به نام خدا

دانشگاه صنعتی بیرجند

گروه مهندسی کامپیوتر و فناوری اطلاعات

رشته مهندسی فناوری اطلاعات

جزوه درس شبکه های کامپیوتری ۲

وحیده بابائیان

vahidebabaiyan@yahoo.com

آدرس وبلاگ:

vahidebabaiyan.blogfa.com

منابع:

۱. اسلاید های آموزشی دانشگاه صنعتی شریف ، دکتر پاکروان

۲. شبکه های کامپیوتری، اندرو.اس. تنن بام، ترجمه دکتر پدرام، احسان ملکیان.

۳. اصول مهندسی اینترنت، احسان ملکیان

۴. شبکه های کامپیوتری، ابوالفضل طرقی حقیقت، انتشارات پارسه

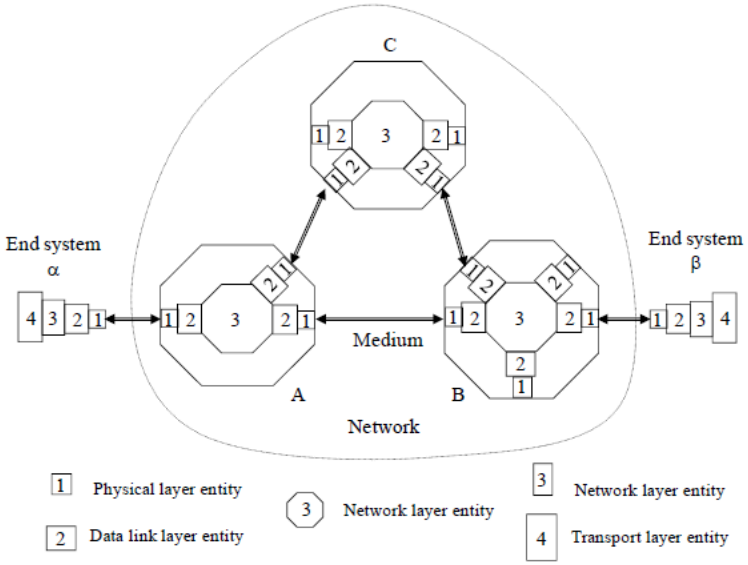
لایه شبکه

لایه شبکه چیست؟

عملکرد آن چگونه است؟

وظایف خود را چگونه انجام می دهد؟

در شکل زیر اگر از سیستم α به سیستم β یک مجموعه از بسته ها بخواهند جابه جا شوند، از یک شبکه ساده که متشکل از A, B, C است باید عبور کنند. پکت ها وقتی به A می رسند، دو راه دارند (یکی به C یکی به B). مهم ترین وظیفه لایه شبکه روتینگ یا مسیریابی است. در اینجا هر نود مثل نود A بر اساس اطلاعاتی که در این فصل با آن آشنا می شوید، باید تصمیم بگیرد که یک بسته بهتر است به B برود یا C. هر نود دیگر مثل C و B نیز همانند A باید تصمیم بگیرند که ترافیک های ورودی خود را از کدام مسیر خارج کنند. منظور از مسیریابی این است که پکت ها را از یک سر که مبداء ارسال است به سر دیگر که مقصد دریافت پکت ها است برساند.



در شکل فوق لایه فیزیکی، لایه پیوند داده، لایه شبکه و لایه انتقال به ترتیب با اعداد ۱ و ۲ و۳ و ۴ مشخص شده اند. همانطور که در شکل مشخص شده است، دو سر هر لینک باید لایه ۱ و ۲ و ۳ را داشته باشد. حوزه دید لایه ۱ و ۲ محدود به دو سر یک لینک می شود. اما در مورد لایه ۳، سر و ته کل مسیر مهم می باشد.

نقش لایه شبکه: (مسیریابی پکت ها)

سروکار داشتن با ارتباطات انتها به انتها: هر نود تصمیم می گیرد تا پکت ها از چه مسیرهایی بروند تا نهایتاً از مبداء به مقصد برسند. برای همین ارتباط انتها به انتها در لایه ۳ خیلی مهم است.

آگاهی از توپولوژی شبکه: وقتی بسته بخواهد از مبداء به مقصد برسد، واضح است که باید با توپولوژی و همبندی شبکه درگیر باشیم.

انتخاب مسیر مناسب: انتخاب مسیری که با توجه به نیازمندی ها و شرایط موجود، بهترین نتایج را به دنبال داشته باشد.

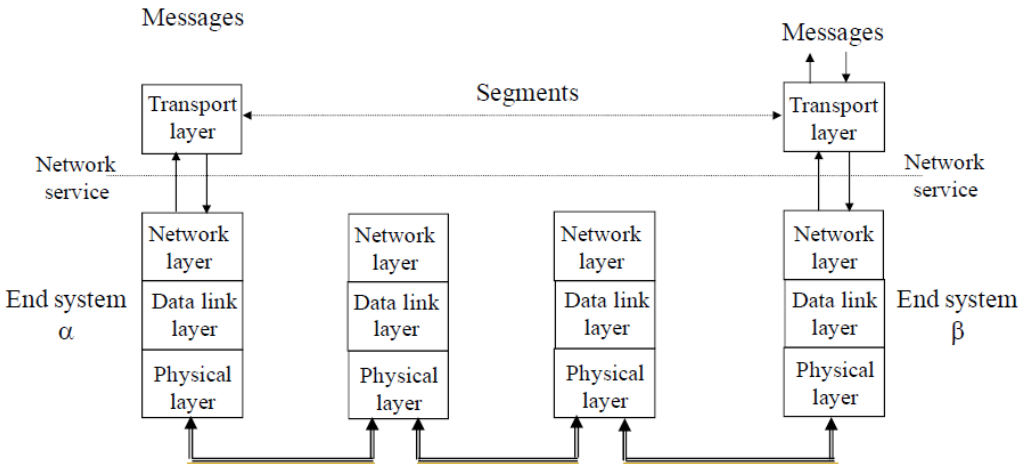
ارتباط داشتن با لایه انتقال و پیوند داده: در واقع وظیفه لایه شبکه سرویس دهی به لایه انتقال است و برای انجام وظایف خود از لایه پیوند داده سرویس می گیرد.

مجموعه مباحث مطرح شده در لایه شبکه:

- ✓ طراحی روتینگ های مناسب
- ✓ الگوریتم های مطرح شده
- ✓ کنترل ازدحام
- ✓ کیفیت سرویس
- ✓ ارتباط بین شبکه های مختلف در دنیای واقعی چگونه است؟
- ✓ پیاده سازی لایه شبکه در اینترنت چگونه است؟

سرویس های ارائه شده به لایه انتقال توسط لایه شبکه:

لایه انتقال مبداء هنگامی که می خواهد داده های خود را به لایه انتقال مقصد برساند و یک ارتباط انتها به انتها را بوجود آورد از لایه ۳ (لایه شبکه) سرویس هایی را می گیرد. لایه ۴ داده های خود را تحویل لایه ۳ می دهد و به جزئیات پیاده سازی و توپولوژی شبکه کاری ندارد. لایه ۳ توپولوژی شبکه را از دید لایه های بالایی خود می پوشاند چرا که توپولوژی شبکه، تعداد نودها و تعداد زیرشبکه ها ربطی به لایه ۴ و لایه های بالاتر از آن ندارند. مکانیزم شناسایی نودها بصورت مشخص بایستی در لایه ۳ مطرح گردد تا بتواند داده های دریافتی از لایه ۴ را در مسیرهای مناسبی از نودها به سمت مقصد هدایت کند. در لایه ۳ نوعی شماره گذاری داریم که مبداء و مقصد ها را مشخص می کند که در واقع آدرس یکتایی از نودهای شبکه فراهم می کند و به آن آدری IP می گوییم که در ادامه مفصل بحث خواهد شد.



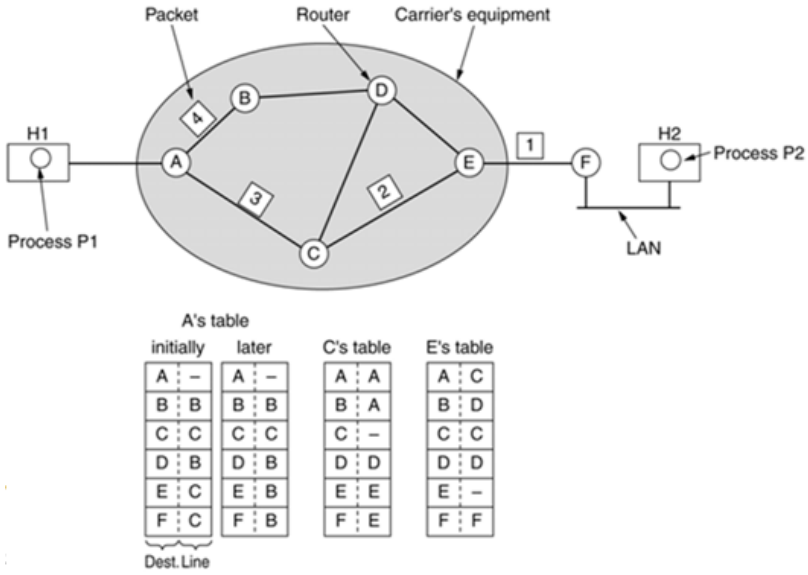
پس سرویس های لایه شبکه به لایه انتقال عبارتند از:

- ✓ مستقل ساختن لایه ۴ از تکنولوژی زیرشبکه
- ✓ مخفی ساختن توپولوژی شبکه، تعداد نودها و تعداد زیرشبکه ها از دید لایه ۴
- ✓ شماره گذاری منحصر به فرد بین شبکه های LAN و WAN .

از گذشته دو طرز تفکر در طراحی روتینگ های لایه شبکه وجود داشته است. برای اینکه بدانیم بهترین مکانیزم مسیریابی چیست باید ابتدا دو مفهوم سرویس اتصال گرا و غیر اتصال گرا (Connection-oriented or connectionless) را بررسی کنیم.

مدل Connectionless:

ایده روش: هر نود یک سری پورت ورودی و خروجی دارد. مثلاً در شکل زیر وقتی بسته ای وارد نود A شد، نگاه می کند که بسته از کجا و به چه مقصدی ارسال شده است. مثلاً از H1 به H2. سپس بهترین راهی که به نظر می رسد را انتخاب کرده و بسته را در آن مسیر قرار می دهد. در این مثال A سه بسته ۱و۲و۳ را از مسیر پایین (C->E) می فرستد. و به علت شلوغی لینک پایین بسته ۴ را از مسیر بالا (B->D) ارسال می کند.



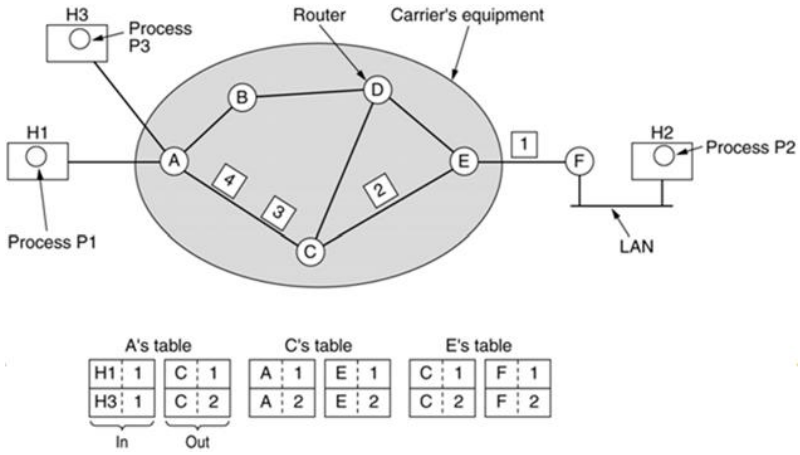
در این روش از قبل، امکان مشخص کردن مسیر برای ارسال بسته ها وجود ندارد و برای جابه جایی بسته ها هیچ اقدام اولیه یا Initialization لازم نیست. در این مدل روترها یا نودهایی که در شبکه نقش روتر را انجام می دهند برای مسیریابی از جدول هایی به نام جدول روتینگ استفاده می کنند. این که این جداول چگونه تنظیم می شوند مربوط به بحث هایی است که در ادامه مطرح خواهیم کرد. ولی به هر حال در این مدل برای روتینگ هر نود با توجه به پورت های خروجی خود و آدرس مقصد درج شده در بسته تصمیم می گیرد که هر بسته را به کدام پورت خروجی خود بفرستد. مثلاً در شکل فوق در ستون اول جدول هر نود مثل نود A دو ستون داریم. در ستون اول آدرس مقصد و در ستون دوم پورت خروجی مشخص می شود. مثلاً نود A مطابق جدول فوق، متوجه می شود که برای ارسال بسته ها به مقصدهای B و C و D و E و F باید از پورت هایی استفاده کند که به ترتیب به نود B و C و B و C و C متصل است. و هر موقع به هر دلیلی مسیر را به روز کرد در ستون دوم شماره پورت خود را به روز می نماید. مثلاً در اینجا ابتدا برای هدایت بسته ها به E و F از نود C استفاده می شد اما بعداً از نود B استفاده شده است.

مثال واقعی از این نوع سرویس دهی **اینترنت** می باشد. در اینترنت شبکه غیر قابل اطمینان داریم. و این مسئله در لایه های بالاتر کنترل خواهد شد. در اینترنت فقط ارسال و دریافت پکت ها لازم است. به علت اینکه سرعت خیلی مهم تر از دقت در این لایه است، با اینکه نودها در شبکه توانایی کنترل پیچیدگی برای فراهم آوردن سرویس مطمئن را دارند، این وظیفه در لایه ۴ گذاشته شده است.

در این مدل هر بسته آدرس مبدا و مقصد دارد و گره های میانی با نگاه به آدرس ها مسیر را انتخاب می کنند و تصمیم گیری هر نود بصورت مستقل صورت می گیرد.

مدل Connection-oriented:

ابتدا شبکه براساس تقاضای مبداء در شبکه بررسی می کند و مسیری را انتخاب می نماید. سپس به مبداء اجازه ارسال داده هایش را می دهد. از این به بعد مبداء در شرایط عادی برای انتقال بسته هایش از این مسیر از پیش تعیین شده استفاده می نماید. مگر اینکه لینک موجود خراب شود که در این صورت شبکه بایستی یک مسیر دیگر را جایگزین مسیر خراب شده قرار دهد. بنابراین در این روش پس از Initialization مسیر ثابتی برای انتقال بسته ها خواهیم داشت.



در شکل فوق بسته ای که از H1 به A می رسد، شناسه ای به همراه خود دارد. شناسه موجود در بسته مشخص می کند که بسته باید به کدام پورت خروجی A تحویل داده و از آنجا خارج گردد. جداول مسیریابی در این روش بدین صورت است که هر نود یا مسیر یاب یک جدول چهار ستونه دارند. در دو ستون اول به ترتیب پورت ورودی و شناسه و دو ستون دوم پورت خروجی و شناسه دیگری مشخص می شود. مثلاً در جدول مسیریابی گره A مشخص شده است که بسته هایی که از پورت متصل به H1 وارد می شوند و شناسه ۱ دارند، بایستی به پورتی ارسال شوند که متصل به C است. در ضمن هر مسیریاب مثل A پس از دریافت بسته و قبل از ارسال آن به پورت خروجی مناسب شناسه آن را مطابق ستون چهارم در جدول مسیریابی تغییر داده و آن گاه آن را ارسال می کند. که در اینجا مجدداً عدد ۱ را به عنوان شناسه برای بسته در نظر می گیرد. همینطور مطابق قبل، مسیریاب A بسته هایی که از پورت متصل به H3 را که شناسه ۱ دارند به پورت متصل به C با شناسه جدید ۲ ارسال خواهند کرد. در این مدل شناسه یا ID جزئی از سرایند بسته است که در هر hop سرایند تغییر می کند.

مثال واقعی از این مدل سرویس دهی، **سیستم تلفن** است که ابتدا بایستی یک فرایند برپاسازی ارتباط انجام شود. بدین صورت که هر ارتباط تلفنی یک شماره منحصر به فرد دارد و در مورد پارامترهای کیفی از قبیل کیفیت، هزینه و ... از قبل مذاکره می شود.

در این مدل یک Connection set داریم. یعنی در مسیریاب های A, B , C همه چیز از قبل مشخص است. و هماهنگی کامل بین روترها صورت گرفته است.

مدارمجازی و دیتاگرام (Virtual Circuits & Datagrams):

مدارمجازی:

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

خصوصیات روش VC را می توان به صورت زیر خلاصه کرد:

- برای ارسال بسته های اطلاعاتی به آدرس های جهانی مبدأ و مقصد نیازی نیست بلکه فقط به شماره VC نیاز است.
- برای هدایت بسته های اطلاعاتی از مبدأ به سمت مقصد نیاز به اجرای الگوریتم مسیریابی به ازای تک تک بسته ها نیست بلکه فقط یک جستجو در جدول هر مسیریاب انجام می شود.
- بسته ها الزاماً به ترتیب به مقصد خواهند رسید.
- احتمال گم شدن بسته ها ناشی از اشتباه در عمل مسیریابی در شبکه وجود ندارد.

دیتاگرام

در روش دیتاگرام هر ماشین میزبان پس از آنکه بسته ای را تولید کرد تحویل اولین مسیریاب در دسترس می دهد. مسیریاب ها مختارند بر اساس شرایط ترافیکی و توپولوژیکی زیرساخت ارتباطی شبکه ، مسیری را برای آن بسته انتخاب کرده و آن بسته را روی آن مسیر ارسال نمایند. بنابراین هیچ مسیر ثابت و از قبل مشخصی برای بسته های متوالی وجود ندارد. یعنی وقتی دو بسته از یک مبدأ تولید و به سمت یک مقصدِ واحد ارسال می شود ممکن است مسیرهای متفاوتی را طی نمایند؛ در ضمن ممکن است بسته ها به ترتیبی که تولید می شوند به مقصد نرسند.
خصوصیات روش دیتاگرام را می توان به صورت زیر خلاصه کرد:

- هر بسته ی اطلاعاتی به آدرس های جهانی مبدأ و مقصد نیازمند است.
- برای هر بسته باید مسیریابی جداگانه انجام شود.
- توزیع و هدایت بسته ها روی مسیرهای متفاوت، بر اساس شرایط توپولوژیکی و ترافیکی لحظه ای شبکه خواهد بود.
- چون بسته ها به ترتیب نمی رسند باید فرآیندی برای تنظیم ترتیب بسته ها در لایه های بالاتر صورت گیرد.
- در لایه بالاتر باید نظارت های ویژه بر گم شدن یا دوبله شدن بسته ها انجام شود.

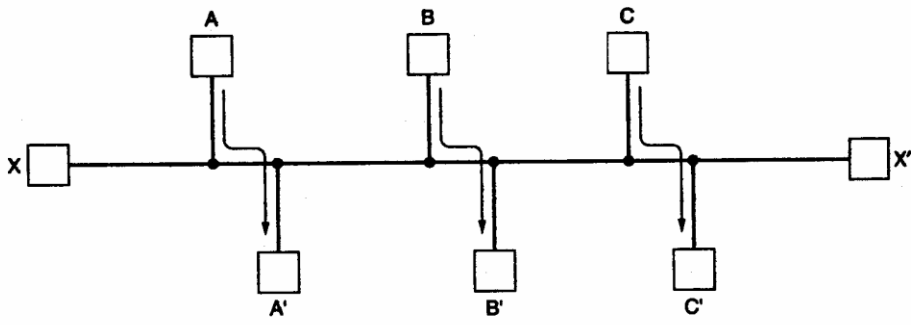
در جدول زیر مقایسه ای از زیرشبکه های مدارمجازی و دیتاگرام مطابق با مطالب ذکر شده بیان شده است.

مورد مقایسه	دیتاگرام	مدارمجازی
تنظیم مدار (Circuit Setup)	مسیریاب نیاز به نگهداری اطلاعات در خصوص وضعیت هر اتصال ندارد.	به ازای هر مدار مجازی تمامی مسیریابها باید اطلاعاتی در خصوص وضعیت آن را نگه دارند (برای تضمین QOS)
آدرس‌دهی	براساس آدرسهای مبدا و مقصد است.	بسته‌ها براساس یک شماره ID مخصوص به VC آدرس‌دهی می‌شوند.
مسیریابی (Routing)	بصورت پویا برای هر بسته مستقلاً انجام می‌شود.	فقط یکبار و آن هم در فاز برقرار اتصال و برپاسازی مدار مجازی انجام شده و همه بسته‌های آن اتصال از آن مسیر هدایت می‌شوند.
تاثیر خرابی مسیریاب	فقط بسته‌هایی خراب می‌شوند که در حافظه مسیریاب خراب در آن در لحظه بار شده بودند.	همه مدارهای مجازی که از مسیریاب خراب عبور می‌کرده‌اند قطع می‌شوند.
تضمین QOS (کیفیت سرویس)	بسیار دشوار است. (مطالب اضافه‌تر در سایت IETF موجود می‌باشد)	در فاز برقراری مدار مجازی یک مذاکره بین کاربر و شبکه انجام می‌شود و کاربر ملزومات QOS خود را اعلام می‌کند و چنانچه شبکه قادر باشد بدون ایجاد مشکلاتی مثل ازدحام آن معیارها را تحقق بخشد و تحقق آنها را تضمین نماید پس از رزرو منابع مورد نیاز، مدار مجازی را برقرار می‌کند و در غیراین صورت مکالمه را نمی‌پذیرد مگر اینکه کاربر توقع خود را کاهش دهد. به این فاز مذاکره CAC (Call Admission Control) می‌گویند.
کنترل ازدحام	بسیار دشوار است اما با مسیریابی پویا امکانپذیر است.	با تخصیص منابع شبکه در فاز CAC از ازدحام جلوگیری می‌شود.

الگوریتم های مسیریابی

هر یک از الگوریتم های مسیریابی به طور کلی 6 ویژگی زیر را باید داشته باشند.

۱. صحت عملکرد (Correctness): الگوریتم باید صحیح عمل کند. یعنی مطمئن باشیم یک بسته حتما از مبداء به مقصد می رسد.
۲. سادگی (Simplicity): الگوریتم باید ساده باشد و سربار اضافی بر شبکه تحمیل نکند.
۳. قابلیت تحمل (Robustness): خرابی سخت افزار و نرم افزار تاثیری بر عملکرد شبکه نگذارد (شبکه را از کار نیاندازد).
۴. پایداری (Stability): الگوریتم همگرا باشد زیرا اگر چنین شرطی وجود نداشته باشد در حلقه ابدی گرفتار خواهد شد. یا اگر تغییری رخ دهد، چقدر طول می کشد تا مجدداً به شرایط پایدار برسد.
۵. عدالت و مساوات (Fairness): منابع به صورت عادلانه تقسیم شوند.
۶. بهینه بودن (Optimality): برخی از این معیارها متاسفانه با هم در تضاد هستند مثلاً مساوات با بهینگی تضاد دارد و باید موازنه برقرار شود. در شکل زیر برای بهینگی باید ارتباط بین x , x' قطع باشد تا ۳ ارتباط دیگر برقرار شود ولی این با مساوات در تضاد است.



به عنوان مثالی در مقایسه مساوات و بهینگی شکل فوق را در نظر بگیرید. فرض کنید هر کدام از □ ها یک کامپیوتر و هر کدام از • ها نقش یک روتر را بازی کنند. اگر هر لینک ظرفیت مشخصی مثلاً 100 Mb/s داشته باشد، در ارسال داده از کامپیوتر A به A' ترافیک از ۳ لینک عبور می کند. برای انتقال داده از B به B' و از C به C' نیز چنین است. فرض کنید که کامپیوتر X برای ارسال 50 Mb داده به ظرفیت های شبکه موجود نیازمند است. دو راهکار برای پاسخگویی خواهیم داشت:

یک راهکار دستیابی به ماکسیموم Throughput یا توان گذردهی است که در این صورت فقط ترافیک کامپیوترهای A و B و C را عبور دادن است که در مجموع 300 Mb/s داده منتقل می شوند.

راهکار دیگر مبتنی بر عدالت است اما با بهینگی در تضاد می باشد. مثلاً ترافیک 75 Mb/s را کامپیوترهای A و B و C به مقصدهایشان برسانیم و از ظرفیت باقیمانده که 25 Mb/s است برای ارسال داده های X به X' استفاده کنیم.

انواع الگوریتم های مسیریابی

الگوریتم های مسیریابی را با دو دیدگاه می توان دسته بندی کرد:

۱. از دیدگاه روش تصمیم گیری و میزان هوشمندی الگوریتم
۲. از دیدگاه چگونگی جمع آوری و پردازش اطلاعاتِ زیرساخت ارتباطی شبکه

با دیدگاه اول الگوریتم های مسیریابی را می توان به دو دسته ی **ایستا** و **پویا** تقسیم بندی کرد.

در الگوریتم های **ایستا** هیچ اعتنایی به شرایط توپولوژیکی و ترافیک لحظه ای شبکه نمی شود. معمولاً در این الگوریتم ها برای هدایت یک بسته، هر مسیریاب از جداولی استفاده می کند که در هنگام برپایی شبکه تنظیم شده و در طول زمان ثابت است. در هنگام وقوع هرگونه تغییر در توپولوژی زیرساختِ شبکه، این جداول باید توسط مسئول شبکه بصورت دستی مجدداً تنظیم شود. اگرچه این الگوریتم ها بسیار سریعند ولی چون ترافیک لحظه ای شبکه متغیر است، نمی توانند بهترین مسیرها را انتخاب نمایند و هرگونه تغییر در توپولوژی زیرساخت ارتباطی شبکه، یک مشکل عمده و جدی ایجاد خواهد کرد.

در الگوریتم های **پویا** مسیریابی بر اساس آخرین وضعیت توپولوژیکی و ترافیک شبکه انجام می شود. جداول مسیریابی در این نوع الگوریتم ها هر T ثانیه یکبار به هنگام می شود. این الگوریتم ها بر اساس وضعیت فعلی شبکه تصمیم گیری می نمایند ولی ممکن است پیچیدگی این الگوریتم ها به قدری زیاد باشد که زمان تصمیم گیری برای انتخاب بهترین مسیر، طولانی شده و منجر به تاخیرهای بحرانی شده و نهایتاً به ازدحام بیانجامد. از انواع شاخص ها در تصمیم گیری یا مسیریابی می توان به حداقل Hop Count، حداقل Distance، حداقل Traffic و حداقل Congestion نام برد.

با دیدگاه دوم الگوریتم های مسیریابی را می توان به سه دسته ی (Centralized) **متمرکز**، (Distributed) **توزیع شده** و (Hieratical) **سلسله مراتبی** تقسیم کرد:

در الگوریتم های **متمرکز** اطلاعات وضعیت شبکه مانند توپولوژی و میزان ترافیک جاری در نقاط مختلف شبکه همگی در یک جا در درون هر مسیریاب متمرکز می شوند و هر مسیریاب کل اطلاعات شبکه را در اختیار دارد و تصمیم گیری به صورت متمرکز و براساس اطلاعات کامل و سراسری انجام می شود. هر مسیریاب باید اطلاعات کاملی از زیرساخت ارتباطی شبکه داشته باشد. یعنی هر مسیریاب باید تمامی مسیریاب های

دیگر، ارتباطات بین آن ها و هزینهٔ هر خط را دقیقاً شناسایی نماید. سپس با جمع آوری این اطلاعات، گراف زیرساخت شبکه را تشکیل بدهد. در چنین شرایطی برای یافتن بهترین مسیر بین هر دو مسیریاب، از الگوریتم های کوتاهترین مسیر نظیر الگوریتم دایجکسترا که در ادامه بحث خواهد شد، استفاده می شود. به چنین الگوریتم هایی که برای مسیریابی به اطلاعات کاملی از زیرساخت شبکه و هزینهٔ ارتباط بین هر دو مسیریاب نیازمندند ، اختصاراً الگوریتم های LS: Link State گفته می شود.

اما در الگوریتم های مسیریابی **توزیع شده** تصمیم گیری به صورت توزیعی است و اطلاعات وضعیت شبکه بر روی مسیریاب های مختلف توزیع شده است و تصمیم گیری یا اجرای الگوریتم نیز به صورت غیر متمرکز و براساس اطلاعات ناقص محلی انجام می شود. مسیریاب اطلاعات کاملی از زیرساخت شبکه ندارد بلکه فقط قادر است هزینهٔ ارتباط با مسیریاب هایی که بطور مستقیم و فیزیکی با آن ها در ارتباط است محاسبه و ارزیابی نماید. سپس در فواصل زمانی منظم، هر مسیریاب جدول مسیریابی خود را برای مسیریاب های مجاور، ارسال می نماید. مسیریاب با دریافت این جداول و مقادیری که خودش مستقیماً اندازه گیری کرده، با یک الگوریتم بسیار ساده جدول خودش را به هنگام می نماید و برای هدایت هر بسته، از آن استفاده می کند. در این الگوریتم ها برای مسیریابی هر بسته، فقط یک جستجو در جدول مسیریابی کافی است و در نتیجه پیچیدگی زمانی بسیار مناسبی دارد چراکه درگیر اجرای الگوریتم های وقت گیری شبیه دایجکسترا نخواهند شد. به این نوع الگوریتم ها به اختصار DV: Distance Vector گفته می شود.

در روش **سلسله مراتبی** برای جلوگیری از بزرگ شدن بیش از حد جداول مسیریابی کل یک شبکه بسیار بزرگ را به تعدادی ناحیه (region) تقسیم می کنیم. هر مسیریاب فقط اطلاعات مسیریابی مربوط به ناحیه خود را دارد ولی چیزی در خصوص جزئیات و ساختار داخلی دیگر نواحی ندارد. البته در شبکه های عظیم سلسله مراتب از دو سطح هم بیشتر است.

این روش ها را با تفصیل بیشتری بررسی می کنیم ولی قبل از آن یکی از روش های ایستا را که در برخی از موارد خاص کاربرد دارد، معرفی می نماییم.

الگوریتم های ایستا:

الگوریتم سیل آسا (Flooding)

این روش که برای ارسال بسته های همگانی کاربرد دارد و جزء *روتینگ های ایستا* در شبکه محسوب می شو، سریع ترین الگوریتم برای ارسال اطلاعات به یک مقصد در شبکه نیز به شمار می رود. طریقه ی ارسال در این روش آنست که هر مسیریاب با دریافت اینگونه بسته ها موظف است آن را روی تمامی مسیرهای خروجی خود به غیر از مسیری که بسته را از آن دریافت کرده است ارسال نماید. در چنین حالتی این تضمین وجود دارد که اولاً هر بسته اطلاعاتی به تمامی مسیریاب های زیرشبکه خواهد رسید. ثانیاً هر بسته در سریع ترین زمان ممکن به مقصد می رسد. در این الگوریتم سیلی از بسته ها از مسیرهای مختلف در آن واحد به سمت مقصد(در واقع در همه جهات) ارسال می شوند. واضح است که در این الگوریتم بسته های تکراری از مسیرهای مختلف به کلیه گره ها خواهد رسید و تولید بسته های تکراری موجب ازدحام و اشباع شبکه خواهد شد. برای حل این مشکل پیشنهاداتی ارائه شده است:

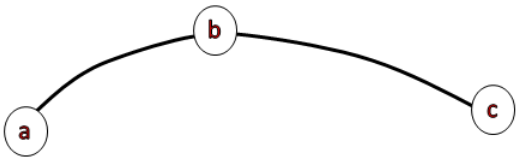
۱. یک شمارنده گام (Hop counter) داشته باشیم و در سرایند بسته قرار دهیم و در هر گام یک واحد از آن کم کنیم و پس از صفر شدن آن، بسته را دور بریزیم.
۲. قراردادن شماره برای بسته ها تا از ارسال مجدد بسته های تکراری جلوگیری کنیم.

موارد کاربردی این الگوریتم عبارتند از:

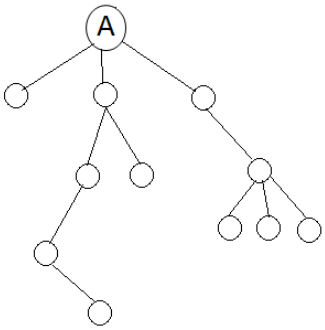
- Extreme robustness (military purpose): جایی که خیلی مهم است اطلاعات به مقصد برسند.
- Broadcast: جایی که پخش همگانی لازم داریم.
- Reference algorithm: با Flooding اولین بسته ای که از مبداء به مقصد می رسد، حتماً بهترین مسیر را طی کرده است. پس به عنوان مرجعی برای مقایسه با دیگر مسیرهای انتخابی از الگوریتم های دیگر می تواند خوب باشد.

اصل بهینگی مسیر (Optimality principle):

اگر در شبکه ای برای رفتن از نود a به c بهترین مسیر بهینه (بر اساس شاخص دلخواه مثل hop count, traffic , ...) به فرم زیر انتخاب شده باشد بطوریکه نود b نیز در مسیر قرار داشته باشد، آنگاه می توان نتیجه گرفت که مسیر b به c نیز بر اساس همان شاخص های قبلی مسیر بهینه می باشد.



با این فرض اگر یک توپولوژی داشته باشیم، در این توپولوژی می توانیم بگوییم برای هر گره مثل A یک درخت وجود دارد که این درخت بهترین مسیر از هر نود به A را نشان می دهد. و حتماً حلقه وجود ندارد. یعنی هیچ وقت از یک نود دو مسیر به A وجود ندارد. به این درخت sink tree می گوییم. درختی که مقصد ریشه درخت است. در واقع sink tree به ما بهترین مسیر از هر نود به ریشه درخت را نشان می دهد. روترها در شبکه می توانند با شناسایی sink tree و ارسال بسته ها در جهت آن، عمل مسیریابی را انجام دهند.



الگوریتم کوتاهترین مسیر (Shortest Path Algorithm):

هدف: پیدا کردن sink tree گره A

در این الگوریتم هر گره دارای یک برچسب دو قسمتی است که حاوی فاصله آن با گره مبدا و نام گره ایست که آن گره را به گره مبدا متصل می کند (با فاصله مذکور).

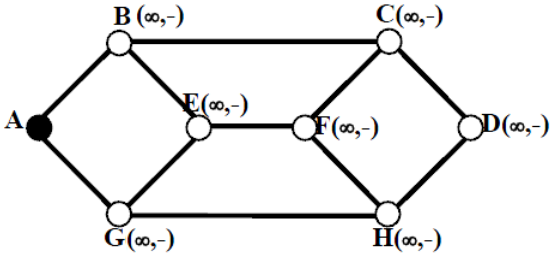
همچنین هر گره در طی پیشرفت الگوریتم یکی از دو وضعیت زیر را دارد:

- Tentative یا T یا موقتی
- Permanent یا P یا دائمی

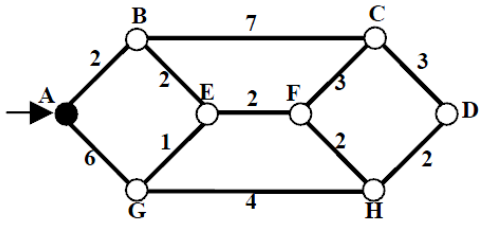
گره دائمی گره ایست که برچسب آن مطمئناً کوتاه ترین مسیر تا مبدأ را نشان می دهد.

برای درک راحتتر الگوریتم فوق به مثال شکل زیر دقت کنید. در این مثال فرض شده است می خواهیم بهترین مسیر از A به D را پیدا کنیم.

ابتدا برچسب همه گره ها تا مبدا را (- , ∞) قرار دهید. یعنی فاصله آن تا مبدا ∞ و از طریق گره نامشخص.

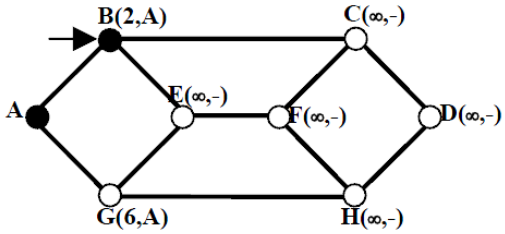


در مرحلهٔ 1 گراف زیرساخت ارتباطی یک شبکهٔ فرضی به همراه هزینهٔ هر ارتباط به تصویر کشیده شده است. گره شروع به عنوان نقطهٔ کار انتخاب شده و حالت آن بصورت" دائمی" علامت خورده است (گره های دائمی با دایرهٔ توپر نشان داده شده است).



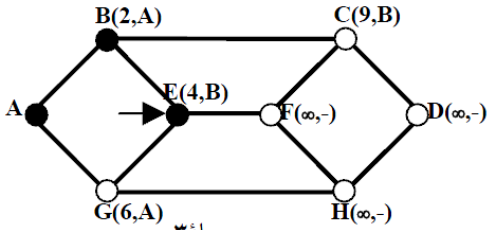
مرحلهٔ ۱

در مرحلهٔ 2 هزینهٔ گره های مجاور A یعنی B , G ، محاسبه شده و رکورد حالت آن ها اصلاح شده است. سپس از بین تمام گره های با حالت "موقت "، گره B که کمترین هزینه را تا A داشته به عنوان گره نقطهٔ کار انتخاب و حالت آن بصورت دائمی علامت زده شده است.



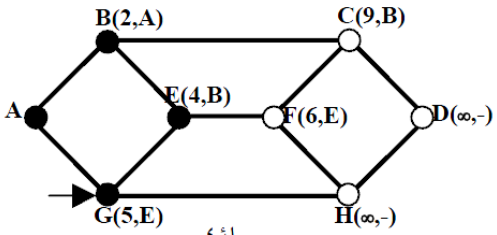
مرحلهٔ ۲

در مرحلهٔ 3 هزینهٔ گره های مجاور با گره های دائمی، تا A محاسبه شده و رکورد حالت آن ها اصلاح شده است. از بین تمام گره های با حالت "موقت "، گره E که کمترین هزینه تا A را داشته به عنوان گره نقطهٔ کار انتخاب و حالت آن بصورت دائمی علامت زده شده است.



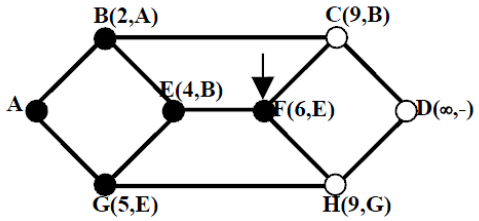
مرحلهٔ ۳

در مرحلهٔ 4 هزینهٔ گره های مجاور با گره های دائمی تا A محاسبه شده و رکورد حالت آن ها اصلاح شده است. (به چگونگی اصلاح رکورد حالت در گره G در این مرحله دقت کنید). از بین تمام گره های با حالت" موقت "، گره G که کمترین هزینه تا A را داشته به عنوان گره نقطهٔ کار انتخاب و حالت آن بصورت دائمی علامت زده شده است.



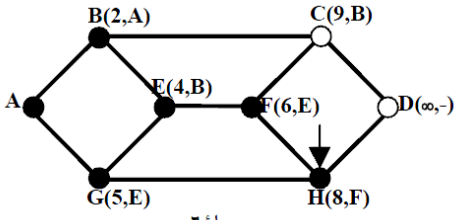
مرحلهٔ ٤

در مرحلهٔ 5 هزینهٔ گره های مجاور با گره های دائمی تا A محاسبه شده و رکورد حالت آن ها اصلاح شده است. از بین تمام گره های با حالت "موقت "، گره F که کمترین هزینه تا A را داشته به عنوان گره نقطهٔ کار انتخاب و حالت آن بصورت دائمی علامت زده شده است. (دقت کنید که F مجاور G نیست)



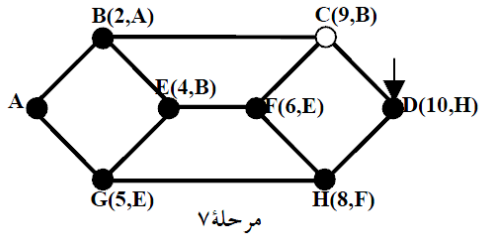
مرحلهٔ ۵

در مرحلهٔ 6 هزینهٔ گره های مجاور با گره های دائمی تا A محاسبه شده و رکورد حالت H اصلاح شده است. از بین تمام گره های با حالت "موقت "، گره H که کمترین هزینه تا A را داشته به عنوان گره نقطهٔ کار انتخاب و حالت آن بصورت دائمی علامت زده شده است.



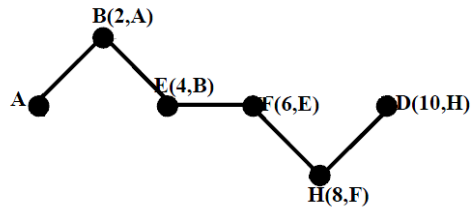
مرحلهٔ ٦

در مرحلهٔ 7 هزینهٔ گره های مجاور با گره های دائمی تا A محاسبه شده و رکورد حالت آن اصلاح شده است. از بین تمام گره های با حالت "موقت "، گره D که کمترین هزینه تا A را داشته به عنوان گره نقطهٔ کار انتخاب می شود. ولی چون D گره مقصد است، الگوریتم در این مرحله پایان می یابد.



مرحلهٔ ۷

برای پیدا کردن مسیر، از رکورد حالتِ گره D شروع کرده و گره قبلی آن را پیدا می کنیم، از این گره مجدداً گره قبلیِ آن پیدا می شود و این کار ادامه می یابد تا به نقطهٔ شروع برسیم.



برای ساخت Sink tree طبق الگوریتم shortest path کافی است توپولوژی و متریک های مورد انتظار مشخص باشد. نتیجه الگوریتم shortest path این است که هر نود برای ارسال بسته ها به مقصد مشخص می داند به چه سمتی بسته ها را ارسال کند.

الگوریتم های پویا:

مسیریابی بردار فاصله Distance vector routing:

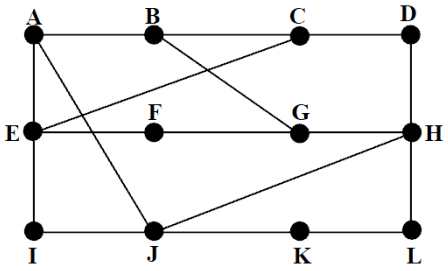
یکی از روش های پویا در مسیریابی، روش" بردار فاصله " یا DV است که در سال های اولیه راه اندازی شبکه ARPA مورد استفاده قرارگرفت و پس از عمومی شدن اینترنت تحت نام RIP عرضه شد و هنوز هم در مسیریاب های کوچک مورد استفاده قرار می گیرد. نام های متفاوتی برای این روش ارائه شده که همه آن ها از یک الگوریتم استفاده می کنند. این نام ها عبارتند از:

- پروتکل RIP (routing information protocol)
- الگوریتم مسیریابی Bellman–Ford
- الگوریتم مسیریابی Ford-Fulkerson
- الگوریتم Distance Vector Routing

در این روش هر مسیریاب بدون آنکه اطلاعی از هزینه خطوط ارتباطی در زیرشبکه داشته باشد، جدولی را در حافظه خود نگه می دارد که جدول مسیریابی نام دارد. در این جدول به ازای هر مسیریاب در زیرشبکه یک رکورد وجود دارد؛ هر رکورد دارای دو فیلد مجزا است:

- **فیلد مسیر:** این فیلد خط خروجی مناسب برای رسیدن به یک مسیریابِ خاص در شبکه را مشخص می کند.
- **فیلد مقدار تقریبی هزینه**: این فیلد هزینهٔ تقریبی رسیدن یک بسته تا مسیریابِ مقصد را مشخص می نماید.

