فصل سوم

لایه ی انتقال

انجمن فناوری اطلاعات
INFORMATION TECHNOLOGY FORUM
www.it98.ir

کاملترین مجموعه نمونه سوالات بهمراه پاسخنامه تستی و تشریحی، جزوه، اسلاید، حل تمرین، گزارش آزمایشگاه و کارآموزی، پروژه و... در سایت:
www.it98.ir
مرجع دانشجویان رشته کامپیوتر دانشگاه پیام نور

در کنار و مابین لایه های کاربرد و شبکه، لایه ی انتقال قطعه ی مزکری معماری شبکه ی لایه بندی شده است. این لایه نقش اساسی در فراهم کردن سرویس های ارتباطی مستقیما به پردازش های لایه کاربرد که برروی میزبان های مختلف در حال اجرا هستند را برعهده دارد.

روش و تئوری تدریس این بخش که ما اتخاذ کردیم این است که ابتدا راجع به لایه ی انتقال (اهداف آن) و سپس راجع به چگونگی پیاده سازی این اهداف با پروتکل های موجود بطور معمول، بحث کنیم. تاکید خاص روی TCP و UDP یعنی پروتکل های لایه انتقال داریم. ما با بحث و گفت و گو راجع به رابطه ی بین لایه انتقال و شبکه آغاز خواهیم کرد. این مجموعه یک قسمت برای درنظر گرفتن اولین کاربرد مهم لایه ی انتقال است – توسعه دادن به سرویس دریافت لایه ی شبکه بین دو سیستم هدف و نهایی به یک سرویس دریافت بین دو پردازش های لایه ی کاربرد در حال اجرا روی سیستم های نهایی.

ما در مواجه شدن با پروتکل انتقال بدون اتصال اینترنت، یعنی UDP، را پوشش خواهیم داد.

ما سپس به اهداف و مواجه شدن با یکی از مسئله های اساسی در شبکه های کامپیوتری – چگونه دو مدل می توانند بیشتر از حد متوسط قابل اعتماد رابطه برقرار کنند که ممکن است داده ها را از دست داده و خراب کنند؟ - می پردازیم.

در بین یک مجموعه از سناریوهای در حال افزایش پیچیدگی (و واقعی!) ما یک آرایه از تکنیک هایی که پروتکل های انتقال از آن ها برای حل این مسئله استفاده می کنند خواهیم ساخت.

ما سپس نشان خواهیم داد که چگونه این اهداف در TCP ترکیب شده اند، پروتکل انتقال اتصال گرای اینترنت.

ما در مرحله ی بعد به دومین مسئله مهم پایه ای در شبکه سازی: کنترل کردن نرخ انتقال مدل لایه ی انتقال برای اجتناب یا بازگرداندن آن ها، ازدحام درون شبکه – خواهیم پرداخت.

ما دلایل ایجاد ازدحام را بررسی خواهیم کرد، همین طور تکنیک های کنترل ازدحامی که بطور معمول و مشترک استفاده می شوند. بعد از اینکه به یک درک و فهم خشک از مسائل پشت کنترل ازدحام رسیدیم، ما شیوه ی TCP را برای کنترل ازدحام خواهیم آموخت.

۱.۳ مقدمه ای بر سرویس لایه ی انتقال

در ۲ بخش قبلی، ما نقش لایه ی انتقال و سرویس هایی که فراهم می کند را احساس کردیم. بیایید سریعا مرور کنیم که تاکنون ما چه چیزهایی راجع به لایه ی انتقال آموخته ایم.

یک پروتکل لایه ی انتقال برای ارتباط منطقی بین پردازش های برنامه های در حال اجرا روی میزبان های متفاوت فراهم شده است. بوسیله ارتباط منطقی، یعنی ما از دید برنامه به آن می نگریم.

این یعنی اگر میزبان پردازش ها را اجرا می کند، جایی که مستقیما متصل باشد، در واقع، میزبان ممکن است در سمت مخالف سیاره(زمین) باشد که بوسیله ی تعدادی مسیریاب و محدوده وسیعی از نوع های مختلف(انواع مختلف) ارتباطی متصل شده باشد.

پروسس های برنامه از ارتباط منطقی فراهم شده بوسیله ی لایه ی انتقال برای ارسال پیام به یکدیگر استفاده می کنند، بدون هیچ گونه نگرانی از جزییات زیرساختار فیزیکی که برای حمل این پیام ها استفاده می شود. شکل ۱.۳ ایده ی ارتباط منطقی را دقیقا نشان می دهد.

همان طور که در شکل ۱.۳ نشان داده شده است، پروتکل های لایه انتقال در سیستم های نهایی پیاده سازی شده اند، و نه در مسیریاب های شبکه در طرف(سمت) انتقال. لایه ی انتقال پیام هایی که از پروسس های برنامه کاربردی ارسال می شوند را به بسته های لایه ی انتقال، که در اصطلاح اینترنت به segment های لایه ی انتقال معروف هستند (یا شناخته می شوند) تبدیل می کند. این کار بوسیله ی شکستن پیام های برنامه به chunk های کوچکتر و اضافه کردن یک سرآیند لایه ی انتقال به هر

wWw.it98.ir

www.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

chunk برای ایجاد کردن یک segment لایه ی انتقال انجام می شود.لایه ی انتقال سپس segment را به لایه ی شبکه سیستم نهایی ارسال می کند، جایی که segment ها درون بسته های لایه ی شبکه کپسوله می شوند (یک دیتاگرام) و به مقصد ارسال می شوند.مهم است یادآوری کنیم که مسیریاب های شبکه فقط روی فیلدهای لایه ی شبکه ی دیتاگرام عمل می کنند، یعنی آن ها فیلدهای سگمنت لایه ی انتقال که درون دیتاگرام کپسوله شده را درنظر نمی گیرند.

در سمت گیرنده، لایه ی شبکه سگمنت های لایه انتقال را از دیتاگرام جدا کرده و آن ها را به لایه ی انتقال می فرستد.سپس لایه ی انتقال سگمنت های دریافتی را پردازش کرده، داده های درون سگمنت ها را برای دریافت برنامه(کاربرد) آماده می سازد.

بیشتر از یک پروتکل لایه انتقال ممکن است برای برنامه های شبکه موجود باشد برای مثال، اینترنت ۲ پرتکل برای لایه انتقال خویش دارد: TCP و UDP .

هرکدام از این پروتکل ها مجموعه ای از سرویس های مختلف لایه انتقال را برای برنامه ی فراخواننده (صدازننده) فراهم می کنند.

۱.۱.۳ رابطه ی بین لایه های انتقال و شبکه

یادآوری می کنیم که لایه انتقال در پشته ی پروتکل ها در سمت بالای لایه ی شبکه پروسس های در حال اجرا روی میزبان های متفاوت فراهم می کند، یک پروتکل لایه ی شبکه یک ارتباط منطقی را بین میزبان ها فراهم می کند.

www.it98.ir

www.it98.ir

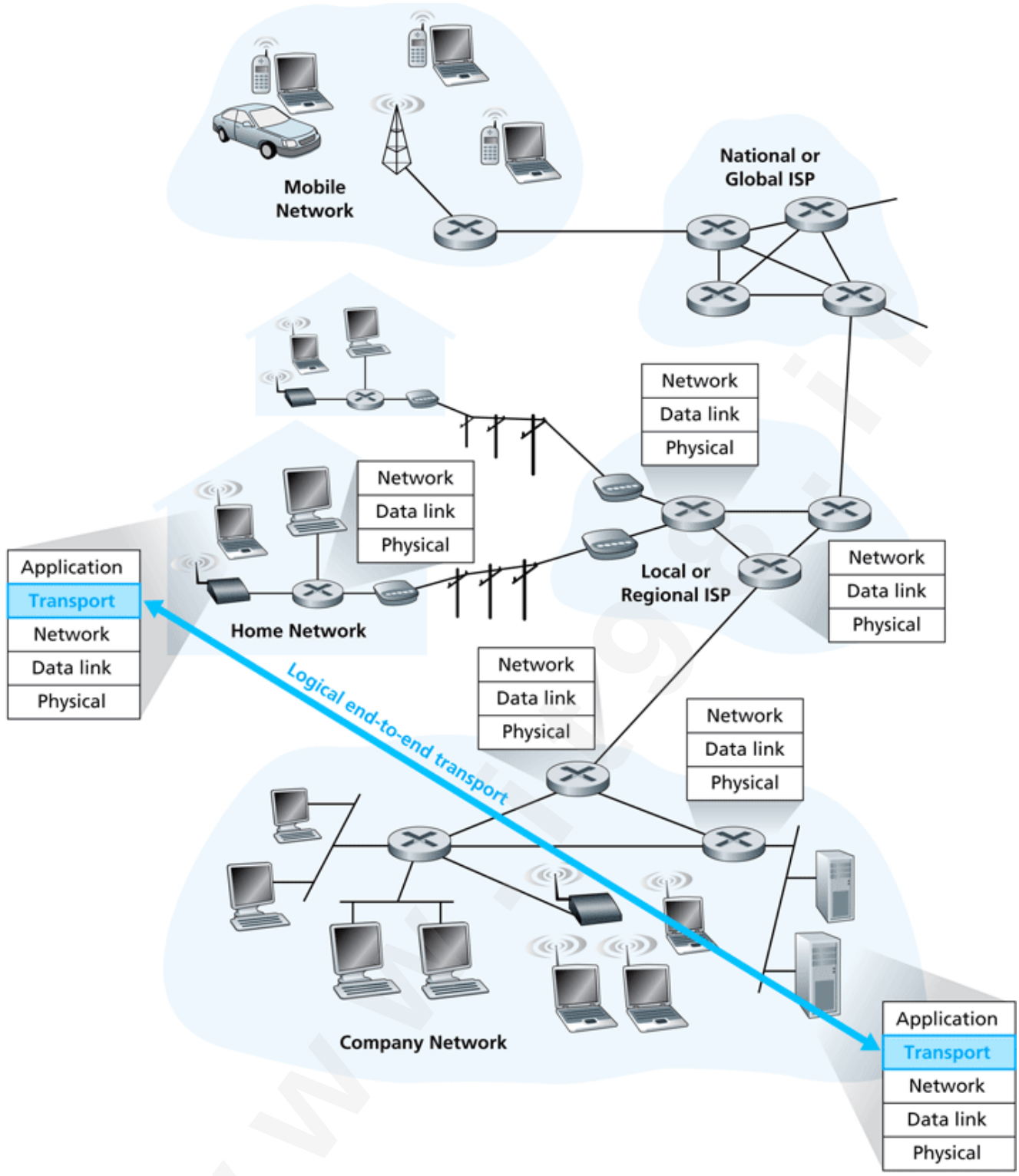


Figure 3.1 ♦ The transport layer provides logical rather than physical communication between application processes.

این تفاوت زیرکانه دقیق اما مهم است. بگذارید این تفاوت را با استفاده از قیاس خانگی مشخص کنیم.

