سرآیند کوتاهتری برای بسته‌ها نیاز است و از مشکلاتی که ناشی از تکراری شدن بسته‌هایی که با تأخیر می‌رسند، احتراز می‌شود.

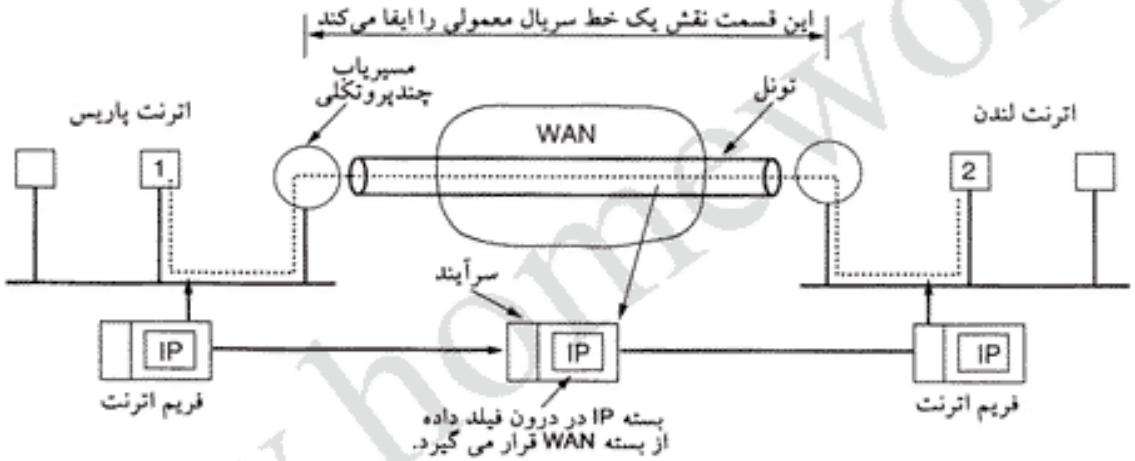
این روش معایبی نیز دارد: در مسیریابها به فضای قابل‌توجهی برای نگاهداری جدول اتصالات باز نیاز است، برای احتراز از مناطق مواجه با ازدحام مکانیزم مسیریابی و مسیرها تغییر نمی‌کند و مسیرها نسبت به خرابی مسیریابها بسیار آسیب‌پذیر هستند. همچنین این اشکال بزرگ وجود دارد که اتصال چنین شبکه‌ای به یک شبکهٔ نامطمئن دیتاگرام اگر غیرممکن نباشد بسیار دشوار است.

ویژگی بهم‌بندی شبکه‌ها به روش دیتاگرام، دقیقاً مشابه با ویژگی زیرشبکه‌های دیتاگرام است: استعداد بروز ازدحام دارد ولی در عوض قابلیت بیشتری نیز برای رفع آن دارد، در مواجهه با خرابی یک مسیریاب قابلیت تحمل بیشتری از خود نشان می‌دهد و به سرآیند طولانی‌تری برای هر بستهٔ نیاز است؛ استفاده از انواع الگوریتمهای وقفی و پویای مسیریابی میسّر است و در مجموع تمام مزایا و معایب یک زیرشبکه واحد و منفرد را بطور مشابه دارد.

بزرگترین مزیت روش دیتاگرام برای بهم‌بندی شبکه آن است که می‌توان این روش را برای وصل هر شبکه‌ای که در درون از مدار مجازی بهره گرفته، استفاده کرد. بسیاری از شبکه‌های LAN، شبکه‌های متحرک (Mobile) مثل شبکهٔ ناوگانهای هوایی و دریایی و حتی برخی از شبکه‌های WAN، در ردهٔ شبکه‌های دیتاگرام قرار می‌گیرند و هر گاه استراتژی بهم‌بندی این شبکه‌ها مبتنی بر مدارات مجازی باشد، مشکلات جدی بروز خواهد کرد.

۵-۵-۵ ایجاد تونل (Tunneling)

اتصال دو شبکهٔ مختلف در حالت کلی بسیار سخت است ولیکن یک حالت خاص و رایج وجود دارد که راهگشا و قابل مدیریت است. این حالت زمانی اتفاق می‌افتد که ماشینهای مبداء و مقصد بر روی شبکه‌هایی از یک نوع هستند ولی یک شبکهٔ متفاوت در میان آنها قرار گرفته است. به عنوان مثال یک بانک بین‌المللی را مدنظر قرار دهید که یک شبکهٔ اترنت مبتنی بر پروتکل TCP/IP در پاریس و همچنین یک اترنت با پروتکل TCP/IP در لندن دارد ولی به نحوی که در شکل ۵-۴۷ نشان داده شده، یک شبکه بغیر از IP (مثل ATM) در میان آنها قرار گرفته است.



شکل ۵-۴۷. ارسال یک بسته از طریق ایجاد تونل بین پاریس و لندن.

راهکار حل این مشکل تکنیکی به نام «**ایجاد تونل**» است. برای ارسال یک بستهٔ IP به ماشین ۲، ماشین ۱، بسته‌ای حاوی آدرس IP ماشین ۲ ساخته و آنرا در درون فریم شبکهٔ اترنت قرار داده و آدرس مسیریاب چند پروتکلی واقع در پاریس را در فیلد آدرس این فریم درج کرده و آن را بر روی اترنت قرار می‌دهد. هرگاه این «مسیریاب چندپروتکلی» این فریم را دریافت کند بستهٔ IP را از درون فریم استخراج کرده و آن را در فیلد حمل داده (Payload) از بستهٔ لایهٔ شبکهٔ WAN قرار داده و آدرس بستهٔ جدید را آدرس مسیریاب چندپروتکلی واقع در لندن قرار می‌دهد. [به عبارتی دو بستهٔ تودرتو تشکیل می‌شود که آدرس بستهٔ درونی، آدرس ماشین مقصد و آدرس بستهٔ بیرونی آدرس مسیریاب لندن است.] هر گاه این بسته به مسیریاب واقع در لندن برسد آن مسیریاب بستهٔ IP اصلی را از درون آن استخراج کرده و آنرا با قرار دادن در یک فریم اترنت برای ماشین ۲ می‌فرستد.

شبکهٔ WAN را می‌توان یک تونل بزرگ در نظر گرفت که از یک مسیریاب چندپروتکلی شروع و به مسیریابی دیگر از همین نوع، در طرف دیگر WAN ختم می‌شود. یک بستهٔ IP در حالی که درون یک بستهٔ دیگر جا گرفته از یک طرف تونل شروع به طی مسیر به طرف دیگر تونل می‌کند و جای هیچگونه نگرانی در خصوص ساختار WAN وجود ندارد [چرا که تنها کاری که WAN در این میان انجام می‌دهد آنست که مبتنی بر پروتکل فعلی خود، بستهٔ IP را به عنوان یک قطعه دادهٔ خام، در یک نقطه دریافت و در نقطهٔ دیگر تحویل می‌دهد.-م] در این مثال در خصوص ماشینهای هر یک از شبکه‌های اترنت نیز نگرانی وجود ندارد [چون هر دو از یک نوع و مبتنی بر پروتکل

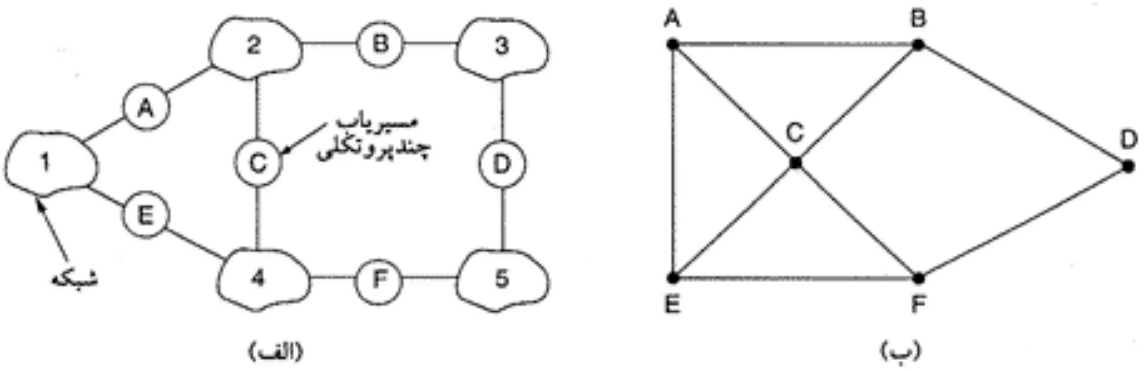
مشابه هستند]. فقط «مسیریاب چندپروتکلی» است که باید هم بسته‌های IP و هم بسته‌های WAN را بفهمد و مدیریت کند. در حقیقت کل شبکه‌ای که در میان این دو مسیریاب چندپروتکلی قرار گرفته نقشی شبیه به یک خط سریال ایفاء می‌کند.

شاید یک تمثیل بتواند مفهوم **تونل** را روشنتر کند. شخصی را در نظر بگیرید که با خودروی خودش از پاریس به لندن مسافرت می‌کند. در فرانسه او با خودروی خود تا کنار کانال انگلیس رانندگی می‌کند ولی با رسیدن به این کانال (که رانندگی در آن ممنوع است)، خودروی او در یک قطار سریع‌السیر بار شده و از طریق قطار به انگلیس منتقل می‌شود. در حقیقت، به نحوی که در شکل ۵-۴۸ نشان داده شده، این خودرو همانند یک محمولهٔ عادی جابجا می‌شود. در طرف دیگر، این خودرو مجدداً بر زمین گذاشته شده و اجازه می‌یابد با نیروی محرکهٔ خودش در جاده‌های انگلیس حرکت نماید. ایجاد تونل برای بسته‌ها از طریق یک شبکهٔ خارجی مشابه با همین عملکرد است.

شکل ۵-۴۸. تونل کردن (Tunneling) یک خودرو از فرانسه به انگلستان.

۶-۵-۵ مسیریابی بین شبکه‌های بهم‌متصل

مسیریابی بین چند شبکهٔ متصل‌بهم مشابه با مسیریابی در یک زیرشبکه واحد است (البته با پیچیدگی‌های بیشتر). به عنوان مثال ساختار ارتباطی شبکهٔ الف-۵-۴۹ را در نظر بگیرید که در آن پنج شبکه توسط شش مسیریاب (احتمالاً از نوع چندپروتکلی) بهم متصل شده‌اند. تشکیل مدل گراف از این وضعیت پیچیده است چراکه در گراف، هر مسیریاب باید مستقیماً با مسیریاب دیگر در ارتباط باشد (بسته بفرستد) در حالیکه اینجا مسیریابها مستقیماً به یک شبکهٔ میانی متصلند. مثلاً مسیریاب B در شکل ۵-۴۹-الف می‌تواند از طریق شبکهٔ ۲ به مسیریابهای A و C دسترسی داشته باشد، همچنین مسیریاب D از طریق شبکهٔ ۳ به B دسترسی دارد. اگر هر یک از شبکه‌ها را به مثابهٔ یک خط سریال فرض نماییم، گراف شکل ۵-۴۹-ب بدست می‌آید.



شکل ۵-۴۹. (الف) «شبکه‌ای از شبکه‌ها» (ب) گراف متناظر.

به محض آنکه گراف شبکه تشکیل شد می‌توان یکی از الگوریتمهای شناخته شدهٔ مسیریابی مثل «الگوریتم بردار فاصله» یا «الگوریتم حالت لینک» (Link State Algorithm) را بر روی مجموعهٔ این مسیریابهای چندپروتکلی اعمال کرد. در نتیجه یک الگوریتم مسیریابی دوسطحی ایجاد می‌شود: در درون هر شبکه از یک «**پروتکل مسیریابی درونی**» (Interior Gateway Protocol) استفاده می‌شود در حالی که بین شبکه‌ها از «**پروتکل مسیریابی بیرونی**» (Exterior Gateway Protocol) بهره گرفته می‌شود. (اصطلاح Gateway نام قدیمی مسیریاب است.) در حقیقت، چون شبکه‌ها مستقل هستند ممکن است از الگوریتمهای مسیریابی مختلفی در درون شبکه بهره گرفته باشند. از آنجایی که با بهم‌بندی شبکه‌ها هر شبکه استقلال داخلی خود را حفظ می‌کند، به هر یک از این شبکه‌های مستقل «**سیستم خودمختار**» یا به اختصار AS گفته می‌شود.

در چنین شبکه‌ای، یک بستهٔ نوعی از یک ماشین بر روی LAN مسیر خود را آغاز کرده و به آدرس «مسیریاب چندپروتکلی» متصل به آن LAN ارسال می‌شود (با قرار دادن آدرس آن مسیریاب در فیلد آدرس از فریم MAC). پس از آن که بسته به مسیریاب رسید، نرم‌افزار آن مسیریاب به کمک جدول مسیریابی خود مشخص می‌کند که این بسته باید به سوی کدامیک از مسیریابهای دیگر هدایت شود. اگر بسته رابتوان با همان پروتکل لایهٔ شبکه (که بسته براساس آن تولید شده) به مقصد رساند، این بسته مستقیماً هدایت و ارسال می‌شود. در غیر این صورت از مکانیزم ایجاد تونل (Tunneling) استفاده شده و کل بسته درون یک بستهٔ متناسب با پروتکل مسیریابهای میانی جاسازی و ارسال می‌گردد. این فرآیند آنقدر تکرار می‌شود تا بسته به شبکهٔ مقصد برسد.

یکی از تفاوتهای بین مسیریابی درون یک شبکهٔ خودمختار (Intranetwork Routing) و مسیریابی بین چند شبکهٔ خودمختار آن است که در مورد دوم ممکن است بسته‌ها نیاز به تردّد از مرزهای بین‌المللی کشورها داشته باشند. در چنین حالتی، ملاحظات قانونی متعدد مطرح خواهد شد چرا که مثلاً طبق قوانین کشور سوئد، خارج کردن اطلاعات فردی شهروندان سوئدی از کشور مجاز نیست. یا مثلاً طبق قوانین کشور کانادا، ترافیک داده‌هایی که مبداء آنها در کشور کانادا و مقصد آنها نیز در کانادا است نباید از کشور خارج شود. طبق این قانون ترافیک داده‌هایی که مبداء آن شهر «وینزور» یا «انتاریو» و مقصد آن «ونکوور» است نمی‌تواند از طریق مسیر نزدیکتری که از «دیترویت» می‌گذرد، مسیریابی و هدایت شود حتی اگر استفاده از این مسیر سریعتر و ارزانتر باشد.

تفاوت دیگر بین مسیریابی درونی و مسیریابی بیرونی (Interior/Exterior Routing) موضوع هزینه (قیمت) است. در یک شبکهٔ خودمختار و واحد، بطور معمول از یک الگوریتم مشخص برای تعیین هزینه استفاده می‌شود ولیکن در شبکه‌های مختلف که مدیریت متفاوتی دارند ممکن است یک مسیر از لحاظ قیمت از مسیر دیگر مقرون به صرفه‌تر باشد. همینطور سطح کیفیت خدمات عرضه شده در شبکه‌های متفاوت فرق می‌کند و همین مورد ممکن است دلیل انتخاب یک مسیر از بین دیگر مسیرها باشد.

۷-۵-۵ قطعه‌قطعه‌سازی بسته‌ها (Fragmentation)

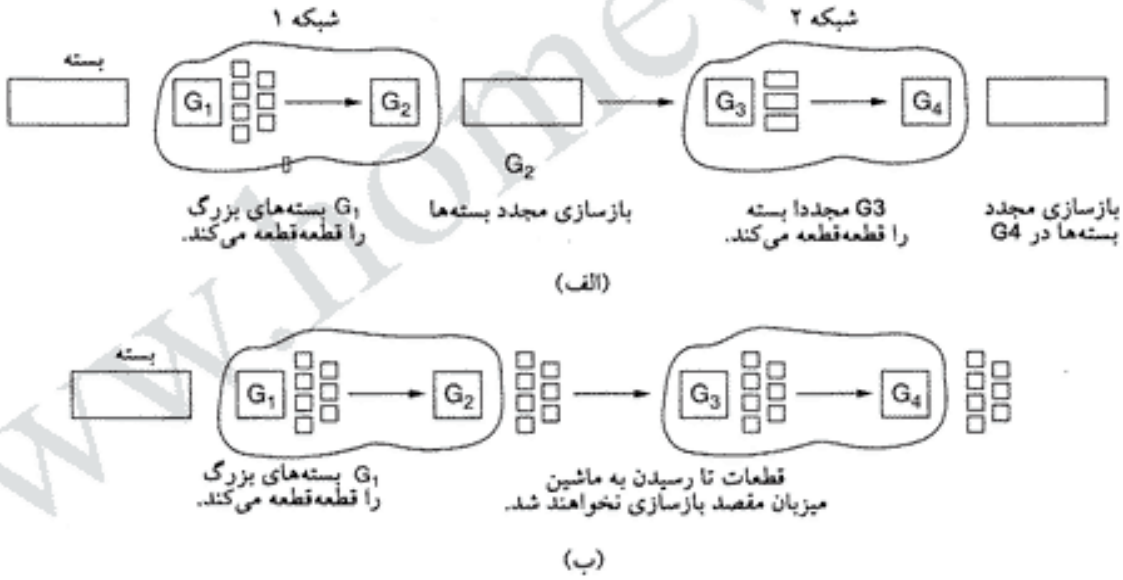
در هر شبکه اندازهٔ هر بسته نمی‌تواند از یک حد مُجاز بیشتر باشد. این محدودیت، علل مختلفی دارد که از این میان می‌توان به موارد ذیل اشاره کرد:

۱. سخت‌افزار (مثلاً طول فریم اترنت نمی‌تواند از حدود ۱۵۰۰ بایت بیشتر باشد.)
۲. سیستم عامل (مثلاً بافرها ۵۱۲ بایتی هستند.)
۳. پروتکل (مثلاً تعداد بیتهای فیلدی که میزان داده‌های موجود در هر بسته را مشخص می‌کند کم است.)
۴. سازگاری با برخی از استانداردهای بین‌المللی
۵. تمایل به کاهش حجم ارسال مجدد داده‌هایی که در اثر خطای انتقال از بین می‌روند
۶. اجتناب از اشغال بیش از حد کانال توسط یک بستهٔ بزرگ

نتیجهٔ این عوامل آن می‌شود که دست طراحان شبکه در انتخاب حداکثر طول بسته‌ها باز نیست. حداکثر طول داده‌هایی که می‌تواند درون یک بسته قرار بگیرد از ۴۸ بایت (در سلولهای ATM) تا ۶۵۵۱۵ بایت (در بسته‌های IP) متغیر است اگرچه طول داده‌ها در لایه‌های بالاتر، می‌تواند از این هم بیشتر باشد.

یکی از مشکلات بدیهی، زمانی رخ می‌دهد که یک بستهٔ بزرگ بخواهد به شبکه‌ای وارد شود که طول حداکثر بستهٔ آن کوچک است. یک راه حل آن است که از همان ابتدا این اطمینان حاصل شود که چنین مشکلی پیش نمی‌آید. به عبارت دیگر در اتصال شبکه‌ها، از یک الگوریتم مسیریابی استفاده شود که از ارسال بسته‌ها به شبکه‌ای که از عهدهٔ پذیرش آن بر نمی‌آید اجتناب کند، ولیکن این راه حل همیشه عملی نیست: اگر همهٔ بسته‌های تولید شده توسط مبداء، دارای اندازه‌ای باشند که شبکهٔ مقصد از عهدهٔ دریافت آن بر نمی‌آید چه اتفاقی می‌افتد؟ الگوریتم مسیریابی به ندرت می‌تواند چنین مقصدی را نادیده گرفته و آنرا کنار بگذارد.

اصولاً در چنین مواقعی می‌توان برای حل مشکل، به «دروازه» (Gateway) اجازه داد که بسته‌ها را به چند «قطعه» (Fragment) شکسته و هر یک از این قطعات را در پوشش یک بستهٔ مستقل ارسال کند؛ ولیکن والدین کودکان خردسال به خوبی می‌دانند که تکه‌تکه کردن اشیاء بزرگ به قطعات کوچک ساده‌تر از سرهم آنهاست!!! (فیزیکدانان برای این پدیده نامی انتخاب کرده‌اند: قانون دوم ترمودینامیک) شبکه‌های سوئیچ بسته نیز برای بازسازی قطعات به بستهٔ اصلی با مشکل روبرو هستند.



شکل ۵-۵۰. (الف) قطعه‌قطعه کردن نامرئی (شفاف) (ب) قطعه‌قطعه کردن غیرشفاف (مرئی).

برای بازسازی قطعات و تشکیل بستهٔ اصلی، دو استراتژی متضاد قابل اعمال است: استراتژی اول آن است که قطعه‌قطعه‌سازی بسته‌ها در شبکه‌ای که بسته‌های کوچک دارد به گونه انجام شود که این عمل از دید مقصد نهایی بسته، پنهان بماند. در این راهکار که در شکل ۵-۵۰-الف نشان داده شده است، در هر شبکه با بستهٔ کوچک یک «دروازه» (یا به عبارتی یک مسیریاب خاص) وجود دارد که واسط دیگر شبکه‌هاست. وقتی بسته‌ای با طول بیش از حدّ به این دروازه می‌رسد، آن را قطعه‌قطعه می‌کند؛ سپس هر یک از قطعات به آدرس دروازهٔ خروجی در مقصد ارسال شده و در آنجا قطعات به شکل اصلی بازسازی می‌شوند. بدین ترتیب عبور بسته‌های بزرگ از شبکه‌ای با بستهٔ کوچک، از دید شبکه‌های بیرونی پنهان می‌ماند؛ بعبارتی شبکه‌های بیرونی از قطعه‌قطعه شدن بسته‌ها آگاه نمی‌شوند. به عنوان مثال، شبکهٔ ATM دارای سخت‌افزار خاصی است که به صورت نامرئی، بسته‌ها را به سلولهای

۵۳ بایتی شکسته و در طرف مقابل آنها را به بستهٔ اصلی بازسازی می‌نماید. در شبکهٔ ATM به عمل شکستن بسته‌ها اصطلاحاً «**قطعه‌بندی**» (Segmentation) گفته می‌شود و مفهومی معادل با آنچه که در بالا بدان اشاره شد دارد، بلکه فقط جزئیات پیاده‌سازی آن متفاوت است.

استراتژی قطعه‌قطعه کردن بسته به صورت نامرئی، اگرچه ساده و سرراست است ولیکن مشکلاتی را در بر دارد. اولین مورد آن است که دروازهٔ خروجی باید تشخیص بدهد که آیا تمام قطعات یک بسته را دریافت کرده است یا آنکه هنوز قطعاتی در راه هستند و به همین دلیل باید یک بیت خاص پایان دنبالهٔ قطعات هر بسته را مشخص نماید. مورد دیگر *آن است که تمام قطعات یک بسته بایستی از یک دروازهٔ مشابه و یکسان خارج شوند.* وقتی اجازه نمی‌دهیم که قطعات یک بسته برای رسیدن به مقصد نهایی خود از مسیرهای مختلفی حرکت کنند، بخشی از کارآیی و بهینگی را از دست خواهیم داد. آخرین اشکال آنست که وقتی یک بستهٔ بزرگ مجبور است مکرراً از شبکه‌هایی بگذرد که آن را قطعه‌قطعه و بازسازی می‌کنند، سربار نسبتاً زیادی تحمیل خواهد شد. به هر تقدیر شبکه‌ای مثل ATM به قطعه‌قطعه‌سازی نامرئی [استراتژی فوق‌الذکر] نیاز دارد.

استراتژی دیگر برای قطعه‌قطعه کردن بسته‌ها آنست که از بازسازی بسته‌ها در دروازه‌های میانی اجتناب شود: هر گاه بسته‌ای قطعه قطعه شد، با هر یک از قطعات به مثابهٔ یک بستهٔ واقعی و اصلی رفتار شود. در این استراتژی به نحوی که در شکل ۵-۵۰-ب نشان داده شده تمام قطعات به همان نحو از دروازهٔ خروجی شبکه گذر می‌کنند. بازسازی بسته‌ها فقط در ماشین مقصد انجام می‌شود. IP بهمین نحو عمل می‌کند.[۱]

قطعه‌قطعه‌سازی غیرشفاف نیز با اشکالاتی مواجه است. به عنوان مثال، هر ماشین باید قادر به بازسازی بسته باشد. مشکل دیگر آنست که وقتی یک بستهٔ بزرگ به قطعات کوچک تقسیم می‌شود، سربار داده‌ها افزایش می‌یابد زیرا هر قطعه نیاز به سرآیند مستقل دارد، در حالی که در استراتژی اول به محض خروج قطعات از شبکه‌ای با بستهٔ کوچک، سربار تحمیل شده حذف می‌شود ولیکن در این روش سربار ناشی از سرآیند اضافی، تا خاتمهٔ مسیر باقی خواهد ماند. مزیت روش قطعه‌قطعه‌سازی غیرشفاف آن است که قطعات هر بسته می‌توانند از مسیری مجزا و دروازه‌های خروجی متفاوت به سوی مقصد نهایی خود هدایت شوند و طبعاً کارآیی بالاتر و بهینه‌تری حاصل خواهد شد. البته اگر شبکه‌های مختلف طبق مدل مدار مجازی به یکدیگر متصل و ملحق شده باشند [بخش ۵-۵-۳] این مزیت بکار نخواهد آمد.

هرگاه بسته‌ای قطعه‌قطعه شود، قطعات آن باید به نحوی شماره‌گذاری شوند که بتوان ترتیب اصلی داده‌ها را بازیابی کرد. یکی از روشهای شماره‌گذاری، روش درختی است: اگر بستهٔ شمارهٔ 0 نیاز به قطعه شدن داشته باشد قطعات به صورت 0.0 , 0.1 , 0.2 و به همین ترتیب، شماره‌گذاری می‌شوند. اگر هر یک از این قطعات، باز هم نیاز به شکسته شدن داشته باشند ترتیب شماره‌ها (از چپ به راست) به صورت 0.0.0 , 0.0.1 , 0.0.2 , ... , 0.1.0 , 0.1.1 , 0.1.2 , 0.1.3, ... خواهد بود. اگر برای این منظور به تعداد کافی فیلد در سرآیند بسته پیش‌بینی شده باشد ، الگوی فوق این اطمینان را می‌دهد که بتوان بسته‌ها را در مقصد بازسازی کرد. (حتی اگر قطعات به ترتیب دریافت نشوند.)

ولی اگر در یکی از شبکه‌های بین راه قطعه‌ای گم شود یا حذف گردد نیاز است که مجدداً کل بسته از مبداء آن ارسال و از نو قطعه قطعه و ارسال شود. فرض کنید که یک بستهٔ ۱۰۲۴ بیتی در ابتدا به چهار قطعهٔ مساوی با شماره‌های 0.0 ، 0.1 ، 0.2 و 0.3 تقسیم شده باشد. اگر قطعهٔ 0.1 از دست برود ولی بقیه قطعات سالم به مقصد

---

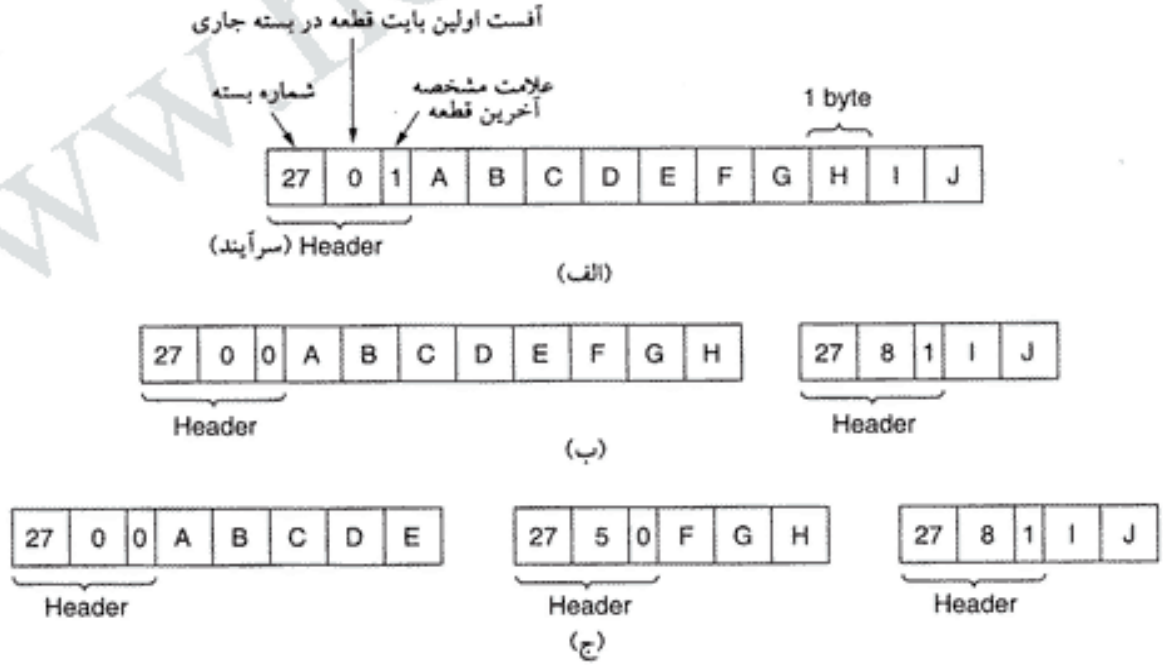
۱. بدین استراتژی ، قطعه قطعه سازی غیرشفاف و مرئی (Nontransparent Fargmentation) گفته می‌شود. هرگاه بسته‌ای در هر نقطه از مسیر قطعه‌قطعه شد هیچکسی آنها را بازسازی نخواهد کرد مگر ماشین مقصدنهایی. ـم

برسند، پس از مدتی مهلت فرستندهٔ بسته به سر آمده و کل بسته را از نو ارسال می‌کند. حال فرض کنید که به ناگاه محدودیت طول بسته از ۲۵۶ بیت به ۵۱۲ بیت افزایش پیدا کرده و در اینجا بسته به جای آنکه ۴ قسمت شود به دو قطعه تقسیم گردد. حال وقتی بستهٔ شمارهٔ 0.1 به مقصد می‌رسد گیرنده تصور می‌کند که تمام چهار قطعه‌ای که منتظر آن بوده، تکمیل شده و بسته را به صورت اشتباه بازسازی می‌کند!

یک روش شماره‌گذاری متفاوت و بهتر آن است که در پروتکل شبکه ، یک اندازهٔ پایه و حداقل برای بسته‌ها تعریف شود و این اندازه به قدری کوچک باشد که بسته بتواند از هر شبکه‌ای عبور نماید. هرگاه بسته‌ای قطعه‌قطعه می‌شود، اندازهٔ تمام قطعات به مقدار پایه تنظیم خواهد شد (به استثنای قطعهٔ آخر که ممکن است کوچکتر باشد). در سرآیند هر یک از قطعات باید شمارهٔ بستهٔ اصلی و همچنین شمارهٔ قطعه مشخص شده باشد. همچنین باید در سرآیند قطعه یک بیت در نظر گرفته شود تا بتوان آخرین قطعهٔ هر بسته را مشخص نمود. (یعنی هر بسته را فقط یکبار می‌توان قطعه قطعه کرد.) در این روش، در سرآیند هر بسته به دو فیلد شماره ترتیب نیاز است: یکی برای شمارهٔ بستهٔ اصلی و دیگری برای شمارهٔ قطعه.

می‌توان یک حالت بینابین برای شماره‌گذاری بسته‌ها انتخاب کرد: به جای آن که برای هر قطعه شمارهٔ آن در سرآیند هر قطعه درج شود، آفست آن قطعه در بستهٔ اصلی، مشخص شود. [یعنی آن که مشخص شود قطعهٔ جاری در کجای بستهٔ اصلی قرار می‌گیرد.] اگر مطابق شکل ۵-۵۱ بجای شمارهٔ هر قطعه، آفست محل قرار گرفتن قطعه در بستهٔ اصلی در نظر گرفته شود می‌توان بسته را به هر اندازهٔ ممکن (حتی یک بایت) کوچک کرد.

در برخی از پروتکلهای شبکه بطور عام از این روش استفاده شده است حتی تا جایی که کل داده‌های ارسالی بر روی یک مدار مجازی به عنوان یک بستهٔ بسیار بزرگ تلقی می‌شود و شمارهٔ هر قطعه، شمارهٔ ترتیب «اولین بایت قطعه» نسبت به «اولین بایت بسته» فرض می‌شود. [یعنی محل قرار گرفتن قطعه نسبت به شماره اولین بایت بستهٔ اصلی مشخص می‌شود.]



شکل ۵-۵۱. قطعه‌قطعه‌سازی بسته‌ها با فرض آنکه مبنای پایهٔ طول داده‌ها ۱ بایت باشد. (الف) بستهٔ اصلی حاوی ده بایت داده. (ب) مجموعهٔ قطعات پس از عبور از شبکه‌ای که در آن طول حداکثر بسته‌ها با احتساب سرآیند، ۸ بایت است. (ج) مجموعهٔ قطعات پس از عبور از «دروازه‌ای» (Gateway) که در آن طول حداکثر بسته‌ها ، ۵ بایت است.

۶ـ۵ لایهٔ شبکه در اینترنت

قبل از آنکه به توصیف لایهٔ شبکه در اینترنت بپردازیم، مروری بر اصول طراحی آن در گذشته و دلایل موفقیت آن در حال حاضر، خالی از لطف نیست. اصولی که به نظر می‌رسد اکنون به دست فراموشی سپرده شده‌اند. این اصول و قواعد در سند RFC 1958 تشریح شده‌اند و مطالعه آن بسیار ارزشمند است. (مطالعهٔ آن برای طراحان پروتکل باید الزامی شود و در پایان نیز از آنان براساس همین سند امتحان به عمل آید!) این RFC بشدت تحت تأثیر افکاری است که دو محقق Clark (۱۹۸۸) و Saltzer (همکاران و ۱۹۸۴) ارائه نموده‌اند. در اینجا به اختصار ده مورد از اصول اساسی طراحی پروتکل لایهٔ شبکه را (به ترتیب اهمیت) بررسی می‌نماییم:

۱. **اطمینان از عملکرد صحیح:** طراحی یا استاندارد را هیچگاه نهایی و مختومه فرض نکنید مگر آنکه چندین نسخهٔ اولیه و آزمایشی (پروتوتایپ) آن بتوانند با یکدیگر مبادلهٔ داده نمایند. بسیار اتفاق افتاده که طراحان شبکه یک استاندارد ۱۰۰۰ صفحه‌ای تدوین و آنرا به تایید رسانده‌اند ولی بعداً متوجه اشکالات اساسی و عدم عملکرد آن شده و مجبور به نوشتن نسخهٔ ۱.۱ استاندارد خود شده‌اند. این روش به نتیجه نخواهد رسید.

۲. **پروتکل را ساده طراحی کنید:** در هر کجا که تردید دارید، از ساده‌ترین راه حل بهره بگیرید. «ویلیام اُوکَم» این اصل را در قرن چهاردهم بیان کرده است. در یک عبارت امروزی و مدرن: «حداقل ویژگیهای زائد». اگر داشتن یک ویژگی، حیاتی و اساسی نیست آنرا رها کنید بالاخص زمانی که این ویژگی را می‌توان براساس ترکیبی از ویژگیهای دیگر بدست آورد.

۳. **تصمیمات روشن و شفاف بگیرید:** اگر برای انجام یک کار چندین راه حل پیش رو دارید فقط یکی را انتخاب نمایید. اگر دو یا چند روش را مدّنظر قرار بدهید برای خود مشکل تراشیده‌اید. امروزه بسیاری از استانداردها دارای گزینه‌های متعدد، حالات و پارامترهای مختلف هستند چراکه هر یک از طراحان ، بر آنکه روش آنها بهترین است، پافشاری کرده‌اند. طراحان باید با قدرت در برابر این گرایشات فردی مقاومت کرده و فقط بگویند نه!!

۴. **طراحی شما ماجولار باشد:** این اصل بدین نظریه منتهی خواهد شد که باید پشته‌ای از پروتکلها را داشت و هر لایه مستقل از دیگر لایه‌ها باشد. بدین ترتیب هر گاه نیاز شد یک ماجول یا یک لایه تغییر کند، بقیهٔ لایه‌ها تحت تأثیر قرار نخواهند گرفت.

۵. **ناهمگونی عناصر را مدّنظر قرار بدهید:** در یک شبکهٔ بزرگ، انواع سخت‌افزار، امکانات مخابراتی و برنامه‌های کاربردی پیدا می‌شود. برای آن که بتوان همهٔ آنها را به خدمت گرفت و اداره کرد، طراحی شبکه باید ساده، عمومی و قابل انعطاف باشد.

۶. **از گزینه‌ها و پارامترهای ثابت پرهیز نمایید:** اگر در جایی الزاماً به تعریف پارامتر نیاز دارید (مثلاً تعریف طول حداکثر بسته‌ها) بهترین کار آن است که به جای تعریف پارامترهای ثابت اجازه بدهید فرستنده و گیرنده با یکدیگر مذاکره و بر سر مقدار آن توافق کنند.

۷. **به دنبال طراحی خوب باشید:** لازم نیست که یک طراحی خوب ایده‌آل و بی‌نقص باشد: بسیاری از طراحان، طرح خوبی را در اختیار دارند ولی طرح آنان نمی‌تواند از عهدهٔ برخی از موارد نادر و عجیب برآید. به جای آن که اینگونه طرحها را بهم بریزید بایستی در پی یک طراحی خوب بوده و از خود در مقابل خواسته‌های عجیب و غریب سلب مسئولیت نمایید.

۸. **هنگام «ارسال» سخت گیر و دقیق و هنگام «دریافت» با تحمل باشید:** به عبارتی دیگر فقط بسته‌هایی را

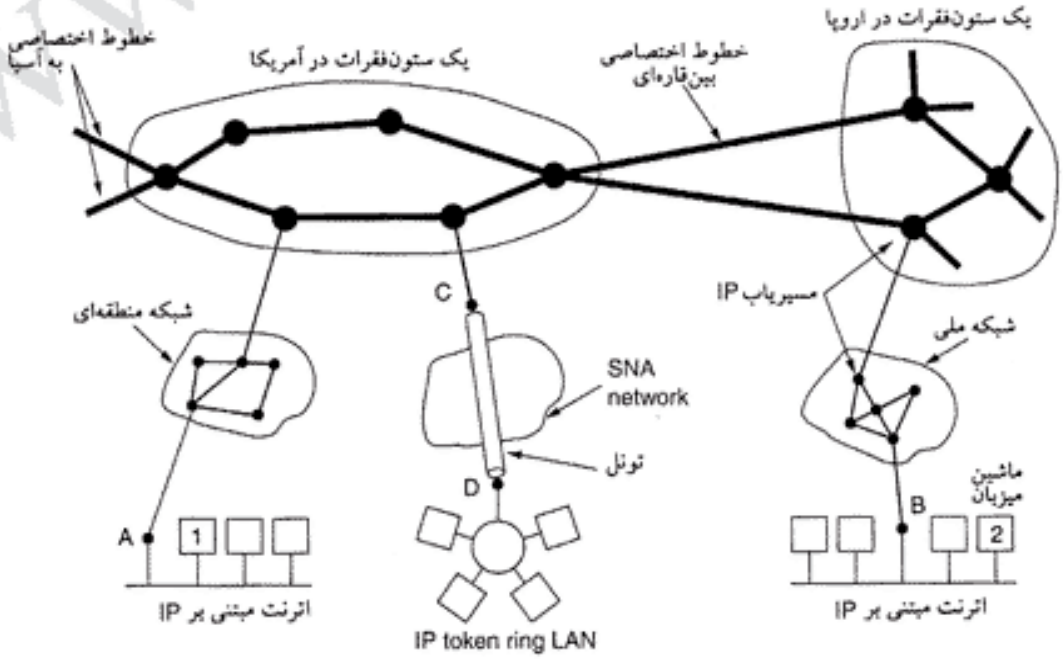
بر روی شبکه بفرستید که دقیقاً منطبق با استاندارد است ولی انتظار آن را داشته باشید که بسته‌های دریافتی انطباق کامل با استاندارد نداشته باشد.

۹. **در اندیشهٔ قابلیت گسترش و توسعه‌پذیری باشید:** اگر یک سیستم مجبور به مدیریت میلیونها ماشین و میلیاردها کاربر است، استفاده از هیچ نوع پایگاه اطلاعات مرکزی قابل تحمل و به صلاح نیست و باید بار آن تا حد ممکن بر روی منابع موجود پخش شود.

۱۰. **به کارآیی و هزینه دقت داشته باشید:** اگر شبکه‌ای کارآیی پایین یا هزینهٔ سرسام‌آور داشته باشد هیچکس از آن استقبال نخواهد کرد.

حال اصول کلی طراحی را کنار گذاشته و به جزئیات لایهٔ شبکه در اینترنت خواهیم پرداخت. از دیدگاه لایهٔ شبکه، اینترنت را می‌توان مجموعه‌ای از زیرشبکه‌ها یا اصطلاحاً «شبکه‌های خودمختار» در نظر گرفت که بهم متصل شده‌اند. هیچ ساختار مشخص و قطعی بر اینترنت حاکم نیست ولی چندین ستون‌فقرات (Backbone) برای آن وجود دارد. ستون‌فقرات از خطوط با پهنای باند بسیار بالا و مسیریابهای سریع تشکیل شده است. شبکه‌های منطقه‌ای (Regional Network) به ستون‌فقرات اینترنت متصل می‌شوند و شبکه‌های محلی دانشگاهها، شرکتها و مؤسسات ارائه دهندهٔ خدمات اینترنت (ISP) با شبکه‌های منطقه‌ای در ارتباط هستند. نمادی از این ساختار نسبتاً سلسله‌مراتبی در شکل ۵-۵۲ نشان داده شده است.

آنچه که تمام اجزاء شبکهٔ اینترنت را به هم چسبانده است همان پروتکل IP است. (IP/Internet Protocol) برخلاف بسیاری از پروتکلهای قدیمی لایهٔ شبکه، این پروتکل از صفر طراحی شده و از همان ابتدا در تفکر وصل شبکه‌ها به یکدیگر بوده است. برای آن که بتوانید وظیفهٔ لایهٔ شبکه را در ذهن خود مجسم کنید بدان بیندیشید که این لایه حداکثر تلاش خود را می‌کند تا یک دیتاگرام را از مبداء به مقصد برساند. [دیتاگرام را یک تودهٔ اطلاعات مستقل و دارای هویت در نظر بگیرید.] IP رسیدن بسته‌ها را تضمین نمی‌کند (بلکه حداکثر تلاش خود را مصروف آن می‌کند) و اینکه آیا ماشینهای مبداء و مقصد بر روی یک شبکه واقعند یا آن که شبکه‌های دیگری در این بین قرار گرفته‌اند، مهم نیست.