برای روشن شدن قضیه به شکل زیر که زیرساخت ارتباطی یک شبکهٔ فرضی را نشان می دهد ، دقت کنید. در این مثال تعداد 12 مسیریاب با نام های A تا L زیرساخت ارتباطی شبکه را تشکیل داده اند؛ کانال هایی که بین این 12 مسیریاب وجود دارد در شکل مشخص است ولی هیچ یک از مسیریاب ها اطلاعی از هزینهٔ هر یک از خطوط ارتباطی ندارند، به همین دلیل مقادیر هزینهٔ هر خط در شکل نشان داده نشده است.



جدول زیر جدول مسیریابی مربوط به J است و هر سطر نشان می دهد که اگر J بخواهد بسته ای را به یک مسیریاب دیگر بفرستد از چه خطی باید استفاده کند. بعنوان مثال اگر مسیریاب J خواست برای G بسته ای ارسال کند، با مراجعه به این جدول نتیجه می گیرد که باید آنرا به سمتِ H ارسال نماید و هزینه رسیدن بسته به G تقریباً ۱۸ است.

جدول مسیریابی مربوط به مسیریاب J

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

حال شاید بپرسید در این روش معیار هزینه و واحد آن چیست؟ پاسخ آنست که معیار هزینه می تواند تاخیر یا " تعداد گام" (Hop) در نظر گرفته شود؛ در چنین حالاتی واحد تاخیر، میلی ثانیه و واحد گام ،" تعداد " خواهد بود. در این مثال معیار هزینه ، زمانِ تاخیر بر حسب میلی ثانیه انتخاب شده است.

تا اینجا به این نکته اشاره کردیم که هر یک از مسیریاب ها جدولی در حافظه خود تشکیل می دهند ولی سؤال اصلی اینجاست که این جداول چگونه ایجاد و به هنگام می شوند. چرا که در زمان های متفاوت شرایط ترافیکی و توپولوژیکی شبکه عوض شده و بالطبع این جداول باید با زمان تغییر داده شود تا همیشه بهترین وضعیت را برای مسیریابی ارائه بدهد.

در روش DV اصول کار بصورت زیر خلاصه می شود:

۱. هر مسیریاب موظف است هزینه خطوطی را که بصورت فیزیکی با مسیریاب های دیگر دارد، محاسبه کرده و در جدول خود درج نماید. هزینه خطوطی که مسیریاب با آن ها در ارتباطِ مستقیم نیست، در این جدول بینهایت در نظر گرفته می شود.

۲. هر مسیریاب موظف است در بازه های زمانی مشخص، ستون هزینه از جدول مسیریابی خودش را برای مسیریاب های مجاور ارسال نماید. یعنی فقط برای مسیریاب هائی که با آن در ارتباط است نه تمام مسیریاب ها. بنابراین هر مسیریاب در فواصل T ثانیه ای، اطلاعاتی را از مسیریاب های مجاور دریافت می کند که جدید است و می تواند بر اساس آن، جدول مسیریابی خود را به هنگام کند.

۳. هر مسیریاب موظف است پس از دریافت جداول مسیریابی از مسیریاب های مجاور، جدول خود را طبق یک الگوریتم بسیار ساده به هنگام نماید. (این الگوریتم با یک مثال، تشریح خواهد شد).

در مثال فوق، فرض کنید مسیریاب های مجاور J (یعنی مسیریاب های A، I ، H ،K) جداول مسیریابی خود را برای J ارسال کرده باشند. این جداول در شکل زیر نشان داده شده است.

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

جداول ارسالی توسط مسیریابهای مجاور J

مسیریاب J می تواند تخمین بزند از لحاظ زمانی تا مسیریاب های A، I ، H ،K چقدر تأخیر وجود دارد. این تخمین بسادگی از طریق ارسال بسته های Echo و دریافت پاسخ آن و محاسبهٔ زمان رفت و برگشت آن امکانپذیر است. فرض کنید با این روش مسیریاب J مقادیر تاخیر را بصورت زیر ارزیابی کرده باشد:

A ← J	۸ میلی ثانیه
I ← J	۱۰ میلی ثانیه
H ← J	۱۲ میلی ثانیه
K ← J	۶ میلی ثانیه

حال فرض کنید پس از رسیدن چهار جدول فوق به J و اندازه گیری J از مقدار تأخیر تا مسیریاب های A، I ، H ،K ، مسیریاب J بخواهد در جدول مسیریابی خود، بهترین کانال را برای ارسال بسته به هر یک از مسیریاب های A تا L بیابد. بعنوان مثال J می خواهد بداند بهترین مسیر برای رسیدن به G کدام است.

ابتدا به جدول رسیده از A مراجعه می کند؛ A ادعا کرده است که برای رسیدن به G تأخیری معادل ۱۸ میلی ثانیه دارد. پس اگر J بخواهد ازطریق A بسته ای برای G بفرستد، معادل ۲۶ میلی ثانیه تأخیر خواهد داشت؛ (یعنی ۸ میلی ثانیه از J به A + ۱۸ میلی ثانیه از A به G بر اساس ادعای A). J این مقدار را بطور موقتی در حافظه ذخیره می نماید.

حال به جدول رسیده از I مراجعه می کند؛ I ادعا کرده است که برای رسیدن به G تأخیری معادل ۳۱ میلی ثانیه دارد. پس اگر J بخواهد ازطریق I بسته ای برای G بفرستد، معادل ۴۱ میلی ثانیه تأخیر خواهد داشت؛ (یعنی ۱۰ میلی ثانیه از J به I + ۳۱ میلی ثانیه از I به G). J این مقدار را بطور موقتی در حافظه ذخیره می نماید.

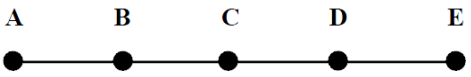
در جدول رسیده از H؛ تأخیر زمانی تا G معادل 6 میلی ثانیه ادعا شده است. بنابراین اگر J بخواهد ازطریق H بسته ای برای G بفرستد، معادل ۱۸ میلی ثانیه تأخیر خواهد داشت؛ (یعنی ۱۲ میلی ثانیه از J به H + ۶ میلی ثانیه از H به G). J این مقدار را بطور موقتی در حافظه ذخیره می نماید.

به همین ترتیب هزینه رسیدن به G از طریق K معادل ۳۷ (۶+۳۱) میلی ثانیه محاسبه و در حافظه ذخیره می شود.

با محاسبات فوق J می تواند با پیدا کردن حداقل مقدار از بین مقادیر محاسبه شده، بهترین مسیر برای ارسال یک بسته به G را پیدا کند. در مثال فوق از بین چهار مقدار ذخیره شده در حافظه، هزینه ارسال از طریق مسیریابِ H حداقل خواهد بود و تأخیری معادل ۱۸ میلی ثانیه خواهد داشت. مسیریاب J در جدول خود در رکورد متناظر با G آدرس مسیریاب H و هزینه ۱۸ را درج می کند.

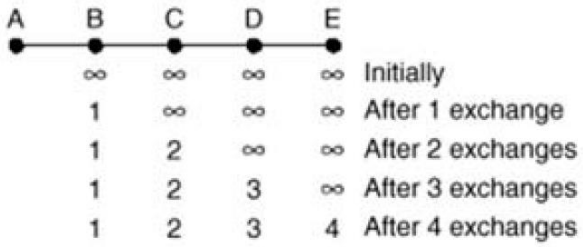
برای تمام مسیریاب های دیگر این روند تکرارمی شود و بهترین مسیر برای رسیدن به یکایک آن ها محاسبه شده و در جدول جدید درج خواهد شد. این جدول تا زمان به هنگام سازی بعدی، که جداول جدید از مسیریاب های مجاور می رسند قابل استناد بوده و بهترین مسیر را برای ارسال بسته ها تعیین می کند. به عنوان مثال برای ارسال بسته ای به مسیر یاب F با استناد به جدول محاسبه شده، باید آن بسته تحویل مسیریاب I شود. این روش در عین سادگی، پویا است و تغییرات ترافیکی شبکه با زمان را در جداول مسیریابی دخالت خواهد داد. از طرفی حجم جدولی که بایستی هر مسیریاب در حافظه خود نگه دارد از درجه ۱ است. یعنی به ازای n مسیریاب فقط n رکورد کافی است.

این الگوریتم خبرهای خوب را به سرعت منتقل می کند اما در انتقال خبرهای بد واگرا می شود و گاهی هرگز همگرا نمی شود. خبر خوب یعنی یک نود یا لینک اضافه شد، ترافیک فلان جا کمتر شد، طول فلان صف کوتاه تر شد (برعکس این ها خبرهای بدی هستند) به طور کلی این الگوریتم Stable نیست و در برخی شرایط می تواند واگرا باشد. این مشکل عمده" عدم همگرایی سریع جداول مسیریابی" در هنگام خرابی یک مسیریاب یا یک کانال ارتباطی می باشد که " شمارش تا بینهایت" نام گرفته است. این اشکال زمانی پیش خواهد آمد که یکی از مسیریاب ها دچار خرابی شود یا آنکه مسیر ارتباطی او با دیگران قطع شود. بعنوان مثال شکل زیر را در نظر بگیرید؛ در این شکل، A تا E مسیریاب ها هستند و هر کدام برای رسیدن به دیگری فقط یک مسیر در اختیار دارند. بعنوان مثال هزینه A تا B، ۱ میلی ثانیه، A تا E، ۴ میلی ثانیه است.

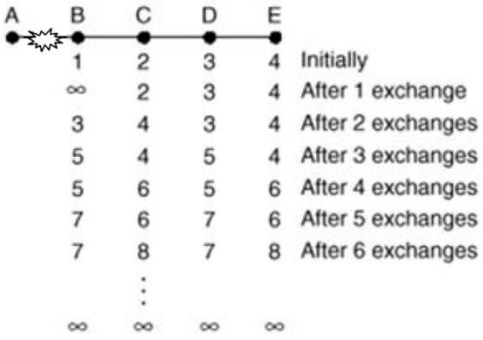


A	0,-	1,A	2,B	3,C	4,D
B	1,B	0,-	1,B	2,C	3,D
C	2,B	1,C	0,-	1,C	2,D
D	3,B	2,C	1,D	0,-	1,D
E	4,B	3,C	2,D	1,E	0,-

در حالت عادی جداول مسیریابی هر یک از مسیریاب ها ، طبق جداولِ شکل فوق تنظیم خواهد شد. روند تکمیل این جدول به عنوان نمونه، به تنهایی برای مسیریاب A به تصویر کشیده شده است که نهایتاً پس از چهار بار تبادل جداول همسایگی، گره A به ترتیب تا گره های B وC و D و E فاصله های ۱و ۲و ۳و ۴ را دارد که همین نتیجه به همراه گره ای که جهت مسیریابی را نیز مشخص می کند، برای همه ی گره ها در شکل فوق بصورت کلی به تصویر کشیده شده است.



حال فرض کنید ناگهان خط ارتباطی A به B قطع شود. بنابراین در این حالت هزینهٔ مسیریاب B به A بینهایت خواهد شد و هزینهٔ ارسال بسته به A در جداول هر یک از مسیریاب ها ، طبق روند شکل زیر تغییر خواهد کرد.



دقت کنید که پس آنکه ارتباط B با A قطع شد، B در جدول خود، هزینهٔ رسیدن به A را مقدار ∞ درج می کند ولی پس از گذشت T ثانیه جدول مسیریابی از C می رسد و C اعلام کرده که مقدار هزینه تا A مقدار ۲ است زیرا C نمی داند که کانالB با A قطع شده است و در نتیجه همان مقدار قبل از وقوع خرابی را اعلام می کند. B که در جدولش مقدار هزینه تا A را بینهایت درج کرده است، به خیال آنکه C با A مسیری مجزا دارد (با هزینه ۲)، در جدول خود هزینه رسیدن به A را از ∞ به ۳ تبدیل می کند (۱ واحد B تا C + ۲ واحد از C به A طبق ادعای C). بنابراین B فرض کرده که از C به A کانال مستقلی وجود دارد و هزینه آن هم ۲ است در حالی که چنین فرضی اشتباه است.

در T ثانیه بعد مجدداً جداول مسیریابی بین همسایه ها رد و بدل می شود C جدولB را گرفته و متوجه می شود که هزینه رسیدن به A مقدار ۳ شده است بنابراین با احتساب مقدار جدید هزینه رسیدن از C به A در جدولش ۴ می شود.

در T ثانیه بعد ، مسیریابB با دریافت جدول مسیریابی C باز هم به اشتباه هزینه رسیدن به A را در جدول خودش ۵ درج می کند. (۱ واحد تا C + ۴ واحد از C به A طبق ادعای C).

این روند تا بی نهایت ادامه دارد و بطور مداوم بین مسیریاب ها اطلاعات غلط در مورد A مبادله می شود. مشکل از آن جائی است که C, D, E نمی دانند که تنها مسیرشان به A از طریق B است. و به B اعلام می کنند که راهی به A دارند و هزینهٔ آن را اعلام می کنند ، در حالیکه که تمامی این راه ها همانی است که فعلاً قطع شده است!

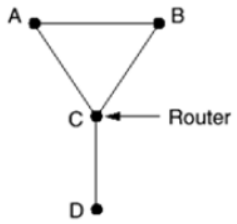
مشکل از این عبارت است که در اجرای این الگوریتم به آن عمل می شود: کوتاه ترین مسیر را از همسایه بپرسیم و آن را در نظر بگیریم!!!!

راه حل Split Horizon hack:

روش های گوناگونی برای حل این مسأله پیشنهاد شده که عمدتاً پیچیده اند یا مقرون به صرفه نیستند. ساده ترین راه حل آن است که وقتی یک مسیریاب می خواهد اطلاعاتی را به همسایه هایش بدهد هزینه رسیدن به آن هایی را که قطعاً باید از همان مسیریاب بگذرند را اعلام نمی کند (یا ∞ اعلام می کنند). بعنوان مثال C چون می داند مسیر A از B می گذرد وقتی خواست جدول مسیریابی خود را به B اعلام کند هزینه رسیدن به A را همیشه ∞ اعلام می کند چرا که برای رسیدن به A قطعاً باید از B عبور کرد؛ در این حالت جداول مسیریابی سریعاً اصلاح خواهد شد. این راه حل در توپولوژی های پیچیده باز هم کار نمی کند. مثالی از شکست این راه حل در شکل زیر آمده است:

Failure of Split Horizon

- Init: CD=1, AD=2, BD=2, AB=1
- CD fails, A and B tell C they can't reach D, so C sets CD=∞
- Unfortunately, A hears from B that BD=2, so it sets AD=3 => wrong!



چرا این اتفاق بد می افتد؟

چون نودها در شبکه دید محدود به همسایه های خود دارند. در DVR نودها به تنهایی هیچ احساسی از توپولوژی ندارند و فقط همسایه هایشان را می شناسند و برای رسیدن به یک مقصد فقط می دانند از طریق چه همسایه ای. ولی نمی دانند سایر نودهای شبکه چگونه به هم متصل هستند. راه حل قطعی این مشکل این است که هر نود توپولوژی شبکه را بداند و sink tree را بسازد و بر اساس آن داده هایش را ارسال نماید. که در روش بعدی از این تکنیک استفاده می شود.

مسیریابی حالت پیوند یا LS (Link State Routing)

مشکل شمارش تا بی نهایت (∞ Count to Infinity Problem) که در بالا شرح داده شد و الگوریتم DVR را واگرا می کرد و موجب ناپایداری آن می شد باعث شد که در سال 1979 الگوریتم دیگری بنام LS جایگزین آن شود. این الگوریتم در ۵ مرحله زیر عمل می کند.

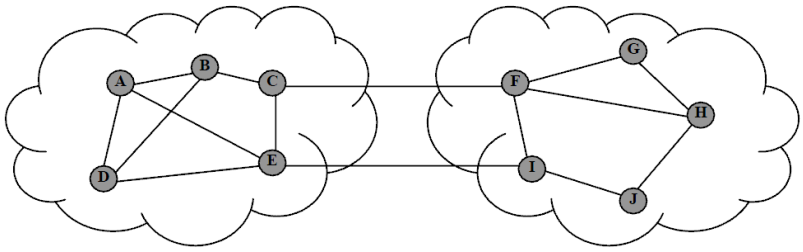
۱. همه همسایگان خود را شناسایی کن و آدرس یکتای هر یک را بدست بیاور.
۲. تاخیر یا هزینه(فاصله) هر یک از همسایگان خود را با خود اندازه گیری کن(تخمین بزن).
۳. بسته ای (Packet) بساز و اطلاعاتی که از همسایگان خود کسب کرده ای در آن جاسازی کن.
۴. این بسته را برای تمامی مسیریاب ها بفرست (به روش سیل آسا).
۵. با استفاده از الگوریتم کوتاه ترین مسیر (Dijkstra) کوتاه ترین مسیر رسیدن به هر یک از مسیریاب های شبکه را محاسبه کن. به عبارت دیگر shortest path را بساز.

مرحله ۱- شناسایی همسایه ها

هر گاه یک مسیریاب boot شده و آغاز به کار می کند بر روی هر یک از پورت های خود بسته ای خاص بنام Hello packet را ارسال می کند و منتظر می نشیند تا پاسخ های سلام خود را بشنود. انتظار می رود مسیریاب های همسایه در پاسخ سلام آدرس خود را ارسال نمایند.

مرحله ۲- اندازه گیری یا تخمین هزینه (تاخیر)

می خواهیم ببینیم وضعیت link بین ما با هر یک از همسایگانمان چگونه است و یک تخمین قابل قبول از تاخیر linkها بدست می آوریم. برای این کار یک بسته به نام Echo ارسال می کنیم و پس از بازگشت بسته Round Trip Time (RTT) را بر 2 تقسیم می کنیم. با فرض تقارن شبکه و تکرار این عمل و میانگین گیری تقریب خوبی از تاخیر بدست می آید. در این حالت میزان بار و زمان انتظار بسته های به صف شده و منتظرِ پردازش در مسیریاب، به حساب نمی آید که چندان صحیح نیست. مسیریاب می تواند بسته های Echo را همانند بسته های معمولی به انتهای صف بفرستد و پس از فرا رسیدن نوبت پردازشِ بسته، به آن پاسخ بدهد؛ در این حالت معیار دقیق تری از تاخیر بدست می آید. ولی منظور کردن این پارامتر در معیار هزینه مشکلی دیگر پدید می آوردکه برای رفع آن باید الگوریتم را پیچیده تر کرد. برای تشریح این مسئله به شکل زیر دقت کنید.



شبکهٔ I **شبکهٔ II**

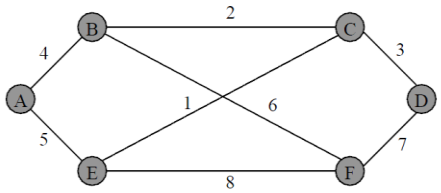
در این مثال فرض کنید تمام مسیریاب ها زمان تاخیر انتشار و میزان بار (زمان انتظار پردازش برای هر بسته) را به عنوان پارامتر هزینه در نظر می گیرند. فرض کنید تمامی مسیریاب ها با اجرای الگوریتم کوتاهترین مسیر، مسیر بهینه از شبکهٔ I به شبکهٔ II را از طریق خط CF تشخیص بدهند و تمامی بسته ها را به این مسیر هدایت نمایند؛ چراکه این خط کمترین تاخیر و ترافیک را داشته است. با این کار در عرض چند ثانیه خط CF با ترافیک زیادی روبرو شده و تاخیر آن زیاد خواهد شد. با به هنگام شدن جدول مسیریابی، کانال CF یک خط با تاخیر زیاد و " بد" گزارش می شود و از آن به بعد تمامی مسیریاب ها از خط EI بعنوان بهترین مسیر از شبکهٔ I به شبکهٔ II استفاده خواهند کرد. مجدداً خط EI با بار زیادی مواجه شده و به بدترین مسیر تبدیل می شود و این روند" نوسان[1]" تا بینهایت تکرار خواهد شد. این قضیه بسیار طبیعی است که اگر همهٔ مسیریاب ها از بهترین مسیر برای هدایت بسته ها استفاده کنند، پس از مدت کوتاهی بهترین مسیر به بدترین مسیر تبدیل خواهد شد.
برای اصلاح این مشکل می توان یا از معیار زمان انتظار و پردازش چشم پوشی کرد یا آنکه روشی انتخاب کرد که درصدی از ترافیک بسته ها روی خطوط غیر بهینه توزیع شود. مثلاً وقتی مسیریاب C بهترین کانال را خط CF تشخیص میدهد % 70 از بسته ها را روی CF و % 30 باقیمانده را روی خط CE ارسال کند تا از طریق EI به سمت شبکهٔ II هدایت شوند. پیاده سازی چنین روشی پیچیدگی زمانی الگوریتم را افزایش می دهد.
چرا Oscilation بد است؟ چون به محض تغییر جداول همگی سوئیچ می کنند روی یک مسیر و در بخشی از شبکه ازدحام بوجود می آید. در حالی که منابع آزادی از شبکه در جای دیگر بلااستفاده مانده است.

مرحله ۳- ساخت بسته های وضعیت LINK (Link State Packet)

بسته وضعیت link حاوی فیلدهای زیر است:

۱. آدرس فرستنده (مسیریاب تولید کننده بسته)
۲. شماره ترتیب (برای تشخیص بسته های تکراری از بسته های جدید)
۳. TTL (Time To Live) یا Age یا طول عمر که یک شمارنده است و از مقدار معینی شروع می شود و هر دفعه (با عبور از هر مسیریاب یا گذشت یک ثانیه) یک واحد از آن کم می شود و هر وقت به صفر رسید این بسته از بین می رود. به عبارتی زمان انقضای بسته را مشخص می کند.
۴. فهرست همسایه ها (مسیریاب های مجاور) و وضعیت (تاخیر link بین ما و هر همسایه)

برای روشن شدن قضیه به شکل زیر توجه کنید. در این شکل شش مسیریاب A تا F و خطوط ارتباطی مابین آن ها و هزینه هر خط نشان داده شده است.



یک زیرساخت از یک شبکهٔ فرضی

[1] Oscillation

قالب کلّی جدولی که هر مسیریاب باید بسازد در شکل زیر نشان داده شده است. ساختن این بسته ها مشکل نیست بلکه قسمت مشکل مسئله، زمان تولید و توزیع آن ها بر روی شبکه می باشد.

نکته: این بسته ها چه زمانی ارسال می شود؟ دو روش داریم:

a) پریودیک یا در زمان های خاص: در این حالت در بازه های زمانی مشخص، کلّ شبکه برای لحظاتی غرق در بسته های LS خواهد شد.

b) هر وقت تغییر ذاتی در توپولوژی شبکه یا وضعیت link ها (میزان تاخیر و غیره) مشاهده شود.

A	
Seq.	
Age	
B	4
E	5

B	
Seq.	
Age	
A	4
C	2
F	6

C	
Seq.	
Age	
B	2
D	3
F	1

D	
Seq.	
Age	
C	3
F	7

E	
Seq.	
Age	
A	5
C	1
F	8

F	
Seq.	
Age	
B	6
D	7
E	8

بسته‌های LS

وظیفهٔ فیلدهای شماره ترتیب و طول عمر در بخش بعدی مشخص شده است.

مرحله ۴- توزیع بسته های Link State

مهمترین نکته در توزیع این بسته های LS همگام سازی مسیریاب های دریافت کننده این بسته ها است. یک راه حل برای توزیع بسته های LS روش سیل آسا (Flooding) است. برای آنکه این بسته ها در یک حلقهٔ بینهایت تکرار نشوند، هر بسته دارای یک شمارهٔ ترتیب است که با ورود به هر مسیریاب ابتدا بررسی می شود که آیا این بسته قبلاً دریافت شده است یا آنکه یک بستهٔ جدید است. در صورت جدید بودن، مسیریاب ضمن درج اطلاعات درون بسته در یک جدول موقت، آنرا روی تمامی خروجی های خود (به غیر از کانالی که بسته را از آن دریافت کرده) ارسال می نماید. بنابراین هر مسیریاب موظف است که شمارهٔ ترتیب بسته های LS خود را که در هر مرحله ارسال می کند، حفظ نماید و در هر بار ارسال، یکی به آن اضافه کرده تا در مراحل بعدی، بسته های LS با شمارهٔ جدید و غیر تکراری ارسال شود.

نکته: اگر محدودهٔ شماره کوچک باشد مثلاً 4 بیتی بعد از 16 بسته دوباره reset شده و طبق الگوریتم فوق بسته های دیگر در نظر گرفته نمی شوند. راه حل این است که محدوده شماره را 32 بیتی و بزرگ در نظر بگیریم. و اگر شمارهٔ بسته ها از صفر شروع و به فرض در هر ثانیه یکی به آن اضافه شود تکرار شمارهٔ یک بسته ، 137 سال طول می کشد!

وقتی مسیریاب بسته ای را با یک شمارهٔ خاص دریافت کرد دیگر بسته های با شمارهٔ کوچکتر از آن شماره را دریافت نخواهد کرد. بعنوان مثال اگر مسیریابی یک بستهٔ LS را از یک مسیریاب خاص با شمارهٔ ترتیب 654320 دریافت کند و آنرا در جدولش درج نماید، از آن به بعد بسته هایی را که شمارهٔ کمتر از 654321 داشته باشد، را از آن مسیریاب قبول نخواهد کرد. همین مسئله مشکل بزرگی برای مسیریاب هایی که به ناگاه از کار می افتند و می خواهند مجدداً به شبکه وارد شوند، پدید خواهد آورد؛ چراکه یک مسیریاب وقتی از شبکه خارج شده و می خواهد مجدداً به شبکه وارد شود مجبور است شمارهٔ بسته ها را از صفر شروع کند و هیچ مسیریابی آنرا نخواهد پذیرفت. برای رفع این مشکل برای هر بسته یک طول عمر در نظر گرفته می شود که به ازای هر ثانیه یک واحد از آن کم می شود؛ هرگاه مسیریابی بسته ای را دریافت و آنرا در جدولی درج نماید ولی در خلال طول عمر بسته (مثلاً 10 دقیقه معادل600 ثانیه) بستهٔ جدیدی دریافت نکند آن مسیریاب از جدول مسیریابی حذف خواهد شد. با این قاعده وقتی مسیریابی که از جدول حذف شده، بخواهد به شبکه برگردد بسته های ارسالی از طرف او با هر شماره ای دریافت خواهد شد. با این روش مشکل دریافت بسته های تکراری حل خواهد شد. این راه حل برای زمانی که به علت خطای فیزیکی در شماره ترتیب (شماره ترتیب خیلی بزرگتر از شماره ترتیب اولیه شود) نیز کاربرد دارد. مثلاً فرض کنید بسته ای با شماره ترتیب ۲۱ ارسال شده است. اما به علت خطای کانال فیزیکی شماره آن به ۲۱۰۰۰ تغییر کرده. لذا هیچ یک از مسیریاب های دیگر بسته های بعدی که شماره ترتیب های ۲۲و ۲۳و ۲۴ و ... دارند را قبول نخواهند کرد. اما به علت وجود فیلد Age یا طول عمر در رکوردهای جدول هر مسیریاب، Entry مربوطه بعد از مدتی حذف و بسته های بعدی دریافت خواهند شد.

مسئلهٔ بحرانی دیگر آنست که اگر یک مسیریاب به هر دلیلی اطلاعات غلط ارائه کند، کلّ مسیریابی شبکه با اشکال مواجه شده و تمام جداول مسیریابی با اطلاعات آلوده تنظیم خواهد شد. این مشکل زمانی بروز می کند که یک مسیریاب به اشتباه اعلام کند که کانالی فیزیکی با

مسیریاب دیگر دارد در حالی که نداشته باشد؛ یا کانالی فیزیکی با یک مسیریاب داشته باشد ولی اعلام نکند. گاهی این مشکلات بصورت عمدی توسط اخلالگران بوجود می آید چراکه ایجاد بسته های LS بصورت مصنوعی و تزریق آن به شبکه توسط کاربران فقط نیاز به برنامه نویسی دارد. برای رفع این مشکل، در مسیریاب های مدرن بسته های LS زمانی پذیرفته می شود که قبل از آن، هویت ارسال کنندهٔ بسته احراز شود.

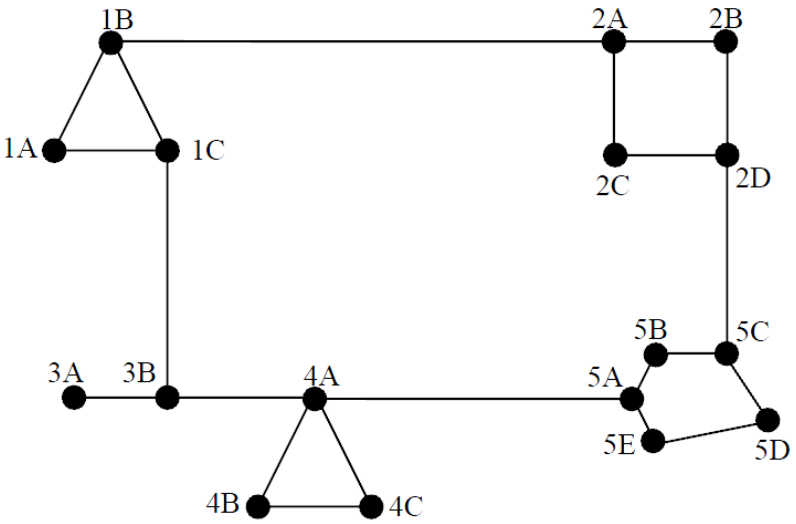
مرحله ۵- محاسبه مسیرهای جدید

وقتی یک مسیریاب بسته های LS را از تمامی مسیریاب های شبکه دریافت کرد، می تواند ساختمان دادهٔ گراف زیرشبکه را تشکیل داده و بر اساس آن اقدام به پیدا کردن بهترین مسیرها (یعنی با کمترین هزینه) بین هر دو گره بنماید. این کار توسط الگوریتم کوتاه ترین مسیر مثل دایْجْکِسترا که قبلاً بحث شده است به راحتی انجام می شود.

اگر سایز شبکه و تعداد نودها زیاد شود، این الگوریتم به سختی قابل اجرا خواهد بود. چون هم سایز جداول زیاد خواهد شد و هم اطلاعات با تأخیر زیاد به سایرین می رسد. لذا در شبکه های با وسعت بیشتر از الگوریتم های hierachical یا سلسله مراتبی استفاده می کنیم.

مسیریابی سلسله مراتبی (hierachical)

در الگوریتم های مسیریابیِ LS , DV هر مسیریاب باید جدولی را به عنوان جدول مسیریابی تشکیل بدهد. وقتی یک شبکه رشد می کند و شبکه های محلی و مسیریاب ها اضافه می شوند حجم جداول مسیریابی و زمان لازم برای تعیین مسیر یک بسته افزایش پیدا می کند؛ تا جاییکه ممکن است این زمان به تاخیرهای بحرانی بیانجامد و عملاً کارآیی شبکه کاهش چشمگیر داشته باشد. در شکل زیر به زیرساخت ارتباطی از یک شبکهٔ فرضی دقت کنید. در این شبکه 17 مسیریاب وجود دارد و مسیریابی به روش DV انجام می شود.

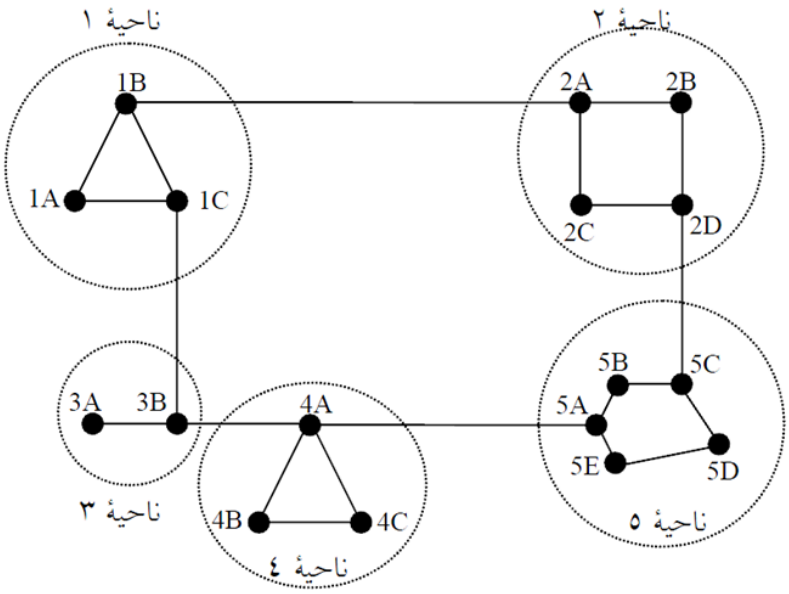


در این مثال معیار هزینه ’’تعداد گام ‘‘می باشد. وقتی معیار هزینه تعداد گام است (براحتی هزینهٔ هر خط را ثابت و مساوی 1 فرض کنید) در شکل زیر جدول مسیریابی 1A نشان داده شده است و بگونه ای که ملاحظه می شود دارای 17 رکورد است. تمام مسیریاب ها دارای چنین جدولی هستند و در فواصل منظم باید آن را برای مسیریاب های مجاور خود ارسال نمایند. حال فرض کنید که تعداد صدهزار مسیریاب در شبکه موجود باشد (صدهزارمین شبکهٔ دنیا که به اینترنت پیوست در سال 1996 ثبت شده است!) در چنین حالتی مسیریاب قادر به ذخیره ، پردازش و ارسال جداول مسیریابی برای دیگر مسیریاب ها نخواهد بود.

مقصد	خطّ	هزینه
1A	–	-
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

جدول مسیریابی 1A

در مسیریابی سلسله مراتبی ، مسیریاب ها در گروه هایی به نام" ناحیه" دسته بندی می شوند. هر مسیریاب فقط" نواحی" و مسیریاب های درون ناحیهٔ خود را می شناسد و هیچ اطلاعی از مسیریاب های درون نواحی دیگر ندارد. در شکل زیر شبکهٔ مثال قبل در قالب پنج ناحیه تقسیم بندی شده و جدول مسیریابِ 1A به تصویر کشیده شده است. در این جدول به ازای هر" ناحیه " و هر مسیریاب درون ناحیه، یک رکورد در حافظه نگهداری می شود. به عنوان مثال مسیریاب 1A برای ارسال یک بسته به مسیریابی که در ناحیهٔ 2 واقع است، آنرا به سمت مسیریاب 1B هدایت می نماید؛ طبق این جدول برای ارسال بسته به نواحی 3 و 4 و 5 ، باید از طریق 1C اقدام شود.



مقصد	خطّ	هزینه
1A	–	–
1B	1B	1
1C	1C	1
Region 2	1B	2
Region 3	1C	2
Region 4	1C	3
Region 5	1C	4

جدول مسیریابی 1A

همانطور که می بینید با استفاده از مسیریابی سلسله مراتبی تعداد رکوردهای جدول مسیریابی از 17 به 7 رکورد کاهش یافته است (۳ رکورد برای مسیریاب های درون ناحیه و ۴ رکورد برای بقیهٔ نواحی).

حال فرض کنید شبکه ای با 720 مسیریاب ایجاد شده است؛ در روش معمولی DV جدول مسیریابی دارای 720 رکورد خواهد بود. اگر این شبکه به 24 ناحیه تقسیم و در هر ناحیه 30 مسیریاب تعریف شود، در این حالت مسیریاب نیاز به 30 رکورد به ازای هر مسیریاب در درون ناحیه و 23 رکورد به ازای 23 ناحیهٔ دیگر دارد(جمعاً 53 رکورد).

برای ساده تر شدن جداول مسیریابی می توان از روش" سلسله مراتبی سه سطحی " استفاده کرد. در این روش کل شبکه به صورت زیر تقسیم بندی می شود:

- کل شبکه به چندین"دسته" تقسیم می شود.
- هر دسته به چند " ناحیه " تقسیم می شود.
- هر ناحیه در برگیرندۀ چند مسیریاب است.

به عنوان مثال شبکه ای با 720 مسیریاب به" 8 دسته "، هر دسته دارای" 9 ناحیه "و هر ناحیه دارای 10 مسیریاب باشد. در این حالت هر مسیریاب به حداکثر 25 رکورد احتیاج دارد:

- ۱۰ رکورد برای مسیریاب های هم ناحیه
- ۸ رکورد برای بقیۀ نواحی (۸=۱-۹)
- ۷ رکورد برای بقیۀ دسته ها (۸=۱-۹)

سلسله مراتب در مسیریابی می تواند بسته به بزرگی شبکه، تا چندین سطح ادامه یابد. تنها اشکالی که می توان برای روش سلسله مراتبی برشمرد، آنست که چون در این روش کل توپولوژی زیرشبکه برای هر مسیریاب مشخص نیست، لذا ممکن است مسیر انتخابی برای ارسال بسته به یک مسیریابِ خاصّ درون یک ناحیه بهینه نباشد ولی در مجموع این روش به حالت بهینه نزدیک بوده و تغییرات توپولوژیکی و ترافیکی در جداول مسیریابی تاثیر داده خواهد شد.

مسیریابی انتشاری یا (Broadcast Routing)

برای انتشار بسته ها در لایه شبکه در شبکه ای مانند اینترنت چه باید کرد؟ هر یک از الگوریتم های زیر را می توان برای انتشار بسته ها از یک مبدا به همه میزبان های درون شبکه پیشنهاد کرد. البته هر روش مزایا و معایب خود را دارد.

روش ۱- Send lots of individual packets

یک لیست از آدرس همه مقاصد داشته باشیم و در یک حلقه به صورت نقطه به نقطه بسته را به یکایک ماشین ها ارسال کنیم. مشکلات این روش عبارتند از اتلاف پهنای باند، کند بودن الگوریتم و نیاز به نگهداری فهرست طولانی از آدرس ها.

روش ۲- استفاده از الگوریتم مسیریابی سیل آسا

این روش پکت های خیلی زیادی را تولید می کند.

روش ۳- مسیریابی چندمقصدی (Multi-Destination Routing) :

در این روش در آدرس بسته یک نگاشت بیتی وجود دارد (Bitmap) که هر بیت آن یکی از گره های شبکه را نشان می دهد. حال فرض کنید یک بسته انتشاری به یک گره می رسد بیت مربوط به خود را reset می کند و بسته را در صورتی به سمت linkهای خروجی می فرستد که بیت مربوط به گره متصل به آن link یک باشد (reset نشده باشد). همانند روش ۱ نیازمند دانستن لیست همهِ آدرس های مقصد هستیم و در فیلد آدرس مقصد نیز باید همگی را بگذاریم که خیلی عملیاتی نیست.

روش ۴- استفاده از درخت پوشا (Spaning Tree):

درخت پوشا درختی است (بدون حلقه) که شامل همه گره های شبکه می شود. اگر بهینه باشد به آن sink tree می گویند. می توان یک پکت را در جهت عکس یک sink tree به مقاصد مختلف ارسال کرد.

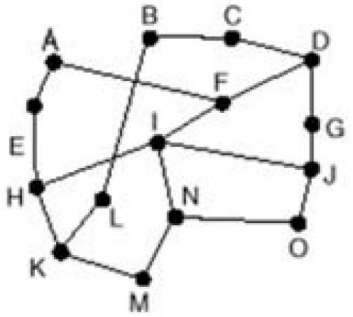
نکته: sink tree واحد نیست. کافی است مسیریاب ها، اطلاعات یکی از درخت های پوشا را داشته باشند و بسته را از طریق این درخت یا شاخه های این درخت به همه گره ها برسند (از ریشه به برگ ها). مسیریاب با استفاده از اطلاعات وضعیت link می تواند این درخت را پیدا کند. اگر

فرض کنیم شبکه متقارن است، این روش بهینه ترین و بهترین روش خواهد بود. چرا که یک sink tree بهترین مسیرها را از گره های مختلف به ریشه نشان می دهد پس در مسیر عکس نیز مسیرهای بهینه همین مسیرها خواهند بود.

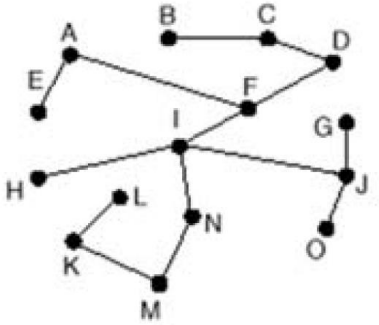
روش ۵ - Reverse Path Forwarding (هدایت بر روی مسیر معکوس)

اگر sink tree گره های دیگر را ندانیم می توانیم از روش پنجم استفاده کنیم. ایده روش به این صورت است که هر نود در شبکه برای ارسال بسته هایش از مسیر خاصی استفاده می کند(نودها مطابق با جدول مسیریابی خود بسته ها را هدایت می کنند). در صورتی که در شبکه ای عمل broadcast بخواهد صورت پذیرد، هر نود وقتی یک بسته همه پخشی به دستش برسد، نگاه می کند که از کدام پورت به دستش رسیده است؟ هر گره فقط بسته های پخشی را در صورتی می پذیرد که از مسیری دریافت شده باشد که برای ارسال داده هایش به سمت گره مبدا بسته های پخشی، از طریق همان مسیر ارسال می کند. به عبارت دیگر بسته هایی که از سایر link ها دریافت می شود دور ریخته می شود تا از تکرار بسته های اضافی جلوگیری شود. بسته ای که از مسیر معکوس دریافت می شود به سمت هر یک از گره های مجاور ارسال می شود. برای درک این روش به مثال زیر توجه کنید.

در شکل زیر یک زیر شبکه به تصویر کشیده شده است.



فرض کنید sink tree گره A همانند شکل زیر است. اما در عین حال سایر نودها از sinktree نود A خبر ندارند. ولی روتینگ آن ها اینگونه تنظیم شده است که هرگاه بخواهند چیزی برای A بفرستند، در جهت های مشخص شده در شکل زیر برای توپولوژی مذکور استفاده می کنند. مثلاً در توپولوژی فوق L دوتا پورت دارد. یک پورت آن به B و دیگری به K متصل است. ولی بطور طبیعی، هرگاه گره L داده ای برای I میفرستد از مسیر $L \rightarrow K \rightarrow M \rightarrow N \rightarrow I$ داده هایش را ارسال می کند. یعنی L بسته هایی با مقصد I را همیشه دست K می دهد.