قیاس = نشان دادن شباهت ۲ چیز در بعضی از قسمت ها

wWw.it98.ir

www.it98.ir

۲ خانه را در نظر بگیرید، یکی در قسمت شرق و یکی در قسمت غرب. همچنین هر خانه منزلی است با ۲ جین بچه. بچه های خانه ی شرقی عموزاده های بچه های خانه ی غربی هستند. بچه های هردو خانه دوست دارند برای یکدیگر نامه بنویسند، هر کودک برای هر عموزاده ی خود هر هفته نامه می نویسد، که هر نامه بوسیله سرویس پست سنتی و در بسته های جداگانه فرستاده می شود. بنابراین، هرخانه ۱۴۴ نامه را به خانه ی دیگر، هر هفته، ارسال می کند (این بچه ها پول زیادی را پس انداز خواهند کرد اگر پست الکترونیک داشتند). در خانه ی غربی و شرقی یک بچه به نام های Ann و Bill حضور دارند، که برای جمع آوری نامه ها و توزیع آن ها مسوول هستند. هر هفته Ann تمام برادران و خواهرانش را ملاقات کرده، نامه ها را جمع آوری می کند، و آن ها را به پست چی یا سرویس پستی حمل کننده ی نامه ها تحویل می دهد، کسی که روزانه خانه را ملاقات می کند. زمانی که نامه ای به خانه ی غربی می رسد نیز Ann مسولیت توزیع نامه بین برادران و خواهران خودش را دارد. Bill عهده دار مسولیت مشابه در قسمت شرقی است.

در این مثال، سرویس پست، فراهم کننده ی ارتباط منطقی بین ۲ خانه است، سرویس پست نامه های را از خانه به خانه ای دیگر جابجا می کند، نه از شخصی به شخص دیگر. درطرف دیگر، Ann و Bill فراهم کننده ی ارتباط منطقی بین عموزاده ها هستند، Ann و Bill نامه ها را از برادر یا خواهر خود گرفته، یا به آن ها می رسانند. توجه کنید که از دید عموزاده ها، Ann و Bill یک سرویس دریافت و ارسال نامه هستند; اگرچه Ann و Bill فقط بخشی(قسمتی از سیستم نهایی) از پردازش رساننده ی انتها به انتها هستند. این مثال خانگی یک قیاس زیبا را برای شرح چگونگی رابطه ی بین لایه ی انتقال و لایه ی شبکه را بکار گرفت.

نامه های درون بسته = پیام های برنامه

عموزاده ها = پروسس ها

خانه ها = میزبان ها (همچنین سیستم نهایی یا هدف نیز خوانده می شود)

Ann و Bill = پروتکل لایه انتقال

سرویس پست (شامل پستچی) = پروتکل لایه ی شبکه

با این قیاس ادامه می دهیم، توجه کنید که Ann و Bill تمام کارهایشان را درون خانه شان انجام می دهند، آن ها درگیر نیستند برای مثال: در مرتب کردن نامه ها در هیچ مرکز پست میانه ای یا در جابجایی نامه ها از یک مرکز پست به دیگری. بطور مشابه، پروتکل لایه ی انتقال در سیستم نهایی قرار گرفته است.درون یک سیستم نهایی، یک پروتکل انتقال پیام ها را از پروسس های برنامه ها به لایه ی شبکه و بلعکس، جابجا می کند، اما هیچ چیزی راجع به اینکه چگونه پیام ها در هسته ی شبکه درحال جابجا شدن هستند نمی گوید. در حقیقت، همانطور که در شکل ۱.۳ نشان داده شده است، مسیریاب های واسطه نه عمل می کنند و نه تشخیص می دهند هیچ اطلاعاتی را که لایه ی انتقال ممکن است به پیام های برنامه اضافه کند.

با خانواده ی خودمان ادامه می دهیم، حالا فرض کنید که زمانی که Ann و Bill به تعطیلات می روند، جفت دیگر عموزاده ها، که سوزان و هاروی نام دارند، ذخیره یا جایگزین آن ها محسوب شده و مسولیت جمع آوری و دریافت نامه ها را در خانه به عهده می گیرند. متاسفانه برای هردو خانواده، سوزان و هاروی جمع آوری و دریافت نامه ها را دقیقا به همان طریقی که Ann و Bill عمل می کردند انجام نمی دهند. بخاطر اینکه بچه های کوچکتری هستند، سوزان و هاروی معمولا کمتر نامه ها را برداشته و در صندوق می اندازند، و اکثرا نامه ها را گم می کنند (که برخی اوقات سگ خانه را خسته می کنند).بنابراین جفت عموزاده ها سوزان و هاروی مجموعه ی مشابهی از سرویس ها را فراهم نمی کنند (این مدل سرویس مشابه است) آنگونه که Ann و Bill فراهم می کردند.

www.it98.ir

در یک شیوه مشابه، یک شبکه کامپیوتری ممکن است چندین پروتکل لایه ی انتقال را آماده سازد (فراهم سازد)، که هر پروتکل مجموعه سرویس های متفاوتی را به برنامه ارائه می دهد.
سرویس های ممکنی که Ann و Bill می توانند فراهم کنند
برای مثال، اگر سرویس پستی یک حداکثر برای اینکه چقدر طول می کشد تا یک نامه را برساند فراهم نکند (برای مثال، ۳ روز) بنابراین هیچ راهی وجود ندارد که Ann و Bill بتوانند حداکثر تاخیرها را بین هر جفت عموزاده ای تضمین کنند. در یک طریق مشابه، سرویسی که پروتکل انتقال می تواند فراهم کند.
اگر پروتکل لایه ی شبکه نتواند تاخیر یا پهنای باند تضمین شده برای انتقال سگمنت های ارسال شده بین ۲ میزبان را فراهم کند، درنتیجه پروتکل لایه ی انتقال نخواهد توانست تاخیر یا پهنای باند تضمین شده برای پیام های فرستاده شده بین پروسس های برنامه را فراهم سازد.
برای مثال، همانطور که در این بخش مشاهده خواهید کرد، یک پروتکل انتقال می تواند یک سرویس انتقال داده ای مطمئن و قابل اتکاء را به یک برنامه پیشنهاد دهد، حتی زمانی که پروتکل شبکه قابل اعتماد نیست. این زمانی است که پروتکل شبکه از دست می رود، دچار اختلال جزییات می شود یا بسته ها را چندین بار می فرستد. یک مثال دیگر (که ما در فصل ۸ زمانی که درباره ی امنیت شبکه بحث می کنیم به آن خواهیم پرداخت) یک پروتکل انتقال می تواند از تکنیک encryption برای تضمین اینکه پیام های برنامه بوسیله بیگانه خوانده نمی شود بهره گیرد، حتی زمانی که لایه ی شبکه نمی تواند آسودگی خاطر از سگمنت های پروتکل انتقال را تضمین کند.

۲.۱.۳ مقدمه ای بر لایه ی انتقال در اینترنت

یادآوری می کنیم که اینترنت و عموما یک شبکه ی مبتنی بر TCP/IP، دو پروتکل متمایز لایه ی انتقال را برای لایه ی کاربرد فراهم می سازد. یکی از این پروتکل ها UDP (پروتکل دیتاگرام کاربر) است، که یک سرویس غیرقابل اعتماد، غیر اتصال گرا را برای برنامه ی فراخواننده فراهم می سازد. دومین پروتکل TCP است که یک سرویس قابل اعتماد اتصال گرا را برای برنامه ی فراخواننده فراهم می سازد. زمانی که یک برنامه تحت شبکه طراحی می شود توسعه دهنده ی برنامه باید یکی از این ۲ پروتکل را مشخص کند. همانطور که در قسمت ۷.۲ و ۸.۲ مشاهده کردید، توسعه دهنده ی برنامه بین UDP و TCP زمان ایجاد سوکت ها یکی را انتخاب می کرد.
برای سادگی اصطلاح شناسی، زمانی که در مجموعه ی اینترنت بودیم ما به بسته های لایه ی انتقال به عنوان یک سگمنت ارجاع می دهیم. ما یادآوری می کنیم که با این وجود، لیترال های اینترنت(برای مثال RFC ها) همچنین به بسته های لایه های انتقال برای TCP به عنوان یک سگمنت اشاره می کنند اما اغلب به بسته ها برای UDP به عنوان یک دیتاگرام ارجاع می دهند. اما لیترال های اینترنت مشابه همچنین از واژه ی دیتاگرام برای بسته های لایه ی شبکه استفاده می کنند. برای یک کتاب مقدماتی شبکه کامپیوتری مثل این کتاب، ما معتقدیم که ارجاع دادن بسته های UDP و TCP به عنوان یک سگمنت کمتر گیج کننده است. و واژه ی دیتاگرام را برای بسته های لایه ی شبکه رزرو می کنیم.
قبل از جلو بردن معرفی خلاصه ی TCP و UDP، گفتن چند جمله راجع به لایه ی شبکه ی اینترنت سودمند خواهد بود. (لایه ی شبکه به تفصیل در فصل ۴ بررسی خواهد شد). پروتکل لایه ی شبکه ی اینترنت IP نام دارد، به معنی پروتکل اینترنت. IP ارتباط منطقی بین میزبان ها را فراهم می کند. مدل سرویس IP یک سرویس به معنی بهترین تلاش است. این یعنی که IP بهترین شرایط را برای رسیدن سگمنت ها بین میزبان ها فراهم می سازد. اما تضمین نمی کند. بطور خاص، هیچ تضمینی برای رسیدن سگمنت ها وجود ندارد. به این دلیل، IP گفته می شود که یک سرویس غیر قابل اعتماد است. ما همچنین یادآور می شویم که هر میزبانی حداقل یک آدرس لایه ی شبکه دارد، که آدرس IP نامیده می شود. ما آدرس دهی IP را در فصل ۴ شرح

wWw.it98.ir
www.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

خواهیم داد، برای این فصل ما احتیاج به این داریم که بخاطر بسپاریم که هر میزبانی یک آدرس IP دارد.
بیایید حالا مدل سرویس فراهم شده بوسیله ی TCP و UDP را خلاصه کنیم. اساسی ترین مسولیت UDP و TCP توسعه بخشیدن به سرویس دریافت IP بین دو سیستم نهایی به یک سرویس دریافت بین ۲ پردازش اجرا شده روی سیستم های نهایی است.گسترش میزبان به میزبان سرویس دریافت به پردازش به پردازش عمل مالتی پلکس لایه انتقال و دی مالتی پلکس آن نامیده می شود. ما درباره ی مالتی پلکس و دی مالتی پلکس لایه ی انتقال در بخش بعد صحبت خواهیم کرد. UDP و TCP همچنین بررسی و چک یکپارچگی را بوسیله فیلد تشخیص خطا در سرآیندهای سگمنت فراهم می کنند. این دو سرویس کوچک لایه ی انتقال - دریافت داده های پردازش به پردازش و بررسی خطا – تنها ۲ سرویسی هستند که UDP فراهم می کند. بطور جزیی تر، شبیه IP، سرویس UDP نیز یک سرویس غیرقابل اعتماد است. این تضمینی نمی کند که داده های ارسال شده بوسیله یک پردازش مربوطه به پردازش مقصد خواهند رسید (یا هرگز!). UDP بطور جزیی تر در بخش ۳.۳ شرح داده خواهد شد. TCP از طرف دیگر، چندین سرویس اضافی را به برنامه ارائه می دهد. اولین و مهم ترین، فراهم کردن انتقال قابل اعتماد است. با استفاده از کنترل جریان، شماره های پشت سرهم، تصدیق نامه، و تایمر (تکنیک ها را ما به تفصیل شرخ خواهیم داد). TCP این اطمینان را می دهد که داده ها از پردازش ارسال کننده به پردازش گیرنده صحیح و به ترتیب برسند. بنابراین TCP سرویس غیرقابل اعتماد IP را بین دو سیستم نهایی به یک سرویس انتقال داده ی داده ی قابل اعتماد بین دو پردازش تبدیل می کند.
TCP همچنین کنترل ازدحام را فراهم می کند. کنترل ازدحام سرویس فراهم شده ی برای برنامه ی فراخواننده نیست بلکه سرویسی است برای تمام اینترنت، سرویسی که برای همگان خوب است.
کنترل ازدحام TCP از اتصالات TCP بین لینک ها و مسیریاب ها بین ۲ میزبان ارتباط با مقدار قابل توجهی از ترافیک جلوگیری می کند. TCP به هر Connection یک لینک با ازدحام معادل پهنای باند لینک به اشتراک گذاشته شده می دهد. این کار با منظم کردن نرخ ارسالی که سمت فرستنده ی اتصال TCP می تواند ترافیک را به شبکه ارسال کند، انجام می گیرد. ترافیک UDP در سمت دیگر، نامنظم است، برنامه ای که از انتقال UDP استفاده می کند، می تواند در هر میزانی که خوشایندش است داده ها را ارسال کند، همچنین به مدت زمانی که خوشایندش باشد.
یک پروتکلی که انتقال داده ی انتقال اعتماد و کنترل ازدحام را فراهم کند لزومش پیچیده است، ما به بخش های مختلف (چندین بخش) برای پوشش اهداف انتقال داده ی قابل اعتماد و همچنین کنترل ازدحام احتیاج خواهیم داشت، و بخش های اضافه برای پوشش دادن خود پروتکل TCP.
برای مثال ما ابتدا در یک نگاه عمومی در مورد انتقال داده ی قابل اعتماد بحث خواهیم کرد و سپس شرح خواهیم داد که چگونه TCP بطور خاص انتقال داده ی قابل اعتماد را فراهم می کند. بطور مشابه، ما ابتدا کنترل ازدحام را در یک بخش توضیح داده و سپس راجع به اینکه چگونه TCP کنترل ازدحام را انجام می دهد بحث می کنیم.
اما قبل از اینکه وارد همه ی موضوعات خوب شویم، بیایید ابتدا به مالتی پلکس و دی مالتی پلکس لایه ی انتقال نگاهی بیاندازیم.