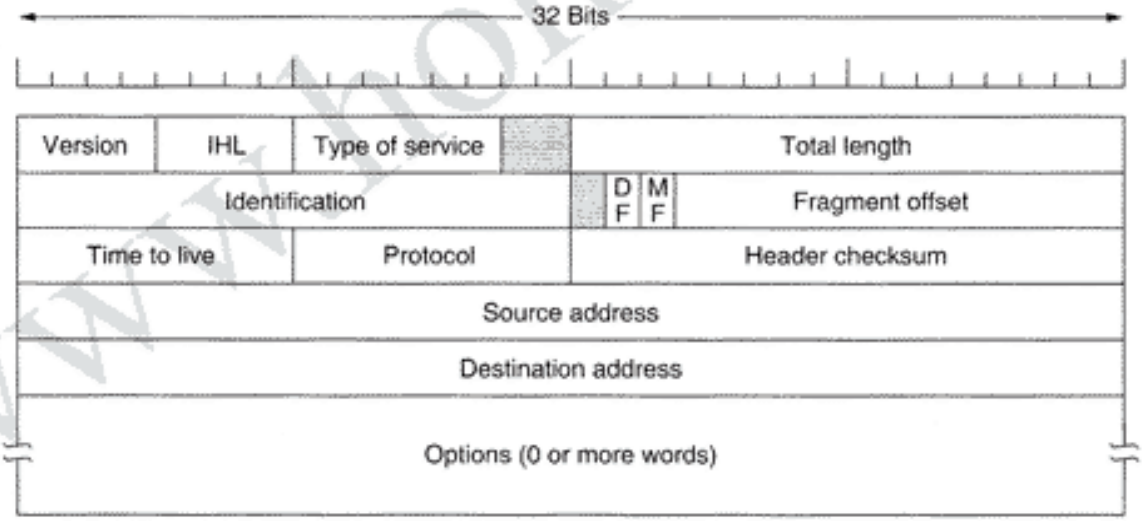


شکل ۵-۵۲. اینترنت مجموعهٔ تعداد بی‌شماری شبکه است.

ارتباط در اینترنت بدین ترتیب است که: لایهٔ انتقال یک دنباله از داده‌ها را تحویل گرفته و آنها را به چند دیتاگرام تقسیم می‌کند. در تئوری، طول حداکثر هر دیتاگرام می‌تواند تا ۶۴ کیلوبایت باشد ولی در عمل، بطور معمول حول و حوش ۱۵۰۰ بایت است (تا بتواند در یک فریم اترنت جا بگیرد). هر دیتاگرام از طریق اینترنت ارسال می‌شود و در صورت نیاز ، در طول مسیر به قطعات کوچکتر شکسته می‌شود. وقتی تمام قطعات به ماشین مقصد رسیدند توسط لایهٔ شبکه (IP) به دیتاگرام اصلی بازسازی می‌شود. این دیتاگرام تحویل لایهٔ انتقال شده و توسط این لایه به پروسهٔ گیرنده تسلیم می‌شود. همانگونه که در شکل ۵-۵۲می‌بینید، بسته‌ای که از ماشین ۱ منشاء گرفته برای رسیدن به ماشین ۲ باید از شش شبکه بگذرد. در عمل این تعداد از شش تا هم بیشتر است.

۵-۶-۱ پروتکل IP

برای شروع به تحلیل پروتکل IP ، بهترین کار آن است که قالب بسته‌های IP را بررسی نماییم.[1] یک بستهٔ IP در برگیرندهٔ دو بخش سرآیند و بخش محتوی است. سرآیند هر بسته دارای یک بخش ثابت ۲۰ بایتی و یک بخش اختیاری با طول متغیر است. ساختار سرآیند بستهٔ IP در شکل ۵-۵۳ نشان داده شده است. ترتیب ارسال بستهٔ IP از بایت پرارزش به کم‌ارزش است یعنی بایتها به ترتیب از چپ به راست ارسال می‌شوند؛ بنابراین اولین فیلدی که ارسال می‌شود فیلد Version (شمارهٔ نسخهٔ پروتکل) است. ماشینهای SPARC داده‌ها را از پرارزش به کم ارزش ذخیره و ارسال می‌کنند و به آنها ماشینهای Big Endian گفته می‌شود در حالیکه ماشینهایی مثل پنتیوم برعکس هستند یعنی داده‌ها را از کم‌ارزش به پرارزش ذخیره و ارسال می‌کنند (ماشینهای Little Endian) و بالطبع نرم‌افزار اینگونه ماشینها باید قبل از ارسال یا پس از دریافت، تبدیل لازم را انجام بدهند.



شکل ۵-۵۳. سرآیند IPv4 در پروتکل اینترنت.

فیلد Version مشخص می‌کند که بسته براساس چه نسخه‌ای از پروتکل IP سازماندهی و ارسال شده است. با قرار دادن شمارهٔ نسخهٔ پروتکل در هر دیتاگرام می‌توان فرآیند گذر از یک نسخهٔ قدیمی به نسخهٔ جدید را به صورت تدریجی و در خلال چندین سال به انجام رساند و در این مدت برخی از ماشینها از نسخهٔ قدیمی و برخی از نسخهٔ جدید استفاده کنند. اکنون در حال گذر از IPv4 به IPv6 هستیم و اگرچه سالهاست که انتظار این جایگزینی می‌رود ولی به نظر می‌رسد هنوز هم به سالها وقت نیاز دارد.

---

۱. به بسته‌های IP اصطلاحاً «دیتا گرام IP» گفته می‌شود. «دیتا گرام IP» یک قطعه دادهٔ دارای هویت و شناسنامه است. ـم

(Durand, 2001; Wiljakka, 2002; Waddington and Chang, 2002) حتی برخی افراد معتقدند این تغییر هیچگاه محقّق نخواهد شد. (Weiser , 2001) مضاف بر این، IPv5 نیز یک بعنوان پروتکل آزمایشی برای «انتقال بی‌درنگ استریم داده»[1] طراحی گردید ولیکن استفادهٔ چندانی از آن نشد.

از آنجایی که طول سرآیند ثابت نیست لذا فیلد IHL بدان منظور در نظر گرفته شده که مشخص کند طول سرآیند (بر مبنای کلمات ۳۲ بیتی) چقدر است. مقدار حداقل این فیلد (یعنی زمانی که هیچ داده‌ای در بخش اختیاری سرآیند وجود ندارد) معادل ۵ است. حداکثر مقدار این فیلد ۴ بیتی، ۱۵ است و طبعاً طول سرآیند به ۶۰ بایت (۱۵×۴) محدود می‌شود؛ یعنی حداکثر طول بخش اختیاری ۴۰ بایت (۲۰-۶۰) است. این فضای ۴۰ بایتی برای برخی از گزینه‌هائی که می‌توان درون این فیلد اختیاری درج کرد (مثل گزینهٔ ثبت مسیر طی شده توسط بسته) بسیار ناکافی است و عملاً فیلد اختیاری را بی‌استفاده کرده است.

فیلد Type of Service (نوع خدمات)، یکی از معدود فیلدهایی است که مضمون و عملکرد آن در طول این سالها تغییر کرده است. از ابتدا (و همچنین اکنون) این فیلد به منظور مشخص کردن کلاسهای مختلف خدمات مدّنظر بوده است. در این فیلد می‌توان ترکیبی از خدمات مثل قابلیت اعتماد (Reliability) و سرعت (Speed) را برای بسته تقاضا کرد. برای انتقال دیجیتال صدا، تحویل سریع داده‌ها ارجح‌تر از تحویل مطمئن و بدون خطاست. برعکس، برای انتقال فایل ارسال بدون خطا بسیار مهمتر از سرعت انتقال است.

فیلد شش بیتی Type of Service (به ترتیب از چپ به راست) در برگیرندهٔ یک فیلد سه بیتی به نام Precedence (فیلد تقدم) و سه بیت علامت به نامهای D و T و R است. فیلد Precedence اولویت بسته را مشخص می‌کند: از صفر (اولویت معمولی) تا ۷ (بالاترین اولویت برای بسته‌های کنترلی شبکه). سه بیت علامت به ماشین میزبان اجازه می‌دهد که از بین سه ویژگی (D: تأخیر، T: ظرفیت خروجی، R: قابلیت اعتماد) نیازهای خود را توصیف نماید. از دیدگاه تئوری این فیلدها امکان آنرا فراهم آورده‌اند که مسیریاب مثلاً از بین یک خط ماهواره‌ای با ظرفیت خروجی بالا و تأخیر زیاد و همچنین یک خط اجاره‌ای با ظرفیت خروجی کم و تأخیر پایین، انتخاب مناسبی داشته باشد، ولیکن در عمل مسیریابها فیلد Type of service را نادیده می‌گیرند.

عاقبت IETF متقاعد شد که مضمون این فیلد را تغییر بدهد تا براساس آن بتوان رده‌های مختلف خدمات مورد نیاز را توصیف کرد. ۶ بیت این فیلد برای مشخص کردن ردهٔ کلاس خدماتی است که بسته بدان کلاس تعلق دارد. این کلاسها که قبلاً در بخش ۵-۴-۴ معرفی شدند، شامل: ۴ اولویت صف‌بندی، ۳ احتمال حذف بسته و تعدادی کلاس برای سازگاری با قبل هستند.

فیلد Total Length طول کل بسته را (شامل سرآیند و داده) مشخص می‌نماید. حداکثر طول یک بسته ۶۵۵۳۵ بایت است. در حال حاضر این مقدار حداکثر کفایت می‌کند ولی در آینده با رواج اترنت گیگابیت ممکن است به دیتاگرام با طول بیشتری نیاز باشد.

فیلد Identification بدان جهت نیاز است که مشخص کنیم قطعهٔ دریافتی، به کدام دیتاگرام متعلق است. تمام قطعات یک دیتاگرام واحد دارای مقداری مشابه در فیلد Identification هستند. [بخش ۵-۵-۷ را مطالعه نمایید.] در ادامه یک بیت بلااستفاده و سپس یک فیلد تک بیتی به نام DF[2] قرار گرفته است. این بیت به مسیریابها فرمان می‌دهد که دیتاگرام (بسته) جاری را قطعه‌قطعه نکنند چرا که مقصد از بازسازی آن عاجز است. مثلاً وقتی

---

۱. Real-Time Stream Protocol

۲. DF مخفف کلمات Don't Fragment به معنای عدم قطعه‌قطعه‌سازی بسته است.

کامپیوتری بوت می‌شود، ROM بوت‌کنندهٔ آن ممکن است تقاضا کند که «تصویری از حافظه» (Memory Image) برای او ارسال شود.[۱] با علامتگذاری در بیت DF، فرستنده مطمئن خواهد شد که کل بسته به صورت یکجا تحویل خواهد شد، حتی اگر این بسته به دلیل محدودیتهایی که مسیر بهینه (در هدایت بسته‌های بزرگ) دارد مجبور به عبور از مسیرهای غیربهینه شود. تمام ماشینها ملزم به پذیرش قطعات با طول ۵۷۶ بایت (یا کمتر) هستند. بیت MF (مخفف More Fragment) به معنای آنست که دنبالهٔ قطعات هنوز ادامه دارد. در تمام قطعات به استثنای قطعهٔ آخر این بیت ۱ است. به این بیت از آن جهت نیاز است که متوجه شویم آیا تمام قطعات یک دیتاگرام دریافت شده یا هنوز دنبالهٔ قطعات ادامه دارد.

فیلد Fragment Offset مشخص کنندهٔ آنست که قطعهٔ جاری در کجای دیتاگرام اصلی واقع شده است. طول تمام قطعات (به استثنای قطعهٔ آخر) باید ضریبی از ۸ بایت باشد. از آنجایی که این فیلد ۱۳ بیتی است حداکثر می‌توان ۸۱۹۲ قطعه در هر دیتاگرام داشت و بدین ترتیب طول حداکثر هر دیتاگرام ۶۵۵۳۶ بایت خواهد شد.

فیلد Time to live (زمان حیات بسته) شمارنده‌ای است که طول عمر بسته را تعیین می‌نماید. فرض بر آنست که این فیلد زمان را برحسب ثانیه مشخص کند و طول عمر بسته حداکثر ۲۵۵ ثانیه باشد. به ازای هر گام (یعنی عبور بسته از یک مسیریاب)، یک واحد از این فیلد کاسته می‌شود. همچنین فرض شده که هر گاه بسته درون یک مسیریاب، زمان زیادی را در صف منتظر ماند تعداد بیشتری از این فیلد کم شود. با تمام این تفاصیل، در عمل فقط تعداد گام محاسبه می‌شود.[۲] به محض آن که مقدار این فیلد به صفر برسد بسته حذف شده و یک پیغام هشدار به مبداء آن برگردانده می‌شود. این ویژگی از سرگردان ماندن بسته‌ها در زیرشبکه (که در اثر اشتباه در جدول مسیریابی بروز می‌کند) جلوگیری می‌نماید.

هر گاه لایهٔ شبکه یک دیتاگرام کامل را بازسازی کرد باید بداند که چه کاری با آن بکند. فیلد Protocol (شمارهٔ پروتکل)، مشخص می‌کند که بسته را باید به کدام پروسه در لایهٔ انتقال تحویل داد. پروسه TCP یکی از گزینه‌هاست؛ ولی پروسهٔ UDP و چندین پروسهٔ دیگر نیز می‌توانند دیتاگرام را تحویل بگیرند. شمارهٔ پروتکلها جهانی و در سرتاسر اینترنت استاندارد هستند. پروتکلها و دیگر شماره‌های استاندارد در سند RFC 1700 ثبت شده است؛ این شماره‌ها را می‌توانید در پایگاه اطلاعاتی www.iana.org بدست بیاورید.

فیلد Header Checksum یک کُد کشف خطا برای سرآیند است. این کد کشف خطا که صرفاً سرآیند بسته را در برمی گیرد قادر است خطاهایی را که از خرابی در حافظهٔ مسیریاب ناشی می‌شود آشکار نماید. الگوریتم کشف خطا بدین ترتیب است که کل سرآیند، در همان بدو ورود در قالب نیم‌کلمات ۱۶ بیتی با یکدیگر جمع شده و نهایتاً «مکمل یک»[۳] آن محاسبه می‌شود. اگر سرآیند بسته بدون خطا باشد جمع تمام نیم کلمات ۱۶ بیتی سرآیند، باید صفر باشد. این الگوریتم قدرتمندتر از جمع معمولی است. دقت کنید که فیلد Header Checksum در هر گام باید از نو محاسبه شود چرا که حداقل یکی از فیلدهای سرآیند تغییر می‌کند (یعنی فیلد Time To Live) ولی برای سرعت بخشیدن به این محاسبه می‌توان از راهی میانبُر استفاده کرد.

فیلدهای Source Address (آدرس مبداء) و Destination Address (آدرس مقصد) شمارهٔ شبکه و شمارهٔ ماشین مبداء و مقصد را مشخص می‌کنند. این دو آدرس را در بخشی مجزا بررسی خواهیم کرد.

فیلد Options بدان منظور طراحی شده تا در نسخه‌های بعدی پروتکل بتوان اطلاعاتی را در سرآیند قرار داد که در نسخهٔ اصلی آن پیش‌بینی نشده است. همچنین می‌توان از آن برای آزمودن ایده‌های جدید بدون درهم

---

۱. تصویری از هستهٔ سیستم عامل که بصورت کدهای اجرایی ارسال می‌شود.

۲. یعنی امروزه فقط به ازای عبور بسته از یک مسیریاب، یک واحد از مقدار این فیلم کم می‌شود و زمان (برحسب ثانیه) هیچ تاثیری در این فیلد ندارد.

۳. One's Compelement

ریختن آن بهره گرفت. فیلد Option دارای طولی متغیر است؛ هر یک از گزینه‌های مندرج در این فیلد با یک کد یک بایتی شروع می‌شود تا ماهیت گزینه را مشخص کند. در برخی از گزینه‌ها پس از این کد یک بایتی، فیلد یک بایتی دیگر، طول گزینه را برحسب بایت مشخص می‌نماید. نهایتاً به فیلد Options تعدادی بایت اضافی و زائد افزوده می‌شود تا طول آن ضریبی از ۴ شود. در ابتدا فقط پنج گزینهٔ فهرست شده در شکل ۵-۵۴ تعریف شده بود ولی پس از آن تعدادی گزینهٔ دیگر بدان افزوده شد. فهرست کامل این گزینه‌ها در آدرس www.iana.org/assignments/ip-parameters در دسترس عموم قرار گرفته است.

گزینه	توصیف عملکرد
Security	میزان محرمانه بودن دیتاگرام جاری را تعیین می‌کند.
Strict source routing	فهرست کامل مسیری را که باید دنبال شود تعریف می‌کند.
Loose source routing	فهرستی از مسیریابها که بسته حتما باید از آنها رد شود.
Record route	مسیریابها را وادار می‌کند تا آدرس IP خود را در بسته درج کنند.
Timestamp	مسیریابها را وادار می‌کند تا آدرس IP خود را بهمراه مهر زمان در بسته درج کنند.

شکل ۵-۵۴. برخی از گزینه‌های IP.

گزینهٔ Security مشخص کنندهٔ آن است که اطلاعات موجود در بسته تا چه حد محرمانه است. از دیدگاه تئوری، کاربرد این گزینه آن است که مسیریابهای نظامی این بسته را از مسیرهایی که از کشورها یا مناطقی می‌گذرند که به زعم آنها نامطمئن و خطرناک هستند، هدایت نکنند. در عمل تمام مسیریابها این فیلد را نادیده می‌گیرند و تنها کاربرد عملی آن می‌تواند کمک به جاسوسان برای انتخاب بسته‌های مورد نظرشان باشد!

با گزینهٔ Strict Source Routing می‌توان مسیر کامل بین مبداء و مقصد را در قالب دنباله‌ای از آدرسهای IP مشخص کرد. بستهٔ ارسالی ملزم به عبور از این مسیر است. این گزینه به مدیران شبکه کمک می‌کند تا در هنگام خرابی جداول مسیریابی یا عملکرد غیرصحیح مسیریابها بتوانند بسته‌های اضطراری خود را به مقصد برسانند یا می‌تواند برای اندازه‌گیری زمانی و تخمین ترافیک یک مسیر مفید واقع شود.

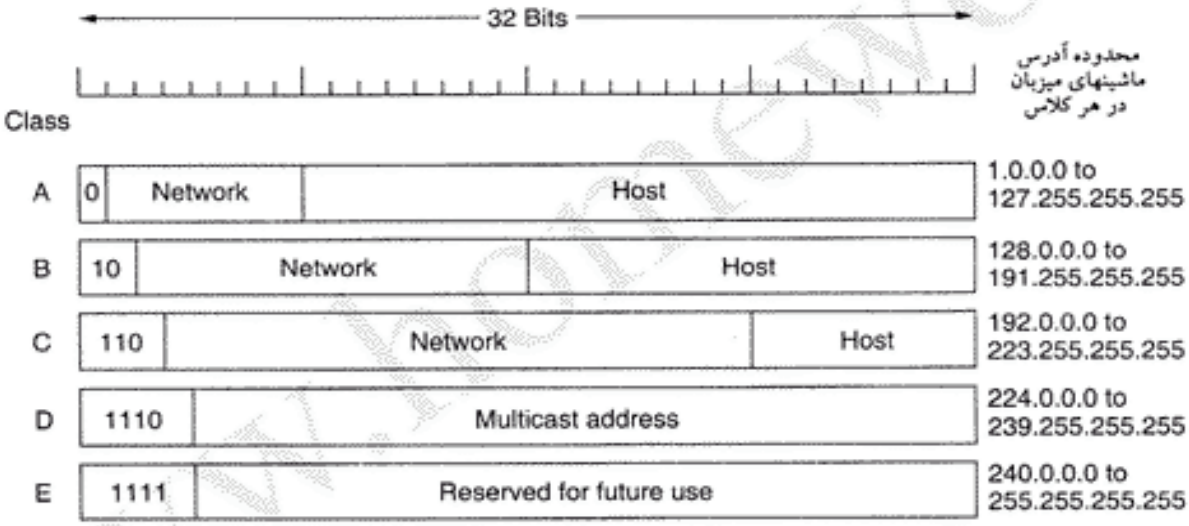
با گزینهٔ Loose Source Routing، بسته ملزم به عبور از مسیریابهای مشخص (طبق ترتیب تعیین شده) می‌باشد ولی اجازه دارد در مسیر خود از مسیریابهای دیگری نیز که در فهرست نیامده، عبور کند. [یعنی در حقیقت فقط برخی از نقاط مسیر مشخص می‌شود و بقیهٔ نقاط توسط مسیریاب انتخاب می‌شود.] بطور معمول در این گزینه فقط تعدادی از مسیریابها مشخص می‌شوند و همین تعداد می‌تواند مسیر مورد نظر را تبیین نماید. به عنوان مثال برای آنکه بسته‌ای را مجبور کنیم برای رفتن از لندن به سیدنی به جای شرق از سمت غرب حرکت کند کافی است در این گزینه سه مسیریاب در نیویورک، لوس آنجلس و هانولولو مشخص شود. این گزینه زمانی سودمند خواهد بود که به دلائل سیاسی یا اقتصادی بخواهیم بسته‌ها از مسیرهایی که در برخی کشورهای خاصّ واقعند عبور نکنند.

گزینهٔ Record Route (ثبت مسیر) به مسیریابهای واقع بر روی مسیر تفهیم می‌کند که باید آدرس IP خود را در همین فیلد (فیلد Option) درج نمایند. این گزینه به مدیران شبکه کمک می‌کند تا بتوانند اشکالات احتمالی در الگوریتم مسیریابی را پیگیری و کشف نمایند (مثلاً: «چرا بسته‌ای که از هیوستون به دالاس روانه شده در توکیو دیده شده است؟») زمانی که برای اولین بار ARPANET راه‌اندازی شد کل مسیریابهای این شبکه ۹ تا بیشتر نبود و فضای چهل بایتی فیلد Options کفایت می‌کرد ولی امروزه به دلیل رشد سرسام‌آور شبکهٔ اینترنت این فضای چهل بایتی بسیار ناکافی و عملاً بی‌کاربرد است.

در آخر، گزینهٔ Timestamp (مُهر زمان) عملکردی شبیه به گزینهٔ قبلی (یعنی Record Route) دارد با این تفاوت که به غیر از ثبت آدرس IP هر مسیریاب، یک «مُهر زمان» ۳۲بیتی نیز در کنار آن درج می‌شود. این گزینه نیز برای اشکال‌زدایی از الگوریتمهای مسیریابی کاربرد دارد.

۵-۶-۲ آدرسهای IP

هر ماشین میزبان و هر مسیریاب در شبکهٔ اینترنت دارای یک آدرس IP است که در آن شمارهٔ شبکه و شمارهٔ ماشین، مشخص می‌شود. ترکیب این دو شماره، منحصر به فرد و یکتا است: بر اساس این اصل اساسی که هیچ دو ماشینی در اینترنت نباید آدرس IP یکسانی داشته باشند. تمام آدرسهای IP، ۳۲بیتی هستند و برای مشخص کردن آدرس ماشین مبداء و مقصد، در درون فیلدهای مربوطه در بسته‌های IP قرار می‌گیرند. اشاره به این نکته بسیار مهم است که آدرس IP در حقیقت یک ماشین میزبان را مشخص نمی‌کند بلکه به یک «کارت واسط شبکه» اشاره دارد، فلذا اگر ماشینی بر روی دو شبکه واقع شده باشد باید دو آدرس IP داشته باشد ولی در عمل بیشتر ماشینها فقط بر روی یک شبکه قرار گرفته‌اند و طبعاً یک آدرس IP بیشتر ندارند.[1]



شکل ۵-۵۵. قالب آدرسهای IP.

در چند دههٔ ابتدایی، آدرسهای IP در پنج ردهٔ مختلف دسته‌بندی شده بودند. این پنج رده در شکل ۵-۵۵ نشان داده شده است. تخصیص فضای آدرس مطابق با این روش، اصطلاحاً روش «آدرس‌دهی دارای کلاس» (Classful Addressing) نامیده می‌شود. این روش امروزه کاربرد خود را از دست داده است [از حدود سال ۱۹۹۶] ولی هنوز هم در کتب مختلف به آن اشاره می‌شود. این روش آدرس‌دهی را به اختصار توضیح می‌دهیم.

کلاسهای A و B و C و D این امکان را فراهم آورده‌اند تا بتوان به ترتیب: ۱۲۸ شبکه با ۱۶ میلیون ماشین میزبان (در کلاس A)، ۱۶۳۸۴ شبکه با ۶۵۵۳۶ ماشین (در کلاس B) و حدود ۲ میلیون شبکه با ۲۵۶ ماشین (در کلاس C) را آدرس‌دهی کرد (اگرچه برخی از این آدرسها برای کارهای ویژه در نظر گرفته شده‌اند و به هیچ ماشینی منتسب نمی‌شوند.) همچنین در کلاس D از ارسال چندپخشی (Multicast) پشتیبانی می‌شود و می‌توان بسته‌ای را مستقیماً برای چندین ماشین در شبکه فرستاد.

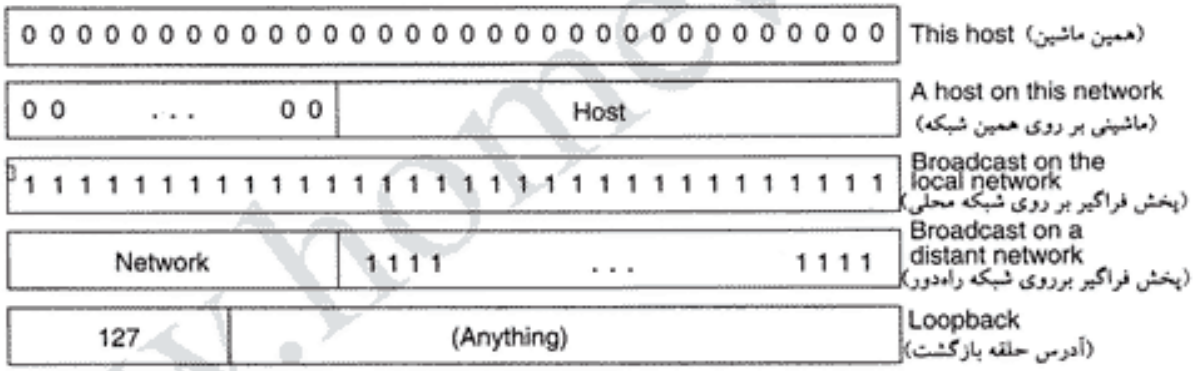
---

۱. مسیریابها که چندین کارت واسط دارند بیش از یک آدرس IP خواهند داشت.

آدرسهایی که با چهار بیت ۱۱۱۱ شروع می‌شوند (یعنی کلاس E) هیچ استفاده‌ای ندارند و برای کاربردهای بعدی رزرو شده‌اند. امروزه بیش از ۵۰۰,۰۰۰ شبکه به اینترنت متصل شده‌اند و این تعداد سال به سال افزایش می‌یابد. شماره‌های شبکه [یعنی فیلد Network در آدرس IP] توسط یک موسسهٔ غیرانتفاعی به نام ICANN[1] مدیریت می‌شود تا تناقضی در انتساب شماره‌های تکراری پدید نیاید. ICANN تخصیص بخشهایی از فضای آدرس‌دهی را به تعدادی «مراکز مجاز منطقه‌ای» واگذار کرده تا آنها آدرسهای IP را بین سرویس دهنده‌های اینترنتی (ISP) و دیگر شرکتها توزیع نمایند.

آدرسهای IP که ۳۲بیتی هستند معمولاً با نماد دهدهی نقطه‌دار نمایش داده می‌شوند. در این روش هر یک از چهار بایتِ آدرس به صورت مجزا و در مبنای ده (از صفر تا ۲۵۵) نوشته می‌شود. مثلاً آدرس ۳۲بیتی C0290614 (در مبنای شانزده) به صورت 192.41.6.20 نمایش داده می‌شود. کمترین آدرس IP معادل 0.0.0.0 و بزرگترین آن معادل با 255.255.255.255 است.

مقدار صفر و ۱- (یعنی فیلدی که تمام بیتهایش 1 است) معانی خاص دارد. شکل ۵-۵۶ آدرسهای خاص را مشخص کرده است. مقدار ۰ خود ماشین یا شبکه را مشخص می‌نماید.[2] مقدار ۱- برای پخش فراگیر (Broadcast) کاربرد دارد، یعنی اگر بیتهای آدرس مقصد بسته تماماً ۱ باشد، گیرندهٔ بسته، تمام ماشینهای آن شبکه می‌باشد.



شکل ۵-۵۶. آدرسهای IP خاص.

آدرس IP معادل با 0.0.0.0 توسط ماشینهای میزبان و در حین راه‌اندازی (بوت) مورد استفاده قرار می‌گیرد. هر گاه شمارهٔ شبکه در آدرس IP صفر باشد چنین آدرسی به شبکهٔ فعلی اشاره می‌کند. این آدرسها اجازه می‌دهد که ماشینها بدون دانستن شمارهٔ شبکهٔ خود، به آن اشاره کنند [یعنی اگر ماشینی بخواهد برای ماشین دیگری در شبکهٔ خودش بسته‌ای بفرستد، دانستن شمارهٔ شبکه، مهم نیست]. البته باید هر ماشین، حداقل کلاس آدرس شبکهٔ خود را بداند تا تشخیص بدهد فیلد شمارهٔ شبکه چند بیتی است و آن را با صفر پر کند. آدرسی که تمام بیتهای آن ۱ است اجازه می‌دهد که بسته به صورت فراگیر بر روی شبکهٔ محلی پخش شود. [یعنی بسته‌ای با چنین آدرسی را همهٔ ماشینهای موجود بر روی شبکهٔ محلّی دریافت می‌دارند.] اگر شمارهٔ شبکه با یک مقدار درست تنظیم شده ولی تمام بیتهای شمارهٔ ماشین مساوی ۱ باشد می‌توان بسته‌ای را برای تمام ماشینهای یک شبکهٔ LAN ، در هر کجای اینترنت ارسال نمود. (البته بسیاری از مسئولین شبکه این ویژگی را غیرفعال و ناممکن کرده‌اند.)

---

۱. Internet Corporation for Assigned Names and Numbers

۲. یعنی اگر فیلد Network در آدرس IP معادل صفر باشد به معنای خود شبکه و اگر فیلد Host صفر باشد به معنای خود آن ماشین است. بسته‌ای با این آدرسها از شبکه یا ماشین بیرون نخواهد رفت. -م

تمام آدرسهایی که به شکل 127.xx.yy.zz هستند به نام «**حلقهٔ بازگشت**» (Loopback) مشهورند و برای آزمایش نرم‌افزار، کنار گذاشته شده‌اند: هر بسته‌ای که به چنین آدرسهایی ارسال شود به صورت واقعی بر روی سیم منتقل نخواهد شد بلکه به صورت داخلی پردازش شده و با آن همانند بستهٔ ورودی رفتار می‌شود. این آدرس اجازه می‌دهد بدون آنکه ماشین فرستنده شمارهٔ خود را بداند برای خودش بسته بفرستد.

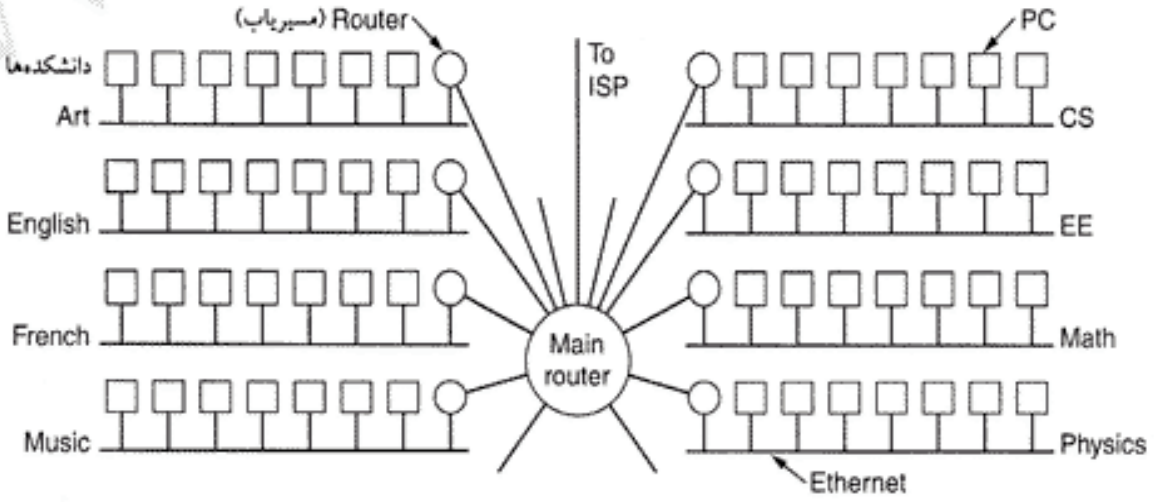
زیرشبکه‌ها

همانگونه که قبلاً بررسی کردیم تمام ماشینهای یک شبکه باید دارای شمارهٔ شبکهٔ یکسانی باشند. با رشد شبکه، این ویژگی در آدرس‌دهی IP، باعث بروز مشکلاتی می‌شود. به عنوان مثال دانشگاهی را مدّنظر قرار بدهید که در ابتدا با یک آدرس کلاس B شروع کرده و آن را در شبکهٔ اترنت متعلق به دانشکدهٔ علوم کامپیوتر بکار گرفته است. یک سال بعد دانشکدهٔ مهندسی برق می‌خواهد که به اینترنت متصل شود و با خرید یک تکرارکننده (Repeater) سعی می‌کند به شبکهٔ اترنت دانشکدهٔ علوم کامپیوتر متصل شود. با گذشت زمان، دانشکده‌های دیگر نیز تقاضای وصل به شبکه می‌کنند و بدین ترتیب به دلیل محدودیت در تعداد تکرارکننده‌های اترنت این کار غیرممکن می‌شود. بنابراین به سازماندهی متفاوتی نیاز است.

از طرف دیگر، ثبت و دریافت یک کلاس آدرس مجزا برای شبکه‌های جدید نیز دشوار است چراکه آدرسهای شبکه محدود و کم هستند و مضاف بر این، آدرس کلاس B اختصاص داده شده به این دانشگاه برای آدرس‌دهی به بیش از ۶۰۰۰۰ ماشین کافی است. مشکل اینجاست که یک کلاس A، B یا C به یک شبکهٔ واحد اشاره می‌کند نه به یک مجموعه از شبکه‌های LAN متصل بهم.

پس از آنکه سازمانهای متعددی به مشکل کمبود آدرس شبکه، دچار شدند یک تغییر بسیار کوچک در این آدرسها مشکل را حل کرد: راه حل آن است که اجازه بدهیم شبکه به چندین بخش مستقل داخلی تقسیم شده ولی کماکان در دنیای خارج به عنوان یک شبکهٔ واحد تلقی شود.

یک «شبکهٔ دانشگاهی» (Campus)، نمادی شبیه به شکل ۵-۵۷ دارد که در آن یک مسیریاب مرکزی از طرفی به یک ISP (یا یک شبکهٔ منطقه‌ای) و از طرف دیگر به چندین شبکه اترنت در دانشکده‌های مختلف وصل شده است. هر یک از شبکه‌های اترنت دارای یک مسیریاب محلی هستند که از طریق آن به مسیریاب مرکزی متصل شده‌اند. (البته ممکن است به جای مسیریاب مرکزی یک ستون فقرات قرار گرفته باشد ولیکن چگونگی اتصال مسیریابها در این مثال مهم نیست.)



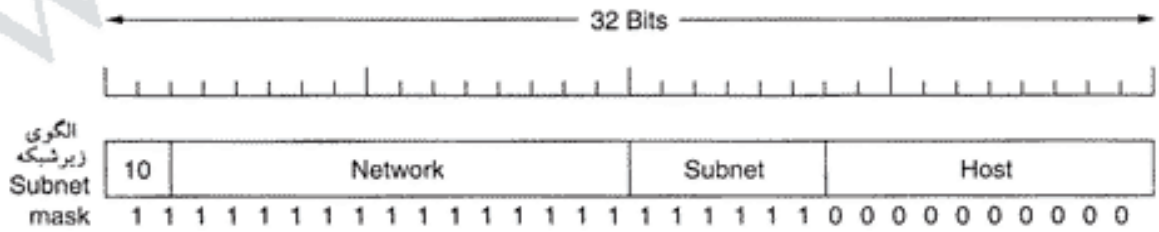
شکل ۵-۵۷. یک شبکهٔ دانشگاهی که شامل چندین LAN متعلق به دانشکده‌های مختلف است.

در ادبیات شبکهٔ اینترنت، به هر بخش از یک شبکهٔ بزرگ و خودمختار، اصطلاحاً «**زیرشبکه**» (Subnet) گفته می‌شود. (مثلاً در مثال بالا هر یک از شبکه‌های اترنت یک «زیرشبکه» تلقی می‌شوند.) همانطور که در فصل اول اشاره کردیم، این استعاره می‌تواند با واژهٔ «زیرشبکه» به معنای مجموعه‌ای از مسیریابها و خطوط ارتباطی در یک شبکه اشتباه شود. البته زمینه و مفاد یک بحث، معنای مورد نظر آن را مشخص خواهد کرد. در این بخش و بخش آتی، هر گاه واژهٔ «زیرشبکه» را بکار بردیم، تعریف جدید آن مدّنظر است.

وقتی بسته‌ای به مسیریاب مرکزی وارد می‌شود، مسیریاب چگونه تشخیص می‌دهد که آنرا به کدام زیرشبکه (اترنت) تحویل بدهد؟ یک راه آن است که جدولی با ۶۵۵۳۶ درایه (Entry) داشته باشیم و هر درایه محل هر ماشین میزبان را در کل شبکه مشخص کند. این ایده عملی است ولی به یک جدول بسیار بزرگ در مسیریاب مرکزی نیاز است و در صورت اضافه شدن، حذف یا تغییر محل ماشینها، باید تنظیمات لازم بصورت دستی انجام شود.

به جای این روش، الگوی جدیدی ابداع شده است. در این روش به جای آنکه یک کلاس B (با ۱۴ بیت برای شمارهٔ شبکه و ۱۶ بیت برای شمارهٔ ماشین) داشته باشیم، بخشی از فضای شانزده بیتی شمارهٔ ماشین برای تعیین «شمارهٔ زیرشبکه» در نظر گرفته می‌شود. به عنوان مثال اگر دانشگاه ۳۵ دانشکده داشته باشد، می‌تواند ۶ بیت را برای شمارهٔ زیرشبکه و ده بیت باقیمانده را برای شمارهٔ ماشین در نظر بگیرد. بدین ترتیب می‌تواند حداکثر ۶۴ زیرشبکه اترنت با حداکثر ۱۰۲۲ ماشین میزبان داشته باشد. (قبلاً اشاره شد که شماره‌های ۰ یا ۱- کاربرد خاص داشته و قابل استفاده نیست). این تقسیم بندی را بعداً می‌توان تغییر داد.

برای تعریف و پیاده‌سازی زیرشبکه‌ها ، مسیریاب مرکزی به یک «**الگوی زیرشبکه**» (Subnet Mask) نیاز دارد تا مطابق با شکل ۵-۵۸بتوان مرز بین شمارهٔ ماشین و شمارهٔ شبکه و شمارهٔ زیرشبکه را مشخص و تفکیک کرد. الگوی زیرشبکه (Subnet Mask) نیز به صورت نماد نقطه‌دار دهدهی نوشته می‌شود. همچنین برای راحتی نمایش، می‌توان پس از آدرس IP یک علامت "/"گذاشت و پس از آن تعداد بیتهای ۱ موجود در الگوی زیرشبکه را تعیین کرد. به عنوان مثال در شکل ۵-۵۸ الگوی زیرشبکه را می‌توان به صورت 255.255.252.0 نوشت. راه دیگر آن است که از نماد 22/استفاده نماییم که مشخص‌کننده آن است که الگوی زیرشبکه ۲۲ بیت طول دارد. (۲۲ بیت ۱ از سمت چپ به راست)



شکل ۵-۵۸. یک کلاس B به ۶۴ زیرشبکه بخش‌بندی شده است.

خارج از محدودهٔ این شبکه، این چنین تقسیم بندی و تعداد زیرشبکه‌ها مشهود و مشخص نیست لذا تعریف زیرشبکه جدید مستلزم آن نیست که با ICANN تماس گرفته شده یا تغییری در پایگاه اطلاعات خارج از آن شبکه داده شود. در این مثال، اولین زیرشبکه از آدرسهایی استفاده می‌کند که مثلاً از 130.50.4.1 شروع می‌شوند. آدرسهای دومین زیرشبکه از 130.50.8.1 و آدرس سومین زیرشبکه از 130.50.12.1 شروع می‌شوند. برای آنکه ببینیم چرا شمارهٔ زیرشبکه‌ها چهار تا چهار تا اضافه می‌شود، شکل دودویی این آدرسها را در نظر می‌گیریم:

زیرشبکهٔ ۱	10000010	00110010	000001|00	00000001
زیرشبکهٔ ۲	10000010	00110010	000010|00	00000001
زیرشبکهٔ ۳	10000010	00110010	000011|00	00000001

در الگوی بالا خط عمودی که به صورت | نشان داده شده مرز بین شمارهٔ زیرشبکه و شمارهٔ ماشین را مشخص می‌نماید. در سمت راست، ۶ بیت برای شمارهٔ زیرشبکه و درست چپ ۱۰ بیت برای شمارهٔ ماشین در نظر گرفته شده است.

بررسی عملکرد زیرشبکه‌ها مستلزم آنست که تشریح کنیم بسته‌های IP در مسیریابها چگونه پردازش می‌شوند. هر مسیریاب جدولی را در اختیار دارد که در آن مجموعه‌ای از آدرسهای IP به صورت زوجهای (۰ , **شمارهٔ شبکه**) و (**شمارهٔ ماشین** , **همین شبکه**)[1] درج شده است. نوع اول مشخص می‌کند که چگونه می‌توان به یک شبکهٔ راه دور رسید. نوع دوم، آدرس ماشینهای محلی را [که مستقیماً به آن مسیریاب متصلند] مشخص می‌کند. همچنین در این جدول، آدرس کارت واسط شبکه برای رسیدن به هر یک از این زیرشبکه‌ها (و پاره‌ای اطلاعات اضافی) درج شده است.