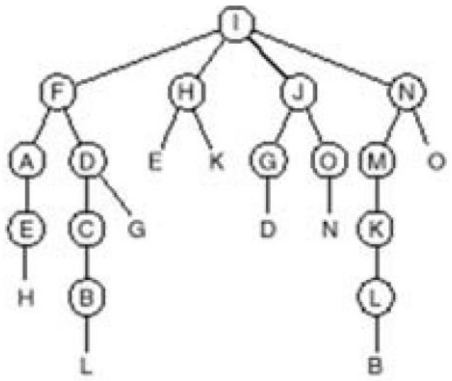


برای تشریح الگوریتم به این مثال توجه کنید. همان که در توپولوژی این شبکه مشاهده می کنید، نود D سه همسایه C, F, G دارد. ولی اگر D داده ای برای ارسال به سمت I داشته باشد فقط از نود F استفاده می کند(طبق جدول روتینگ).

حال فرض کنید قضیه برعکس باشد. یعنی گره I به عنوان مبداء broadcast بسته ای را به همه ی همسایگانش ارسال کند. در این صورت بسته به چهار همسایه I یعنی نودهای F, H, J, N ارسال می شود.

عکس العمل گره F: F پس از دریافت بسته از I و بررسی مبداء تولیدکننده بسته همگانی، متوجه می شود که این بسته از همان جهتی به دستش رسیده که اگر خودش می خواست برای آن مبداء داده ای ارسال کند، از آن مسیر استفاده می کرد. پس بسته را به همسایگان دیگرش یعنی نودهای A , D هدایت یا forward می کند.

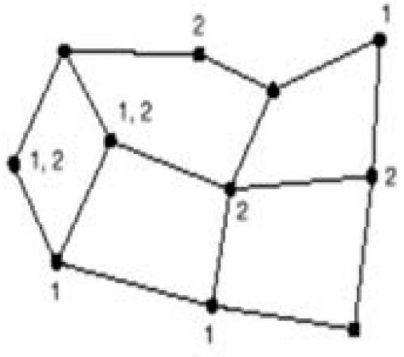
عکس العمل گره H: H می بیند از همان پورتی که برای I ارسال می کند، از I بسته ای را دریافت کرده است. پس نتیجه میگیرد که بسته از مسیر مناسبی آمده است و مجدداً بسته را برای سایر همسایگان خود یعنی E, K ارسال می کند.
عکس العمل J , N نیز همانند دو گره قبلی خواهد بود.
تقاوت از این به بعد مشخص خواهد شد. دو گره A , D پکت دریافتی از F که فرستنده واقعی آن I بوده است را از پورت هایی دریافت می کنند که به هنگام ارسال داده هایشان برای I از همان پورت استفاده می کرده اند. لذا این دو نود نیز بسته را برای همسایگان خود ارسال خواهند کرد.
اما K , E وقتی به فرستنده واقعی بسته دریافتی از H نگاه می کنند، می بینند که از سمت I آمده است. اما زمانی که داده ای برای ارسال به سمت I دارند از طریق H ارسال نمی کرده اند. لذا نتیجه می گیرند که بسته از مسیر مناسبی به دستشان نرسیده است و برای همین، این بسته را دیگر forward نمی کنند. اگر همین روند در سایر نودهای شبکه انجام شود درختی بوجود می آید که مسیر پخش بسته broadcast در شبکه را نشان می دهد. که در شکل زیر نشان داده شده است.



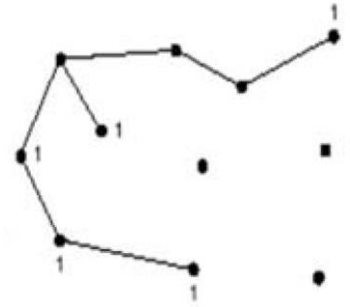
همان که در شکل فوق می بینید، بسته broadcast بطور نسبتاً خوبی پخش می شود. اما همچنان یک سری ارسال اضافه هنوز وجود دارد. اما در شرایطی که sink tree را نمی دانیم، روش خوبی است و از flooding بهتر عمل می کند.

Multicast Routing:

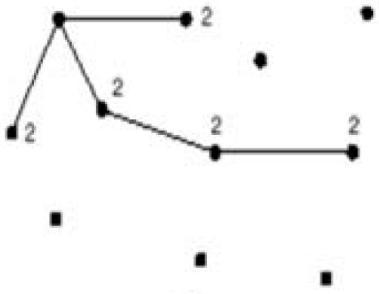
برای ارسال پکت ها به گروه خاصی از نودها در شبکه به الگوریتم های دیگری احتیاج داریم. برای نیل به این هدف، نیازمند مدیریت گروه های متفاوتی از نودها خواهیم بود. Multicast routing مفهومی است بین unicast و broadcast.
ایده اصلی کار بدین صورت است که بایستی در گراف شبکه یک subset یا زیر مجموعه ایجاد کنیم. که در آن زیرمجموعه، همه ی نودهایی که متعلق به یک گروه هستند، وجود داشته باشند. مثلاً فرض کنید گراف زیر را داریم.



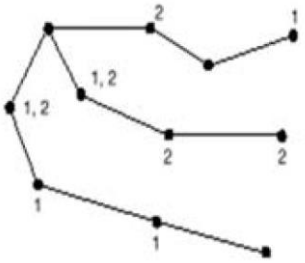
در این شکل با شماره مشخص شده است که هر نود عضو چه گروهی می باشد. مثلاً بعضی از نودها فقط عضو گروه ۱ هستند (آن هایی که برچسب ۱ کنارشان زده شده است). بعضی از نودها فقط عضو گروه ۲ هستند و بعضی از نودها عضو هر دو گروه می باشند. عضو بودن در یک گروه به این معنی است که می خواهیم از یک مبداء خاص به همه ی اعضای آن گروه داده ای را ارسال کنیم.
مثال: وقتی می خواهیم یک stream را در شبکه broadcast کنیم، اما همه نودها نمی خواهند آن را ببینند. مثلاً فرض کنید یک درس در یک کلاس داده می شود و در دانشگاه پنج کلاس دیگر هم بخواهند از محتوای کلاس بهره برداری کنند. یک راه حل این است که ۵ استریم جداگانه به کلاس های مذکور ارسال شود. اما راه حل دیگر این است که یک استریم واحد در شبکه ارسال کنیم و در شبکه تا جایی که امکان دارد مشترک برود و جایی که لازم هست، تفکیک شود. مثلاً فرض کنید در توپولوژی نشان داده شده فوق می خواهیم برای نودهای عضو گروه۱ محتوایی را ارسال کنیم. در این صورت لازم است ابتدا یک درخت پوشا (spanning tree) از نود های گروه۱ بگونه ای بسازیم که مبداء ارسال داده در ریشه قرار گیرد. آنگاه برای ارسال داده به اعضای گروه۱، مطابق این درخت داده را ارسال می کنیم.



همانطور که در شکل فوق می بینید درخت پوشایی از نودهای عضو گروه۱ ساخته شده است. که برای ساخت این درخت مجبور شده ایم نودهای واسطی را نیز در این درخت شرکت دهیم تا اتصال بین همه نودهای عضو گروه۱ بوجود آید. زمانی که استریم پخش شده از مبداء مطابق این درخت پخش می شود، نودهایی که عضو گروه۱ هستند هم مصرف کننده هستند و هم می توانند مجدداً استریم دریافتی را forward کنند و لازم نیست استریم جداگانه دیگری برای نود بعدی تولید و ارسال شود.
درخت زیر، نیز درختی است که حاوی همه ی نودهای گروه۲ است.



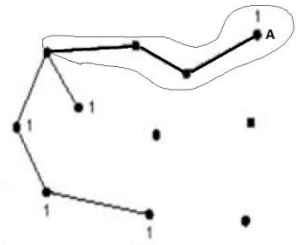
هرگاه بخواهیم یک استریم به اعضای گروه۱ یا ۲ برسد از درخت دیگری مطابق شکل زیر استفاده می کنیم.



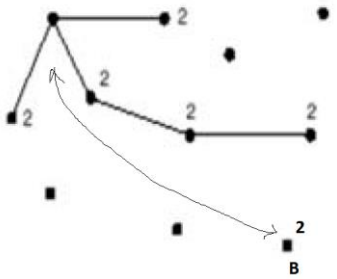
همانطور که مشاهده می کنید، هرگاه چندین درخت پوشا در مسیرهایی مشترک هستند، درخت احتیاج به هرس کردن دارد تا از ارسال های اضافی اجتناب شود.

نکته ۱: چگونگی ساخت درخت است. این که در ساخت درخت چه معیارها یا متریک هایی را مد نظر داشته باشیم.

نکته۲: ساخت درخت به شدت پویا است. بسیاری از کاربردهای multicast بگونه ای است که مثلاً ممکن است یک استریم داده ای مدت زمان محدودی پخش شود. اما در طول این مدت ممکن است گره های جدیدی به این گروه بپیوندند و افرادی نیز ارتباطشان را قطع کنند یعنی نخواهند محتوای پخش شده را دریافت کنند. در این صورت ممکن است تغییرات بنیادی در درخت بوجود آید. مثلاً در شکل زیر اگر نود A که عضو گروه۱ است، نخواهد در گروه بماند، کل شاخه بالا دیگر لازم نیست وجود داشته باشد و کلاً می تواند حذف شود.



یا برعکس اگر بخواهد نود جدیدی به گروه بپیوندد، ممکن است لازم باشد یک شاخه جدید در درخت اضافه شود. مثلاً در گروه ۲ اگر گره B بخواهد به گروه بپیوندد، لازم است شاخه جدیدی به درخت افزوده شود.



ساخت و اصلاح درخت یکی از فرایندهای مهم در multicast است. زمانی به نظر می آمد که multicast خیلی استفاده شود، اما هم اکنون multicast زیاد مطرح نیست و جایی بیشتر مطرح است که نزدیک و انتهای مصرف کنندگان باشد.

یک مثال واقعی سرویس IPTV برای منازل است. بگونه ای که خود مصرف کنندگان تصمیم می گیرند که چه کانال تلویزیونی را به کمک پکت های IP دریافت کند. در این صورت ۲ حالت رخ می دهد:

۱- مثلاً وقتی کانال۲ را می زند، هر آنچه در کانال۲ پخش می شود نشان داده می شود.

۲- کاربر بتواند در منویی یک فیلم خاص را انتخاب کرده و آن فیلم برایش پخش گردد.

تفاوت روش های فوق بدین شرح است:

در روش ۱- برای همه یک استریم ارسال می شود ولی broadcast نیست، بلکه multicast می باشد. چرا که یک نفر کانال دو را می بیند و همسایه ی او کانال سه. و همه مجبور نیستند کانال دو را مشاهده کنند.

در روش ۲- سرور یک استریم یکتا برای من پخش می کند.

Multicast به دلایلی هم اکنون رایج نیست. در این مدل خیلی وقت ها نود های دیگری که عضو گروه نیستند درگیر عمل forwarding می شوند که هیچ فایده ای برایشان ندارد. همچنین مسائلی از قبیل اینکه چه کسی مسئول پرداخت هزینه است بطور دقیق حل نشده است. و ساخت و اصلاح درخت پوشا سربار زیادی دارد.

مسیریابی در شبکه های بی سیم اقتضائی (ad-hoc Routing)

شبکه‌های بی‌سیم ادهاک، شامل مجموعه‌ای از گره‌های توزیع شده‌اند که با همدیگر به طور بی‌سیم ارتباط دارند. نودها می‌توانند کامپیوتر میزبان یا مسیریاب باشند. نودها به طور مستقیم بدون هیچگونه نقطه دسترسی با همدیگر ارتباط برقرار می‌کنند و سازمان ثابتی ندارند و بنابراین در یک توپولوژی دلخواه شکل گرفته‌اند. هر نودی مجهز به یک فرستنده و گیرنده می‌باشد. مهم‌ترین ویژگی این شبکه‌ها وجود یک توپولوژی پویا و متغیر می‌باشد که نتیجه تحرک نودها می‌باشد. نودها در این شبکه‌ها به طور پیوسته موقعیت خود را تغییر می‌دهند که این خود نیاز به یک پروتکل مسیریابی که توانایی سازگاری با این تغییرات را داشته، نمایان می‌کند.

در شبکه‌های ادهاک، نودهای شبکه دانش قبلی از توپولوژی شبکه‌ای که در آن واقع اند، ندارند به همین دلیل مجبورند برای ارتباط با سایر نودها، محل مقصد را در شبکه کشف کنند.

سؤال: چرا از روش های مسیریابی که تاکنون در شبکه های سیمی استفاده شده است، در اینجا نمی توان استفاده کرد؟

جواب: به دلیل نرخ بالای تحرک (mobility) و سربار زیاد روش های قبلی به صرفه نیستند. در این شبکه ها مصرف انرژی خیلی مهم می باشد. چون بسیاری از نودها در این شبکه با باتری کار می کنند.

دسته بندی پروتکل های مسیریابی

پروتکل‌های مسیریابی بین هر دو نود این شبکه به دلیل اینکه هر نودی می‌تواند به طور تصادفی حرکت کند و حتی می‌تواند در زمانی از شبکه خارج شده باشد، مشکل می‌باشند. به این معنی یک مسیری که در یک زمان بهینه است ممکن است چند ثانیه بعد اصلاً این مسیر وجود نداشته باشد. در زیر سه دسته از پروتکل‌های مسیریابی که در این شبکه‌ها وجود دارد را معرفی می‌کنیم.

۱. Table Driven Protocols

در پروتکل های از این نوع node ها مدام در حال جستجوی اطلاعات مسیر یابی جدید درون شبکه هستند به صورتی که حتی با تغییر مکان node ها در صورت نیاز به راحتی می توان مسیر مناسبی را یافته و برای ارسال و دریافت اطلاعات بین هر دو node استفاده کرد. به عبارت بهتر می توان گفت که در این شبکه ها مسیر ها از قبل موجود هستند و به محض آنکه node ی اقدام به ارسال داده به node دیگری کند قادر خواهد بود مسیر موجود را از روی اطلاعات از قبل جمع آوری شده شناسایی کرده و مورد استفاده قرار دهد و لذا تاخیری در این مورد متوجه node نیست.

۲. On Demand Protocols

در این نوع پروتکل مسیرها تنها زمانی کشف می شوند که مبدا بخواهد اقدام به برقراری ارتباط با node دیگری کند. زمانی که یک node بخواهد با node دیگری ارتباط برقرار کند بایستی فرایند کشف مسیر (Route Discovery Process) را در شبکه فراخوانی کند. در این حالت قبل از برقرار شدن ارتباط, تاخیر قابل توجهی مشاهده می شود.

۳. Hybrid Protocols

این مورد با ترکیب دو روش قبلی سعی در کاهش معایب کرده و از ویژگی های خوب هر دو مورد بهره می برد.

روش ها و الگوریتم های زیادی در هر دسته از روتینگ های مذکور وجود دارد. در اینجا فقط به تشریح دو روتینگ DSR و AODV می پردازیم که جزء روتینگ های On Demand محسوب می شوند. در هر دوی این روتینگ ها طی یک فرایند اولیه، مسیر مناسبی در شبکه کشف می شود، و می توان از این مسیر برای مدت زمانی استفاده کرد.

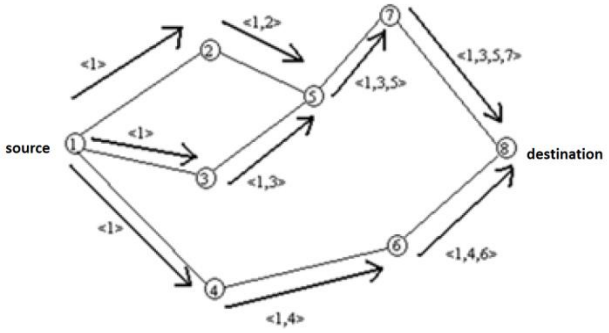
Dynamic Source Routing (DSR):

مراحل زیر انجام می شود:

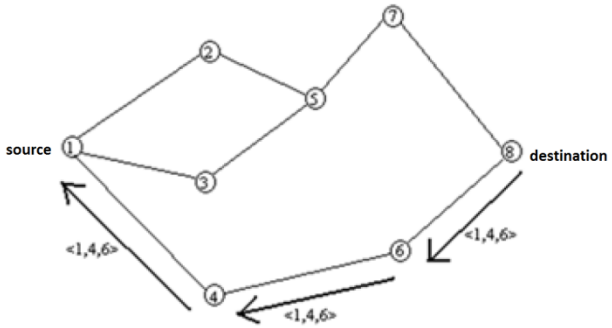
هرگاه یک گره مبداء نیاز به دانستن یک مسیر به مقصد خاصی بود، یک (RRQ) Route Request درست می کند و آن را به روش Flooding پخش می کند تا به همه ی گره های همجوارش برسد.

هر RRQ یک شماره توالی یا #Seq دارد. هر نود میانی وقتی یک RRQ ببیند، ابتدا بررسی می کند که #Seq آن چند است؟ مثلاً اگر یک RRQ با 10= #Seq از نود a دریافت شده باشد، معلوم است که این پیام تکراری است، و آن را دور می اندازد. اما اگر تکراری نبود، آن را در

جدول خود ثبت می کند. در این بسته یک فیلد دیگر به نام TTL هم وجود دارد که در هر ارسال مجدد یکی از آن کم می شود. هرگاه به صفر رسید دیگر بسته ارسال مجدد نمی شود تا از پخش بسته های قدیمی در شبکه اجتناب ورزیده شود.
هر کس یک RRQ غیر تکراری و معتبر دریافت کرد، شماره یا شناسه خود را در آن بسته درج می کند و مجدداً بسته را به سایر همسایه های خود forward می نماید.
اگر هر نود میانی شماره خود را در بسته اضافه کرده و مجدداً بسته را در شبکه پخش کند، بسته نهایتاً به مقصد می رسد. و مقصد دیگر این پیام تقاضا را به دیگران ارسال نخواهد کرد. در شکل زیر نحوه پخش پیام های RRQ و رسیدن آن به مقصد نشان داده شده است.



ممکن است گره مقصد RRQ های مختلفی را با توالی نودهای متفاوت دریافت کند. مقصد با هر متریک دلخواه خود بهترین مسیر را انتخاب کرده و پیام پاسخی تحت عنوان Route Reply (RRP) را از یکی از مسیرهای در دست خود به سمت متقاضی ارسال می کند. اغلب مقصد از مسیری پاسخ را می فرستد که اولین RRQ را دریافت کرده است. چرا که این مسیر به هر دلیلی از لحاظ زمانی کوتاه تر بوده است. مثلاً فرض کنید پیام RRQ ای که از مسیر 1,4,6 به گره مقصد (8) رسیده است از همگی زودتر رسیده است. لذا مقصد می تواند در همین مسیر پاسخ خود را به متقاضی مسیر ارسال کند. همانند آن چه در شکل زیر می بینید.

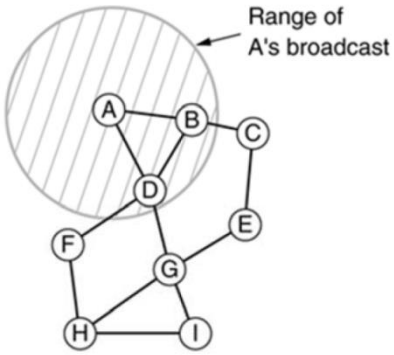


پیام RRP به صورت unicast برای متقاضی ارسال شده و از این به بعد متقاضی (source) می تواند در مسیر مشخص شده به سمت مقصد داده هایش را ارسال کند. یعنی گره فرستنده در هدر بسته های داده ای اش کل مسیر نودهای میانی تا مقصد را مشخص می کند برای همین است که این روش را source routing نامیده اند. چرا که در مبدأ ارسال، کل مسیر مشخص می شود.

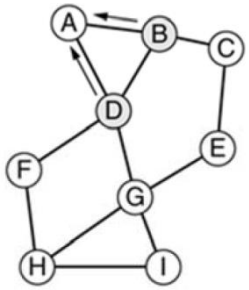
:Ad-hoc On-Demand Distance Vector Routing (AODV)

در روتینگ AODV سعی شده تا ایده جدیدی بکار گیرد که کارایی شبکه را به نسبت روش DSR بالاتر ببرد.
ایده: دانش در مورد بهترین مسیر را به جای اینکه در source نگهداری کنیم، در شبکه نگهداری کنیم. اینکه پکت ها در source بزرگ شوند یک سربار محسوب می شود. در روش DSR اگر شبکه بزرگ باشد و بخواهیم یک پیام کوچک ارسال کنیم و بین مبدأ و مقصد تعداد نودهای میانی زیادی باشند، آنگاه مجبور هستیم اسم همه ی نودها را در هدر بسته بیاوریم و به نسبت سایز داده ارسالی اطلاعات کنترلی زیادی را منتقل کنیم که جالب نیست. AODV برای حل این مشکل پیشنهاد جالبی دارد. که در شکل زیر با یک مثال تشریح خواهد شد. فرایند کشف

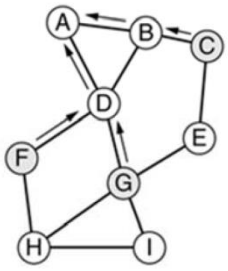
مسیر در AODV مشابه DSR است. در شکل زیر نود A می خواهد برای نود I داده هایی را ارسال کند و نیازمند کشف مسیر می باشد. لذا یک بسته RRQ درست می کند و به گره های همجوار خود یعنی B , D ارسال می نماید. همانند قبل مکانیزمی برای تشخیص پیام های تکراری وجود دارد.



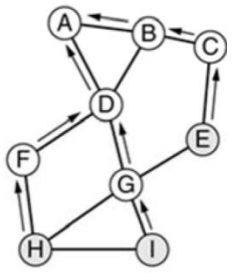
هر گره مثل D و B به محض دریافت یک پیام RRQ در جداول محلی خود اطلاعاتی را ثبت می کنند. مثلاً D وقتی یک تقاضا از A دریافت می کند نتیجه می گیرد که یک مسیر به A دارد و B نیز همین طور. در شکل زیر مشاهده می کنید که این نودها مسیر عکس را برای خود تنظیم می کنند.



نود D پیام RRQ را مجدداً به بقیه مثل G و F ارسال می کند. به عنوان مثال در این جا G و F نیز متوجه می شود که یک راهی به A دارد و آن هم از طریق نود D است. که در شکل زیر نشان داده شده است.



سپس G این پیام را مجدداً پخش می نماید تا به I می رسد. I نیز متوجه می شود که یک مسیر از طریق نود G به A دارد و این اطلاع را در جدولش ثبت می کند. (همانند شکل زیر)



ایده روش این بود که با عبور RRQ در شبکه، نودها یک سری اطلاعات را یاد می گیرند و این اطلاعات را در جداول محلی خود می نویسند تا بعداً برای forwarding در جهت عکس از آن اطلاعات استفاده کنند. در این مثالاز این به بعد:

I می داند که پکتی با مقصد A را از طریق G باید بفرستد.

G می داند که پکتی با مقصد A را از طریق D باید بفرستد.

D می داند که پکتی با مقصد A را از طریق A باید بفرستد.

وقتی پیام RRQ به مقصد رسید، گره مقصد یک پیام پاسخ یا RRP می سازد و برای متقاضی ارسال می کند. ولی برای ارسال لازم نیست کل نودهای میانی را در هدر بسته همانند روش DSR قرار دهد و فقط در بسته A و I (دو نقطه انتهایی) را می نویسد. و آن را دست G می دهد. G می داند که بسته RRP را باید به D تحویل دهد و D نیز می داند که بسته را باید به A تحویل دهد.

همانند زمانی که RRQ در شبکه پخش می شد و نودهای میانی اطلاعاتی را برای خود ذخیره می کردند در حین ارسال RRP نیز همین فرایند صورت می گیرد. به عبارتی در این مثال نود G به هنگام تحویل RRP از I در جدول خود ثبت می کند که یک مسیر به I از طریق نود I دارد. نود D نیز ثبت می کند که یک مسیر به I از طریق G دارد و نود A نیز ثبت می کند که مسیری به I از طریق D دارد. پس با یک بار عبور RRQ و RRP مسیر رفت و برگشت بین دو نود مبدأ و مقصد در شبکه مشخص می شود.

در AODV ممکن است RRP ها از مسیرهای مختلفی به متقاضی برسند که اغلب متقاضی RRP اول را مبنای کار خود قرار می دهد. البته این الگوریتم پیچیدگی های زیادی دارد که فعلاً به آن نمی پردازیم. به عنوان مثال در فرمت بسته های RRP فیلدهایی از قبیل Hop count و life time نیز وجود دارد که می تواند در تصمیم گیری بهترین مسیر دخیل باشند.

در AODV چون دانش در بین نودهای شبکه ذخیره شده است، الگوریتم های بروزرسانی نیز برای آن در نظر گرفته شده است که Route Maintenance نامیده می شوند. چرا که وضعیت نودها پیوسته در حال تغییر است و جداول مسیریابی بایستی مرتب بروز گردند.

Route Maintenance:

نودها بصورت پریودیک به همسایگان خود hello message ارسال می کنند و منتظر دریافت پاسخ می مانند.

- اگر پاسخ دریافت کردند که همه چیز مثل سابق پابرجاست.
- اما اگر پاسخی دریافت نکردند، یعنی یک گره در شبکه از بین رفته است. از بین رفتن یک نود در شبکه ممکن است باعث خرابی یک مسیر شود و یک رخداد مهم تلقی می شود.

مثلاً در همین مثال پیش که مسیر دسترسی A به I بصورت ($A \rightarrow D \rightarrow G \rightarrow I$) بود، فرض کنید D پس از چک کردن همسایگانش متوجه شد که نود G وجود ندارد. به علت اینکه تا الان D بسته های A را که به مقصد I بودند را تحویل نود G می داد و حالا تشخیص داده است که گره G وجود ندارد باید این اطلاع را به A بدهد. به عبارتی به A بگوید برای رسیدن به I دیگر روی من حساب نکن. پس گره ها در شبکه به هنگام تشخیص خرابی همسایگانشان بایستی به نودهایی که برای ارسال به آن ها وابسته هستند خبر بدهند که در پروتکل AODV این کار با ارسال پیام های Route Error صورت می پذیرد. A پس از شنیدن خبر خرابی G در آینده هرگاه احتیاج به ارسال اطلاعات به سمت I داشت مجدداً پی یک فرایند RRQ و دریافت RRP مسیر جدیدی را تنظیم می کند.

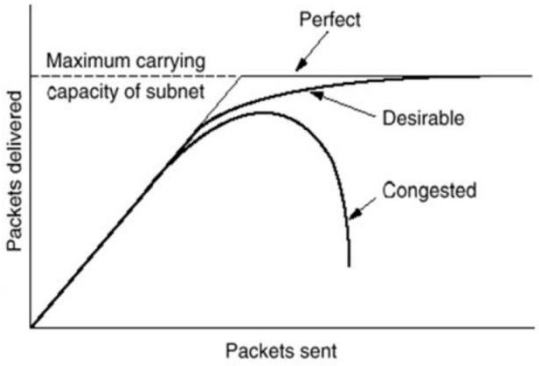
در AODV به علت ثبت اطلاعات در جداول نودهای میانی یک فایده دیگر هم حاصل می شود و آن این است که به هنگام نیاز برای کشف مسیرهای مختلف دیگر، در صورت وجود مسیرهای معتبر در بین نودهای میانی، می توان پیام RRP را زودتر به متقاضی ارسال کرد. برای درک بیشتر موضوع به این مثال توجه کنید:

اگر پس از تنظیم شدن مسیر دسترسی A به I بصورت ($A \rightarrow D \rightarrow G \rightarrow I$) و ثبت اطلاعات در نودهای میانی، گره B یک RRQ برای I داشت، این تقاضا را به نودهای A, D, C می دهد. گره D می داند که همین تازگی یک مسیر برای I درست شده است و خودش نیز جزء مسیر

می باشد. یعنی D می داند که حامل پکت های A به I است. اگر D یک RRQ از گره دیگر مثل B دریافت کند، دیگر لازم نیست RRQ را در شبکه فوروارد نماید. همان جا خودش RRP را ساخته و به B ارسال می کند. در واقع D به B می گوید اگر چیزی برای ارسال به I داری، بده به دست من تا آن را به I برسانم.
پس برای هر مبدأ و مقصدی لازم نیست RRQ و RRP ها از مبدأ به مقصد صورت بگیرد.خیلی وقتا اگر مسیر بروز مناسبی باشد می توان یک RRP برگردد. پس AODV می تواند کاراتر باشد.

کنترل ازدحام (Congestion control):

کنترل ازدحام از حوزه های چالش برانگیز در شبکه های دیتا محسوب می شود. بار ترافیکی وارد شده به این شبکه ها، بسیار پویا و متغیر در زمان است. بنابراین تعیین رفتار شبکه بسیار پیچیده و دشوار خواهد بود.
ابتدا سؤالاتی مطرح می کنیم. ازدحام چیست؟ چرا ازدحام داریم؟ و چرا ازدحام خوب نیست؟
در شکل زیر محور افقی ترافیک وارد شده به شبکه را نشان می دهد و محور عمودی موفقیت شبکه در رساندن پکت ها نشان داده شده است.



معمولاً دو ویژگی را از شبکه انتظار داریم:

۱. با در نظر گرفتن توپولوژی شبکه و ظرفیت لینک های موجود، یک کران بالایی از توانایی شبکه در ارسال پکت ها باید وجود داشته باشد. که در شکل فوق با Maximum carrying capacity of subnet مشخص شده است.
در عین حال می توانیم تصور کنیم هر چه پکت ارسالی در شبکه بیشتر شود، به همان نسبت پکت بیشتری به مقصد خواهد رسید. یعنی انتظار داریم با افزایش حجم ترافیک ورودی، بسته های بیشتری به مقصد برسند. حالت ایده ال این است که این افزایش پیدا کند تا به ظرفیت ماکسیمومی که شبکه می تواند حمل کند برسد.

۲. در دنیای واقعی اتفاق دیگری می افتد. و آن این است که با افزایش ترافیک ورودی شبکه، لزوماً ترافیک بیشتری به مقصد نمی رسد و حتی از یک جایی به بعد ترافیک کمتر به مقصد می رسد. و این ناشی از رخدادهای نامناسب در شبکه است.
مثال: خیابان های یک شهر ظرفیت گذردهی محدودی دارند. ابتدا با افزایش تعداد ماشین ها در خیابان ها، تعداد مسافرت هایی که به مقصد می رسند افزایش می یابند. اما از یک حدی بیشتر ممکن است مسافرت هایی که به مقصد می رسند افزایش پیدا نکند بلکه ممکن است ترافیک قفل شود و هیچ کس به مقصد نرسد. به این پدیده ازدحام یا شلوغی در شبکه می گویند.

دلایل بروز ازدحام در شبکه های دیتا عبارتند از:

- عدم هماهنگی بین اجزای شبکه
- روترها ظرفیت پردازش کافی ندارند.
- خطوط اتصالی بین روترها خیلی کند هستند و ظرفیتشان سریعاً پر می شود.
- روترها از چند نقطه ورودی دارند وقتی همگی می خواهند به یک خروجی داده بفرستند (N to 1) پس در خروجی گلوگاه درست می شود. که به آن Hot-spot نیز گفته می شود.
- توپولوژی شبکه بگونه ای است که بعضی از جاها ظرفیت شبکه به نسبت تقاضا کم است. پس ازدحام بوجود می آید.

نتیجه ازدحام:

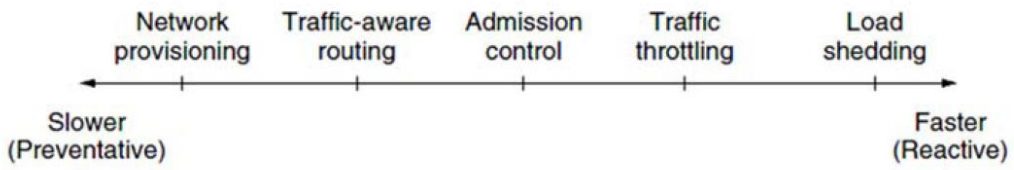
رفتاری که از شبکه می بینید، رفتار خوبی نیست. چرا که منابع شبکه به هدر می رود. چون پکت ها یا به مقصد نمی رسند و یا دیر می رسند و شبکه در بعضی جاها کارایی مورد انتظار خود را برآورده نمی سازد.

تفاوت بین کنترل ازدحام و کنترل جریان:

وقتی به کنترل ازدحام توجه می کنیم، یک نگاه کلان به رفتار شبکه و مؤلفه های آن داریم و مسأله را لزوماً در یک لینک خاص جستجو نمی کنیم. ولی کنترل جریان معمولاً در لایه پیوند داده استفاده می شود و دو سر یک لینک مطرح می گردد. در لایه پیوند داده از عبارت Congestion control یا کنترل ازدحام استفاده نمی شود. اما بعضی وقت ها برای کنترل ازدحام در سطح شبکه از واژه Flow control یا کنترل جریان استفاده می شود. چرا که کنترل جریان یکی از راه های کنترل ازدحام است.

راه حل های کنترل ازدحام:

روش های مختلفی برای کنترل ازدحام وجود دارد. از روش های خیلی خیلی کند گرفته تا خیلی خیلی تند. که در نمودار زیر مقیاس بندی از تکنیک های کنترل ازدحام به تصویر کشیده شده است که از سمت چپ به راست به ترتیب از راه حل های خیلی کند به سمت راه حل های خیلی تند اشاره شده است.



Network Provisioning: از ابتدا طراحی شبکه و تخصیص منابع به گونه ای باشد که شرایط بحرانی کمتر بوجود آید.

Traffic aware routing: در مسیریابی، سعی شود ترافیک از مسیرهای شلوغ مسیردهی نشود.

Admission control: اگر شبکه شلوغ است اجازه ورود ترافیک جدید به شبکه را ندهیم.

Traffic throttling: وقتی ترافیک زیادی از نقطه ای دریافت می کنیم با فیدبک مناسب نرخ آن را پایین بیاوریم. یعنی به فرستنده ترافیک اعلام کنیم که از این به بعد ترافیک کمتری ارسال کند.

Load shedding: در مواقعی که چاره دیگری نداریم پکت های اضافی را دور بریزیم. مثلاً وقتی یک روتر می بیند که شلوغ شده است و توانایی پاسخگویی به پکت های ورودی را ندارد یک سری از پکت ها را دور می اندازد.

راه حل های خیلی کند برای مقیاس بالا در نظر گرفته شده اند. یعنی راه حل های زمان بری هستند که برای مدت زمان بیشتری قابل استفاده هستند (مثلاً در حد ماه و سال) و راه حل های خیلی تند برای مقیاس زمانی کم در نظر گرفته شده اند (مثلاً در حد میکروثانیه).

در کنترل ازدحام بسته به شرایط و ویژگی هایی که با آن مواجه هستیم نوع راه حل ها و مقیاس آن ها می تواند متفاوت باشد.

اصول کنترل ازدحام:

مسأله ازدحام می تواند با دو مدل مکانیزم کنترل شود. که استراتژی های Open loop و Closed loop نامیده می شوند.

مکانیزم های Open loop:

در طراحی اولیه شبکه و تخصیص منابع بگونه ای عمل کنیم که در آینده شرایط بحرانی کمتر بوجود آید. و یا اگر ترافیکی که می خواهد وارد شود در حال حاضر باعث بروز ازدحام می شود، از ورود آن جلوگیری کنیم. در این مدل به وضعیت فعلی شبکه کاری نداریم. بلکه به طراحی های اساسی اولیه توجه می کنیم. این راه حل ها در لایه های مختلف قابل اعمال است.

سیاست های لایه Data link:

به دلیل اینکه عمل retransmition باعث تولید پکت های اضافی در شبکه می شود، می توانیم اعمالی از قبیل تنظیم صحیح Time out را داشته باشیم تا اینکه اشتباهاً فرستنده با بروز time out فرض نکند بسته ها به مقصد نرسیده اند. اتخاذ روش Selective repeat به جای Go back n باعث کاهش ارسال بسته های تکراری در شبکه می شود و همچنین ارسال Ack ها به روش Piggy backing باعث می شود طرفین

پیام های تصدیق دریافت خود را جداگانه ارسال نکنند و آن ها را سوار بر بسته های دیتای خود بفرستند تا از شلوغی شبکه کاسته شود. همچنین سایز پنجره لغزان را می توانیم کمتر نگه داریم تا به فرستنده اجازه ارسال تعداد زیادی پکت را ندهیم تا منجر به شلوغی در شبکه شود.

سیاست های لایه Network:

در مدارمجازی می توانیم با رزرو مناسب منابع، مانع از ازدحام در شبکه شویم. در مسیریابی ترافیک را پخش کنیم و کاری نکنیم که همه بسته ها از نقطه خاصی عبور کنند. در هنگام شلوغی نیز بسته هایی که life-time طولانی دارند را دور بیندازیم تا از این به بعد نیز منجر به شلوغی بیشتر در شبکه نشوند.

سیاست های لایه Transport:

به علت اینکه ترافیک تحویلی به لایه شبکه از طریق لایه انتقال صورت گرفته پس می توان کاری کرد که لایه انتقال ترافیک کمتری به لایه شبکه تحویل دهد. لایه انتقال می تواند تعیین کند یک پیام با چه نرخی یا سرعتی در شبکه تزریق شود. پس با بررسی وضعیت شبکه می تواند جریان ورودی به شبکه توسط لایه انتقال کاهش داده شود. در این لایه نیز انتخاب درست مقدار time out براساس وضعیت ارتباطی شبکه و مستقل از ترافیک موجود می تواند تعیین شود (سیاست open loop) اما بیشتر سیاست های لایه انتقال open loop نیستند. مثل TCP که در فصل بعدی گفته خواهد شد خیلی real time انجام می شود.

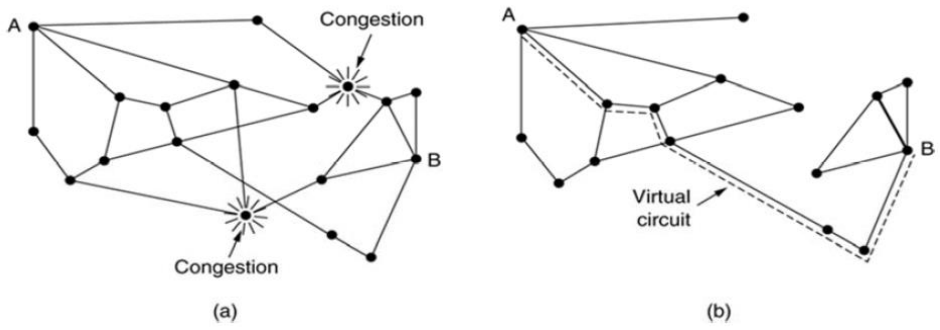
مکانیزم های Closed loop:

روش ها و الگوریتم هایی که بصورت منظم وضعیت شبکه را بررسی می کند و تصمیم می گیرد در هر وضعیت پیش آمده، چه عملی را انجام دهد. به عبارتی با انجام عمل فیدبک از داخل سیستم، رفتار سیستم تغییر کند. مثلاً با اندازه گیری متریک هایی مثل درصد پکت های دور ریخته شده، طول صف ها، تعداد پکت های منقضی شده (time out شده) و میانگین تأخیر و ارائه راه حل های احتمالی از قبیل کاهش بار، افزایش تعداد منابع و ... می تواند مؤثر باشد.

کنترل ازدحام در زیرشبکه های Virtual Circuit و Datagram

کنترل ازدحام در زیرشبکه های Virtual Circuit

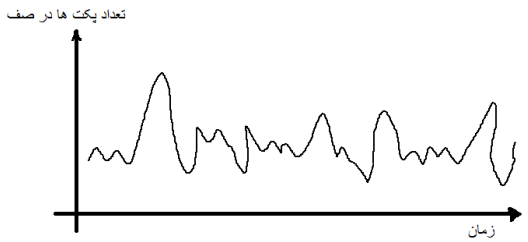
در این زیرشبکه ها، از Admitssion control برای اجتناب از ازدحام استفاده می شود. وقتی یک گره ترافیکی برای ارسال دارد، ابتدا باید بررسی کند که آیا مسیری می تواند پیدا کند که از نقاط بحرانی ازدحام عبور نکرده و به مقصد برسد و الزامات مشتری خود را رعایت کند. اگر چنین مسیری پیدا شد، پس از برپایی مدار اجازه ارسال ترافیک را به نود مربوطه می دهد و در صورت عدم پیدا شدن مسیر مناسب، اجازه این کار را نمی دهد. همچنین در مرحله برپایی مدار مجازی سعی می شود منابع لازم و کافی به ارتباط تخصیص داده شود تا از ازدحام جلوگیری کند. به عنوان مثال فرض کنید A می خواهد یک لینک به B برقرار کند. با توجه به شکل a در دو نقطه شبکه ازدحام وجود دارد. پس مدار مجازی و منابع رزرو شده باید خارج از نقاط ازدحام اتخاذ گردند. همانند شکل b که مسیر مدار با نقطه چین مشخص شده است.



کنترل ازدحام در زیرشبکه های Datagram

در این شبکه ها از قبل، قول و قراری نمی گذاریم. در واقع شبکه ترافیک نودها را گرفته و سعی می کند به مقصد برساند ولی اینکه با چه سرعت و کیفیتی، از قبل مشخص نیست. موضوع ازدحام در این شبکه ها یک خطر دائمی محسوب می شود. سؤالی که مطرح می شود این است که از کجا بفهمیم که ازدحام بوجود آمده است؟ به عبارتی ملاک های تشخیص برای ازدحام چیست؟

به عنوان مثال در شبکه می توانیم رفتار روتر ها را کنترل کنیم. هر روتر تعدادی لاین دارد که این خطوط می توانند هم ترافیک وارد کنند و هم ترافیک را به خارج ارسال کنند. در هر خط نیز با موجودیتی تحت عنوان صف مواجه هستیم که ظرفیت محدودی دارد. یکی از راه ها مونیتور کردن صف یا بافر می باشد. تعداد بسته های موجود در بافر در طول زمان متغیر است. همانند نمودار زیر:



برای اندازه گیری طول بافر می توانیم از مکانیزم های متوسط گیری داینامیک استفاده کنیم. فرمول زیر طول یک صف یا بافر را نشان می دهد که بیانگر ازدحام در آن خط می باشد. یک ضریب α که باعث می شود وزنی از مشاهدات قبلی با تأثیر مقدار باقیمانده از وزن از مشاهده لحظه ای در محاسبه طول صف دخیل باشد.

$$u_{new} = au_{old} + (1-a)f$$

دلیل استفاده از این فرمول این است که یک افزایش لحظه ای در طول صف را نباید به معنی افزایش واقعی بدانیم. روش هایی برای کنترل ازدحام در صورت مشاهده بروز ازدحام وجود دارد که در ادامه بحث می کنیم.