۲.۳ Multiplexing and Demultiplexing

در این بخش ما درباره ی مالتی پلکس و دی مالتی پلکس لایه ی انتقال بحث خواهیم کرد. که مفهوم آن، توسعه دادن سرویس دریافت میزبان به میزبان فراهم شده بوسیله ی لایه ی شبکه به یک سرویس دریافت پردازش به پردازش برای برنامه های در حال اجرا روی میزبان است. برای اینکه بحث را جذاب نگه داریم، ما اساس و پایه ی سرویس لایه ی انتقال را در مجموعه اینترنت شرح خواهیم داد. ما تاکید می کنیم با این وجود، یک سرویس مالتی پلکس و دی مالتی پلکس برای تمام شبکه های

www.it98.ir

کامپیوتری مورد نیاز است.

در میزبان مقصد، لایه ی انتقال سگمنت ها را از لایه ی شبکه بدلیل اینکه در زیر آن واقع شده دریافت می کند.لایه ی انتقال مسولیت رساندن داده ها در این سگمنت به پردازش در حال اجرا برنامه مناسب در میزبان را عهده دار است. اجازه بدهید به یک مثال نگاهی بیاندازیم. فرض کنید شما روبروی کامپیوترتان نشسته اید و شما صفحات وب را بارگذاری می کنید. درحالی که یک نشست FTP و یک نشست Telnet در حال اجرا دارید. بنابراین شما ۴ پردازش برنامه ی شبکه در حال اجرا دارید، دو پردازش مربوط به Telnet، یک پردازش FTP و یک پردازش HTTP. زمانی که کامپیوتر شما داده ها را از لایه ی شبکه دریافت می کند، لازم است تا داده های رسیده را به یکی از این ۴ پروسه مسیردهی کند. حالا بگذارید ببینیم این کار چگونه انجام می شود. ابتدا از بخش ۷.۲ و ۸.۲ یادآور می شویم که یک پردازش (به عنوان یک بخش از برنامه تحت شبکه) می تواند یک یا بیشتر سوکت داشته باشد، درگاه هایی که از طریق آن ها داده ها از شبکه به پردازش فرستاده می شوند و همچنین داده هایی که از طریق آن ها از پردازش به شبکه ارسال می شوند. بنابراین همانطور که در شکل ۲.۳ نشان داده شد، لایه ی انتقال در میزبان گیرنده واقعا داده ها را مستقیما به پردازش ارسال نمی کند، اما بجای آن به یک سوکت واسطه می فرستد.بدلیل اینکه در یک زمان مشخص ممکن است بیش از یک سوکت در میزبان گیرنده وجود داشته باشد هر سوکت یک علامت مشخص دارد. قالب و فرمت علامت مشخص کننده به اینکه این سوکت یک سوکت UDP یا یک سوکت TCP است بستگی دارد، که ما بطور کوتاه شرح دادیم.

حالا بگذارید ببینیم چگونه یک میزبان گیرنده یک سگمنت لایه ی انتقال آمده را مسیردهی می کند.هر سگمنت لایه ی انتقال مجموعه ای از فیلدها را برای این هدف درون خود دارد. در گیرنده نهایی، لایه ی انتقال این فیلدها را برای مشخص کردن سوکت گیرنده و سپس مسیردهی سگمنت ها به آن سوکت، محسوب می کند. این مسولیت یعنی ارسال داده ها از سگمنت های لایه ی انتقال به سوکت صحیح و درست Demultiplex نامیده می شود. مسولیت جمع آوری chunk های داده ها در میزبان مبدا از سوکت های مختلف، کپسوله سازی chunk های داده در اصلاعات سرآیند(Header) (که بعدا در Demultiplex مورد استفاده قرار خواهند گرفت) برای ساختن سگمنت ها و همچنین ارسال آن به لایه ی شبکه Multiplexing نامیده می شود. تذکر اینکه لایه انتقال در میزبان واسط در شکل ۲.۳ باید سگمنت های رسیده از لایه ی شبکه را Demultiplex کرده تا به پردازش های p1 یا p2 مافوق خود تحویل دهد، این کار بوسیله ی مسیردهی به داده های سگمنت های رسیده به سوکت پردازش مربوطه انجام می گیرد. لایه ی انتقال در میزبان وسط باید همچنین داده هایی که درحال خارج شدن از این سوکت هستند را جمع آوری نماید، از سگمنت های لایه انتقال و این سگمنت ها را به لایه ی شبکه در پایین بفرستد.

www.it98.ir

wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

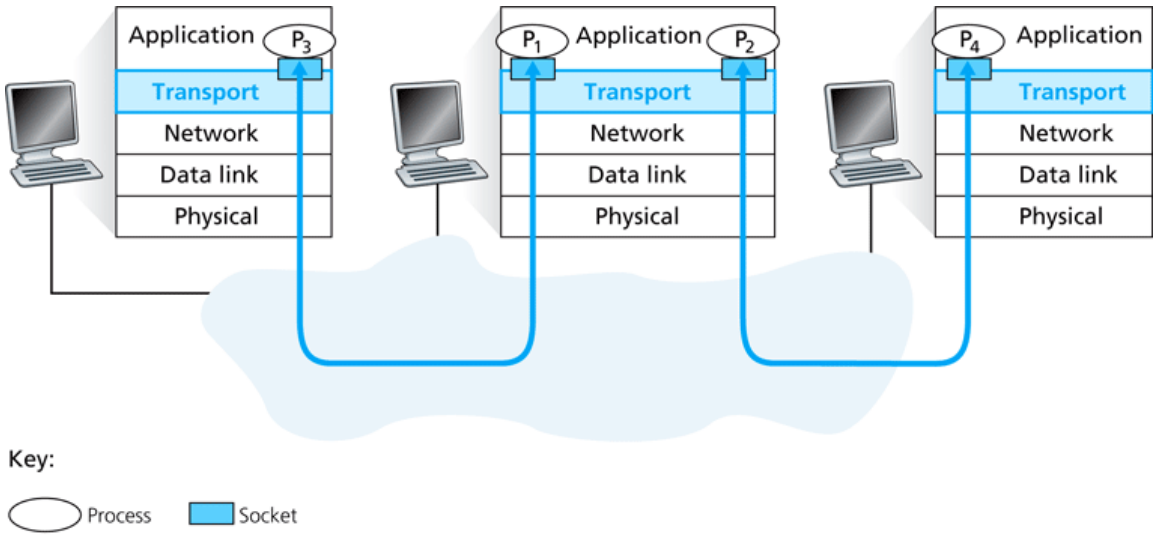


Figure 3.2 ♦ Transport-layer multiplexing and demultiplexing

اگرچه ما multiplexing و demultiplex را در مجموعه پروتکل های انتقال اینترنت معرفی کردیم، این مهم است که پی ببریم که آن ها در هرجایی یک پروتکل محسوب می شوند.در یک لایه(در لایه ی انتقال یا هرجای دیگری) که بوسیله ی چندین پروتکل در لایه بالاتر بعدی مورد استفاده قرار می گیرد.

برای نشان دادن وظیفه ی Demultiplex، دوباره قیاس خانگی را در بخش قبلی بکار می گیریم. هر بچه ای بوسیله نامش شناخته می شود. هنگامی که bill یک کیسه نامه از پستچی دریافت می کند، او یک عمل Demultiplex را انجام داده و مشاهده می کند که نامه آدرس چه کسی را می دهد و سپس با دست نامه را به برادر یا خواهرش تحویل می دهد. Ann عمل Multiplexing را انجام داده زمانی که او نامه ها را از برادر و خواهرش جمع اوری کرده و نامه های جمع آوری شده را به پستچی تحویل می دهد.

حالا که ما نقش Multiplex و Demultiplex لایه ی انتقال را متوجه شدیم و درک کردیم بیایید ببینیم این کار چگونه در میزبان انجام می شود. از توضیحات بالا، ما دریافتیم که Multiplex لایه ی انتقال به :

۱ : سوکتی که نشان مشخص دارد

۲ : هر سگمنتی فیلد خاصی را برای مشخص کردن سوکتی که سگمنت باید به آن برسد،

احتیاج دارد. این فیلدهای خاص، که در شکل ۳.۳ نشان داده شده است، فیلد شماره پورت مبدا و فیلد شماره پورت مقصد هستند.(segment های TCP و UDP همچنین در زیربخش های این فصل شرح داده خواهد شد).هر شماره پورت یک عدد ۱۶ بیتی است که از ۰ تا ۶۵۵۳۵ هستند. پورت هایی که در محدوده ی ۰ تا ۱۰۲۳ شماره پورت های شناخته شده نامیده می شوند.

www.it98.ir

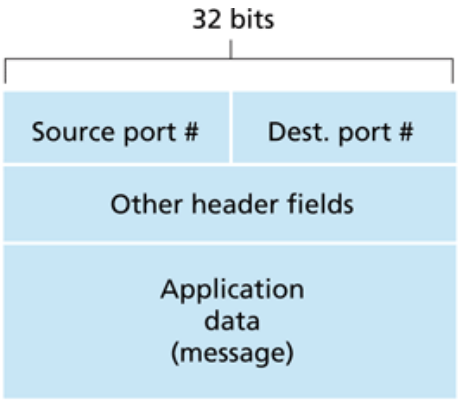


Figure 3.3 ♦ Source and destination port-number fields in a transport-layer segment

یعنی اینکه برای استفاده ی پروتکل های برنامه های شناخته شده رزرو شده اند. از قبیل HTTP (که از پورت شماره ۸۰ استفاده می کند) و FTP (که از پورت شماره ۲۱ استفاده می کند). لیست شماره پورت های شناخته شده در RFC 1700 داده شده است و در iana.org [3232] I بروزرسانی می شود. زمانی که ما یک برنامه جدید را توسعه می دهیم (از قبیل یکی از برنامه های توسعه داده شده در بخش ۷.۲ و ۸.۲) ما باید یک شماره پورت را به برنامه اختصاص دهیم.
این موضوع اکنون باید واضح باشد که لایه ی انتقال چگونه می تواند سرویس Demultiplex را پیاده سازی کند. هر سوکتی در میزبان می تواند یک شماره پورت داشته باشد، و زمانی که سگمنت به میزبان برسد، لایه ی انتقال مسیر شماره پورت مقصد را درون مقصد محسوب کرده و سگمنت را به سوکت مربوطه مسیردهی می کند. داده های سگمنت سپس از طریق سوکت به پردازش ضمیمه شده فرستاده می شوند. همانطور که خواهیم دید، این پایه ی چگونگی کار UDP نیز هست، ما همچنین خواهیم دید که multiplex و Demultiplex در TCP هنوز دقیق تر است.