وقتی یک بستهٔ IP توسط یک مسیریاب دریافت می‌شود، آدرس مقصد آن در جدول مسیریابی جستجو می‌شود. اگر مقصد بسته یک شبکهٔ راه دور باشد از طریق یکی از کارتهای واسط به سوی مسیریاب بعدی هدایت می‌شود. اگر مقصد بسته یکی از ماشینهای محلی باشد (مثلاً یکی از ماشینهای LAN متصل به مسیریاب)، مستقیماً برای آن ماشین ارسال می‌شود. اگر مقصد بسته وجود نداشته یا شناسایی نشود، بسته به سوی یک مسیریاب پیش فرض (Default Router) هدایت می‌شود. این مسیریاب دارای جدول مسیریابی کاملتر بوده و احتمالاً یک «مسیریاب مرزی» است. پس طبق این الگوریتم، هر مسیریاب فقط باید مشخصات دیگر شبکه‌ها (یا به عبارت بهتر زیرشبکه‌ها) و همچنین ماشینهای محلی خود را بداند و بدین ترتیب اندازهٔ جدول مسیریابی بشدت کاهش می‌یابد.

وقتی زیرشبکهٔ جدیدی تعریف می‌شود، جدول مسیریابی تغییر می‌کند و در آن درایه‌هایی به شکل (0 , **شمارهٔ زیرشبکه** , **شمارهٔ همین شبکه**) و (0 , **شمارهٔ همین زیرشبکه** , **شمارهٔ همین شبکه**)[2] اضافه می‌شود. بدین ترتیب یک مسیریاب که به زیرشبکه k متصل است فقط می‌داند که چگونه باید به زیرشبکه‌های دیگر برسد و همچنین آدرس تمام ماشینهای متصل به زیرشبکه خودش (یعنی زیرشبکه k) را می‌داند. بنابراین مسیریابها مجبور نیستند جزئیات آدرس ماشینهای واقع در زیرشبکه‌های دیگر را بدانند. در حقیقت تنها کاری که مسیریاب در برخورد با آدرسهای IP متعلق به ماشینهای خارج از زیرشبکه خود انجام می‌دهد آنست که: آدرسهای IP را با «**الگوی زیرشبکه**» (Subnet Mask) به صورت بولین AND می‌کند تا بخش شمارهٔ ماشین آن حذف شود. سپس آدرس حاصل را در جدول خود جستجو می‌کند. به عنوان مثال فرض کنید بسته‌ای به آدرس 130.50.15.6 ارسال و توسط مسیریاب مرکزی (شکل ۵.۵۷) دریافت شده باشد. مسیریاب مرکزی این آدرس را با الگوی زیرشبکه 255.255.252.0/22 به صورت بولین AND می‌کند و نتیجهٔ 130.50.12.0 بدست می‌آید. این آدرس در جدول مسیریابی جستجو می‌شود تا خط خروجی مناسب برای رسیدن به زیرشبکه ۳ مشخص گردد.

تعریف زیرشبکه در حقیقت با ایجاد یک سلسله مراتب سه‌سطحی حجم جدول مسیریابی را کاهش می‌دهد. این سلسله مراتب سه‌سطحی تشکیل شده از: **شمارهٔ شبکه** + **شمارهٔ زیرشبکه** + **شمارهٔ ماشین میزبان**.[3]

---

۱. (Network, 0) (this-network, host)

۲. (this-network, subnet, 0) (this-network,this subnet, host)

۳. مسیریابی سلسله‌مراتبی را در بخش ۵-۲-۶ از همین فصل مطالعه کنید.

CIDR[1] : مسیریابی براساس آدرسهای بدون کلاس

چندین دهه است که از پروتکل IP استفادهٔ گسترده‌ای می‌شود. عملکرد این پروتکل بسیار عالی بوده و همانگونه که اشاره کردیم به رشد نمایی شبکهٔ اینترنت کمک کرده است. متأسفانه IP در حال قربانی شدن در پای محبوبیت خود است چراکه با کمبود فضای آدرس مواجه شده است . از زمانی که افق این مشکل پدیدار شد، بحثها و مناقشات بسیار گسترده‌ای را در جامعهٔ اینترنت دامن زد که چه کاری می‌توان برای آن انجام داد. در بخش جاری، این مشکل و راه حلهای پیشنهادی را تشریح می‌نماییم.

تا سال ۱۹۸۷ افراد دوراندیشی که حدس می‌زدند در روزگاری تعداد شبکه‌های متصل به اینترنت به مرز ۱۰۰,۰۰۰ برسد، انگشت‌شمار بودند و حتی اغلب متخصصین این فنّ نیز چنین ایده‌ای را به مسخره می‌گرفتند ولی صدهزارمین شبکهٔ دنیا در سال ۱۹۹۶ به اینترنت پیوست و به نحوی که در بالا اشاره شد اینترنت در حال مواجه شدن با کمبود آدرسهای IP است. در اصل بیش از دو میلیارد آدرس IP وجود دارد ولی در عمل به دلیل تقسیم‌بندی نامناسب فضای آدرس در قالب چند کلاس، میلیونها آدرس IP به هدر رفته است. (شکل ۵-۵۵ را ببینید.) بالاخص، بیشترین محدودیت در کلاس B است. برای بسیاری از مؤسسات یک شبکهٔ کلاس A با شانزده میلیون آدرس، بسیار بزرگ و بی‌مصرف و کلاس C با ۲۵۶ آدرس بسیار کوچک و ناکافی است؛ فقط شبکهٔ کلاس B با ۶۵۵۳۶ آدرس مناسب است. این مشکل در فرهنگ رایج اینترنت به نام مشکل «سه خرس» مشهور شده است. (برگرفته از داستان موطلایی و سه خرس)

حقیقت آنست که حتی کلاس B هم برای بسیاری از سازمانها و مؤسسات، بسیار بزرگ است. بررسیها نشان داده که بیش از نیمی از شبکه‌های کلاس B ، کمتر از ۵۰ ماشین دارند! برای چنین شبکه‌هایی استفاده از کلاس C کفایت می‌کند ولی بی‌تردید تمام مؤسساتی که متقاضی کلاس B بوده‌اند بدان می‌اندیشیده‌اند که روزی فیلد هشت بیتی کلاس C برای شبکهٔ آنها کافی نخواهد بود. نگاهی به گذشته نشان می‌دهد که بهتر آن بوده در کلاس C فیلد شمارهٔ شبکه به جای هشت بیت، ده بیت می‌بود تا در هر شبکه ۱۰۲۲ ماشین میزبان قابل آدرس‌دهی باشد. اگر چنین حالتی را می‌داشتیم شاید بسیاری از مؤسسات و سازمانها به کلاس C اکتفا می‌کردند و در عین حال نیم میلیون از چنین شبکه‌هایی قابل تعریف بود. (برخلاف کلاس B که فقط ۱۶۳۸۴ شبکه قابل تعریف است).

نمی‌توان تقصیر را به گردن طراحان اینترنت انداخت که چرا تعداد کلاسهای B را بیشتر (و با فضای آدرس کوچکتر) در نظر نگرفتند. در روزگاری که تصمیم گرفته شد فقط سه کلاس وجود داشته باشد، اینترنت شبکه‌ای تحقیقاتی بود که فقط مراکز پژوهشی و دانشگاهی مهم ایالات متحده را بهم متصل می‌کرد. (البته به اضافهٔ معدود شرکتها و سایتهای نظامی که آنها نیز در کار پژوهش بودند.) هیچکس نمی‌توانست پیش‌بینی کند که اینترنت روزی بتواند در حد وسیع و به عنوان سیستمی ارتباطی حتی با شبکه‌های تلفن رقابت نماید! در آن زمان هیچکس در این ادعا شکی نداشت که اگر تمام ۲۰۰۰ کالج و دانشگاه ایالات متحده و حتی اکثر دانشگاههای جهان به اینترنت بپیوندند باز هم ۱۶۰۰۰ تا نمی‌شود چراکه این تعداد دانشگاه در کل دنیا وجود نداشت. به علاوه در آن زمان برای آنکه پردازش بسته‌ها سریعتر انجام بشود بهتر بود فیلد شمارهٔ ماشین در آدرس IP ، تعداد صحیحی از بایت باشد. [به دلیل سرعت پایین پردازنده‌ها در آن زمان، پردازش بیتی فیلدهای آدرس، از سرعت مسیریابی می‌کاست.]

ولیکن در طرف مقابل اگر مثلاً برای فیلد شمارهٔ شبکه در کلاس B ، بیست بیت کنار گذاشته می‌شد مشکل دیگری بروز می‌کرد: «مشکل رشد انفجاری جداول مسیریابی». از دید مسیریابها فضای آدرسهای IP ، سلسله مراتب دوسطحی شامل شمارهٔ شبکه و شمارهٔ ماشینها است. مسیریابها مجبور به دانستن شمارهٔ ماشینهای میزبان

---

۱. CIDR : Classless InterDomain Routing

نیستند ولی باید شمارهٔ شبکه‌ها را بدانند. اگر حتی نیمی از شبکه‌های کلاس C بکار گرفته شده باشند هر مسیریاب در کل اینترنت نیاز به جدولی با نیم میلیون درایه (Entry) دارد تا بتواند خط خروجی مناسب برای رسیدن به هر شبکه را مشخص نماید. (گذشته از اطلاعات دیگری که به ازای هر شبکه باید در جدول مسیریابی درج شود.)

شاید تهیهٔ فضای فیزیکی لازم برای ذخیرهٔ نیم میلیون درایه (Entry) در جدول مسیریابی، امکان‌پذیر باشد هر چند برای مسیریابهایی که جداول مسیریابی را در حافظهٔ نوع ایستا (Static RAM) ذخیره می‌کنند، چنین فضایی بسیار گران تمام می‌شود. مشکل اساسی در پیچیدگی الگوریتمهای مدیریت و پردازش چنین جدولی است. پیچیدگی زمانی این الگوریتمها غیرخطی است.[1] از این بدتر آنکه نرم‌افزار یا سخت‌افزار طراحی شده برای مسیریابهای موجود، زمانی طراحی شده که به اینترنت بیش از هزار شبکه متصل نبود و به نظر می‌رسید که یک دهه طول می‌کشد تا این تعداد به ۱۰۰۰۰ برسد. امروزه اینگونه طراحیها بهینه نیستند.

مضاف بر این، در الگوریتمهای مختلف مسیریابی نیاز است که جداول مسیریابی بطور متناوب ارسال و مبادله شود. (مثل الگوریتم بردار فاصله) هر چه جداول مسیریابی بزرگتر باشد احتمال آنکه بخشی از آن در حین مبادله از دست برود بیشتر خواهد شد و به نقص داده‌های جدول مسیریابی و احتمالاً ناپایداری فرآیند هدایت بسته‌ها خواهد انجامید.

مشکل جداول مسیریابی را می‌توان با افزایش «سطوح سلسله‌مراتب» حلّ کرد. مثلاً می‌توان آدرسهای IP را بدین نحو تعریف کرد که شامل فیلدهای کشور، ایالت / استان، شهر، شبکه و شمارهٔ ماشین میزبان باشد. بدین ترتیب مسیریابهای ستون فقرات در جهان فقط باید در خصوص مسیرهای رسیدن به هر کشور آگاهی داشته باشند، مسیریابهای درون کشور در خصوص مسیرهای رسیدن به هر ایالت یا استان، مسیریابهای ایالت یا استان در مورد مسیرهای رسیدن به هر شهر و مسیریابهای هر شهر فقط باید راه رسیدن به هر شبکه را بدانند. متأسفانه چنین راه حلی نیازمند فضایی بزرگتر از ۳۲ بیت برای آدرس IP است و طبعاً از فضای آدرس استفادهٔ بهینه نخواهد شد. (چرا که مثلاً در این الگو کشور لیختن اشتاین به همان تعداد بیت در آدرس IP دارد که کشور ایالات متحده!)

کوتاه سخن آنکه هر یک از راه حلهای ارائه شده مشکلی را حل و مشکل جدیدی را ایجاد می‌کردند. راه حل نهایی و پیاده شده در اینترنت که توانست به اینترنت اجازهٔ نفس کشیدن بدهد، روش CIDR (مسیریابی براساس آدرسهای بدون کلاس) بود. ایدهٔ اصلی در CIDR که در سند RFC 1519 تشریح شده آنست که آدرسهای IP بدون در نظر گرفتن کلاس و به صورت بلوکهایی با طول متغیر تخصیص یابد. مثلاً اگر یک سایت نیاز به ۲۰۰۰ آدرس داشته باشد یک بلوک آدرس ۲۰۴۸ تایی به او داده می‌شود.

حذف کلاسهای آدرس، فرآیند هدایت بسته‌ها را پیچیده‌تر می‌کند. در سیستم مبتنی بر کلاس، فرآیند هدایت بدین نحو بود که وقتی بسته‌ای به یک مسیریاب می‌رسید، یک کپی از آدرس IP به اندازهٔ ۲۸ بیت به راست شیفت داده می‌شد تا فقط چهار بیت سمت چپ آدرس (که کلاس آدرس را مشخص می‌کند) باقی بماند. براساس این چهار بیت (۱۶ حالت مختلف) بسته‌ها در یکی از کلاسهای A و B و C (و D در صورت پشتیبانی از آن) مرتب می‌شدند. (از این ۱۶ حالت مختلف، هشت حالت برای کلاس A است ـ 0xxx ـ چهار حالت برای کلاس - 10xx - B ، دو حالت برای کلاس C - 110x - و دو حالت برای کلاسهای D و E است.) پس از تشخیص کلاس آدرس، برای بدست آوردن شمارهٔ شبکه، آدرس IP با یکی از الگوهای -8 ، -16 ، -24 به صورت بولی AND و بخش شمارهٔ ماشین حذف می‌شد. سپس شمارهٔ شبکه در هر یک از جداول مربوط به آدرسهای کلاس A ، B و C

---

۱. پیچیدگی غیرخطی این الگوریتمها عموماً $O(n^2)$ و $O(n.\log n)$است. ـم

جستجو می‌شد. جداول مسیریابی برای کلاسهای A و B برحسب شمارهٔ شبکه ایندکس شده بودند.[1] در عوض جدول مسیریابی برای کلاس C مبتنی بر روش جداول Hash (Hash Table) پیاده شده بود. پس از آنکه درایهٔ متناظر با آدرس شبکه در یکی از این جداول پیدا می‌شد خط خروجی متناسب با آن شبکه مشخص شده و بسته بر روی آن خط هدایت می‌گردید.

در CIDR این الگوریتم ساده، کار نخواهد کرد. در عوض به هر یک از درایه‌های جدول مسیریابی یک فیلد ۳۲بیتی جدید افزوده شده که الگوی آدرس را [از طریق یک MASK سی و دو بیتی] مشخص می‌کند. بدین ترتیب برای تمام شبکه‌ها فقط یک جدول مسیریابی یکتا وجود دارد که در حقیقت یک آرایهٔ سه ستونی متشکل از آدرس IP، الگوی زیرشبکه (Subnet Mask) و خط خروجی است. وقتی بسته‌ای وارد می‌شود ابتدا آدرس IP آن استخراج می‌شود. سپس جدول مسیریابی درایه به درایه (Entry by Entry) جستجو و آدرس مقصد بسته پس از AND شدن با الگوی زیرشبکه از هر درایه با آدرس IP از آن درایه مقایسه می‌شود. این فرآیند آنقدر تکرار می‌گردد تا به موارد مطابقت برسد. این امکان وجود دارد که چندین درایه با یک آدرس IP مطابقت داشته باشد (به دلیل طول متفاوت الگوهای زیرشبکه). در این حالت درایه‌ای که طول الگوی زیرشبکه آن از همه بزرگتر است از بین آنها انتخاب می‌شود. به عبارتی اگر دو مورد تطابق با طول الگوی 20/ (255.255.240.0) و الگوی 24/ (255.255.255.0) پیدا شود، درایهٔ دوم انتخاب می‌شود.

برای سرعت بخشیدن به فرآیند جستجو و مطابقت، الگوریتمهای پیچیده‌ای ابداع شده است. (Ruiz-Sanches et al. 2001) مسیریابهای تجاری در بازار امروز از تراشه‌های VLSI خاصی بهره گرفته‌اند که الگوریتم مذکور را به صورت یک «سخت‌افزار درون‌کار» (Embedded Hardware) پیاده‌سازی کرده‌اند.

برای آنکه فهم فرآیند هدایت بسته‌ها در CIDR را ساده‌تر کنیم مثالی را مدنظر قرار بدهید که در آن میلیونها آدرس تعریف شده است و آدرس شروع 194.24.0.0 است. فرض کنید که دانشگاه کمبریج به ۲۰۴۸ آدرس نیاز دارد و آدرسهای 194.24.0.0 تا 194.24.7.255 به آن اختصاص داده شده است. (الگوی زیرشبکه نیز 255.255.248.0 است.) بعداً دانشگاه آکسفورد تقاضای ۴۰۹۶ آدرس IP می‌دهد. از آنجایی که بلوکهای آدرس ۴۰۹۶ تایی باید در مرز ۴۰۹۶ بایتی قرار بگیرد نمی‌توان آدرسهایی که از 194.24.8.0 شروع می‌شود را به آن اختصاص داد. در عوض آدرس اختصاص داده شده به او در محدودهٔ 194.24.16.0 تا 194.24.31.255 و با الگوی 255.255.240.0 خواهد بود. در اینجا دانشگاه ادینبورو تقاضای ۱۰۲۴ آدرس داده و فضای 194.24.8.0 تا 194.24.11.255 با الگوی 255.255.252.0 به او تعلق می‌گیرد. این انتسابها در جدول ۵-۵۹ خلاصه شده‌اند.

دانشگاه	اولین آدرس	آخرین آدرس	تعداد آدرس	الگوی نمایش
Cambridge	194.24.0.0	194.24.7.	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
در دسترس و آزاد	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

شکل ۵-۵۹. انتساب آدرسهای IP.

۱. به عبارت ساده به دلیل کم بودن تعداد شبکه‌ها جداول مسیریابی برای کلاسهای A و B در ساختمان داده‌ای شبیه به‌آرایه ذخیره می‌شد. ـم

حال جداول مسیریابی در تمام مسیریابهای واقع بر ستون فقرات اینترنت در جهان باید با این سه درایهٔ جدید به‌هنگام شود. هر درایه دارای یک آدرس مبنا و یک الگوی زیرشبکه است. این درایه‌ها در مبنای دو عبارتند از:

	آدرس	الگوی زیرشبکه (Subnet Mask)
C:	11000010 00011000 00000000 00000000	11111111 11111111 11111000 00000000
E:	11000010 00011000 00001000 00000000	11111111 11111111 11111100 00000000
O:	11000010 00011000 00010000 00000000	11111111 11111111 11110000 00000000

حال ببینیم وقتی که بسته‌ای با آدرس 194.24.17.4 وارد یک مسیریاب می‌شود چه اتفاقی می‌افتد. این آدرس به صورت دودویی عبارت است از:

11000010 00011000 00010001 00000100

ابتدا این آدرس با الگوی زیرشبکه کمبریج، AND می‌شود و نتیجهٔ زیر بدست می‌آید:

11000010 00011000 00010000 00000000

این مقدار با آدرس مبنای دانشگاه کمبریج مطابقت ندارد. حال مجدداً آدرس اصلی با الگوی زیرشبکه دانشگاه ادینبورو AND شده و نتیجهٔ زیر بدست می‌آید:

11000010 00011000 00010000 00000000

این مقدار نیز با آدرس مبنای دانشگاه ادینبورو تطابق ندارد و همین کار برای دانشگاه آکسفورد تکرار شده مقدار زیر بدست می‌آید:

11000010 00011000 00010000 00000000

این مقدار با آدرس مبنای دانشگاه آکسفورد مطابقت دارد. اگر هیچ مورد تطبیق دیگری در جدول یافت نشد بسته بر روی خطی ارسال می‌شود که در درایهٔ متناظر با شبکهٔ دانشگاه آکسفورد درج شده است.

حال اجازه بدهید، آدرس این سه دانشگاه را از دید یک مسیریاب در نیراسکای اوهاما بررسی کنیم. این مسیریاب چهار خط به مینیاپولیس، نیویورک، دالاس و دِنوِر دارد. وقتی نرم‌افزار مسیریاب اوهاما، این سه درایهٔ جدید را جهت درج در جدول مسیریابی خود دریافت می‌دارد، متوجه می‌شود که قادر است هر سه تای آنها را در یک «درایهٔ واحد و تجمیع شده» (Aggregate Entry) به صورت 194.24.0.0/19 ادغام نماید.[1] آدرس و الگوی زیرشبکه در مبنای دو به صورت زیر است:

11000010 00000000 00000000 00000000 11111111 11111111 11100000 00000000

طبق این درایه تمام بسته‌هایی که به مقصد یکی از این سه دانشگاه روانه شده‌اند به سوی نیویورک هدایت می‌شوند. با تجمیع این سه درایه، مسیریاب اوهاما توانسته به میزان دو درایه حجم جدول خود را کاهش بدهد.

به همین ترتیب اگر مسیریاب نیویورک برای تمام ترافیک منتهی به انگلستان فقط یک خط به لندن داشته باشد او نیز سه درایهٔ فوق را در یک درایه ادغام می‌کند ولیکن اگر برای لندن و ادینبورو دو خط مجزا داشته باشد باید هر سه‌تای آنها را بطور مجزا در جدول خود ذخیره کند. عمل تجمیع (Aggregation) در اینترنت به طور گسترده‌ای مورد استفاده قرار گرفته تا حجم جداول مسیریابی کاهش یابد.

آخرین نکته در این مثال آن است که بر طبق درایهٔ ادغام شده در جدول مسیریابی مسیریاب واقع در اوهاما

---

۱. از آن جهت امکان تجمیع این سه آدرس وجود داشته که بسته‌هایی که مقصدشان هریک از این سه دانشگاه است باید بر روی خط خروجی یکسانی بروند. ـم

حتی بسته‌هایی که به آدرس اختصاص داده نشده روانه هستند [یعنی آدرسهای بین 194.24.12.0 تا 194.24.15.255] نیز به سوی نیویورک هدایت می‌شوند. مادامی که این آدرسها به کسی اختصاص داده نشده هیچ مشکلی به وجود نمی‌آید چرا که بنا نیست بسته‌هایی با این آدرسها تولید شوند. ولی اگر این بلوک آدرس ، به شرکتی در کالیفرنیا داده شود باید درایه‌ای جدید به شکل 194.24.12.0/22 در جداول مسیریابی تمام مسیریابها درج شود تا بسته‌هایی به مقصد این شبکه نیز بدرستی مسیریابی شوند.

NAT[1] : ترجمهٔ آدرسهای شبکه

آدرسهای IP کمیاب و ارزشمند هستند: یک ISP ممکن است یک بلوک آدرس با الگوی 16/ (همان کلاس B سابق) و توانایی آدرس‌دهی ۶۵۵۳۴ ماشین میزبان، داشته باشد. اگر تعداد مشتریان این ISP از این تعداد بیشتر شود مشکل بهم می‌زند. برای مشتریان خانگی که از طریق خطوط تلفن متصل می‌شوند، راه حل این مشکل آن است که وقتی مشتری شماره‌گیری کرد و وارد شد به او موقتاً یک آدرس IP پویا اختصاص داده شود و پس از پایان نشست و قطع ارتباط، این آدرس پس گرفته شود. در این روش شبکه‌ای با آدرس 16/ (کلاس B) می‌تواند حداکثر ۶۵۵۳۴ کاربر فعّال داشته باشد که این تعداد حتی برای یک ISP با چند صدهزار مشتری نیز کفایت می‌کند. به محض آنکه یک نشست خاتمه یافت آدرس IP منتسب شدهٔ قبلی، به تماس گیرندهٔ بعدی داده می‌شود. این استراتژی اگرچه برای یک ISP با تعداد متوسطی از کاربران خانگی به خوبی کار می‌کند ولی برای ISPهایی که به مشتریان اداری خدمات می‌دهند مفید نیست.

مشکل از اینجا ناشی می‌شود که مشتریان اداری انتظار دارند که حداقل در ساعات کاری روز خطی دائم و فعال (On-Line) داشته باشند. امروزه، چه دفاتر کوچک اداری مثل یک آژانس مسافرتی با سه کارمند و چه شرکتهای بزرگ که دارای تعداد زیادی کامپیوتر و شبکهٔ محلی هستند، نیاز به خط دائم و فعال دارند. برخی از این کامپیوترها، PC کارمندان و برخی دیگر مثلاً سرویس دهنده‌های وب هستند. عموماً در هر LAN یک مسیریاب وجود دارد که از طریق یک خط اجاره‌ای (Leased) به ISP متصل شده است. چنین ساختاری متضمن آن است هر کامپیوتر آدرس IP خود را داشته باشد و به طور روزانه تغییر نکند. در نتیجه، تعداد کل کامپیوترهایی که در اختیار مشتریان اداری است نباید از تعداد آدرسهای IP متعلق به ISP بیشتر شود. برای آدرس 16/ ، حداکثر تعداد کامپیوترها ۶۵۵۳۴ است. برای یک ISP با دهها هزار مشتری اداری، این فضا سریعاً اشباع می‌شود.

آنچه که مشکل را حادّتر می‌کند آن است که روزبه‌روز بر تعداد مشترکین اینترنت از طریق مودمهای کابلی ADSL افزوده می‌شود. ویژگی چنین سرویسی عبارت است از: (۱) کاربر یک آدرس IP دائم و ثابت می‌گیرد. (۲) هزینهٔ شماره‌گیری و اتصال ندارد (مگر یک هزینهٔ ثابت ماهانه) بدین ترتیب اینگونه کاربران همیشه در شبکه حضور دارند. این موضوع، مشکل کمبود آدرسهای IP را افزایش می‌دهد. در اینجا تخصیص موقت آدرسهای IP (شبیه به مکانیزمی که برای کاربران تلفنی داشتیم) عملی نیست.

حتی از این هم پیچیده‌تر آنکه ممکن است کاربران ADSL و اینترنت کابلی دارای دو یا چند کامپیوتر در خانه باشند و تمام اعضای خانواده از طریق همین خط فعال و مشترک به ISP متصل شوند. یک راه حل آن است که تمام PCها از طریق یک LAN به هم متصل شده و با یک مسیریاب به ISP وصل شوند. از دیدگاه ISP شبکهٔ این خانواده فرقی با یک دفتر اداری کوچک ندارد.

مشکل کمبود آدرسهای IP یک مسئله تئوریک نیست که در آینده‌ای دور رخ بدهد. همین الان با این مشکل مواجه هستیم. راه حل طولانی مدت و همیشگی این مشکل آنست که به سوی IPv6 (که آدرسهای آن ۱۲۸ بیتی

---

۱. Network Address Translation

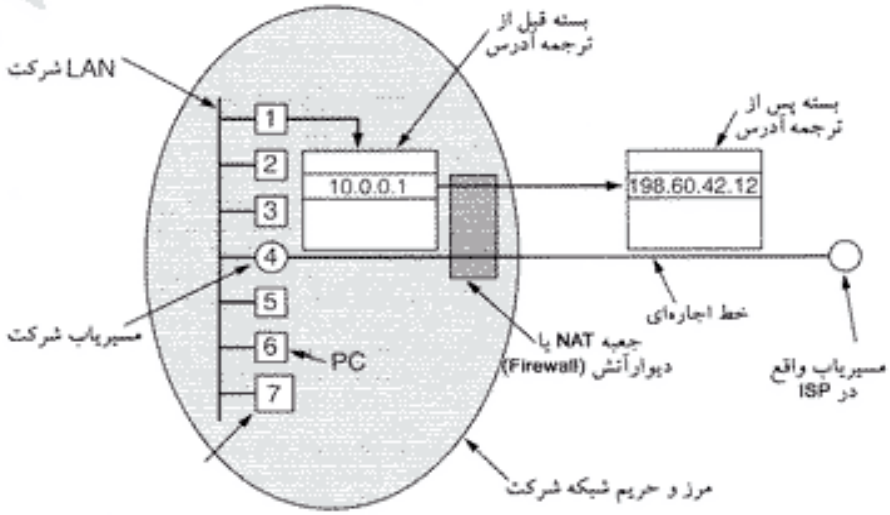
است) حرکت نماییم ولیکن گذار از نسخهٔ ۴ به نسخهٔ ۶ به آهستگی صورت می‌گیرد و سالها طول می‌کشد تا این تغییر به طور کامل انجام شود. در نتیجه بسیاری افراد احساس کردند که به یک راه حل سریع و کوتاه مدت نیاز است. این راه حل سریع، NAT (ترجمهٔ آدرس شبکه) است که در RFC 3022 تشریح شده و در ادامه آنرا مختصراً بررسی می‌نماییم. برای کسب آگاهی بیشتر به مرجع (Dutcher, 2001) مراجعه نمایید.

ایدهٔ اصلی در NAT آن است که به هر شرکت یک یا تعداد کمی آدرس IP معتبر و جهانی اختصاص بدهیم. درون این شرکت، هر کامپیوتر دارای یک آدرس IP یکتا است که برای مسیریابی ترافیک داخلی بکار می‌آید. با این حال وقتی بسته‌ای بخواهد شرکت را ترک کرده و به ISP برود باید قبل از خروج ترجمهٔ آدرس صورت بگیرد. برای آنکه این روش ممکن باشد سه محدوده از فضای آدرس IP جهت بکارگیری در شبکه‌های داخلی، به صورت «خصوصی» (Private) تعریف شده است و شرکتها می‌توانند به صورت دلخواه از آنها استفاده کنند. [نیازی به ثبت جهانی آنها نیست.] تنها قانون آن است که هیچ بسته‌ای نباید با چنین آدرسی بر روی اینترنت ظاهر شود. این سه محدودهٔ رزرو شده عبارتند از:

10.0.0.0	- 10.255.255.255/8	(16,777,216 Hosts)
172.16.0.0	- 172.31.255.255/12	(1,048,576 Hosts)
192.168.0.0	- 192.168.255.255/16	(65535 Hosts)

در محدودهٔ اول ۱۶۷۷۷۲۱۶ آدرس (به استثنای ۰ و ۱-) در دسترس است و عموماً اکثر شرکتها از آن استفاده می‌کنند هر چند نیازی به چنین تعداد آدرسی نداشته باشند.

عملکرد NAT در شکل ۵-۶۰ نشان داده شده است. ماشینها در درون شبکه دارای یک آدرس یکتا به فرم 10.x.y.z هستند. ولیکن وقتی بسته‌ای بخواهد مرز شرکت را ترک کند ابتدا باید از درون یک «جعبهٔ NAT» (NAT BOX) عبور کرده و آدرس مبداء آن با آدرس IP حقیقی شرکت جانشین شود. مثلاً در شکل ۵-۶۰ آدرس 10.0.0.1 با آدرس 198.60.42.1 عوض شده است. اغلب «جعبهٔ NAT» در یک «دیوار آتش» (Firewall) ادغام می‌شود تا این ابزار ضمن ترجمهٔ آدرس، امنیت شبکه را نیز با نظارت دقیق بر ورود و خروج اطلاعات تضمین نماید. در فصل ۸ مفهوم «دیوار آتش» را بررسی خواهیم کرد. همچنین می‌توان «جعبهٔ NAT» را در مسیریاب شرکت قرار داد. اکثر مسیریابهای امروزی از فرآیند NAT پشتیبانی می‌کنند.



شکل ۵-۶۰. مکان و عملکرد «جعبهٔ NAT»

تا اینجا جزئیات کمی از فرآیند NAT مطرح کرده‌ایم: وقتی پاسخ یک بسته بر می‌گردد (مثلاً از سرویس‌دهندهٔ وب) طبعاً آدرس ماشین گیرندهٔ پاسخ، 192.60.42.1 است. سؤال این است که جعبهٔ NAT از کجا بداند که آدرس کدام ماشین داخلی را به جای آن قرار بدهد؟ مسئلهٔ اصلی در NAT همین نکته است. اگر فیلدی اضافی در سرآیند بستهٔ IP وجود داشت می‌شد از آن برای درج آدرس واقعی گیرندهٔ بسته بهره گرفت ولیکن در سرآیند بسته تنها یک بیت بلااستفاده مانده است. همچنین می‌توان یک گزینهٔ جدید [در فضای فیلد اختیاری Option] تعریف کرد تا آدرس حقیقی ماشین مبداء بسته را نگاه دارد ولی انجام این کار مستلزم آن است که کُد نرم‌افزار IP در تمام ماشینها و در کل اینترنت تغییر کند تا گزینهٔ جدید به رسمیت شناخته شده و به درستی تعبیر شود. این راه حل نیز فرآیند زمان‌بری است و مشکل را در کوتاه مدت حل نخواهد کرد.

آنچه که بطور واقعی اتفاق می‌افتد به نحو ذیل است: طراحان NAT بدین نتیجه رسیده بودند که اغلب بسته‌های IP در درون فیلد دادهٔ خود یک بستهٔ TCP یا UDP حمل می‌کنند. هرگاه در فصل ششم TCP و UDP را بررسی کردیم، خواهید دید که هر دوی این پروتکلها دارای سرآیندی برای بسته‌های خود هستند که دو فیلد «**شمارهٔ پورت مبداء**» و «**شمارهٔ پورت مقصد**» جزو آنهاست. در زیر اگرچه تمرکز ما بر پورتهای TCP است ولی همین قضیه برای پورتهای UDP نیز صادق است. این پورتها که اعداد صحیح ۱۶ بیتی هستند، مشخص می‌کنند که اتصال TCP از چه پروسه‌ای شروع و به چه پروسه‌ای ختم می‌شود. این شمارهٔ پورتها فیلدهای لازم برای عملکرد NAT را فراهم آورده‌اند.

هرگاه یک پروسه بخواهد یک اتصال TCP با یک پروسهٔ راه دور برقرار کند یک شمارهٔ پورت بلااستفاده برای خود بر می‌گزیند. این پورت، اصطلاحاً «پورت مبداء» نام دارد و به کد برنامه TCP تفهیم می‌کند که باید بسته‌های ورودی با این شمارهٔ پورت را برای او بفرستد. همچنین هر پروسه یک شمارهٔ پورت مقصد تعیین می‌کند تا مشخص شود که بسته‌ها باید به کدام پروسه در ماشین مقصد تحویل شود. شمارهٔ پورتهای صفر تا ۱۰۲۳ برای سرویس دهنده‌های مشهور رزرو شده است. به عنوان مثال پورت شمارهٔ ۸۰ توسط سرویس دهنده‌های وب بکار گرفته شده است، لذا برنامه‌های مشتری (Client) براحتی با آنها ایجاد ارتباط می‌کنند. کوتاه سخن آنکه، هر پیام خروجی از TCP دارای شمارهٔ پورت مبداء و شماره پورت مقصد است و این دو شمارهٔ پورت هویت پروسه‌های طرفین ارتباط را مشخص می‌نمایند.

تمثیلی از یک نمونه می‌تواند به فهم شماره‌های پورت کمک کند: یک شرکت را در نظر بگیرید که دارای یک شماره تلفن اصلی و واحد است. وقتی افراد با این شماره تماس می‌گیرند بلافاصله اپراتور مربوطه از آنها سؤال می‌کند که کدام شمارهٔ داخلی مدّنظر آنهاست؛ سپس خط داخلی را وصل می‌کند. شمارهٔ اصلی به مثابهٔ آدرس IP است و شماره‌های داخلی، مشابه با شماره پورت هستند. پورتها، ۱۶ بیت آدرس اضافی دیگر هستند که هویت پروسهٔ گیرندهٔ بسته‌ها را مشخص می‌نمایند.

با استفاده از فیلد «شمارهٔ پورت مبداء» می‌توان مشکل نگاشت آدرسها در NAT را حل کرد هر گاه بسته‌ای برای خروج از شبکه به NAT وارد شود، آدرس مبداء آن که به شکل 10.x.y.z است با آدرس IP حقیقی و معتبر شرکت عوض می‌شود. مضاف بر این فیلد شمارهٔ پورت مبداء (TCP Source Port) با عددی عوض می‌شود که در حقیقت این عدد اندیس جدول ترجمهٔ آدرس در جعبهٔ NAT است. هر یک از درایه‌های این جدول، آدرس IP اصلی و همچنین شمارهٔ پورت واقعی آن بسته را نگه می‌دارند. در آخر کد کشف خطای بستهٔ TCP و بستهٔ IP از نو محاسبه و در بسته قرار داده می‌شود. [چرا که هم فیلد شمارهٔ پورت و هم فیلد آدرس IP مبداء در جعبهٔ NAT عوض می‌شود. -م] عوض کردن مقدار فیلد پورت مبداء (Source Port) الزامی است چراکه ممکن است بطور همزمان از دو ماشین به آدرسهای 10.0.0.1 و 10.0.0.2 یک اتصال TCP اتفاقاً با شمارهٔ پورت مبداء یکسان (مثلاً

۵۰۰۰) ایجاد شود، لذا شمارهٔ پورت مبداء نمی‌تواند هویت واقعی پروسهٔ ارسال کنندهٔ بسته‌ها را مشخص کند.[۱]

وقتی بسته‌ای از طریق ISP به جعبهٔ NAT وارد می‌شود ابتدا مقدار فیلد پورت مبداء استخراج شده و از آن به عنوان اندیس جدول نگاشت در جعبهٔ NAT استفاده می‌شود. پس از پیدا شدن درایهٔ متناظر، آدرس IP داخلی و شمارهٔ اصلی پورت مبداء بسته استخراج شده و در درون بسته قرار می‌گیرد. سپس این بسته از طریق مسیریاب داخلی شرکت، برای تحویل به آدرس 10.x.y.z مسیر طبیعی خود را طی می‌کند.

همچنین از NAT می‌توان برای تخفیف مشکل کمبود آدرس IP برای کاربران ADSL و کاربران کابلی بهره گرفت. هر گاه ISP بخواهد به هر یک از این کاربران آدرسی انتساب بدهد، از آدرسی در فضای 10.x.y.z بهره می‌گیرد. قبل از آنکه بسته‌های ماشین کاربران، ISP را ترک کنند و به اینترنت وارد شوند باید ابتدا وارد جعبهٔ NAT شده و آدرس غیرحقیقی و محلی آنها به آدرس واقعی متعلق به ISP نگاشته شود. در مسیر برگشت، عکس فرآیند نگاشت انجام می‌شود. بدین نحو از دیدگاه اینترنت، این ISP (و کاربران ADSL یا کابلی آن) دقیقاً مثل یک شرکت بزرگ به نظر می‌رسند، هر چند تعداد آدرسهای واقعی و معتبر ISP ناچیز است.

اگرچه این روش مشکل کمبود آدرسهای IP را حل می‌کند ولیکن بسیاری از افراد در جامعه اینترنت از آن به عنوان کاری بی‌ارزش و مردود یاد می‌کنند. برخی از مخالفتهای آنان را به اختصار ارائه می‌نماییم. اول آنکه NAT مدل معماری IP را نقض می‌کند چراکه در این مدل بیان شده که آدرس IP به صورت یکتا ماشینی واحد را در کل جهان مشخص می‌نماید. ساختار تمام نرم‌افزارهای اینترنت با تکیه بر این واقعیت بنیان گذاشته شده است. با NAT ممکن است هزاران ماشین از آدرس 10.0.0.1 استفاده کنند (و می‌کنند).

دوم آنکه NAT اینترنت را از حالت «بدون‌اتصال» به شبکه‌ای «اتصال‌گرا» تبدیل می‌نماید. مسئله اینجاست که جعبهٔ NAT باید اطلاعاتی را در خصوص نگاشت اتصالهایی که از آن می‌گذرند در خود نگاه دارد. نگهداری وضعیت هر اتصال ویژگی شبکه‌های اتصال‌گراست و سنخیتی با شبکه‌های بدون‌اتصال ندارد. اگر جعبهٔ NAT به ناگاه از کار بیفتد و جدول نگاشت آن از دست برود تمام اتصالات TCP برقرار شده از دست می‌رود. بدین ترتیب با وجود NAT، اینترنت به یک شبکهٔ آسیب‌پذیر مدار مجازی تبدیل می‌شود.

سوم آنکه، NAT اصول بنیانی لایه‌بندی پروتکلها را نقض می‌کند: لایهٔ k نباید هیچ تصوری از آنچه که لایهٔ k+1 در فیلد حمل داده از بستهٔ او قرار می‌دهد، داشته باشد یا در آن دخالتی کند.[۲] اصل اساسی در معماری لایه‌به‌لایه آنست که تمام لایه‌ها مستقل از دیگری باشند. اگر مثلاً زمانی TCP به نسخهٔ TCP-2 ارتقاء یابد و سرآیند بسته‌ها تغییر کنند (مثلاً شمارهٔ پورتها ۳۲ بیتی شوند)، NAT از کار خواهد افتاد. ایدهٔ اصلی در پروتکلهای لایه‌ای آن بوده که تغییر در یک لایه نیازی به تغییر در لایه‌های دیگر نداشته باشد در حالیکه NAT این عدم وابستگی را از بین می‌برد.

چهارم آنکه پروسه‌های اینترنت مجبور به استفاده از TCP یا UDP نیستند. اگر فرضاً کاربری بر روی ماشین A تصمیم بگیرد برای محاوره و مبادلهٔ داده با کاربری بر روی ماشین B از یک پروتکل جدید در لایهٔ انتقال استفاده کند (مثلاً برای کاربردهای چند رسانه‌ای)، وجود NAT منجر به عدم کارکرد آن برنامهٔ کاربردی خواهد شد چراکه NAT نخواهد توانست فیلد پورت مبداء را به درستی پیدا کرده و از آن استفاده نماید.