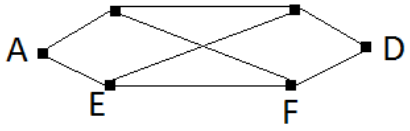
I. Warning bit:

روتر با تنظیم کردن یک بیت هشدار در هدر همه ی بسته هایی که به سمت لینک ازدحام می روند، مشخص کند که دارد دچار ازدحام می شود. وقتی این بسته ها به مقصد می رسند، مقصد با مشاهده این بیت متوجه می شود که این بسته ها از مسیری که ازدحام در آن وجود داشته رسیده اند. پس بهتر است این ترافیک عبوری از مسیر شلوغ کاهش یابد. لذا مقصد یک فیدبک به مبدأ ارسال ترافیک می دهد تا نرخ ارسال خود را کاهش دهد. در واقع یک روتر به دلیل اینکه ترافیک زیادی را برای مصرف کننده های مختلف از خود عبور می دهد دچار ازدحام شده است. پس با تنظیم این بیت هشدار، به همه ی مصرف کننده های ترافیک بطور غیر مستقیم اعلام می کند که ترافیک کمتری برای آن ها جابه جا کند.

در این روش یک مدت زمانی طول می کشد که مقصد متوجه ازدحام شده و به مبدأ اعلام کند تا مبدأ ترافیک ارسالی اش را کم کند. بنابراین این روش با تأخیر همراه خواهد بود. مشکل در راه حل بعدی حل می شود.

II. Choke packets:

اگر در روش قبلی به جای اینکه صبر کنیم تا مقصد به مبدأ فیدبک بدهد، خود روترها به روتر قبلی شان یک پکت کنترلی خاص بفرستند (choke packet) و به آن اعلام کنند که دچار ازدحام شده اند تا روتر قبلی مجدداً درخواست را به روتر قبلی تر برساند و بالاخره به مبدأ ترافیک اعلام گردد که ترافیک کمتری به این مسیر بفرستد. مثلاً در شکل زیر که ترافیکی از سمت A به D در حال انتقال است و D دچار ازدحام شده است، D به F توسط Choke packet اعلام می کند که دچار ازدحام شده، D این بسته را به F و F آن را به E و E آن را به A می رساند. و A در نتیجه نرخ ترافیک خود را کاهش می دهد. در این روش نیز مدتی طول می کشد که اثر کاهش نرخ ترافیک A در نقطه D دیده بشود.



لذا روش بعدی نیز پیشنهاد می گردد.

III. Hop-by-hop choke packets:

در این روش اگر d متوجه بشود ترافیک ورودی اش از F زیاد است، و دارد دچار ازدحام می شود خود D از F مستقیماً درخواست کند که ترافیک کمتری برایش ارسال کند. به عبارتی نرخ ارسالش را کم کند. F که نرخ ترافیک خود را پایین می آورد هر گاه احساس کرد که خود نیز دچار ازدحام شده است، این درخواست را به E بدهد. و E نیز همین طور. ولی در این روش روترها به فضای بافر بیشتری احتیاج دارند.

مقایسه سه روش فوق:

در هر سه این روش ها هرگاه گره ای دچار ازدحام می شود به یک روش خاصی به سایرین اطلاع می دهد که نرخ ارسال اطلاعات خود را کاهش دهند. اما روش اول بصورت خیلی غیر مستقیم، روش دوم اندکی مستقیم و روش سوم کاملاً مستقیم انجام می گردد.

IV. Load Shedding:

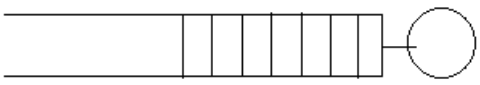
در مدیریت بافرها در خیلی از مواقع چاره ای نداریم جز اینکه ترافیک را دور بریزیم. مثلاً یک روتر در یک پورت خروجی دچار ازدحام شده و فیدبک های لازم را نیز به روترهای قبلی ارسال کرده اما فعلاً شرایطش آنقدر وخیم است که کاری جز دور ریختن ترافیک نمی تواند انجام دهد. سیاست دور ریختن پکت ها می تواند متفاوت باشد. مطالعات زیادی در این حوزه صورت گرفته است که نتیجه آن به شرح زیر می باشد:

۱. دور ریختن پکت هایی که بیشتر از همه در صف بوده اند (old packet). مخصوصاً در کاربردهای realtime مثل پخش ویدئو زنده. چون ممکن است پکت هایی که خیلی تأخیر در صف داشته اند، حتی اگر به مقصد برسند از لحاظ زمانی دیگر اعتبار نداشته باشند.
۲. دور ریختن پکت هایی که جدیداً وارد صف شده اند. مخصوصاً در کاربردهایی مثل انتقال فایل.
۳. استفاده از اولویت در پکت های مختلف. بطوریکه پکت ها با اولویت کم برای دورانداخته شدن انتخاب شوند. مثلاً در ATM یک فیلد یک بیتی مبین اولویت وجود دارد که در کاربردهای مهم آن را بالا می گذارند و در پکت IP فیلد Type Of Service مشخص کننده نوع سرویسی است که باید به این بسته داده شود که در کلاس های مختلف طبقه بندی می شوند.

V. Random Early Detection:

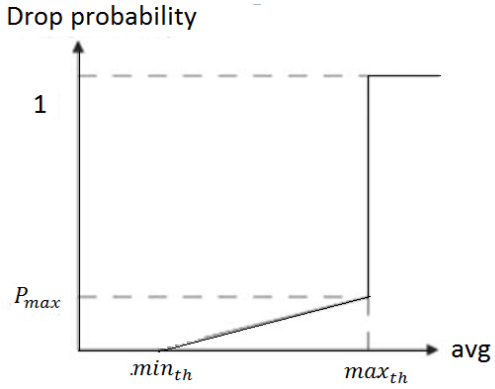
این روش مبحث جالبی در مدیریت صف ها یا بافرها می باشد. یک سری از مطالعات نشان داده اند در شرایطی که شبکه به سمت ازدحام پیش می رود خیلی بهتر است که هر چه زودتر تشخیص دهیم که به بحران نزدیک می شویم و این بحران را زودتر مدیریت کنیم و اگر لازم باشد چیزی دور ریخته شود، زودتر دور ریخته شود. یعنی صبر نکنیم صف پر شده و بعد بسته ها را دور بریزیم.

فرض کنید یک صف داریم که پکت ها در آن قرار می گیرند. طول این صف را که مبین تعداد پکت های موجود در صف است را اندازه می گیریم. در این اندازه گیری مقدار لحظه ای را اندازه نمی گیریم. بلکه متوسط گیری می کنیم.



نمودار زیر را در نظر بگیرید. محور افقی شاخصی است که از متوسط گیری روی طول صف بدست می آید و محور عمودی احتمال drop کردن پکت هایی است که در این صف وجود داشته اند. min_{th} و max_{th} دو حد هستند که در تصمیم گیری ها لحاظ می شوند. روش بدین صورت است:

اگر از یک اندازه ای (min_{th}) طول صف کوچکتر بود، لازم نیست هیچ پکتی دور ریخته شود. به عبارتی احتمال drop کردن پکت ها صفر است. اگر این اندازه از min_{th} بالاتربرود، مطابق اندازه اش طبق نمودار، متناظر با آن در محور عمودی عددی است که احتمال دور ریختن پکت ها را نشان می دهد (هرچه متوسط طول بافر بالا برود احتمال دور ریخته شدن بسته ها نیز بصورت خطی می رود). اما اگر این متوسط طول بافر به حد بالایی(max_{th}) رسید به احتمال ۱ یا صد درصد پکت ها را دور می ریزیم.

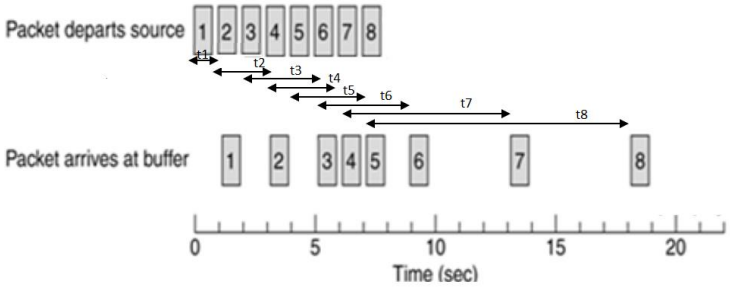


در این نوع نگاه سه نوع پارامتر کنترلی داریم. P_{max} و max_{th} و min_{th}. می توانیم به P_{max} و min_{th} مقادیر مختلفی نسبت دهیم و رفتارهای مختلفی را در شبکه مشاهده کنیم اما max_{th} تابع و وابسته به حافظه صف است و نمی توان آن را به دلخواه تغییر داد.

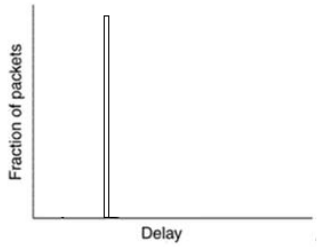
Jitter Control:

در کاربردهای پخش صدا و ویدئو، نوسان در تأخیر (jitter) بایستی کنترل شود. برای کنترل jitter در شبکه ها بایستی پکت ها، اطلاعات زمانی را به همراه داشته باشند. در هر روتر با توجه به برچسب زمانی هر بسته چک می شود که یک بسته چقدر زودتر یا دیرتر از زمانبندی خود به آن جا رسیده است. اگر زودتر از موعد رسیده باشد که برای مدتی نگه داشته می شود و مشکلی پیش نمی آید. اما برای آن هایی که دیر می رسند گیرنده منتظر می ماند و این امر یکنوایی دریافت استریم صدا یا ویدئو را بهم می زند.

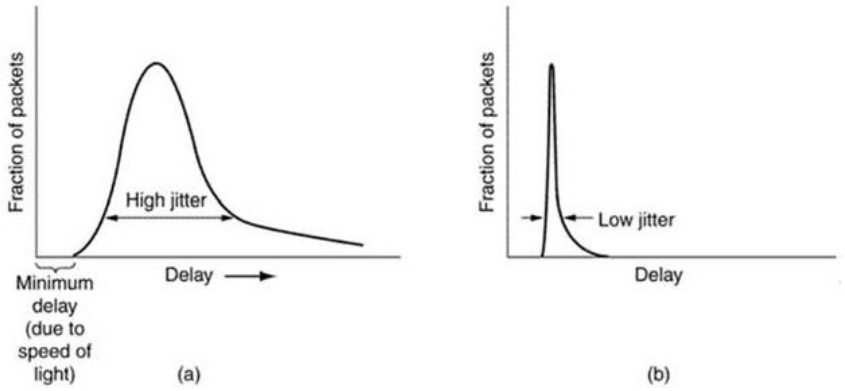
در شکل زیر فرض کنید از یک مبدأ هشت پکت ارسال می شود اما همگی بصورت یکنواخت و با تأخیر یکسان نمی رسند. در شکل مدت زمانی که طول می کشد هر پکت به مقصد برسد با t1, t2, t3, … , t8 نشان داده شده است. بعضی زمان ها خیلی کوتاه و بعضی زمان ها خیلی بلند هستند. اگر به دنباله عددی t1, t2, t3, … , t8 به عنوان یک رخداد آماری نگاه کنیم و نمودار هیستوگرام آن را بکشیم مشاهده می کنیم پکت ها با اختلافات زمانی به مقصد می رسند.



اگر همه پکت ها با تأخیر یکسان به مقصد برسند نمودار بصورت زیر خواهد بود.



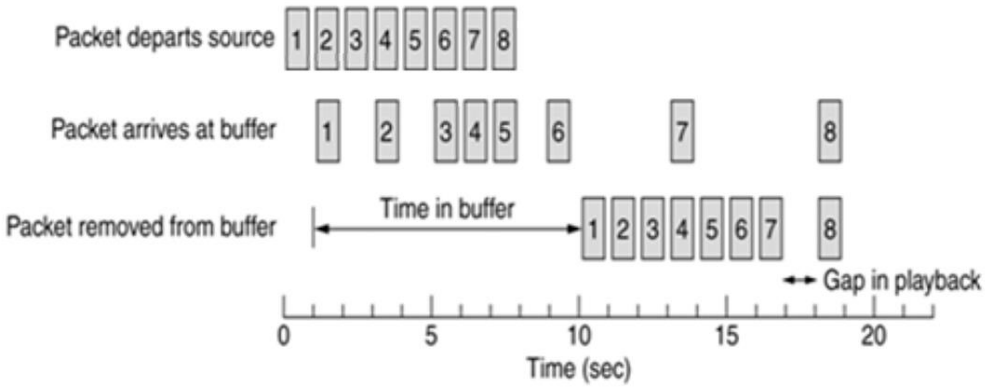
اما واقعیت چیز دیگریست و بصورت آنچه در نمودارهای زیر می بینید خواهد بود.



در شکل a زمان تأخیر پکت ها اختلاف زیادی دارند اما در شکل b تأخیر پکت ها با اختلاف کمی حول یک عدد متمرکز است. در کاربردهای Realtime، پدیده jitter خیلی نامناسب است. اگر تأخیر همگی یکنواخت باشد یک مدت زمان اولیه صبر می کنیم تا اولین پکت برسد و از آن به بعد بقیه پکت ها را دریافت خواهیم کرد. اما اگر این تأخیر از یکنوایی برخوردار نباشد گاهی اوقات بسته ها زودتر از انتظار و بعضی اوقات دیرتر از مدت زمان مورد انتظار می رسند که این امر باعث به هم زدن یکنوایی پلیر در گیرنده خواهد شد.

راه حل:

یک trade off بین delay و jitter داشته باشیم. در کاربردهای realtime وقتی بر روی فیلم یا صدایی کلیک می کنید ابتدا مدت زمانی در انتظار می مانیم تا عمل buffering صورت پذیرد و در این مدت هیچ چیزی را پخش نمی کند. اما همیشه این کار مشکل را حل نمی کند. مثلاً در همین مثال فعلی در شکل زیر که پکت ها در زمان های غیر یکنواختی به مقصد رسیده بودند با عمل بافرینگ و تأخیر اولیه تا حدی موفق شده است پکت های ۱و ۲و ۳ و ۴ و ۵ و ۶ و ۷ به خوبی پخش کند اما پکت ۸ آنقدر دیر به مقصد می رسد که باز هم به اندازه یک واحد زمانی gap در پخش آن وجود دارد که پلیر سعی می کند مجدداً بافرینگ را انجام داده تا بعداً پخش کند.



بنابراین می شود از delay هزینه کرد تا jitter تا حدی پوشانده شود. اما شرایط در بعضی جاها اجازه نمی دهد از delay هزینه کرد. زیر ا خود delay هم مهم می باشد. بطور مثال وقتی دو طرف می خواهند با هم از طریق VOIPصحبت کنند هم jitter و هم delay اهمیت دارند. سخت ترین مثال ها کاربردهای تعاملی است. مثال دیگر آن video conference است. در این مثال ها کیفیت سرویس از شاخص های خیلی مهم محسوب می شود. که راجع به کیفیت خدمات (QOS) در ادامه بحث می کنیم.

کیفیت خدمات (Quality of Services):

در تمامی شبکه های کامپیوتری پیشرفته تکنیک هایی متعدد وجود دارد که تمرکز ویژه ای بر روی تضمین کیفیت خدمات (QOS) دارند. خیلی وقت ها مفهومی که ما به عنوان مصرف کننده از کیفیت خدمات مد نظرمان است با آنچه در دنیای شبکه اتفاق می افتد تفاوت دارد. مثلاً اگر در مورد کیفیت سرویس اینترنت adsl از افراد مختلف بپرسید هر کدام از نقطه نظر خاصی اظهار نظر می کنند. اما در شبکه بیشتر به

پارامترهای تکنیکی توجه می شود. این نیازها با چهار پارامتر قابلیت اطمینان[۲] ، تاخیر[۳]، لرزش یا نوسان در تأخیر[۴]، و پهنای باند[۵] مشخص می شوند. راهکارهای مختلف دستیابی به کیفیت خوب در لایه های مختلف مطرح شده است. بسته به کاربرد مورد نظر ممکن است تعدادی از این پارامترها مهم تر از سایرین باشد. مثلاً همانطور که در جدول زیر مشاهده می کنید، می بینید که در ارسال ایمیل، کنترل delay, jitter و bandwidth اصلاً مهم نیست. به عنوان مثل فرقی بین ۲ میلی ثانیه و ۵ میلی ثانیه وجود ندارد. اما برای video on demand پارامتر jitter و bandwidth مهم است و delay مهم نیست. چون در ابتدا اشکال ندارد اندکی تأخیر وجود داشته باشد. همچنین reliability برای ویدئو خیلی مهم نیست به عبارتی لازم نیست نگران باشیم که همه ی پکت ها برسند. اما در ایمیل خیلی مهم است که دقیق برسد.
این که بخواهیم برای همه ی کاربردها همه ی شاخص های ذکر شده را مهم بپنداریم کار خوبی نیست. چون سربار زیادی به سیستم تحمیل می شود. به عنوان مثال لازم نیست برای ایمیل، jitter کنترل شود.

نوع کاربرد	قابلیت اطمینان	تأخیر	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 بر حسب تقاضا	پایین	پایین	بالا	متوسط
دریافت ویدئو بر حسب تقاضا	پایین	پایین	بالا	بالا
تلفن اینترنتی	پایین	بالا	بالا	پایین
ویدئو کنفرانس	پایین	بالا	بالا	بالا

سطح نیازمندی برنامه های کاربردی به کیفیت خدمات

بنابراین شناخت کاربردها و ماهیت آن ها خیلی مهم است و باید روی نیازمندی های هر کاربرد مطالعه داشته باشیم.
در ارائه کیفیت خدمات سه حوزه نسبتاً مهم وجود دارد:

- ✓ Reliability: این که پکت اصلاً می رسد یا نه؟
- ✓ Troughput / rate: نرخ انتقال پکت چقدر است؟
- ✓ Delay / jitter: زمان رفتن پکت چگونه است؟

برای تضمین یا کنترل کیفیت خدمات چه باید کرد؟
QOS به شدت به کنترل ازدحام وابسته است. چون اگر ازدحام وجود داشته باشد نمی توانیم راجع به delay , jitter, rate صحبت کنیم. به عبارتی ارتباط تنگاتنگی بین این دو برقرار است. برای کنترل ازدحام و کنترل jitter راهکارهایی را تاکنون مرور کردیم. در ادامه به مباحثی تحت عناوین Over-provisioning ، Traffic Shaping، Resource Reservation، Admission Control، Packet Scheduling می پردازیم.

Over-provisioning:

ظرفیت های زیادی را در شبکه از قبل مهیا سازیم. مثلاً روتری را بخریم که از نیازمان خیلی بالاتر عمل کند. این راه حل اقتصادی نیست. اغلب وقتی کسی کار مهندسی بلد نیست از این روش استفاده می کند.

شکل دهی به ترافیک (traffic shaping):

از کارهایی است که در شبکه جلوی رخداد ازدحام را می گیرد. مفهوم ساده ای است که به هنگام مواجهه با حجم ترافیک زیاد در یک نقطه خاص، سرویس دهی به آن ترافیک را کنترل و مدیریت می کند. به عنوان مثال زمانی که شما ترافیک زیادی را در یک روتر از پورت خاصی

[2] reliability
[3] delay
[4] jitter
[5] bandwidth

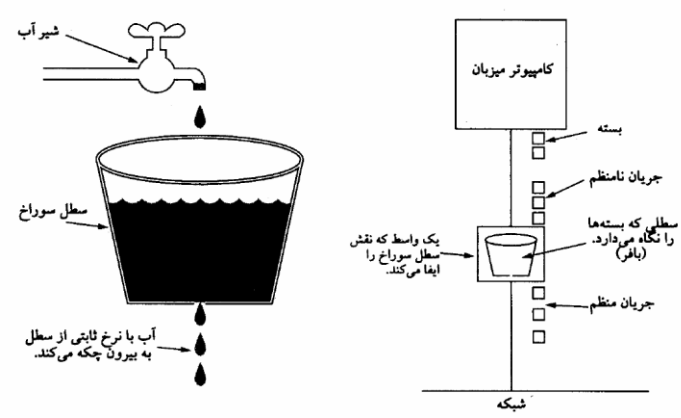
بخواهید به یک مقصد ارسال کنید، اگر ترافیکی که شما می خواهید ارسال کنید از ظرفیت لاینی که می خواهد این ترافیک را عبور دهد بیشتر باشد، در آن پورت ازدحام بوجود می آید. traffic shaping این مشکل را مدیریت می کند تا ازدحام بوجود نیاید. عمل traffic shaping خصوصاً در شرایطی لازم است که سرویس دهنده و سرویس گیرنده می خواهند در مورد ویژگی های نوع سرویس ترافیکی شان مذاکره کنند و قرارداد امضا نمایند که به آن SLA: Service Level Agreement می گوییم. مثلاً به عنوان یک مشتری خدمات اینترنت از یک ISP سرویس اینترنت با سرعت 128 k یا 512 k خریداری می کنیم. به این معنی که توقع داریم ۱۲۸ کیلوبیت بر ثانیه یا ۵۱۲ کیلوبیت بر ثانیه دیتای ما عبور کند. حال یک مکانیزمی وجود دارد که به هنگام آپلود یا دانلود، نرخ تبادل داده ای که از طرف کامپیوتر ما می خواهد صورت بگیرد با سرعت ۱۲۸ یا ۵۱۲ تنظیم می شود. به این فرایند traffic shaping می گویند.

انواع traffic shaping:

Leaky bucket

الگوریتم سطل سوراخ دار(Leaky bucket) یکی از الگوریتم هایی است که در جهت افزایش کیفیت خدمات و کاهش ازدحام و جلوگیری از تحمیل بار اضافی توسط مشتری ها یا میزبان ها بر شبکه طراحی شده است.

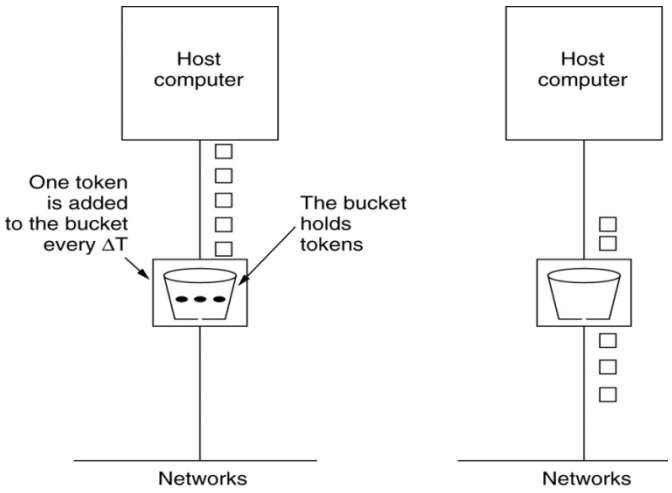
یک سطل آب را تصور کنید که کف آن یک سوراخ دارد. بنابراین آب در صورت وجود با نرخ ثابتی از آن بیرون می آید. ورودی این سطل یک شیر آب است که می تواند آب زیاد، آب کم یا هیچ آبی در ظرف بریزد. این سطل ظرفیت محدودی دارد که می تواند حجمی از آب را در خود نگه دارد. تا زمانی که حجمی از آب در سطل وجود دارد، آب با نرخ ثابتی از کف سطل خارج می گردد و اگر هم آبی در سطل نباشد، چیزی از آن خارج نمی شود. این که شیر آب چه وقت باز و بسته می شود و تا چه حد آب وارد سطل می کند کاملاً مستقل از خصوصیات سطل است. بنابراین نرخ ورودی آب به سطل می تواند نوسانات زیادی داشته باشد. حال این مثال در دنیای شبکه های کامپیوتری نیز به این صورت تفسیر می شود: فرض کنید ترافیک نامنظمی را که یک کاربر ارسال می کند که آن را با قطره های نامنظم و تصادفی که به یک سطل سوراخ دار وارد می شوند مدل کنیم. همچنین فرض کنید یک فضای خاص با حجم معین به آن مشتری اختصاص دهیم (مدل این بخش از بافر، سطلی است که ظرفیت مشخصی دارد و در صورت پر شدن سرریز شده و بار اضافی ورودی دور ریخته می شود). همانطور که اندازه سوراخ زیر سطل ثابت است و قطرات از خروجی به طور منظم و با نرخ ثابت از سطل خارج می شوند، ترافیک کاربر نیز می تواند با نرخ ثابتی از بافر خارج گردد.



پس در این جا نرخ خروج آب از سطل همان نرخ سرویس دهی به بافر می باشد و حجم آب قابل ذخیره در صف نیز همان حجم بافر تصور می شود.

Token bucket

سطل را در نظر بگیرید که ثانیه ای یک توکن در آن اضافه شود. این سطل به یک اندازه یا ظرفیتی قابلیت نگهداری توکن ها را دارد. هر گاه پکتی وارد سطل می شود و می خواهیم آن را از سطل به بیرون ارسال کنیم، لازم است در ازای هر پکت یک توکن مصرف شود. اگر در مدت زمان مشخصی هیچ پکتی از سطل رد نشود، توکن ها در آن انبار می شوند اما به محض اینکه پکت های زیادی یک دفعه وارد سطل می شود می توان به تعداد توکن ها، پکت ها را از سطل خارج کرد. و اگر پکت دیگری ماند و توکن دیگری وجود نداشت باید صبر کرد تا در ثانیه بعدی به ازای هر توکن تولید شده یک پکت ارسال و آن را مصرف کنیم.

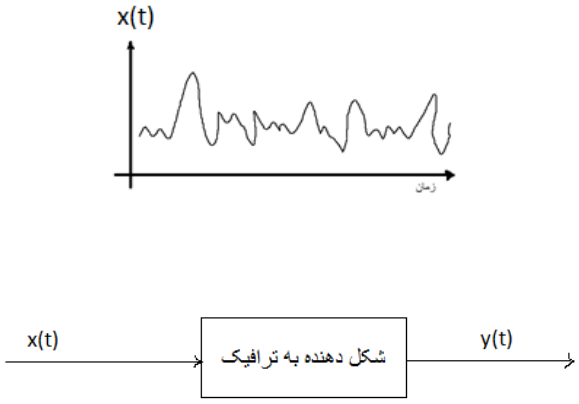


به عبارتی در این روش می توان ظرفیت اجازه داده شده را در زمان هایی که ترافیکی برای ارسال نداریم تا حد مشخصی برای زمان هایی که ترافیک لحظه ای بالایی داریم ذخیره کرده و بعداً استفاده نماییم.

تفاوت دو روش فوق:

در Leaky bucket اگر ترافیک ورودی نداشته باشیم ترافیک خروجی نیز نداریم. و مهم هم نیست چه مدت ترافیک نداریم. و هر موقع ترافیک وارد شد با همان نرخ مشخص ثابت، از سطل خارج می شود. اما در tocken buckets اگر مدتی ترافیک ورودی نداشتیم توکن ها ذخیره می شوند و وقتی بعداً ترافیک ورودی داشته باشیم می توانیم از ظرفیت توکن های ذخیره شده استفاده کنیم و بعد از مصرف توکن های ذخیره شده مطابق با نرخ توکن گذاری بسته ها به ارسال ترافیک بپردازیم.

تفاوت این دو روش به زبان ریاضی بدین صورت است که اگر ترافیک ورودی به سطل را x(t) بنامیم و آن را وارد یک شکل دهنده ترافیک کنیم، ترافیک خروجی y(t) حاصل خواهد شود.

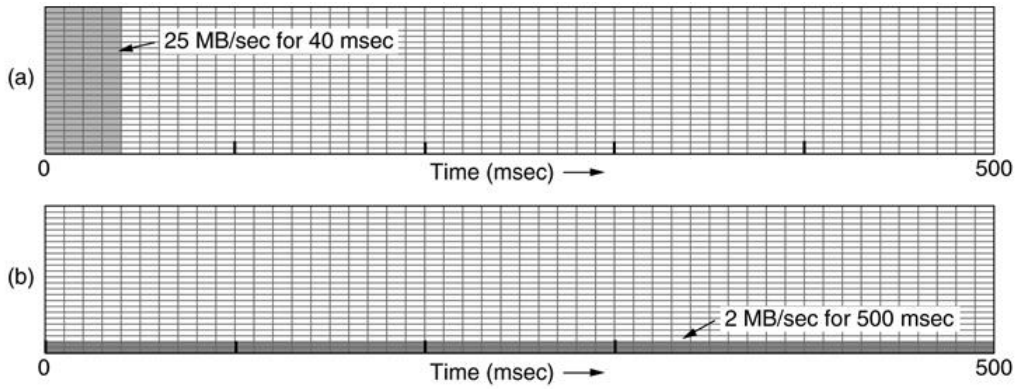


شکل دهنده ترافیک می تواند از هر کدام از دو نوع ذکر شده باشد. در مورد y(t) دو پارامتر داریم: $max_{y(t)}$ و $avg_{y(t)}$. کاری که Leaky bucket انجام می دهد کنترل کردن $max_{y(t)}$ و کاری که tocken buckets انجام می دهد کنترل کردن $avg_{y(t)}$ است.

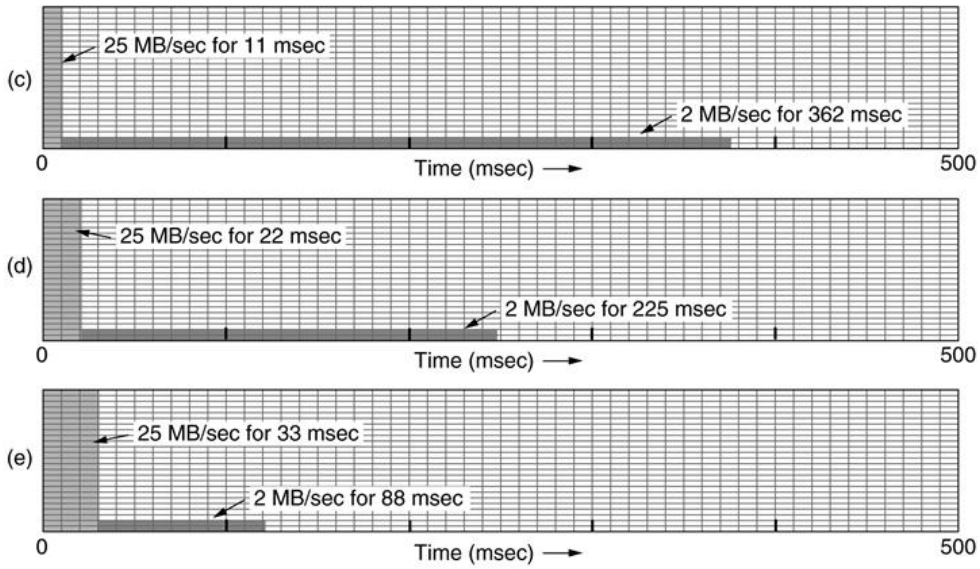
Hybrid solution (راه حل ترکیبی)

خیلی وقت ها بصورت ترکیبی از دو روش قبل استفاده می کنیم. ابتدا token buckets و سپس Leaky bucket استفاده شود. که هم $avg_{y(t)}$ و هم $max_{y(t)}$ کنترل شود.

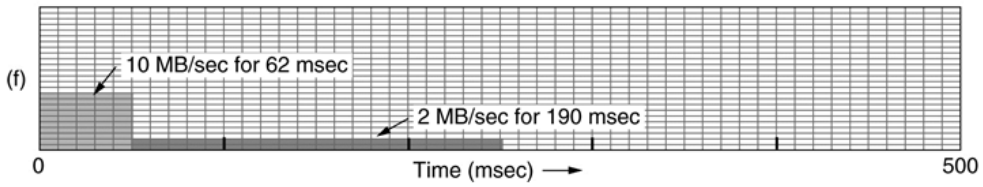
شکل زیر نمایشی از عملکرد ترکیبی را ارائه می دهد. در شکل a ترافیک ورودی25 MB/sec به سطل سوراخ داری را نشان می دهد که در مدت زمان 40 msec وارد شده است. شکل b ترافیک خروجی از این سطل سوراخ دار را نشان می دهد. همان طور که مشاهده می کنید نرخ ترافیک خروجی بصورت ثابت و به اندازه 2 MB/sec است که به مدت زمان 500 msec طول کشیده است تا از سطل خارج گردد.



در شکل c خروجی ترافیک ورودی نشان داده شده در شکل a را، بعد از عبور از Tocken bucket با ظرفیت های 250 KB، در شکل d خروجی ترافیک ورودی، بعد از عبور از Tocken bucket با ظرفیت 500 KB و در شکل های e خروجی بعد از عبور از Tocken bucket با ظرفیت750 KB نشان داده شده است. به علت ظرفیت کم c، شاهد یک burst کوچکتر و در d یک burst بزرگتر و در e یک burst خیلی بیشتری هستیم.



خروجی شکل d مجدداً پس از عبور از یک leaky bucket با حداکثر خروجی 10 MB/sec در شکل f به تصویر کشیده شده است. بنابراین هیچ گاه ترافیک بالاتر از 10 MB/sec را نخواهیم داشت.



Data path و Control path :

عملکرد ساده این Traffic shaper ها از اهمیت بالایی برخوردار است. چون این ها مؤلفه های سیستمی هستند که در مسیر Data Path سیستم قرار دارند. قبل از پرداختن به راهکار بعدی در ارائه کیفیت خدمات به توضیحی راجع به دو مفهوم Data path و Control path می پردازیم که در سیستم های مخابراتی یا شبکه های کامپیوتری دو عملکرد خیلی مهم هستند.

عملکردهایی که به هنگام جابجایی ترافیک با آن مواجه هستیم عملکردهای Data path نامیده می شوند. مثل روتینگ، که بایستی در مورد همه ی پکت هایی که وارد روتر می شود در مورد مسیریابی شان تصمیم گیری شود. مثال دیگر Traffic shaping است که بایستی در خصوص همه ی ترافیک ورودی تصمیم گیری کرد که با چه نرخی به بیرون ارسال شوند. پس Routing و Traffic shaping هر دو از عملکردهای Data path هستند.

Control path به مجموعه کارهایی که نودهای شبکه باید انجام دهند تا کنترل نحوه عملکرد خود را به خوبی داشته باشند، گفته می شود. مثلاً در بخشی از روتینگ های LSR و DVR ، مجموعه پیام های کنترلی است که بروزرسانی جداول مسیریابی را انجام می دهند. و این ارسال و دریافت پیام های کنترلی ربطی به حجم ترافیک ورودی ندارد. به عنوان مثال روتری با ده پورت 10 GB/s داریم. آیا این روتر لازم است به اندازه 100GB دیتا جابجا کند تا جداول مسیریابی اش تنظیم گردد؟ پاسخ منفی است. چرا که این روتر ممکن است هر 5ms بر روی هر یک از پورت هایش یک پکت hello ردو بدل کند تا امور مربوط به جداول و توپولوژی شبکه را انجام دهد.

عملیات Control path در بسیاری از شبکه ها، عملیات خیلی complex تر، ولی در موقع اجرا نرخ اجرای کمتری دارد. بیشتر عملیات Control path در نرم افزار صورت می گیرد.

اما عملیات Data path عملیاتی است که در ازای همه ی ترافیک های ورودی به یک روتر بایستی انجام شوند. پس نرخ اجرای بالایی دارند. به عنوان مثال باید در خصوص همه ی دیتایی که وارد یک پورت می شود عمل traffic shaping صورت گیرد. بیشتر پیاده سازی های Data path در سطح سخت افزار صورت می گیرد. برای همین هزینه کمتری دارد و مقیاس پذیری بالاتری هم دارد.

نکته قابل توجه: وقتی می خواهیم در حوزه Data path روشی ارائه دهیم بایستی سادگی روش را مدنظر داشته باشیم. چرا که قرار است در آینده عملیات پیچیده ای با حجم بالا و در مقیاس های بزرگ توسط این روش ها انجام شوند. پس سعی می کنیم عملکردهای Data path ساده طراحی شوند اما در مورد Control path چنین محدودیتی نداریم.

رزرو منابع (Resource reservation):

تامین کیفیت خدمات باید بین مبدأ و مقصد چیزی شبیه به یک مدار مجازی ایجاد و تنظیم شود و تمام بسته ها از این مسیر حرکت کنند. اگر برای جریان داده ها مسیر ویژه داشته باشیم می توان منابع لازم (پهنای باند، فضای بافر و سیکل های CPU) را در طول این مسیر رزرو کرده و موجود بودن ظرفیت مورد نیاز را تضمین کرد. هر صف یک بافر دارد و یک پردازنده که به پکت ها سرویس می دهد. پکت ها در این صف می نشینند و با یک نرخ ثابت به آن ها سرویس داده می شود تا از صف خارج گردند. اختصاص دادن منابع کار خیلی آسانی نیست. چرا که پارامترهای ذکرشده مستقل از هم نیستند و با هم تعامل دارند.

مثال:

در یک سیستم اگر متوسط ظرفیت پردازش $\mu = 10^6$p/s و ورود پکت ها به صف از توزیع پواسون با متوسط زمان ورود $\lambda = 0.95 \times 10^6$ پیروی کند، متوسط تأخیری که در صف برای پکت ها بوجود می آید از رابطه زیر حاصل می شود:

$$\rho = \frac{\lambda}{\mu} \Rightarrow T = \frac{1}{\mu} \times \frac{1}{1-\rho} = 1\mu\text{sec} \times \frac{1}{1-0.95} = 20\mu\text{sec}$$

معمولاً زمانی که تأخیر پکت ها در صف μsec 20 و متوسط طول پکت ها (زمانی که هر پکت وارد به بافر می شود) در صف μsec 1 باشد، پس بطور متوسط در این صف 20 پکت وجود دارد. بنابراین اگر صفی داریم که قرار است 20 پکت را در خود نگه دارد، برای احتیاط سایز صف را طوری در نظر می گیریم که 40 پکت را در خود نگهداری کند. از محاسبات نتیجه می گیریم طول بافر چقدر باشد، خوب است.

مثال:

در مثال قبلی اگر متوسط زمان ورود را به جای $\lambda = 0.95 \times 10^6$ افزایش دهیم تا $\lambda = 0.99 \times 10^6$ شود، چه اتفاقی می افتد؟

$$\rho = \frac{0.99}{1} \quad T = 100\mu\text{sec}$$

یعنی طول صف ۵ برابر حالت قبل می شود. پس یک تغییر جزئی می تواند نتیجه خیلی متفاوتی را در طول بافر ایجاد کند.

کنترل فضای بافر از اهمیت زیادی برخوردار است. برای درک بیشتر این موضوع روتری را فرض کنید که از پورت های مختلف، حجم ترافیک بالایی در واحد زمان به آن وارد می شود. چرا که نرخ خطوط متصل به روتر بالا است. ترافیک ورودی در روتر باید جایی نوشته شود و مجدد

خوانده شود و یک پردازش روی آن صورت گیرد و نهایتاً از طریق یکی از خروجی ها به بیرون ارسال گردد. قاعدتاً حافظه هایی که بتوانند در این نرخ ها کار کنند حافظه های معمولی نیستند. فضای بافر یک نوع RAM است. دلیل اهمیت این حافظه ها به دلیل نوع آن می باشد که از نوع حافظه های Control path نیستند. بلکه این حافظه ها از نوع Data path می باشند تا قادر به کارکردن در نرخ های بالا باشند. از این رو این حافظه ها خیلی هم گران می باشند. اموری که در حوزه Data path عمل می کنند بایستی قادر به کارکردن در نرخ های بالا باشند.

کنترل پذیرش (Admission Control):

فرایندی است که در آن منابعی برای یک مسیر ترافیک رزرو می شوند. یعنی در تمام طول مسیری که یک Flow می خواهد جریان داشته باشد، تمامی منابع را برای آن رزرو کنیم.

مفهوم Flow:

به مجموعه ای از داده ها از یک مبدأ خاص به یک مقصد خاص و با یک پروتکل خاص ارسال می شود Flow گفته می شود. مثلاً در دریافت یک ویدئو آنلاین، Stream آمده از سرور به متقاضی را Flow می گویند.

در این فرایند قبل از ارسال یک Flow به سمت مقصد منابع لازم یا ظرفیت موردنیاز در طول مسیر، پیش بینی و رزرو می شوند. با چنین فرضی، هر گاه جریانی از بسته ها به یک مسیریاب تسلیم شود بر اساس ظرفیت موجود خود و سطح تعهداتی که در خصوص دیگر جریان ها پذیرفته، باید در خصوص قبول یا رد آن تصمیم بگیرد.

از آنجایی که برای رسیدن به توافق نهایی در خصوص تامین نیازهای یک Flow باید مؤلفه های متعددی در مذاکرات شرکت داشته باشند (اعم از فرستنده، گیرنده و تمام مسیریاب های واقع بر روی مسیر)، لذا هر جریان باید بر حسب پارامترهای مشخصی به دقت توصیف شود تا بتوان بر روی این پارامترها مذاکره و توافق کرد. مجموعهٔ چنین پارامترهایی اصطلاحاً Flow Specification نامیده می شوند. تصمیم گیری در خصوص پذیرش یا رد یک جریان باید بر حسب پارامتر های زیر باشد:

- نرخ Tocken bucket
- سایز Tocken bucket
- ماکسیموم نرخ داده
- کوچکترین سایز پکت دیتا
- بیشترین سایز پکت دیتا

مثلاً در ویدئو بعضی مواقع شبکه باید، یک burst از پکت ها را پخش کند. اگر نکند ویدئو به درستی مشاهده نمی شود. پس سایز Tocken bucket مهم می باشد. همچنین تفاوت بین این که یک میلیون بیت در بسته های ۱۰۰۰۰ بایتی باشد یا ۱۰۰۰ بایتی باشد را در نظر بگیرید؛ حالتی که ۱۰۰۰ بایتی باشد، تعداد پکت بیشتری تولید می کند و هر پکت نیاز به روتینگ دارد و منبعی از پروسسور را استفاده می کند. پس کوچکترین سایز پکت دیتا می تواند مهم باشد. همچنین بیشترین سایز پکت دیتا در خصوص تصمیم گیری برای طول بافرها می تواند تعیین کننده باشد.

بنابراین اگر شبکه قادر باشد این الزامات را رعایت کند اجازه انتقال این Flow را می دهد و به عهد خود نیز پایبند است.

زمان بندی بسته ها (Packet schedualing):

هرگاه یک مسیریاب هدایت چندین ترافیک را بر عهده داشته باشد این خطر وجود داردکه یک جریان از حدود و ظرفیت مجاز خود تجاوز نماید و در نتیجه جریان های دیگر را با کمبود منابع مواجه سازد. اگر پردازش بسته ها به ترتیب ورودشان انجام گیرد باعث می شود که یک فرستنده متجاوز بتواند بیشتر ظرفیت مسیریاب هایی را که بر روی خط سیر بسته های او هستند اشغال کرده و کیفیت خدمات دیگران کاهش یابد. برای خنثی کردن چنین تلاشی، الگوریتم هایی جهت زمان بندی بسته ها پیشنهاد شده است که Fair Queuing و Weighted Fair Queuing نامیده می شوند.