Multiplexing و Demultiplex بدون اتصال

از بخش ۸.۲ یادآور می شویم که یک برنامه ی جاوا در حال اجرا روی کامپیوتر میزبان می تواند یک سوکت UDP با خط

```
DatagramSocket mySocket = new DatagramSocket();
```

ایجاد کند. زمانی که یک سوکت UDP در این قسمت ایجاد شد، لایه ی انتقال بصورت اتوماتیک یک شماره پورت به سوکت اختصاص می دهد. لایه ی انتقال شماره ی پورت را در محدوده ۱۰۲۴ تا ۶۵۵۳۵ اختصاص می دهد که هم اکنون بوسیله ی هیچ پورت UDP دیگری مورد استفاده قرار نگرفته باشد (در میزبان). بر خلاف این، یک برنامه جاوا می تواند یک سوکت را با این خط :

```
DatagramSocket mySocket = new DatagramSocket(19157);
```

ایجاد کند، در این مورد، برنامه یک پورت خاص را به سوکت UDP اختصاص می دهد، که ۱۹۱۵۷ نام دارد.اگر توسعه دهنده ی برنامه کدی را بنویسد که در سمت سرور پروتکل های شناخته شده را پیاده سازی کند، توسعه دهنده باید شماره پورت شناخته شده مربوطه را مشخص می کند. به سادگی، برنامه سمت کلاینت نیز اجازه می دهد که لایه ی انتقال بصورت اتوماتیک شماره پورتی که اختصاص دهد، در حالی که قسمت سرور برنامه یک شماره پورت خاص را اختصاص داده است.

www.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

با شماره پورت هایی که به سوکت UDP اختصاص داده شده است، ما اکنون می توانیم پایداری multiplexing و Demultiplex پروتکل UDP را شرح دهیم.فرض کنید یک پردازش در میزبان A، با پورت UDP 19157 قصد دارد یک chunk از داده های برنامه را به پردازشی با پورت ۴۶۴۲۸UDP در میزبان B بفرستد.لایه ی انتقال در میزبان A یک سگمنت لایه ی انتقال که شامل داده های برنامه، شماره پورت مبدا، شماره پورت مقصد و ۲ مقدار دیگر (که بعدا شرح داده خواهد شد اما برای بحث فعلی مهم نیستند) ایجاد می کند. سپس لایه ی انتقال سگمنت حاصل را به لایه ی شبکه می فرستد. لایه ی شبکه سگمنت را در یک دیتاگرام IP کپسوله سازی می سازد و یک attempt برای رسیدن سگمنت به میزبان گیرنده را می سازد.

اگر سگمنت به میزبان گیرنده B برسد، لایه ی انتقال در میزبان گیرنده، پورت مقصد را در سگمنت یافته، و سگمنت را به سوکت مشخص شده بوسیله ی پورت شماره ۴۶۴۲۸ می فرستد. نکته اینکه در میزبان B چندین پردازش ممکن است در حال اجرا باشند، که هرکدام بوسیله سوکت UDP خود و شماره پورت مقصد نسبت داده شده شناخته می شوند.از آنجایی که سگمنت UDP از شبکه رسیده است، میزبان B هر سگمنت را به سوکت مناسب بوسیله ی شماره پورت مقصد مسیردهی (Demultiplex) می کند.

این مهم است که تذکر دهیم که سوکت UDPz کاملا بوسیله دو تاپل شامل آدرس IP، شماره پورت مقصد مشخص شده است. به عنوان یک توالی، اگر ۲ سگمنت UDP آدرس های IP مقصد مشابهی داشته باشد و شماره پورت مقصد مشابهی، بنابراین هردو سگمنت به پردازش مقصد مشابه با سوکت مقصد مشابه آدرس دهی می شوند. شما ممکن است اکنون متعجب شده باشید، هدف شماره پورت مبدا چیست؟. همانطور که در شکل ۴.۳ نشان داده شده است در سگمنت A به B شماره پورت مبدا به عنوان آدرس بازگشت ذخیره شده است. هنگامی که B می خواهد سگمنتی را به A بازگرداند پورت مقصد در سگمنت B به A مقدار خودش را از مقدار پورت مبدا یعنی از سگمنت B به A می گیرد. (آدرس بازگشت کامل آدرس IP میزبان A و شماره ی پورت مبدا است). به عنوان یک مثال، برنامه سرور UDP را که در بخش ۸.۲ در UDPServer.java مطالعه کردید را بخاطر آورید.

سرور از یک متد برای بازگشایی شماره پورت مبدا از سگمنت رسیده از کلاینت استفاده می کند. سپس این سرور یک سگمنت جدید را به کلاینت ارسال می کند، همراه با شماره پورت مبدا که بعنوان شماره پورت مقصد در این سگمنت جدید بکار گرفته شده است.

Multiplex و Demultiplex اتصال گرا

برای اینکه Demultiplex در TCP را متوجه شویم، ما باید یک نگاه متمرکز به سوکت TCP و اتصال TCP داشته باشیم.یک تفاوت دقیق بین سوکت TCP و سوکت UDP این است که سوکت TCP بوسیله ۴ تاپل مشخص می شود : (آدرس IP مبدا، شماره پورت مبدا، آدرس IP مقصد، شماره پورت مقصد).

www.it98.ir

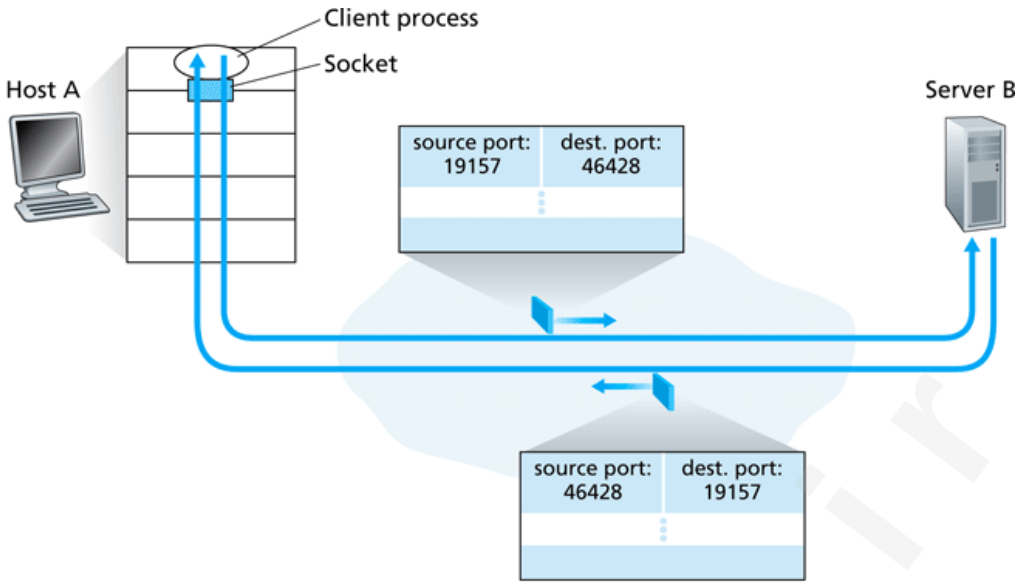


Figure 3.4 ♦ The inversion of source and destination port numbers

بنابراین، زمانی که یک سوکت TCP از شبکه به یک میزبان می رسد، میزبان از هر ۴ مقدار برای مسیردهی یا همان Demultiplex سگمنت به سوکت مناسب استفاده می کند. بطور خاص، و در تضاد با UDP، دو سگمنت TCP رسیده با آدرس های IP متفاوت یا شماره پورت های متفاوت (همراه با یک استثناء از سگمنت TCP حامل درخواست برقراری اتصال اصلی) به ۲ سوکت متفاوت مسیردهی خواهد شد، برای بدست آوردن اطلاعات بیشتر، بیایید دوباره برنامه کلاینت سرور TCP مثال بخش ۷.۲ را بررسی کنیم.

برنامه سرور TCP یک سوکت خوش آمد گویی دارد که برای درخواست برقراری اتصال از کلاینت TCP روی پورت شماره ۶۷۸۹۰ منتظر است (شکل ۳۱.۲ را ببینید)

TCP مشتری سگمنت برقراری اتصال را با خط

Socket clientSocket = new Socket(“ ServerHostName”,6789) ;

تولید می کند.

یک درخواست برقراری اتصال بیشتر از یک سگمنت TCP با شماره پورت مقصد ۶۷۸۹ و مجموعه بیت برقراری اتصال خاص در سرآیند TCP (در بخش ۵.۳ شرح داده شده) نیست.سگمنت همچنین شامل شماره پورت مبدا است که بوسیله ی کلاینت انتخاب شده است. خط فوق همچنین یک سوکت TCP برای پردازش کلاینت می سازد، که از طریق آن داده ها می توانند وارد شوند یا از پردازش کلاینت خارج شوند.

هنگامی که سیستم عامل میزبان کامپیوتر در حال اجرای پردازش سرور سگمنت درخواست اتصال را با پورت ۶۷۸۹ دریافت کرد، این پردازش سرور را که منتظر پذیرش اتصال روی پورت شماره ی ۶۷۸۹ است را پیدا می کند(مشخص می کند). پردازش سرور سپس یک سوکت جدید می سازد :

Socket connection = welcomeSocket.accept();

همچنین، لایه انتقال در سرور به ۴ مقداری که در سگمنت درخواست اتصال در ادامه می آید توجه می کند،

۱ : شماره پورت مبدا در سگمنت

wWw.it98.ir

www.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

۲ : آدرس IP میزبان مبدا
۳ : شماره پورت مقصد در سگمنت
۴ : آدرس IP خودش

سوکت اتصال تازه ایجاد شده بوسیله این ۴ مقدار مشخص می شود. همه ی سگمنت هایی که در زیرتوالی می رسند که پورت مبدا، آدرس IP مبدا، پورت مقصد، و آدرس IP مقصدشان با این ۴ مقدار تطبیق داشته باشد. بوسیله اتصال TCP درجا به این سوکت Demultiplex می شود، سپس کلاینت و سرور می توانند به یکدیگر داده ارسال کنند.
سرور میزبان ممکن است همزمان از چندین سوکت TCP پشتیبانی کند، که هر سوکت نیز خود ضمیمه ی یک پردازش است، و هر سوکت بوسیله ی ۴ تاپل خودش شناسایی می شود.زمانی که یک سگمنت TCP به میزبان می رسد، همه ی ۴ فیلد(آدرس IP مبدا،پورت مبدا، آدرس IP مقصد، پورت مقصد) برای مسیردهی سگمنت به سوکت مناسب مورد استفاده قرار می گیرد.