پنجم آنکه برخی از برنامه‌های کاربردی آدرس IP ماشین خود را در متن اطلاعات ارسالی قرار می‌دهند. گیرنده نیز این آدرس را استخراج کرده و از آن در جایی استفاده می‌کند. از آنجایی که NAT چیزی در مورد این

---

۱. بعبارت روشنتر چون تمام بسته‌ها در برگشت آدرس IP یکسانی دارند فلذا این آدرس پورت هر بسته است که هویت گیرندهٔ واقعی بسته را مشخص می‌کند و طبعاً NAT باید در هنگام خروج بسته‌ها ضمن عوض کردن شمارهٔ پورت، یکتا بودن آنرا تضمین کند. -م

۲. بعبارتی از دیدگاه لایهٔ k آنچه که از لایهٔ k+1 می‌رسد تعدادی بایت خام است. -م

آدرسهای مخفی نمی‌داند فلذا قادر به تغییر آنها نبوده و هر گونه تلاش برای استفاده از این آدرسهای ناصحیح (در ماشین راه دور) با شکست مواجه می‌شود. FTP ، یعنی استاندارد انتقال فایل در اینترنت به همین ترتیب عمل می‌کند و با وجود NAT از کار می‌افتد مگر آنکه اقدامات احتیاطی خاصی به عمل آید. به دلیل مشابه، پروتکل H.323 که برای تلفن اینترنتی کاربرد دارد (و در فصل هفتم به معرفی آن خواهیم پرداخت) با وجود NAT کار نخواهد کرد. البته می‌توان با تغییرات اصلاحی در NAT آن را بکار گرفت ولی این که با معرفی هر برنامهٔ کاربردی جدید مجبور به اصلاح NAT شویم، اصلاً ایدهٔ جالبی نیست.

ششم آنکه چون فیلد آدرس پورت مبداء، ۱۶ بیتی است، حداکثر ۶۵۵۳۶ ماشین را می‌توان به یک آدرس IP واحد نگاشت. این تعداد حقیقتاً مقدار کمی است (گذشته از آن، ۴۰۹۶ شمارهٔ پورت نیز برای کاربردهای خاص کنار گذاشته شده‌اند)، ولیکن اگر تعداد آدرسهای IP معتبر و جهانی که در اختیار ISP قرار دارد، بیش از یکی باشد به ازای هر یک می‌توان ۶۱۴۴۰ ماشین را با آدرسهای غیرحقیقی مدیریت کرد.

این مشکلات بهمراه مسائل دیگر NAT ، در RFC 2993 تشریح شده است. عموماً مخالفین NAT می‌گویند که با حل نازیبا و موقتی مسئله کمبود آدرسهای IP، اصرار بر روی راه حل واقعی و نهایی که همانا رفتن به طرف IPv6 است، کم می‌شود و تعویق انداختن در این تغییر و تحول اصلاً خوب نیست!

۳-۶-۵ پروتکلهای کنترل اینترنت

اینترنت مضاف بر IP که برای انتقال داده‌ها کاربرد دارد، چندین پروتکل کنترلی دیگر دارد که همگی در لایهٔ شبکه به کار گرفته می‌شوند. این پروتکلها عبارتند از: ICMP ، ARP ، RARP ، BOOTP و DHCP. در این بخش، این پروتکلها را به ترتیب مرور می‌کنیم.

ICMP[1] : پروتکل ارسال پیامهای کنترلی در اینترنت

عملکرد اینترنت توسط مسیریابها به صورت دقیق نظارت و کنترل می‌شود. هر گاه اتفاق غیرمنتظره‌ای بیفتد، رخداد مزبور توسط پروتکل ICMP گزارش می‌شود. این پروتکل همچنین برای آزمایش و رفع عیب شبکه کاربرد دارد. تقریباً یک دوجین از پیامهای ICMP به منظور اطلاع‌رسانی تعریف شده است. مهمترین این پیامها در شکل ۵-۶۱ فهرست شده‌اند. هر پیام ICMP مستقیماً درون یک بستهٔ IP جاسازی و ارسال می‌شود.

نوع پیام	توصیف عملکرد
Destination unreachable	بهر دلیلی بسته را نمی‌توان به مقصد تحویل داد.
Time exceeded	زمان حیات بسته به صفر رسیده است.
Parameter problem	فیلدی از سرآیند بسته مقدار نامعتبر داشته است.
Source quench	بسته دعوت به آرامش
Redirect	حاوی اطلاعاتی در خصوص جغرافیای مسیر و اعلام اشتباه در مسیریابی
Echo	درخواست از یک ماشین تا اگر فعال است پاسخ بدهد.
Echo reply	پاسخ به پیام Echo بمنظور تایید فعالیت
Timestamp request	همانند پیام Echo بهمراه مهر زمان
Timestamp reply	همانند پیام Echo Reply بهمراه مهر زمان

شکل ۵-۶۱. انواع پیامهای اساسی ICMP.

۱. Internet Control Message Protocol

از پیام DESTINATION UNREACHABLE زمانی استفاده می‌شود که زیرشبکه یا مسیریاب نتواند موقعیت مقصد را تشخیص بدهد یا وقتی که بیت DF[1] در بسته‌ای به ۱ تنظیم شده باشد و آن بسته مجبور به عبور از شبکه‌ای باشد که طول بسته‌های آن کوچک است. [طبعاً بسته حذف می‌شود.]

پیام TIME EXCEEDED زمانی ارسال می‌شود که بسته به دلیل صفر شدن شمارندهٔ TTL (شمارندهٔ زمان حیات) حذف گردد. این رخداد نشانهٔ آن است که بسته‌ها در حلقهٔ بی‌نهایت افتاده‌اند یا آنکه ازدحام سنگینی رخ داده یا آنکه مقدار اولیهٔ این شمارنده بسیار کم بوده است.

پیام PARAMETER PROBLEM مشخص‌کنندهٔ آنست که در سرآیند بستهٔ IP مقدار نامعتبر و اشتباهی درج شده است. [لذا مسیریاب آن را حذف کرده و این پیام را ارسال نموده است.] این مشکل از وجود یک اشکال در نرم‌افزار IP ماشین فرستنده یا احتمالاً نرم‌افزار مسیریابهای میانی پرده بر می‌دارد.

پیام SOURCE QUENCH سابقاً برای آن بکار می‌رفته تا به ماشینهایی که بیش از اندازه بسته ارسال می‌کنند اخطار داده شود. هر ماشین که این پیام را دریافت کند باید نرخ ارسال بسته‌های خود را کاهش بدهد. ولیکن از این پیام به ندرت استفاده می‌شود چرا که وقتی ازدحام رخ بدهد، ارسال چنین پیامهایی بر هیزم این آتش می‌افزاید. کنترل ازدحام در اینترنت عمدتاً در لایهٔ انتقال انجام می‌شود. این روش را در فصل ششم خواهیم آموخت.

از پیام REDIRECT زمانی استفاده می‌شود که مسیریاب احساس کند بسته‌ای در مسیر غلط قرار گرفته است. این پیام برای گزارش خطای احتمالی به ماشین مبداء بسته، کاربرد دارد.

پیامهای ECHO و ECHO REPLY بدین منظور کاربرد دارند که ببینیم آیا یک ماشین مقصد در دسترس و فعال است یا خیر. انتظار می‌رود هر ماشینی که پیام ECHO دریافت می‌کند، در پاسخ به آن، پیام ECHO REPLY را برگرداند.

پیامهای TIMESTAMP REQUEST و TIMESTAMP REPLY مشابه با دو پیام قبلی هستند با این تفاوت که در بدنهٔ پاسخ، زمان دریافت پیام و همچنین زمان ارسال پیام درج می‌شود. از این قابلیت می‌توان برای اندازه‌گیری کارآیی شبکه بهره گرفت.

مضاف بر پیامهای فوق‌الذکر، پیامهای دیگری نیز تعریف شده‌اند که می‌توانید فهرست آنها را در آدرس زیر بیابید: www.iana.org/assignments/icmp-parameters

ARP[2]: پروتکل تحلیل آدرس

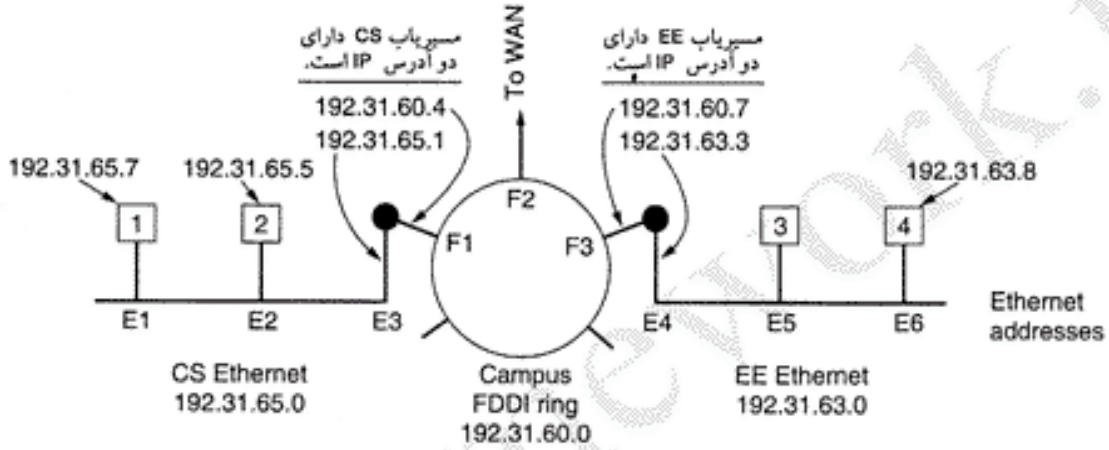
اگرچه هر ماشین در اینترنت دارای یک (یا چند) آدرس IP است، ولیکن این آدرسها حقیقتاً نمی‌توانند برای ارسال بسته‌ها بکار گرفته شوند چرا که سخت‌افزار لایهٔ پیوند داده‌ها هیچ درکی از آدرسهای IP ندارد. امروزه اغلب ماشینهای میزبان در شرکتها یا دانشگاهها به کمک یک کارت واسط شبکه به LAN متصل شده‌اند و این کارتها فقط آدرسهای MAC (آدرس سخت‌افزاری کارت شبکه) را می‌شناسند. به عنوان مثال هر کارت شبکهٔ اترنت پس از تولید در کارخانه، دارای یک آدرس ۴۸ بیتی یکتا است. سازندگان کارتهای اترنت برای تعبیهٔ آدرس در کارتهای تولیدی خود ابتدا از یک مرکز مجاز تقاضا می‌کنند که یک بلوک آدرس به آنها اختصاص بدهد و بدین ترتیب اطمینان حاصل می‌شود که هیچ دو کارت شبکه در کل جهان دارای آدرس یکسان نیست. (تا از هر گونه برخورد دو کارت شبکه با شماره‌های یکسان بر روی LAN اجتناب شود.) کارتهای شبکه، فریمها را براساس آدرسهای ۴۸ بیتی اترنت، ارسال یا دریافت می‌نمایند و هیچ چیزی در خصوص آدرسهای IP نمی‌دانند.

اکنون این سؤال پیش می‌آید که چگونه آدرسهای IP به آدرسهای لایهٔ پیوند داده (مثل آدرسهای اترنت)

---

۱. Don't Fragment ۲. Address Resolution Protocol

نگاشته و تبدیل می‌شوند؟ برای تشریح عملیات نگاشت آدرس از مثال شکل ۵-۶۲ استفاده کرده‌ایم. در این شکل یک دانشگاه کوچک با شبکه‌ای با چند کلاس C (به عبارتی شبکهٔ 24/) به تصویر کشیده شده است. در این شکل دو شبکهٔ اترنت وجود دارد که یکی از آنها با آدرس 192.31.65.0 به دانشکدهٔ کامپیوتر و دیگری با آدرس 192.31.63.0 به دانشکدهٔ برق تعلق دارد. این دو شبکه از طریق یک شبکهٔ حلقه (مثل FDDI) با آدرس 192.31.60.0 که نقش ستون فقرات شبکهٔ کل دانشگاه را ایفاء می‌کند، به یکدیگر متصل شده‌اند. هر ماشین متصل به اترنت دارای یک آدرس MAC یکتا است که در شکل با برچسبهای E1 تا E6 مشخص شده‌اند. هر ماشین متصل به شبکهٔ حلقه نیز آدرس FDDI خود را دارد که آنها نیز با F1 تا F3 نشان داده شده‌اند.



شکل ۵-۶۲. سه شبکهٔ بهم متصل با الگوی 24/: دو شبکهٔ اترنت و یک شبکهٔ حلقه از نوع FDDI.

حال ابتدا بررسی کنیم که کاربری بر روی ماشین میزبان ۱ چگونه بسته‌ای را برای کاربری بر روی ماشین ۲ می‌فرستد. فرض را بر آن گذاشته‌ایم که فرستنده، نام گیرندهٔ مورد نظر را می‌داند و مثلاً این نام mary@eagle.cs.uni.edu است. در اولین گام باید آدرس IP ماشین eagle.cs.uni.edu را بدست بیاورد. این جستجو از طریق «سیستم نام دامنه» (DNS) که در فصل هفتم آن را مطالعه خواهیم کرد، انجام می‌گیرد. در اینجا فرض می‌کنیم که DNS، آدرس IP ماشین ۲ را 192.31.65.5 برگردانده است.

در اینجا نرم‌افزار لایهٔ بالاتر در ماشین ۱ بسته‌ای می‌سازد و درون فیلد «آدرس مقصد» آن مقدار 192.31.65.5 را درج می‌کند و آن را جهت ارسال تحویل نرم‌افزار IP می‌دهد. نرم‌افزار IP با بررسی آدرس بدین نتیجه می‌رسد که ماشین مقصد، در شبکهٔ خودش قرار گرفته است[1] ولی محتاج روشی است تا بتواند آدرس اترنت ماشین مقصد را پیدا کند. یک راه حل آنست که یک فایل پیکربندی در جایی از سیستم ذخیره شده باشد و از طریق آن آدرس IP به آدرسهای اترنت نگاشته شود. اگرچه این روش عملی است ولی برای موسساتی که هزاران ماشین دارند به روز نگه داشتن چنین فایلهایی، کاری بسیار وقتگیر و منشاء خطاهای سهوی است.

راه حل بهتر آنست که ماشین میزبان ۱ بسته‌ای را به صورت فراگیر (Broadcast) بر روی اترنت پخش کند و از همه سؤال کند که: «آدرس 192.31.65.5 متعلق به کیست؟» این اعلام همگانی به یکایک ماشینهای شبکهٔ اترنت با آدرس 192.31.65.0 خواهد رسید. تنها ماشین ۲ به این سؤال پاسخ خواهد داد و آدرس اترنت خود یعنی E2 را اعلام خواهد کرد. بدین طریق ماشین ۱ متوجه می‌شود که آدرس 192.31.65.5 متلعق به ماشینی است که آدرس

---

۱. نرم‌افزار IP از آنجا متوجه می‌شود که ماشین مقصد در شبکهٔ محلی خودش واقع است که طبق الگوی 24/ بخش شمارهٔ شبکه خودش و شمارهٔ شبکهٔ مقصد یکسان است. (شمارهٔ شبکهٔ هردو 192.31.65.0 است.) -م

اترنت آن E2 است. پروتکلی که برای این پرسش و پاسخ بکار می‌رود **ARP** نام دارد و تقریباً تمام ماشینهای اینترنت آنرا اجرا کرده‌اند. ARP در سند RFC 826 تبیین شده است.

مزیت استفاده از ARP به جای ذخیره فایلهای پیکربندی، سادگی آنست. مدیر سیستم فقط باید برای هر ماشین، آدرس IP و الگوی زیرشبکهٔ (Subnet Mask) آنرا مشخص کند. مابقی کارها را ARP انجام می‌دهد.

پس از بدست آمدن آدرس E2، نرم‌افزار IP در ماشین میزبان ۱، یک فریم اترنت به آدرس E2 ساخته و بستهٔ IP (با آدرس 192.31.65.5) را در درون فیلد دادهٔ آن قرار داده و آن را بر روی اترنت روانه می‌کند. کارت شبکهٔ ماشین ۲ این فریم را گرفته و تشخیص می‌دهد که متعلق به خود اوست؛ آنرا پذیرفته و یک «وقفه»[1] تولید می‌کند. نرم‌افزار راه‌انداز اترنت، بستهٔ IP را از درون فریم استخراج کرده و آنرا به نرم‌افزار IP تحویل می‌دهد. نرم‌افزار IP متوجه می‌شود که این بسته به آدرس صحیحی آمده و طبعاً آن را پردازش می‌نماید.

برای آنکه ARP کارآمدتر عمل کند می‌توان بهینه‌سازیهایی انجام داد. اولین بهینه‌سازی آنست که هر گاه ARP اجرا شد و آدرسی را بدست آورد، آن را در «حافظهٔ نهان»[2] خود ذخیره نماید تا در دفعات بعدی بتواند با استفاده از این آدرس، سریعاً با ماشین مربوطه ارتباط برقرار کند. دفعهٔ بعد، ARP نگاشت آدرس IP به آدرس اترنت را در حافظهٔ نهان خود خواهد یافت و نیاز مجدد به سؤال همگانی نیست. در بسیاری از حالات، ماشین ۲ (پس از دریافت بسته) باید پاسخی پس بفرستد و در نتیجه او نیز مجبور است برای یافتن آدرس اترنت فرستندهٔ تقاضا، ARP را اجرا نماید. برای آنکه در اینجا از ارسال فراگیر بسته‌های ARP جلوگیری شود می‌توان بدین نحو عمل کرد که وقتی مثلاً ماشین ۱ برای یافتن آدرس ماشین ۲ به صورت پخش فراگیر از همه سؤال می‌کند، «نگاشت آدرس IP به آدرس اترنت» خود را نیز اعلام نماید. وقتی بسته‌های پخشی ARP به ماشین ۲ می‌رسد جفت آدرس (E1 و 192.31.65.7) وارد حافظهٔ نهان ARP می‌شود تا برای استفاده‌های آتی نیاز به سؤال نباشد. در حقیقت تمام ماشینهای متصل به اترنت می‌توانند این جفت آدرس نگاشته شده را در حافظهٔ نهان ARP خود ذخیره نمایند.[3]

روشی دیگر برای بهینه‌سازی ARP آنست که هر ماشین به محض راه‌اندازی (بوت) «نگاشت آدرس سخت‌افزاری» به آدرس IP خود را به صورت پخش فراگیر به همه اعلام نماید.[4] عموماً این کار بدین نحو انجام می‌گیرد که ماشین در حال بوت، آدرس سخت‌افزاری معادل با IP خودش را سؤال می‌کند!! طبعاً چنین سؤالی هیچ‌پاسخی ندارد ولی نتیجهٔ جانبی آن این است که در اثر پخش این سؤال (که پاسخ خود را بهمراه دارد) بر روی کل شبکهٔ محلی، نگاشت آدرس او به صورت یک درایه (Entry) درون حافظهٔ نهان ARP ماشینهای فعال درج می‌شود. اگر پاسخ چنین سؤالی به صورت غیرمنتظره دریافت گردد، مشخص می‌شود که دو ماشین دارای آدرس IP مشابه هستند. ماشین دوم باید این موضوع را به مدیر سیستم اطلاع داده و بوت نشود.

برای آنکه جدول نگاشت آدرس در ARP بتواند تغییر کند، درایه‌های موجود در حافظهٔ نهان ARP باید پس از چند دقیقه اعتبار خود را از دست بدهند و با سؤال مجدّد، نگاشت آدرسها از نو انجام شود چراکه مثلاً اگر یک کارت شبکهٔ اترنت، خراب و با کارتی جدید عوض شود آدرس اترنت معادل با آدرس IP آن ماشین عوض می‌شود و اگر بقیهٔ ماشینها بخواهند از جدول نگاشت قدیمی خود استفاده کنند این ماشین از دسترس دیگران

---

۱. Interrupt ۲. ARP Cache

۳. بعبارت دیگر وقتی یک ماشین، آدرس سخت‌افزاری یک ماشین دیگر را سؤال می‌کند، جفت آدرس سخت‌افزاری و IP خود را نیز به همه اعلام می‌کند و از آن به بعد ماشینها نیازی به سؤال کردن ندارند. بدین نحو حافظهٔ نهان ARP سریعاً پر شده و فرآیند پرسش و پاسخ بیهوده و اضافی انجام نخواهد شد. ـم

۴. واژه‌های آدرس MAC، آدرس فیزیکی، آدرس سخت‌افزاری، آدرس اترنت همگی معادلند و به آدرس تعبیه شده بر روی کارت شبکه (آدرس لایهٔ پیوند داده‌ها) اشاره دارند. ـم

خارج می‌شود.

حال مجدداً به شکل ۵-۶۲ نگاهی بیندازید: هنگامی که ماشین ۱ بخواهد بسته‌ای برای ماشین ۴ (با آدرس 192.31.63.8) بفرستد با شکست مواجه می‌شود زیرا ماشین ۴ نمی‌تواند پخش فراگیر بسته‌های پرسش را دریافت کند (مسیریابها بسته‌های پخش فراگیر را به شبکه‌های دیگر هدایت نمی‌کنند). برای حل این مشکل دو راه وجود دارد: اول آنکه مسیریاب CS به گونه‌ای پیکربندی شود تا به تمام بسته‌های ARP که در مورد شبکهٔ 192.31.63.0 (یا سایر شبکه‌های محلی) آدرسی را سؤال می‌کنند، جواب بدهد و آدرس سخت‌افزاری خودش را اعلام نماید. در این حالت، ماشین ۱ در جدول خود یک درایه به صورت (E3 و 192.31.63.8) درج می‌نماید و تمام ترافیک خود برای ماشین ۴ را برای مسیریاب محلی می‌فرستد. این راه حل اصطلاحاً Proxy ARP نامیده می‌شود. (ARP وکالتی) راه حل دوم آنست ماشین ۱ فوراً تشخیص بدهد که مقصد مورد نظر او بر روی شبکه‌ای دیگر قرار گرفته و تمام ترافیک راه دور و غیرمحلی خود را به آدرس اترنت مسیریاب پیش‌فرض که در این مثال E3 است بفرستد. [تشخیص محلی یا غیرمحلی بودن آدرسهای IP به کمک الگوی زیرشبکه ـSubnet Maskـ میسّر است.] در این راه حل نیازی نیست که مسیریاب CS بداند که به چه شبکه‌هایی سرویس می‌دهد.

در هر حال آنچه که در ادامه اتفاق می‌افتد آنست که: ماشین ۱ بستهٔ IP را درون فیلد حمل داده از فریم اترنت جاسازی کرده و مقصد فریم را آدرس اترنت E3 قرار می‌دهد. وقتی مسیریاب CS این فریم اترنت را دریافت می‌کند و بستهٔ IP را از درون آن جدا کرده و آدرس IP آن را درون جدول مسیریابی خود جستجو می‌نماید و بر این اساس متوجه می‌شود که تمام بسته‌های متعلق به شبکهٔ 192.31.63.0 باید به مسیریاب با آدرس 192.31.60.7 تحویل شود و اگر از قبل آدرس سخت‌افزاری کارت FDDI متناظر با 192.31.60.7 را نداند، با پخش فراگیر یک بستهٔ ARP بر روی حلقه، آن را سؤال کرده و متوجه می‌شود که این آدرس F3 است. لذا بستهٔ IP را درون فیلد حمل داده از فریم FDDI قرار داده و با درج آدرس F3 در فریم جدید آن را بر روی حلقه قرار می‌دهد.

در مسیریاب EE، نرم‌افزار راه‌انداز کارت FDDI، بسته را از درون فریم استخراج کرده و آن را به نرم‌افزار IP تحویل می‌دهد و IP نیز به روش مشابه متوجه می‌شود که باید این بسته را به آدرس 192.31.63.8 بفرستد. اگر این آدرس IP درون حافظهٔ نهان ARP پیدا نشد، مجدداً با پخش فراگیر یک بستهٔ ARP بر روی شبکهٔ اترنت EE، آن را سؤال می‌کند و متوجه می‌شود که آدرس سخت‌افزاری مقصد E6 است، لذا یک فریم اترنت به آدرس E6 ساخته و بسته را مجدداً درون فیلد حمل دادهٔ آن قرار داده و فریم را بر روی شبکهٔ اترنت EE قرار می‌دهد. وقتی فریم اترنت به ماشین ۴ می‌رسد، بسته از درون آن استخراج شده و جهت پردازش تحویل نرم‌افزار IP می‌شود.

حرکت بسته‌ها از ماشین ۱ به یک شبکهٔ دور در WAN نیز به روشی مشابه با روش قبلی انجام می‌شود با این تفاوت که در مثال بالا مسیریاب CS از طریق جدول مسیریابی خود متوجه می‌شود که باید بسته را برای مسیریاب متصل به WAN با آدرس F2 بفرستد.

پروتکلهای RARP، BOOTP و DHCP

پروتکل ARP مسئله پیدا کردن آدرس اترنت متناظر با یک IP مشخص را حل می‌کند. [به عبارت ساده‌تر آدرس IP یک ماشین را گرفته و آدرس سخت‌افزاری آن ماشین را پیدا کرده، بر می‌گرداند.] برخی اوقات وارون این مسئله باید حل شود، یعنی با داشتن آدرس اترنت یک ماشین، به آدرس IP متناظر با آن نیاز است. بالاخص زمانی با این مسئله مواجه هستیم که یک ایستگاه بدون دیسک سخت بخواهد از طریق شبکه بوت شود. بطور معمول چنین ماشینی «تصویر باینری سیستم عامل» خود را از یک سرویس‌دهندهٔ فایل تحویل گرفته و بارگذاری می‌کند. ولی چگونه می‌تواند آدرس IP خود را بفهمد؟

اولین راه حل ابداعی برای این مسئله آن بود که از پروتکل RARP[1] (تشریح شده در سند RFC 903) استفاده شود. این پروتکل امکان آن را فراهم آورده که ماشین تازه بوت شده آدرس اترنت خود را به صورت فراگیر بر روی شبکه پخش کند و بگوید: «آدرس اترنت ۴۸ بیتی من مثلاً 14.04.05.18.01.25 است. آیاکسی آدرس IP مرا می‌داند؟» سرویس دهندهٔ RARP این درخواست را می‌بیند و در فایلهای پیکربندی خودش، آدرس اترنت اعلام شده را جستجو می‌نماید و آدرس IP متناظر با آن را بر می‌گرداند.

استفاده از RARP بهتر از آنست که آدرس IP یک ماشین در «تصویر حافظه»[2] ارسالی جاسازی شود چراکه این امکان فراهم می‌آید که از «تصویر حافظهٔ» مشابهی برای تمام ماشینها بهره گرفته شود. اگر قرار باشد آدرس IP درون تصویر حافظهٔ ارسالی جاسازی شود، هر ایستگاه در شبکه به «تصویر» خاص خود احتیاج خواهد داشت.

اشکال RARP آنست که بیتهای فیلد آدرس مقصد را در تمام فریمهای ارسالی خود ۱ می‌گذارد تا بدین ترتیب این فریمها به صورت پخش فراگیر به سرویس دهندهٔ RARP برسند ولیکن فریمهای پخش شده بر روی شبکهٔ محلی، توسط مسیریابها به خارج از شبکه هدایت نمی‌شوند، فلذا بر روی هر شبکهٔ محلی باید یک سرویس دهندهٔ RARP وجود داشته باشد. برای حل این مشکل یک پروتکل خاص دیگر به نام BOOTP برای راه‌اندازی ایستگاههای بدون دیسک (Diskless) ابداع شده است. این پروتکل می‌تواند به غیر از آدرس IP ایستگاه بدون دیسک، اطلاعات اضافه‌تری را مثل آدرس IP سرویس دهندهٔ فایل (که تصویر اجرایی سیستم عامل را در اختیار دارد)، آدرس IP مسیریاب پیش فرض، الگوی زیرشبکه (Subnet Mask)، به ایستگاهها ارائه بدهد. برخلاف RARP ، در پروتکل BOOTP از بسته‌های UDP استفاده شده و مسیریابها این بسته‌ها را هدایت می‌نمایند. [لذا به ازای چندین شبکهٔ محلی که از طریق مسیریاب بهم متصل شده‌اند فقط به یک سرویس دهندهٔ BOOTP نیاز است.] پروتکل BOOTP در RFCهای 951 ، 1048 و 1084 تشریح شده است.

مشکل جدی پروتکل BOOTP آنست که جدول نگاشت آدرس IP به آدرس اترنت باید به صورت دستی تنظیم و پیکربندی شود. وقتی ماشین جدیدی به LAN اضافه می‌شود قادر به بوت شدن نیست مگر آنکه مسئول شبکه یک آدرس IP به آن انتساب داده و آن را در قالب: (**آدرس IP + آدرس اترنت**) به صورت دستی در فایل پیکربندی BOOTP وارد نماید. برای آنکه این مرحلهٔ اشکال‌زا حذف شود پروتکل BOOTP پیشرفته‌تر شد و با نام جدید DHCP[3] معرفی گردید. پروتکل DHCP این امکان را فراهم آورده که بتوان آدرس IP ایستگاهها را هم به صورت دستی و هم به صورت خودکار به آنها انتساب داد. این پروتکل در RFCهای ۲۱۳۱ و ۲۱۳۲ تشریح شده است و در اغلب سیستمها جایگزین RARP و BOOTP شده است.

شبیه به RARP و BOOTP ، پروتکل DHCP نیز متکی به یک سرویس دهندهٔ ویژه در شبکه است تا به ماشینهایی که تقاضای آدرس IP می‌کنند، آدرس و اطلاعات لازم را تقدیم نماید. لازم نیست که این سرویس دهنده بر روی همان LAN باشد که ماشین متقاضی آدرس بر روی آن واقع شده است و از آنجایی که ممکن است دسترسی به این سرویس دهنده از طریق پخش فراگیر بسته‌های تقاضا میسر نباشد فلذا بر روی هر LAN به یک «**عامل رلهٔ DHCP**» (DHCP Relay Agent) نیاز است. (به شکل ۵-۶۳ نگاه کنید.)

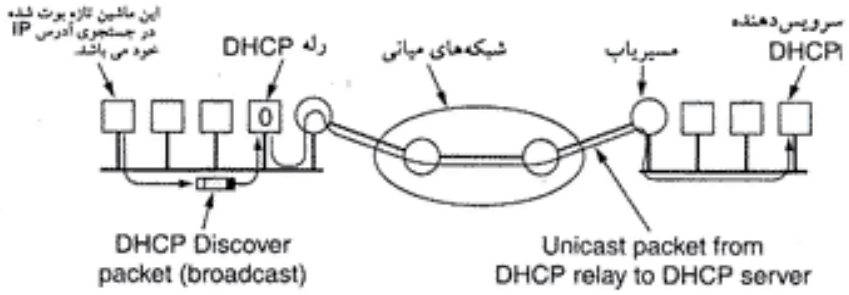
ماشینی که تازه بوت شده برای بدست آوردن آدرس IP خود، بسته‌ای به نام DHCP DISCOVER را به صورت فراگیر بر روی LAN خود منتشر می‌کند. «عامل رلهٔ DHCP» در هر LAN تمام بسته‌های پخش شده

---

۱. Reverse Address Resolution Protocol

۲. منظور از تصویر حافظه یا Memory Image قطعه کد اجرایی و باینری از هستهٔ سیستم عامل است که بلافاصله پس از بارگذاری، اجرا می‌شود. -م

۳. Dynamic Host Configuration Protocol



شکل ۵-۶۳. عملکرد DHCP.

DHCP را می‌گیرد و وقتی متوجه شود که یک بسته از نوع DHCP DISCOVER است آن را به صورت تک‌پخشی (Unicast) برای سرویس دهندهٔ اصلی DHCP (که می‌تواند بر روی شبکه‌ای دور واقع باشد) می‌فرستد. «عامل رلهٔ DHCP» فقط به آدرس IP از سرویس دهندهٔ DHCP احتیاج دارد.

یکی از مسائلی که در خصوص انتساب خودکار آدرسها پیش می‌آید آنست که یک آدرس IP برای چه مدت زمانی در اختیار ماشین قرار بگیرد. اگر ماشینی بدون اطلاع قبلی شبکه را ترک کند و آدرس IP خود را به سرویس دهندهٔ DHCP بر نگرداند، آن آدرس برای همیشه گم می‌شود. به مرور زمان ممکن است آدرسهای زیادتری به این نحو گم شوند. برای آنکه چنین مشکلی اتفاق نیفتد، انتساب آدرسهای IP فقط برای مدت زمان محدود و ثابتی انجام می‌شود. این تکنیک «اجارهٔ IP» (IP Leasing) نام دارد. هر ماشین قبل از آنکه مهلت اجارهٔ آدرس IP او به سر برسد، باید با ارسال بسته‌ای خاص تقاضای تجدید اجاره کند. اگر ماشین موفق به ارسال این تقاضا نشد یا تقاضای تجدید پذیرفته نشود، ماشین میزبان نمی‌تواند بیش از این از آدرس IP اجاره‌ای خود استفاده نماید.[۱]

۵-۶-۴ OSPF : پروتکل مسیریابی برای دروازه‌های درونی

تا اینجا بررسی پروتکلهای کنترلی اینترنت را به اتمام رسانده‌ایم. زمان آن فرا رسیده که به مورد بعدی بپردازیم: «مسیریابی در اینترنت». قبلاً اشاره کردیم که شبکهٔ اینترنت از تعداد بسیار زیادی «**سیستم خودمختار**» (Autonomous System) تشکیل شده است. هر سیستم خودمختار که اصطلاحاً AS نامیده می‌شود توسط سازمان یا نهاد خاصی پیاده و اداره می‌شود و طبیعی است که آن مؤسسه برای مسیریابی بسته‌ها در درون شبکه، از الگوریتم مورد نظر خود بهره بگیرد. به عنوان مثال شبکه‌های داخلی سه شرکت X و Y و Z، از منظر اینترنت در قالب سه AS دیده می‌شود. (البته به شرطی که به اینترنت متصل شده باشند.) از دیدگاه اینترنت، جزئیات درونی یک شبکهٔ AS قابل رویت نیست. علیرغم آنکه جزئیات درونی هر AS مستقل از دیگری است ولیکن داشتن یک استاندارد برای مسیریابی در درون یک AS پیاده‌سازی «مرز»[۲] ASها را آسان می‌کند. در این بخش مسیریابی در درون یک AS و در بخش بعدی مسیریابی بین ASها را مطالعه خواهیم کرد. مسیریابی در درون یک AS اصطلاحاً «پروتکل دروازهٔ درونی»[۳] و الگوریتم مسیریابی بین ASها «پروتکل دروازهٔ خارجی»[۴] نامیده می‌شود.

---

۱. در DHCP مسئول شبکه تعدادی آدرس IP یا محدوده‌ای از فضای آدرس IP مورد نظر خود را مشخص می‌کند تا این سرویس دهنده آنها را برحسب نیاز به ماشینهای شبکه اجاره بدهد. به این آدرسها اصطلاحاً IP Pool گفته می‌شود. م-

۲. منظور از «مرز» نقطه‌ای است که ASها به یکدیگر متصل می‌شوند و مسیریابهایی که این اتصال را برقرار می‌کنند مسیریابهای مرزی (Border Gateway) نامیده می‌شوند. این مسیریابها هم با درون AS و هم با دیگر ASها در ارتباط فعّال هستند؛ یعنی هم مسیرهای بهینهٔ داخلی و هم مسیرهای بهینهٔ بین ASها را می‌دانند. -م

۳. Interior Gateway Protocol ۴. Exterior Gateway Protocol

اولین پروتکل «دروازهٔ درونی اینترنت»، یک پروتکل بردار فاصله با نام RIP بود که از الگوریتم «بلمن‌فورد» بهره می‌گرفت. [بخش ۵-۲-۴] اگرچه این پروتکل در سیستمهای کوچک به خوبی کار می‌کند ولی با بزرگ شدن ASها، کارآیی خود را از دست می‌دهد. این پروتکل همچنین از مشکل «شمارش تا بی‌نهایت» رنج می‌برد و «همگرایی» کندی دارد. به همین دلیل در ماه می ۱۹۷۹ یک پروتکل مبتنی بر «حالت لینک» (Link State) جایگزین آن شد. در ۱۹۸۸، IETF کار را بر روی یک پروتکل جدید آغاز کرد. این پروتکل OSPF (Open Shortest Path First) نام گرفت و در سال ۱۹۹۰ استاندارد شد. امروزه اغلب تولیدکنندگان مسیریاب از آن حمایت می‌کنند و تقریباً به مهمترین پروتکل مسیریابی دروازه‌های درونی تبدیل شده است. در زیر شمائی از عملکرد OSPF ارائه می‌دهیم. برای آگاهی دقیقتر از کل قضیه به سند RFC 2328 مراجعه نمایید.

گروه طراح این پروتکل با داشتن تجربهٔ طولانی از دیگر پروتکلهای مسیریابی، تصمیم گرفتند با تدوین فهرست بالا بلندی از نیازها و انتظارات، آنرا از نو طراحی کنند. نخستین نیاز آن بود که به صورت «باز» (Open) طراحی شود بدین معنا که امتیاز راه‌حلها و روشهای خاصّ آن برای موسسهٔ خاصّی ثبت نشود و همچنین به بستر سخت‌افزاری یا نرم‌افزاری ویژه وابسته نباشد. حرف 'O' در نام OSPF به همین مضمون است.

نیاز دوم آن بود که پروتکل جدید بتواند برای تعیین مسیر بهینه، از معیارهای گوناگون هزینه مثل معیار «فاصلهٔ فیزیکی»، «تاخیر» و نظایر آن پشتیبانی کند. نیاز سوم آن بود که الگوریتم پویا باشد و هر گونه تغییر در توپولوژی زیرشبکه را به سرعت و به صورت خودکار تشخیص داده و خود را با آن تطبیق بدهد.

نیاز چهارم آنکه OSPF می‌بایست بسته‌ها را برحسب نوع خدمات درخواستی، مسیریابی و هدایت نماید. این پروتکل باید قادر می‌بود که ترافیک داده‌های بی‌درنگ را نسبت به ترافیک داده‌های معمولی به روش متفاوتی مسیریابی نماید. پروتکل IP در هر بسته یک فیلد به نام Type of Service تعریف کرده بود که هیچ پروتکل مسیریابی از آن حمایت نمی‌کرد. اگرچه این فیلد در OSPF گنجانده شد ولیکن باز هم کسی از آن استقبال نکرد و عاقبت حذف گردید.

نیاز پنجم (که با موارد فوق مرتبط است) آن بود که پروتکل جدید مکانیزم «**موازنهٔ بار**» (Load Balancing) را اعمال کند و بار را بر روی چندین خط تقسیم نماید. اغلب پروتکلها تمام بسته‌ها را بر روی بهترین مسیر ارسال می‌کنند و از مسیری که از لحاظ بهینگی در رتبهٔ دو قرار می‌گیرد هیچ استفاده‌ای نمی‌شود؛ در بسیاری از حالات تقسیم بار بر روی چندین خط، کارآیی بهتری دارد.

ششم آنکه نیاز بود از مسیریابی سلسله‌مراتبی پشتیبانی شود. تا سال ۱۹۸۸ اینترنت آنقدر بزرگ شده بود که نمی‌شد انتظار داشت یک مسیریاب بتواند توپولوژی کل آن را بداند. پروتکل جدید می‌بایست به گونه‌ای طراحی می‌شد که هیچ مسیریابی نیاز به دانستن توپولوژی کل شبکه نداشته باشد.

هفتم آنکه به سطحی از امنیت نیاز بود تا یک دانشجوی بازیگوش نتواند با ارسال اطلاعات جعلی و نادرست در خصوص مسیرها، مسیریاب را به اشتباه بیندازد. در آخر آنکه به تمهیداتی نیاز بود تا مسیریابهایی که با مکانیزم «ایجاد تونل» (Tunneling) و از طریق اینترنت بهم متصل می‌شوند را به درستی مدیریت نماید.

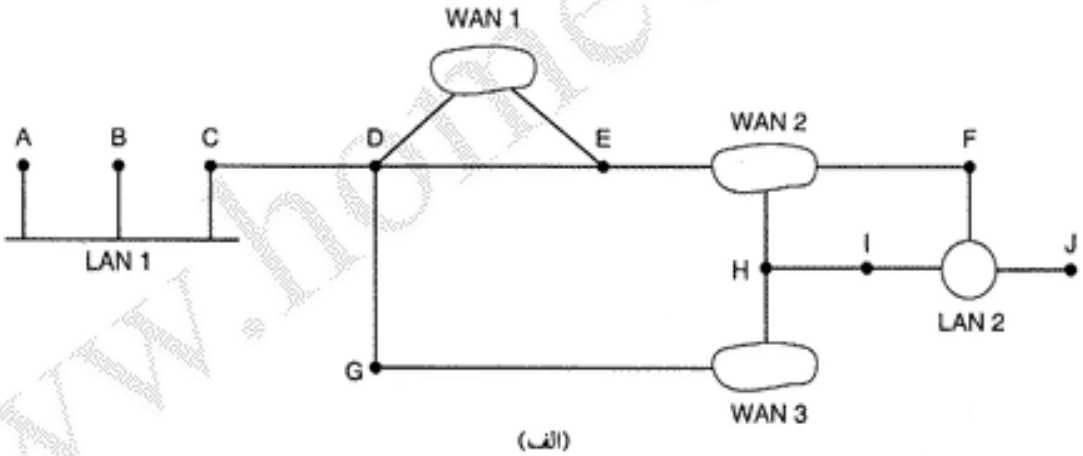
OSPF از سه نوع شبکه و خطوط اتصال پشتیبانی می‌کند:

۱. خطوط نقطه به نقطه بین دو مسیریاب
۲. شبکه‌های با دسترسی چندگانه که کانال آنها از نوع پخش فراگیر (Broadcost) است. (مثل اغلب شبکه‌های LAN)
۳. شبکه‌های با دسترسی چندگانه بدون آنکه کانال آنها از نوع پخش فراگیر باشد. (مثل اغلب شبکه‌های سوئیچ بسته در WAN)

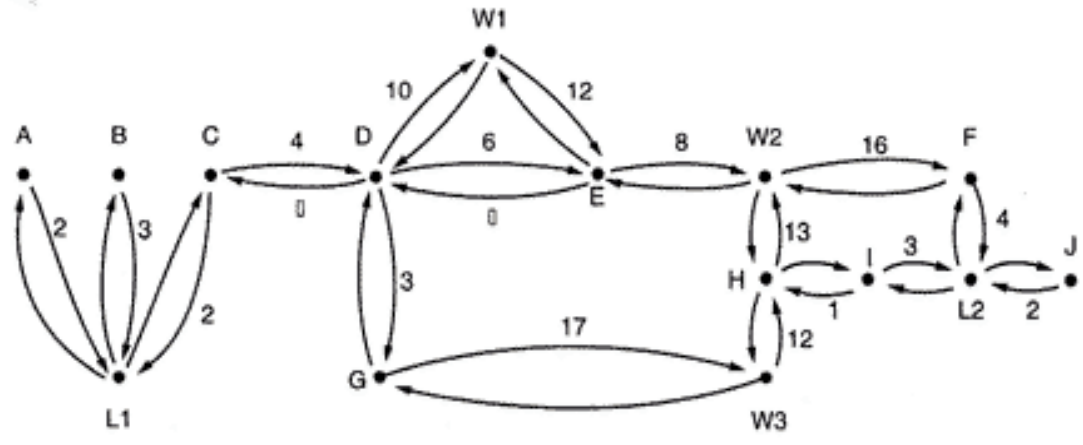
شبکهٔ با دسترسی چندگانه (Multiaccess Network)، شبکه‌ای است که می‌تواند چندین مسیریاب داشته باشد و هر یک از مسیریابها می‌توانند مستقیماً با یکدیگر در ارتباط باشند. تمام شبکه‌های LAN و WAN دارای این ویژگی هستند. شکل ۵-۶۴-الف یک AS را نشان می‌دهد که در آن هر سه نوع شبکه وجود دارد. دقت کنید که از دیدگاه OSPF، ماشینهای میزبان شبکه عموماً نقش خاصّی را ایفاء نمی‌کنند.