Fair Queuing:

در این الگوریتم مسیریاب ها باید برای هر خط خروجی و به ازای هر Flow که از آن خط خروجی می گذرد، صف های جداگانه ای تشکیل بدهند. هر گاه خطی بیکار شود، مسیریاب ها صف ها را به ترتیب پویش کرده و از سر هر صف یکی را بر می دارد. بدین ترتیب، در شرایطی که n ماشین میزبان برای یک خط خروجی رقابت می کنند، از هر n بسته ارسالی بر روی خط یک بسته به هر ماشین میزبان تعلق می گیرد. افزایش نرخ ارسال بسته ها، در نسبت سهم هر ماشین تغییری ایجاد نخواهد کرد. در اینجا فرض شده است که سایز همه ی پکت ها یکسان می باشد. اگر سایز پکت ها یکسان نباشد، بر اساس بایت تصمیم گیری می شود و بصورت میانگین بایستی عادلانه رفتار شود. چون اگر بخواهد دقیق انجام شود ممکن است یک پکت نتواند ارسال گردد. یک اشکال این الگوریتم آن است که به تمام ماشین های میزبان، اولویت یکسانی می دهد. که در روش بعدی این مشکل حل شده است.

Weighted Fair Queuing:

به عنوان مثال در بسیاری از محیط ها، مطلوب تر آن است که به سرویس دهنده های ویدئو (video server) اولویت بیشتری نسبت به یک سرویس دهنده معمولی فایل داده شود و در هر تیک ساعت، سهم آن دو یا چند بسته باشد. این الگوریتم اصلاح شده به نام الگوریتم صف بندی بی طرفانهٔ وزن دار (Weighted Fair Queuing) مشهور است.

// برای مطالعه بیشتر //

خدمات مجتمع (Integrated Services):

برای انتقال داده های مالتی مدیا پروژه ای با نام کلی الگوریتم های مبتنی بر جریان (Flow – based algorithms) یا خدمات مجتمع (Integrated Services) شناخته می شود و کاربردهای چند پخشی (Multicast) و تک پخشی (Unicast) را در بر می گیرد. به عنوان مثالی از کاربردهای چند پخشی، ایستگاه های پخش تلویزیون دیجیتال را در نظر بگیرید که برنامه های خود را در قالب جریانی از بسته های IP به گیرندگان بی شمار و پراکندهٔ خود ارسال می دارند.

اصلی ترین پروتکل پیشنهاد شده توسط IETF برای ارائه خدمات مجتمع، RSVP نامیده می شود و برای رزرو کردن پهنای باند به کار می رود و از بروز ازدحام جلوگیری می کند.

خدمات متمایز (Differentiated Services)

الگوریتم های مبتنی بر جریان قابلیت عرضه کیفیت خوب خدمات به یک یا چند جریان را دارند زیرا در طول مسیر هر منبعی را که نیاز است از قبل رزرو می کنند. ولی این روش ها یک اشکال دارند: در این الگوریتم ها نیاز است که برای هر جریان پیشاپیش تنظیمات لازم انجام شود در حالی که در مقیاس کلان یعنی وقتی که هزاران یا میلیون ها جریان وجود دارد قابلیت اجرایی خود را از دست می دهند. از طرفی در هر مسیریاب وضعیت هر جریان به طور جداگانه نگهداری می شود و عملکرد این الگوریتم ها در مقابل خرابی یک مسیریاب آسیب پذیر خواهد بود. نهایتاً آن که برای تنظیم و ایجاد جریان باید تبادل اطلاعات پیچیده ای بین مسیریاب ها انجام گیرد. در نتیجه RSVP یا الگوریتم های مشابه آن بسیار کم پیاده سازی عملی شده اند. به همین دلایل IETF راهکارهای ساده تر برای تامین کیفیت خدمات ابداع کرد؛ روشی که بدون نیاز به هیچ تنظیمات قبلی یا تعیین کل مسیر می تواند به صورت محلی و مجزا در هر مسیریاب پیاده سازی شود. این راهکار اصطلاحاً روش مبتنی بر کلاس (Class – Based) برای تضمین کیفیت خدمات نامیده می شود. IETF یک معماری مناسب به نام خدمات متمایز برای آن طراحی و استانداردسازی کرده است.

خدمات متمایز (به اختصار DS گفته می شود) می تواند توسط مجموعه ای از مسیریاب ها که در یک حوزهٔ مدیریتی واحد قرار می گیرند (مثلاً مخابرات یا یک ISP) عرضه شود. مدیریت مسئول شبکه، مجموعه ای از کلاس های متفاوت خدمات و متناظر با آن، قواعد هدایت بسته ها (Forwarding Rules) را تعریف می کند. پیاده سازی خدمات DS بسیار آسان است.

سوئیچ برچسب MPLS[6] :

در ابتدای هر بسته یک برچسب (Lable) اضافه شود و بجای آن که مسیریابی و هدایت بسته ها 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 می شود. به کمک این روش، مسیریابی بسته ها به سرعت انجام شده و منابع موردنیاز در طول مسیر رزرو خواهد شد.

البته برچسب گذاری بر روی هر جریان شباهت عجیبی به مدارهای مجازی پیدا می کند. در هر زیرشبکه ی مدار مجازی یک برچسب یا به عبارتی یک شناسهٔ مدار مجازی در هر بسته قرار داده می شود و با استفاده از آن به عنوان یک اندیس برای درایه های جدول، مسیر مناسب به دست می آید.

ایدهٔ جدید سوئیچینگ با نام های متنوعی مثل سوئیچینگ برچسب[7] یا سوئیچینگ علامت[8] شناخته می شود. در نهایت IETF آن را تحت نام MPLS استاندارد کرد.

مضاف بر این، برخی افراد بین مسیریابی و سوئیچینگ فرق می گذارند. 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 می شود و با استفاده از این اندیس بلافاصله خط خروجی پیدا می شود، بدون آن که نیازی به جستجو باشد.

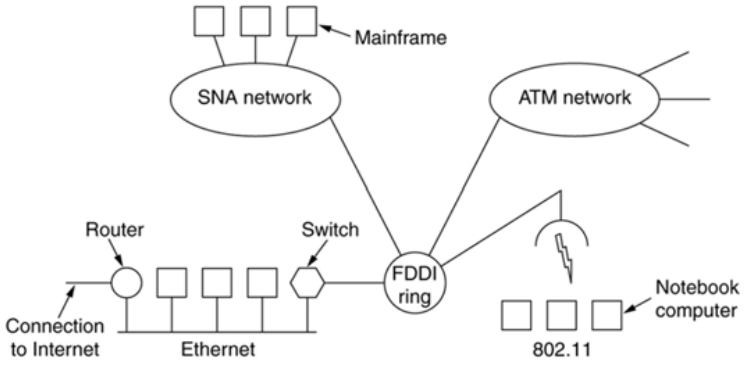
اولین مسئله آن است که این برچسب در کجا قرار داده شود. از آنجایی که بسته های IP برای شبکه های مدار مجاز ی طراحی نشده بودند، طبعاً هیچ فیلدی در سرآیند بستهٔ IP برای درج شماره های مدار مجازی وجود ندارد به همین دلیل سرآیند جدید MPLS باید در جلوی سرآیند هر بسته IP قرار بگیرد.

// انتهای برای مطالعه بیشتر //

Internetworking:

در دنیای واقعی، اینترنت چگونه کار می کند؟

برای جابجایی داده ها، بایستی ترکیبی از تکنولوژی ها و شبکه های مختلف بکار گرفته شود. به عنوان مثال برای یک کار ساده مثل چک کردن ایمیل، مجموعه وسیعی از شبکه ها درگیر می شوند. برای همین در دنیای واقعی مفهوم Internetworking یا ایجاد شبکه بین شبکه ای مطرح می شود که اساساً Internet را درست کرده است که ترکیب شبکه های مختلف می باشد.



شبکه های مختلف برای برقراری ارتباط، انواع و اقسام روش های ارتباطی دارند. همانند موارد زیر:

[6] Multi-Protocol Label Switching
[7] Label Switching
[8] Tag Switching

- ممکن است فقط در لایه ۱ یا لایه فیزیکی به هم متصل باشند. مثل یک شبکه اترنت که به کمک تکرارگر فقط طول کابل آن افزایش پیدا کرده است. (Repeaters)
- ممکن است اتصال در لایه ۲ یا پیوند داده مطرح گردد. مثل ارتباط بین شبکه ی دانشکده های مختلف دانشگاه. یعنی ترافیک ها وارد سوئیچ ها می شوند و بر اساس آدرس به مقصد ارسال می گردند. (Bridges, L2 Switches)
- بیشتر ارتباطات در شبکه در لایه ۳ مطرح هستند. به عبارتی نودها الزاماً در لایه ۱ و ۲ به هم وصل نیستند و آدرس های لایه ۳ پکت ها تعیین کننده اتفاقی است که برای آن ها می افتد تا به مقصد برسند.(Multi-protocol routers)
- بعضی وقت ها دروازه های ارتباطی لایه ۴ و ۵ هم می توانند در ارتباط شرکت کنند. مثلاً اپراتور بزرگ شبکه موبایل، Transport gateway داشته باشد تا ارتباط با دنیای شبکه ایجاد کند. همچنین Application gateway هایی مثل firewall وجود دارند که در نهادها و شرکت ها واسط ارتباطی بین دنیای بیرون با شبکه داخلی می باشند.

ایجاد ارتباط مسئله آسانی نیست. خیلی وقت ها شبکه ها از جنبه های مختلفی، متفاوت هستند. لذا ایجاد شبکه بین شبکه ای کار خیلی پیچیده ای خواهد بود.

در جدول زیر لیستی از آیتم ها را مشاهده می کنید که ممکن است شبکه ها بر اساس این آیتم ها متفاوت باشند.

Item	Some Possibilities
Service offered	Connection oriented versus connectionless
Protocols	IP, IPX, SNA, ATM, MPLS, AppleTalk, etc.
Addressing	Flat (802) versus hierarchical (IP)
Multicasting	Present or absent (also broadcasting)
Packet size	Every network has its own maximum
Quality of service	Present or absent; many different kinds
Error handling	Reliable, ordered, and unordered delivery
Flow control	Sliding window, rate control, other, or none
Congestion control	Leaky bucket, token bucket, RED, choke packets, etc.
Security	Privacy rules, encryption, etc.
Parameters	Different timeouts, flow specifications, etc.
Accounting	By connect time, by packet, by byte, or not at all

مثال۱: آدرس دهی لایه ۳ در دو نوع شبکه متفاوت یکسان نیست. ممکن است یک طرف آدرس IP داشته باشد و طرف دیگر اصلا آدرس IP نداشته باشد. به عنوان مثال، قبلاً پروتکلی تحت عنوان IPx خیلی رایج بود. اگر یک طرف IP و طرف دیگر IPx باشد، یک نهادی باید کار تبدیل آدرس ها را انجام دهد.

مثال۲: ممکن است در یک شبکه QOS ارائه گردد و در شبکه دیگر این امکان فراهم نباشد. وقتی بخواهیم یک ارتباط با کیفیت خدمات بین دو نقطه موجود در این دو نوع شبکه داشته باشیم، کار خیلی مشکل و شاید نشدنی باشد.

مثال۳: سایز پکت ها ممکن است متفاوت باشد. مثلاً در ATM سایز همیشه 53 بایتی است و cell نامیده می شود اما در IP می تواند 1500 بایت یا کوچکتر و یا بزرگتر هم باشد. پس لازم است نهادی این بسته ها را بتواند به هم تبدیل کند.

سرویس های غیر متصل نیز ممکن است ترتیب پکت ها را به هم بزند.

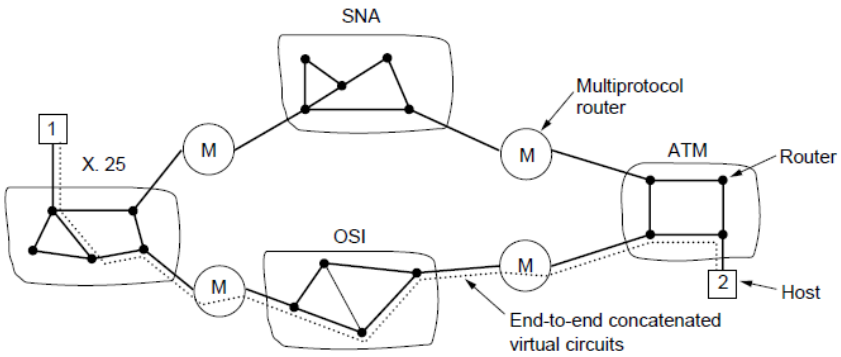
از مفاهیم بنیادین در تفاوت شبکه ها، نوع connection ها است.

۱. Concatenated Virtual Circuits

در شکل زیر مثالی از چند شبکه به تصویر کشیده شده است که همگی Connection Oriented هستند. وقتی نود ۱ می خواهد به نود ۲ اتصالی برقرار کند، در شبکه X.25 یک connection به شبکه OSI توسط یک Gateway زده می شود. همچنین بین OSI و ATM یک Gateway دیگر وجود دارد. همه ی مسیرها به هم وصل می شوند تا یک ارتباط Connection Oriented بین ۱ و ۲ برقرار گردد. اما اگر در بین یکی از شبکه ها Connection Oriented نباشد، به عنوان مثال وسط ارتباط یک شبکه IP قرار داشته باشد، نمی توانیم end to end connection داشته باشیم.

مزایای end to end connection:

۱- رزرو شدن فضای بافر ۲- گارانتی کردن ترتیب رسیدن بسته ها به مقصد ۳- هدرهای کوچک

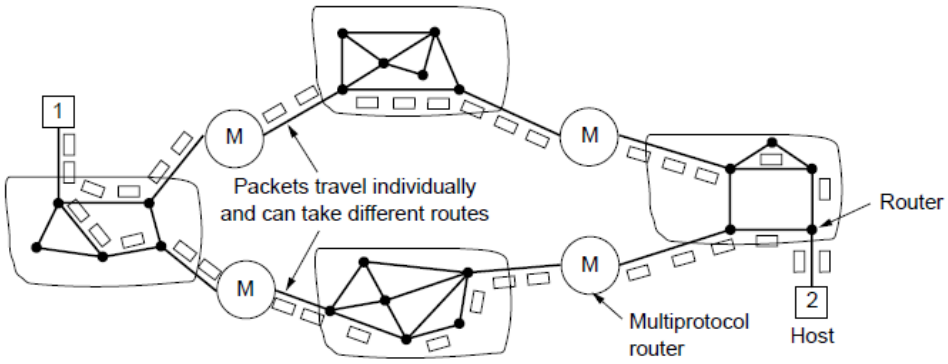


۲. Datagram service spanning an internet (multiple networks)

ممکن است ترکیبی از شبکه های Datagram را داشته باشیم که هر کدام بصورت on demand routing عمل کنند و Packet based باشند. در این حالت برای ارتباط بین ۱ و ۲ هر شبکه ممکن است ترافیک خود را از مسیرهای متفاوتی به مقصد برساند که در واقع مدل ارتباطی است که در اینترنت داریم. در این شبکه هیچ اتصالی از قبل برقرار نمی شود و به محض نیاز بسته به وضعیت شبکه برای مسیریابی بسته ها تصمیم گرفته می شود.

مزایای connection less:

۱- روتینگ پویا ۲- قابل استفاده در زیرشبکه هایی که مدار مجازی را پشتیبانی نمی کنند (مثل زیرشبکه های متحرک موبایل)



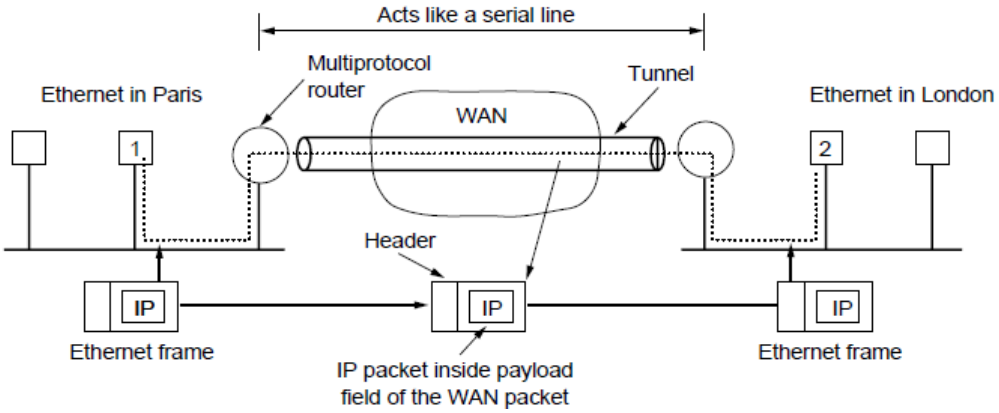
Tunnelling :

هرگاه مبدأ و مقصد در شبکه های هم نوعی قرار گرفته باشند اما در بین مسیر، شبکه ای متفاوت وجود داشته باشد، از مفهومی تحت عنوان تونل سازی (Tunneling) استفاده می کنیم. در این مکانیزم روترهایی وجود دارند که با پروتکل های مختلف کار می کنند و روند کار بدین صورت است:

گاهی اوقات می خواهیم ارتباطی ایجاد کنیم که از دید یک شبکه جزئیاتش پنهان باشد. به عنوان مثال می خواهیم ازAdsl یا Wimax به دانشگاه وصل شویم و داده هایی را تبادل کنیم که جزئیات آن از نظر ناظر شبکه قابل مشاهده نباشد. اگر به یک طریقی تونلی زده شود که داده های خارج شده از شبکه داخلی خانه، وارد آن گردند و به هنگام خروج از تونل وارد شبکه دانشگاه شوند، و این وسط هیچ جای دیگری نروند که به اصطلاح به آن VPN گوییم.

Virtual Private Network یا VPN:

برای داشتن یک شبکه اختصاصی عملاً امکان پذیر نیست که یک لینک اختصاصی واقعی پیاده سازی شود. اما می توان یک ارتباط مجازی مستقیم بین آن ها ایجاد کرد. در این روش همه ی تراکنش ها را در یک شبکه public به کمک یک تونل که روش های مختلفی برای پیاده سازی آن وجود دارد، از یک مسیر private عبور می کنند. VPN در لایه های ۲و۳و۷ قابل پیاده سازی هستند که الان وارد جزئیات آن نمی شویم.

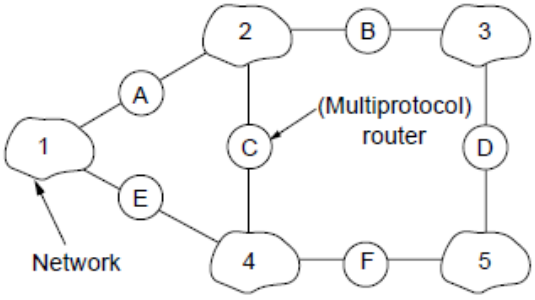


در VPN، آنچه در شبکه مبدأ تولید می شود، در ابتدای تونل بسته بندی می شود (مثل نامه ای که در پاکت گذاشته می شود و در پاکت را می بندیم). سپس برای فیلدهای آدرس بسته جدید، آدرس مسیریاب ابتدای تونل و مسیریاب انتهای تونل را درج می کنیم. بنابراین در بین، هیچ کس متوجه نمی شود نامه از کجا و برای چه کسی ارسال شده است. بلکه فقط می داند در گام بعدی، بسته را تحویل چه نقطه ای باید بدهد.

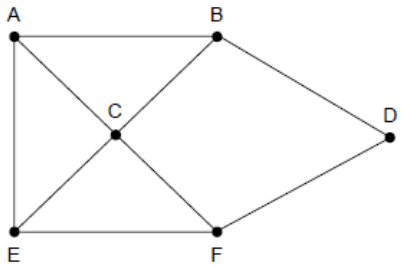
در شبکه مقصد وقتی بسته رسید، آن را باز می کنند و نامه اصلی را استخراج می نمایند. و بررسی می شود نامه از کجا و برای چه مقصدی ارسال شده است. به عبارتی در شبکه مقصد، به دست مقصد اصلی می رسد. VPN باعث می شود از جزئیات شبکه میانی خلاص شویم و امکان خصوصی کردن یک ارتباط را می دهد. بسیاری از نهادها مثل بانک ها، دانشگاه ها و ... از این روش استفاده می کنند.

Internetwork Routing:

مسیریابی بین شبکه های بهم متصل متفاوت که در آن قواعد و دستورات از یک جنس نیستند را با مثالی تشریح می کنیم. در شکل زیر شبکه ای متشکل از شبکه های دیگر می بینید. شبکه ای متشکل از شبکه های 1, 2, 3, 4, 5 وجود دارد که هرکدام از شبکه ها، متشکل از صدها نود می باشد. این شبکه ها، توسط یک Gateway Router به هم متصل هستند.



در این توپولوژی می توانید چنین فرض کنید که یک سری روتر داریم به نام های A, B, C, D, E, F که با هم توپولوژی زیر را می سازند. در توپولوژی زیر فقط روترها و کانال های ارتباطی مشخص شده است که اصطلاحاً زیر شبکه نامیده می شود.



به عنوان مثال داده هایی که دست A می آید، باید A بررسی کنید که دست B یا C یا E بدهد بهتر است؟ توپولوژی فوق، توپولوژی اتصال Gateway Router ها می باشد. حال سؤال این است که Gatewayها ترافیکی که می خواهند به دست Gateway های دیگر برسد را از کدام مسیر ارسال کنند؟

ممکن است در مواقعی برای تصمیم گیری در الگوریتم های روتینگ، ملاحظات مهندسی به تنهایی برای تصمیم گیری کافی نباشد. ملاحظاتی از قبیل ملاحظات سیاسی، اقتصادی و امنیتی نیز می تواند در روتینگ ها مؤثر باشد.

مثلاً ممکن است هرگاه بخواهیم از A به F داده ای ارسال کنیم، یکی از سیاست ها این باشد که روتر A هیچ وقت نباید از طریق C این کار را انجام دهد و این جزء policy هایی است که در A تنظیم می شود و کاری به ملاحظات مهندسی ندارد.

مثال:

کانادا کشوری با عرض زیاد و همسایه شمالی آمریکا است. ۹۰ درصد جمعیت کانادا در حاشیه جنوبی آن زندگی می کنند. آمریکا نیز یک کشور بزرگ و ثروتمند است بطوریکه جمعیت در بیشتر مساحت آن توزیع شده است. وقتی در کانادا کامپیوترها بخواهند با هم در ارتباط باشند یک راه حل خیلی اقتصادی این است که ترافیک خود را از زیرساخت های موجود حاشیه شمالی آمریکا عبور دهند. ولی در کانادا قانونی وجود دارد که هر پکتی که مبدأ و مقصدش کانادا باشد حق ندارد از خاک کانادا خارج گردد. مثلاً اینترنت ملی بحث شبکه ای داخل یک کشور برای فراهم کردن مقاصد سیاسی، اقتصادی و ... بدون توجه به ملاحظات مهندسی می باشد.

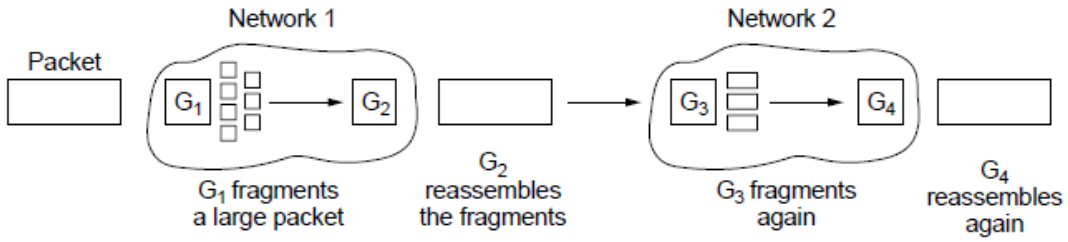
Fragmentation:

در اتصال شبکه های مختلف به هم ممکن است سایز بسته ها با هم مساوی نباشند. اگر حداکثر سایز بسته ای که دو شبکه می توانند داشته باشند، متفاوت باشد، موضوعی تحت عنوان Fragmentation مطرح خواهد شد.

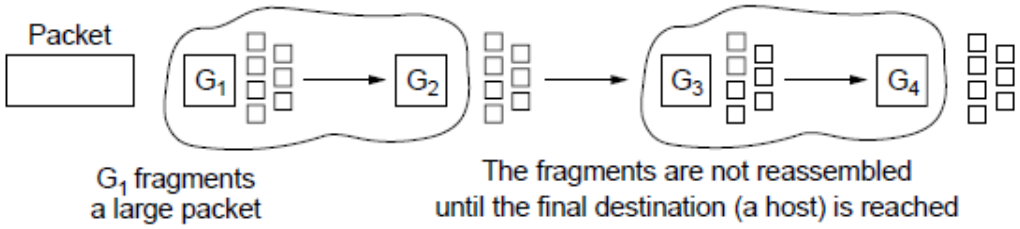
فرض کنید شبکه ای دارید که حداکثر سایز بسته در آن خیلی بزرگ تر از شبکه دوم باشد. اگر بسته ای بخواهد از شبکه اول به شبکه دوم وارد گردد، بایستی قبل از ورود به شبکه دوم به بسته های کوچکتر مطابق با سایز قابل قبول آن شبکه شکسته شود. و نهایتاً به هنگام خروج از آن شبکه مجدداً بسته های کوچک سرهم شده تا بسته اصلی را که بزرگتر بود، ساخته شود. اگر یک بسته در طول مسیرش از مبدأ به مقصد از شبکه هایی عبور کرد که سایز بسته قابل حمل در آن کوچکتر از سایز اصلی باشد این فرایند بایستی مکرراً صورت پذیرد. به فرایند شکسته شدن Fragmentation و سرهم شدن قطعات reassembly گفته می شود.

مثال: عبور یک پکت IP از یک شبکه ATM. در ATM حداکثر سایز بسته ها ۵۳ بایت است که cell نیز نامیده می شوند و ۴۸ بایت آن مخصوص payload می باشد. اما بسته های IP ممکن است هزار بایت نیز باشند. پس بایستی یک پکتIP شکسته شود و پس از تبدیل به cell در انتهای شبکه ATM مجدداً سرهم گردد و تبدیل به پکت IP گردد.

فرایند Fragmentation و Reassembly فرایندی است که دوست نداریم خیلی اتفاق بیفتد. چون جزء گروه کارهای Data path شبکه است. به عبارتی به ازای هر بسته این کار باید صورت پذیرد. هرگاه یک کار اضافی بدون توجیح در مسیر Data path درست می شود در واقع تحمیل بار اضافی بر سیستم می باشد. در این جا سربار، شکستن و سرجمع کردن مجدد بسته ها می باشد. و به دلیل اینکه این اتفاق ممکن است برای همه ی بسته های وارد شده و خارج شده از این شبکه پیش بیاید، امری نامطلوب می باشد اما در عین حال بعضی مواقع چاره ای جز این نداریم. این روش قطعه قطعه کردن شفاف می نامند. در شکل زیر این امر به تصویر کشیده شده است.



شیوه دیگری از Fragmentation وجود دارد که به نسبت قبلی سربار کمتری دارد. فرض کنید می خواهیم یک بسته ۱۰۰۰۰ بایتی را به ۱۰۰ بسته ۱۰۰ بایتی کوچکتر بشکنیم. ولی در انتهای شبکه اول، این تکه ها را سرهم نکنیم و همین تکه های کوچک را تحویل شبکه بعدی بدهیم. تا جایی که این تکه ها می توانند عبور کنند، همین روند را پیش می گیریم تا بالاخره به مقصد برسد. در مقصد نهایتاً تکه ها را سرهم می کنیم. پس در مجموع یک بار Fragmentation و یک بار Reassembly انجام می شود.



این استراتژی جالب تر به نظر می آید. اما اشکال خیلی مهمی دارد. کوچک بودن بسته ها به حدی که کوچک تر از آن چیزی باشد که شبکه می تواند مدیریت کند، یک بار اضافه بر روتینگ است.

اگر در شبکه اول تکه ای گم شود (۱۰۰۰۰ بایت به ۱۰۰ بسته ۱۰۰ بایتی شکسته شده بود) بنابراین ۹۹ بسته به شبکه دوم وارد خواهد شد که در مقصد نهایی متوجه خواهیم شد و نه زودتر. پس تمامی زحماتی که در شبکه دوم برای ۹۹ بسته کشیده بودیم هدر می رود. در مدل اول به محض خروج از شبکه اول به دلیل Reassemble کردن، متوجه نبود تکه از بسته ها می شویم و انتقال به شبکه ۲ انجام نمی گردد.

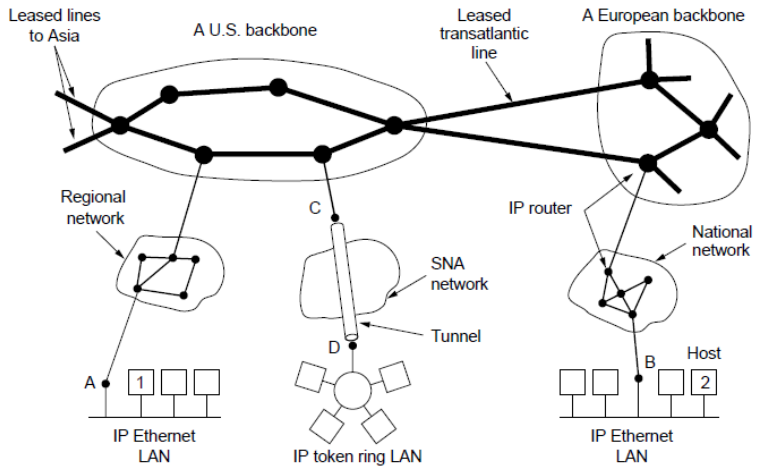
بنابراین هر دو مدل ممکن است سربار اضافی بر سیستم تحمیل کند. متناسب با شبکه های موجود بایستی از راه حل مناسب تر استفاده کرد.

:The Internet Network Layer (IP)

IP تلاش انجام شده ای است تا شبکه های مختلف را به هم بچسباند و یک اتصال خوب بین نودها برقرار سازد. در ابتدا که IP معرفی شد، مثل الان غالب نبود. اما در کنار رقبای دیگر، موفقیت چشمگیری داشته است.

جوهرهٔ اینترنت به گونه ای شکل گرفته است که مجموعه ای از شبکه های خودمختار[9] را به همدیگر وصل می نماید. هیچگونه ساختار حقیقی و ثابتی نمی توان برای اینترنت متصوّر شد. این نکته را بایستی یادآور شویم که در قسمت زیرشبکه از شبکهٔ اینترنت تعدادی از خطوط ارتباطی با پهنای باند بسیار بالا و مسیریاب های بسیار سریع و هوشمند، برای پیکرهٔ شبکهٔ جهانیِ اینترنت یک ستون فقرات[10] تشکیل داده است. شبکه های منطقه ای و محلّی پیرامون این ستون فقرات شکل گرفته و ترافیک دادهٔ آن ها به نحوی از این ستون فقرات خواهد گذشت.

[9] Autonomous
[10] Backbone

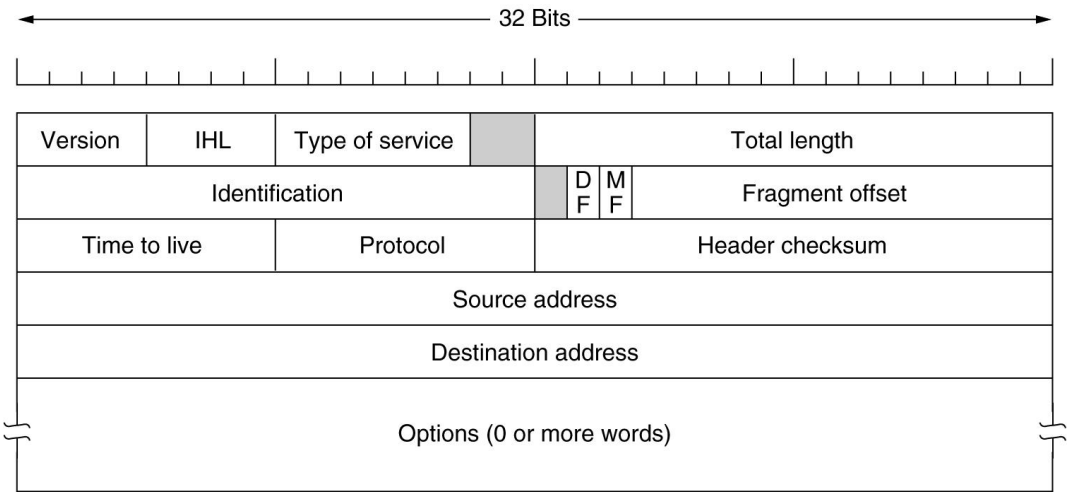


برای فهم عملکرد IP، مفاهیم Control path و Data path را یادآوری می کنیم.

از لحاظ Data path، شکل و شمایل بسته هایی که در این پروتکل مطرح می شوند مهم می باشد و از لحاظ Control path، قواعدی که بر شبکه حاکم هستند مورد بحث قرار می گیرند. هر دوی این جنبه ها مهم می باشد اما برای فهمیدن بیشتر یک پروتکل Control path آن پیچیده تر می باشد. دو نکته ابتدایی در مورد IP این است که اولاً پروتکل لایه ۳ می باشد، ثانیاً وظیفه آن روتینگ و کنترل ازدحام می باشد.

بخش Data path:

IP بسته هایی که از لایه های بالاتر خود (Segment) دریافت می کند را به تکه هایی تحت عنوان Packet می شکند. هر پکت یک سرایند یا هدری دارد که مشخص کننده ی نوع رفتار با این بسته در طول شبکه خواهد بود. به عبارتی روترها با بررسی هدر هر بسته در خصوص آن بسته تصمیم می گیرند. هدر IP در شکل زیر نشان داده شده است.



فیلد Version :

مشخص کننده نسخه پروتکل IP است که چهار بیتی می باشد. (نسخه شماره 4 پروتکلIP یا نسخه شماره 6 پروتکلIP) در حال حاضر تمامی شبکه ها و مسیریاب ها از نسخهٔ شماره ۴ پروتکل IP پشتیبانی می کنند. شمارهٔ ۶ پروتکل IP در حال بررسی و نصب است.

فیلد(IP Header Length) IHL :

این فیلد هم چهاربیتی است و طول کل سرآیند بسته را بر مبنای کلمات ۳۲ بیتی مشخص می نماید. .غیر از فیلد Options که اختیاری است، وجود تمامی فیلدهای سرآیند الزامی می باشد . طول قسمت اجباریِ سرآیند ۲۰ بایت است و بهمین دلیل حداقل عددی که در فیلد IHL قرار می گیرد ۵ یا (0101) خواهد بود و هر مقدار کمتر از ۵ به عنوان خطا تلقّی شده و منجر به حذف بسته خواهد شد. با توجه به طول ۴ بیتی این فیلد ، بدیهی است که حداکثر مقدار آن ۱۵ یا (1111) خواهد بود که در این صورت طول قسمت سرآیند ۶۰ بایت (۱۵×۴) و طول قسمت

اختیاری ۴۰ بایت می باشد .قسمت اختیاری در سرآیند برای اضافه کردن اطلاعاتی مثل آدرس مسیرهای پیموده شده،'' مهر زمان ''و برخی دیگر از گزینه هاست که در ادامه توضیح داده خواهد شد.

فیلد Type of service:

این فیلد هشت بیتی است و توسط آن ماشین میزبان (یعنی ماشین تولید کنندهٔ بستهٔ IP) از مجموعهٔ زیرشبکه (یعنی مجموعهٔ مسیریاب های بین راه) تقاضای سرویس ویژه ای برای ارسال یک دیتاگرام می نماید. از طریق این فیلد نوع سرویس درخواستی مشخص می شود، این فیلد خودش به چند بخش تقسیم شده است:

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

سه بیت سمت چپ: اولویت بستهٔ IP را تعیین می کند. اگر در این سه بیت صفر قرار گرفته باشد بسته اطلاعاتی از نوع معمولی تلقی می شود، یعنی دارای پایین ترین مقدار اولویت است و اگر مقدار ۷ یعنی (111) در این سه بیت قرار گرفته باشد بالاترین اولویت برای بسته درنظر گرفته می شود.

بیت های R , T , D: بیت D (Delay) به معنی تأخیر، بیت R(Reliability) به معنای قابلیت اطمینان و بیت T (Troughput) به معنی توان خروجی خط است.

فیلد Total Length :

در این فیلدِ ۱۶ بیتی عددی قرار می گیرد که طول کل بستهٔ IP را که شامل مجموع اندازهٔ سرآیند و ناحیه داده است، تعیین می کند. مبنای طول برحسب بایت است و بنابراین حداکثر طول کل بستهٔ IP می تواند ۶۵۵۳۵ بایت باشد.

فیلد Identification:

همانگونه که قبلاً اشاره شد برخی از مواقع مسیریاب ها یا ماشین های میزبان مجبورند یک دیتاگرام را به قطعات کوچکتر بشکنند و ماشین مقصد مجبور است آن ها را بازسازی کند، بنابراین وقتی یک دیتاگرامِ واحد شکسته می شود باید مشخصه ای داشته باشد تا در هنگام بازسازیِ آن در مقصد بتوان قطعه های آن دیتاگرام را از بقیه جدا کند. در این فیلدِ ۱۶ بیتی عددی قرار می گیرد که شمارهٔ یک دیتاگرام واحد را مشخص می کند.

فیلد Fragment offset:

این فیلد در سه بخش سازماندهی شده است:

I. بیت DF (Don’t Fragment): با یک شدن این بیت در یک بستهٔ IP هیچ مسیریابی حق ندارد آن را قطعه قطعه کند، چرا که مقصد قادر به بازسازی دیتاگرام های تکه تکه شده نیست.
II. بیت MF (More Fragment): این بیت مشخص می کند که آیا بستهٔ IP آخرین قطعه از یک دیتاگرام محسوب می شود یا باز هم قطعه های بعدی وجود دارد. در آخرین قطعه از یک دیتاگرام بیت MF صفر خواهد بود و در بقیه الزاماً ۱ است.
III. Fragment Offset: این قسمت که سیزده بیتی است در حقیقت شماره ترتیب هر قطعه در یک دیتاگرام شکسته شده محسوب می شود. با توجه به سیزده بیتی بودن این فیلد، یک دیتاگرام حداکثر می تواند به ۸۱۹۲ تکه تقسیم شود.

فیلد Time To Live:

این فیلد هشت بیتی در نقش یک شمارنده، طول عمر بسته را مشخص می کند. طول عمر یک بسته بطور ضمنی به زمانی اشاره می کند که یک بسته IP می تواند بر روی شبکه سرگردان باشد. حداکثر طول عمر یک بسته، ۲۵۵ خواهد بود که به ازای عبور از هر مسیریاب از مقدار این فیلد یک واحد کم می شود. هر گاه یک بستهٔ IP به دلیل بافر شدن در حافظهٔ یک مسیریاب زمانی را معطّل بماند، به ازای هر ثانیه یک واحد از این

فیلد کم خواهد شد به محض آنکه مقدار این فیلد به صفر برسد بستهٔ IP در هر نقطه از مسیر باشد حذف شده و از ادامهٔ سیرِ آن به سمت مقصد جلوگیری خواهد شد.

فیلد Protocol:

دیتاگرامی که در فیلد داده از یک بستهٔ IP حمل می شود با ساختمان دادهٔ خاصّ از لایهٔ بالاتر تحویلِ پروتکل IP شده تا روی شبکه ارسال شود. بعنوان مثال ممکن است این داده ها را پروتکل TCP در لایه بالاتر ارسال کرده باشد و یا ممکن است این کار توسط پروتکل UDP انجام شده باشد. بنابراین مقدار این فیلد شمارهٔ پروتکلی است که در لایه بالاتر تقاضای ارسال یک دیتاگرام کرده است؛ بسته ها پس از دریافت در مقصد باید به پروتکل تعیین شده تحویل داده شود.

فیلد Header Checksum:

این فیلد که شانزده بیتی است به منظور کشف خطاهای احتمالی در سرآیند هر بستهٔ IP استفاده می شود. برای محاسبهٔ کد کشف خطا، کلّ سرآیند بصورت دو بایت، دوبایت با یکدیگر جمع می شود. نهایتاً حاصل جمع به روش مکمل یک منفی می شود و این عدد منفی در این فیلد از سرآیند قرار می گیرد. در هر مسیریاب قبل از پردازش و مسیریابی ابتدا صحتِ اطلاعات درون سرآیند بررسی می شود و بستهٔ IP فاقد اعتبار حذف خواهد شد. دقت کنید که فیلد checksum در هر مسیریاب باید از نو محاسبه و مقداردهی شود زیرا وقتی یک بستهٔ IP وارد یک مسیریاب می شود حداقّل فیلد TTL از آن بسته عوض خواهد شد.

فیلد Source Address:

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

فیلد Destination Address:

در این فیلد آدرس ۳۲ بیتیِ مربوط به ماشین مقصد که باید بسته IP تحویل آن بشود، قرار می گیرد.

فیلد payload:

در این فیلد داده های دریافتی از لایهٔ بالاتر قرار می گیرد.

....

آدرس IP:

در این فضای ۳۲ بیتی چگونه آدرس دهی انجام می شود؟

نحوه آدرس دهی بصورت سلسله مراتبی می باشد و برعکس آدرس MAC که مسطح یا Flat بود، در آن سلسله مراتبی برای آدرس ها در نظر گرفته شده است. به مثال زیر توجه کنید تا مفهوم سلسله مراتب را بیشتر درک کرده باشید:

در شماره های تلفن، یک پیش شماره داریم که شهر را مشخص می کند و بقیه شماره ها مشخص می کند که در آن شهر به کجا وصل شویم. پیش شماره شهر های مختلف، متفاوت می باشد اما ممکن است شماره ای در داخل یک شهر با شماره دیگری در داخل شهر بعدی یکسان باشد که قاعدتاً با پیش شماره متفاوت تمییز داده می شوند. شبکه سوئیچ تلفن ابتدا با بررسی پیش شماره متوجه می شود که یک تماس مخصوص چه شهری می باشد. بعد از آن سراغ شماره تلفن داخلی آن شهر می رود تا مشترک مورد نظر را پیدا کند.

در شبکه نیز تقریباً چنین عمل می کنیم. سلسله مراتب آدرس دهی را دو لایه فرض می کنیم. درلایه اول تعدادی شبکه مشخص می شوند و در لایه دوم تعدادی host یا کامپیوتر که بایستی به آن ها آدرس اختصاص دهیم، تعیین می گردند.

آیا همه شبکه ها هم اندازه هستند؟ پاسخ منفی می باشد. برای همین گروه های مختلفی از آدرس را تعریف می کنیم که در ادامه به تشریح آن خواهیم پرداخت.