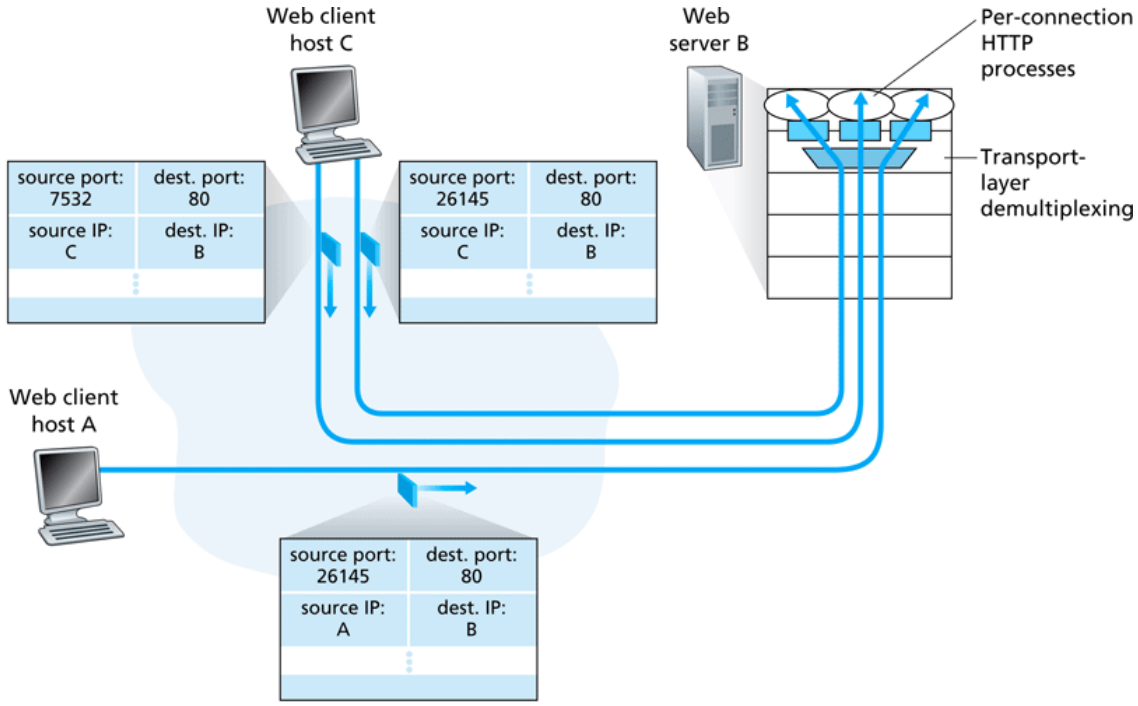


Figure 3.5 ♦ Two clients, using the same destination port number (80) to communicate with the same Web server application

موقعیت در شکل ۵.۳ نشان داده شده است، که در آن میزبان، بوسیله ی ۲ بخش HTTP به سرور B متصل شده است. و میزبان A یک نشست HTTP با B دارد. میزبان A و C و سرور B هرکدام آدرس IP خودشان را دارند، میزبان C دو شماره پورت متفاوت (۲۶۱۴۵ و ۷۵۳۲) به ۲ اتصال HTTP اش نسبت داده است. بدلیل اینکه میزبان A دو شماره پورت را مستقل از C انتخاب کرده است این ممکن است یک پورت مبدا ۲۶۱۴۵ را به اتصال HTTP اش نسبت داده باشد. اما این مشکلی ندارد. سرور B هنوز قادر خواهد بود تا دو اتصالی که شماره پورت مشابهی دارند را به درستی Demultiplex کند، تا زمانی که اتصالات آدرس IP متفاوتی دارند.

www.it98.ir

وب سرورها و TCP

قبل از بستن این بحث، سودمند خواهد بود که کمی راجع به وب سرورها و اینکه آن ها چگونه از شماره پورت ها استفاده می کنند مطالبی را بگوییم. یک میزبان را فرض کنید که یک وب سرور را در حال اجرا دارد، از قبیل یک وب سرور آپاچی، روی پورت ۸۰. زمانی که کلاینت(برای مثال، مرورگرها) سگمنت ها را به سرور ارسال می کنند، همه ی سگمنت ها پورت مقصد ۸۰ دارند. بطور خاص، هر دو سگمنت برقراری ارتباط و سگمنت حاصل پیام درخواست HTTP، پورت مقصد ۸۰ خواهند داشت. همانطور که شرح دادیم، سرور سگمنت ها را از کلاینت های متفاوت با استفاده از آدرس IP مبدا و شماره پورت مبدا توزیع می کند.

شکل ۵.۳ یک وب سرور را نشان می دهد که برای هر اتصال یک پردازش جدید نگه داری می کند. همانطور که در شکل ۵.۳ نشان داده شده است، هرکدام از این پردازش ها سوکت اتصال خودش را دارد که از طریق ان درخواست HTTP می رسند پاسخ های HTTP ارسال می شوند. ما یادآور می شویم، با این وجود، همیشه بین پردازش ها و سوکت های اتصال رابطه ی ۱ به یک وجود ندارد. در واقع، امروزه، وب سرورها با قدرت عملکرد بالا اغلب از یک پردازش استفاده می کنند و یک مکانیزم Thread می سازند و یک سوکت اتصال جدید برای هر اتصال مشتری جدید.(یک Thread می تواند به عنوان یک زیرپردازش سبک وزن دیده شود).

اگر شما اولین برنامه نویسی را در فصل ۲ انجام دادید، شما یک وب سرور ساختید که فقط این کار را انجام می داد، برای هر سرور در هر زمانی ممکن است آنجا سوکت های اتصال زیادی با مشخص کننده های متفاوت که به پردازش مشابهی ضمیمه شده اند وجود داشته باشد.

اگر مشتری و خدمتگذار (کلاینت و سرور) از پایداری HTTP استفاده کنند، سپس از طریق طول پایداری اتصال، مشتری و سرور پیام های HTTP را بوسیله سوکت سرور مشابه تبادل می کنند.با این وجود، اگر مشتری و سرور از HTTP ناپایدار استفاده کنند، آنگاه اتصال TCP جدید برای هر درخواست / دریافت ساخته و بسته می شود. و از این به بعد یک سوکت جدید ساخته و بعدا برای هر درخواست/دریافت بسته می شود. این معمولا یعنی ساختن و بستن سوکت می تواند تاثیر چندانی بر عملکرد وب سرور شلوغ بگذارد. اگرچه تعدادی حقه های سیستم عامل می تواند برای حل این مسئله استفاده شود.

حالا که درباره ی Multiplex و Demultiplex لایه ی انتقال بحث شد، بگذارید بیشتر درباره ی پروتکل های اتصال اینترنت صحبت کنیم.

۳.۳ انتقال بدون اتصال UDP

در این بخش، ما یک نگاه دقیق به UDP خواهیم داشت. این چگونه کار می کند، و این چه هست. ما شما را تشویق می کنیم تا به بخش ۱.۲ بازگشته، که شامل یک نگاه کلی راجع به مدل سرویس دهی UDP است و به بخش ۸.۲ که در مورد برنامه نویسی سوکت با استفاده از UDP است.

برای آغاز بحثمان راجع به UDP، فرض کنید شما علاقمند به طراحی این پروتکل انتقال هستید.شما چگونه ممکن است این کار را انجام دهید؟

عدم برقراری ارتباط. TCP قبل از شروع انتقال داده ها، از یک دست دادن یا تبادل نظر سه حالته (three way hand shake) استفاده می کند. UDP بدون هیچ مقدمه رسمی ارسال می شود. بنابراین UDP هیچ تاخیری در برقراری ارتباط ندارد. شاید این دلیل اصلی ترجیح UDP به TCP از نظر دیدگاه DNS باشد چون سرعت آن در صورت استفاده از TCP پایین تر می آید. HTTP بیشتر از TCP استفاده می کند تا UDP، چون برای صفحات وب، قابلیت اطمینان بسیار حیاتی

www.it98.ir

wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

است. و همانطور که در بخش ۲.۲ بطور اختصار بیان کردیم تاخیر در برقراری ارتباط TCP در HTTP منجر به تاخیر در بارگزاری (دانلود) اسناد وب می شود.

عدم کیفیت(حالت) ارتباط. TCP، کیفیت ارتباط را در سیستم های نهایی حفظ می کند. این کیفیت ارتباط شامل دریافت و ارسال بافرها، پارامترهای کنترل ازدحام، و پارامترهای عددی تایید و توالی است. در بخش ۵.۳ خواهیم دید که این وضعیت اطلاعات، برای پیاده سازی سرویس انتقال مطمئن اطلاعات TCP و فراهم آوردن کنترل ازدحام لازم است. به عبارت دیگر،UDP کیفیت ارتباط را حفظ نمی کند و هیچ یک از این پارامترها را پیگیری نمی نماید. به همین دلیل، وقتی برنامه ای بجای TCP از UDP استفاده می کند، سرور اختصاص داده شده به یک برنامه خاص می تواند مشتریان فعال بیشتری را پشتیبانی کند.

سربار هدر بسته کوچک. بخش TCP دارای ۲۰ بیت سربار هدر در هر بخش است در صورتی که UDP تنها ۸ بیت سربار دارد. شکل ۶.۳، برنامه های اینترنتی و پروتکل های انتقال که مورد استفاده قرار می گیرند را فهرست می کند. همانطور که انتظار داریم،ایمیل،دسترسی از راه دور به ترمینال،وب و انتقال فایل از TCP استفاده می کنند. همه این برنامه ها نیاز به سرویس انتقال داده های TCP دارند. با این حال برنامه های مهم متعددی بجای TCP از UDP بهره می برند. UDP برای بروزرسانی جدول مسیریابی RIP استفاده می شود. چون بروزرسانی RIP بصورت دوره ای ارسال می شود(معمولا هر پنج دقیقه)، آپدیت هایی از دست رفته با آپدیت های جدیدتر جایگزین می شوند، بنابراین آپدیت از دست رفته و تاریخ مصرف گذشته را بی ثمر می سازد. همچنین UDP برای انتقال اطلاعات مدیریت شبکه کاربرد دارد. در این مورد UDP بر TCP ارجح است چون برنامه های مدیریت شبکه اغلب وقتی شبکه در حالت تحت فشار است، اجرا می شوند – دقیقا وقتی دسترسی به انتقال انبوه اطلاعات کنترل شده و مطمئن سخت است. همانطور که قبلا ذکر کردیم DNS بیشتر از UDP استفاده می کند در نتیجه از تاخیرهای برقراری ارتباط TCP جلوگیری می شود.