OSPF بدین نحو عمل می‌کند که مجموعهٔ شبکه‌ها، مسیریابها و خطوط ارتباطی را در قالب یک «گراف جهت‌دار» (Directed Graph) مدل می‌کند و به هر کمان در گراف (Arc) یک مقدار «هزینه» (مثل تأخیر، فاصله یا امثال آن) انتساب می‌دهد. سپس براساس وزن هر یک از کمانها، مسیر بهینه را پیدا می‌کند. یک خط ارتباطی سریال بین دو مسیریاب، در گراف با یک جفت کمان (Arc) نشان داده می‌شود (یک کمان به ازای هر جهت) و وزنهای هر کمان می‌تواند با دیگری متفاوت باشد. یک شبکه با دسترسی چندگانه (LAN)، با یک گره به ازای خود شبکه و یک گره به ازای هر مسیریاب مدل می‌شود. وزن کمانی که از گره شبکه به یک مسیریاب وارد می‌شود، صفر در نظر گرفته شده و از گراف حذف می‌گردد.

شکل ۵-۶۴-ب، نمایش گراف متناظر با شبکهٔ ۵-۶۴-الف را نشان می‌دهد. وزنها متقارن هستند مگر آنهایی که به صراحت مشخص شده‌اند. آنچه که OSPF انجام می‌دهد مدل کردن شبکه در قالب گرافی شبیه به این مثال و سپس محاسبهٔ مسیرهای بهینه از هر مسیریاب به هر مسیریاب دیگر است.



(الف)



(ب)

شکل ۵-۶۴. (الف) یک سیستم خودمختار (ب) نمایش گراف از شکل الف.

بسیاری از ASها در اینترنت خودشان بسیار عظیم هستند و مدیریت آنها ساده نیست. OSPF این امکان را فراهم آورده که بتوان چنین شبکه‌هایی را به تعدادی «**ناحیهٔ شماره‌گذاری شده**» (Numbered AREA) تقسیم کرد. هر ناحیه خود یک شبکه یا مجموعه‌ای از شبکه‌های بهم پیوستهٔ مجاور است. نواحی با یکدیگر همپوشانی ندارند[1] ولی لازم نیست که نواحی تعریف شده کل شبکهٔ AS را پوشش بدهد و ممکن است برخی از مسیریابها در هیچ ناحیه‌ای قرار نگیرند. یک ناحیه، شکل کلی و عمومی یک زیرشبکهٔ مستقل است و در خارج از ناحیه، توپولوژی و جزئیات درونی آن مشهود نیست.

در هر شبکهٔ خودمختار (AS) ناحیه‌ای به نام «**ستون فقرات**» وجود دارد که ناحیهٔ صفر نامیده می‌شود. تمام نواحی به ستون فقرات متصل می‌شوند (به صورت مستقیم یا توسط ایجاد تونل) لذا براحتی می‌توان به کمک ستون‌فقرات از هر ناحیه در شبکهٔ AS به ناحیهٔ دیگر رفت. یک «تونل» نیز در گراف توسط یک کمان مشخص می‌شود که دارای هزینه است. هر مسیریاب که به دو یا چند ناحیه متصل باشد (یعنی مسیریابهای مشترک بین دو یا چند ناحیه) جزیی از ستون فقرات شبکه محسوب می‌شود. همانند بقیهٔ نواحی، توپولوژی ناحیهٔ ستون فقرات نیز در خارج از آن مشهود نیست. (بعبارتی مسیریابهای درون دیگر نواحی از توپولوژی ستون‌فقرات چیزی نمی‌دانند.)

درون یک ناحیه، هر یک از مسیریابها نسخهٔ مشابهی از پایگاه اطلاعاتی در خصوص مسیرها و هزینه‌ها در اختیار دارند و الگوریتم محاسبهٔ کوتاهترین مسیر آنها یکسان است. هر مسیریاب وظیفه دارد که کوتاهترین مسیرها از خودش به تمام مسیریابهای دیگر ناحیه را محاسبه نماید. از جمله هر مسیریاب باید کوتاهترین مسیر از خود تا یک مسیریاب واقع بر روی ستون فقرات را پیدا کند. یک مسیریاب که در مرز دو ناحیه واقع است باید پایگاه اطلاعاتی هر دو ناحیه را در اختیار داشته باشد و الگوریتم کوتاهترین مسیر را بطور جداگانه بر روی آنها اجرا کند. [تا تمام مسیرهای بهینه از خودش تا بقیهٔ مسیریابها در هر دو ناحیه بدست بیاید.]

در حین عملیات طبیعی، احتمالاً به سه نوع مسیر نیاز است: (۱) مسیرهای درون ناحیه[2] (Intra-Area) (۲) مسیرهای بیرون ناحیه (Inter-Area) (۳) مسیرهای بین AS (Inter-AS).

مسیرهای درون ناحیه ساده‌ترین نوع مسیر هستند چراکه مسیریاب مبداء، از قبل و به دقت کوتاهترین مسیر رسیدن به مسیریاب مقصد را می‌داند. مسیریابی بین نواحی (Inter-Area)، همیشه در سه مرحله انجام می‌گیرد: حرکت از مبداء تا ستون فقرات، سپس حرکت از ستون فقرات به سوی ناحیهٔ مقصد و نهایتاً حرکت در درون ناحیه به سمت مقصد. این الگوریتم ایجاب می‌کند که در پروتکل OSPF یک توپولوژی «**ستاره**» برای مسیریابها قائل شویم: ستون فقرات در مرکز قرار می‌گیرد و بقیهٔ نواحی از آن منشعب می‌شوند. در OSPF بسته‌ها به همان نحوی که از مبداء تولید شده‌اند به سوی مقصد مسیریابی و هدایت می‌شوند یعنی بسته‌ها در درون بستهٔ دیگری «جاسازی» (Encapsulate) یا «تونل» (Tunnel) نمی‌شوند، مگر آنکه بسته مجبور باشد برای رسیدن از یک ناحیه به ستون‌فقرات، از مسیری حرکت کند که از طریق تونل ایجاد شده است.[3] شکل ۵-۶۵ بخشی از اینترنت را با چهار AS و تعدادی ناحیه نشان می‌دهد.

OSPF چهار کلاس مسیریاب را به رسمیت می‌شناسد:

۱. مسیریابهای درونی (که کاملاً در داخل یک ناحیه قرار می‌گیرند.)
۲. مسیریابهای واقع بر مرز دو ناحیه (که دو یا چند ناحیه را به هم متصل می‌کنند.)

---

۱. یعنی هر مسیریاب صرفاً به یک ناحیه متعلق است. -م

۲. یعنی مسیرهایی که از یک مسیریاب در درون ناحیه شروع و به یک مسیریاب در همان ناحیه ختم می‌شود. -م

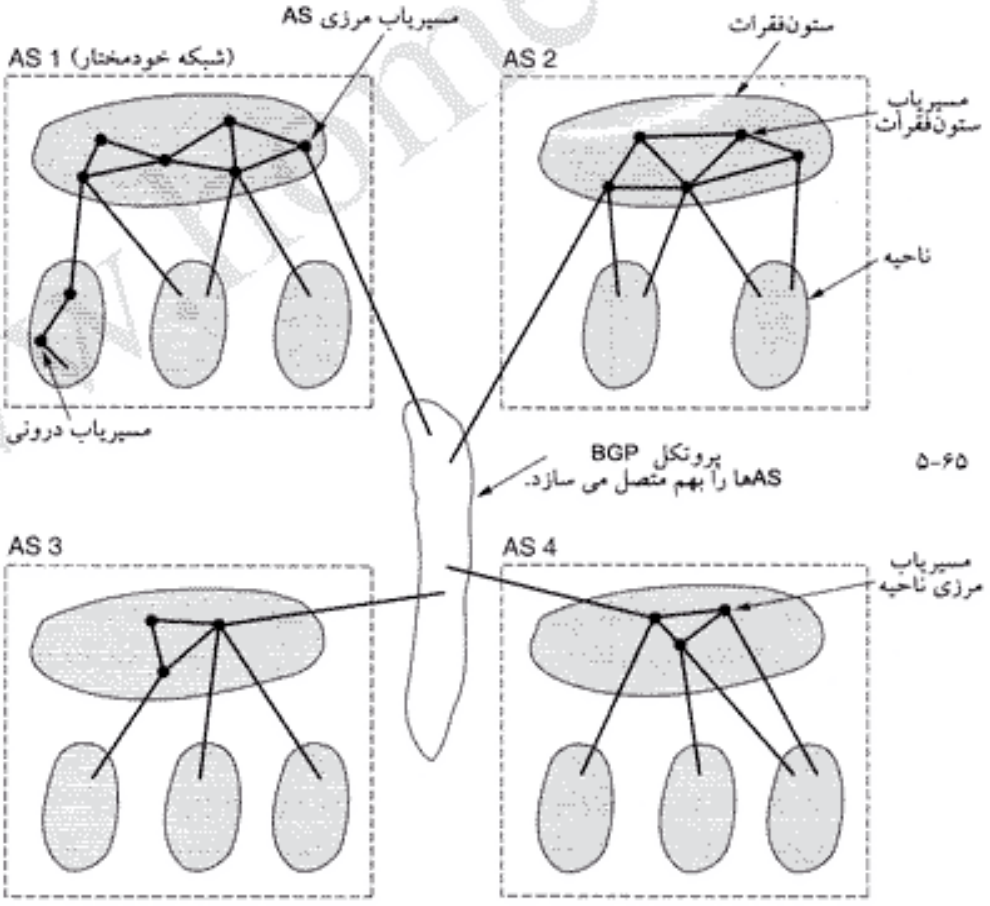
۳. مفهوم تونل را در بخش ۵-۵-۵ مطالعه نمائید.

۳. مسیریابهای ستون‌فقرات (که بر روی ستون‌فقرات قرار می‌گیرند.)
۴. مسیریابهای مرزی AS که می‌توانند با مسیریابهای AS دیگر محاوره کنند.

البته امکان آن وجود دارد که مسیریابها در دو یا چند کلاس قرار بگیرند. به عنوان مثال تمام مسیریابهای مرزی، به صورت خودکار جزیی از ستون فقرات نیز هستند. مضاف بر این، یک مسیریاب که بر روی ستون فقرات واقع است ولی در هیچ ناحیه‌ای قرار نگرفته، خودش یک مسیریاب داخلی محسوب می‌شود. در شکل ۵-۶۵ نمونه‌ای از تمام کلاسهای مسیریاب، دیده می‌شود.

وقتی یک مسیریاب بوت می‌شود، ابتدا بر روی تمام خطوط نقطه‌به‌نقطه و کانالهای اشتراکی (مثل کانالهای LAN که چند مسیریاب از طریق آن بهم متصل شده‌اند)، پیامی به نام «پیام سلام» (Hello Message) می‌فرستد. در خطوط WAN [مثل خطوط ISDN] احتمالاً قبل از هر گونه مبادلهٔ پیام به اطلاعات پیکربندی خاصّ نیاز است تا هویت تماس‌گیرنده مشخص شود.[1] هر مسیریاب با دریافت پاسخ سلام، مسیریابهای همسایهٔ خود را شناسایی می‌کند. مسیریابهایی که به یک LAN واحد متصلند، همگی همسایهٔ یکدیگر محسوب می‌شوند.

عملکرد OSPF مبتنی بر مبادلهٔ اطلاعات با «مسیریابهای مجاور» (Adjacent) است. مفهوم مسیریابهای مجاور با مفهوم «مسیریابهای همسایه» (Neighbor) فرق می‌کند؛ اینکه هر مسیریاب متصل به LAN بتواند با مسیریاب دیگر محاوره و مبادلهٔ پیام کند کافی و کارآمد نیست. برای اجتناب از این وضعیت، از بین کل مسیریابهای



شکل ۵-۶۵. ارتباط بین ASها، ستون‌فقرات و نواحی در OSPF.

۱. مثل تنظیمات خط و شمارهٔ PIN

واقع بر LAN ، یک مسیریاب به نام «**مسیریاب برگزیده**» (Designated Router) انتخاب می‌گردد. این مسیریاب ، «مسیریاب مجاور» بقیهٔ مسیریابهای واقع بر LAN محسوب می‌شود و با آنها به مبادلهٔ اطلاعات می‌پردازد. برای «مسیریاب برگزیده» یک مسیریاب پشتیبان نیز در نظر گرفته شده که آن نیز همیشه اطلاعات به‌روز از وضعیت مسیرها در اختیار دارد تا در صورت از کار افتادن «مسیریاب برگزیده»، به سرعت جایگزین آن شود.

در حین عملکرد طبیعی، هر مسیریاب بطور متناوب پیامهای LINK STATE UPDATE خود را به صورت سیل‌آسا(Flooding) برای تمام مسیریابهای مجاور خود می‌فرستد. [این پیامها حاوی اطلاعاتی در خصوص هویت همسایه‌ها ، لینکها و هزینهٔ آنهاست.] بدین ترتیب، با این پیامها حالت هر لینک (Link State) و هزینهٔ آن، برای درج در «**پایگاه اطلاعات توپولوژیکی**» به دست می‌آید. وصول پیامهایی که به صورت سیل‌آسا ارسال می‌شوند، تصدیق خواهد شد تا از دریافت آنها اطمینان حاصل شود.[1] هر پیام یک شمارهٔ ترتیب دارد تا مسیریاب بفهمد آیا پیام LINK STATE UPDATE قدیمی یا جدید است. ارسال این پیامها به صورت سیل‌آسا زمانی آغاز می‌شود که خطی از کار بیفتد یا مجدداً فعال شود یا هزینهٔ آنها تغییر کند. البته حتی اگر چنین اتفاقی نیفتد، این پیامها بطور متناوب و هر از چند ده ثانیه ارسال خواهند شد.

پیام DATABASE DESCRIPTION تمام شماره‌های ترتیب از درایه‌های جدول حالت لینک را که اخیراً در پایگاه اطلاعاتی فرستندهٔ آن ذخیره شده، اعلام می‌کند. هر یک از گیرندگان این پیام، شماره‌های اعلام شده را با شماره‌های ترتیب درایه‌های خودش مقایسه می‌کند تا بفهمد چه کسی جدیدترین مقادیر را در اختیار دارد. از این پیامها زمانی استفاده می‌شود که خطی فعال شود.

هر یک از طرفین یک لینک می‌توانند با استفاده از پیام LINK STATE REQUEST ، «اطلاعات حالت پیوند» یکدیگر را مطالبه نمایند. نتیجهٔ این الگوریتم آنست که هر جفت مسیریاب مجاور، آخرین اطلاعات به‌روز شدهٔ یکدیگر را بررسی کرده و بدین نحو اطلاعات جدید در کل ناحیه پخش می‌شود. تمام این پیامها به کمک بسته‌های معمولی IP ارسال می‌شود. فهرست پنج پیام فوق در جدول ۵-۶۶ خلاصه شده است.

نوع پیام	توصیف عملکرد
Hello	از این پیام برای شناسایی همسایه‌ها استفاده می‌شود.
Link state update	هزینه فرستنده پیام تا همسایه‌هایش را معین می‌کند.
Link state ack	دریافت بسته Link State Update را تایید می‌کند.
Database description	مسیریاب با این پیام فهرست درایه‌های بهنگام‌سازی خود را اعلام می‌کند.
Link state request	از شریک خود اطلاعاتی را درخواست می‌کند.

شکل ۵-۶۶. پنج نوع از پیامهای OSPF.

حال می‌توان مجموعهٔ عوامل فوق را بدین نحو جمع‌بندی کرد: به کمک روش ارسال سیل‌آسا، هر مسیریاب به تمام اعضاء ناحیهٔ خود، از همسایه‌هایش و هزینهٔ رسیدن به آنها خبر می‌دهد. این اطلاعات امکان آن را فراهم می‌آورد تا یکایک مسیریابها بتوانند گراف ناحیهٔ خود را تشکیل داده و کوتاهترین مسیرها را محاسبه نمایند. ستون فقرات نیز همین کار را می‌کند. [چراکه ستون فقرات خود یک ناحیهٔ مستقل است.] مضاف بر این، مسیریابهای واقع بر روی ستون فقرات، اطلاعات ارسالی از مسیریابهای مرزی هر ناحیه را هم می‌پذیرند تا بتوانند کوتاهترین مسیرها از ستون فقرات تا دیگر مسیریابها را محاسبه نمایند. این اطلاعات مجدداً به مسیریابهای مرزی هر ناحیه

---

۱. یعنی به ازای دریافت هر پیام از این نوع یک پیام ACK بر می‌گردد.

بازگردانده می‌شود تا آنها نیز در درون ناحیهٔ خودشان اعلام نمایند. به کمک این اطلاعات، یک مسیریاب که می‌خواهد بسته‌ای را به خارج از ناحیهٔ خود بفرستد، بهترین مسیریاب مرزی متصل به ستون فقرات را به عنوان دروازهٔ خروج بسته برمی‌گزیند.

۵-۶-۵ BGP[1]: پروتکل مسیریابی برای دروازهٔ خارجی

در درون یک AS واحد، پروتکل مسیریابی OSPF، بهترین گزینه است (اگرچه OSPF، تنها پروتکل مورد استفاده و رایج به حساب نمی‌آید). بین ASها از پروتکل متفاوتی به نام BGP استفاده می‌شود. به دلیل آنکه اهداف پروتکل مسیریابی درونی با پروتکل مسیریابی خارجی فرق می‌کند به پروتکل متفاوتی برای مسیریابی بین ASها نیاز است. پروتکل مسیریابی درونی باید به سریعترین وجه ممکن بسته‌ها را از مبداء به مقصد برساند و اهمیتی به سیاستهای جانبی نمی‌دهد.

پروتکلهای مسیریابی خارجی بایستی در حد وسیعی ملاحظات سیاسی را مدنظر قرار بدهند. (Metz, 2001) برای مثال ممکن است یک شبکهٔ AS بخواهد که بسته‌ها را به هر سایت اینترنت بفرستد یا بسته‌ها را از هر سایت اینترنت دریافت کند ولی تمایلی به حمل بسته‌هایی که از یک AS خارجی تولید شده و به یک AS خارجی دیگر می‌رود نداشته باشد، حتی اگر AS او بر روی کوتاهترین مسیر بین دو AS خارجی قرار گرفته باشد. (یعنی با این استدلال که: «مشکل آنها به ما ربطی ندارد!» بسته‌های دیگران را ترانزیت نکند.) از طرف دیگر ممکن است بخواهد در ازای دریافت هزینهٔ خدمات، ترافیک همسایه‌های خود و ASهای خاصّی را مسیریابی و هدایت کند. مثلاً شرکتهای مخابرات تلفن ممکن است مایل باشند به عنوان حامل مشتریان خود عمل کنند ولی نه برای دیگران (و بدون اخذ وجه)! پروتکلهای مسیریابی خارجی بطور عامّ و پروتکل BGP بطور خاصّ این امکان را فراهم کرده‌اند که بتوان سیاستهای گوناگونی را بر روی ترافیک بین ASها اعمال کرد.

سیاستهای کلی حول مسائل سیاسی، امنیتی یا ملاحظات اقتصادی دور می‌زنند. چند مثال از ملاحظات مسیریابی عبارتند:

۱. عدم اجازهٔ عبور ترافیک داده‌ها از میان ASهای خاصّ
۲. مسیری که از پنتاگون شروع می‌شود هرگز از عراق عبور نکند!
۳. مسیر رسیدن از بریتیش‌کلمبیا به اونتاریو از ایالات متحده نگذرد!
۴. فقط زمانی از مسیر آلبانی عبور شود که هیچ راه دیگری به مقصد وجود نداشته باشد!
۵. ترافیک داده‌هایی که از IBM شروع شده یا بدان ختم می‌شود از مایکروسافت عبور نکند!!

عموماً سیاستهای مسیریابی، به صورت دستی بر روی مسیریابهای BGP تنظیم و پیکربندی می‌شود (یا در قالب نوعی «اسکریپت» بر روی مسیریاب اجرا می‌گردد). البته این تنظیمات جزیی از پروتکل بحساب نمی‌آید.

از دیدگاه یک مسیریاب BGP، کل جهان از چندین AS و خطوط ارتباطی بین آنها تشکیل شده است. دو AS وقتی متصل تلقی می‌شوند که بین مسیریابهای مرزی هر یک از آنها حداقل یک خط وجود داشته باشد. با توجه به تأکید BGP بر حمل (یا بعبارتی ترانزیت) ترافیک، هر شبکهٔ AS در یکی از سه ردهٔ ذیل قرار می‌گیرد: (۱) رده اول «**شبکه‌های پایانی**» (Stub Network) نامیده می‌شوند؛ این گونه از شبکه‌ها فقط و فقط یک اتصال با گراف BGP دارند و نمی‌توانند ترافیک داده‌ها را از خود عبور بدهند چراکه در طرف دیگر آنها شبکه‌ای وجود ندارد.[2] (۲) ردهٔ بعدی، «**شبکه‌های چنداتصالی**» (Multiconnected Networks) هستند. این شبکه‌ها می‌توانند حمل ترافیک

---

۱. Border Gateway Protocol

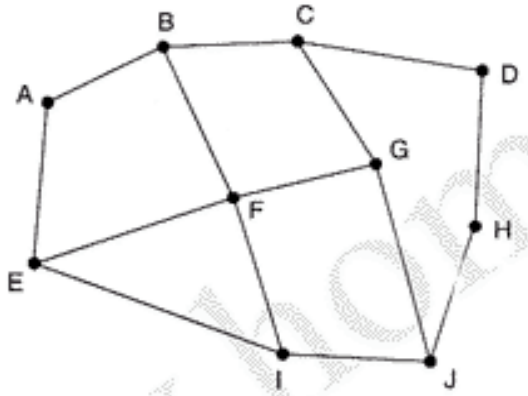
۲. شبکه‌های پایانی را به مثابهٔ یک کوچهٔ بن‌بست فرض کنید. -م

بسته‌های دیگران را بر عهده بگیرند مگر آنکه عمداً از این کار امتناع کنند. (۳) در ردهٔ آخر، «**شبکه‌های ترانزیت**» (Transit Network) قرار می‌گیرند که در نقش ستون فقرات، تمایل دارند بسته‌های شبکهٔ دیگران را (طبق برخی از محدودیتها و احتمالاً دریافت هزینه) حمل (ترانزیت) کنند.

یک جفت مسیریاب BGP از طریق برقراری یک اتصال TCP ، با یکدیگر مبادلهٔ اطلاعات می‌نمایند. این مکانیزم، امکان مبادلهٔ مطمئن و بدون خطای داده‌ها را فراهم کرده و جزئیات درونی شبکه را پنهان نگاه می‌دارد.

پروتکل BGP ذاتاً مبتنی بر الگوریتم بردار فاصله است ولی از بسیاری جهات با RIP فرق دارد؛ تفاوت بنیانی در اینجاست که به جای آنکه فقط هزینه و خط رسیدن به یک مقصد در شبکه محاسبه و نگهداری شود، مسیریاب BGP کل مسیر رسیدن به هر مقصد را نگه می‌دارد. همچنین به جای آنکه به همسایه‌های خود در خصوص هزینهٔ رسیدن به هر مقصد در شبکه ، خبر بدهد کلّ مسیرهای واقعی را اعلام می‌کند.

به عنوان مثال مسیریابهای BGP در شکل ۵-۶۷-الف و به ویژه جدول مسیریابی F را در نظر بگیرید. فرض کنید که این مسیریاب برای رسیدن به D از مسیر FGCD استفاده می‌کند. وقتی همسایه‌ها اطلاعات مسیریابی خود را به او می‌دهند مسیرهای کامل را مشابه فهرست ۵-۶۷-ب اعلام می‌نمایند. (برای سادگی فقط بخشی از مسیرها که به D ختم می‌شوند نشان داده شده است.)



(الف)

اطلاعاتی که F از همسایه‌های خود در خصوص D دریافت می‌کند.

B از : "I use BCD"
G از : "I use GCD"
I از : "I use IFGCD"
E از : "I use EFGCD"

(ب)

شکل ۵-۶۷. (الف) مجموعه‌ای از مسیریابهای BGP. (ب) اطلاعات ارسالی برای F

پس از آنکه F تمام مسیرها را از همسایه‌های خود دریافت کرد، آنها را بررسی می‌کند تا ببیند کدامیک از آنها بهترین است. مثلاً F برای تعیین بهترین مسیر رسیدن به D ، فوراً مسیرهای اعلام شده توسط I و E را حذف می‌کند چراکه این مسیرها خودشان از F می‌گذرند. انتخاب نهایی از بین مسیرهای اعلام شدهٔ B و G است. هر مسیریاب BGP دارای یک ماجول خاص است که مسیرهای رسیدن به یک مقصد خاص را بررسی کرده و به آنها «نمره» می‌دهد و در نهایت برای هر مسیر عددی را به عنوان «معیار فاصله» (Distance) بر می‌گرداند. هر مسیری که ملاحظات و محدودیتهای اتخاذ شده را نقض کرده باشد، نمرهٔ بی‌نهایت می‌گیرد. پس از این مرحله، مسیریاب مسیری با کمترین هزینه را می‌پذیرد. تابع نمره‌دهی[1] به مسیرها بخشی از پروتکل BGP بحساب نمی‌آید و می‌تواند به دلخواه مدیر سیستم انتخاب و تعریف شود.

پروتکل BGP به سادگی مشکل «شمارش تا بی‌نهایت» (که الگوریتمهای بردار فاصله را آزار می‌دهد) حل کرده است.[2] به عنوان مثال فرض کنید G از کار بیفتد یا خط FG قطع شود. لذا F فقط از سه همسایهٔ باقیماندهٔ خود (یعنی B و I و E) اطلاعات مسیر می‌گیرد. مسیرهائی که همسایه‌های او برای رسیدن به D اعلام می‌کنند عبارتند از :

---

۱. Scoring Function ۲. زیرا مسیرهای کامل به همسایه‌ها اعلام می‌شود. بخش ۵-۲-۴ را ببینید. -م

BCD ، IFGCD و EFGCD . مسیریاب F بلافاصله متوجه می‌شود که مسیرهای دوم و سوم بی‌فایده هستند چراکه از خود F می‌گذرند، لذا FBCD به عنوان مسیر جدید انتخاب می‌شود. الگوریتمهای دیگر «بردار فاصله» بدان دلیل در انتخاب مسیر به اشتباه می‌افتند که همسایه‌ها نمی‌توانند اعلام کنند کدامیک از مسیرهای رسیدن به مقصد، مسیر مستقلی است. (یعنی از خود همسایه نمی‌گذرد). پروتکل BGP در اسناد RFC 1771 تا RFC 1774 تشریح شده است.

۵-۶-۶ ارسال چندپخشی در اینترنت (Internet Multicasting)

عملکرد طبیعی پروتکل IP ، مبادلهٔ بسته‌های داده بین **یک** گیرنده و **یک** فرستنده است؛ با این وجود در برخی از کاربردها این قابلیت که یک پروسه بتواند بطور همزمان برای تعداد بسیار زیادی گیرنده، ارسال داشته باشد، کارساز و مفید است. مثالهایی از این قبیل عبارتست از: به هنگام‌سازی همزمان نسخه‌های توزیع شده یا تکراری پایگاههای اطلاعاتی، اعلام همزمان قیمت سهام به متولیان و واسطه‌ها، مدیریت و اجرای کنفرانسهای دیجیتال و گردهم‌آیی تلفنی.

IP با استفاده از آدرسهای کلاس D از فرآیند چندپخشی (Multicasting) پشتیبانی می‌نماید. هر آدرس کلاس D ، هویت یک «گروه» از ماشینها را مشخص می‌کند. برای مشخص کردن هر گروه در کلاس D ، ۲۸ بیت در اختیار است، لذا بطور همزمان می‌توان بیش از ۲۵۰ میلیون گروه تعریف کرد. هر گاه یک پروسه، بسته‌ای را به یک آدرس کلاس D ارسال کند، برای رساندن آن بسته به یکایک اعضای گروه، مسیریابها حداکثر تلاش ممکن را بعمل می‌آورند ولیکن هیچ تضمینی داده نمی‌شود و احتمال دارد برخی از اعضا بسته را دریافت نکنند!

در IP از دو نوع آدرس گروهی حمایت می‌شود: **آدرسهای دائم** (Permanent) و **آدرسهای موقت** (Temporary). «گروه دائم» همیشه وجود دارد و نیازی به تنظیم و هماهنگی قبلی نیست. هر گروه دائم دارای آدرس ثابت و بلاتغییر است. چند نمونه از آدرس گروههای ثابت عبارتند از:

گروه 224.0.0.1 : تمام سیستمهای متصل به یک LAN

گروه 224.0.0.2 : تمام مسیریابهای متصل به یک LAN

گروه 224.0.0.5 : تمام مسیریابهای OSPF متصل به یک LAN

گروه 224.0.0.6 : تمام مسیریابهای «برگزیدهٔ OSPF» متصل به یک LAN

گروههای موقتی باید قبل از استفاده، ایجاد شوند. یک پروسه می‌تواند از ماشین خود درخواست کند که به یک گروه خاص بپیوندد؛ همچنین می‌تواند از آن بخواهد که گروه را ترک کند. وقتی که آخرین پروسه در یک ماشین، گروهی را ترک کند، ماشین مربوطه از آن گروه خارج خواهد شد. هر ماشین می‌داند که پروسه‌های او عضو چه گروههایی هستند.

ارسال چندپخشی توسط مسیریابهای خاصی پیاده‌سازی و پشتیبانی می‌شود و طبعاً ممکن است در کنار مسیریابهای استاندارد، اینگونه مسیریابها وجود نداشته باشد، لذا توانایی ارسال چندپخشی به نوع مسیریابهای شبکه و پیکربندی آنها برمی‌گردد. مسیریابهای چندپخشی تقریباً در هر دقیقه یکبار، یک پیام سخت‌افزاری (یعنی یک فریم لایهٔ پیوند داده‌ها) را به صورت چندپخشی برای تمام ماشینهای متصل به LAN خودش (یعنی به آدرس 224.0.0.1) می‌فرستند و از آنها می‌خواهد که اعلام کنند پروسه‌هایشان در چه گروههایی عضو هستند. در پاسخ به این سؤال، یکایک ماشینها، آدرسهای کلاس D گروههای مورد نظر خود را بر می‌گردانند.

این بسته‌های پرسش و پاسخ مبتنی بر پروتکلی به نام IGMP[1] است که تا حدودی به ICMP شبیه است. در

---

۱. Internet Group Management Protocol

این پروتکل فقط دو نوع بسته تعریف شده است: بستهٔ پرسش و بستهٔ پاسخ. هر یک از این بسته‌ها دارای قالبی ساده و ثابت هستند: در اولین کلمه از فیلد داده (PayLoad) برخی اطلاعات کنترلی قرار می‌گیرد و در کلمهٔ بعدی یک آدرس کلاس D درج می‌شود. [کلمات چهار بایتی هستند.] این پروتکل در سند RFC 1112 تشریح شده است.

مسیریابی چندپخشی در اینترنت به کمک «درختهای پوشا» (Spanning Tree) انجام می‌شود. هر مسیریاب که از فرآیند چندپخشی حمایت می‌کند، به کمک «پروتکل اصلاح شدهٔ بردار فاصله» با همسایه‌های خود اطلاعاتی را مبادله می‌کند تا هر کدام بتوانند به ازای هر گروه یک «درخت پوشا» تشکیل بدهند. (به نحوی که تمام اعضای هر گروه را در بر بگیرد.) به منظور آنکه درخت به نحوی سازماندهی و پیرایش شود تا مسیریابها یا شبکه‌های غیر عضو در گروه از ساختار درخت حذف گردند، بهینه‌سازیهایی صورت گرفته است. این پروتکل از مکانیزم «ایجاد تونل» استفادهٔ زیادی می‌کند تا بدین نحو برای گره‌هایی که در درخت پوشا قرار ندارند سردرگمی و اشکال ایجاد نشود.

۷-۶-۵ IP متحرک (Mobile IP)

بسیاری از کاربران اینترنت از کامپیوترهای کیفی قابل حمل استفاده می‌کنند و طبعاً علاقمندند وقتی به یک محل جدید نقل مکان می‌نمایند یا حتی در طول راه، اتصالشان با اینترنت برقرار بماند. متأسفانه، سیستم آدرس‌دهی IP به گونه‌ای طراحی شده که کار کردن با آن دور از خانه، فقط در گفتار ساده است تا در عمل! در این بخش این مشکل و راه حل آن را بررسی می‌کنیم. (Perkins, 1998a)

نقطه ضعف واقعی IP، در روش آدرس‌دهی آن نهفته است. یک آدرس IP متشکل از یک شمارهٔ شبکه و یک شمارهٔ ماشین است. به عنوان مثال ماشینی با آدرس 160.80.40.20/16 را در نظر بگیرید. شمارهٔ شبکه 160.80 (یا 8272 در مبنای ده) و شمارهٔ ماشین 40.20 (یا 10260 در مبنای ده) است. مسیریابهای کل جهان یک جدول مسیریابی دارند که در آن خط مناسب برای رسیدن به شبکهٔ 160.80 مشخص شده است. وقتی بسته‌ای با آدرس IP بشکل 160.80.xxx.yyy، به مسیریاب وارد گردد بر روی آن خط ارسال خواهد شد.

بدین نحو اگر ماشینی با آدرس فوق به محل جدیدی نقل مکان کند، کماکان بسته‌های ارسالی برای او به شبکه یا مسیریاب خانگیش هدایت شده و از آن به بعد صاحب ماشین قادر به دریافت نامه‌های الکترونیکی یا نظائر آن نخواهد بود. انتساب آدرس IP جدید (متناظر با محل جدید) به ماشین سیّار، راهکار جالبی نیست چراکه این تغییر باید به تعداد زیادی از افراد، بانکهای اطلاعاتی و برنامه‌های کاربردی اطلاع داده شود.

راهکار دیگر آنست که مسیریابها در جدول مسیریابی به جای آنکه فقط شمارهٔ شبکه را نگه دارند آدرس کامل ماشینها را درج نمایند ولیکن این روش نیز مستلزم ذخیرهٔ میلیونها درایه (Entry) در جدول مسیریابی است و هزینه‌های نجومی به اینترنت تحمیل می‌کند.

از زمانی که تقاضا برای این قابلیت (که افراد بتوانند در هر کجا که هستند به اینترنت متصل شوند)، بالا گرفت، IETF یک گروه ویژه برای پیدا کردن راه حل مناسب تشکیل داد. این گروه کاری به سرعت اهدافی را که هر پیشنهاد یا راه حل باید آنها را برآورده می‌ساخت، تدوین و تعریف کردند. مهمترین این اهداف عبارت بود از:

۱. هر ماشین متحرک باید بتواند هر جا که هست از آدرس IP همیشگی و خانگی خود استفاده کند.
۲. هر گونه تغییر نرم‌افزاری در ماشینها مجاز شمرده نمی‌شود.
۳. هر گونه تغییر در نرم‌افزار مسیریابها یا جداول آنها مجاز نیست.
۴. اکثر بسته‌هایی که به سوی ماشینهای متحرک هدایت می‌شوند نباید از مسیر واقعی منحرف شوند.

۵. وقتی ماشین میزبان در محل همیشگی خود است نباید هیچ سربار اضافه‌ای تحمل کند.[۱]

راه حل انتخابی همانی بود که در بخش ۵-۲-۸ آن را تشریح کردیم. در یک مرور اجمالی: هر سایتی که تمایل داشته باشد امکان سیار بودن را برای کاربران خود فراهم کند موظف به ایجاد یک «عامل خانگی» (Home Agent) است. همچنین هر سایتی که تمایل به پذیرش کاربران خارجی (کاربران سیار) داشته باشد موظف به ایجاد یک «عامل خارجی» (Foreign Agent) است. هر گاه یک ماشین متحرک در محدودهٔ یک سایت خارجی ظاهر می‌شود، ابتدا با عامل خارجی تماس برقرار کرده و ثبت نام می‌کند. عامل خارجی با عامل خانگی آن کاربر تماس گرفته و آدرسی جدید از محل کاربر متحرک به آن اعلام می‌کند. این آدرس عموماً آدرس IP خود عامل خارجی است.

وقتی بسته‌ای به LAN محل استقرار دائمی کاربر می‌رسد قبل از انتشار بر روی LAN به مسیریاب متصل به آن LAN وارد می‌شود. آن مسیریاب به روش معمول و همیشگی سعی می‌کند ماشین آن کاربر را پیدا کند لذا یک بستهٔ ARP منتشر کرده و مثلاً می‌پرسد: «آدرس اترنت ماشین 160.80.40.20 چند است؟». «عامل خانگی» به نیابت از کاربر، به این سؤال پاسخ داده و آدرس اترنت خود را اعلام می‌کند. از آن به بعد، مسیریاب تمام بسته‌های متعلق به 160.80.40.20 را برای «عامل خانگی» می‌فرستد. «عامل خانگی»، با مکانیزم ایجاد تونل بستهٔ اصلی را درون یک بستهٔ IP جدید جاسازی کرده و آدرس مقصد بستهٔ جدید را آدرس عامل خارجی قرار داده و آنرا ارسال می‌نماید. عامل خارجی پس از دریافت ، بستهٔ اصلی را از درون آن استخراج کرده و آنرا به آدرس MAC ماشین متحرک (یعنی آدرس سخت‌افزاری تعریف شده در لایهٔ پیوند داده‌ها) ارسال می‌دارد. مضاف بر این، عامل خانگی کاربر، آدرس IP عامل خارجی را (که کاربر فعلاً مهمان اوست) به فرستندهٔ بسته‌ها اعلام می‌کند تا بسته‌های بعدی به کمک مکانیزم تونل مستقیماً به آدرس عامل خارجی ارسال شوند. این راه حل تمام نیازها و اهداف فوق‌الذکر را برآورده می‌کند.

اشاره به برخی از جزئیات ارزشمند است: در لحظه‌ای که ماشین متحرک، محل خود را ترک می‌کند احتمالاً مسیریاب در جدول نهان ARP[۲] آدرس اترنت این ماشین را (که دیگر معتبر نیست) درج کرده و از آن استفاده می‌کند.[۳] برای جایگزین کردن آدرس اترنت «ماشین عامل» به جای آدرس ماشین متحرک از راهکاری زیرکانه به نام «Gratuitous ARP» استفاده می‌شود. این روش مبتنی بر ارسال یک پیام خاص و ناخواسته برای مسیریاب است که باعث می‌شود یک درایهٔ دلخواه در جدول ARP به مقداری جدید تغییر کند. در اینجا درایه‌ای که باید تغییر کند متعلق به ماشین متحرک و در حال خروج از شبکه است. وقتی بعداً ماشین متحرک به شبکهٔ خود باز می‌گردد از همین راهکار برای بهنگام‌سازی مجدد جدول ARP در مسیریاب، استفاده می‌شود.[۴]

هیچ مانعی وجود ندارد که یک ماشین متحرک نتواند «عامل خارجی» خودش باشد ولی این راهکار زمانی عملی است که ماشین متحرک در موقعیت جدید خود ، به اینترنت دسترسی داشته باشد. در ضمن ماشین متحرک باید بتواند در محل جدید یک آدرس IP موقت جهت اعلام به «عامل خانگی» خودش بدست بیاورد. این آدرس

---

۱. یعنی اگر سربار کوچکی تحمیل می‌شود باید فقط زمانی باشد که ماشین میزبان در محل همیشگی خود نیست. -م

۲. ARP Cache

۳. یعنی چون پس از خروج ماشین متحرک، هنوز آدرس MAC آن در جدول ARP مسیریاب، وجود دارد تا موقعی که این آدرس اعتبار خود را حفظ می‌کند بسته‌ها به آدرسی که دیگر وجود ندارد ارسال خواهد شد. -م

۴. این راهکار آنست که «ماشین عامل» بدون آنکه هیچ سؤالی از او شده باشد یک بستهٔ پاسخ ARP (یعنی ARP Response) به صورت مصنوعی تولید و ارسال می‌کند. بدین ترتیب مسیریاب به اشتباه افتاده و محتوای آنرا در جدول خود درج می‌کند، هر چند هرگز در این خصوص سؤالی نپرسیده بود! -م

IP متعلق به LAN جدیدی است که ماشین متحرک موقتاً بدان متصل شده و مهمان آنست.

راه حل پیشنهادی IETF برای ماشینهای متحرک، مسائل متعدد دیگری را هم حل کرد که تا آن زمان بدان توجه نشده بود. به عنوان مثال یکی از مسائل این بود که ماشینهای عامل چگونه پیدا شوند؟ راه حل این مسئله آنست که هر عامل بطور متناوب آدرس خود و نوع سرویسهایی را که علاقمند به ارائه آنهاست، به صورت فراگیر منتشر نماید. وقتی ماشین متحرک به جایی وارد می‌شود ابتدا سعی می‌کند به این پیامهای انتشاری که اصطلاحاً «**اعلان**» (Advertisement) نامیده می‌شود، گوش فرا بدهد. گزینهٔ دیگر آنست که ماشین متحرک ورود خود را با انتشار بسته‌ای فراگیر (Broadcast) اعلام کرده و در انتظار پاسخ ماشین «عامل خارجی» باقی بماند.