IP address به عنوان شناسه در یک شبکه استفاده می گردد. به عنوان مثال Shariaty Ali اسم یک فرد در یک سازمان است. در صورتی که برای وی بسته ای به سازمان ارسال شود، از روی اسم او مشخص است که بسته متعلق به چه کسی است.

IPv4 یا همان IP یک شناسه ۳۲ بیتی است که اگر به صورت دسیمال نوشته شود، ۴ عدد است که با نقطه از هم جداشده و هرکدام بین 0 تا 255 می تواند مقدار بگیرد.
قسمتی از این آدرس به Network و بخش دیگر به Host اختصاص دارد. مثلاً:

Network	Host
192.168.0.	3

Host های داخل یک شبکه، قسمت Network یکسان و بخش Host متفاوت دارند. Host هائی که داخل یک شبکه قرار ندارند، قسمت Network متفاوتی دارند. به عنوان مثال:

Network	Host
Shariaty	Ali
Shariaty	Hossein
Shariaty	Arman
Shariaty	Neda
Shariaty	Reza

در مثال فوق بخش Network (Shariaty) به ما نشان می دهد که تمامی Host ها (اسامی کوچک)، عضو یک خانواده اند. ولی در مثال زیر

Network	Host
Shariaty	Ali
Shariat panah	Hossein

می بینید که این دو فرد عضو یک خانواده نیستند چون عبارت بخش Network متفاوتی دارند.
برای اینکه به استانداردی در خصوص Network و Host برسند، تصمیم گرفتند که کلاس (دسته بندی) های مشخصی را برای آدرس های IP در نظر بگیرند.

گروه اول (کلاس A):

Class A

Network	Host		
1 to 126	x	x	x

8 bits	24 bits		
Network	Host		
0xxxxxxx	x	x	x

بایت اول برای شبکه در نظر گرفته می شود. بیت اول آن 0 است. 7 بیت دیگر برای تمایز شبکه ها از هم بکارگرفته می شود.
سه بایت بعدی برای Host ها در نظر گرفته شده است.

گروه دوم (کلاس B):

Class B

Network		Host	
128 to 191	x	x	x

16 bits		16 bits	
Network		Host	
10xxxxxx	x	x	x

دو بایت اول برای شبکه در نظر گرفته می شود. بیت اول و دوم آن 10 است. 6 بیت دیگر برای تمایز شبکه ها از هم بکارگرفته می شود.
دو بایت بعدی برای Host ها در نظر گرفته شده است.

گروه سوم (کلاس C):

Class C

Network			Host
192 to 223	x	x	x

	24 bits		8 bits
Network			Host
110 x x x x x	x	x	x

سه بایت اول برای شبکه در نظر گرفته می شود. بیت اول و دوم و سوم آن 110 است. 5 بیت دیگر برای تمایز شبکه ها از هم بکارگرفته می شود.
بایت نهایی برای Host ها در نظر گرفته شده است.

گروه چهارم (کلاس D):

Class D

1110	multicast address

گروه پنجم (کلاس E):

Class E

11110	Reserved for future use

هدف از این کلاس بندی این بود که مشخص شود دو آدرس IP از یک خانواده هستند (در یک شبکه هستند) یا خیر.

مثال۱: آیا 80.83.26.70 و 80.81.25.32 در یک شبکه قرار دارند؟
جواب: ابتدا نگاه می کنیم که این آدرس ها عضو کدام کلاس هستند. چون هر دو با 80 شروع شده اند پس عضو کلاس A می باشند. در شکل زیر بخش Network را از Host جداگانه نشان داده ایم. هر دو به علت اینکه ادرس شبکه یکسان دارند پس در یک شبکه واقع شده اند.

Network	Host
80.	81.25.32

Network	Host
80.	83.26.70

مثال ۲: آیا 130.41.35.50 و 130.42.39.50 در یک شبکه قرار دارند؟
جواب: ابتدا نگاه می کنیم که این آدرس ها عضو کدام کلاس هستند. چون هر دو با 130 شروع شده اند پس عضو کلاس B می باشند. در شکل زیر بخش Network را از Host جداگانه نشان داده ایم. هر دو به علت اینکه ادرس شبکه یکسانی ندارند پس در یک شبکه واقع نیستند.

Network	Host
130.42.	39.50

Network	Host
130.41.	.35.50

مثال۳: آیا 190.25.30.48 و 190.25.35.42 در یک شبکه قرار دارند؟
جواب: ابتدا نگاه می کنیم که این آدرس ها عضو کدام کلاس هستند. چون هر دو با 190 شروع شده اند پس عضو کلاس B می باشند. در شکل زیر بخش Network را از Host جداگانه نشان داده ایم. هر دو به علت اینکه ادرس شبکه یکسان دارند پس در یک شبکه واقع شده اند.

Network	Host
190.25.	30.48

Network	Host
190.25.	35.42

تبدیل اعداد دسیمال به باینری و بلعکس

برای اینکه بتوانیم یک آدرس IP را تحلیل کنیم و یا در جلوتر Subnet کنیم، بایستی یاد بگیریم که با اعداد باینری کار کنیم. هر قسمت از چهار قسمت دسیمال آدرس IP را به یک عدد ۸ بیتی باینری تبدیل خواهیم نمود. و آن را در بیت های ۰ تا ۷ قرار خواهیم داد:

بیت ۰	بیت ۱	بیت ۲	بیت ۳	بیت ۴	بیت ۵	بیت ۶	بیت ۷

دو را به توان شماره هر بیت برسانید و مقدار عددی آن را یادداشت کنید:

۱	۲	۴	۸	۱۶	۳۲	۶۴	۱۲۸
دو به توان ۰	دو به توان ۱	دو به توان ۲	دو به توان ۳	دو به توان ۴	دو به توان ۵	دو به توان ۶	دو به توان ۷
بیت ۰	بیت ۱	بیت ۲	بیت ۳	بیت ۴	بیت ۵	بیت ۶	بیت ۷

حالا وقتی می خواهیم یک عدد دسیمال را به باینری تبدیل کنیم ، عدد را بصورت متوالی به مقادیر بالا ، از چپ به راست ، کسر می کنیم. در صورتیکه مقادیر توانی دو قابلیت کسر شدن از عدد باقیمانده را داشت؛ در جدول مربوطه عدد ۱ و اگر نداشت عدد صفر را قرار می دهیم. برای اینکه درک بهتری داشته باشیم عدد ۲۴۹ را به باینری تبدیل می کنیم :

مرحله اول)

$$249 - 128 = 121$$

بنابراین ۱۲۸ در ۲۴۹ وجود دارد بنابراین:

۱	۲	۴	۸	۱۶	۳۲	۶۴	۱۲۸
							۱

مرحله دوم)

$$121 - 64 = 57$$

بنابراین ۶۴ داخل ۱۲۱ وجود دارد پس:

۱	۲	۴	۸	۱۶	۳۲	۶۴	۱۲۸
						۱	۱

مرحله سوم)

$$57 - 32 = 25$$

بنابراین ۳۲ داخل ۵۷ وجود دارد پس:

۱	۲	۴	۸	۱۶	۳۲	۶۴	۱۲۸
					۱	۱	۱

مرحله چهارم)

$$25 - 16 = 9$$

بنابراین ۱۶ داخل ۲۵ وجود دارد پس:

۱	۲	۴	۸	۱۶	۳۲	۶۴	۱۲۸
				۱	۱	۱	۱

مرحله پنجم)

$$9 - 8 = 1$$

بنابراین ۸ داخل ۹ وجود دارد پس:

۱۲۸	۶۴	۳۲	۱۶	۸	۴	۲	۱
۱	۱	۱	۱	۱			

مرحله ششم)

1– 4 = ERROR

بنابراین ۴ داخل ۱ وجود ندارد پس:

۱۲۸	۶۴	۳۲	۱۶	۸	۴	۲	۱
۱	۱	۱	۱	۱	۰		

مرحله هفتم)

1– 2 = ERROR

بنابراین ۲ داخل ۱ وجود ندارد پس :

۱۲۸	۶۴	۳۲	۱۶	۸	۴	۲	۱
۱	۱	۱	۱	۱	۰	۰	

مرحله آخر)

1 – 1 = 0

بنابراین ۱ داخل ۱ وجود دارد پس :

۱۲۸	۶۴	۳۲	۱۶	۸	۴	۲	۱
۱	۱	۱	۱	۱	۰	۰	۱

به عبارتی عدد دسیمال ۲۴۹ به باینری می شود : ۱۱۱۱۱۰۰۱

تمرین) عدد دسیمال ۶۳ را به باینری تبدیل کنید .

:Network ID & Broadcast address

هر Range آدرس IP داخل یک شبکه واحد، مجموعه ای از یکسری آدرس IP است که همگی داخل آن شبکه اند. از این مجموعه، دو آدرس منحصر به فرد وجود دارد :

Network ID : که مشخصه و معرف آن شبکه است.

Broadcast address : که برای دسترسی به همه نود های آن شبکه استفاده می شود.

این دو آدرس را نمی توان به عنوان آدرس معتبر، به نود ها اختصاص داد. برای محاسبه NetID تمام بیت های Host را صفر می کنیم. و برای دسترسی به Broadcast address تمام بیت های Host را یک می کنیم.

مثال) NetID و Broadcast address شبکه ای که آدرس IP ، 80. 32.51.60 در آن وجود دارد را پیدا کنید.

ابتدا نگاه می کنیم که این آدرس ها عضو کدام کلاس می باشند، چون می خواهیم بخش Network را از Host جدا کنیم. این آدرس متعلق به Class A است. بنابراین:

Network	Host
80.	32.51.60

اگر آدرس را بصورت باینری بنویسیم خواهیم داشت:

Network	Host
01010000 .	00100000 . 00110011 . 00111100

حالا برای اینکه NetID را بدست آوریم ، تمام بیت های Host را صفر می کنیم.

Network	Host
01010000 .	00000000 . 00000000 . 00000000

پس NetID می شود : 80.0.0.0

برای بدست آوردن آدرس Broadcast همه بیت های Host را یک می کنیم.

Network	Host
01010000 .	11111111 . 11111111 . 11111111

پس Broadcast address می شود : 80.255.255.255

چون این دو آدرس را نمی توانیم به نود ها اختصاص دهیم، بنابراین اولین آدرس قابل استفاده می شود یکی بالاتر از NetID به عبارتی:

Network	Host
01010000 .	00000000 . 00000000 . 00000001

اولین آدرس این شبکه می شود : 80.0.0.1

آخرین آدرس شبکه نیز می شود یکی مانده به آدرس Broadcast یعنی:

Network	Host
01010000 .	11111111 . 11111111 . 11111110

آخرین آدرس قابل استفاده در این شبکه می شود : 80.255.255.254

بنابراین وقتی می خواهیم تعداد آدرس های قابل استفاده در یک شبکه را حساب کنیم از فرمول 2^h-2 استفاده می کنیم که h در آن، تعداد بیت های Host می باشد.

تمرین) شبکه ای که آدرس IP ، 201.202.32.40 در آن وجود دارد را تحلیل کنید.

ابتدا نگاه می کنیم که این آدرس ها عضو کدام کلاس می باشند ، چون می خواهیم بخش Network را از Host جدا کنیم.

این آدرس متعلق به Class C است. بنابراین:

Network	Host
201.202.32.	40

اگر آدرس را بصورت باینری بنویسیم خواهیم داشت :

Network	Host
11001001 . 11001010 . 00100000 .	00101000

حالا برای اینکه NetID را بدست آوریم ، تمام بیت های Host را صفر می کنیم.

Network	Host
11001001 . 11001010 . 00100000 .	00000000

پس NetID می شود : 201.202.32.0

برای بدست آوردن آدرس Broadcast همه بیت های Host را یک می کنیم.

Network	Host
11001001 . 11001010 . 00100000 .	11111111

پس Broadcast address می شود : 201.202.32.255

چون این دو آدرس را نمی توانیم به نود ها اختصاص دهیم ، بنابر این اولین آدرس قابل استفاده می شود یکی بالاتر از NetID به عبارتی:

Network	Host
11001001 . 11001010 . 00100000 .	00000001

اولین آدرس این شبکه می شود : 201.202.32.1

آخرین آدرس شبکه نیز می شود یکی مانده به آدرس Broadcast یعنی:

Network	Host
11001001 . 11001010 . 00100000 .	11111110

آخرین آدرس قابل استفاده در این شبکه می شود : 201.202.32.254

تعداد آدرس IP قابل استفاده در شبکه : $(2^8)-2$ می شود ۲۵۴ آدرسIP .

وقتی از شما می خواهند که شبکه ای را تحلیل کنید ، بایستی موارد زیر را حساب کنید:

Class	C
Network ID	201.202.32.0
First IP address	201.202.32.1
Last IP address	201.202.32.254
Broadcast address	201.202.32.255
Number of Available IP addresses	254

تمرین) شبکه ای که آدرس IP ، 130.64.33.25 در آن وجود دارد را تحلیل کنید.

سوال) شبکه ای ۲۰۰ عدد نود دارد. از کدام Class برای آدرس دهی استفاده کنیم ؟

$$(2^h) - 2 >= 200$$

h = 8 پس تعداد بیت های Host باید حداقل ۸ باشد.

Network	Host
xxxxxxxx.xxxxxxxx.xxxxxxxx	hhhhhhhh

پس از Class C استفاده خواهیم کرد.

Subnet Mask

آموختیم که چگونه می توانیم بفهیم دو آدرس IP متعلق به یک شبکه اند یا خیر. کامپیوتر برای اینکه این موضوع را بفهمد از مفهومی به نام Subnet Mask استفاده می کند. به این صورت که تمام بیت های Network را یک و تمام بیت های Host را صفر در نظر می گیرد تا Subnet mask را بسازد. سپس Subnet Mask را در آدرسIP ، Boolean AND می کند.

Boolean AND :
1 AND 0 = 0
0 AND 1 = 0
0 AND 0 = 0
1 And 1 = 1

به عبارتی:

Subnet mask برای Class A:

11111111.00000000.00000000.00000000

255.0.0.0

Subnet mask برای Class B:

11111111.11111111.00000000.00000000

255.255.0.0

Subnet mask برای Class C:

11111111.11111111.11111111.00000000

255.255.255.0

به عنوان مثال کامپیوتر می خواهد متوجه شود که آدرس های 80.23.45.2 و 80.24.35.1 در یک شبکه اند یا خیر؟

80.23.45.2

01010000.00010111.00101101.00000010

این را در subnet Mask مربوط به Class A ، AND می کند:

01010000.00010111.00101101.00000010
AND
11111111.00000000.00000000.00000000

نتیجه ۱ :

01010000.00000000.00000000.00000000

همین کار را برای آدرس بعدی می کند:

80.24.35.1

01010000.00011000.00100011.00000001

این را در subnet Mask مربوط به Class A ، AND می کند:

01010000.00011000.00100011.00000001
AND
11111111.00000000.00000000.00000000

نتیجه ۲ :

01010000.00000000.00000000.00000000

اگر نتیجه۱ با نتیجه۲ مقدار یکسانی داشت (که در اینجا یکسان است)، برداشت می کند که این دو آدرس در یک شبکه اند.
از روی خود آدرس می توان تشخیص داد که کدام قسمت مربوط به Network و کدام قسمت مربوط به Host است. پس لزوم استفاده از Subnet Maskچیست؟
می توان بدون توجه به Class ، آدرس دهی نمود (Subnetting) . در چنین مواردی برای تشخیص این مطلب که کدام قسمت مربوط به Network و کدام قسمت مربوط به Host است، استفاده از Subnet Mask ضروری است. با توجه به تعاریف گفته شده می توان نتیجه گرفت که عدد Subnet Mask هیچ ارتباطی با آدرس IP ندارد و فقط نمایانگر این است که کدام قسمت مربوط به Network و کدام قسمت مربوط به Host است.

Public & Private IP addresses

ابتدا آدرس های IP که اختصاص داده می شد، بنا بود از طریق اینترنت، مسیریابی شوند. ولی همه آدرس ها لازم نبود از اینترنت دیده شوند و به عبارتی یکسری آدرس داخلی نیاز بود تا در خود سازمان استفاده گردند. اینجا بحث آدرس های Public و Private مطرح گردید. یعنی یک Range از IP ها در اختیار قرار گرفت تا بصورت داخلی و خصوصی استفاده شوند و از روی اینترنت مسیریابی نشوند.

Class A : 10.0.0.0 – 10.255.255.255/8 (16,772,216 hosts)
Class B : 172.16.0.0 – 172.31.255.255/12 (1, 0478,576 hosts)
Class C: 192.168.0.0 – 192.168.255.255/16 (65,636 hosts)

به این Range از آدرس ها، آدرس های Private گفته می شوند (در ادامه کاربرد این آدرس ها را در NAT خواهید خواند). سایر آدرس ها، آدرس های Public هستند که عموماً هزینه دارند و از طریق ISP ها و یا سازمان های مرجع اینترنت بالاتر، مدیریت می شوند و در اختیار کاربران قرار می گیرند.

آدرس های خاص:

در بین تمامی کلاس های آدرسIP پنج گروه از آدرس ها، معنای ویژه ای دارند و با آن ها نمی توان یک شبکهٔ خاص را تعریف و آدرس دهی کرد. این پنج گروه آدرس عبارتند از:

آدرس 0.0.0.0 :

هر ماشین میزبان که از آدرس IP خودش مطلع نیست این آدرس را بعنوان آدرس خودش فرض می کند. البته از این آدرس فقط به عنوان آدرس مبداء و برای ارسال یک بسته می توان استفاده کرد و گیرندهٔ بسته نمی تواند پاسخی به مبداء بسته برگرداند.

آدرسHostID .0 :

این آدرس زمانی به کار می رود که ماشین میزبان، آدرس مشخصهٔ شبکه ای که بدان متعلق است را نداند. در این حالت در قسمت NetID مقدار صفر و در قسمت HostID شمارهٔ مشخصهٔ ماشین خود را قرار می دهد.

آدرس 255.255.255.255 :

برای ارسال پیام های فراگیر برای تمامی ماشین های میزبان بر روی شبکهٔ محلی که ماشین ارسال کننده به آن متعلق است.

آدرس NetID.255 :

برای ارسال پیام های فراگیر برای تمامی ماشین های یک شبکهٔ راه دور که ماشین میزبان فعلی متعلق به آن نیست. آدرس شبکهٔ مورد نظر در قسمت NetID تعیین شده و تمامی بیت های قسمت مشخصهٔ ماشین میزبان ۱ قرار داده می شود. البته بسیاری از مسیریاب ها برای مصون ماندن شبکه از مزاحمت های بیرونی، چنین بسته هایی را حذف می کنند.

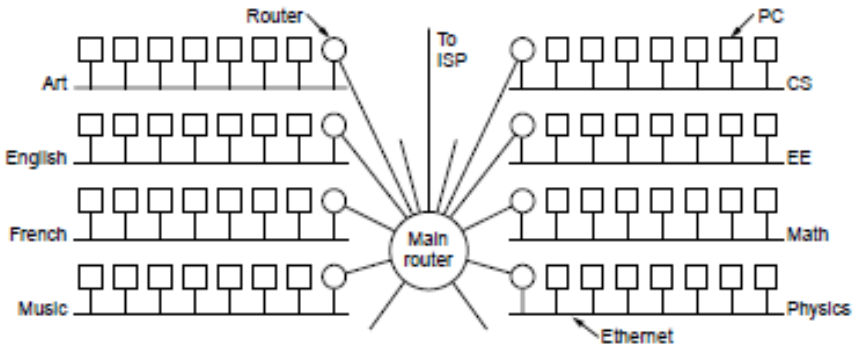
آدرس 127.xx.yy.zz :

این آدرس بعنوان "آدرس بازگشت" شناخته می شود و آدرس بسیار مفیدی برای اشکال زدایی از نرم افزار می باشد. به عنوان مثال اگر بسته ای به آدرس 127.0.0.1 ارسال شود، بسته برای ماشین تولیدکنندهٔ آن بر خواهد گشت. در این حالت اگر نرم افزارهای TCP/IP درست و بدون اشکال نصب شده باشد فرستندهٔ بسته باید آنرا مجدداً دریافت کند. همچنین از این آدرس می توان برای آزمایش برنامه های تحت شبکه، قبل از نصب آن ها بر روی ماشین های میزبان استفاده کرد.

زیرشبکه سازی (subnetting):

زیر شبکه های استاندارد[11]

در ادامهٔ بحث بایستی مسئله زیر شبکه را در خصوص آدرس دهی ها مطرح نمائیم. مبحث را با یک مثال آغاز می نمائیم:
فرض کنید دانشگاه شما یک کلاس C با قابلیت تعریف ۲۵۴ ماشین میزبان ثبت می نماید (مثلاً 211.11.121.0)؛ یعنی شبکهٔ دانشگاه توانایی آدرس دهی ۲۵۵ ایستگاه را در شبکه دارد. در نظر بگیرید که دانشگاه دارای یک شبکهٔ محلی واحد و یکپارچه برای کل دانشگاه نیست بلکه دارای هشت شبکهٔ محلی مجزا است که برای هر دانشکده تهیه دیده شده است. هر کدام از این شبکه ها که می تواند توپولوژی متفاوتی داشته باشد، از طریق مسیریاب به هم متصل شده اند و طبعاً برای ارتباط بین شبکه های هر دانشکده باید مسیریابی صورت گیرد. از دیدگاه بیرونی کلّ مجموعهٔ شبکه های محلی دانشگاه با یک آدرس مشخصه یعنی 211.11.121.0 شناخته می شود و مسیریاب های بیرونی هیچ شناختی از ساختار شبکه بندی داخلی دانشگاه ندارند. (هر یک از شبکه های محلی داخل دانشگاه یک زیرشبکه نامیده می شود.) بنابراین باید روشی وجود داشته باشد تا از طریق آدرس های کلاس C (یا هر کلاس دیگر) بتوان زیر شبکه ها را نیز مشخص کرد تا مسیریاب های داخلی نیز قادر باشند زیر شبکه های مختلف را شناسایی و تفکیک کنند.
این مسئله برای آدرس های کلاس B و A بسیار ضروری و اجتناب ناپذیر می نماید چرا که نمی توان انتظار داشت که یک موسسه که آدرس کلاس B با قابلیت تعریف حدود ۶۶ هزار ماشین میزبان ثبت کرده است فقط یک شبکهٔ یکپارچه داشته باشد بلکه چنین موسسه ای ممکن است دارای صدها زیر شبکهٔ کوچک و بزرگ باشد.

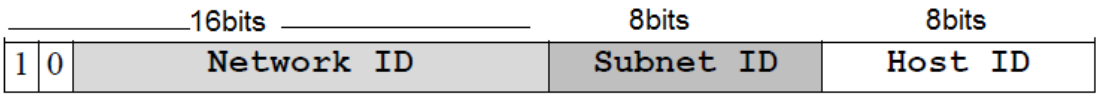


برای آنکه بتوان زیرشبکه ها را تفکیک کرد جدای از قسمت آدرس شبکه که کل شبکهٔ دانشگاه شما را مشخص می کند بایستی در قسمت مشخصهٔ ماشین میزبان نیز به گونه ای زیر شبکه ها مشخص شوند. این کار از طریق مفهومی به نام

[11] Standard Subnet Mask

الگـــوی زیرشبکه [۱۲] انجـــام می شـود. شما با نگاه اول به اولین عـــدد سـمت چـپ متوجـه خواهیـد شـد کـه ایـن آدرس از چـه کلاســی اســت ولــی هنـــوز مـــوارد مبـــهمی وجـــود دارد: آیـــا شـــبکه ای کـــه آدرس آنرا پیش رو دارید فقط یک شـبکه اسـت یـا خـودش زیـر شـبکه بنـدی شـده اسـت؛ یعنی از چند شبکهٔ محلی متصـل بـهم تشـکیل شـده اسـت؟

فرآیند استفاده از "الگوی زیرشبکه" را با استفاده از مثالی با آدرس کلاس B آموزش می دهیم:
فرض کنید شما کاربری روی یک ایستگاه در شبکه دانشگاه خودتان هستید، آدرس IP متعلق به دستگاه شما بصورت زیر اختصاص داده شده است : 131.55.213.73
با یک نگاه متوجه می شوید که آدرس از کلاس B است که مشخصهٔ شبکهٔ آن معادل 131.55.0.0 و مشخصهٔ ماشین شما 0.0.213.73 است؛ ولی هنوز نمی دانید شبکه ای که مشخصهٔ آن معادل 131.55.0.0 است آیا زیر شبکه دارد یا خیر؟
فرض کنید که دانشگاه شما با آدرس شبکه 131.55.0.0، می خواهد حداکثر دارای ۲۵۴ زیر شبکه باشد، بهمین دلیل فرض کرده است که در فیلد مشخصهٔ ماشین میزبان (Host ID) که در کلاس B دو بایت سمت راست را شامل می شود، بایت دومِ آن به عنوان مشخصهٔ مربوط به زیر شبکه تعریف شود (این مدل آدرس دهی آدرس دهی سه سطحی است). یعنی فیلد دوبایتی مربوط به مشخصهٔ ماشین میزبان به دو بخش تقسیم شده است:



ماشین شما تصمیم دارد بسته ای را برای یک ماشین میزبان با آدرس IP معادل 131.55.108.75 بفرستد؛ ماشین از کجا می تواند بفهمد که مقصد روی همین شبکه محلی که شما بدان متعلق هستید واقع است یا آنکه به شبکهٔ محلی در یک دانشکدهٔ دیگر متعلق است.
دانستن این موضوع بسیار با اهمیت خواهد بود چرا که اگر ماشین میزبان مورد نظر روی شبکهٔ دیگری باشد بسته باید با آدرس فیزیکی "مسیریاب پیش فرض" روی کانال ارسال شود. بنابراین تمام ماشین های روی شبکه بایستی از وضعیت زیر شبکه ها مطلع باشند.
با توجه به آنچه که در بالا اشاره شد دومین بایت از سمت راست بعنوان مشخصهٔ زیر شبکه اختصاص داده شده است و بهمین دلیل هر ماشین برای دانستن آنکه آیا ماشین مقصد در شبکه محلی خودش واقع است یا در خارج از شبکه قرار دارد باید قسمت "مشخصهٔ شبکه" و "مشخصهٔ زیرشبکه" از آدرس IP خودش را با همین مشخصه ها از آدرس مقصد مقایسه نماید.
اینجاست که یک الگوی ۳۲ بیتی تعریف می شود که یک عدد ۳۲ بیتی و در این مثال بصورت 255.255.255.0 است:

هر گاه ماشین بخواهد یک آدرس IP را تحلیل کند. الگوی فوق را با آدرس IP خودش AND می کند. (با اینکار در حقیقت Host ID خودش را صفر می نماید) سپس مجدداً الگو را با آدرس IP مقصد AND می کند (مشخصهٔ ماشین مقصد هم صفر می شود) حال نتیجه دو مرحله را با هم مقایسه می نماید. اگر نتیجه دو مرحله یکسان بود، هم مشخصهٔ شبکه و هم مشخصهٔ زیرشبکه از آدرس های مبدأ و مقصد یکی است و هر دو روی یک شبکهٔ محلی قرار دارند. در صورت عدم تساوی، ماشین مبدأ به این نتیجه می رسد که مقصد مورد نظر روی شبکه محلی خودش نیست و آن بسته بایستی به آدرس فیزیکی مسیریابِ پیش فرض ارسال شود.
فرض کنید بسته ای با آدرس های مشخص زیر بخواهد ارسال شود:
آدرس ماشین مبدأ: 131.55.213.73
آدرس ماشین مقصد: 131.55.108.75
الگوی زیر شبکه: 255.255.255.0

[12] Subnet mask

حاصل AND این دو آدرس با الگوی زیرشبکه مساوی نیستند و بنابراین ماشین مبدأ متوجه خواهد شد که ماشین مقصد روی شبکه محلی خودش نیست و بسته اطلاعاتی را بایستی به آدرس فیزیکی مسیریاب پیش فرض ارسال نماید.
به عنوان مثالی دیگر فرض کنید ماشین مبدأ می خواهد برای ماشینی با آدرس IP زیر بسته ای را ارسال نماید:
آدرس ماشین مقصد: 131.55. 213.84
حاصل AND آدرس ماشین مبدأ و مقصد فوق با الگوی زیرشبکه مساوی هستند و بالطبع مبدأ و مقصد روی یک شبکه محلی واقعند و هیچ لزومی ندارد که ارسال بسته به آدرس فیزیکی مسیریاب پیش فرض انجام شود، بلکه باید مستقیماً از آدرس فیزیکی ایستگاه مقصد استفاده شود.
الگوی زیرشبکه در مثال های بالا ساده ترین حالت بود که به آن ها "الگوی زیرشبکهٔ استاندارد " گفته می شود چرا که الگوها دقیقاً مبتنی بر تقسیمات هشت بیتی هستند.

زیر شبکه های غیر استاندارد

هدف از Subnetting این است که یک Range از آدرس های IP ، که به ما تعلق دارد را به چند Range آدرس مجزا خورد کنیم تا بتوانیم از هر Range جداگانه استفاده کنیم. مثلاً ممکن است بخواهیم برای کاهش ترافیک، شبکه را به چند سگمنت، تقسیم کنیم و بین سگمنت ها روتر قرار دهیم.
وقتی یک شبکه را Subnet می کنیم، Subnet Mask جدیدی خواهیم داشت که معرف بخش Network وHost خواهد بود.
یک آدرس Class A را در نظر بگیرید. مثلا 10.0.0.0 . این شبکه ۲ به توان ۲۴ منهای ۲ ، آدرس IP قابل استفاده به ما می دهد. یعنی ۱۶۷۷۷۲۱۴ . حالا فرض کنید این تعداد IP در یک شبکه در یک سازمان که ۴۰ شعبه دارد و در هر کدام ۱۰۰ کامپیوتر ؛ به چه دردی می خورد. در اینجا ما ترجیح می دهیم که به جای یک شبکه بزرگ، ۴۰ شبکه کوچک تر داشته باشیم تا بتوانیم در صورت نیاز بین آن ها مسیریابی کنیم.

مسیریابی بر اساس آدرس های بدون کلاس CIDR[۱۳]

کاری که می کنیم این است که دیگر از Class های پیش فرض استفاده نمی کنیم. بلکه آن Class ها را بسته به میل خودمان به بخش های کوچکتر تقسیم می کنیم. به این عمل Subnetting می گوئیم.
چه چیز این تقسیم بندی جدید را به ما می دهد ؟ Subnet Mask جدید
یک مثال از همان اسم و فامیل که در ابتدا زدیم:

Network	Host
Shariaty	Seyed Ali
Shariaty	Haj Hossein
Shariaty	Seyed Arman
Shariaty	Haj Ebrahim
Shariaty	Seyed Naser

زمانی که Subnet Mask بصورت بالا در نظر گرفته می شود، همه عضو یک خانواده حساب می شوند. حالا فرض کنید خط Subnet Mask (خط مجزا کننده Network از Host) را به سمت راست بکشیم:

Network	Host
Shariaty Seyed	Ali
Shariaty Haj	Hossein
Shariaty Seyed	Arman
Shariaty Haj	Ebrahim
Shariaty Seyed	Naser

[13] Classless Inter Domain Routing

می بینید که دیگر همه عضو یک خانواده نیستند و ما یک خانواده بزرگ را به دو خانواده کوچک تقسیم کردیم. چگونه ؟این کار با افزایش فضای Network و در نتیجه کاهش فضای Host انجام شد.

حالا با دیدی که از این مثال به دست آوردیم ، متوجه شدیم که برای اینکه یک Network بزرگ را به چند Network کوچکتر تقسیم کنیم، راه حل این است که قسمت Network را بزرگتر و در نتیجه Host را کوچکتر کنیم. اینکار با قرض کردن بیت های قسمت Host و اضافه کردن این بیت ها به قسمت Network صورت می گیرد.

حالا سوال این است که چند بیت را قرض بگیریم ؟

یک فرمول ساده:

اگر تعداد حداقل IP های قابل استفاده مورد نظر ما مشخص باشد، از فرمول زیر استفاده کنید:

$$2^h - 2 \geq \text{Number of available IP addressees}$$

که در آن h تعداد بیت هائی است که از سمت چپ Host می شماریم تا تعداد بیت هائی که باید قرض بگیریم معلوم شود.

اگر تعداد Subnet های مورد نیاز ، مطرح بود از فرمول زیر استفاده می کنیم :

$$2^n \geq \text{Number of Subnets Needed}$$

که در آن n تعداد بیت هائی است که باید از سمت راست از Host قرض بگیریم و به Network اضافه کنیم.

مثال) شبکه 10.0.0.0 متعلق به سازمان ما می باشد. می خواهیم آنرا طوری تقسیم کنیم که ۵ شبکه به ما بدهد.

شبکه Class A می باشد. پس:

Network	Host
00001010 .	00000000 . 00000000 . 00000000

حال فرمول را اعمال می کنیم :

$2^n >= 5$

پس $n=3$

حالا از سمت Network ، ۳ بیت به داخل Host جلو می رویم و به Network اضافه می کنیم :

Network	Host
00001010 . 000	00000 . 00000000 . 00000000

اولین موردی که باید آنالیز کنیم Subnet Mask جدید است.

11111111.11100000.00000000.00000000
255.224.0.0

به این Subnet Mask می توانیم ۱۱/ بگوئیم. N/ یعنی Subnet Mask به گونه ایست که در بخش Network آن ، N مقدار ۱ وجود دارد و قسمت Host آن 32-N صفر دارد.

حالا تمام حالت هائی را که با تغییر Network جدید (سه بیت اضافه شده) را حساب می کنیم:

Network	Host
00001010 . 000	00000 . 00000000 . 00000000
00001010 . 001	00000 . 00000000 . 00000000
00001010 . 010	00000 . 00000000 . 00000000
00001010 . 011	00000 . 00000000 . 00000000
00001010 . 100	00000 . 00000000 . 00000000
00001010 . 101	00000 . 00000000 . 00000000
00001010 . 110	00000 . 00000000 . 00000000
00001010 . 111	00000 . 00000000 . 00000000

پس شبکه های جدید ما به این صورت خواهند بود :

10.0.0.0 /11
10.32.0.0 /11
10.64.0.0 /11
10.96.0.0 /11
10.128.0.0 /11
10.160.0.0 /11
10.192.0.0 /11
10.224.0.0 /11

به عنوان تمرین شبکه سوم را آنالیز می کنیم (شبکه 10.64.0.0 /11) :

Network	Host
00001010 . 010	00000 . 00000000 . 0000000

حالا برای اینکه NetID را بدست آوریم ، تمام بیت های Host را صفر می کنیم.

Network	Host
00001010 . 010	00000 . 00000000 . 0000000

پس NetID می شود : 10.64.0.0

برای بدست آوردن آدرس Broadcast همه بیت های Host را یک می کنیم.

Network	Host
00001010 . 010	11111.11111111.11111111

پس Broadcast address می شود : 10.95.255.255

چون این دو آدرس را نمی توانیم به نود ها اختصاص دهیم ، بنابر این اولین آدرس قابل استفاده می شود یکی بالاتر از NetID به عبارتی:

Network	Host
00001010 . 010	00000 . 00000000 . 0000001

اولین آدرس این شبکه می شود : 10.64.0.1

آخرین آدرس شبکه نیز می شود یکی مانده به آدرس Broadcast یعنی:

Network	Host
00001010 . 010	11111.11111111.11111110

آخرین آدرس قابل استفاده در این شبکه می شود : 10.95.255.254

تعداد آدرس IP قابل استفاده در شبکه : ۲ به توان ۲۱ منهای ۲ می شود ۲۰۹۷۱۵۰ آدرسIP .

بنابراین در تحلیل این شبکه موارد زیر را محاسبه کردیم:

Subnet Mask	255.224.0.0
Network ID	10.64.0.0
First IP address	10.64.0.1
Last IP address	10.95.255.254
Broadcast address	10.95.255.255
Number of Available IP addresses	2097150

مثال) شبکه 172.16.0.0 را طوری Subnet کنید که در هر شبکه جدید ۳۰۰ آدرس IP قابل استفاده وجود داشته باشد. دومین شبکه بدست آمده را آنالیز کنید.

شبکه Class B می باشد . پس:

Network	Host
10101100 . 00010000 .	00000000 . 00000000

حال فرمول را اعمال می کنیم:

$$(2^h)-2 >= 300$$

پس h=9

یعنی باید ۹ بیت را حداقل به Host تخصیص دهیم. با این حساب n = 7 (تعداد بیت هایی که بایستی از host به network قرض داده شود)

حالا از سمت Network ، ۷ بیت به داخل Host جلو می رویم و به Network اضافه می کنیم:

Network	Host
10101100 . 00010000 . 0000000	0 . 00000000

اولین موردی که باید آنالیز کنیم Subnet Mask جدید است.

11111111.11111111.11111110.00000000

255.255.254.0

به این Subnet Mask می توانیم 23/ بگوئیم. حالا حالت هائی را که با تغییر Network جدید (۷ بیت اضافه شده)؛ به دست می آوریم را حساب می کنیم:

Network	Host
10101100 . 00010000 . 0000000	0 . 00000000
10101100 . 00010000 . 0000001	0 . 00000000
10101100 . 00010000 . 0000010	0 . 00000000
10101100 . 00010000 . 0000011	0 . 00000000
...	...

پس شبکه های جدید ما به این صورت خواهند بود:

172.16.0.0/23
172.16.2.0/23
172.16.4.0/23
172.16.6.0/23
…

شبکه دوم را آنالیز می کنیم:
شبکه 172.16.2.0/23 :

Network	Host
10101100 . 00010000 . 0000001	0 . 00000000

حالا برای اینکه NetID را بدست آوریم، تمام بیت های Host را صفر می کنیم. پس NetID می شود : 172.16.2.0
برای بدست آوردن آدرس Broadcast همه بیت های Host را یک می کنیم. پس Broadcast address می شود : 172.16.3.255
چون این دو آدرس را نمی توانیم به نود ها اختصاص دهیم، بنابراین اولین آدرس قابل استفاده می شود یکی بالاتر از NetID به عبارتی: 172.16.2.1
آخرین آدرس شبکه نیز می شود یکی مانده به آدرس Broadcast یعنی: 172.16.3.254
تعداد آدرس IP قابل استفاده در شبکه : 2-(2^9) می شود ۵۱۰ آدرسIP
بنابراین:

Subnet Mask	255.255.254.0
Network ID	172.16.2.0
First IP address	172.16.2.1
Last IP address	172.16.3.254
Broadcast address	172.16.3.255
Number of Available IP addresses	510

مثال) آدرس 13.13.13.0/24 را می خواهیم به ۴ شبکه ، طوری تقسیم کنیم که:
۱ - شبکه اول ، ۱۲۶ آدرس قابل استفاده
۲ - شبکه دوم ، ۶۲ آدرس قابل استفاده
۳ - شبکه سوم ، ۱۴ آدرس قابل استفاده
۴ - شبکه چهارم ، ۲ آدرس قابل استفاده
کاری که می خواهیم بکنیم این است که از بالا به پائین مسئله را حل می کنیم.
ابتدا شبکه را طوری Subnet می کنیم که به ما ۱۲۶ آدرس IP قابل استفاده بدهد:

$$(2^h) - 2 >= 126$$

پس چون h = 7 است بنابراین n = 1

Network	Host
13.13.13. 0	0000000

Subnet Mask جدید ما می شود /25
حالا حالت هائی را که با تغییر Network جدید (۱ بیت اضافه شده) ؛ به دست می آوریم را حساب می کنیم :

Network	Host
13.13.13. 0	0000000
13.13.13. 1	0000000

پس شبکه های جدید ما به این صورت خواهند بود:

13.13.13.0 /25
13.13.13.128 /25

13.13.13.0 /25 را برای شبکه اول نگه می داریم و13.13.13.128 /25 را برای سایر شبکه ها Subnet می کنیم.

حال شبکه 13.13.13.128 /25 را طوری Subnet می کنیم که به ما ۶۲ آدرس IP قابل استفاده بدهد:

$$(2^h) - 2 >= 62$$

پس چون h = 6 است بنابراین n = 2

Network	Host
13.13.13. 10	000000

Subnet Maskجدید ما می شود /26

حالا حالت هائی را که با تغییر Network جدید (۱ بیت اضافه شده) ؛ به دست می آوریم را حساب می کنیم:

Network	Host
13.13.13. 10	000000
13.13.13. 11	000000

پس شبکه های جدید ما به این صورت خواهند بود:

13.13.13.128 /26
13.13.13.192 /26

13.13.13.128 /26 را برای شبکه دوم نگه می داریم و13.13.13.192 /26 را برای سایر شبکه ها Subnet می کنیم.

حال شبکه13.13.13.192 /26 را طوری Subnet می کنیم که به ما ۱۴ آدرس IP قابل استفاده بدهد:

$$(2^h) - 2 >= 14$$

پس چون h = 4 است بنابراین n = 4

Network	Host
13.13.13. 1100	0000

Subnet Maskجدید ما می شود /28

حالا حالت هائی را که با تغییر Network جدید (۲ بیت اضافه شده) ؛ به دست می آوریم را حساب می کنیم:

Network	Host
13.13.13. 1100	0000
13.13.13. 1101	0000
13.13.13. 1110	0000
13.13.13. 1111	0000

پس شبکه های جدید ما به این صورت خواهند بود:

13.13.13.192 /28
13.13.13.208 /28
13.13.13.224 /28
13.13.13.240 /28

13.13.13.192 /28 را برای شبکه سوم نگه می داریم و13.13.13.208 /28 را برای آخرین شبکه Subnet می کنیم.

حال شبکه13.13.13.208 /28 را طوری Subnet می کنیم که به ما ۲ آدرس IP قابل استفاده بدهد:

$$(2^h) - 2 >= 2$$

پس چون h = 2 است بنابراین n = 6

Network	Host
13.13.13. 110100	00

Subnet Mask جدید ما می شود /30

حالا حالت هائی را که با تغییر Network جدید (۲ بیت اضافه شده) ؛ به دست می آوریم را حساب می کنیم:

Network	Host
13.13.13. 110100	00
13.13.13. 110101	00
13.13.13. 110110	00
13.13.13. 110111	00

پس شبکه های جدید ما به این صورت خواهند بود :

13.13.13.208 /30
13.13.13.212 /30
13.13.13.216 /30
13.13.13.220 /30

13.13.13.220 /30 را برای شبکه چهارم نگه می داریم.