همانطور که در شکل ۶.۳ نشان داده شده، امروزه هم از TCP و هم UDP در برنامه های چند رسانه ای مثل تلفن اینترنتی، ویدیو کنفرانس همزمان، جریان صوت و تصویر به صورت Stream استفاده می شود. ما نگاه کوتاهی به این برنامه ها در فصل ۷ خواهیم انداخت. در اینجا فقط این نکته را ذکر می کنیم که همه این برنامه ها می توانند مقدار کمی افت و از دست رفتن بسته اطلاعاتی را تحمل کنند، بنابراین انتقال مطمئن اطلاعات برای موفقیت برنامه ها کاملا حیاتی نیست. علاوه بر این، برنامه های فوری مثل تلفن اینترنتی و ویدیو کنفرانس، در برابر کنترل ازدحام TCP ضعیف عمل می کنند. به این دلایل، توسعه دهندگان برنامه های چند رسانه ای ممکن است برنامه های خود را بجای TCP با UDP اجرا کنند. به هرحال TCP بطور فزاینده ای در انتقال جریان رسانه ای مورد استفاده قرار می گیرد. بطور مثال تقریبا ۷۵ درصد از جریان های زنده و درخواستی از TCP استفاده می کند. وقتی سرعت افت بسته اطلاعاتی پایین است و بعضی سازمان ها مخابره UDP را بدلایل امنیتی مسدود می کنند، TCP تبدیل به یک پروتکل جذاب برای انتقال جریان رسانه ای می شود.

www.it98.ir

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP	TCP
Remote terminal access	Telnet	TCP
Web	HTTP	TCP
File transfer	FTP	TCP
Remote file server	NFS	Typically UDP
Streaming multimedia	typically proprietary	UDP or TCP
Internet telephony	typically proprietary	UDP or TCP
Network management	SNMP	Typically UDP
Routing protocol	RIP	Typically UDP
Name translation	DNS	Typically UDP

Figure 3.6 ♦ Popular Internet applications and their underlying transport protocols

اگرچه امروزه این امر عادی بنظر می رسد، اما اجرای UDP برای برنامه های چند رسانه ای جنجال برانگیز است. همانطور که در بالا بیان کردیم UDP کنترل ازدحام ندارد. اما این کنترل برای جلوگیری از ورود شبکه به حالت متراکم که در آن کار مفید خیلی کمی انجام می شود، ضرورت دارد. اگرهرکسی بخواهد ویدیو با سرعت بالای بیت را بدون استفاده از کنترل ازدحام مسیردهی کند، سرریز بسته اطلاعاتی زیادی در مسیریاب ها وجود خواهد داشت که در نتیجه بسته های اطلاعاتی UDP با موفقیت مسیر منبع به مقصد را می پیمایند. علاوه براین، افت بالای سرعت که بوسیله فرستنده های غیر کنترلی UDP القا می شود،باعث می گردد فرستنده های TCP سرعت خود را کاهش دهند. از این رو فقدان کنترل ازدحام در UDP منجر به افت بالای سرعت بین فرستنده UDP و گیرنده و ازدحام خارج از دوره های TCP می شود که یک مشکل بالقوه جدی است. بیشتر محققان مکانیزم های جدیدی را پیشنهاد داده اند که تمام منابع از جلمه منابع UDP را وا می دارد کنترل ازدحام سازگاری را اجرا کنند.

[Mahdavi 1997;Floyd 2000;Kohler 2006;RFC 4340]

قبل از بحث درباره ساختار سگمنت های UDP، این نکته را ذکر می کنیم که یک برنامه می تواند در هنگام استفاده از UDP، اطلاعات مطمئنی را انتقال دهد. این کار را در صورتی می توان انجام داد که قابلیت اطمینان در درون خود برنامه ساخته شده باشد(بطور مثال با اضافه کردن مکانیزم های تایید و ارسال مجدد نمی توان برای یک مدت زمان طولانی، توسعه دهنده برنامه (نرم افزار) را مشغول اشکال زدایی داشت. با این حال، ساختن قابلیت اطمینان برای برنامه به آن اجازه می دهد تا ((کیک خود را داشته باشد و آن را بخورد)) . این بدان معناست که فرآیندهای برنامه بدون محدودیت سرعت انتقال که توسط مکانیزم کنترل ازدحام TCP اعمال می شود، می توانند با اطمینان ارتباط برقرار کنند.

wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

۳-۳-۱ : ساختار سگمنت UDP

ساختار سگمنت UDP که در شکل ۷.۳ نشان داده شده در RFC 768 تعریف شده است. اطلاعات برنامه ها، فیلد اطلاعاتی بخش UDP را اشغال می کنند. مثلا برای DNS، فیلد اطلاعاتی یا شامل یک پیام سوال است یا پیام پاسخ. برای مسیردهی یک برنامه صوتی، نمونه های صوتی فیلد اطلاعاتی را پر می کنند. هدر UDP دارای ۴ فیلد است که هرکدام از ۲ بایت تشکیل می شود. همانطور که در بخش پیشین بحث شد، اعداد پورت به میزبان مقصد اجازه می دهند اطلاعات برنامه را به سمت یک روند صحیح هدایت دهند که روی سگمنت سیستم نهایی مقصد اجرا می شود. هاست دریافت از یک مجموع مقابله ای (checksum) استفاده می کند تا چک کند آیا اشتباهی در بخش وجود دارد یا نه. در حقیقت checksum از طریق چند فیلد در هدر IP و بخش UDP محاسبه می شود. اما ما از این جزییات چشم پوشی می کنیم تا از میان درختان، جنگل را ببینیم. ما در زیر به محاسبه checksum می پردازیم. اصول اولیه ردیابی خطاها در بخش ۲.۵ توضیح داده شده است. طول فیلد، طول بخش UDP را تعیین می کند که شامل همه طول هدر به بایت است.

۳-۳-۲ : UDP Checksum

برای ردیابی خطاها UDP checksum را استفاده می کنیم تا تعیین کنیم آیا بیت ها در بین بخش UDP هنگام حرکت از مبدا به مقصد تغییر می کنند یا خیر. (مثلا بوسیله نویز در لینک ها یا درهنگام ذخیره در یک مسیریاب). UDP در بخش فرستنده، تکمیل جمع ۱ های همه کلمات ۱۶ بیتی را در بخش انجام می دهد که اگر در طول جمع زدن با سرریزی مواجه شد، آنرا پنهان می کند(می پوشاند). این نتیجه در فیلد checksum بخش UDP قرار می گیرد، در اینجا یک نمونه ساده از محاسبه checksum را می بینیم.

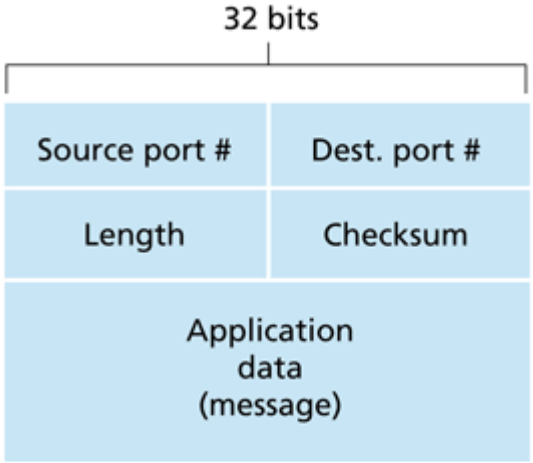


Figure 3.7 ♦ UDP segment structure

شما می توانید جزییات اجرای موثر محاسبات را در RFC 1071 و اجرای اطلاعات واقعی را در [Stone 1998 Stone 2000] ببینید. بعنوان نمونه فرض کنید ما سه کلمه ۱۶ بیتی زیر را داریم :

0110011001100000
0101010101010101
1000111100001100

www.it98.ir

جمع دو کلمه ۱۶ بیتی اولی می شود :
0110011001100000
0101010101010101
1011101110110101
با اضافه کردن کلمه سوم جمع بالا می شود :
1011101110110101
1000111100001100
0100101011000010
که آخرین جمع سرریز (overflow) دارد که پوشش داده شده است. تکمیل ۱ها را از طریق تبدیل ۰ ها به ۱ ها و ۱ ها به ۰ ها می توان انجام داد. بنابراین تکمیل ۱ های جمع ۰۱۰۰۱۰۱۰۱۱۰۰۰۰۱۰ می شود ۱۰۱۱۰۱۰۱۰۰۱۱۱۱۰۱ checksum می شود. در گیرنده ۱۱۱۱۱۱۱۱۱۱۱۱۱۱۱۱ خواهد بود. اگر یکی از بیت ها ۰ باشد متوجه می شویم که خطایی در بسته وجود دارد.

ممکن است تعجب کنید که چرا UDP مجموع checksum را در اولین مکان فراهم می کند. با اینکه پروتکل های لایه ارتباطی متعددی (از جمله پروتکل معروف Ethernet) هم چک کردن خطاها را انجام می دهند. دلیل این امر آن است که ضمانتی وجود ندارد که همه لینک ها بین منبع و مقصد خطاها را چک کنند، یعنی ممکن است یکی از لینک ها از پروتکل لایه ارتباطی استفاده کند که خطاها را چک نمی کند. علاوه بر این، اگر هم سگمنت ها بدرستی از طریق یک لینک منتقل شوند، وقتی که یک بخش در حافظه یک مسیریاب ذخیره می شود، ممکن است خطاهای بینی (کوچک) اتفاق بیافتد. مسلم است که نه تضمینی به قابلیت اطمینان لینک به لینک است و نه ردیابی خطا در حافظه، سپس UDP باید ردیابی خطا را در لایه انتقال انجام دهد، از طریق پایه پایان – پایان (on end – end basis)، اگر سرویس انتقال اطلاعات پایان – پایان، ردیابی خطا را تامین کند، این یک نمونه از قانون پایان – پایان مشهور از طراحی سیستم است که طبق این قانون چون قابلیت کارکردی خاص (در این مورد ردیابی خطا) باید بر پایه پایان-پایان پیاده سازی شود، کارکردهایی (توابعی) که در سطوح پایین قرار دارند ممکن است در مقایسه با هزینه تامین آن ها در سطح بالاتر اضافی باشند یا ارزش کمی داشته باشد.

چون فرض می شود که IP فقط در مورد هر پروتکل ۲ لایه اجرا می شود، برای لایه انتقال مفید است که چک کردن خطا را بعنوان یک مقیاس امن تامین نماید.اگرچه UDP چک کردن خطا را انجام می دهد، اما کاری در مورد بازیابی (recover) یک خطا را انجام نمی دهد. بعضی کاربردهای UDP به آسانی بخش صدمه دیده را رد می کنند اما بقیه بخش صدمه دیده را با یک پیام هشدار به برنامه می فرستند.

این موضوع، بحث UDP را به پایان می رسانند. بزودی خواهیم دید که TCP، کار انتقال اطلاعات را به برنامه های خود و سرویس های دیگری که UDP عرضه نمی کند را بعهده دارد. بطور طبیعی TCP از UDP پیچیده تر است.

قبل از بحث در رابطه با TCP، شایسته است که یک گام به عقب برگردیم و اول اصول اساسی انتقال مطمئن اطلاعات را مورد بحث قرار دهیم.

۴-۳ : اصول انتقال مطمئن اطلاعات

در این بخش به مساله انتقال مطمئن اطلاعات در یک قالب کلی می پردازیم. مساله پیاده سازی (اجرای) انتقال مطمئن اطلاعات نه تنها در لایه انتقال صورت می پذیرد بلکه در لایه ارتباط و لایه برنامه هم صورت می گیرد. پس این مسئله اهمیت خاصی برای شبکه سازی دارد. اینرو، اگر کسی بخواهد لیست ۱۰ مساله ی اساسی و مهم در همه شبکه سازی ها را معرفی کند، این مساله در صدر قرار خواهد گرفت. در بخش بعد TCP را بررسی می کنیم و بطور خاص نشان می دهیم که TCP از قوانین

wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

متعددی که ما درباره آنها توضیح خواهیم داد استفاده می کند.