مشکل دیگری که باید حل شود آنست که با ماشینهای متحرکی که با عجله و بدون خداحافظی شبکهٔ فعلی خود را ترک می‌کنند، چه باید کرد؟ راه حل آنست که حضور یک ماشین متحرک فقط برای مدت زمان ثابت و محدودی اعتبار داشته باشد. اگر ماشین متحرک بطور مرتب، اعتبار حضور خود را تمدید نکند و مهلت اعتبار او منقضی شود، عامل خارجی جداول خود را از مشخصات او پاک می‌نماید.

مشکل دیگر، موضوع امنیت است: وقتی یک عامل خانگی پیامی دریافت می‌کند که از او خواسته شده تمام بسته‌های متعلق به کاربری به نام روبرتا را به آدرس IP خاصی بفرستد، بهتر است این تقاضا پذیرفته نشود مگر آنکه اثبات شود که مبداء این درخواست، واقعاً روبرتا است نه کسی که خودش را به جای او جا زده است. بدین منظور از پروتکلهای احراز هویت مبتنی بر رمزنگاری استفاده می‌شود. این پروتکلها را در فصل هشتم تشریح خواهیم کرد.

آخرین نکته‌ای که گروه کاری IETF بدان پرداخت در خصوص «**سطوح تحرّک**» (Levels of Mobility) ماشینهاست. هواپیمایی را در نظر بگیرید که برای وصل کامپیوترهای مخصوص ناوبری و کنترل پرواز شبکه‌ای از نوع اترنت را بکار گرفته است. بر روی این شبکه یک مسیریاب استاندارد وجود دارد که از طریق یک لینک رادیویی با شبکهٔ اینترنت مستقر بر روی زمین در ارتباط است. از طرفی این ایده در ذهن برخی از افراد زیرک جرقه زده که در کنار دستهٔ صندلی هر مسافر یک کانکتور اترنت تعبیه شود تا مسافرین بتوانند در خلال پرواز، کامپیوترهای کیفی خود را به آن متصل کرده و به اینترنت وصل شوند. در این حالت باید برای کامپیوترهای متحرک دو سطح قائل شد: کامپیوترهای خود هواپیما، که نسبت به اترنت موجود در آن ثابت (Stationary) محسوب می‌شوند و کامپیوتر مسافران که نسبت به آن متحرک هستند. البته مسیریاب هواپیما نسبت به مسیریاب مستقر بر روی زمین، متحرک است. می‌توان متحرک بودن کامپیوترها نسبت به مسیریابهایی که خودشان متحرک هستند را از طریق ایجاد تونلهای متوالی پیاده‌سازی و اداره کرد.

IPv6 ۸-۶-۵

اگرچه ممکن است مکانیزمهای CIDR و NAT چند سالی دیگر به دوام نسخهٔ چهارم IP کمک کنند ولی تقریباً بر همه آشکار شده که نفسهای پروتکل IP در شکل کنونی آن، به شماره افتاده است. مضاف بر مشکلات فنی IP، برخی از موارد پشت صحنه و زمینه‌ای دیگر نیز مطرح است. در سالهای اولیه، از اینترنت عموماً در دانشگاهها، صنایع پیشرفته و دولت ایالات متحده (خصوصاً وزارت دفاع) استفاده می‌شد. با گرایش بسیار زیاد مردم به اینترنت که از اواسط دههٔ نود شروع شد، گروههای مختلفی از افراد به آن رو آوردند؛ افرادی که نیازها و انتظارات متفاوتی داشتند. یکی از موارد آنست که افراد با کامپیوترهای بی‌سیم قابل حمل برای در ارتباط بودن با محل استقرار دائمی خود (ایستگاههای خانگی) می‌خواهند از اینترنت بهره بگیرند. مورد دیگر آن که با همگرایی قریب‌الوقوع صنایع کامپیوتر و مخابرات و صنایع تولید بازی و ابزار تفریح، دیری نخواهد پایید که حتی دستگاههای تلفن و تلویزیون در دنیا، به عنوان گرهی از اینترنت، به آن خواهد پیوست و در آن زمان میلیاردها

ماشین، از صدا و تصویر بهره خواهند گرفت. با در نظر داشتن چنین چشم‌اندازی ، IP بوضوح نیازمند تغییرات اساسی است و باید انعطاف بیشتری داشته باشد.

IETF که چنین افقی را پیش روی خود می‌دید در اوان ۱۹۹۰ کار را بر روی نسخهٔ جدیدی از پروتکل IP شروع کرد که در آن فضای آدرس هرگز با کمبود مواجه نشود و مشکلات عدیده‌ای را حل کند؛ قابلیت انعطاف بیشتری داشته باشد و در ضمن کارآمدتر باشد. اهداف عمدهٔ IPv6 عبارت بودند از:

۱. پشتیبانی از میلیاردها ماشین میزبان حتی در صورتی که تخصیص فضای آدرس ناکارآمد و با اسراف انجام شود.
۲. کاهش اندازهٔ جداول مسیریابی
۳. ساده‌سازی پروتکل به منظور افزایش سرعت پردازش مسیریابها
۴. ارائه امنیت بهتر در مقایسه با نسخهٔ فعلی IP (شامل احراز هویت و سرّی ماندن داده‌ها)
۵. توجه بیشتر به نوع خدمات و QoS ، به ویژه برای داده‌های بی‌درنگ
۶. کمک به فرآیند ارسال چندپخشی از طریق توصیف حوزه‌ها (Scopes)
۷. فراهم آوردن امکان جابجایی ماشینهای میزبان بدون تغییر در آدرس
۸. امکان ایجاد تغییر و پیشرفت در آینده
۹. امکان همزیستی پروتکلهای جدید و قدیم در طی سالها

برای توسعهٔ پروتکلی که تمام نیازهای فوق‌الذکر را برآورده نماید، IETF با انتشار RFC 1550 و در طی یک فراخوان، خواستار پیشنهادات دیگران در این خصوص شد. ۲۱ پیشنهاد دریافت گردید که اغلب آنها جامع نبودند. تا دسامبر ۱۹۹۲ فقط هفت طرح پیشنهادی قابل توجه در دستور کار قرار داشت. این طرحهای پیشنهادی از اصلاحات جزیی در نسخهٔ فعلی IP تا پیشنهاد دور انداختن آن و جایگزینی با یک پروتکل کاملاً متفاوت را شامل می‌شد.

یک پیشنهاد آن بود که TCP بر روی CLNP اجرا شود؛ پروتکلی که با آدرسهای ۱۶۰ بیتی فضای آدرس‌دهی نامحدود و جاویدان را فراهم کرده بود و دو پروتکل عمده و مهم لایهٔ شبکه را متحد و یکنواخت می‌کرد. ولیکن بسیاری افراد احساس کردند که پذیرش آن مهر تأییدی است بر این ادعا که هر کاری که OSI انجام داده صحیح تلقی می‌شود، داعیه‌ای که لااقل در حوزهٔ اینترنت به دلایل خاص نادرست است. الگوی CLNP بسیار شبیه به IP بود و این دو، تفاوت چندانی با هم ندارند. در آخر نیز طرحی انتخاب شد که تفاوت بسیار زیادی با IP و از آن بیشتر با CLNP دارد. ضربهٔ دیگری که CLNP خورد از آنجا بود که پشتیبانی ضعیفی از «نوع خدمات» (Type of Service) می‌کرد، خصوصیتی که برای انتقال کارآمد داده‌های چند رسانه‌ای به شدت نیاز بود.

سه تا از بهترین طرحهای پیشنهادی در ژورنال IEEE Network منتشر شد.[1] پس از مباحثات فراوان، بازبینی ، ارزیابی موقعیت و سنجش استقبال عمومی، نسخه‌ای ترکیبی از طرحهای پیشنهادی Deering و Francis ک SIPP[2] نامیده می‌شد به عنوان طرح برگزیده معرفی و با عنوان IPv6 معرفی گردید.

IPv6 بخوبی اهداف مورد نظر را برآورده می‌کند: ویژگیهای خوب IP را نگه داشته، ویژگیهای بد را کنار گذاشته یا کمرنگ کرده و ویژگیهای جدیدی به آن افزوده است. بطور کلی IPv6 با IPv4 سازگار نیست ولی با تمام پروتکلهای جانبی اینترنت مثل TCP ، UDP ، ICMP ، IGMP ، OSPF ، BGP و DNS سازگار است. (البته

۱. Deering, 1993; Francis, 1993; and Katz and Ford, 1993

۲. Simple Internet Protocol Plus

ممکن است به دلیل آنکه آدرسها طولانی‌تر شده‌اند نیاز به اندکی تغییر داشته باشند.) ویژگیهای اساسی IPv6 در زیر تشریح شده است. برای آگاهی بیشتر در خصوص آن به RFCهای 2460 تا 2466 مراجعه نمایید.

اولین و مهمترین ویژگی آنست که IPv6 آدرسهای بسیار طولانی‌تری نسبت به IPv4 دارد. این آدرسها ۱۶ بایت طول دارند و دقیقاً مشکلی را حل کرده که به همان دلیل طراحی شد: یعنی تقریباً فضای نامحدودی از آدرسهای IP را فراهم آورده است. در این خصوص بیشتر صحبت خواهیم کرد.

دومین پیشرفت عمدهٔ IPv6 ساده‌سازی سرآیند آنست. این سرآیند جمعاً هفت فیلد دارد (در مقابل سیزده فیلد در IPv4). این تغییر، امکان آنرا فراهم آورده که مسیریاب بسته‌ها را سریعتر پردازش نماید و ظرفیت مفید مسیریاب را افزایش و تأخیر را کاهش دهد. در ادامه مختصراً به سرآیند خواهیم پرداخت.

سومین بهبود عمدهٔ آن پشتیبانی از گزینه‌های اختیاری (Options) است. این تغییر برای سرآیند جدید حیاتی بود چراکه برخی از فیلدهایی که در نسخهٔ قبلی وجودشان الزامی است در نسخهٔ فعلی اختیاریند. مضاف بر این، روش درج گزینه‌ها در این فیلد متفاوت از قبل است و اجازه می‌دهد مسیریابها بتوانند به سادگی از گزینه‌هایی که برایشان مهم نیست رد شوند. [یعنی در نسخهٔ جدید، دسترسی تصادفی و مستقیم به گزینه‌ها ممکن شده است.] این ویژگی سرعت پردازش بسته‌ها را افزایش می‌دهد.

چهارمین موضوعی که IPv6 در آن پیشرفت عمده‌ای داشته است، «**امنیت**» است. IETF با انبوهی از گزارشهای مطبوعاتی در خصوص نابغه‌های دوازده ساله‌ای مواجه بود که با کامپیوترهای شخصی خود، به شبکه‌های بانکی یا پایگاههای نظامی در سراسر اینترنت، نفوذ کرده بودند و جوّ شدیدی براه افتاده بود که باید برای بهبود امنیت شبکه‌ها کاری کرد. در نسخهٔ جدید IP، «احراز هویت» (Authentication) و «حفظ امنیت اطلاعات» (Privacy) جزو ویژگیهای کلیدی به شمار می‌رود. البته این ویژگیها بعداً به IPv4 نیز افزوده شد [با عنوان IPsec]، فلذا در حال حاضر این دو، در زمینهٔ امنیت تفاوت چندانی با هم ندارند.

موضوع آخر آنکه در نسخهٔ جدید به «کیفیت خدمات» (QoS) دقت بیشتری شده است. قبل از آن نیز تلاشهای جَسته و گریخته‌ای در این رابطه انجام شده بود ولیکن با رشد کاربردهای چند رسانه‌ای در اینترنت، این موضوع جدی‌تر و حساس‌تر به نظر می‌رسید.

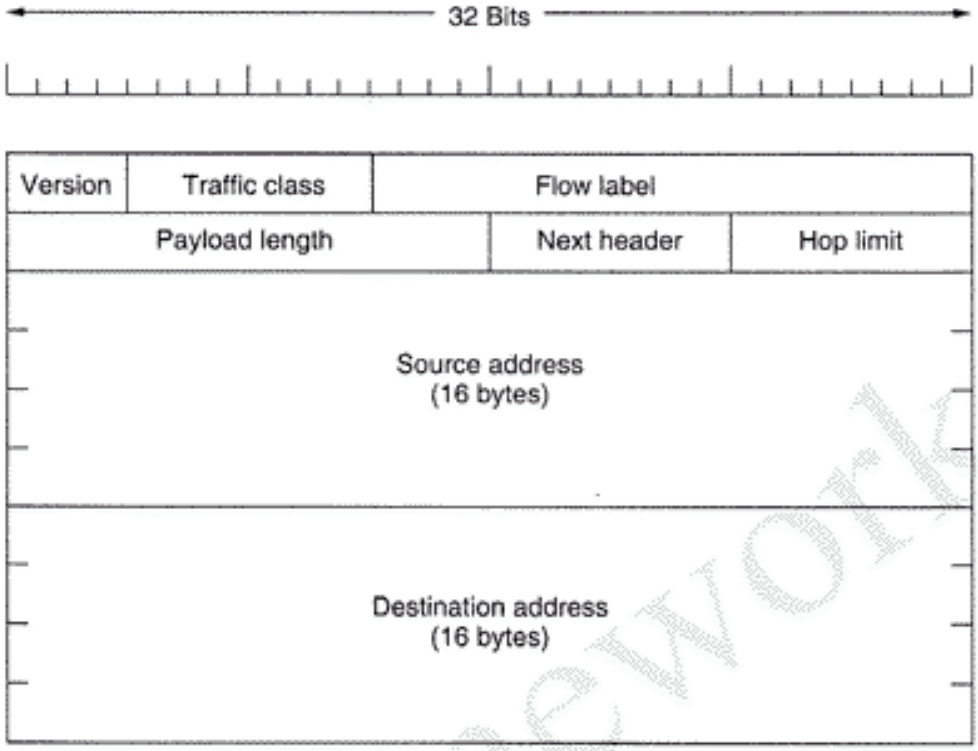
سرآیند اصلی IPv6

سرآیند اصلی IPv6 در شکل ۵-۶۸ نشان داده شده است. فیلد Version (شمارهٔ نسخهٔ پروتکل) برای IPv6 همیشه ۶ است. (کما اینکه برای IPv4 نیز همیشه ۴ است!). در دوران گذار از IPv4 به نسخهٔ جدید که ممکن است یک دهه طول بکشد، مسیریابها قادرند با بررسی این فیلد تشخیص بدهند که با چه نوع بسته‌ای روبرو هستند. البته از آنجایی که بررسی این فیلد به چندین دستورالعمل اجرایی CPU نیاز دارد و این کار زمان مفید پردازش هر بسته را هدر می‌دهد لذا در بسیاری از پیاده‌سازیهای عملی، برای اجتناب از این زمان تلفاتی، تشخیص آنکه یک بسته از نوع IPv4 است یا IPv6، با استفاده از فیلد خاصی در سرآیند لایهٔ پیوند داده‌ها بر عهدهٔ سخت‌افزار گذاشته شده است.[1] بدین ترتیب بسته‌ها براساس نوعشان مستقیماً به نرم‌افزار مناسب در لایهٔ شبکه هدایت می‌شوند. البته این الزام که لایهٔ پیوند داده از جزئیات نوع بسته‌های لایهٔ شبکه آگاه باشد با این اصل اساسی که «هر لایه نباید از معنای بیتهایی که از لایهٔ بالاتر تحویل او می‌شود، آگاه باشد» در تناقض است. بدون شک بحث و مناقشه بین طرفداران ایده‌های «انجام اصولگرایانه و صحیح کار» و «تسریع کار» به شدت ادامه خواهد داشت.

فیلد Traffic Class (کلاس ترافیک) برای تشخیص تفاوت بسته‌ها از لحاظ نیازمندیهای تحویل بی‌درنگ و

---

۱. مثلاً در فریم اترنت فیلد Type نوع بستهٔ درون فریم را تعیین می‌کند. -م



شکل ۵-۶۸. سرآیند ثابت و الزامی در IPv6.

QoS درخواستی، بکار می‌آید. فیلدی با همین منظور از ابتدا در IP وجود داشت ولیکن استفاده از آن به صورت پراکنده و سلیقه‌ای بر روی مسیریابها پیاده‌سازی شد و اغلب مسیریابها آن را نادیده می‌گرفتند. اکنون تجربیات گذشته چراغ راهی شده تا بتوان بهترین راه و روش تحویل بسته‌های اطلاعات چندرسانه‌ای را تعیین کرد.

فیلد Flow Label (برچسب جریان) همچنان آزمایشی است ولی کاربرد مورد نظر آن، این بوده که بتوان یک **«شبه اتصال»** (Pseudoconnection) بین مبداء و مقصد، با ویژگیها و نیازمندیهای خاصّ ایجاد کرد. به عنوان مثال، جریانی از بسته‌ها که از یک پروسه در مبداء خاص تولید و به سوی یک پروسه بر روی مقصد خاص روانه می‌شوند احتمالاً نیاز به تضمین تأخیر محدود و مشخص دارد و در نتیجه باید پهنای باند لازم را رزرو کرد. در چنین مواردی می‌توان پیشاپیش یک «جریان» (Flow) با مشخصات درخواستی تنظیم کرد و به آن یک شناسه اختصاص داد. هر گاه مسیریاب بسته‌ای دریافت کند و فیلد Flow Label آن غیرصفر باشد، با مراجعه به جداول درونی خود تشخیص می‌دهد که با این بسته چگونه رفتار کند. در حقیقت استفاده از مفهوم «جریان»[1] در IPv6، تلاشی است برای رسیدن به قابلیت انعطاف در زیرشبکه‌های دیتاگرام و تضمین کیفیت خدمات در زیرشبکه‌های مدار مجازی.

هویت هر «جریان» برحسب آدرس مبداء، آدرس مقصد و شمارهٔ جریان (برچسب جریان) مشخص می‌شود فلذا بین دو مبداء و مقصد در شبکه می‌توان بطور همزمان چندین «جریان» فعال تنظیم کرد. همچنین در این روش حتی اگر دو جریان متفاوت با شمارهٔ جریان یکسان از دو ماشین میزبان مختلف تولید و از مسیریابهای مشابهی عبور کنند، مسیریابها به کمک آدرس مبداء و مقصد قادر به تشخیص آنها خواهند بود. انتظار آنست که «برچسبهای جریان» به جای آنکه به صورت ترتیبی و از ۱ شروع شوند به صورت کاملاً تصادفی انتخاب گردند تا مسیریاب

---

۱. برای آشنایی با مفهوم جریان رجوع کنید به بخش ۵-۴-۱

بتواند آنها را در Hash Table خود درج کند.[۱]

فیلد Payload Length (طول قسمت حمل داده) مشخص می‌کند که پس از سرآیند ۴۰ بایتی در شکل ۵-۶۸ چند بایت داده قرار گرفته است. همین فیلد در IPv4 با نام Total Length وجود داشت. تغییر نام به آن دلیل بوده که در نسخهٔ جدید ، سرآیند جزو طول بسته به حساب نمی‌آید بلکه فقط اندازه بخش حمل داده تعیین می‌شود.

فیلد Next Header ساختار بسته را سبکبار کرده است! دلیل آنکه سرآیند بسته ساده شده آنست که می‌توان در صورت لزوم سرآیند اضافی و انتخابی داشت. این فیلد مشخص می‌کند که پس از سرآیند ۴۰ بایتی کدامیک از سرآیندهای ششگانهٔ اضافی قرار گرفته است (در صورت وجود). اگر سرآیند اخیر، آخرین سرآیند بستهٔ IP باشد، این فیلد مشخص می‌کند که کدام پروسه در لایهٔ انتقال محتوای بسته را تحویل خواهد گرفت (مثلاً TCP ، UDP و نظائر آن).[۲]

کاربرد فیلد Hop Limit آنست که بسته‌ها عمر محدودی داشته باشند. این فیلد در عمل مشابه با فیلد Time to Live (زمان حیات بسته) در IPv4 است یعنی به ازای عبور بسته از یک مسیریاب، یک واحد از مقدار آن کاسته می‌شود. در تئوری، مبنایی که IPv4 برای این فیلد در نظر داشت، «زمان بر مبنای ثانیه» بود در حالی که هیچ مسیریابی از چنین مبنایی استفاده نمی‌کند (بلکه به ازای هر گام یک واحد از آن می‌کاهد) لذا نام این فیلد را به گونه‌ای عوض کردند که عملکرد واقعی آن را نشان بدهد. هر گاه مقدار این فیلد در یک بسته به صفر برسد آن بسته حذف خواهد شد.

در ادامه فیلدهای Source Address و Destination Address (آدرس مبداء و مقصد) قرار گرفته‌اند. در طرح پیشنهادی آقای Deering آدرسها ۸بایتی انتخاب شده بودند در حالی که در مراحل بازبینی دیگران احساس کردند که شاید این فضای آدرس نیز در خلال چند دهه، IPv6 را نیز با کمبود فضای آدرس مواجه کند، در حالی که با آدرسهای ۱۶ بایتی هرگز چنین کمبودی رخ نخواهد داد. برخی از افراد معتقد بودند که آدرسهای ۱۶ بایتی بیش از حد بزرگ هستند در حالی برخی دیگر اعتقاد داشتند باید از آدرسهای ۲۰ بایتی استفاده شود تا با پروتکل دیتاگرام پیشنهادی OSI سازگار باشد. گروه دیگری نیز به آدرسهای با طول متغیر گرایش داشتند. پس از بحث و جدل فراوان، به این نتیجه رسیدند که آدرسهای با طول ثابت ۱۶ بایتی بهترین انتخاب است.

با توجه به طول زیاد آدرسهای IP ، نماد جدیدی برای نوشتن آنها پیشنهاد شد. این آدرسها به صورت هشت **گروه** که با علامت : از هم جدا شده، نوشته می‌شوند. هر گروه نیز به صورت چهار رقم هگزادیسمال نمایش داده‌می‌شود:

8000:0000:0000:0000:0123:4567:89AB:CDEF

از آنجایی که در آدرسها، تعداد ارقام صفر زیاد است، سه نوع بهینه‌سازی مجاز شمرده شده: صفرهای سمت چپ در هر گروه نوشته نمی‌شوند یعنی 0123 به صورت 123 نشان داده می‌شود؛ دوم آنکه اگر یک یا چند گروه شانزده بیتی تماماً صفر باشند با یک زوج علامت :: نشان داده می‌شود. بنابراین آدرس مثال بالا به صورت زیر نوشته خواهد شد:

8000::123:4567:89AB:CDEF

نهایتاً آنکه آدرسهای IPv4 را می‌توان با یک جفت :: و سپس آدرس نقطه‌دار قدیمی، نشان داد:

::192.31.20.46

---

۱. به عبارت دیگر مسیریاب انتظار دارد این شماره‌ها را Hash کند لذا این شماره‌ها نباید متوالی باشند.

۲. به عبارت دیگر محتوای این فیلد به صورت بازگشتی سرآیندهای بعدی را مشخص می‌کند تا نهایتاً به سرآیند آخر برسد که نوع بستهٔ لایهٔ انتقال را تعیین می‌نماید. -م

شاید لازم به گفتن نباشد که آدرسهای شانزده بایتی، فضایی معادل 2^{128} آدرس هستند که چنین فضایی تقریباً معادل 3×10^{38} آدرس است. اگر کل کرهٔ زمین شامل خشکیها و دریاها پر از کامپیوتر شوند باز هم IPv6 می‌تواند برای هر مترمربع 7×10^{23} آدرس IP فراهم کند. دانشجویان رشته شیمی می‌دانند که این عدد حتی از عدد آووگادرو نیز بزرگ است. [عدد آووگادرو $6/02\times10^{23}$ است]. چون در نظر نبوده که حتی به مولکولهای سطح زمین آدرس بدهیم آدرسهای IP شانزده بایتی، بهیچوجه کم نخواهد آمد!!

در عمل از فضای آدرس IP، بخوبی استفاده نخواهد شد. (دقیقاً همانند فضای شماره‌های تلفن که مثلاً فضای شماره‌های تلفن منهتن با پیش شمارهٔ ۲۱۲ پر شده ولی فضای شماره‌های ویومینگ (Wyoming) با پیش شمارهٔ ۳۰۷ تقریباً خالی مانده است.) دو نفر به نامهای Durand و Huitema در سند RFC 3194 محاسبه کرده‌اند که با ایده گرفتن از تخصیص شماره‌های تلفن و حتی در بدبینانه‌ترین حالت ممکن، باز هم می‌توان برای هر مترمربع از کرهٔ زمین، ۱۰۰۰ آدرس IP کنار گذاشت. در حالت کلی نیز می‌توان تریلیونها آدرس IP برای هر مترمربع از زمین در نظر گرفت. کوتاه سخن آنکه، در آینده هیچگاه به مشکل فضای آدرس برنخواهیم خورد.

مقایسهٔ سرآیند IPv4 (شکل ۵-۵۳) با سرآیند IPv6 (شکل ۵-۶۸) از این دیدگاه که چه فیلدی و چرا حذف شده است، آموزنده خواهد بود: فیلد IHL حذف شده زیرا سرآیند بسته‌های IPv6 طول ثابتی دارد. فیلد «پروتکل» وجود ندارد چرا که فیلد Next Header مشخص می‌کند که پس از آخرین سرآیند چه بستهٔ دیگری آمده است (بستهٔ TCP، UDP یا نظائر آن).

تمام فیلدهایی که در ارتباط با «قطعه‌قطعه‌سازی» بسته‌ها در IPv4 تعریف شده بود در IPv6 حذف گردیده است زیرا پروتکل اخیر راهکار دیگری برای مکانیزم قطعه‌قطعه سازی برگزیده است. انتظار آنست که تمام ماشینهای سازگار با IPv6 بتوانند به صورت خودکار و پویا اندازهٔ دیتاگرامها را تعیین کنند و بدین نحو نیاز به قطعه‌قطعه شدن بسته‌ها کمتر اتفاق می‌افتد. همچنین حداقل طول بسته‌ای که هر ماشین موظف به پذیرش آنست از ۵۷۶ بایت به ۱۲۸۰ بایت افزایش یافته تا بتوان یک قطعه دادهٔ ۱۰۲۴ بایتی را به همراه تعداد زیادی سرآیند (۲۵۶ بایت)، بدون نیاز به قطعه‌قطعه شدن ارسال و دریافت کرد. مضاف بر این، وقتی ماشینی یک بستهٔ بیش از حد بزرگ IPv6 را ارسال می‌دارد مسیریاب ناتوان از هدایت آن، به جای قطعه‌قطعه کردن بسته آن را حذف کرده و پیام خطایی را باز می‌گرداند. این پیام به ماشین میزبان تفهیم می‌کند که باید بسته‌هایش را بشکنند. البته اگر ماشین میزبان خودش بسته‌ها را با اندازهٔ مناسب ارسال کند کارآمدتر از آنست که بسته‌ها در طول مسیر شکسته شود.

فیلد Checksum نیز حذف شد زیرا محاسبهٔ آن کارآیی و سرعت پردازش بسته‌ها را به نحو چشمگیری کاهش خواهد داد. با توجه به قابلیت اعتماد شبکه‌های کنونی و با در نظر داشتن این حقیقت که در لایهٔ پیوند داده و لایهٔ انتقال نیز (بطور مجزا) صحت داده‌ها بررسی می‌شود، محاسبهٔ یک کد کشف خطای دیگر مثل checksum در مقایسه با کاهش کارآیی ارزشی ندارد. حذف این ویژگیهای زائد، IPv6 را به پروتکل متعادل و جمع و جور تبدیل کرده است. بدین ترتیب IPv6 به اهداف مورد نظر خود که همانا انعطاف، سرعت و فضای بزرگ آدرس بوده، نائل شده است.

سرآیندهای اضافی (سرآیندهای توسیع یا Extension Header)

گاهی به برخی از فیلدهای حذف شدهٔ IPv4 نیاز می‌شود و به همین منظور در IPv6 مفهوم جدیدی به نام «سرآیندهای توسیع» معرفی شده است. این سرآیندهای اختیاری برای افزودن اطلاعات به هر بسته بکار می‌آیند ولیکن روش کدینگ (و جاسازی) آنها کارآمد و سریع است. شش نوع مختلف سرآیند توسیع که تاکنون معرفی شده، در شکل ۵-۶۹ فهرست گردیده است. هر کدام از این سرآیندها اختیارند ولیکن اگر به بیش از یک سرآیند نیاز باشد باید بطور پیاپی، پس از سرآیند ثابت و ترجیحاً به ترتیب فهرست، قرار بگیرند.

نام سرآیند توسیع (سرآیند اضافی)	توصیف عملکرد
Hop-by-hop options	حاوی اطلاعات گوناگون برای مسیریابها
Destination options	اطلاعات اضافی برای مقصد
Routing	فهرست ناکاملی از مسیریابها که بسته باید از آنها بگذرد.
Fragmentation	مدیریت قطعات دیتاگرام
Authentication	بررسی هویت فرستنده
Encrypted security payload	اطلاعاتی در خصوص محتوای رمزنگاری شده بسته

شکل ۵-۶۹. سرآیندهای توسیع در IPv6 (Extension Header).

برخی از سرآیندها دارای قالب ثابتی هستند در حالی که برخی دیگر تعداد متغیری فیلد با طول متفاوت دارند. به همین دلیل هر آیتم در قالب سه تایی **(نوع، طول، مقدار)**[1] سازماندهی و کُد می‌شود. فیلد Type مشخص می‌کند که نوع گزینه چیست. مقدار فیلد **نوع** (Type) به نحوی انتخاب شده است که دو بیت ابتدایی آن به مسیریابهایی که نمی‌دانند آن گزینه را چگونه پردازش کنند، راه و چگونگی کار را نشان می‌دهند. این راهکارها عبارتند از: (۱) گزینهٔ مربوط را نادیده بگیر (۲) بسته را حذف کن (۳) بسته را حذف و یک بستهٔ ICMP برگردان (۴) بسته را حذف کن و یک بستهٔ ICMP برگردان ولیکن بستهٔ ICMP را برای آدرسهای «چندپخشی» (Multicast) نفرست. (تا یک بستهٔ چندپخشی اشتباه، منجر به تولید میلیونها گزارش ICMP نشود.)

فیلد یک بایتی **طول** (Length) مشخص می‌کند که فیلد **مقدار** (Value) چند بایتی است. (صفر تا ۲۵۵ بایت). فیلد **مقدار** (Value) در برگیرنده اطلاعات مورد نیاز است و حداکثر می‌تواند ۲۵۵ بایت باشد.

«سرآیند توسیع Hop-by-Hop» (گام به گام) برای حمل اطلاعاتی کاربرد دارد که مسیریابهای واقع بر مسیر باید آنها را بررسی نمایند. قبلاً یکی از گزینه‌ها را معرفی کردیم: پشتیبانی از دیتاگرامهایی با طول بیش از ۶۴ کیلوبایت. قالب این سرآیند در شکل ۵-۷۰ نشان داده شده است. وقتی از این سرآیند استفاده می‌شود باید فیلد Payload Length (در سرآیند اصلی) به صفر مقداردهی شود.

Next header	0	194	4
Jumbo payload length			

شکل ۵-۷۰. «سرآیند توسیع Hop-by-Hop» (گام به گام) برای دیتاگرامهای بسیار بزرگ (جامبوگرام).

همانند تمام سرآیندهای اختیاری دیگر، این سرآیند نیز با فیلدی یک بایتی به نام Next Header شروع می‌شود و مشخص می‌کند که سرآیند بعدی از چه نوع است. پس از این بایت، بایت دیگری قرار گرفته که طول سرآیند Hop-by-Hop را بر مبنای بایت تعیین می‌کند ولیکن در مقدار آن، ۸ بایت ابتدایی (که وجود آن الزامی است) لحاظ نمی‌شود. تمام سرآیندهای توسیع دیگر نیز به همین نحو شروع می‌شوند. در ادامه فیلدی دو بایتی آمده که بایت اول مشخص می‌کند که این گزینه (Option) قرار است اندازهٔ دیتاگرام را تعریف کند (کد ۱۹۴) و بایت بعدی مشخص می‌کند که اندازهٔ دیتاگرام یک شمارهٔ چهار بایتی است. چهار بایت آخر این سرآیند، طول دیتاگرام را مشخص می‌کند. اندازهٔ زیر ۶۵۵۳۶ مجاز نیست و منجر به حذف بسته در اولین مسیریاب و بازگشت پیام خطای ICMP خواهد شد. دیتاگرامهایی که از این سرآیند اختیاری (یعنی Hop-by-Hop Header) استفاده

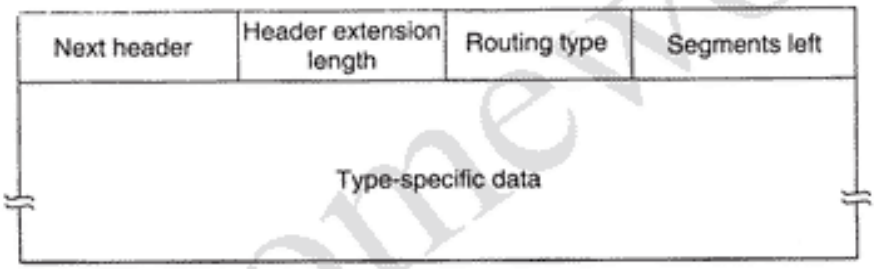
---

1. (Type,Length,Value)

کرده‌اند اصطلاحاً Jumbogram (دیتاگرام عظیم) نامیده می‌شوند. [بدین ترتیب در IPv6 می‌توان قطعات دادهٔ بسیار بزرگ را به کمک سرآیند فوق به صورت یکجا ارسال کرد.] کاربرد جامبوگرامها در سوپرکامپیوترها که باید چندین گیگابایت اطلاعات را از طریق اینترنت منتقل کنند، بسیار حیاتی است.

«سرآیند توسیع Destination Options» برای درج فیلدهایی در نظر گرفته شده که صرفاً توسط ماشین مقصد پردازش و تفسیر می‌شوند. در نسخهٔ اولیهٔ IPv6، مقدار این گزینه پوچ در نظر گرفته شده و کاربردی نداشته است. وجود چنین فیلدی برای آن بوده که نرم‌افزار ماشینهای میزبان و مسیریابها چنین سرآیندی را به رسمیت بشناسند تا اگر روزگاری به آن نیاز شد، شرایط مهیا باشد وگرنه باید پروتکل عوض شود.

«سرآیند توسیع Routing»، فهرست مسیریابهایی را مشخص می‌نماید که بسته باید در راه رسیدن به مقصد از آنها عبور کند. این گزینه شباهت زیادی به گزینهٔ Loose Source Routing در IPv4 دارد. فهرست آدرسهایی که در این سرآیند مشخص شده باید در طول مسیر و به ترتیب ملاقات شوند ولی این امکان وجود دارد که مسیریابهایی هم که آدرس آنها در فهرست نیست مابین مسیر باشند. قالب سرآیند Routing در شکل ۵-۷۱ مشخص شده است.



شکل ۵-۷۱. سرآیند توسیع برای مسیریابی (Routing).

چهار بایت اول از این سرآیند، شامل چهار فیلد یک بایتی است: دو فیلد Next Header و Header Extension Length را قبلاً تعریف کردیم. فیلد Routing Type ، ساختار مابقی سرآیند را مشخص می‌نماید: مقدار صفر مشخص کنندهٔ آنست که پس از کلمهٔ چهار بایتی اول، یک کلمهٔ چهار بایتی دیگر قرار گرفته و پس از آن آدرس IPv6 (یعنی آدرسهای ۱۲۸ بیتی) مسیریابها قرار می‌گیرد. به غیر از این ساختار، فعلاً ساختار دیگری تعریف نشده مگر آنکه در آینده چیز جدیدی ابداع شود. فیلد آخر یعنی Segment Left تعداد آدرسهایی را مشخص می‌کند که هنوز ملاقات نشده‌اند. مقدار اولیهٔ این فیلد معادل با تعداد مسیریابهایی است که آدرس آنها در فهرست مورد نظر درج شده است و به ازای ملاقات در هر مسیریاب که آدرس آن در فهرست آمده یک واحد از این فیلد کاسته می‌شود. وقتی مقدار این فیلد در بسته به صفر برسد بسته روال طبیعی طی مسیر خود را از سر می‌گیرد بدون آنکه اجبار به عبور از مسیر خاصی داشته باشد. معمولاً در چنین لحظه‌ای بسته به مقصد خود نزدیک شده است.

سرآیند Fragmentation مشابه با IPv4 ، با مسئله قطعه‌قطعه سازی بسته‌ها سر و کار دارد. در این سرآیند نیز فیلدهای «شمارهٔ شناسایی دیتاگرام»، «شمارهٔ قطعه» و یک بیت MF تعریف شده‌اند که بیت MF مشخص می‌کند که آیا قطعهٔ جاری آخرین قطعهٔ دیتاگرام است یا آنکه قطعات دیگری در ادامه وجود دارند. البته در IPv6 برخلاف IPv4 ، فقط ماشین مبداء می‌تواند بسته‌ای را قطعه‌قطعه کند و مسیریابهای واقع بر روی مسیر قادر به چنین کاری نیستند. اگرچه این موضوع از لحاظ فلسفی یک واپسگرایی محسوب می‌شود ولی در عوض کار مسیریابها را ساده‌تر کرده و فرآیند مسیریابی سریعتر خواهد شد. همانگونه که قبلاً اشاره کردیم هر گاه یک مسیریاب با بسته‌ای

بیش از حد بزرگ مواجه گردد آن را حذف کرده و بستهٔ ICMP (حامل پیغام خطا و اطلاعات مفید دیگر) به مبداء آن بر می‌گرداند. اطلاعات ارسالی به مبداء بسته، امکان آنرا می‌دهد که به کمک این سرآیند، بسته را به قطعات کوچکتر تقسیم و آنها را از نو ارسال کند.

سرآیند Authentication (سرآیند احراز هویت) مکانیزمی را فراهم آورده تا گیرنده بتواند از هویت فرستندهٔ بسته مطمئن شود. سرآیند Encrypted Security Payload اجازه می‌دهد تا محتوای بسته رمزنگاری شود و بدین ترتیب فقط گیرندهٔ مورد نظر قادر به خواندن آنست. این گونه سرآیندها برای انجام مأموریت خود از تکنیکهای رمزنگاری بهره می‌گیرند.

اختلاف نظرها و مناقشات

نظر به آنکه فرآیند طراحی IPv6، «باز» بوده و افراد درگیر در طراحی، بر عقاید خود تأکید داشته‌اند فلذا شگفت‌آور نیست که بسیاری از گزینه‌های انتخابی در IPv6 متناقض باشند. در زیر اجمالاً برخی از آنها را بررسی خواهیم کرد. برای آگاهی از جزئیات ماجرا به RFCهای مربوطه مراجعه نمائید.

قبلاً اشاره کردیم که بحث و جدل گسترده‌ای پیرامون طول آدرسها وجود داشت و توافق نهایی آن بود که آدرسها با طول ثابت و ۱۶ بایتی باشند.

جدل دیگری بر سر حداکثر تعداد گام (Hop Limit) در گرفت. یک گروه احساس می‌کرد که محدود کردن حداکثر تعداد گام (Hop) به ۲۵۵ یک اشتباه محض است چراکه اگرچه در آن زمان حداکثر طول مسیرها عموماً از ۳۲ تجاوز نمی‌کرد ولی مدعی بودند که ممکن است ده سال بعد مسیرها طولانی‌تر از ۲۵۵ باشند. استدلال آنها این بود که فضای آدرس ۱۶ بایتی آینده‌نگری بیش از اندازه و در عوض مقدار کم Hop Count، کوته‌نظری است. از دیدگاه آنها بزرگترین اشتباه یک دانشمند کامپیوتر، آنست که برای هر فیلدی، تعداد بیت کمی در نظر بگیرد.

پاسخ گروه مقابل آن بود که افزایش بی‌مورد فضای هر فیلد منجر به تشکیل یک سرآیند حجیم خواهد شد. همچنین استدلال دیگرشان آن بود که وظیفهٔ فیلد Hop Count جلوگیری از سرگردانی بسته‌ها به مدت طولانی است و ۶۵۵۳۵ گام بیش از حد زیاد است. استدلال آخر آنکه با رشد اینترنت، لینکهای بسیار طولانی ساخته می‌شوند و این امکان فراهم می‌شود که برای رسیدن از یک کشور به کشور دیگر به کمتر از ده گام نیاز باشد. اگر یک بسته برای رسیدن از مبداء به مقصد مجبور شود از ۱۲۵ مسیریاب بین‌المللی بگذرد، ستون فقرات این شبکهٔ بین‌المللی در جایی اشکال دارد! بدین ترتیب طرفداران فیلد ۸ بیتی در عقیدهٔ خود پیروز شدند.

یکی دیگر از بحثهای داغ بر سر حداکثر طول بسته‌ها بود. سوپرکامپیوترها به بسته‌هایی با طول بیش از ۶۴ کیلوبایت احتیاج داشتند. وقتی یک سوپرکامپیوتر شروع به ارسال می‌کند و به کار خود مشغول می‌شود نباید به ازای هر ۶۴ کیلوبایت یکبار متوقف شود. استدلال گروه مخالف آن بود که اگر یک بستهٔ یک مگابایتی در طول مسیر به یک خط T1 برسد آن خط به مدت حداقل ۵ ثانیه اشغال شده و کاربران دیگری که در این خط سهیم هستند با تأخیر قابل توجهی روبرو خواهند شد.[1] توافق نهایی بدینجا ختم شد که بسته‌های معمولی حداکثر ۶۴ کیلوبایتی باشند ولی به کمک سرآیند اختیاری Hop-by-Hop بتوان جامبوگرامهایی با هر طول دلخواه ارسال کرد.

موضوع سوم مناقشه، حذف فیلد IPv4 checksum (کد تشخیص خطاهای احتمالی در سرآیند) بود. برخی از افراد حذف این فیلد را مشابه با برداشتن ترمزهای یک خودرو می‌دانستند که اگرچه ماشین را سبکبار و سریعتر می‌کند ولی اگر اتفاق غیرمترقبه‌ای رخ بدهد مشکل [illegible] می‌آید. [illegible] گروه مقابل آن بود که هر برنامه کاربردی که نگران صحت داده‌های خود است باید از پروتکلی در لایهٔ انتقال بهره بگیرد که داده‌ها را از لحاظ

۱. نرخ ارسال خط T1 را 1.544 مگابیت در نظر بگیرید.