Supernetting

دقیقا عمل عکس Subnetting است. یعنی چند شبکه کوچک را با هم ادغام کرده و یک شبکه بزرگ ایجاد می کنیم. نکته مهم این است که هر دو شبکه ای را نمی توان با هم Supernet کرد. به عنوان مثال آدرس IP بدست آورید که نمایانگر چهار شبکه زیر باشد:

10.128.0.0 /11
10.160.0.0 /11
10.192.0.0 /11
10.224.0.0 /11

ابتدا آدرس ها را بصورت باینری می نویسیم:

Network	Host
00001010 . 100	00000 . 00000000 . 00000000
00001010 . 101	00000 . 00000000 . 00000000
00001010 . 110	00000 . 00000000 . 00000000
00001010 . 111	00000 . 00000000 . 00000000

همانطور که از رنگ قرمز مشاهده می کنید، هر چهار شبکه در ۹ بیت اول مشترک اند، بنا بر این می توان Network را ۲ بیت از سمت چپ کم کرد. به عبارتی

Network	Host
00001010 . 1	0000000 . 00000000 . 00000000
00001010 . 1	00000 . 00000000 . 0000000001
00001010 . 1	00000 . 00000000 . 0000000010
00001010 . 1	00000 . 00000000 . 0000000011

NetID جدید را بدست می آوریم:

Network	Host
00001010 . 1	0000000 . 00000000 . 00000000

NetID جدید می شود : 10.128.0.0

Subnet Maskجدید را حساب می کنیم :

11111111.10000000.00000000.00000000
255.128.0.0
/9

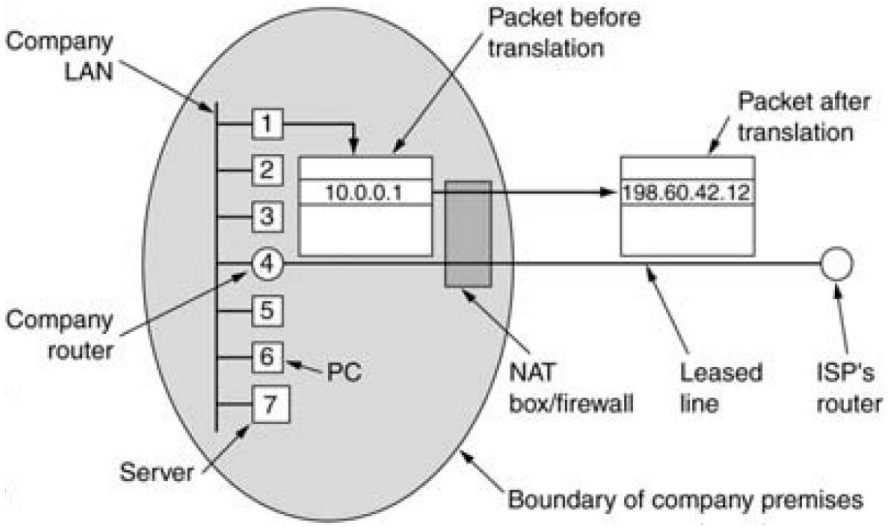
پس آدرس مورد نظر10.128.0.0/9 است.

Network Address Translation (NAT):

خیلی وقت ها اتفاق می افتد که ما می خواهیم یک آدرس IP را بین چندین نفر به اشتراک بگذاریم. به عنوان مثال فرض کنید در خانه یک لپ تاب، یک کامپیوتر و یک گوشی موبایل داریم ولی همگی توسط یک خط Adsl به ISP متصل است. این لپ تاب و کامپیوتر و موبایل از دید ناظر خارجی، همه مصرف کننده های یک خانه هستند اما در عین حال هر کدام کار جداگانه ای را مستقلاً انجام می دهند و نیاز به آدرس اختصاصی دارند. در این مثال و موارد مشابه آن لازم نیست به هر کدام آدرس IP جداگانه تخصیص دهیم. در واقع یک آدرس IP برای خانه یا

شبکه محلی تخصیص داده می شود و از دید ناظر خارجی همگی استفاده کننده های داخل این شبکه تحت یک آدرس قابل شناسایی است. و با یک آدرس IP دیده می شود.

عملکردی تحت عنوان Network Address Translation در نقطه ای از شبکه پیاده سازی می شود که بصورت زیر است:



برای تحقق این هدف بایستی یک NAT در شبکه وجود داشته باشد (در مثال ذکر شده این عمل در مودم Adsl که به مخابرات وصل است، انجام می شود). این Box (منظور همان NAT است) وقتی به ISP وصل می شود، یک آدرس IP از شبکه می گیرد. پشت سر یک NAT یک شبکه محلی با تعدادی کامپیوتر وجود دارد. به عنوان مثال در این شکل ۷ کامپیوتر در شبکه محلی وجود دارد. ابتدا NAT به هر کدام یک آدرس اختصاص می دهد (آدرس دهی غیر اتوماتیک هم وجود دارد). فرض کنید کامپیوتر اول آدرس 10.0.0.1، کامپیوتر دوم آدرس 10.0.0.2 و .... و آدرس کامپیوتر هفتم 10.0.0.7 باشد. به این آدرس ها آدرس مجازی می گویند که هیچ کس جز NAT لازم نیست آن ها را بشناسد. در دنیای واقعی نیز این آدرس ها، آدرس این کامپیوترها تلقی نمی شود.

وقتی یک کامپیوتر پکت را می سازد و اقدام به ارسال آن به شبکه می نماید، یک سری تغییرات در هدر آن توسط NAT صورت می گیرد. به عنوان مثال وقتی پکتی به مقصد سایت www.google.com ساخته شده و قرار است در شبکه ارسال گردد ابتدا NAT آدرس IP خود را به جای آدرس مجازی کامپیوتر فرستنده در بسته درج می کند. مثلاً فرض کنید آدرس مبدأ 10.0.0.1 و آدرس مقصد 81.31.71.52 باشد. قبل از ارسال به خارج از شبکه فیلد آدرس مبدأ توسط NAT به آدرس IP خود NAT تغییر می کند. مثلاً به جای 10.0.0.1 آدرس 192.60.42.12 قرار می گیرد. سپس بسته به بیرون ارسال می گردد.

source		NAT	
آدرس مبدأ	آدرس مقصد	آدرس مبدأ	آدرس مقصد
10.0.0.1	81.31.71.52	192.60.42.12	81.31.71.52

خارج از شبکه داخلی، تصور می شود این پکت ساخته ی NAT می باشد. هرگاه در جواب به این پکت، پاسخی برگردد در شبکه جهانی مسیریابی می شود و به NAT می رسد. حال بسته ای که به NAT رسیده است چگونه می تواند به مبدأ اصلی برسد؟

برای حل این مسأله از مفهوم پورت که در لایه انتقال بحث خواهد شد، استفاده می گردد. در واقع می توان گفت این کار صرفاً کار لایه ۳ نیست و آدرس پورت که در لایه ۴ می باشد به کارگرفته می شود. در لایه انتقال به هنگام برقراری یک ارتباط end to end از آدرس های لایه چهار برای هر قطعه استفاده می کنیم که آدرس پورت نامیده می شود. به عنوان مثال کامپیوتر مبدأ از پورت ۱۵۰ خود به پورت ۸۰ کامپیوتر مقصد یک اتصال برقرار کرده است.

NAT برای هر کامپیوتر و پورت اش یک شماره پورت جدید در نظر می گیرد. مثلاً در مثال ذکر شده، NAT برای کامپیوتر 10.0.0.1 که از پورت 150 خود به پورت 80 یک کامپیوتر دیگر اتصال باز کرده بود، شماره پورت 1000 را در نظر می گیرد. ولی ثبت می کند که این عدد

1000 مربوط با کامپیوتر 10.0.0.1 که از پورت 150 خود به پورت 80 یک کامپیوتر دیگر اتصال باز کرده بود. لذا جدول اخیر را به صورت زیر بازنویسی می کنیم:

NAT		**source**	
آدرس مقصد	آدرس مبدأ	آدرس مقصد	آدرس مبدأ
81.31.71.52 TCP destination Port#: 80	192.60.42.12 TCP source Port#: 1000	81.31.71.52 TCP destination Port#: 80	10.0.0.1 TCP source Port#: 150

حال وقتی بسته پاسخ در برگشت به آدرس 192.60.42.12 با شماره پورت 1000 می رسد، NAT متوجه می شود که این بسته مخصوص کامپیوتر 10.0.0.1 ای است که قبلاً با پورت 150 پکتی را ارسال کرده بوده. بنابراین NAT به جای آدرس IP خودش، آدرس 10.0.0.1 و به جای شماره پورت 1000 همان شماره پورت 150 را در پکت درج و آن را به کامپیوتر اول عودت می دهد.

فایده این روش این است که پشت NAT هر چه دلمان بخواهد می توانیم کامپیوتر داشته باشیم. کلاً پشت NAT هر چه باشد از دید ناظر خارجی پوشیده است. این روش برای استفاده مجدد آدرس های IP بکار می رود. در مثال قبلی اگر یک کامپیوتر 10.0.0.2 با پورت 150 داده ای ارسال کرد، NAT به جای آدرس آن کامپیوتر (10.0.0.2) آدرس IP خودش (192.60.42.12) و به جای شماره پورت هم یک شماره دیگر مثلاً 2000 را در نظر می گیرد تا وقتی جواب به NAT برگردد با توجه به پورت 2000 متوجه بشود که این پکت مخصوص کامپیوتر 10.0.0.2 با پورت 150 می باشد.

NAT به دلایل زیر می تواند مورد پسند واقع نگردد:

۱- به علت ماهیت Data Path بودن این عمل، اگر قدرت پردازشی NAT پایین باشد می تواند به گلوگاه تبدیل شود.

۲- نمی توان پشت یک NAT، سرور قرار داد. چرا که همانطور که در مثال ها دیدید پکت ها ابتدا از شبکه داخلی به خارج ارسال می شد تا شماره پورتی به ازای هر اتصال در NAT در نظر گرفته شود و در برگشت پاسخ، از آن استفاده می گردید. اما اگر جریان برعکس باشد، مثل زمانی که متقاضی ها به سرورها تقاضایی ارسال می کنند. اگر یک سرور در پشت یک NAT مخفی باشد، متقاضی از کجا بداند که چه شماره پورتی را برای بسته خود تنظیم کند تا به سرور پشت NAT حتما برسد.

آدرس هایی برای NAT رزرو شده که می توان در آدرس دهی شبکه داخلی از آن استفاده کرد. این آدرس ها نباید در اینترنت به عنوان آدرس های پکت IP ظاهر گردند.

10.0.0.0 – 10.255.255.255/8 (16,772,216 hosts)
172.16.0.0 – 172.31.255.255/12 (1, 0478,576 hosts)
192.168.0.0 – 192.168.255.255/16 (65,636 hosts)

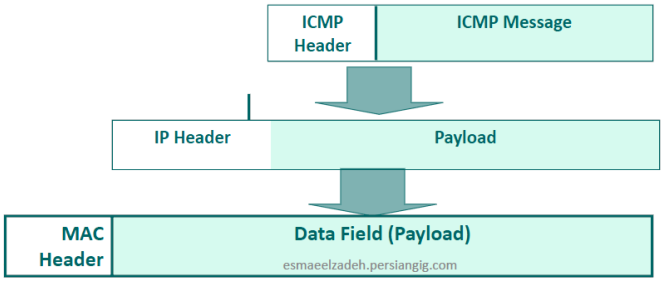
پروتکل ICMP[۱۴]

پروتکل IP ، پروتکلی بدون اتصّال و غیر قابل اعتماد است! بدون اتصّال بدین معنا که مسیریاب هر بسته را بدون هیچگونه هماهنگی با مقصد بسته یا مسیریاب بعدی ارسال می نماید، بدون آنکه بتواند اطلاعی از وجود یا عدم وجود مقصد داشته باشد. در ضمن هر مسیریاب پس از ارسال یک بسته آنرا فراموش می کند و منتظر "پیام دریافت بسته" از گیرندهٔ آن نخواهد ماند. اگر یک بستهٔ IP با خطا به مقصد برسد و یا اصلا به مقصد نرسد این پروتکل هیچ اطلاعی در مورد سرنوشت آن به فرستندهٔ بسته نمی دهد. دلایل مختلفی برای نرسیدن یک بسته به مقصد وجود دارد: ممکن است "زمان حیات" بسته قبل از رسیدن به مقصد منقضی شود؛ ممکن است مسیر یاب بسته را به مسیری اشتباه هدایت کند؛ ممکن است در هنگام قطعه قطعه کردن بسته و ارسال آن ها، یکی از قطعات دچار خطا شود یا به هر دلیلی به مقصد نرسد بنابراین کل دیتاگرام قابل بازسازی نخواهد بود؛ ممکن است مقصدِ بسته آمادگی دریافت بسته را نداشته باشد یا اصلاً وجود خارجی نداشته باشد. در هنگام بروز هرگونه خطا، پروتکل IP به فرستندهٔ بسته هیچ اطلاعی در مورد سرنوشت آن نخواهد داد.

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

[14] internet control message protocol

پروتکلICMP در کنار پروتکل IP ، برای بررسی انواع خطا و ارسال پیام برای مبدأ بسته در هنگام بروز اشکالات ناخواسته استفاده می شود. در حقیقت ICMP یک سیستم گزارش خطا است که بر روی پروتکلIP نصب می شود تا در صورت بروز هرگونه خطا به فرستندۀ بسته، پیام مناسب را بدهد تا آن خطا تکرار نشود. در واقع ICMP وظیفه ای در قبال وقوع خطا ندارد بلکه فقط پیامی که بیانگر بروز خطا و نوع آن است به فرستنده برمی گرداند. این پروتکل اشکالات موجود را در قالب یکسری پیام گزارش می کند که این پیام خود در یک بسته IP قرار می گیرد که از جانب یک مسیریاب یا ماشین مقصد به آدرس فرستنده باز می گردد.



پیام Destination Unreachable :

عدم تشخیص آدرس توسط مسیریاب و یا زیر شبکه

نرسیدن بسته به مقصد به هر علت

پیام Time Exceeded

ارسال پیام به فرستنده بسته جهت آگاهی از اتمام طول عمر بسته و حذف آن توسط مسیریاب

پیام Parameter Problem

نشان دهنده وجود مقدار نامعتبر در یکی از فیلدهای سرآیند بسته IP

پیام Source Quench

تقاضای کاهش نرخ تولید و ارسال بسته های IP از ماشین میزبان

پیام Redirect

وجود اشکال در مسیریابی

پیام های Echo Request , Echo Reply

پیام Echo Request : موجود و قابل دسترس بودن یک ماشین خاص در شبکه توسط مسیریاب

پیام Echo Reply: پاسخ مقصد مبنی بر دریافت پیام Echo Request

پیام های Timestamp Request و Timestamp Reply

دریافت کننده پیام Timestamp Request زمان دریافت و زمان ارسال بسته را نیز مشخص می کند.

پروتکل ARP [۱۵]

نکته ظریفی که در مورد شبکه اینترنت وجود دارد آن است که اگر چه تمامی ماشین های میزبان و ابزارهای شبکه ای از آدرس IP که آدرس منحصر به فرد و یکتا است استفاده می کنند ولیکن یک بسته IP فقط در لایه شبکه قابل شناسائی و تحلیل است. یک بسته IP قبل از ارسال

[15] Address Resolution Protocol

روی کانال از لایه اول یعنی لایهٔ فیزیکی عبور می کند و ضمن اضافه شدن اطلاعات لازم و تشکیل یک فریم، روی کانال فیزیکی ارسال می شود. به عبارت روشن تر بسته IP قبل از ارسال درون فیلد داده از فریمی قرار می گیرد که بعداً در لایه اول تشکیل می شود؛ لایه اول وظیفه ای در قبال مسیریابی و کارهائی از این قبیل ندارد و فقط با آدرس های فیزیکی کار می کند. به عنوان مثال اگر ماشین شما بخواهد بسته ای را برای ماشینی که روی شبکه محلی خودتان واقع است بفرستد، در لایه اول الزاماً بایستی آدرس فیزیکی ماشین شما (مبداء) و آدرس فیزیکی ماشین طرف مقابل (مقصد) معین باشد. (این آدرس ها بصورت سخت افزاری در کارت شبکه درج شده است) عدم دانستن آدرس های فیزیکی عملاً مساوی عدم توانایی برای ارتباط خواهد بود چرا که روی کانال انتقال آدرس های IP بی معنا هستند.

وظیفه پروتکل ARP در اینجا آن است که یک "بسته فراگیر[16]" روی کل شبکه محلی منتشر کند که این بسته در حقیقت سوال می کند:

" کسی که آدرس IP او فلان است ، آدرس فیزیکی او چیست؟"

با توجه به آنکه بسته های فراگیر توسط تمامی ماشین های روی شبکهٔ محلی دریافت می شود، ماشینی که آدرس IP خودش را درون این بسته می بیند، بدان پاسخ می دهد و آدرس فیزیکی خود را برای ارسال کنندهٔ آن بسته می فرستد. پس از آنکه آدرس فیزیکیِ مقصد بدست آمد، یک فریم اترنت ساخته شده بر روی کانال منتقل می شود.

مثال: فرض کنید تعدادی کامپیوتر در یک شبکه محلی به هم وصل هستند. هر گاه کامپیوتری بخواهد برای کامپیوتر دیگر بسته ای را ارسال کند در لایه۳ نیازمند درج آدرس های IP فرستنده و گیرنده و در لایه ۲ نیازمند این است که در فریم، آدرس MAC مبدأ و مقصد را درج نماید. فرض کنید به هر دلیلی آدرس IP کامپیوتر مقصد را می داند. در ضمن ادرس های IP و MAC خود را که قطعاً می شناسد. فقط تنها چیزی که باقی می ماند آدرس MAC کامپیوتر مقصد است.

پروتکل ARP برای حل این مشکل پیاده سازی شده است.

روش اول:

زمانی که کامپیوتر اول می خواهد داده ای برای کامپیوتر دیگری بفرستد که از MAC آن بی خبر است، ابتدا پیامی می فرستد که در آن آدرس IP خود و آدرس IP گیرنده را درج می کند و این پیام تقاضایی است که بیان می کند هر کس آدرس MAC مورد نظر را دارد به وی خبر دهد. این پیام یک فرمت خاص دارد و در شبکه محلی پخش همگانی می شود. بنابراین همه ی نودهایی که در شبکه هستند مجاز هستند این پیام را بردارند و بخوانند. فرض کنید در این شبکه ۶ کامپیوتر باشد که همگی به جز کامپیوتر مد نظر با بررسی این پیام می بینند که IP هیچکدامشان با IP موجود در بسته تقاضا یکی نیست. اما کامپیوتر مقصد وقتی پیام تقاضا را می گیرد، متوجه می شود که یک نفر تقاضای آدرس MAC او را کرده است. لذا در پاسخی برای متقاضی آدرس MAC تقاضا شده برای IP مشخص شده در پیام را می فرستد.

این عمل دو فایده دارد:

۱- source آدرس MAC گره مقصد را می فهمد و می تواند فریم لایه ۲ خود را بسازد و در شبکه ارسال کند.

۲- به هنگام پخش همگانی تقاضای کامپیوتر اول، سایر نودهای شبکه با آدرس MAC کامپیوتر اول آشنا می شوند و می توانند با ثبت آن در جداول خود در استفاده های بعدی از آن استفاده کند. بنابراین در خلال تقاضای نودهای دیگر، جداول کل نودها در شبکه محلی می تواند تکمیل گردد. این جداول به فرم زیر است:

IP Address	MAC Address	Expiring

روش دوم:

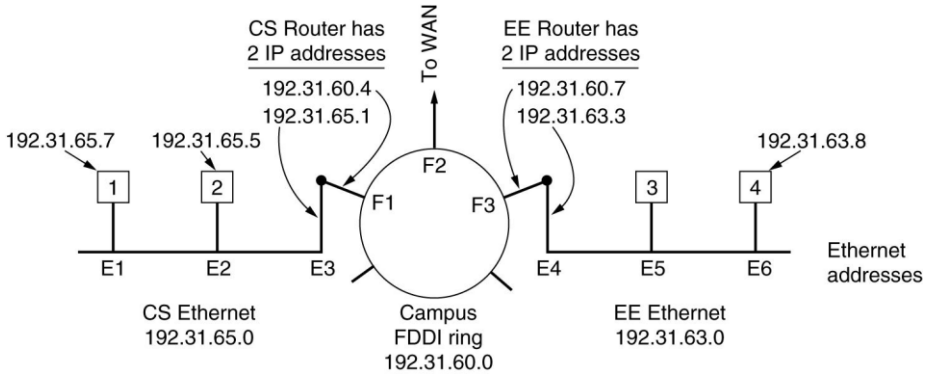
در هنگام Boot شدن، هر کامپیوتر یک ARP Message می سازد و در آن پیام آدرس IP و MAC خود را به عنوان متقاضی درج می کند و با یک تقاضای سوری به جست جوی MAC معادل IP خود می پردازد. این پیام به علت اینکه در شبکه broadcast می شود، همگی از آدرس IP و MAC آن باخبر می گردند.

روش سوم:

به صورت پریودیک نودهای شبکه پیام های ARP بدهند تا آدرس IP و MAC خود را به بقیه اعلام نمایند. تا بدین وسیله اعتبارشان در نودهای دیگر renew شود.

[16] broadcast

سوال: اگر کامپیوتر مورد نظر ما در شبکه محلی نباشد چه اتفاقی می افتد؟ به عنوان مثال کامپیوتر شما یک تقاضا برای سایت یاهو در شبکه محلی پخش می کند. به عبارتی در پیام تقاضای خود آدرس IP را نوشته است که در شبکه محلی وجود ندارد. در این حالت یک نود هست که Gateway شبکه می باشد. این نود به متقاضی پاسخ می دهد. در واقع این نود در جایی قرار گرفته است که بسته شما را در لایه ۳ مسیردهی می کند. بسته ای که برای سایت یاهو از کامپیوتر متقاضی ارسال می گردد ابتدا دست این روتر می رسد که نقش واسط بین شبکه های دیگر را بازی می کند و می خواهد در لایه ۳ آن را مسیردهی کند. پس بایستی آدرس پورتی از روتر که متصل به شبکه محلی است را داشته باشد تا بسته را به مقصد آن پورت ارسال نماید. یعنی از دید کامپیوتر متقاضی، آدرس MAC مقصد بایستی آن پورت از روتر باشد. وقتی بسته به روتر می رسد با توجه به آدرس IP آن در مسیر درست به سمت سایت یاهو فوروارد خواهد شد.



پروتکل DHCP[۱۷]:

پروتکلی برای تخصیص پویای آدرس IP در شبکه های محلی است. برای تخصیص IP ، می توان بصورت استاتیک به هنگام بوت شدن یک کامپیوتر این کار را انجام داد. اما در DHCP سرور این کار بصورت داینامیک انجام می شود. شیوه کار بدین شرح است:

هر کامپیوتر به هنگام بوت شدن در شبکه محلی پیام تقاضای مخصوصی می سازد و در آن ذکر می کند که کامپیوتر جدیدی است که در شبکه بوت شده است و نیاز به آدرس IP دارد. سپس این پیام را در شبکه Broadcast می کند. سرور DHCP ، پس از شنیدن این پیام و دیدن آدرس لایه دو (MAC) کامپیوتر متقاضی، یک آدرس IP در صورت امکان برای مدت زمان مشخصی به آن تخصیص داده و این این آدرس را در قالب پیام پاسخی به وی برمی گرداند.

به این کار اصطلاحاً اجاره کردن یا Leased کردن آدرس IP می گویند. یعنی یک آدرس IP را برای مدت زمان معینی به یک کامپیوتر خاص تخصیص دهیم.

هر کامپیوتر بایستی قبل از اتمام مهلت در خواست تجدید آدرس IP را از سرور DHCP بکند تا سرور DHCP آن آدرس را renew یا تمدید نماید. در صورتی که این کار صورت نگیرد آدرس IP که مهلت اعتبار آن گذشته و درخواست تمدیدی هم برای آن نیامده، توسط سرور DHCP آزاد می گردد. سرور DHCP لیستی از آدرس های IP آزاد یا تخصیص نیافته را دارد تا در صورت تقاضای سایرین از این لیست به متقاضیان آدرس IP را تخصیص دهد.

اگر از سرور DHCP یک آدرس IP گرفتیم و یک کامپیوتر دیگر در همین شبکه بصورت دستی یا استاتیک همان آدرس را برای خود انتخاب کند رخداد IP address conflict بروز می کند که شبکه می تواند سیاست های مختلفی را در پی این رخداد داشته باشد.

مسیریابی در اینترنت

در بحث روتینگ مباحث گذشته مفاهیمی بیان شد و وارد توصیف IP شدیم. در بررسی یک پروتکل به دو دیدگاه Data path و Control path پرداختیم. مهمترین عملکرد حاکم بر IP عمل روتینگ یا مسیریابی است که جزء دیدگاه Control path محسوب می شود.

در روتینگ دنیای اینترنت دو نوع نگاه داریم:

[17] Dynamic Host Configuration Protocol

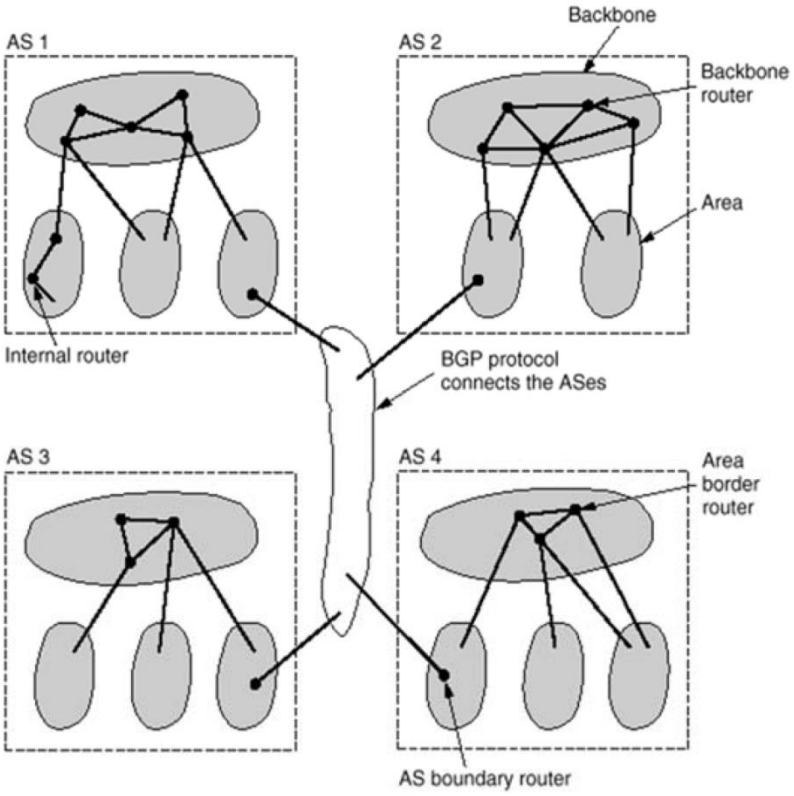
۱- روتینگ های درون یک محدوده یا Zone : شبکه اینترنت از تعداد بسیاری **سیستم خود مختار**(Autonomous System) یا اختصاراً AS تشکیل شده است. مسیریابی درون یک AS را مسیریابی دورنی (Interior routing) می نامیم.
۲- روتینگ های بین AS های مختلف: مسیریابی بین AS ها را مسیریابی خروجی یا بیرونی (Exterior routing) گویند.
یک AS محدوده ای از نودها است که همه ی نودهای آن مجموعه توپولوژی Zone را می دانند تا مسیریابی بهتری انجام دهند. اما AS های مختلف لزومی ندارند که توپولوژی داخلی سایرین را بدانند. همانند روتینگ سلسله مراتبی که در آن لازم نیست توپولوژی داخلی یک دانشگاه در سطح استان دانسته شود ولی در عوض کافیست Gateway دانشگاه را بشناسند که کل پکت های مجموعه را تحویل آن بدهند.

مسیریابی دورنی (Interior routing)

از ابتدای پیدایش اینترنت تا سال ۱۹۷۹ الگوریتم های Distance vector بکارگرفته می شدند. اما به دلیل کندی convergence و عدم مقیاس پذیری بالا که داشت، از سال ۱۹۷۹ به بعد الگوریتم های link state بکار گرفته می شدند و کم کم به فکر الگوریتم های جدید افتادند.

:OSPF

چیزی که امروزه بصورت وسیع استفاده می شود OSPF[۱۸] می باشد که از دهه ۱۹۹۰ پیدایش شده و یک استاندارد باز است و انواع متریک ها را پشتیبانی می کند. در این پروتکل به مباحثی مثل روتینگ های سلسله مراتبی، توازن بار (load balancing)، پشتیبانی از Tunneling، مسائل امنیتی می پردازد و به مسائلی از قبیل انواع سرویس دهی مثلاً ترافیک real time و روتینگ های پویا یا قابل تطبیق با شرایط می پردازد. شکل زیر تصور طراحان روتینگ در دنیای اینترنت می باشد.

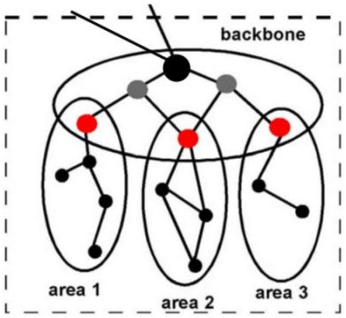


یک AS داخل خودش می تواند به چند Area تقسیم شود. هر AS یک Area خاص دارد که نقش backbone برای Area های دیگر را بازی می کند و به آن Area0 گویند. هرگاه از یک Area به Area دیگر چیزی بخواهد ارسال شود بایستی از Area0 عبور کند. Area ها در درون خود روتینگ را بصورت پویا انجام می دهند و هرگاه بخواهد بسته ای به یک Area دیگر روت شود بایستی از backbone درون یک AS رد شود.
در ادبیات پروتکل OSPF ، سلسله مراتب تعیین شده برای نواحی بصورت زیر می باشد:

[18] Open Shortest path First

یک شبکهٔ خودمختار (AS) به تعدادی ناحیه تقسیم می شود. تمام مسیریاب های درون یک ناحیه باید مسیریاب های هم ناحیهٔ خود و هزینهٔ ارتباط بیـــن آن ها را بداننـد و در جدولـی ذخـیره کنند. در لحظات به هنگام سازی، این جداول برای تمام مســـیریاب های هم ناحیـه ارسـال خواهـد شد. مسیریاب هیچ اطلاعی از وضعیت مسیریاب های درون نواحی دیگر ندارد. در شکل زیر این مسیریاب ها با علامت • نشان داده شده است. درون هر ناحیه یک یا چند مسیریاب وجود دارند که ارتباط بین نواحی را برقرار می کننـــد؛ به آن ها ، "مسیریاب های مرزی" گفته می شود. مجموعهٔ مسیریاب های مرزی و مســـیریاب هایی کـه در خارج از هر ناحیه نقش توزیع ترافیک بین نواحی را بر عهده دارند (به همراه ســـاختار ارتبـاطی بیـن ایـن مسـیریاب ها "سـتون فقرات" شـبکهٔ AS را تشـکیل می دهـد.

درون ستون فقراتِ شبکهٔ AS ممکن است مسیریاب هایی وجـــود داشـته باشـند کـه بـا دیگـر شبکه های AS در ارتباط باشد. به این مسیریاب ها "دروازه های مرزی" یا BGP گفته می شـــود. در شکل زیر این مسیریاب با دایرهٔ بزرگ و تیره رنگ نشان داده شده است.



سلسله‌مراتب مسیریابی در پروتکل OSPF

توپولوژی داخل As، تعداد Area ها و چگونگی مسیردهی بسته ها درون AS از مسائل داخلی خود AS می باشد. هر گاه چند AS بخواهند با هم در ارتباط باشند، از Exterior routing و backbone ای که AS ها را به هم متصل نموده استفاده می کنند.

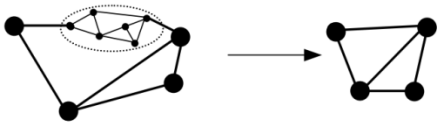
OSPF در Interior routing از مفاهیم اولیه الگوریتم های Link State استفاده می کند بدین صورت که هر کس همسایه هایش را می شناسد با ارسال hello packet و گرفتن جواب دانسته هایش را برای همگی Broadcast نماید. پس همگی توپولوژی داخلی را می شناسند. در OSPF با متریک های مختلفی (Troughput, delay, reliability, cost) می توان shortest path را درست کرد.

مسیریابی خروجی یا بیرونی (Exterior routing)

در Exterior routing ملاحظاتی که الزاماً فنی نیستند دخیل می شوند. مثلاً چه پکتی از کدام Zone اجازه عبور دارد یا نه. همانگونه که مؤکداً اشاره کردیم در شبکه ای مثل اینترنت، مسیریابی در درون یـــک شـبکهٔ خودمختار (AS) بـــا مسـیریابی خـارج از شـبکه های خودمختـار، کـاملاً متفـاوت اسـت و از پارامترهای متنوع و متناقضی تبعیت می کند. "مسیریابی برونی" نه تنها تابع شـــرایط ترافیکـی، توپولوژیکی ، پهنای باند و سرعت پـــردازش مسـیریاب ها اسـت ، بلکـه از یکسـری سیاسـت های اقتصادی، امنیتی و ملّی نیز تاثیر می پذیرد. شاید این نکته جالب باشد که مسیریاب هائی کـــه بیـن شبکه های خودمختار عمل می کنند نه تنها بایستی در مورد بهترین مسیر از لحاظ تاخیر کمـــتر و سرعت بیشتر تصمیم بگیرند بلکه بایستی در این تصمیم گـــیری مسـائل سیاسـی، اقتصـادی و امنیتی را نیز بعنوان جزئی از الگوریتم دخالت بدهند.

:BGP

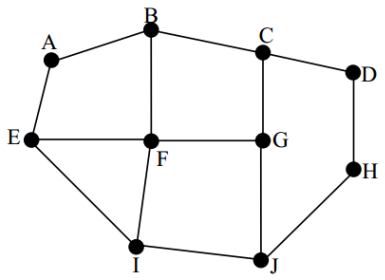
الگوریتم های مسیریابی بیـــن شـبکه های خودمختـار در اینـترنت ، BGP[19] نـامیده می شـود و تمامی این پارامترها را در تصمیم گیری لحاظ می کند. شبکه های خودمختار را یک ستون فقرات خارجی به هم متصل کـــرده کـه مسـیریاب های نـوع BGP بـر روی آن قـرار گرفتـه اند. بنـابراین مسیریاب های نوع BGP از طریق ستون فقرات شبکهٔ اینترنت با مسیریاب های دیگر از نوع BGP در ارتباطند . ممکن است بین دو مسیریاب از نوع BGP یک شبکهٔ خودمختار قرار گرفته باشد کـه کل آن شبکه یک خط واحد در نظر گرفته می شود. همانند شکل زیر:



[19] Border Gateway Protocol

حـال بـه اختصار بررسی می کنیم که این مسیریاب ها چگونه عمل می نمایند:
در پروتکل BGP بجای آنکه جداول مسیریابی و هزینه ها بیـــن مسـیریاب های مجـاور مبادلـه شود ، در بازه های زمانی T ثانیه ای ، فهرستی از مسیرهای کامل بین هر دو مسیریاب در شبکه ، برای مسیریاب های مجاور ارسال می شود. (بدون تعیین هزینه)
برای روشن شدن قضیه فرض کنید در شکل زیر مســیریاب F از مسـیریاب های مجـاور خود یعنی G، I، B و E ، اطلاعاتی در مورد D بصورت زیر دریافت کند:

From B : "BCD"
From G : "GCD"
From I : "IFGCD"
From E: "EFGCD"



ساختار فرضی از ارتباط بین مسیریابهای BGP

البتـه از B و G و I و E مسـیرهای دیگـری کـه بـه سـایر مسـیریاب ها وجـــود دارد ، اعــلان می شود ولی برای سادگی در مثال فوق، فقط مسیرهائی که بـــه D ختـم می شـوند ارائـه شـده است. مسیریاب F برای یافتن راهی به D چهار مسیر پیشـــنهاد شـده ، توسـط همسـایگانش را بررسی می کند و همان ابتدا مسیرهای I و E را کنار می گذارد چرا کـــه ایـن مسـیرها از خـود F عبور می کنند و بنابراین مسیری مستقل محسوب نمی شوند. بنابراین F برای رسیدن به D فقط دو راه خواهد داشت که یکی BCD و دیگری GCD اســت و بایستی یکی از این دو مسیر را انتخاب کند.
در اینجاست که باید تمامی پارامترهای ترافیکـــی، توپولوژیکی، اقتصادی، سیاسی و امنیتی برای انتخاب بهینه ترین مسیر، محاسبه شود. مثلاً اگر سیاست بر این باشد که بسته هایی که از F به D می روند نباید از C عبور کنند، F متوجه می شود که هیچ مسیری به D ندارد.
الگوریتم BGP مشکل "شمارش تا بینهایت" را نخواهد داشت: بعنوان مثال فرض کنیـــد خـط بین F و G قطع شود. F اطلاعاتی را از سه مسیریاب مجاور دریافت می کند که اعلان کرده انـــد برای رسیدن به D، از مســـیرهای EFGCD، IFGCD، BCD اسـتفاده می کننـد و چـون در بیـن مسیرهای EFGCD، IFGCD ، خط FG قرار دارد، بنـــابراین نمی تواننـد بـه عنـوان مسـیرهایی مستقل مورد استفاده قرار گیرند و تنها مسیر باقیمانده BCD خواهد بود. الگوریتم هائی که در تبادل اطلاعات با همســـایگان مسـیرهای کـامل را بـه اطـلاع یکدیگـر می رسانند اولاً مشکل "شمارش تا بینهایت" را نخواهد داشت؛ ثانیاً مسیریاب های دیگـــر می تواننـد بر روی کل مسیر، بررسی های امنیتی، اقتصادی، سیاسی و ملی انجام دهند و بر اســـاس ایـن پارامترها مسیر مناسب را انتخاب نمایند.

:IPv6

پروتکلی است که استفاده از آن به شدت در حال توسعه است. تمام شدن فضای آدرس در IPv4 دلیلی برای پیدایش ورژن ۶ شد. در طراحی ورژن ۶ از تجارب ورژن ۴ استفاده شده است. اهدافی از قبیل پشتیبانی از میلیارد ها ماشین میزبان، کاهش اندازه جداول مسیریابی، ساده سازی پروتکل به منظور افزایش سرعت پردازش مسیریاب ها، ارائه امنیت بهتر در مقایسه با نسخه فعلی شامل احراز هویت و سری ماندن داده ها، توجه بیشتر به نوع خدمات و QoS به ویژه برای داده های بی درنگ، کمک به فرآیند ارسال چند پخشی از طریق توصیف حوزه ها، فراهم آوردن امکان جابجایی ماشین های میزبان بدون تغییر در آدرس، امکان ایجاد تغییر و پیشرفت در آینده و امکان همزیستی پروتکل های جدید و قدیم در طی سال ها در IPv6 دنبال شده است.

ویژگی های IPv6:

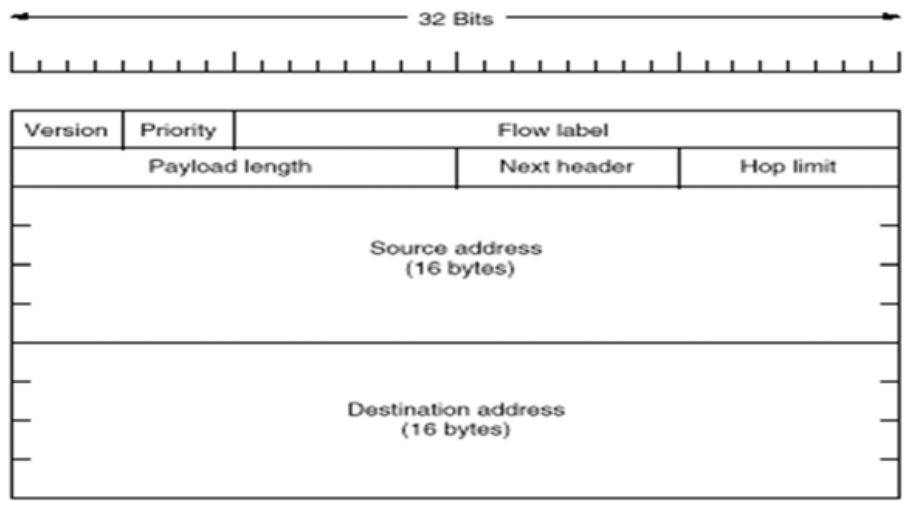
۱. IPv4 از ۴ بایت برای آدرس دهی استفاده می کند ولی IPv6 از ۱۶ بایت.
۲. طول هدر در IPv6 کاملاً فیکس می باشد تا پیاده سازی های سخت افزاری کاراتری را بتوان انجام داد. در عوض در IPv4 طول هدر متغیر بود. همچنین ساده سازی از ۱۳ فیلد سرآیند در IPv4 به ۷ فیلد، که باعث سریع تر شدن پردازش بسته ها توسط مسیریاب، افزایش ظرفیت مسیریاب و کاهش تاخیر شده است.
۳. پشتیبانی از گزینه های اختیاری در ورژن ۶.
۴. امنیت که شامل احراز هویت و حفظ امنیت اطلاعات می باشد.
۵. کیفیت خدمات در IPv6 خیلی جدی تر از IPv4 در نظر گرفته شده است.
۶. پشتیبانی از مفهومی تحت عنوان Flow که در ورژن ۴ وجود نداشت.
۷. امکان انتقال سایز خیلی بزرگ بسته ها در ورژن ۶ برای اتصالاتی که نیاز به Troughput زیاد دارند و کارکردن با سایز کوچکتر برایشان خوب نیست.

Flow چیست؟

هر جریان داده ای که از یک مبدأ به یک مقصد با با یک پروتکل خاص مشخص می شود، Flow نامیده می شود. چیزی که در یک Flow از اهمیت زیادی برخوردار است این است که بتوانیم QOS را فراهم کنیم و لازمه این امر این است که از قبل منابع لازم از قبیل پهنای باند، سایز بافر و زمان پردازش را به این کار اختصاص دهیم.

//**برای مطالعه بیشتر**//

هدر بسته های IPv6:



فیلد Version:

شماره نسخه پروتکل را نشان می دهد

Priority:

برای تشخیص تفاوت بسته ها، که اولویت آن را مشخص می کند.

Flow Label

یک شناسه برای Flow است. هر بسته به محض رسیدن به یک روتر، اگر از قبل مسیری برای آن در نظر گرفته شه باشد از روی شماره پورت ورودی و برچسبی که دارد تشخیص داده می شود و منابع لازم در اختیارش قرار می گیرد. بنابراین اولویت لازم در شبکه به آن داده می شود. Flow lable کاملاً مستقل از آدرس مبدأ و مقصد است که در ATM بیشتر بحث خواهیم کرد.

Payload length:

طول قسمت حمل داده می باشد که مشخص می کند پس از سرآیند ۴۰ بایتی چند بایت داده قرار گرفته است.

Next Header:

مشخص می کند که پس از سرآیند ۴۰ بایتی کدامیک از سرآیند های شش گانه اضافی قرار گرفته است. اگر این سرآیند آخرین سرآیند بسته باشد این فیلد مشخص می کند که کدام پروسه در لایه انتقال محتوای بسته را تحویل خواهد گرفت.