شکل ۸.۳ قالبی را برای مطالعه ما پیرامون انتقال مطمئن اطلاعات فراهم می آورد. چکیده خدمات فراهم شده برای موجودیت(هویت) سطح بالاتر مربوط به یک کانال مطمئن است که از آن طریق اطلاعات را می توان انتقال داد. با یک کانال مطمئن هیچ بیت اطلاعات منتقل شده ای معیوب (معکوس شده از ۰ به ۱ و برعکس) و یا گم نمی شود و همه آن ها به همان ترتیبی که ارسال شده اند دریافت می شوند. این الگو سرویسی است که توسط TCP به برنامه های اینترنتی که آن را فعال می کنند پیشنهاد می شود.

این مسولیت پروتکل انتقال مطمئن اطلاعات است که این چکیده خدمات را اجرا (پیاده سازی) کند. مشکل در انجاست که ممکن است لایه زیر پروتکل انتقال، غیرمطمئن باشد. بطور مثال TCP یک پروتکل انتقال مطمئن اطلاعات است که در بالای لایه شبکه پایان به پایان (IP) نامطمئن پیاده سازی می شود. بطور کلی ،لایه زیر دو نقطه پایانی که ارتباط مطمئنی دارند ممکن است شامل یک لینک فیزیکی مجزا (مثل یک پروتکل لینک ارتباطی) یک یک شبکه گسترده سراسری (مثل پروتکل لینک انتقال) باشد. بهرحال، ما می توانیم لایه پایین تر را بعنوان یک کانال نقطه به نقطه غیر مطمئن در نظر بگیریم.

در این بخش، سگمنت های گیرنده و فرستنده یک پروتکل مطمئن را با در نظر گرفتن الگوهای پیچیده رو به ازدیاد کانال اساسی، توسعه خواهیم داد. شکل ۸.۳ (b)، واسط های پروتکل انتقال اطلاعات مارا نشان می دهد. بخش ارسال پروتکل انتقال اطلاعات مارا نشان می دهد. بخش ارسال پروتکل انتقال اطلاعات از بالا توسط تماس با ()rdt_send برانگیخته خواهد شد. این بخش اطلاعات را به لایه بالاتر در بخش دریافت ارسال می دارد. (rdt نشان دهنده پروتکل انتقال مطمئن اطلاعات است و send_ نشان می دهد که بخش ارسال rdt فراخوانده شده است. اولین قدم در توسعه هر پروتکل، انتخاب یک نام خوب برای آن است) . در بخش دریافت، زمانی که بسته اطلاعاتی به بخش دریافتی کانال رسید، () rdt_rcv فراخوانده می شود.

پروتکل rdt می خواهد اطلاعات را به لایه بالاتر تحویل دهد، این کار را با فراخواندن () deliver_data انجام می دهد. در زیر ما بجای سگمنت لایه ی انتقال از تکنولوژی pACKet استفاده می کنیم. چون تئوری ارائه شده در این بخش نه فقط به لایه انتقال اینترنت بلکه به شبکه های کامپیوتری بطور کلی اعمال می شود. در اینجا واژه عام pACKet مناسب تر است.

www.it98.ir

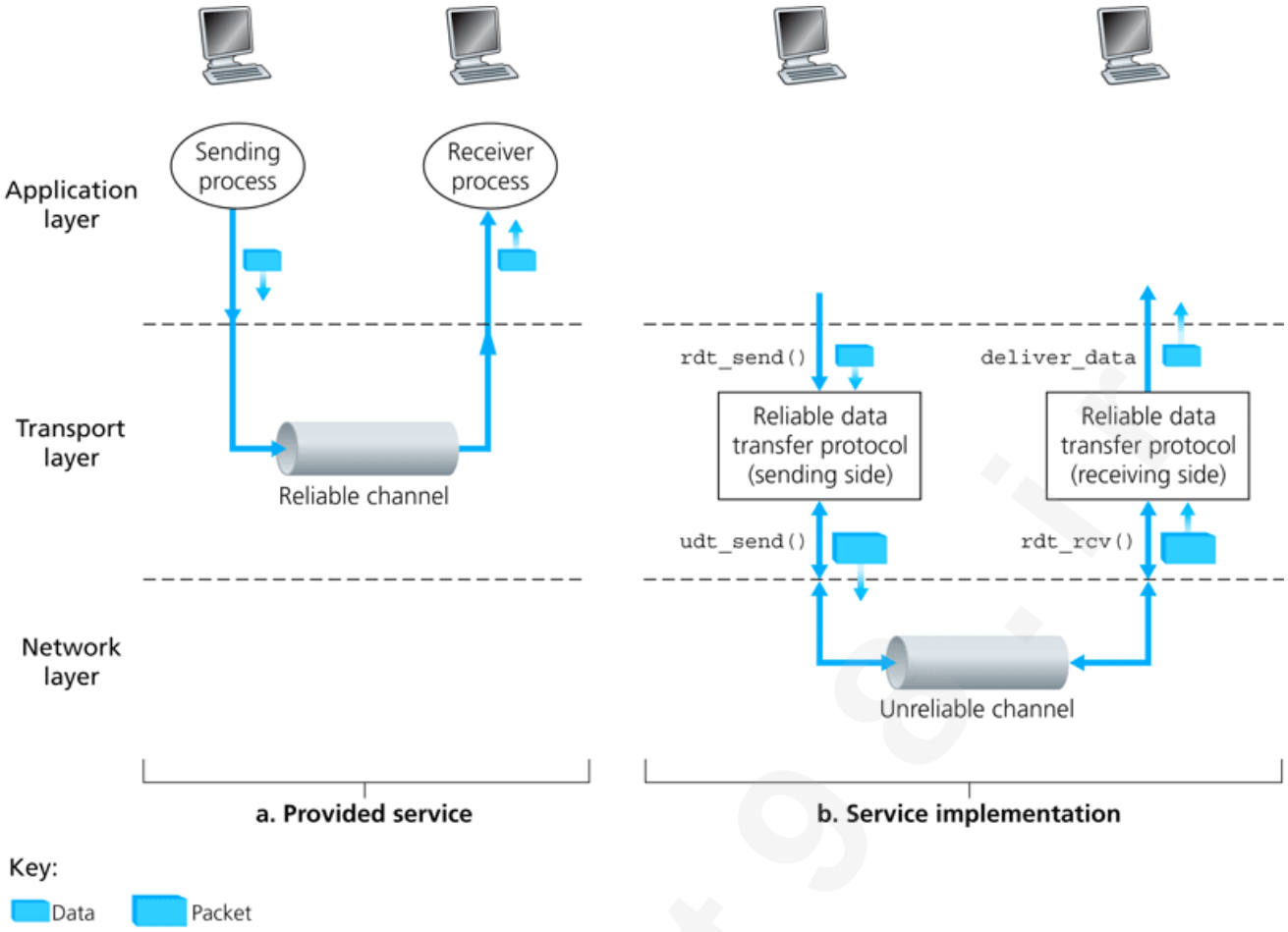


Figure 3.8 ♦ Reliable data transfer: Service model and service implementation

در این بخش فقط مورد انتقال یک سویه اطلاعات را در نظر می گیریم یعنی انتقال اطلاعات از بخش ارسال به دریافت. انتقال دوسویه اطلاعات هم زیاد سخت نیست اما توضیح آن خسته کننده است. اگرچه ما انتقال یک سویه اطلاعات را درنظر می گیریم، اما همانطور که در شکل ۸.۳ نشان داده شده، توجه به این نکته ضروری است که بخش های ارسال و دریافت پروتکل ما نیاز به انتقال بسته ها در دو جهت دارند. بطور مختصر، خواهیم دید که علاوه بر مبادله بسته ها شامل اطلاعاتی که باید انتقال داده شود، بخش های ارسال و دریافت rdt نیاز به مبادله رفت و برگشت بسته های کنترل دارند. هر دو بخش ارسال و دریافت rdt، بسته ها رابط فرمان udt_send() به سمت دیگر ارسال می کنند (udt یعنی انتقال نامطمئن اطلاعات)

۳-۴-۱ : ساختن یک پروتکل انتقال مطمئن اطلاعات

تاکنون ما در مجموعه ای از پروتکل ها گام برداشته ایم که یکی از دیگری پیچیده تر شده اند و حالا به پروتکل انتقال مطمئن و بی عیب اطلاعات رسیده ایم.

انتقال مطمئن اطلاعات از طریق یک کانال کاملا مطمئن : rdt 1.0

ابتدا ساده ترین مورد را در نظر می گیریم که در آن کانال اصلی کاملا مطمئن است. خود پروتکل که آن را rdt 1.0 خواهیم نامید، ناچیز (جزیی) است. تعاریف ماشین حالت-محدود(FSM) برای فرستنده و گیرنده rdt 1.0 در شکل ۹.۳ نشان داده شده

wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

است. FSM در شکل (a)۹.۳ عملکرد فرستنده را در حالیکه FSM های جداگانه ای برای فرستنده و گیرنده وجود دارند. FSM های فرستنده و گیرنده در شکل ۹.۳ (b) هرکدام فقط یک حالت دارند. پیکان ها نشان دهنده انتقال پروتکل از یک حالت به حالت دیگر است. (چون هر FSM در شکل ۹.۳ فقط یک حالت دارد، برگشت انتقال از یک حالت به خودش ضروری است. ما نمودارهای با حالت پیچیده تر را خواهیم دید.) اتفاقی که باعث انتقال شده در بالای خط افقی نشان دهنده انتقال و اعمالی که بواسطه این اتفاق بوقوع می پیوندد در زیر خط افقی نشان داده شده اند. وقتی هیچ عملی با وجود وقوع اتفاق صورت نمی پذیرد یا اتفاق نمی افتد و عمل انجام می شود از نماد A در پایین یا بالای خط افقی استفاده می کنیم که بترتیب نشان دهنده فقدان عمل یا فقدان اتفاق است. حالت اولیه FSM با پیکان تیره نشان داده می شود. اگرچه FSM ها در شکل ۹.۳ تنها یک حالت دارند، FSM هایی که خواهیم دید حالت های چندگانه دارند، بنابراین مهم است که حالت اولیه هر FSM را تشخیص دهیم.

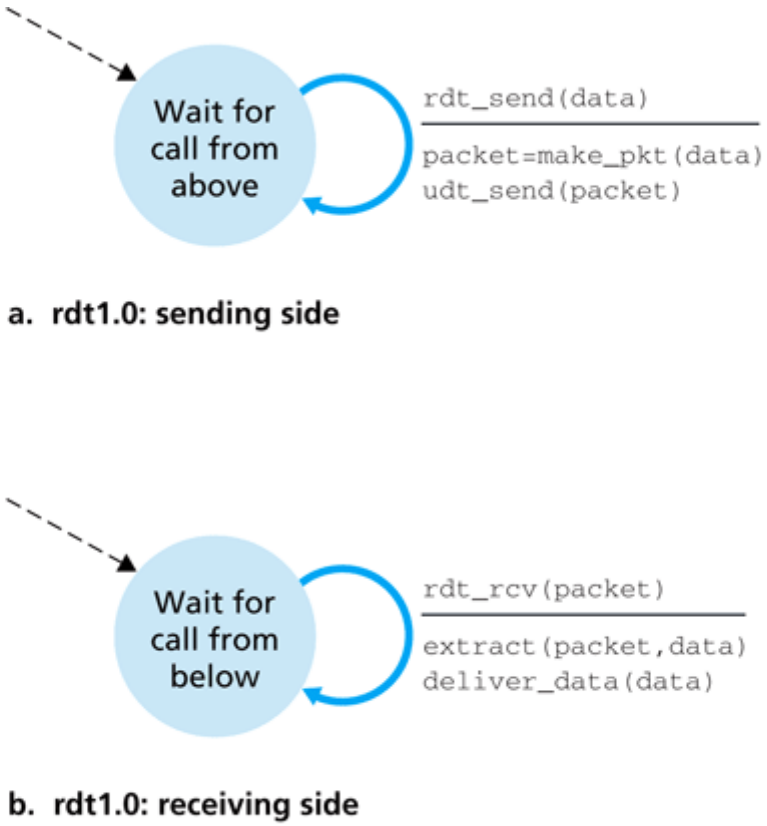


Figure 3.9 ♦ rdt1.0 – A protocol for a completely reliable channel

بخش ارسال rdt، به آسانی اطلاعات را از لایه بالایی از طریق اتفاق (واقعه) rdt_send(data) دریافت می کند، یک بسته شامل اطلاعات ایجاد می کند (از طریق عمل make_pkt(data)) و آن را به کانال می فرستد. در عمل اتفاق rdt_send(data) از یک فراخوان پروسس بوسیله برنامه لایه بالاتر (مثلا () rdt_send) منتج می شود.