سلامت بررسی می‌کند. لذا اضافه کردن کد کنترلی دیگر به لایهٔ IP برای کشف خطا (در حالی که هر بسته یکبار هم در لایهٔ پیوند داده بررسی می‌شود) بیهوده و زائد است. مضاف بر آن، تجربه نشان داده بود که محاسبهٔ جمع کنترلی (Checksum) در IPv4 هزینهٔ بالایی دارد. در این مناقشه نیز طرفداران حذف کد کشف خطا پیروز شدند.

موضوع دیگر، بحث ماشینهای همراه بود: وقتی یک کامپیوتر قابل حمل، در نیمی از کل دنیا حرکت می‌کند (مثلاً درون هواپیما)، آیا می‌تواند با همان آدرس IPv6 قبلی ، کار خود را ادامه بدهد یا آنکه مجبور به استفاده از ساختار «عامل خانگی» و «عامل خارجی» است؟ ماشینهای همراه مشکل «عدم تقارن» را به سیستم مسیریابی تحمیل می‌کنند: یک کامپیوتر همراه و کوچک براحتی قادر به شنیدن سیگنال قوی منتشره از مسیریاب ثابت خود هست ولی مسیریاب ثابت براحتی قادر به احساس سیگنال ضعیف ارسال شده توسط کامپیوتر همراه نیست. در نتیجه برخی از افراد گرایش داشتند که در IPv6 از ماشینهای همراه حمایت شود ولی تمام این تلاشها به دلیل آنکه بر روی هیچیک از طرحهای پیشنهادی توافقی بدست نیامد، با شکست مواجه گردید.

شاید بزرگترین مناقشه بر سر موضوع «امنیت» بود: همه بر این اصل که «امنیت لازم است» اشتراک نظر داشتند. دعوا بر سر چگونگی رسیدن به امنیت و محل پرداختن به آن بود. اولین محل پرداختن به امنیت لایهٔ شبکه است. استدلال موافقین مبنی بر آن بود که پیاده‌سازی امنیت در لایهٔ شبکه، سرویسی استاندارد فراهم می‌کند که تمام برنامه‌های کاربردی بدون هیچگونه برنامه‌ریزی قبلی می‌توانند از آنها بهره بگیرند. استدلال مخالفین نیز آن بود که برنامه‌های کاربردی امن، عموماً به هیچ مکانیزمی کمتر از رمزنگاری انتهابه‌انتها (End-to-End Encryption) احتیاج ندارند، به نحوی که پروسهٔ مبداء خودش داده‌های ارسالی خود را رمز کرده و پروسهٔ مقصد آنها را از رمز خارج کند. هر چیزی کمتر از این، می‌تواند کاربر را با خطراتی مواجه کند که از اشکالات امنیتی لایهٔ شبکه ناشی می‌شود و هیچ تقصیری از او نیست. پاسخ به این استدلال آن بود که کاربر می‌تواند امنیت لایهٔ IP را نادیده بگیرد و کار خودش را انجام بدهد! پاسخ نهایی مخالفین نیز آن بود که افرادی که به عملکرد صحیح شبکه (در خصوص امنیت) اعتماد ندارند چرا باید هزینهٔ پیاده‌سازی سنگین و کندی IP را بپردازند!!

یکی دیگر از جنبه‌های مربوط به امنیت این حقیقت بود که بسیاری از کشورها قوانین سخت‌گیرانه‌ای در مورد صادرات محصولات مرتبط با رمزنگاری وضع کرده‌اند. از مثالهای بارز می‌توان به فرانسه و عراق اشاره کرد که حتی استفاده از رمزنگاری در داخل را نیز محدود کرده‌اند و عموم افراد نمی‌توانند چیزی را از پلیس مخفی نگه دارند. در نتیجه هر گونه پیاده‌سازی از IP که از روشهای رمزنگاری قوی استفاده می‌کند مجوز صدور از ایالات متحده (و بسیاری از کشورهای دیگر) را نخواهد گرفت. پیاده‌سازی دو نرم‌افزار یکی برای کاربرد داخلی و یکی برای صادرات، موضوعی است که عرضه‌کنندگان صنعت کامپیوتر با آن مخالفند.

موضوعی که پیرامون آن هیچ اختلاف نظر پیش نیامد آن بود که نمی‌توان انتظار داشت صبح روز یکشنبه IPv4 را در اینترنت از کار انداخت و صبح دوشنبه IPv6 را روشن نمود. در عوض مسیریابها و ماشینهایی که به IPv6 مجهز می‌شوند به مثابهٔ جزایر مستقل با استفاده از تونل با یکدیگر مبادلهٔ داده می‌کنند و با افزایش این جزایر، در یکدیگر ادغام شده و جزیرهٔ بزرگتری پدید می‌آید. در نهایت تمام این جزایر به هم می‌پیوندند و اینترنت کاملاً متحول می‌شود.

سرمایه‌گذاری حجیمی که از قبل بر روی مسیریابهای مبتنی بر IPv4 صورت گرفته، فرآیند تغییر و تحول اینترنت را سالها به تأخیر می‌اندازد. به همین دلیل تلاش زیادی صورت می‌گیرد تا این اطمینان حاصل شود که گذار از IPv4 به IPv6 حتی‌الامکان بدون زحمت و گرفتاری انجام گیرد. برای کسب آگاهی بیشتر در خصوص IPv6 به مرجع (Loshin, 1999) مراجعه نمایید.

۷-۵ خلاصه

لایهٔ شبکه به لایهٔ انتقال خدماتی را عرضه می‌کند. این لایه می‌تواند مبتنی بر «مدارات مجازی» یا «دیتاگرام» باشد. در هر دو حالت، وظیفهٔ اصلی این لایه آنست که بسته‌های داده را از مبداء به مقصد برساند. در «زیرشبکه‌های مدار مجازی»، تصمیم‌گیری در خصوص مسیر هدایت بسته‌ها، فقط یکبار و آن هم در حین تنظیم مدار مجازی انجام می‌شود. در «زیرشبکه‌های دیتاگرام»، این تصمیم‌گیری به ازای هر بسته تکرار می‌شود.

در شبکه‌های کامپیوتری از الگوریتمهای مسیریابی متنوعی استفاده می‌شود: الگوریتمهای ایستا نظیر «مسیریابی کوتاهترین مسیر» و «ارسال سیل‌آسا» (Flooding) و الگوریتمهای پویا شامل «مسیریابی بردار فاصله» و «مسیریابی حالت لینک» (Link State). در اغلب شبکه‌های واقعی بزرگ، از یکی از این دو روش مسیریابی پویا استفاده شده است. از موارد مهم دیگر در مسیریابی می‌توان به روش مسیریابی سلسله‌مراتبی، مسیریابی در محیط ماشینهای سیّار، مسیریابی پخش فراگیر، مسیریابی چندپخشی، و مسیریابی در شبکه‌های همتابه‌همتا اشاره کرد.

زیرشبکه‌ها ممکن است به سادگی با ازدحام مواجه شوند که منجر به افزایش تأخیر و کاهش ظرفیت مفید خروجی مسیریابها خواهد گردید. طراحان شبکه سعی می‌کنند با طراحی مناسب از بروز آن جلوگیری نمایند. این تکنیکها عبارتند از: تغییر در سیاستهای ارسال مجدد بسته‌ها، ذخیره‌سازی در حافظهٔ نهان، کنترل جریان و نظائر آنها. به هر حال اگر ازدحام رخ داد باید به نحوی به رفع آن اقدام کرد. می‌توان «بسته‌های دعوت به آرامش» (Choke Packet) پس فرستاد، بخشی از بار را دور ریخت و یا مکانیزمی نظیر به اینها را بکار بست.

مرحلهٔ بعدی پس از حل مسئله ازدحام، تلاش در تضمین کیفیت خدمات است. بدین منظور می‌توان از روشهایی نظیر «بافر کردن داده‌ها در ماشین مشتری»، «شکل‌دهی به ترافیک»، «رزرو منابع» و «کنترل پذیرش» استفاده کرد. راهکارهایی که برای تضمین کیفیت خوب خدمات طراحی و پیاده‌سازی شده عبارتند از: خدمات مجتمع (شامل RSVP)، خدمات متمایز و MPLS.

شبکه‌ها از جهات متعدد با هم تفاوت دارند، لذا وقتی می‌خواهند به یکدیگر متصل شوند مشکلاتی رخ خواهد داد. برخی از این مشکلات را می‌توان با تکنیک ایجاد تونل (Tunneling) از میان شبکهٔ واسط کاهش داد ولی اگر شبکه‌های مبداء و مقصد خودشان متفاوت باشند این راهکار نیز با شکست مواجه می‌شود. وقتی شبکه‌های متفاوت حداکثر طول بسته‌هایشان فرق کند به تکنیک قطعه‌قطعه‌سازی بسته‌ها نیاز خواهد شد.

در اینترنت مجموعهٔ وسیعی از پروتکلهای مرتبط با لایهٔ شبکه وجود دارد. از بین تمام اینها، پروتکل IP انتقال واقعی بسته‌ها را بر عهده دارد ولیکن پروتکلهای کنترلی دیگر مثل ICMP ، ARP و RARP و همچنین پروتکلهای مسیریابی مثل OSPF و BGP در کنار آن به فرآیند هدایت و انتقال بسته‌ها کمک می‌کنند. اینترنت سریعاً با مشکل کمبود فضای آدرس مواجه شد و برای رفع این مشکل نسخهٔ جدید IP یعنی IPv6 طراحی گردید.

مسائل

۱. دو مثال از برنامه‌های کاربردی کامپیوتر ارائه بدهید که برای آنها سرویس اتصال‌گرا (Connection Oriented) مناسب است. دو نمونهٔ دیگر از برنامه‌های کاربردی معرفی کنید که در آنها سرویس بدون اتصال مناسبتر است.

۲. آیا وضعیتی پیش می‌آید که در سرویس اتصال‌گرا بسته‌ها خارج از ترتیب تحویل مقصد شوند؟ پاسخ خود را تشریح کنید.

۳. زیرشبکه‌های دیتاگرام، هر بسته را به صورت واحدهای مجزا و مستقل از هم مسیریابی و هدایت می‌کنند. زیرشبکه‌های مدار مجازی بدین نحو عمل نمی‌کنند فلذا بسته‌های داده، مسیری از قبل تعیین‌شده را

می‌پیمایند. آیا این تعبیر بدان معناست که زیرشبکه‌های مدار مجازی به این قابلیت که بتوانند بسته‌های مجزا و مستقل را از یک مبداء دلخواه به مقصد مورد نظر هدایت کنند نیاز ندارند؟ پاسخ خود را شرح بدهید.

٤. سه نمونه از پارامترهایی را که می‌توان در حین تنظیم یک اتصال، بر روی آنها مذاکره و توافق کرد، نام ببرید.

۵. به مسئلهٔ زیر در خصوص پیاده‌سازی «سرویس مدار مجازی» دقت نمائید: اگر در درون زیرشبکه از الگوی مدار مجازی استفاده شده باشد هر بستهٔ داده باید یک سرآیند سه بایتی داشته باشد و هر مسیریاب نیز برای شناسایی هر مدار مجازی به ۸ بایت احتیاج دارد. اگر در درون زیرشبکه از الگوی دیتاگرام استفاده شده باشد، به سرآیندهای ۱۵ بایتی نیاز خواهد بود ولی در عوض مسیریاب به فضای حافظهٔ اضافی نیاز نخواهد داشت. هزینه ارسال را برای هر 10^6 بایت یک سنت (به ازای عبور از هر مسیریاب) فرض کنید. حافظه بسیار سریع لازم برای مسیریابها را یک سنت به ازای هر بایت در نظر بگیرید که با فرض ۴۰ ساعت کار در هر هفته، در عرض دو سال مستهلک می‌شود. بطور متوسط هر نشست هزار ثانیه طول می‌کشد و در خلال نشست ۲۰۰ بسته منتقل می‌شود.[1] هر بسته نیز بطور متوسط از چهار مسیریاب عبور می‌کند. پیاده‌سازی کدامیک از این الگوها ارزان‌تر تمام می‌شود و به چه میزان؟

۶. فرض کنید که تمام مسیریابها و ماشینها به درستی کار می‌کنند و تمام نرم‌افزارها از خطا مصون باشند. آیا باز هم احتمال آنکه یک بسته به اشتباه تحویل ماشینی دیگر شود وجود دارد؟ (حتی اگر این احتمال بسیار ناچیز باشد.)

۷. به شبکهٔ شکل ۵-۷ دقت کنید ولیکن وزنهای خطوط را نادیده بگیرید. فرض کنید مسیریابی طبق الگوریتم سیل‌آسا (Flooding) انجام می‌شود. اگر بسته‌های ارسالی از A به D دارای «شمارهٔ گام» (Hop count) معادل ۳ باشند فهرست مسیرهایی را که بسته از آنها عبور می‌کند، مشخص نمائید. همچنین تعیین کنید که کل گامهایی که بسته‌ها پیموده و پهنای باندی که به ازای آن تلف می‌شود چقدر است.

۸. روشی ساده و ذهنی برای پیدا کردن دو مسیر از یک مبداء مشخص به مقصدی خاص در شبکه ارائه بدهید به نحوی که بتواند در مقابل از بین رفتن خطوط انتقال، دوام بیاورد. (با فرض آنکه حداقل دو مسیر وجود دارد). مسیریابها را قابل اعتماد فرض کنید لذا نگران از کار افتادن آنها نباشید.

۹. به زیرشبکه شکل ۵-۱۳-الف دقت کنید. در این زیرشبکه از «مسیریابی بردار فاصله» استفاده شده و بردارهای ذیل توسط مسیریاب C دریافت شده است:

از B : (5,0,8,12,6,2) از D : (16,12,6,0,9,10) از E : (7,6,3,9,0,4)

تأخیر اندازه‌گیری شده تا B و D و E به ترتیب ۶ و ۳ و ۵ است. جدول جدید مسیریابی در C چیست؟ خط خروجی و تأخیر تخمینی رسیدن به هر مقصد را مشخص نمائید.

۱۰. اگر میزان تأخیر به صورت اعداد هشت بیتی ذخیره شود و شبکه ۵۰ مسیریاب داشته باشد و بردارهای تأخیر در هر ثانیه دو بار مبادله شوند چه مقدار از پهنای باند یک خط دو طرفهٔ همزمان (Full Duplex)، در اثر استفاده از الگوریتم مسیریابی توزیع شده [بردار فاصله] تلف می‌شود؟ فرض کنید هر مسیریاب سه خط با مسیریابهای دیگر دارد.

۱۱. در شکل ۵-۱۴، حاصل OR کردن دو مجموعهٔ بیتهای ACF با یکدیگر ۱۱۱ می‌شود. آیا این موضوع تصادفی است یا در هر زیرشبکه و با هر وضعیتی رخ می‌دهد؟

۱۲. برای مسیریابی سلسله‌مراتبی در زیرشبکه‌ای با ۴۸۰۰ مسیریاب، اندازهٔ هر منطقه (Region) و دسته (Cluster) چقدر باشد تا جدول مسیریابی (در سلسله‌مراتب سه‌سطحی) حداقل شود؟ نقطهٔ شروع مناسب

---

۱. نشست را در یک عبارت غیرفنی می‌توانید ارتباط و محاورهٔ یک زوج ماشین در طول زمان فرض کنید.

آنست که فرض کنید داشتن k دسته و k منطقه و k مسیریاب تقریباً بهینه است، فلذا k تقریباً ریشهٔ سوم ۴۸۰۰ (تقریباً ۱۶) می‌شود. با سعی و خطا در همسایگی عدد ۱۶، بهترین تعداد دسته، منطقه و مسیریاب را پیدا کنید.

۱۳. در متن گفته شد که وقتی ماشین متحرک در محل همیشگی خود نیست، بسته‌های ارسالی به سوی LAN خانگی، متوقف و به سمت LAN موقت او تغییر مسیر داده می‌شود. در یک شبکهٔ IP که بر روی اترنت پیاده شده، این تغییر مسیر چگونه انجام می‌شود؟

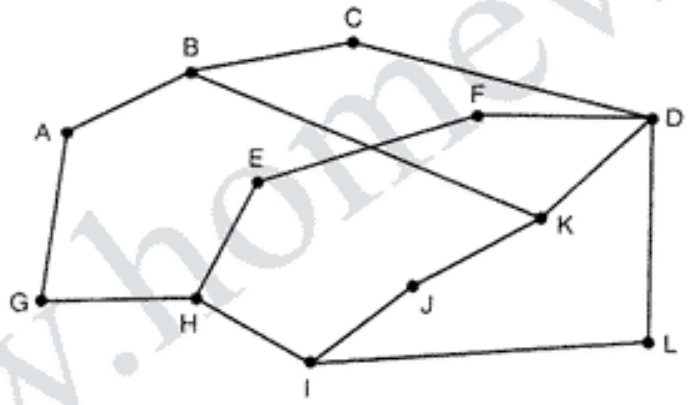
۱۴. با در نظر گرفتن زیرشبکه شکل ۵-۶، اگر B بسته‌ای را به صورت فراگیر پخش کند، چه تعداد بسته در زیرشبکه تولید می‌شود اگر:

الف) اگر از روش هدایت در مسیر معکوس (Reverse Path Forwarding) استفاده شود؟

ب) از روش Sink Tree استفاده شود؟

۱۵. شبکهٔ شکل ۵-۱۶-الف را مدنظر قرار بدهید. فرض کنید که خط جدیدی بین F و G اضافه می‌شود ولی کماکان Sink Tree شکل ۵-۱۶-الف بی‌تغییر باقی می‌ماند. در شکل ۵-۱۶-ج چه تغییری حاصل می‌شود؟

۱۶. برای زیرشبکه بعدی درخت پوشای چندپخشی (Multicast Spanning Tree) را برای مسیریاب C ترسیم کنید به نحوی که اعضای گروه مورد نظر مسیریابهای A و B و C و D و E و F و I و K باشند.



۱۷. در شکل ۵-۲۰، آیا گره‌های H یا I در جستجویی که از مبداء A شروع شده، هیچگونه ارسال فراگیر داشته‌اند؟

۱۸. در شکل ۵-۲۰ فرض کنید که گره B از نو راه‌اندازی شده است و هیچگونه اطلاعات مسیریابی در جدول خود ندارد. این گره به ناگاه احتیاج به یافتن مسیری به H پیدا می‌کند و بسته‌هایی پخشی (Broadcast) با مقدار TTL معادل ۱ و ۲ و ۳ و... را منتشر می‌کند. این کار چند دور طول می‌کشد تا نهایتاً مسیری پیدا شود؟

۱۹. در ساده‌ترین نسخهٔ الگوریتم Chord برای «جستجوی همتابه‌همتا» (Peer-to-Peer)، برای جستجو از جدول Finger استفاده نمی‌شود و به جای آن، جستجو به صورت خطی (یعنی گره‌به‌گره) در دو جهت بر روی دایره انجام می‌گیرد. آیا یک گره می‌تواند به درستی حدس بزند که جستجو را باید در کدام یک از جهات انجام بدهد؟ پاسخ خود را شرح بدهید.

۲۰. دایرهٔ Chord نشان داده شده در شکل ۵-۲۴ را مدنظر قرار بدهید. فرض کنید که گره ۱۰ به ناگاه فعال و وارد خط می‌شود. آیا ورود این گره در جدول Finger از گره ۱ تأثیری می‌گذارد و اگر این چنین است چگونه؟

۲۱. به عنوان یک مکانیزم کنترل ازدحام در زیرشبکه‌ای که در درون از مدار مجازی بهره گرفته است، هر مسیریاب می‌تواند از اعلام وصول یک بسته (Ack) طفره برود مگر آنکه: (۱) آگاه شود که آخرین ارسال او

بر روی مدار مجازی، با موفقیت دریافت شده است و (۲) بافر آزاد داشته باشد. برای سادگی فرض کنید مسیریاب برای اعلام وصول از پروتکل «توقف و انتظار» (Stop & Wait) استفاده کرده و برای هر مدار مجازی یک بافر اختصاصی (برای ترافیک هر یک از دو جهت) اختصاص داده است. اگر انتقال هر بسته T ثانیه طول بکشد (انتقال بسته یا پیام Ack به یک اندازه طول می‌کشد) و n مسیریاب بر روی مسیر قرار گرفته باشد، در چنین شرایطی بسته‌ها با چه نرخی تحویل ماشین مقصد می‌شوند؟ فرض کنید که خطای انتقال بسیار ناچیز و اتصال بین ماشین و مسیریاب بی‌نهایت سریع است.

۲۲. در یک زیرشبکه دیتاگرام به مسیریابها اجازه داده شده تا در صورت نیاز بسته‌ها را حذف نمایند. احتمال حذف بسته در هر مسیریاب را p فرض کنید. حالتی را در نظر بگیرید که ماشین مبداء به مسیریاب مبداء، مسیریاب مبداء مستقیماً به مسیریاب مقصد و مسیریاب مقصد نیز مستقیماً به ماشین مقصد متصل شده است. اگر یکی از این مسیریابها بسته‌ای را حذف نمایند، نهایتاً مهلت ماشین مبداء به سر می‌رسد و آن را از نو ارسال می‌نماید. اگر خط بین ماشین و مسیریاب را مثل خط بین دو مسیریاب، یک «گام» (Hop) فرض کنیم، مقدار هر یک از موارد زیر را حساب کنید:

الف) میانگین تعداد گامی که یک بسته در هر بار ارسال طی می‌کند.

ب) میانگین دفعات ارسال یک بسته

ج) میانگین گامهایی که یک بستهٔ دریافتی طی کرده است.

۲۳. دو تفاوت عمدهٔ روش «بیت هشدار» (Warning Bit) و روش RED را توضیح بدهید.

۲۴. دلیلی ارائه بدهید که چرا در الگوریتم «سطل سوراخ»، در هر تیک ساعت مجوز ارسال یک بسته صادر می‌شود، فارغ از آنکه طول بسته چقدر است.

۲۵. گونه‌ای از الگوریتم سطل سوراخ که مبتنی بر شمارش بایت است در سیستم خاصی بکار گرفته شده است. قاعده آنست که در هر تیک ساعت یک بستهٔ ۱۰۲۴ بایتی یا دو بستهٔ ۵۱۲ بایتی یا معادل آن، ارسال شود. یکی از محدودیتهای جدی چنین سیستمی را که در متن درس بدان نپرداخته‌ایم، بیان کنید.

۲۶. در یک شبکهٔ ATM از روش سطل نشانه‌دار برای شکل‌دهی به ترافیک استفاده شده است. در هر پنج میکروثانیه یک توکن جدید در سطل قرار داده می‌شود. هر توکن برای ارسال یک سلول حاوی ۴۸ بایت داده، مناسب می‌باشد. حداکثر نرخ ارسال مجاز چقدر است؟

۲۷. ترافیک یک کامپیوتر متصل به شبکهٔ 6 Mbps طبق الگوریتم سطل نشانه‌دار، شکل‌دهی و منظم می‌شود. سطل نشانه‌دار با نرخ یک مگابیت بر ثانیه پر می‌شود. ظرفیت سطل هشت مگابیت است که در همان ابتدا پر شده است. کامپیوتر در چه مدت می‌تواند با سرعت ۶ مگابیت بر ثانیه ارسال داشته باشد؟

۲۸. فرض کنید در توصیف مشخصات یک «جریان» حداکثر طول هر بسته ۱۰۰۰ بایت، نرخ سطل نشانه‌دار ده میلیون بایت در ثانیه، اندازهٔ سطل نشانه‌دار یک میلیون بایت و حداکثر نرخ ارسال ۵۰ میلیون بایت در ثانیه تعیین شده است. یک ماشین در چه مدتی قادر به ارسال انفجاری داده‌ها با حداکثر سرعت خود می‌باشد؟

۲۹. شبکهٔ شکل ۵-۳۷ از RSVP و درخت چندپخشی نشان داده شده برای ماشینهای ۱ و ۲، بهره گرفته است. فرض کنید که ماشین ۳ برای دریافت یک «جریان» از ماشین ۱، تقاضای کانالی با پهنای باند 2MB/sec می‌دهد. همچنین برای دریافت جریانی از ماشین ۲ تقاضای 1MB/sec می‌دهد. بطور همزمان ماشین ۴، 2MB/sec برای دریافت جریان از ماشین ۱ و همچنین ماشین ۵ برای دریافت جریان از ماشین ۲، 1MB/sec پهنای باند درخواست می‌کنند. مجموع کل پهنای باند رزرو شده در مسیریابهای A و B و C و E و H و J و K و L برای این درخواستها چقدر است؟

۳۰. پردازندهٔ یک مسیریاب قادر به پردازش ۲ میلیون بسته در هر ثانیه است. بار عرضه شده به آن ۱/۵ میلیون بسته در ثانیه است. اگر در مسیر رسیدن از مبداء به مقصد، ۱۰ مسیریاب قرار گرفته باشند، تأخیر ناشی از صف‌بندی و پردازش بسته چقدر است؟

۳۱. فرض کنید کاربری از خدمات متمایز و «هدایت پرشتاب» بهره می‌گیرد. آیا تضمینی وجود دارد که بسته‌های پرشتاب تأخیری کمتر از بسته‌های معمولی داشته باشند؟ دلیل خود را تشریح کنید.

۳۲. آیا به فرآیند قطعه‌قطعه‌سازی بسته‌ها در شبکه‌های بهم متصل شدهٔ مدار مجازی نیز احتیاج است یا آنکه این نیاز فقط در شبکه‌های دیتاگرام وجود دارد؟

۳۳. فرآیند ایجاد تونل از میان یک شبکهٔ الحاق شده مدار مجازی ساده و سرراست است: مسیریاب چند پروتکلی در یکی از دو طرف یک مدار مجازی با مسیریاب طرف دیگر ایجاد و تنظیم می‌کند و بسته‌ها را از طریق این مدار مجازی مبادله می‌نماید. آیا ایجاد تونل از میان یک شبکهٔ دیتاگرام نیز ممکن است؟ اگر جواب مثبت است چگونه؟

۳۴. فرض کنید که ماشین میزبان A به مسیریاب R1 ، R1 نیز به مسیریاب دیگری به نام R2 ، و R2 نیز به ماشین میزبان B متصل است. فرض کنید که یک پیام TCP حاوی ۹۰۰ بایت داده و ۲۰ بایت سرآیند، جهت تحویل به B به نرم‌افزار IP در ماشین A تسلیم می‌شود. محتوای فیلدهای Total Length ، Identification ، DF، MF و Fragment offset را در هر بستهٔ IP که از یکی از سه لینک A-R1 ، R1-R2 ، R2-B عبور می‌کند، مشخص نمایید؛ لینک A-R1 می‌تواند از فریمی با طول حداکثر ۱۰۲۴ بایت (که ۱۴ بایت آن هم سرآیند فریم لایهٔ پیوند داده‌ها محسوب می‌شود) پشتیبانی می‌کند. لینک R1-R2 می‌تواند از فریمی با طول ۵۱۲ بایت حمایت کند (با احتساب ۸ بایت سرآیند فریم) و لینک R2-B از فریمهایی با طول حداکثر ۵۱۲ بایت (با احتساب ۱۲ بایت سرآیند فریم) پشتیبانی می‌کند.

۳۵. یک مسیریاب بسته‌های IP با طول ۱۰۲۴ بایت (با احتساب داده و سرآیند) در خروجی خود گسیل می‌دارد. با فرض آنکه بسته‌ها فقط برای ده ثانیه زنده می‌مانند، مسیریاب حداکثر با چه سرعتی می‌تواند بسته‌ها را بفرستد بدون آنکه خطر به صفر برگشتن فیلد شمارهٔ شناسایی دیتاگرام (Datagram ID) وجود داشته باشد؟

۳۶. یک دیتاگرام IP که از گزینهٔ Strict Source Routing استفاده کرده، مجبور است قطعه قطعه شود. به نظر شما آیا این گزینه بایستی در تمام قطعات آن قرار داده شود یا آنکه قرار دادن این گزینه در قطعهٔ اول کفایت می‌کند؟ پاسخ خود را شرح بدهید.

۳۷. فرض کنید در کلاس B به جای ۱۶ بیت برای شمارهٔ شبکه ، ۲۰ بیت در نظر گرفته می‌شد. در چنین حالتی چند شبکهٔ کلاس B می‌توان داشت؟

۳۸. آدرس IP با نمایش هگزادیسمال C22F1582 را به نماد نقطه‌دار تبدیل کنید.

۳۹. شبکه‌ای در اینترنت، از الگوی زیرشبکه 255.255.240.0 استفاده کرده است. این شبکه حداکثر چند ماشین میزبان می‌تواند داشته باشد؟

۴۰. تعداد بسیار زیادی آدرس IP متوالی از نقطهٔ شروع 198.16.0.0 در اختیار می‌باشد. فرض کنید چهار سازمان A و B و C و D به ترتیب ۴۰۰۰، ۲۰۰۰، ۴۰۰۰ و ۸۰۰۰ آدرس IP درخواست می‌کنند. برای هر یک از اینها اولین آدرس IP ، آخرین آدرس IP و الگوی زیرشبکه را به شکل w.x.y.z/s ، معین کنید.

۴۱. یک مسیریاب آدرسهای IP جدیدی طبق فهرست زیر دریافت می‌کند (جهت درج در جدول مسیریابی):

57.6.120.0/21 و 57.6.112.0/21 و 57.6.104.0/21 و 57.6.96.0/21

اگر برای رسیدن به تمام این شبکه‌ها از خط مشترکی استفاده شود، آیا می‌توان این آدرسها را «تجمیع»

(Aggregate) کرد؟ اگر می‌توان به چه نحو؟ اگر خیر، چرا؟

۴۲. مجموعه‌ای از آدرسهای IP از 29.18.0.0 تا 29.18.128.255 به صورت 29.18.0.0/17 «تجمیع» شده‌اند. ولیکن یک فاصلهٔ خالی ۱۰۲۴ تایی منتسب نشده در محدودهٔ 29.18.60.0 تا 29.18.63.255 وجود داشته که به ناگاه به شبکه‌ای اختصاص داده می‌شود که خط خروجی رسیدن به آن، با بقیه فرق می‌کند. آیا در اینجا باید فضای تجمیع شدهٔ آدرسها را مثل اول به بلوکهای اولیه تقسیم کرد و بلوک جدید به جدول اضافه و فرآیند تجمیع از نو انجام گیرد؟ اگر نه چه کار دیگری می‌توان انجام داد؟

۴۳. یک مسیریاب در جدول مسیریابی خود درایه‌های CIDR زیر را در اختیار دارد:

Address/mask (آدرس/الگوی زیرشبکه)	Next Hop (گام بعدی)
135.46.56.0/22	Interface 0
135.46.60.0/22	Interface 1
192.53.40.0/23	Router 1
default	Router 2

اگر بسته‌ای با یکی از آدرسهای IP زیر دریافت شود، مسیریاب با آن بسته چه می‌کند:

الف) 135.46.63.10
ب) 135.46.57.14
ج) 135.46.52.2
د) 192.53.40.7
ه) 192.53.56.7

۴۴. سیاست بسیاری از شرکتها بر آنست که دو (یا چند) مسیریاب متصل به اینترنت در شبکهٔ خود داشته باشند تا در صورت از کار افتادن یکی از آنها، از دیگری استفاده شود. آیا با چنین سیاستی باز هم می‌توان از NAT بهره گرفت؟ پاسخ خود را تشریح کنید.

۴۵. فرض نمائید پروتکل ARP را برای دوستتان توضیح داده‌اید. پس از اتمام توضیحاتتان او می‌گوید: «متوجه شدم: ARP سرویسی را به لایهٔ شبکه ارائه می‌دهد و طبعاً جزئی از لایهٔ پیوند داده‌ها است!» چه حرفی برای گفتن به او دارید؟

۴۶. ARP و RARP هر دو آدرسهایی را از یک فضا به فضایی دیگر می‌نگارند (IP به MAC و بالعکس). از این دیدگاه هر دو مشابه یکدیگر هستند، ولی پیاده‌سازی آنها متفاوت از یکدیگر است. عمده‌ترین تفاوت آنها در چیست؟

۴۷. روشی را برای بازسازی قطعات یک بستهٔ IP در مقصد، معرفی و تشریح نمائید.

۴۸. اغلب الگوریتمهای بازسازی دیتاگرامهای IP ، دارای زمان‌سنج (تایمر) خاصی هستند تا در صورت از بین رفتن یک قطعه، بقیهٔ قطعات تا ابد در بافر نمانند. فرض کنید یک دیتاگرام به چهار قطعه تقسیم شده است. سه قطعهٔ اول سر موقع دریافت می‌شوند ولی قطعهٔ آخر با تأخیر مواجه می‌شود. عاقبت مهلت دریافت آن منقضی شده و سه قطعهٔ دیگر از حافظهٔ گیرنده ، پاک می‌گردد. اندکی بعد، آخرین قطعه از راه می‌رسد. گیرنده با آن چه باید بکند؟

۴۹. چه در IP و چه در ATM ، کد کشف خطای checksum فقط سرآیند بسته را در بر می‌گیرد نه بخش داده را. فکر می‌کنید منطق این طراحی چه بوده است؟

۵۰. شخصی که در بوستون زندگی می‌کند در سفری به مینیاپولیس کامپیوتر کیفی خود را به همراه می‌برد. خوشبختانه، شبکهٔ LAN او در مینیاپولیس یک شبکهٔ محلی بی‌سیم و مبتنی بر IP است و او مجبور به وصل

کابل به جایی نیست. آیا برای آنکه ترافیک پست الکترونیکی یا دیگر داده‌ها به درستی تحویل او شود، کماکان لازم است که از دو عامل خانگی و خارجی استفاده شود؟

۵۱. آدرسها در IPv6 شانزده بایتی هستند. اگر در هر پیکوثانیه یک بلوک یک میلیون تایی آدرس IP رزرو شود چه مدت طول می‌کشد تا آدرسها تمام شوند؟

۵۲. فیلد پروتکل که در IPv4 وجود داشت اکنون در سرآیند ثابت IPv6 وجود ندارد. به چه دلیل؟

۵۳. آیا وقتی که پروتکل IPv6 به صحنهٔ عمل آمد نیازی به تغییر در ARP هست؟ اگر این چنین است آیا این تغییرات در عملکرد و مفهوم است یا در پیاده‌سازی فنی؟

۵۴. برنامه‌ای بنویسید که فرآیند مسیریابی به روش سیل‌آسا را شبیه‌سازی نماید. هر بسته دارای یک فیلد شمارنده است که به ازای هر گام (Hop) یک واحد از آن کم می‌شود. وقتی این شمارنده به صفر برسد، بسته حذف می‌گردد. زمان را «گسسته» (Discrete) فرض کنید و در هر واحد زمان فقط یک بسته بر روی خط ارسال یا دریافت می‌شود. برنامهٔ خود را در سه نسخه تهیه کنید: (۱) بستهٔ ورودی بر روی تمام خطوط ارسال گردد. (۲) بستهٔ ورودی بر روی تمام خطوط به استثنای خطی که از آن داخل شده، فرستاده شود. (۳) بستهٔ ورودی بر روی k تا از بهترین خطوط (به صورت آماری و غیردقیق) ارسال گردد. روش سیل‌آسا را با روش معمولی (یعنی وقتی بستهٔ ورودی بر روی بهترین خط خروجی ارسال می‌شود) از لحاظ پهنای باند و تأخیر مقایسه نمائید.

۵۵. برنامه‌ای بنویسید که یک شبکهٔ کامپیوتری را به صورت زمان گسسته شبیه‌سازی کند. در هر واحد زمان، بسته‌ای که سر صفّ هر یک از خطوط خروجی مسیریابها قرار گرفته‌اند، ارسال و یک گام جلو می‌روند. هر مسیریاب فضای بافر محدودی دارد. اگر بسته‌ای دریافت شود ولی جایی برای آن وجود نداشته باشد، حذف شده و از نو ارسال نخواهد شد. در عوض یک پروتکل انتها به انتها (End to End) وجود دارد که در صورت ارسال داده‌ای که دریافت آن تصدیق نشده از مبداء ، داده‌ها را مجدداً ارسال می‌کند. توان مفید (ظرفیت مفید یا Throughput) شبکه را برحسب تابعی از زمان انقضای مهلت (Timeout) و پارامتر نرخ خطا ترسیم نمائید.

۵۶. تابعی بنویسید که عملیات هدایت (Forwarding) را در یک مسیریاب مبتنی بر IP انجام بدهد. این تابع یک پارامتر ورودی دارد و آن هم آدرس IP است. فرض کنید این تابع به یک جدول مسیریابی سراسری دسترسی دارد. این جدول آرایه‌ای است که سه ستون دارد و محتویات هر ستون، اعداد صحیح هستند. این ستونها عبارتند از: آدرس IP ، الگوی زیرشبکه (Subnet Mask) و شمارهٔ خط خروجی مورد استفاده [برای رسیدن به شبکه‌ای با آدرس IP متناظر]. این تابع، باید آدرس IP دریافتی از پارامتر ورودی را در جدول فوق‌الذکر به روش CIDR جستجو کرده و شمارهٔ خط خروجی متناسب با آن را به عنوان مقدار برگشتی باز گرداند.

۵۷. با استفاده از فرامین اجرایی traceroute (در یونیکس) یا tracert (در ویندوز)، مسیر رسیدن از کامپیوتر خودتان به چند دانشگاه در دیگر قاره‌ها را پیدا کنید. برخی از سایتهای متعلق به دانشگاههای معروف که می‌توانید در بررسی خود از آنها استفاده کنید، عبارتند از:

www.berkeley.edu	(کالیفرنیا)
www.mit.edu	(ماساچوست)
www.vu.nl	(آمستردام)
www.ucl.ac.uk	(لندن)
www.usyd.edu.au	(سیدنی)
www.u-tokyo.ac.jp	(توکیو)
www.uct.ac.za	(کیپ تاون)

۶ لایه انتقال

لایهٔ انتقال (Transport Layer) ، فقط در یک لایه خلاصه نمی‌شود بلکه قلب تپندهٔ سلسلهٔ پروتکلهای شبکه است. وظیفهٔ این لایه آن است که داده‌ها را به روشی قابل اعتماد و کم هزینه از ماشین مبداء به ماشین مقصد انتقال بدهد، فارغ از آن که ماهیت شبکه یا شبکه‌های فیزیکی مورد استفاده چیست. بدون لایهٔ انتقال مفهوم پروتکلهای لایه‌ای، معنای حقیقی خود را پیدا نخواهند کرد. در این فصل جزئیات لایهٔ انتقال را شامل خدماتی که عرضه می‌کند، طراحی آن، پروتکلهای مرتبط و کارآیی آنها را بررسی خواهیم کرد.

۶-۱ خدمات انتقال (The Transport Service)

در بخشهای آتی، خدماتی را که لایهٔ انتقال ارائه می‌نماید، اجمالاً بررسی کرده و به انواع سرویسهایی که به لایهٔ کاربرد عرضه می‌شود نگاهی خواهیم انداخت. برای آن که حقیقت خدمات انتقال را آشکارتر کرده باشیم دو مجموعه از «**عملکردهای اولیه**» (Primitives) برای لایهٔ انتقال تعریف کرده‌ایم: مجموعهٔ اول عملکردهای ساده و فرضی هستند که به فهم ایدهٔ اصلی کمک می‌کند. سپس مجموعه‌ای از واسطها (Interfaces) را که عموماً در اینترنت بکار گرفته می‌شوند، معرفی خواهیم کرد.

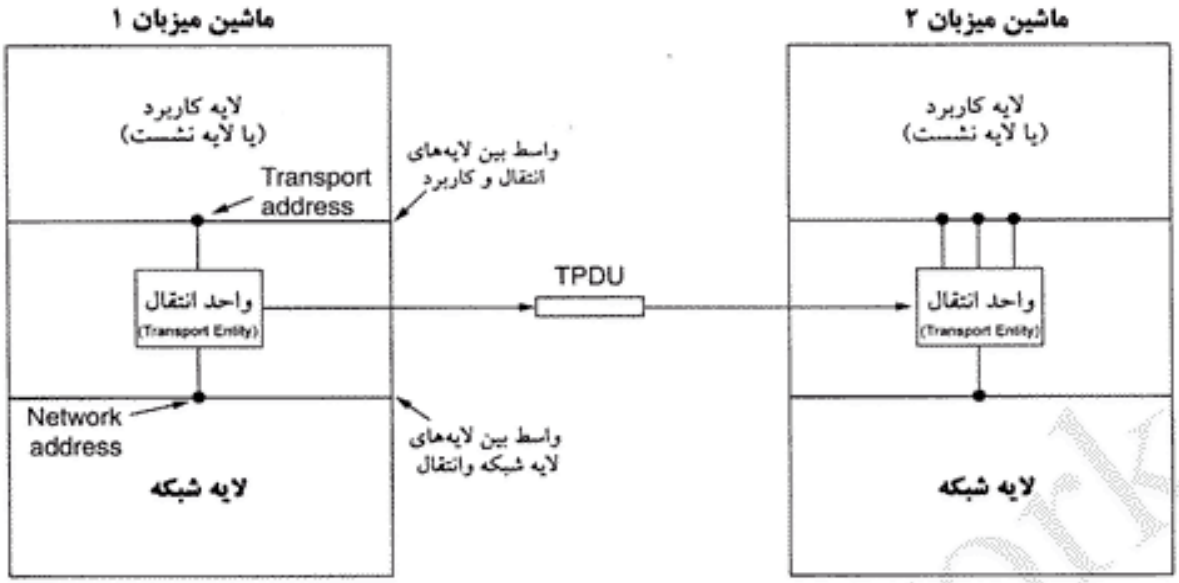
۶-۱-۱ خدمات ارائه شده به لایه‌های بالاتر

هدف نهایی لایهٔ انتقال آن است که سرویسی کارآمد، مطمئن و کم هزینه به کاربران خود بدهد. کاربران لایهٔ انتقال، پروسه‌های لایهٔ کاربرد (یا به عبارتی برنامه‌های کاربردی) هستند. برای نائل آمدن به این هدف، لایهٔ انتقال از خدماتی که لایهٔ شبکه عرضه کرده، بهره می‌گیرد. نرم‌افزار یا سخت‌افزاری که در لایهٔ انتقال این عملیات را انجام می‌دهد اصطلاحاً «**واحد انتقال**» (Transport Entity) نامیده می‌شود. واحد انتقال ممکن است درون هستهٔ سیستم عامل یا به صورت یک پروسهٔ کاربری مجزا یا در قالب یک بستهٔ کتابخانه‌ای[1] در بطن برنامه‌های کاربردی شبکه و یا حتی در درون کارت واسط شبکه تعبیه شده باشد. در شکل ۶-۱ ارتباط منطقی بین لایه‌های شبکه، انتقال و کاربرد به تصویر کشیده شده است.