Hop Limit:

طول عمر بسته بر حسب تعداد گام

Source Address , Destination Address:

آدرس هاس ۱۶ بایتی IP می باشند که با توجه به طول زیاد آدرس ها نماد جدید برای نوشتن آن ها پیشنهاد شد:

8000:0000:0000:0000:0123:4567:89AB:CDEF

از آنجایی که تعداد صفر ها زیاد است بهینه سازی های زیر مطرح شد:

۱. صفرهای سمت چپ هر گروه نوشته نمی شود یعنی 0123 تبدیل به 123 می شود.

۲. اگر یک یا چند گروه ۱۶ بیتی تماماً صفر باشند با :: نشان داده شود.

8000::0123:4567:89AB:CDEF

آدرس های IPv4 به صورت زیر نوشته می شوند:

::192.31.20.46

//**انتهای برای مطالعه بیشتر**//

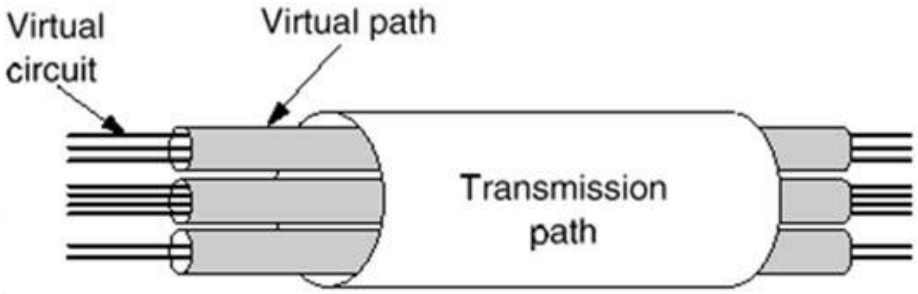
ATM:

پروتکل یا پشته پروتکلی است که در دهه ۹۰ خیلی روی آن کار شده است و پخش مهمی از توسعه دنیای Packet بر مبنای ATM صورت گرفته است. اما هم اکنون استفاده از ATM در حال کاهش می باشد. اما از ایده های ATM هم اکنون استفاده می شود.

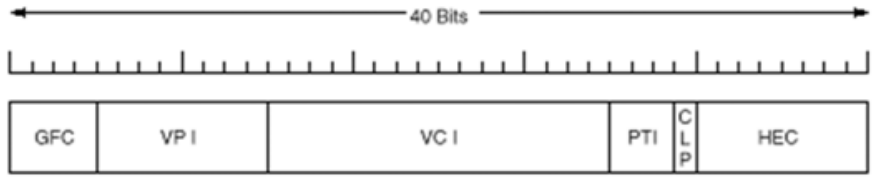
یکی از اهداف ATM، ایجاد backbone ای است که بتواند همزمان voice, video, data را همزمان با کیفیت خوب در شبکه سرویس دهد. که این امر الزامات پیچیده ای پیدا می کند. راه حل تضمین کیفیت خدمات، ایجاد connection در شبکه است. برای همین اساس ATM بر مبنای درست کردن connection می باشد.

در ATM ساخت Connection بصورت سلسله مراتبی صورت گرفته است. در مسیر بین Switch های ATM، دو سطح از connection داریم که باعث کارایی بیشتر عمل سوئیچینگ می شوند:

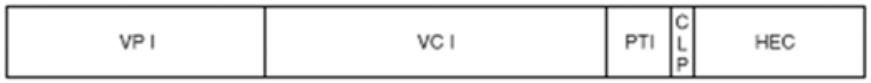
۱. Connection های Virtual Path که دسته بزرگتری را شامل می شوند.
۲. Connection های کوچکتری که داخل Virtual path هستند و به آن ها Virtual Circuit گوییم.



بسته های ATM، تحت عنوان cell شناخته می شوند. در ATM وقتی cell ها جابجا می شوند، ACK در لایه روتینگ نمی گذاریم. اگر بسته ها به مقصد رسیدند که چه بهتر ولی اگر نرسیدند، مهم نیست. در ATM سعی بر این شده که کیفیت خدماتی که تضمین شده را ارائه دهد، اما اگر به هر دلیلی بسته ای به مقصد نرسید، در این لایه کاری به آن نداریم (همانند IP). اگر لازم باشد بسته هایی با گارانتی رسیدن به مقصد در شبکه جابجا شوند، این گارانتی توسط لایه بالایی داده خواهد شد. در بالای لایه ATM، لایه ای داریم تحت عنوان ATM Adaption Layer (AAL)، که در آن انواع و اقسام پروفایل هایی تعریف شده است. به عنوان مثال پروفایلی دارد که cellها را بصورت reliable تعریف می کند و یا برعکس پروفایلی داریم که reliable نیست. بنابراین تضمین رسیدن بسته به مقصد در لایه ATM بصورت unreliable می باشد.
در ATM بسته ها حتماً به ترتیب تحویل مقصد داده می شوند یعنی call ها مرتب می رسند. اما در IP اینگونه نبود چراکه IP بسته ها را از راه های مختلف به مقصد می رساند. در ATM بسته ها به همان ترتیبی که در لوله گذاشته می شوند، به همان ترتیب به مقصد تحویل داده می شوند. هر cell سایز 53 byte ای دارد. 5 بایت اول هدر بسته ATM است. در شکل زیر دو نوع فرمت با کمی تفاوت را مشاهده می کنید.
UNI(user – network interface): واسط کاربر- شبکه
NNI(network-network interface): واسط شبکه - شبکه



The ATM layer header at the UNI.



The ATM layer header at the NNI.

فیلدهای هدر:

GFC: فقط در UNI وجود دارد و کنترل جریان بین host و شبکه را انجام می دهد (در عمل استفاده نمی شود).

VPI (virtual path ID): شناسه مسیر مجازی می باشد در شکل، همان لوله بزرگ می باشد که شناسه هر کدام از آن ها می باشد. در مدل NNI، ۱۲ بیت و در مدل UNI، ۸ بیت می باشد.

VCI (virtual circuits ID): شناسه مدار مجازی می باشد که در شکل لوله کوچکترها هستند وVirtual circuit نامیده می شوند و ۱۶ بیت می باشد.

هر cell که در ATM جابه جا می شود، هم شماره شناسه لوله بزرگ و هم شناسه لوله کوچکتر را دارد. به عبارتی شناسه مسیر VC , VP روی هر cell ثبت شده است.

PTI (payload type): بر اساس payload های مختلف، شناسه های مختلف می گیرد. همچنین اطلاعاتی در مورد ازدحام شبکه در آن ثبت می شود.

مثلاً دو نوع داده 111 ,000 داریم که نوع 1 , 0 هستند. این که کجا صفر یا یک می گذاریم، مربوط به جزئیات عملکرد خود سوئیچ ها می باشد. اما به همراه آن، این نوع داده ها حاوی این پیام هستند که ازدحامی برای این cell وجود ندارد. ولی اگر cell دچار ازحام شود، به بعدی اعلام می کند تا متوجه شود که نود قبلی اش دچار ازدحام شده است (با 011 , 010 =PTI) تا در صورت لزوم در مسیر معکوس یا بواسطه پروتکل های دیگری که بتوانند شرایط ازدحام را رفع کنند، اقدامی انجام دهند.

انواع دیگری هم هستند که اطلاعات داده ای کاربر نیستند بلکه اطلاعات کنترلی هستند که خود سوئیچ ها بین هم ردوبدل می کنند.

100: اطلاعات مربوط به نگهداری مسیر بین سوئیچ های همجوار.

101: اطلاعات مربوط به نگهداری مسیر بین سوئیچ های مبدأ و مقصد.

110: برای کنترل ازدحام ABR استفاده می شود. (ABR در ادامه توضیح داده خواهد شد).

111: برای عملکردهای آتی رزرو شده است.

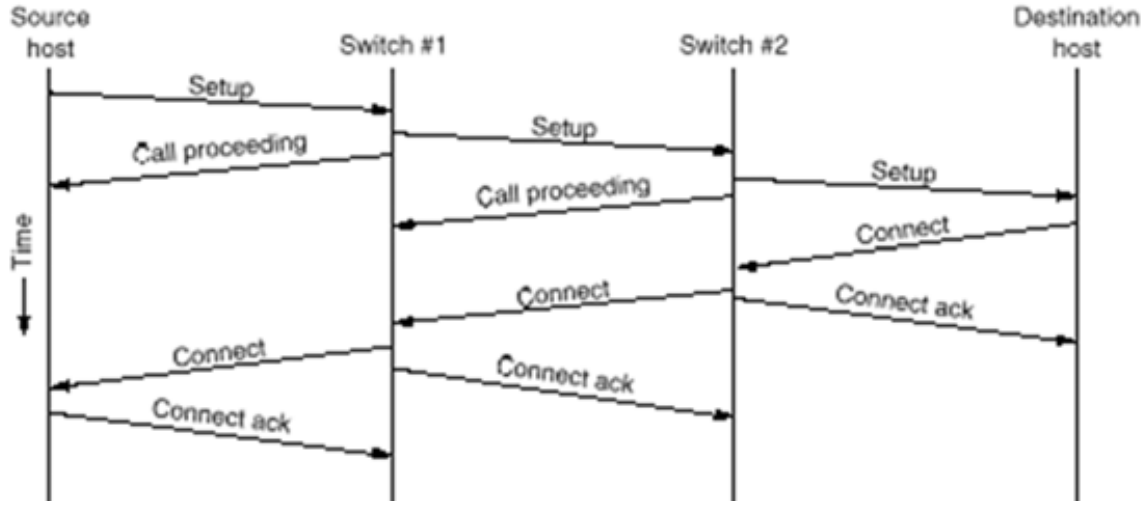
Payload type	Meaning
000	User data cell, no congestion, cell type 0
001	User data cell, no congestion, cell type 1
010	User data cell, congestion experienced, cell type 0
011	User data cell, congestion experienced, cell type 1
100	Maintenance information between adjacent switches
101	Maintenance information between source and destination switches
110	Resource Management cell (used for ABR congestion control)
111	Reserved for future function

CLP (cell loss priority): دو مقدار 0 و 1 را می گیرد. Cell های با اولویت بالا با 0 و cell های با اولویت معمولی با 1 مشخص می شوند. هرگاه سوئیچ ATM برای رفع ازدحام بخواهد یک سری cell را دور بریزد، کم اولویت ها را انتخاب می کند.

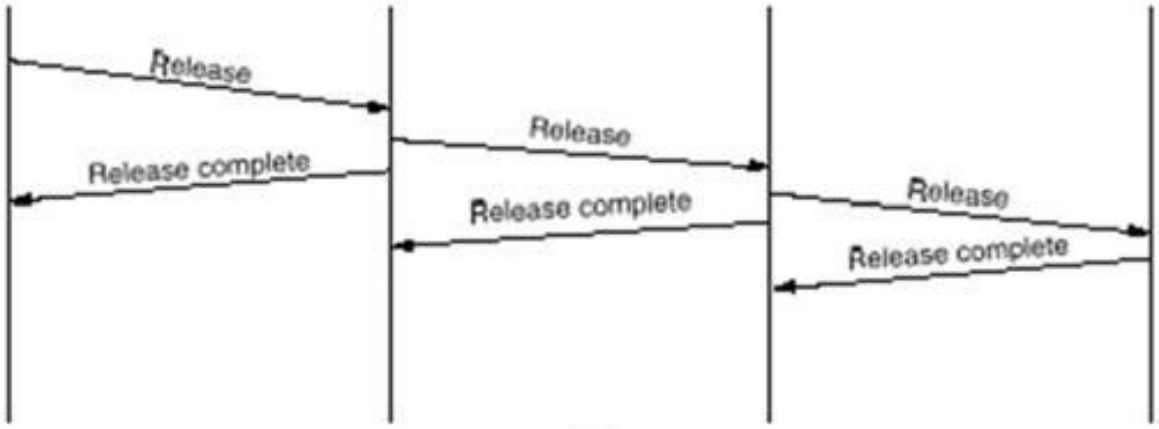
HEC(header checksum): بررسی خطا در ۴ بایت قبلی و امکان تصحیح خطاهای یک بیتی را دارد. و البته ۹۰ درصد خطاهای چند بیتی را نیز تشخیص می دهد.

ATM Connection setup:

هرگاه در ATM بخواهد بین مبدأ و مقصد یک connection برقرار گردد، بایستی یک setup انجام گردد (همانند سیستم تلفن). مبدأ به سوئیچ اول تقاضای connection را می دهد، سوئیچ اول به سوئیچ بعدی این تقاضا را می رساند. این تقاضای setup ادامه پیدا می کند تا به مقصد برسد. هر سوئیچ با دریافت پیام تقاضا، به نود قبلی اعلام می کند که تقاضایش را پردازش می کند. پیام setup به محض رسیدن به آخرین نود (مقصد)، با پیام connect پاسخ داده می شود. به عبارتی، نود آخر وقتی می خواهد جواب مثبت برای برقراری ارتباط بدهد با یک پیام connect به سوئیچ قبلی اش، موافقت خود را اعلام می نماید. هر سوئیچ هم به محض دریافت پیام connect، با پیام connect ack اعلام می کند که از این به بعد بین خودش و سوئیچ ای که پیام connect به وی داده یک مسیر برقرار شده است. سپس خودش یک پیام connect به سوئیچ قبلی خود می فرستد و connect ack آن را می گیرد و یک مسیر بین این دو سوئیچ نیز برقرار می گردد. این کار ادامه پیدا می کند تا بالاخره مبدأ از سوئیچ بعدی خود پیام connect دریافت کند. مبدأ نیز همانند بقیه سوئیچ ها، با ارسال پیام connect ack برقراری ارتباط خود را تکمیل می کند. بنابراین طی این روند، یک connection بین مبدأ و مقصد برقرار می شود.



همان طور که ذکر شد، بین جفت جفت سوئیچ ها، connect و connect ack ردوبدل می شود. در connection ack اعلام می گردد که با الزامات ذکر شده در پیام setup موافقت می شود. بنابراین در طی این ردو بدل ها منابع لازم نیز اختصاص می یابند. به محض اتمام ارتباط بایستی منابع آزاد گردند. همانند شکل زیر:

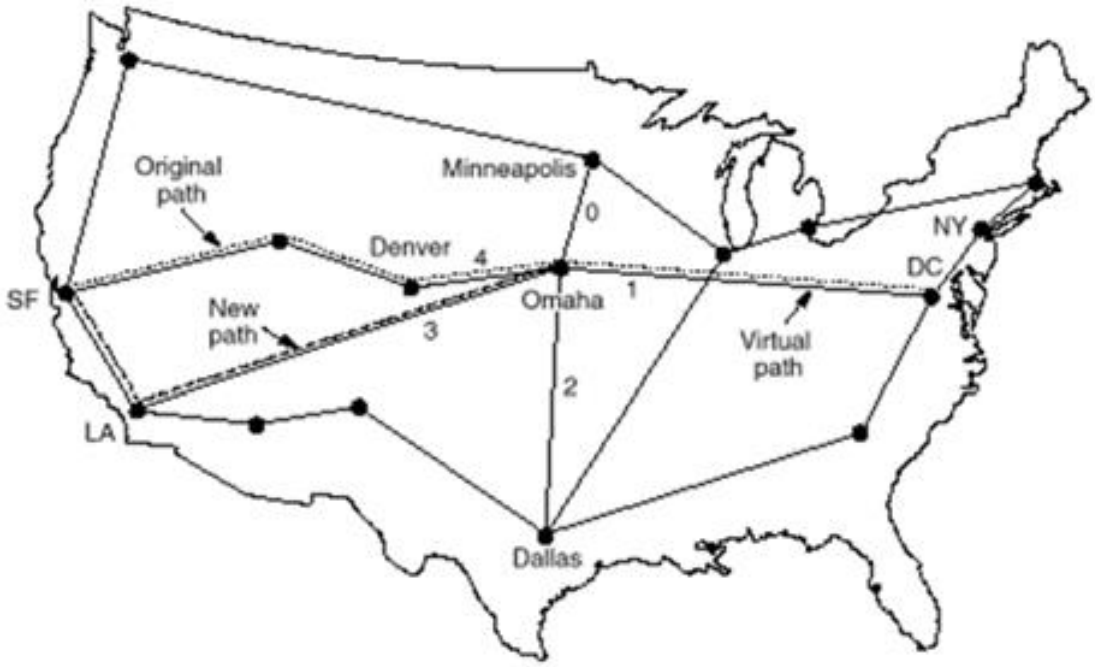


روتینگ های شبکه ATM:

در ATM برای مسیریابی، الگوریتم خاصی وجود ندارد. به عبارتی روش های مختلفی را می توان استفاده کرد. اما با هر روشی که بهترین مسیر شناخته شود بایستی یک connection برقرار گردد و منابع تخصیص داده شود و بعد از آن cell ها در این مسیر منتقل گردند.

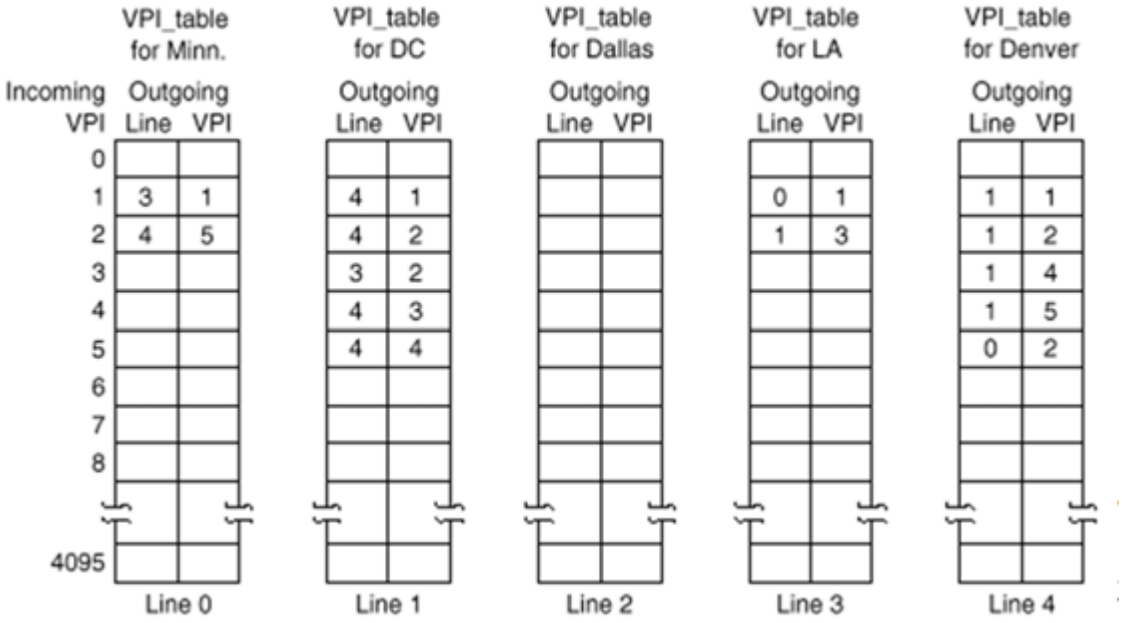
در ATM با برقراری ارتباط، کیفیت سرویس تضمین می گردد. شکل زیر را در نظر بگیرید. فرض کنید در این شکل شبکه ای به تصویر کشیده شده است که در آن از روتینگ های مبتنی بر ATM استفاده می گردد. به عبارتی، نودهای میانی سوئیچ هایی هستند که وقتی cell ها وارد آن ها می گردند از پورت مناسب خارج می شوند.

هر سوئیچ یا نود میانی ممکن است چند پورت داشته باشد. به عنوان مثال DC سه پورت دارد و Omaha پنج پورت دارد. هر پورت شماره ای دارد، به عنوان مثال در Omaha پنج پورت با شماره های 0,1,2,3,4 داریم. پورت 0 شهر Omaha به شهر minniapolise وصل می باشد، پورت 1 به DC، پورت 2 به Dallas، پورت 3 به LA و پورت 4 به Denver متصل است.



کاربرد VPI:

هر نود مثل Omaha برای هر خط خروجی یک جدول شبیه جدول زیر دارد:



در این جداول به اندازه تعداد شماره های VPI، سطر دارد. (VPI دوازده بیت داشت بنابراین 2^{12} سطر دارد.) همچنین در هر سطر جدول، دو ستون Line و VPI داریم.

هرگاه یک cell به یک سوئیچ می رسد از یکی از لاین ها (در اینجا 0/1/2/3/4) رسیده است. همچنین یک شماره VPI در آن درج شده است. با توجه به این که cell از چه پورتی وارد شده است سراغ جدول مناسب می رود و همچنین سوئیچ با توجه به شماره VPI نوشته شده در cell،

سراغ سطر مناسب از آن جدول رفته و VPI را مطابق با شماره VPI نوشته شده در آن سطر از جدول خاص بروزرسانی می کند و به پورت یا لاین نوشته شده در آنجا ارسال می نماید. برای فهم بیشتر به مثال زیر توجه کنید:

فرض کنید در شهر Omaha یک cell از line1 می رسد. پس فوراً مشخص می شود که سوئیچ بایستی جدول line1 را بررسی نماید. همچنین در هدرcell ورودی شماره VPI مشخص شده است. فرض کنید در اینجا شماره VPI=2 باشد. بنابراین نتیجه می گیرد که در جدول line1 سراغ سطر متناظر با VPI=2 برود. که در آنجا line خروجی و VPI جدید نوشته شده است. که در این مثال VPI=2 , line=4 می باشد. بنابراین سوئیچ Omaha بسته را از خط ۴ با VPI=2 ارسال خواهد کرد.

نکته: در این مثال خاص VPI ورودی و خروجی هر دو ۲ بودند اما همیشه این گونه نیست و می توانند متفاوت باشند. به عنوان مثال اگر VPI ورودی ۵ بود، همانند روش ذکر شده سراغ سطر متناظر می رفتیم و در آنجا VPI خروجی ۴ نوشته شده بود.

همانطور که دیدید کار یک سوئیچ ساده است. چون هم یک جستجوی محدود در یک جدول با سایز محدود دارد و هم یک تغییر در فیلد VPI بسته ورودی.

برقراری ارتباط و رزرو منابع، به عبارتی برقراری یک Flow را با ارائه مثالی توضیح می دهیم.

فرض کنید از مبدأ DC به مقصد SF، سوئیچ های مسیر می خواهند مسیری را ایجاد بکنند. کافیست بین جفت جفت سوئیچ های همجوار در طول مسیر، توافق صورت گیرد که در هر کدام از این لینک ها شماره VPI ای که روی cell نوشته می شود تا آن بسته در شبکه مسیردهی شود، چه باشد؟

در اینجا مثلاً Omaha به DC پاسخ می دهد که VPI را مساوی ۴ قرار دهد و Denver به Omaha می گوید که VPI را ۱۸ ثبت کند و سوئیچ بعدی به Denver می گوید که VPI را ۶۲ قرار دهد و SF هم به آن سوئیچ می گوید که VPI را ۲۷ ثبت کند. در نتیجه وقتی نود اول VPI=4 میزند و به سوئیچ بعدی می رسد، در واقع در سطر ۴ لاین ورودی به Omaha شماره VPI خروجی نوشته شده است که در اینجا ۱۸ است که قبلاً Denver به Omaha گفته بود. همین طور Denver در سطر ۱۸ لاین ورودی خودش VPI بعدی را نوشته که در اینجا ۶۲ می باشد و این روند تا مقصد همین گونه است.

نکته: از شماره VPI به عنوان ایندکسی استفاده می کنیم که وقتی یک cell به نود وارد شد بدانیم برای ادامه مسیر به کدام پورت برود و آن پورت خروجی در سطر متناظر با VPI ورودی نوشته شده است. بنابراین VPI یک شناسه محلی برای اتصال در سطح لینک می باشد.

در رد و بدل شدن پیام های setup تا connect ack بخشی از امور مربوط به این است که هر نود به نود قبلی اش بگوید چه شماره VPI ای را در cell خروجی بگذارد تا مسیر برقرار شود. در جدول زیر بعضی از مسیرهایی که از Omaha می گذرد، مشخص شده است.

Source	Incoming line	Incoming VPI	Destination	Outgoing line	Outgoing VPI	Path:
NY	1	1	SF	4	1	New
NY	1	2	Denver	4	2	New
LA	3	1	Minneapolis	0	1	New
DC	1	3	LA	3	2	New
NY	1	1	SF	4	1	Old
SF	4	3	DC	1	4	New
DC	1	5	SF	4	4	New
NY	1	2	Denver	4	2	Old
SF	4	5	Minneapolis	0	2	New
NY	1	1	SF	4	1	Old

Some routes through the Omaha switch.

تفاوت بنیادین ATM و IP در این است که هدر در ATM مرتب تغییر می کند لذا بایستی فیلد HEC مجدد بروزرسانی گردد. اگر به هر دلیلی در این شبکه یک لینک از بین برود، تنها راه این است که نودهای میانی در Control path شبکه با هم مسیر جایگزین و جدیدی را برپا

کنند. به عنوان مثال یک مسیر جایگزین برای DC- Omaha – Denver – SF می تواند مسیر DC – Omaha – LA – SF باشد. اگر سوئیچ های میانی بر سر مسیر جدید توافق کنند کافیست LA قبول کند با همان کیفیت سرویس داده ها را جابجا کند. جداول نیز باید بگونه ای تغییر کنند که مسیر ارتباطی جدید مشخص شود. به عنوان مثال اگر قبلاً cell های آمده از DC به Omaha که مربوط به این ارتباط بودند و با VPI=4 از لاین ۱ وارد Omaha می شدند برای آن در جدول متناظر و سطر متناظر لاین خروجی و VPI خروجی بگونه ای تنظیم شده بود که از مسیر بالایی روت می شدند، لذا بایستی این مقادیر برای مسیر جایگزین تنظیم گردند. به عبارتی به جای لاین خروجی۴ ، لاین خروجی را ۳ قرار می دهد. تا مسیر از پایین شکل گیرد. ممکن است در فرایند setup، سوئیچ LA به Omaha گفته باشد که VPI را با مقدار خاصی تنظیم نماید. بنابراین علاوه بر خط خروجی بایستی VPI خروجی نیز برای مسیر جدید در نودهای لازم تنظیم گردد.

۱۲ بیت VPI برای مسیردهی سطح بالا در ATM کافی می باشد. و توپولوژی های خیلی پیچیده را می توان با ۱۲ بیت سرویس داد.

کاربرد VCI:

اگر بخواهیم در یک مسیر تعیین شده از یک مبدأ به یک مقصد تعداد زیادی ارتباط وجود داشته باشد، به عنوان مثال تعداد زیادی ارتباط بین DC و SF بخواهد وجود داشته باشد می توانیم چنین تصور کنیم که مسیر تعیین شده در مرحله قبل که با خط چین بر روی نقشه به تصویر کشیده شده است همان لوله های بزرگتر می باشد که یک سرش در DC و سر دیگرش در SF می باشد. حال می توانیم بین DC وSF اتصالات زیادی داشته باشیم که با خط در مسیر این لوله مشخص می شود. اتصالات کوچک، با مدل ساده تری می تواند شکل بگیرد. بنابراین داریم:

VPI: همان تونل درست شده بین مبدأ و مقصد است که در ابتدا ساخته می شود.

VCI: اتصالات بین هر دو کامپیوتر موجود بین این مبدأ و مقصد می باشد.

روش VCI زدن بین مبدأ و مقصد بدین صورت است که هر گاه یک سوئیچ درخواستی مبنی بر برقراری ارتباط تا یک نقطه خاص با کیفیت خدمات درخواستی دریافت می کند، بایستی با توجه به تقاضای دریافتی بررسی کند که کدام یک از VPI های موجود می تواند ارتباط بین متقاضی و مقصد را برقرار سازد. پس از پیدا کردن VPI ، یک VCI داخل آن که یک سرش در مبدأ و سر دیگرش در مقصد است، می زند. این عملکرد دو لایه ای باعث کاهش سربار ناشی از فرکانس و جزئیات ایجاد ارتباط می شود.

VPI در سطح بالا زده می شود و VCI ها می توانند داخل VPI هایی که قبلاً Setup شده اند زده شود. به دلیل ۱۶ بیت بودن فیلد VCI در هدر، به اندازه 2^{16} عدد VCI می تواند داخل یک VPI شکل بگیرد.

هرگاه یک تقاضا به DC مبنی بر ارتباط با SF می رسد، DC با بررسی وجود یک VPI به مقصد SF متناظر با انتظارات متقاضی، می تواند یک اتصال از نوع VCI درست کند. بدین صورت که VPI را همان شماره VPI از قبل تعیین شده می گذارد و VCI را با نود نهایی تعیین می کند. به عنوان مثال مبدأ به مقصد می گوید هر گاه شماره VCI خاصی را دریافت کند به پورت مربوطه تحویل دهد. نودهای میانی اصلاً به VCI کاری ندارند. مسیریابی در سطح کلان فقط بر اساس VPI انجام می شود.

VPI به علتی که در سطح کلان مطرح می گردد بایستی سوئیچ های با ظرفیت بالایی داشته باشیم. اما در VCI سطح ترافیک کمتر است.

یکی از کارهای اساسی به هنگام setup یک ارتباط، رزرو منابع می باشد. تخصیص ظرفیت ها و روش های مختلف آن در ادامه بحث می شود. ولی بطور کلی می توان گفت روش های مختلفی برای ارائه QOS وجود دارد. وجود روش های مختلف و انعطاف پذیری آن ها باعث شده است اتلاف منابعی که در روش circuit switching وجود داشت در اینجا دیده نشود.

سوالی مطرح می شود و آن این است که اگر از منابع تخصیص داده شده به یک ارتباط کاملاً استفاده نشود، آیا مجاز هستیم از منابع آزاد به سایر تقاضاها استفاده کنیم؟

پاسخ این است که بستگی به قولی است که ابتدا داده ایم. ممکن است همانند circuit switching صددرصد گارانتی کنیم که منابع را همیشه به همین اتصال تخصیص دهیم. اما راهکار دیگر این است که می توانیم منابع آزاد را به سایرین نیز تخصیص دهیم و یا روش های دیگر که بحث خواهیم کرد.

ATM Service Categories:

ATM تلاش دارد ترافیک هایی که ماهیتاً متفاوت است را انتقال دهد، لذا چهار کلاس مختلف برای سرویس دهی تعریف می کند که عبارتند از:

۱. CBR
۲. VBR
   - RT-VBR
   - NRT-VBR
۳. ABR
۴. UBR

انواع سرویس ارائه شده در ATM برای انواع ترافیک هایی است که از شبکه انتظار می رود انتقال دهد. این که چه الگوریتم هایی استفاده شود از مسائل طراحی می باشد و بسته به نیاز کاربردی ما تعیین می گردد.

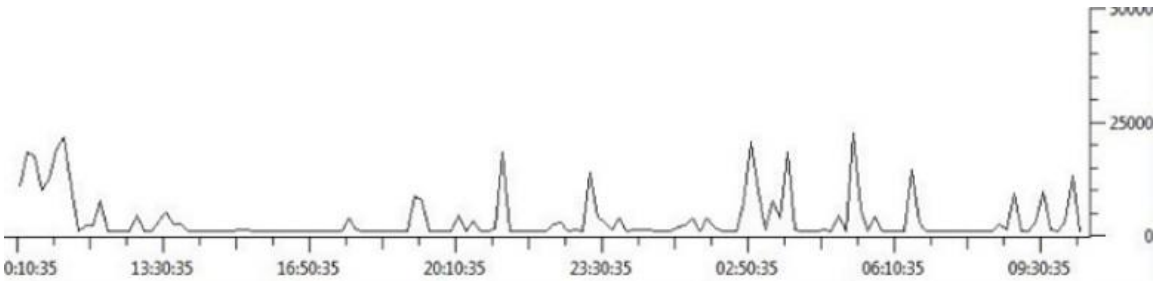
CBR: constant bit rate

گارانتی خیلی محکمی ارائه می دهد که تحت هر شرایطی منابع تخصیصی را به ارتباط ارائه دهد. و هیچ گاه از منابع آزاد برای دیگران استفاده نکند. مثلاً 2 Mb/s به معنی این است که واقعاً در هر ثانیه ۲ مگابیت داده را منتقل کند. در این نوع سرویس Delay و Jitter کنترل شده داریم. چرا که اگر Delay بخواهد کم یا زیاد شود به این معنی است که Rate در آن ثانیه تغییر کرده است. چون اگر یک بیت دیرتر برسد، سایر بیت ها هم دیرتر می رسند و در هر ثانیه دیگر ۲ مگابیت داده انتقال نشده و ممکن است به جای آن ۱۹۹۹۹۹۹ بیت برسد و این خلاف قرارداد می باشد. کاربرد های Real-time نیازمند چنین سرویسی هستند.

VBR : variable bit rate (guaranteed bandwidth)

می توان گارانتی کرد که بصورت میانگین با نرخ مشخصی سرویس گرفت. به عنوان مثال شبکه قول می دهد 1 Mb/s را انتقال دهد. اما بعضی مواقع اجازه می دهد سرعت بیشتری هم داشته باشد. به عبارتی بعضی وقت ها می توان نرخ ترافیک را بالا برد و بعضی وقت ها پایین آورد.

مثال بارز ترافیک VBR، ترافیک ویدئو است. اگر نمودار ترافیک ویدئو بر حسب زمان را نگاه کنید، می بینید که ترافیک بر حسب زمان ثابت نیست. دلیل هم این است که در تکنیک های فشرده سازی بعضی فریم ها خیلی فشرده می شوند و بعضی با حجم بالاتری ارسال می گردند.



این نوع سرویس در مورد پهنای باند بصورت میانگین گارانتی می کند. و این نوع سرویس خود دو زیرنوع real time (RT-VBR) و non real time (NRT-VBR) را شامل می شود.

در نوع real time راجع به delay و jitter گارانتی می دهد (کاربردهایی مثل ویدئو کنفرانس / VOIP)

در نوع non real time فقط Rate گارانتی می شود (کاربردهایی مثل ارتباط بین کامپیوتر و سرور)

ABR : available bit rate

شبکه یک نرخ ارسال را به متقاضی قول می دهد. اما در عین حال اجازه می دهد در صورتی که منابع آزادی داشت تا حدی سرعت ترافیک را بالا ببرد. این که چه وقت می تواند با سرعت بالاتر نیز داده ها را منتقل کند را با فیدبک به متقاضی اعلام می کند.

فایده این روش این است که هرگاه سرویس VBR در بازه ای از زمان از کمتر از نرخ ارسالش استفاده کند، می توان منابع اضافه آن را شبکه در اختیار متقاضیان دیگر قرار دهد تا از منابه بصورت کاراتر (more efficient) استفاده شود.

UBR : unspecified bit rate

هیچ گارانتی راجع به پهنای باند نمی دهد. به عبارتی شبکه هر چه توانست داده ها را انتقال می دهد. به هنگام ازدحام cell هایی که سرویس UBR می گیرند می تواند دور ریخته شود بدون اینکه حتی فیدبک لازمی داده شود. و مسئول کنترل این مشکل خود لایه کاربرد می باشد.

در جداول زیر انواع سرویس های ارائه شده ATM لیست شده اند.

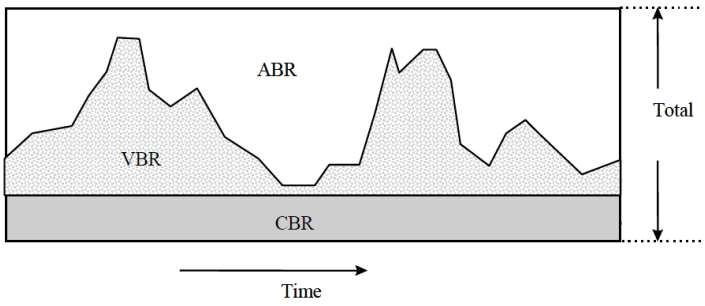
Service characteristic	CBR	RT-VBR	NRT-VBR	ABR	UBR
Bandwidth guarantee	Yes	Yes	Yes	Optional	No
Suitable for real-time traffic	Yes	Yes	No	No	No
Suitable for bursty traffic	No	No	Yes	Yes	Yes
Feedback about congestion	No	No	No	Yes	No

Class	Description	Example
CBR	Constant bit rate	T1 circuit
RT-VBR	Variable bit rate: real time	Real-time videoconferencing
NRT-VBR	Variable bit rate: non-real time	Multimedia email
ABR	Available bit rate	Browsing the Web
UBR	Unspecified bit rate	Background file transfer

The ATM service categories.

یک مثال مفهومی:

اگر نمودار ترافیک بر ثانیه که در شکل زیر به تصویر کشیده شده است را نگاه کنید درک راحت تری از انواع سرویس های ارائه شده ATM خواهید داشت.



CBR در همه ی زمان ها ثابت است و نرخ ارسال آن تغییر نمی کند.

VBR بر حسب زمان کم و زیاد می شود.

ABR از منابع بلااستفاده دیگران استفاده می کند.

UBR در صورتی از منابع شبکه استفاده می کند که ABR از ظرفیت خالی شبکه استفاده نکند.

پارامترهایی برای QOS در ATM تعریف شده است که بصورت استاندارد بیان شده اند و برای هر پارامتر بدترین مقدار نیز بایستی مشخص شود. این پارامترها در جدول زیر لیست شده اند.

Parameter	Acronym	Meaning
Peak cell rate	PCR	بیشترین نرخ ارسال **cell** ها
Sustained cell rate	SCR	میانگین بلند مدت نرخ **cell** ها
Minimum cell rate	MCR	کوچکترین نرخ قابل قبول **cell** ها
Cell delay variation tolerance	CDVT	بیشترین **jitter** قابل قبول **cell**
Cell loss ratio	CLR	تعداد **cell** های گم شده یا**cell** هایی که خیلی دیر به مقصد می رسند
Cell transfer delay	CTD	کمترین و بیشترین مدتی که طول می کشد تا cell تحویل مقصد داده شود
Cell delay variation	CDV	میزان انحراف در زمان تحویل **cell** ها
Cell error rate	CER	تعداد **cell** هایی که با خطا به مقصد می رسند
Severely-errored cell block ratio	SECBR	تعداد **cell** های درهم و اشفته
Cell misinsertion rate	CMR	تعداد **cell** هایی که به مقصد نادرست برسند

Some of the quality of service parameters.

به عنوان مثال در سرویس CBR، سه پارامتر PCR, SCR, MCR یک مقدار را دارند.

گروه بندی کلی از مؤلفه های مهم کیفیت خدمات در ATM:

پارامترهای جدول فوق را می توان در سه گروه کلی زیر طبقه بندی کرد:

Time/Delay: پارامترهایی از قبیل Delay، نوسانات Delay و حد تغییرات تأخیر cell ها

Quality: تعداد خطاها، تعداد cell هایی که drop می شود و تعداد cell هایی که به مقصد نمی رسد.

Rate: نرخ پیک، مینیموم و متوسط ارسال cell ها

این سه گروه اغلب از هم مستقل هستند. اما کابران عادی اغلب تصور اشتباهی دارند. به عنوان مثال شما دو نوع سرویس دارید. سرویس اول با نرخ 64 Kb/s ولی با کنترل خیلی دقیق روی jitter و delay. و سرویس دوم با نرخ میانگین 10Mb.s اما بدون کنترل بر delay و jitter. به نظر شما کدام یک برای سرویس دهی به VOIP مناسب تر است؟ چرا؟

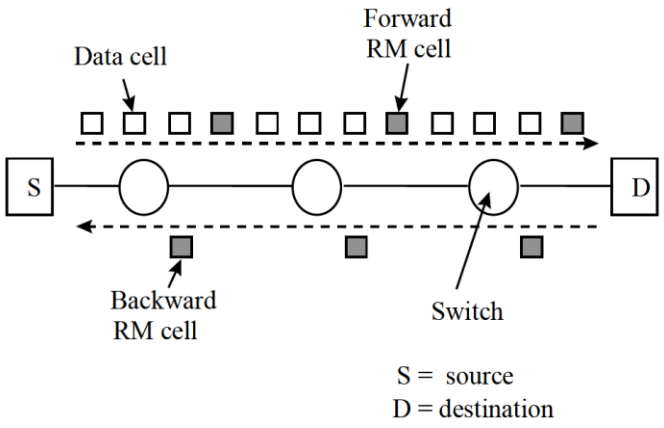
پاسخ: سرویس نوع اول

هر سرویس گیرنده با شبکه در مورد هر سه این دسته پارامترها مذاکره می کند. موضوعات تحقیقاتی زیادی راجع به تخصیص منابع در زمان پیاده سازی در سوئیچ ها صورت گرفته تا گارانتی های لازم را بتواند اعمال کنند. زمانی که با انواع ترافیک در شبکه مواجه می شویم شرایط پیچیده ای بوجود می آید. از طرفی می خواهیم به قول هایی که داده ایم عمل کنیم و از طرف دیگر نمی خواهیم اتلاف منابع داشته باشیم. آنالیز تأخیر صف ها ، رفتار تصادفی بسته ها و .... خارج از محدوده این درس است.

کنترل ازدحام در ATM:

در نوع سرویس دهی ABR شبکه به متقاضی اعلام می کند که چقدر منابع اضافه تر دارد تا متقاضی بتواند تا حد تعیین شده از منابع شبکه استفاده کند. در ATM یک سری Cell های خاصی از مبدأ به مقصد ارسال می شوند تا پاسخ آن را پس بگیرند. به این بسته ها Resource Management Cell (RM Cell) گویند.

هر سوئیچ به محض دریافت این نوع Cell و قبل از ارسال آن برای سوئیچ بعدی ظرفیت آزادی که دارد را در آن ثبت می کند. به عنوان مثال فرض کنید در شکل زیر سوئیچ اول یک مگا بیت ظرفیت خالی را در RM Cell ثبت کرده است. سوئیچ بعدی که این بسته را دریافت می کند ظرفیت خالی خود را که مثلاً ۲۰۰ کیلو بیت است در آن ثبت کرده و به سوئیچ بعدی ارسال می کند. سوئیچ سوم نیز طبق همین روال ظرفیت خالی خود را که مثلاً ۲ مگا بیت است، اعلام می نماید تا بالاخره بسته به مقصد برسد. پس اگر سوئیچ اولی 1 Mb ، دومی 200 Kb و سومی 2 Mb باشد، نتیجه می گیریم ظرفیت خالی VPI هم اکنون 200 Kb است (مینیمم مقادیر).



RM Cell ها در مسیر رفت و برگشت حرکت می کنند تا در مسیر بازگشت مقدار منابع آزاد را مجدداً تأیید کنند تا اگر لازم باشد در مسیر بازگشت بروزرسانی در آن صورت گیرد. وقتی این بسته به مبدأ می رسد، مبدأ متوجه می شود که تا سقف 200 Kb اگر احتیاج دارد می تواند نرخ ارسالش را افزایش دهد. این نوع مدیریت منابع بصورت پویاست که اگر منبعی در شبکه آزاد باشد بتوانیم آن را در اختیار ABR قرار دهیم.