همانگونه که خدمات لایهٔ شبکه بر دو نوع اتصال‌گرا و بدون‌اتصال ارائه می‌شود، خدمات لایهٔ انتقال نیز بر همین دو نوع است. خدمات مبتنی بر اتصال (هماهنگیهای قبلی) در لایهٔ انتقال از بسیاری جهات مشابه با خدمات اتصال‌گرای لایهٔ شبکه است. در هر دو مورد، «**اتصال**» سه مرحلهٔ تکوین دارد: «ایجاد اتصال» (Establishment)،

---

۱. Library Package



شکل ۶-۱. لایه‌های شبکه، انتقال و کاربرد.

«انتقال داده» (Data Transfer)، «ختم اتصال» (Release). آدرس‌دهی و کنترل جریان نیز در هر دو لایه مشابه هم هستند. مضاف بر این، خدمات بدون‌اتصال در لایهٔ انتقال نیز شباهت بسیار زیادی به خدمات بدون‌اتصال در لایهٔ شبکه دارد.

در اینجا یک سؤال بدیهی به ذهن خطور می‌کند: اگر خدمات لایهٔ انتقال این قدر به خدمات لایهٔ شبکه شبیه است، پس چرا این دو لایه از هم جدا هستند؟ چرا فقط به یکی از آنها بسنده نمی‌شود؟ پاسخ این سؤال سرراست ولی بنیادی است و از شکل ۱-۹ نشأت می‌گیرد. کد اجرایی لایهٔ انتقال کلاً بر روی ماشین کاربران اجرا می‌شود در حالی که لایهٔ شبکه اغلب بر روی مسیریابها اجرا می‌گردد که آنها نیز تحت مدیریت یک «حامل» (Carrier) هستند (حداقل در WAN اینگونه است). اگر لایهٔ شبکه خدماتی ناکافی عرضه کند (که اغلب اینگونه است) چه اتفاقی می‌افتد؟ اگر لایهٔ شبکه (به دلائلی مثل ازدحام، بروز حلقه یا پایان عمر بسته) تعدادی از بسته‌ها را از دست بدهد چه باید کرد؟ اگر یک مسیریاب هر از گاهی از کار بیفتد چه می‌شود؟

در موارد فوق فقط می‌توان گفت که مشکلات جدّی رخ می‌دهد! کاربران هیچ کنترل واقعی بر لایهٔ شبکه ندارند و طبعاً نمی‌توانند مشکلات ناشی از خدمات ناقص مسیریابها را مثلاً با تعویض مسیریاب یا مدیریت بهتر خطا در لایهٔ پیوند داده برطرف کنند. تنها راه ممکن آن است که بر روی لایهٔ شبکه، لایهٔ دیگری قرار داده شود تا کیفیت خدمات را بهبود بدهد. در یک زیر شبکهٔ اتصال‌گرا، اگر «**واحد انتقال**» (Transport Entity) در حین انتقال طولانی مدت بسته‌ها، به ناگاه متوجه شود که اتصال شبکه قطع شده و از سرنوشت داده‌های در حال انتقال نیز بی‌خبر باشد، می‌تواند یک اتصال جدید با «واحد انتقال راه دور» (Remote Transport Entity) برقرار نماید؛ به کمک اتصال جدید، واحد انتقال می‌تواند از همتای خود سؤال کند که کدام بخش از داده‌ها دریافت کرده و کدام بخش را دریافت نکرده و از جایی که داده‌ها نرسیده‌اند، شروع به ارسال مجدد نماید.

در حقیقت، وجود لایهٔ انتقال این امکان را فراهم آورده تا خدمات انتقال داده‌ها، قابل اعتمادتر از خدمات لایهٔ زیرین یعنی لایهٔ شبکه باشد. بسته‌های گمشده یا تکراری توسط لایهٔ انتقال کشف می‌شوند. مضاف بر اینها، عملکردهای اولیهٔ لایهٔ انتقال می‌تواند به صورت فراخوانی توابع کتابخانه‌ای پیاده‌سازی و در اختیار پروسه‌های لایهٔ بالاتر گذاشته شود تا لایهٔ بالاتر از عملکردهای اولیه و توابع پایهٔ لایهٔ شبکه (و درگیری با جزئیات این لایه که

امکانات ضعیفی نیز ارائه می‌کند) مستقل باشد زیرا خدماتی که لایهٔ شبکه عرضه می‌نماید در شبکه‌های متفاوت می‌تواند تفاوتهای بنیادی داشته باشد (مثلاً خدمات یک شبکهٔ بدون‌اتصال با خدمات اتصال‌گرای یک WAN تفاوت اساسی و ماهوی دارد). پنهان کردن خدمات لایهٔ شبکه در پشت یک مجموعه از «عملکردها و توابع اولیه» موجب می‌شود که در صورت تغییر در خدمات لایهٔ شبکه بتوان با عوض کردن توابع کتابخانه‌ای (به گونه‌ای که همان خدمات و عملکرد قبلی خود را در شبکهٔ جدید ارائه کنند)، این تغییر را در سطح همین لایه محدود نگاه داشت، [بدین ترتیب هیچ یک از اجزاء لایه‌های بالاتر از چنین تغییری آگاه نخواهند شد.]

با تکیه بر لایهٔ انتقال، برنامه‌نویسان نرم افزارهای کاربردی می‌توانند کد برنامهٔ خود را مبتنی بر یک مجموعهٔ استاندارد از توابع و عملکردهای اولیه بنویسند و هیچگونه نگرانی از بابت تفاوت در اجزاء و مکانیزمهای زیر شبکه یا انتقال نامطمئن و غیرقابل اعتماد نداشته باشند. اگر تمام شبکه‌های واقعی، بدون اشکال و قابل اعتماد بودند و همهٔ آنها خدمات مشابهی ارائه می‌کردند و این تضمین وجود می‌داشت که هیچگاه تغییر نکنند، آنگاه به خدمات لایهٔ انتقال نیازی نبود. ولیکن [به دلیل اختلافات بنیانی و تنوع بسیار زیاد لایه‌های زیرین] در دنیای واقعی، وجود این لایه، لایه‌های بالایی را از درگیری با جزئیات تکنولوژی، طراحی و نواقص زیر شبکه دور نگه می‌دارد.

به همین دلیل بسیاری از افراد، لایه‌های یک تا چهار را در یک دسته و لایه‌های بالاتر از ۴ را در دسته‌ای دیگر قرار می‌دهند. چهار لایهٔ اول را می‌توان «**ارائه دهندهٔ خدمات انتقال**» (Transport Service Provider) تصور کرد در حالی که بقیهٔ لایه‌های فوقانی، «**استفاده‌کنندگان از خدمات انتقال**» (Transport Service User) محسوب می‌شوند. تفکیک بین لایه‌های ارائه دهندهٔ خدمات و لایه‌های استفاده‌کنندهٔ از این خدمات، نقش لایهٔ انتقال را حساس و کلیدی کرده است چرا که این لایه در مرز بین این دو دسته قرار گرفته و باید به لایه‌های فوقانی خدمات انتقال مطمئن داده‌ها را ارائه بدهد.

۶-۱-۲ عملکردهای اولیه و توابع بنیانی لایهٔ انتقال

برای آن که امکان دسترسی به خدمات لایهٔ انتقال برای استفاده‌کنندگان این خدمات فراهم شود، این لایه باید در قالب یک «**واسط خدمات انتقال**» (Transport Service Interface) مجموعه‌ای از عملیات و توابع را در اختیار برنامه‌های کاربردی بگذارد.[1] هر رده از خدمات لایهٔ انتقال، «واسط» مختص به خود را دارد. در این بخش خدمات ساده و فرضی لایهٔ انتقال و واسط آن را بررسی می‌کنیم تا با اصول پایه آشنا شویم. سپس در بخشهای آتی، مروری بر مثالهای واقعی خواهیم داشت.

خدمات لایهٔ انتقال در عین شباهت با خدمات لایهٔ شبکه، تفاوتهای مهم دارند: تفاوت عمدهٔ آنها در این است که لایهٔ شبکه خدمات شبکهٔ واقعی و در حال استفاده را مدل می‌کند و شبکه‌های واقعی به دلایل مختلفی ممکن است بسته‌ها را از دست بدهند فلذا خدمات لایهٔ شبکه عموماً قابل اعتماد نیستند.

در عوض خدمات انتقال اتصال‌گرا در لایهٔ انتقال کاملاً قابل اعتماد است. در حالی که شبکه‌های واقعی مصون از خطا نیستند، لایهٔ انتقال بر روی این شبکهٔ نامطمئن قرار می‌گیرد و با مکانیزمهای خاصی که بدان خواهیم پرداخت، تمام این خطاها و مشکلات را جبران و برطرف می‌نماید.

به عنوان مثال، دو پروسه را در نظر بگیرید که در محیط یونیکس از طریق یک «لوله» (Pipe) به یکدیگر مرتبط شده‌اند. این دو پروسه فرض را بر آن می‌گذارند که ارتباط آنها بی‌نقص و مصون از هر نوع خطایی است. این دو تمایلی ندارند که در خصوص مکانیزمهایی مثل تصدیق دریافت داده‌ها (Acknowledgement)، بسته‌هایی که از بین می‌روند، ازدحام یا مسائلی نظیر آن چیزی بدانند. آنچه که این پروسه‌ها انتظار دارند یک ارتباط صددرصد

---

۱. مفهوم «واسط یا interface» را همان مفهوم نرم‌افزاری آن در مدل برنامه‌نویسی در نظر بگیرید. ـم

مطمئن و مصون از خطاست. پروسهٔ A داده‌های خود را در انتهای «لوله» قرار می‌دهد و پروسهٔ B داده‌ها را از ابتدای این لوله بر می‌دارد. تمام آنچه که خدمات اتصال‌گرای لایهٔ انتقال باید ارائه بدهند چیزی شبیه به همین مفهوم است: یعنی پروسه‌های کاربری واقع بر هر نقطه از شبکه‌ای که نامطمئن است، بتوانند یک دنبالهٔ بیت مصون از خطا (Error Free Bit Stream) را ارسال یا دریافت کنند و نقایص زیر شبکه از چشم آنها پنهان بماند.

البته لایهٔ انتقال می‌تواند خدمات نامطمئن و بدون‌اتصال (از نوع دیتاگرام) نیز ارائه بدهد ولیکن حرف زیادی برای گفتن ندارد و تمرکز اصلی ما در این فصل بر روی خدمات اتصال‌گرا و قابل اعتماد در لایهٔ انتقال است. علیرغم آن، برخی از برنامه‌های کاربردی مثل عملیات چندرسانه‌ای، از مزایای انتقال بدون‌اتصال بهره گرفته‌اند لذا چند کلمه‌ای در خصوص آن صحبت خواهیم کرد.

تفاوت دیگر بین خدمات لایهٔ شبکه و لایهٔ انتقال آن است که استفاده‌کنندگان از خدمات آنها تفاوت بنیادی دارند. خدمات لایهٔ شبکه به «**واحد انتقال**» (Transport Entity) ارائه می‌شود. کاربران بسیار کمی هستند که بخواهند «واحد انتقال» اختصاصی برای خود بنویسند یا برنامه‌های آنها مستقیماً از خدمات حداقل و ناقص لایهٔ شبکه استفاده نماید. برعکس، اکثر برنامه‌ها (و طبعاً برنامه‌نویسان) فقط عملکردهای اولیه و توابع بنیانی لایهٔ انتقال را می‌بینند. در نتیجه، خدمات لایهٔ انتقال باید سهل‌الوصول و استفاده از آنها ساده و سرراست باشد.

برای آن که احساسی از ماهیت خدمات لایهٔ انتقال پیدا کنید به پنج عملکرد (تابع) اولیه (Primitives) که در جدول ۶-۲ فهرست شده، دقت کنید. این توابع که نقش یک «واسط انتقال» (Transport Interface) را ایفاء می‌کنند، حداقل نیازهای مورد انتظار از لایهٔ انتقال محسوب می‌شوند. این مجموعهٔ توابع، به برنامه‌های کاربردی امکان می‌دهد تا بتوانند اتصالاتی «ایجاد»، «استفاده» و نهایتاً آن را «ختم» نمایند. چنین امکانی برای اغلب برنامه‌های کاربردی کافی است.

نام عملکرد (تابع) اولیه	بسته ارسالی	توصیف
LISTEN	(none)	متوقف (بلوکه) می‌شود تا آنکه پروسه‌ای سعی در برقراری اتصال کند.
CONNECT	CONNECTION REQ.	بصورت فعال سعی در برقراری یک اتصال می‌کند.
SEND	DATA	داده می‌فرستد.
RECEIVE	(none)	متوقف می‌شود تا آنکه بسته داده برسد.
DISCONNECT	DISCONNECTION REQ.	تلاش برای قطع (خاتمه) اتصال

شکل ۶-۲. عملکردهای (توابع) اولیه برای ارائهٔ خدمات سادهٔ انتقال.

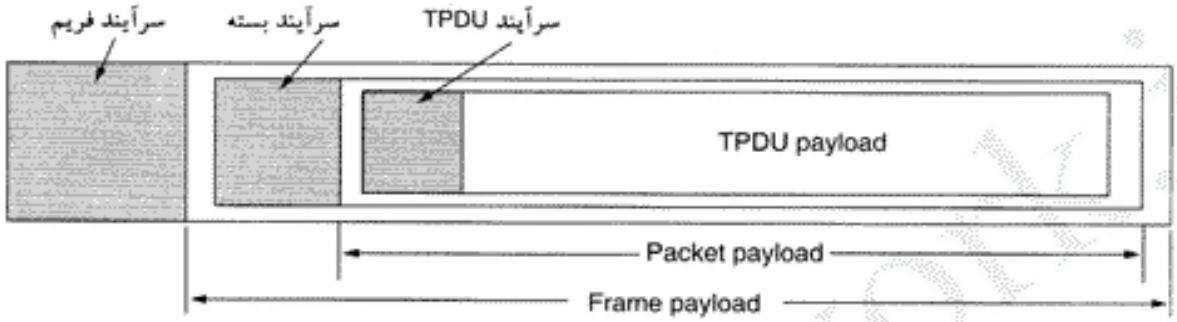
برای آن که ببینیم از این عملکردها و توابع چگونه استفاده می‌شود، یک برنامهٔ کاربردی شامل یک «سرویس‌دهنده» و تعدادی «مشتری» راه دور را مدّ نظر قرار بدهید. برای شروع، برنامهٔ سرویس‌دهنده با فراخوانی سیستمی، تابع LISTEN را اجرا می‌کند. اجرای این تابع سرویس‌دهنده را بلوکه کرده و به حالت انتظار می‌برد تا آن که یک «مشتری» ظاهر شود (بعبارت دیگر از راه دور تقاضای ارتباط کند). در طرف مقابل، وقتی مشتری می‌خواهد با سرویس‌دهنده محاوره کند، تابع CONNECT را اجرا می‌کند. «واحد انتقال» (Transport Entity) ضمن بلوکه کردن برنامهٔ مشتری، با ارسال بسته‌ای به سرویس‌دهنده این تقاضا را به اطلاع او می‌رساند. در درون فیلد داده از این بسته، پیام لایهٔ انتقال قرار گرفته که نهایتاً تحویل «واحد انتقال» خواهد شد.

در اینجا بایستی یک واژه را معرفی نمائیم: به دلیل فقدان واژهٔ بهتر، با بی‌میلی مجبوریم برای نامگذاری ساختار پیامهایی که بین «واحدهای انتقال» مبادله می‌شوند، از واژهٔ **TPDU**[1] استفاده نمائیم. طبعاً TPDU (که بین لایه‌های

---

۱. Transport Protocol Data Unit

انتقال مبادله خواهند شد) در درون «بستهٔ لایهٔ شبکه» [مثلاً درون یک بستهٔ IP] جاسازی و حمل می‌شود. خود «بسته» نیز در درون یک فریم جاسازی و توسط لایهٔ پیوند داده، مبادله می‌شود. وقتی فریمی دریافت گردد، ابتدا لایهٔ پیوند داده، سرآیند آن فریم را پردازش کرده و محتوای آن را به «واحد شبکه» (Network Entity) تسلیم می‌کند. واحد شبکه نیز سرآیند بسته را پردازش کرده و محتوای فیلد دادهٔ آن بسته را تحویل «واحد انتقال» در لایهٔ بالا می‌نماید. شکل ۶-۳، تودرتویی این بسته‌ها را به تصویر کشیده است.



شکل ۶-۳. تودرتویی TPDUها، بسته‌ها و فریمها.

به مثال برنامهٔ سرویس‌دهنده/مشتری خودمان برگردیم: فراخوانی تابع CONNECT در برنامهٔ مشتری موجب می‌شود که یک بستهٔ CONNECTION REQUEST TPDU به سوی سرویس‌دهنده، ارسال شود. هر گاه این بسته دریافت شود، واحد انتقال بررسی می‌کند که آیا سرویس‌دهنده‌ای با اجرای LISTEN بلوکه شده و منتظر است؟ (به عبارتی بررسی می‌کند که آیا پروسه‌ای علاقمند به پردازش و پاسخ به چنین تقاضایی هست یا خیر)؛ اگر چنین باشد، پروسهٔ مربوطه را از حالت بلوکه خارج کرده و در پاسخ، بستهٔ CONNECTION ACCEPTED TPDU به مشتری برگردانده می‌شود. وقتی این TPDU دریافت شود، برنامهٔ مشتری نیز از حالت بلوکه خارج شده و اتصال برقرار می‌شود.

پس از این مراحل، داده‌ها می‌توانند با استفاده از توابع SEND و RECEIVE بین دو برنامه مبادله شوند. در ساده‌ترین حالت، هر یک از طرفین می‌توانند به کمک تابع RECEIVE به حالت انتظار وارد شده و منتظر بمانند تا طرف مقابل با تابع SEND اقدام به ارسال داده نماید. وقتی این TPDU دریافت شد، گیرنده از حالت بلوکه خارج می‌گردد. این پروسه نیز TPDU مربوطه را دریافت و پاسخ لازم را برمی‌گرداند. مادامی که طرفین ارتباط به نوبت ارسال نمایند این الگو به خوبی کار می‌کند.

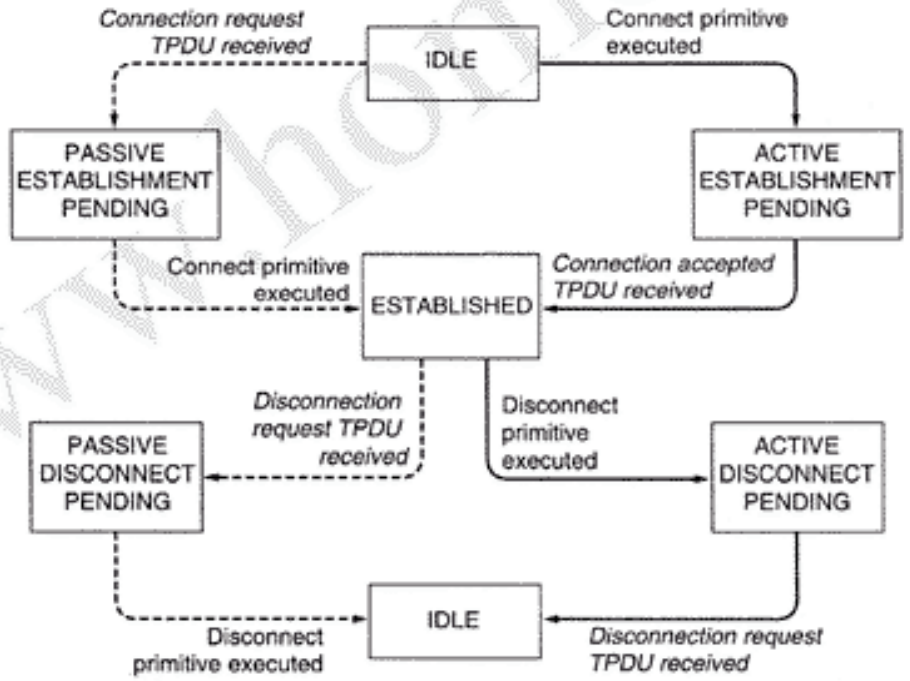
دقت کنید که لایهٔ انتقال حتی برای مبادله یکطرفه و سادهٔ داده‌ها بسیار پیچیده‌تر از لایهٔ شبکه عمل می‌کند. دریافت یکایک بسته‌هایی که ارسال می‌شوند باید به تایید طرف مقابل برسد؛ (با ارسال پیغام Ack). حتی بسته‌هایی که حامل TPDUهای کنترلی هستند نیز باید (به صورت مستقیم یا ضمنی) اعلام وصول شوند. تصدیق وصول بسته‌ها بر عهدهٔ «واحد انتقال» (Transport Entity) است و این کار از دید کاربران لایهٔ انتقال مخفی می‌ماند. بنابراین (طبق آنچه که در فصل سوم اشاره شد)، واحد انتقال باید نگران زمان‌سنجها (تایمرها) و ارسال مجدّد (Retransmission) داده‌ها باشد. هیچیک از این عملیات برای کاربران لایهٔ انتقال مشهود نیست. از دید کاربران لایهٔ انتقال، یک «اتصال» (Connection) به مثابهٔ یک «لولهٔ مطمئن انتقال بیت» (Bit Pipe) است: یکی از کاربران بیتها را در ابتدای این لوله تزریق می‌کند و به همان صورت در انتهای دیگر لوله، تحویل داده می‌شود. توانایی پنهان‌سازی پیچیدگیهای زیرین، پروتکلهای لایه‌ای را به یک ابزار قدرتمند تبدیل کرده است.

دیگر به یک اتصال نیازی نباشد باید آن را خاتمه داد تا فضای جدولی که در حافظهٔ هر کدام از

«واحدهای انتقال» به آن اختصاص داده شده، آزاد شود. قطع یک اتصال (Disconnection) به دو صورت ممکن است: نامتقارن و متقارن. در روش «**نامتقارن**» (Asymmetric) هر یک از پروسه‌های روبرو می‌توانند با صدور تابع پایهٔ DISCONNECT ، به صورت یکطرفه اقدام به ختم ارتباط نماید. صدور DISCONNECT موجب ارسال یک TPDU کنترلی خاص (DISCONNECT TPDU) به سوی «واحد انتقال» طرف مقابل می‌شود. پس از دریافت این TPDU ارتباط قطع می‌شود.

در روش «**متقارن**» (Symmetric) [چون ارتباط دوجهته -Bidirectional- است] هر یک از جهتهای انتقال به طور جداگانه بسته می‌شود: وقتی یکی از طرفین DISCONNECT می‌کند، منظورش آن است که دادهٔ دیگری برای ارسال ندارد ولیکن کماکان آمادهٔ پذیرش داده‌های شریک مقابل خود است. در این مدل، یک «اتصال» ، صرفاً زمانی بطور کامل قطع می‌شود که هر دو طرف DISCONNECT نمایند.

در شکل ۶-۴ یک «دیاگرام حالت»[1] برای فرآیند ایجاد و ختم اتصال (به کمک توابع اولیه) ترسیم شده است. «**گذار**»[2] از یک «**حالت**»[3] به حالت دیگر بر اثر بروز یک «**رخداد**»[4] اتفاق می‌افتد، خواه این رخداد در اثر صدور یک تابع اولیه در پروسهٔ محلی کاربر، حادث شود و خواه در اثر ورود یک بستهٔ کنترلی (مثل Connection Request) رخ بدهد. برای سادگی فرض می‌کنیم دریافت هر TPDU بطور مجزا تصدیق (Ack) شود.[5] همچنین فرض کرده‌ایم که قطع ارتباط به صورت متقارن انجام می‌شود و پروسهٔ مشتری زودتر تقاضای قطع اتصال می‌دهد. لطفاً دقت کنید که این مدل کاملاً ساده است. بعداً در خصوص مدلهای ختم ارتباط بیشتر توضیح می‌دهیم.



شکل ۶-۴. یک دیاگرام حالت برای الگوی مدیریت سادهٔ اتصال. «گذار از یک حالت» که با قلم ایتالیک نشان داده شده در اثر ورود یک بسته حادث می‌شود. خطوط توپر توالی حالات برنامهٔ مشتری و خطوط نقطه‌چین توالی حالات برنامهٔ سرویس دهنده را نشان می‌دهند.

---

۱. State Diagram ۲. Transition ۳. State ۴. Event

۵. یعنی از روش Piggybacking (جاسازی Ack درون بسته‌های دادهٔ ارسالی) که در فصل سوم تشریح کردیم استفاده نشده باشد. -م

۶-۱-۳ سوکتهای برکلی (Berkeley Socket)

حال اجازه بدهید به اختصار مجموعهٔ دیگری از عملکردهای اولیه و توابع بنیادی لایهٔ انتقال را که با نام «توابع سوکت» در یونیکس برکلی برای پروتکل TCP تعریف شده، بررسی نمائیم. از این توابع به طرز گسترده‌ای برای برنامه‌نویسی اینترنت استفاده می‌شود. این توابع در شکل ۶-۵ فهرست شده‌اند. در یک عبارت نادقیق، این توابع از همان مدل مثال قبلی ما پیروی می‌کنند با این تفاوت که ویژگیهای بهتر و انعطاف بیشتری دارند. فعلاً در اینجا به جزئیات ساختار TPDU در هر یک از این توابع نمی‌پردازیم و بررسی آن را تا تشریح TCP در همین فصل به تعویق می‌اندازیم.

نام عملکرد (تابع)	توصیف
SOCKET	یک نقطه ارتباط پایانی جدید ایجاد می‌کند.
BIND	به سوکت ایجاد شده یک آدرس محلی (شماره) مقید می‌کند.
LISTEN	تمایل برنامه کاربردی به پذیرش تقاضاهای اتصال را مشخص نموده و طول صف را معین می‌کند.
ACCEPT	فراخواننده را آنقدر متوقف و منتظر نگاه می‌دارد تا کسی سعی در ایجاد اتصال کند.
CONNECT	تلاش جهت ایجاد اتصال بصورت فعال
SEND	مقداری داده بر روی اتصال مشخص شده می‌فرستد.
RECEIVE	مقداری داده از اتصال مشخص‌شده می‌خواند.
CLOSE	ختم اتصال

شکل ۶-۵. توابع اولیهٔ سوکت برای TCP.

در هر برنامهٔ سرویس‌دهنده، چهار تابع پایهٔ جدول ۶-۵، به ترتیب اجرا می‌شوند. تابع SOCKET در پروسهٔ کاربردی یک نقطهٔ پایانی (End-Point) تعریف کرده و فضای حافظهٔ لازم را در واحد انتقال (Transport Entity) اختصاص می‌دهد.[1] پارامترهای لازم برای فراخوانی این تابع عبارتند از: (۱) قالب آدرس‌دهی مورد نظر (Addressing Format)، (۲) نوع خدمات مورد انتظار (مثل استریم مطمئن از بایتها یا سرویس نامطمئن دیتاگرام) و (۳) پروتکل مورد نظر.

اگر فراخوانی تابع SOCKET موفق باشد یک «**اشاره‌گر فایل**» به عنوان مقدار برگشتی بازگردانده می‌شود تا در فراخوانیهای بعدی مورد استفاده قرار گیرد؛ دقیقاً مشابه با همان کاری که تابع OPEN برای باز کردن یک فایل انجام می‌دهد.

سوکتهایی که ایجاد می‌شوند دارای آدرسهای شبکه نیستند.[2] برای انتساب آدرسها به هر سوکت از تابع پایهٔ BIND استفاده می‌شود. به محض آنکه سرویس‌دهنده، آدرسی را به یک سوکت نسبت داد هر مشتری راه دور می‌تواند اقدام به برقراری اتصال با آن کند.[3] دلیل آن که با فراخوانی تابع SOCKET ، آدرس لازم به آن انتساب داده نمی‌شود و تابعی مجزا برای آن در نظر گرفته شده، آن است که برخی از پروسه‌ها در مورد آدرس مورد نظر خود مطمئن هستند (مثلاً برای سالها از یک آدرس استفاده می‌نمایند و همهٔ مشتریان از این شماره آگاهند) در حالی که برای برخی دیگر از پروسه‌ها، مقدار این آدرس و ثبات آن اصلاً اهمیت ندارد.

---

۱. در حقیقت با تابع SOCKET هویت هر یک از طرفین در دو سر خط لولهٔ انتقال مشخص می‌شود. -م

۲. به عبارت دیگر، هر چند یک نقطهٔ پایانی در پروسه ایجاد می‌شود ولی هیچ آدرسی که بتوان از راه دور با آن اقدام به برقراری تماس کرد وجود ندارد. -م

۳. به فرآیند انتساب آدرس به یک سوکت عمل «مقیدسازی» سوکت ـ Binding ـ گفته می‌شود. -م

در ادامه، پروسهٔ سرویس‌دهنده تابع پایهٔ LISTEN را فراخوانی می‌کند. اجرای این تابع فضای لازم را برای صف‌بندی تقاضاهای ایجاد اتصال، اختصاص می‌دهد. بدین ترتیب سرویس‌دهنده می‌تواند بطور همزمان با چندین مشتری اتصال ایجاد کند. برخلاف مدل مثال اول، در اینجا فراخوانی تابع LISTEN پروسهٔ سرویس‌دهنده را بلوکه و معلق نخواهد کرد.

برای آن که پروسه تا دریافت تقاضای ایجاد اتصال، معلق و منتظر بماند سرویس‌دهنده، تابع ACCEPT را فراخوانی می‌کند. هر گاه یک TPDU مبنی بر تقاضای ایجاد یک اتصال دریافت شود، واحد انتقال سوکت جدیدی با همان مشخصات سوکت اصلی ایجاد کرده و اشاره‌گر آن را بر می‌گرداند. برنامهٔ سرویس‌دهنده می‌تواند یک پروسهٔ فرزند (Child Process) یا یک «ریسمان» (Tread) برای آن ایجاد کند تا به سرویس‌دهی به اتصال جدید مشغول شود. سپس مجدداً به حالت انتظار برگشته تا بتواند تقاضاهای اتصال جدید را بپذیرد. تابع ACCEPT یک اشاره‌گر معمولی برمی‌گرداند تا با استفاده از آن بتوان به روش معمولی داده‌ای خواند یا نوشت. (دقیقاً شبیه به عملیات خواندن و نوشتن از فایل که با اشاره‌گر مربوطه انجام می‌شود.)

حال اجازه بدهید نگاهی به پروسهٔ سمت مشتری بیندازیم: در اینجا نیز بایستی با فراخوانی تابع پایهٔ SOCKET ، یک سوکت ایجاد شود ولیکن نیازی به فراخوانی تابع BIND نیست چرا که در اینجا آدرس پروسهٔ مشتری برای سرویس‌دهنده اهمیتی ندارد.[1] سپس با فراخوانی تابع CONNECT ، پروسهٔ فراخواننده معلق شده و همان وقت فرآیند ایجاد اتصال آغاز می‌شود. وقتی این فرآیند تکمیل شود (یعنی وقتی پاسخ مناسب از سرویس‌دهنده دریافت گردد) پروسهٔ مشتری از حالت تعلیق به در آمده و اتصال مورد نظر ایجاد می‌شود. در این نقطه هر یک از طرفین این اتصال می‌توانند با فراخوانی توابع پایهٔ SEND و RECV اقدام به ارسال یا دریافت دوطرفه و همزمان (Full Duplex) داده‌ها بنمایند. در یونیکس حتی می‌توان از فراخوانیهای سیستمی READ و WRITE به جای SEND و RECV استفاده کرد.

مدل قطع ارتباط در سوکتهای فوق، متقارن است یعنی زمانی که هر دو طرف، تابع CLOSE را اجرا کنند، اتصال ایجاد شده خاتمه خواهد یافت.

۶-۱-٤ مثالی از برنامه‌نویسی سوکت: یک سرویس‌دهندهٔ اینترنتی فایل

به عنوان مثالی از چگونگی فراخوانی توابع سوکت، کدهای برنامهٔ سرویس‌دهنده و مشتری را در شکل ۶-۶ مدنظر قرار بدهید. در این مثال یک سرویس‌دهندهٔ ابتدایی فایل به همراه یک برنامهٔ مشتری (که از آن بهره می‌گیرد) ، ارائه شده است. این قطعه کُد از محدودیتهای بی‌شماری که در زیر تشریح کرده‌ایم، رنج می‌برد ولیکن در هر حال می‌توان کد برنامهٔ سرویس‌دهنده را کامپایل کرده و آن را بر روی هر سیستم یونیکس متصل به اینترنت اجرا کرد. کد برنامهٔ مشتری را نیز می‌توان پس از کامپایل، بر روی هر ماشین یونیکس متصل به اینترنت در سراسر دنیا اجرا و از آن استفاده کرد. برنامهٔ مشتری در خط فرمان (به همراه دو آرگومان) اجرا می‌شود تا بکمک آن هر فایلی را که برنامهٔ سرویس‌دهنده بدانها دسترسی دارد، دریافت کرد. فایل دریافتی بر روی «خروجی استاندارد» نوشته می‌شود و بدیهی است که می‌توان به کمک مفهوم «**تغییر مسیر**» (Redirection) در یونیکس، آنرا به درون یک فایل یا یک «لوله» (Pipe) هدایت کرد.

اجازه بدهید به کد برنامهٔ سرویس‌دهنده نگاهی بیندازیم. این برنامه با تعریف چند include file[2] شروع می‌شود. سه فایل آخر، در برگیرنده تعاریف مرتبط با اینترنت و ساختمان دادهٔ مورد نیاز شبکه است. سپس ثابت

---

۱. دقت کنید که پروسهٔ مشتری قطعاً دارای آدرس هست ولی فقط مقدار آن مهم نیست و در اینجا انتخاب آن بر عهدهٔ لایهٔ انتقال گذاشته شده است. -م

۲. مفهوم #include را در زبان C ، در نظر داشته باشید.

SERVER-PORT معادل با 12345 تعریف شده است. این عدد کاملاً اختیاری است: اعداد بین ۱۰۲۴ تا ۶۵۵۳۵ (به شرط آن که توسط پروسهٔ دیگری استفاده نشده باشد) قابل انتخاب است. البته برنامه سرویس‌دهنده و مشتری باید از شمارهٔ مشابهی استفاده کنند. اگر روزی این سرویس‌دهنده جهانی شود باید به آن یک شمارهٔ پورت زیر ۱۰۲۴ انتساب داده شود و در سایت www.iana.com به اطلاع همه برسد!!

دو خط بعدی از برنامهٔ سرویس‌دهنده، دو ثابت مورد نیاز برنامه را تعریف کرده است. اولین آنها حداکثر طول قطعات ارسالی فایل را مشخص نموده است. دومین ثابت، تعیین می‌کند که حداکثر چند اتصال معلق و منتظر را می‌توان حفظ کرد.

پس از تعریف متغیرهای محلی، کد برنامهٔ سرویس‌دهنده آغاز می‌شود. در ابتدا یک ساختمان داده با آدرس IP ماشین سرویس‌دهنده ، مقداردهی اولیه می‌شود. در ادامه، این ساختمان داده به سوکت سرویس‌دهنده، «**مقید**» (Bind) خواهد شد. فراخوانی تابع *memset* کلّ ساختمان داده را با صفر پر می‌کند. سپس سه دستور انتساب بعدی فیلدهای این ساختمان داده را مقداردهی می‌نمایند. آخرین دستور انتساب، شمارهٔ پورت سرویس‌دهنده را مشخص کرده است. توابع htonl و htons مقادیر فیلدها را به قالب استاندارد تبدیل می‌کند تا این برنامه بتواند بدرستی بر روی ماشینهای Big Endian (مثل ماشینهای SPARC) و ماشینهای Little Endian (مثل ماشینهای پنتیوم) اجرا شود.[1]

در ادامه، برنامهٔ سرویس‌دهنده سوکتی را ایجاد کرده و خطای احتمالی را بررسی می‌نماید. (هر گونه خطا با $s<0$ مشخص می‌شود.) در نسخهٔ واقعی این برنامه، پیامهای خطا می‌توانند گویاتر و مفصل‌تر باشند. فراخوانی تابع *setsockopt* بدان جهت نیاز است که بتوان از آن پورت به دفعات استفاده کرد و برنامهٔ سرویس‌دهنده بتواند به صورت نامحدود اجرا شود. حال به سوکت ایجاد شده آدرس IP و پورت، مقید (Bind) شده و بررسی می‌شود که آیا این عمل موفق بوده است. آخرین گام در مقداردهی اولیه، فراخوانی تابع *listen* است تا تمایل سرویس‌دهنده به پذیرش تقاضاهای اتصال را به اطلاع سیستم رسانده و مشخص کند که سیستم فقط حق دارد به تعداد QUEUE_SIZE از متقاضیان اتصال را (در حین پردازش یکی از آنها) پذیرفته و معلق نگه دارد. اگر صف مربوطه پر باشد و تقاضای اتصال جدیدی برسد، نادیده گرفته می‌شود.

در این نقطه، برنامهٔ سرویس‌دهنده به حلقهٔ اصلی خود وارد می‌شود؛ حلقه‌ای که هرگز از آن خارج نخواهد شد. تنها راه متوقف کردن این برنامه، «کُشتن» آن از بیرون (با فرمان kill) است. فراخوانی تابع accept ، سرویس‌دهنده را بلوکه می‌کند تا آنکه یک یا چند مشتری سعی کنند با آن اتصالی را برقرار نمایند. اگر فراخوانی تابع accept موفق باشد، یک اشاره‌گر معمولی فایل، بازگردانده می‌شود تا به کمک آن بتوان داده ارسال یا دریافت کرد. (به همان نحوی که با اشاره‌گر فایل می‌توان درون یک لوله (Pipe) نوشت یا از آن خواند.) ولیکن برخلاف «لوله» در یونیکس که یک طرفه است، سوکتها دوطرفه هستند لذا با در اختیار داشتن آدرس سوکت (یعنی متغیر *sa*) می‌توان بر روی اتصال ایجاد شده نوشت (معادل ارسال) یا از آن خواند (معادل دریافت).

پس از آن که اتصال ایجاد شد، برنامهٔ سرویس‌دهنده نام فایل مورد نظر مشتری را از روی آن اتصال می‌خواند. اگر داده‌ای دریافت نشده باشد، سرویس‌دهنده بلوکه شده و منتظر می‌ماند. پس از دریافت نام فایل مورد نظر مشتری، سرویس‌دهنده، فایل مربوطه را باز کرده و در حلقهٔ دیگری وارد می‌شود تا متوالیاً بلوکهای فایل را خوانده و آن را بر روی سوکت ارسال نماید. این حلقه آنقدر ادامه می‌یابد تا کل فایل منتقل شود. سپس فایل و اتصال متناظر را می‌بندد و منتظر اتصال بعدی می‌ماند. این حلقه تا ابد ادامه دارد.

---

۱. برای آشنایی با تعریف این واژه‌ها به فصول قبلی مراجعه کنید.

```
/* This page contains a client program that can request a file from the server program
 * on the next page. The server responds by sending the whole file.
 */

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345                /* arbitrary, but client & server must agree */
#define BUF_SIZE 4096                    /* block transfer size */

int main(int argc, char **argv)
{
  int c, s, bytes;
  char buf[BUF_SIZE];                    /* buffer for incoming file */
  struct hostent *h;                     /* info about server */
  struct sockaddr_in channel;            /* holds IP address */

  if (argc != 3) fatal("Usage: client server-name file-name");
  h = gethostbyname(argv[1]);            /* look up host's IP address */
  if (!h) fatal("gethostbyname failed");

  s = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
  if (s <0) fatal("socket");
  memset(&channel, 0, sizeof(channel));
  channel.sin_family= AF_INET;
  memcpy(&channel.sin_addr.s_addr, h->h_addr, h->h_length);
  channel.sin_port= htons(SERVER_PORT);

  c = connect(s, (struct sockaddr *) &channel, sizeof(channel));
  if (c < 0) fatal("connect failed");

  /* Connection is now established. Send file name including 0 byte at end. */
  write(s, argv[2], strlen(argv[2])+1);

  /* Go get the file and write it to standard output. */
  while (1) {
      bytes = read(s, buf, BUF_SIZE);    /* read from socket */
      if (bytes <= 0) exit(0);           /* check for end of file */
      write(1, buf, bytes);              /* write to standard output */
  }
}

fatal(char *string)
{
  printf("%s\n", string);
  exit(1);
}
```

شکل ۶-۶. کُد برنامهٔ مشتری با بهره‌گیری از سوکتها. کُد برنامهٔ سرویس‌دهنده در صفحهٔ بعدی آمده است.

```
#include <sys/types.h>                  /* This is the server code */
#include <sys/fcntl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345               /* arbitrary, but client & server must agree */
#define BUF_SIZE 4096                   /* block transfer size */
#define QUEUE_SIZE 10

int main(int argc, char *argv[])
{
  int s, b, l, fd, sa, bytes, on = 1;
  char buf[BUF_SIZE];                   /* buffer for outgoing file */
  struct sockaddr_in channel;           /* holds IP address */

  /* Build address structure to bind to socket. */
  memset(&channel, 0, sizeof(channel));  /* zero channel */
  channel.sin_family = AF_INET;
  channel.sin_addr.s_addr = htonl(INADDR_ANY);
  channel.sin_port = htons(SERVER_PORT);

  /* Passive open. Wait for connection. */
  s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);   /* create socket */
  if (s < 0) fatal("socket failed");
  setsockopt(s, SOL_SOCKET, SO_REUSEADDR, (char *) &on, sizeof(on));

  b = bind(s, (struct sockaddr *) &channel, sizeof(channel));
  if (b < 0) fatal("bind failed");

  l = listen(s, QUEUE_SIZE);            /* specify queue size */
  if (l < 0) fatal("listen failed");

  /* Socket is now set up and bound. Wait for connection and process it. */
  while (1) {
      sa = accept(s, 0, 0);             /* block for connection request */
      if (sa < 0) fatal("accept failed");

      read(sa, buf, BUF_SIZE);          /* read file name from socket */

      /* Get and return the file. */
      fd = open(buf, O_RDONLY);         /* open the file to be sent back */
      if (fd < 0) fatal("open failed");

      while (1) {
          bytes = read(fd, buf, BUF_SIZE); /* read from file */
          if (bytes <= 0) break;        /* check for end of file */
          write(sa, buf, bytes);        /* write bytes to socket */
      }
      close(fd);                        /* close file */
      close(sa);                        /* close connection */
  }
}
```