در قسمت دریافت، rdt بسته اطلاعاتی را از کانال اصلی و از طریق رویداد rdt_rcv دریافت می کند، اطلاعات را از بسته استخراج می کند. (از طریق عمل extract (pACKet , data) و اطلاعات را به لایه بالایی می فرستد. (از طریق deliver_data(data). در عمل اتفاق rdt_rcv از فراخوانی پروسس از پروتکل پایین تر ناشی می شود. (مثلا () rdt_rcv).

www.it98.ir

در این پروتکل ساده، تفاوتی بین یک واحد از اطلاعات و یک بسته وجود ندارد. همچنین، همه گردش (جریان) بسته اطلاعاتی از فرستنده به گیرنده و یا از طریق کانال کاملا مطمئن انجام می شود و نیازی نیست که قسمت دریافت هیچ بازخوردی به فرستنده بفرستد چون چیزی اشتباه ارسال نمی شود ! توجه کنید که ما فرض می کنیم که دریافت کننده قادر است به همان سرعتی که فرستنده اطلاعات را ارسال می کند، آن ها را دریافت کند. بنابراین نیازی نیست که گیرنده از فرستنده بخواهد سرعت ارسال را کم کند.

ارسال مطمئن اطلاعات از یک کانال با خطاهای جزئی (بیتی) : rdt 2.0

الگوی واقعی تر کانال اصلی، الگویی است که در ان بیت ها در یک بسته اطلاعاتی ممکن است معیوب(خراب) باشند. چنین خطاهای بیتی در اجزای فیزیکی یک شبکه در هنگامی که بسته ارسال، منتشر(تکثیر) و یا بافر می شود، اتفاق می افتد. برای لحظه ای فرض می کنیم که همه بسته های ارسالی به همان ترتیبی که ارسال شده اند، دریافت شده اند.

قبل از توسعه یک پروتکل برای ارتباط منطقی از طریق یک کانال، اول در نظر می گیریم که افراد چگونه ممکن است با چنین شرایطی سروکار داشته باشند. درنظر بگیرید که چطور خود شما ممکن است یک پیام طولانی را از متن دیکته کنید. در سناریو فرضی، گیرنده پیام ممکن است بعد از هر جمله ای که می شنود، می فهمد و ضبط می کند بگوید. اگر او یک جمله درهم بشنود از شما می خواهد جمله را تکرار کنید. این پروتکل دیکته – پیام از هردو تایید مثبت (OK) و تایید منفی (please repeat that) استفاده می کند. این کنترل پیام ها به گیرنده امکان می کند فرستنده را از آنچه بطور صحیح دریافت شده و آنچه خطایی در آن بوده و باید تکرار شود، آگاه سازد. در نصب یک شبکه کامپیوتری، پروتکل های ارسال مطمئن اطلاعات که به چنین ارسال های مجددی متکی هستند، بعنوان پروتکل ها (Automatic Repeat request) یا ARQ شناخته می شوند.

اساساً سه پروتکل اضافی در پروتکل های ARQ مورد نیاز است تا وجود خطاهای بیتی را اداره کند.

ردیابی خطا : اول، به یک مکانیزم نیاز داریم تا به گیرنده امکان دهد وقتی خطاهای بیتی اتفاق می افتند، آنها را شناسایی(ردیابی) کند. از بخش به یاد دارید که UDP برای این منظور از فیلد اینترنتی مجموع checksum استفاده می کرد. در فصل ۵ تکنیک های ردیابی و اصلاح خطا را با جزییات بیشتری بررسی خواهیم کرد، این تکنیک ها به گیرنده امکان می دهند خطاهای بیتی بسته اطلاعاتی را ردیابی و در صورت امکان اصلاح کند. برای حالا تنها باید بدانیم که این تکنیک ها نیاز دارند که بیت های اضافه ای از فرستنده به گیرنده ارسال شوند. این بیت ها در فیلد checksum بسته اطلاعاتی rdt 2.0 جمع آوری خواهد شد.

بازخورد گیرنده : چون گیرنده و فرستنده معمولا روی سیستم های نهایی متفاوتی اجرا می شوند ،و ممکن است هزاران مایل از هم دور باشند، تنها راه برای آگاهی فرستنده از دید گیرنده نسبت به جهان (در این مورد آیا بسته بطور صحیح دریافت شده یا نه) این است که گیرنده بازخورد صریحی را به فرستنده بدهد. تاییدات مثبت (ACK) و منفی (NAK) که در سناریو دیکته-پیام بیان شدند نمونه ای از چنین بازخوردی هستند. پروتکل rdt 2.0 ما، بسته های ACK و NAK را از گیرنده به فرستنده پس می فرستد. در اصل، این بسته ها تنها نیاز دارند اندازه یک بیت طول داشته باشند مثلا داده ۰ یک NAK را نشان می دهد و ۱ نشان دهنده یک ACK است.

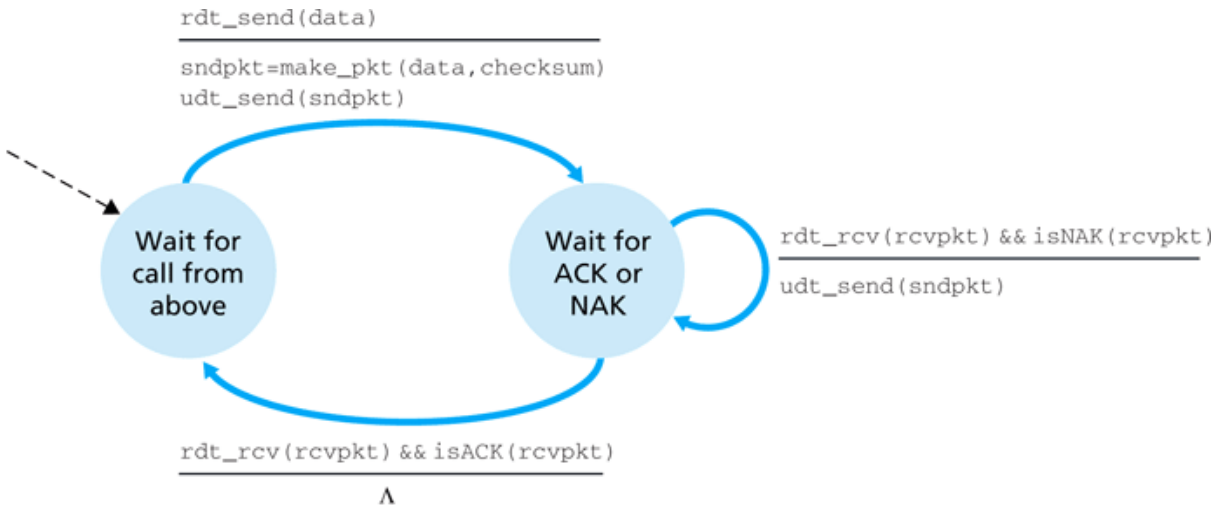
ارسال مجدد : بسته ای که بصورت اشتباه دریافت شده مجددا به فرستنده بازپس فرستاده می شود.

شکل ۱۰.۳ نشان می دهد که نمایش rdt 2.0 FSM، یک پروتکل ارسال اطلاعات از ردیابی خطا، تاثیرات مثبت و منفی بهره می برد. قسمت ارسال rdt 2.0 دو حالت دارد. در سمت چپ، پروتکل قسمت ارسال منتظر است تا اطلاعات از لایه بالایی به پایین حرکت کند. وقتی رویداد rdt_send (data) با آرگومان (data) بوقوع می پیوندد، فرستنده یک بسته اطلاعاتی (sndpkt) خواهد ساخت که شامل اطلاعاتی است که باید ارسال شوند همراه با checksum بسته را از طریق send_send

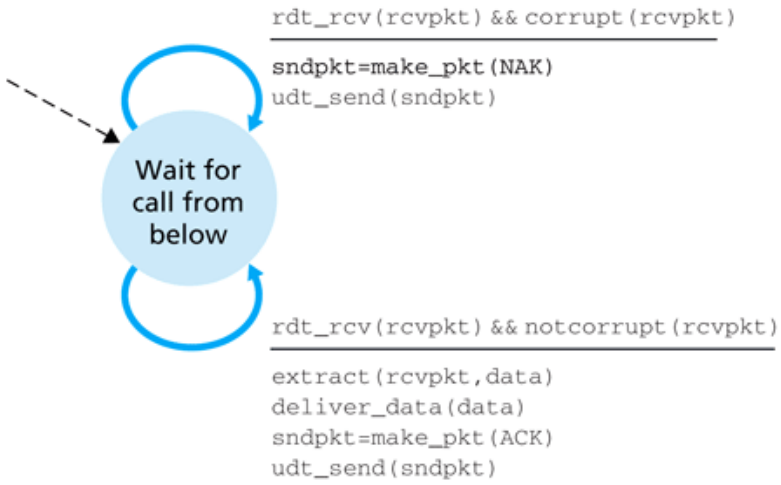
wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

با آرگومان (data) ارسال می کند. (برای مثال همان طور که در بخش ۳,۳,۲ در مورد سگمنت UDF بحث شد) در سمت راست، پروتکل فرستنده منتظر بسته ACK یا NAK از گیرنده است. اگر بسته ACK دریافت شود (نماد rdt_rcv(rcvpkt) و isACK(rcvpkt) در شکل ۱۰.۳ با این اتفاق مطابقت دارند)، فرستنده می داند که بسته هایی که اخیرا ارسال شده بطور صحیح دریافت شده اند و بنابراین پروتکل به حالت انتظار برای اطلاعات از لایه بالاتر برمی گردد. اگر پیام NAK دریافت شد، پروتکل آخرین بسته را دوباره ارسال می کند و منتظر پاسخ ACK یا NAK از گیرنده برای ارسال مجدد بسته ها می ماند. شایان توجه است که وقتی فرستنده در حالت انتظار برای حالت NAK یا ACK است نمی تواند اطلاعات بیشتری را از لایه بالایی بگیرد، یعنی اتفاق rdt_send() رخ نمی دهد چون این اتفاق تنها بعد از دریافت پاسخ ACK رخ می دهد و این حالت را ترک می کند. بنابراین فرستنده اطلاعات جدیدی را نمی فرستد تا زمانی که مطمئن شود گیرنده بسته اخیر را بطور صحیح دریافت کرده است. به خاطر این رفتار، پروتکل هایی مثل rdt 2.0 به پروتکل های stop-and-wait معروفند.



a. rdt2.0: sending side



b. rdt2.0: receiving side

Figure 3.10 ♦ rdt2.0–A protocol for a channel with bit errors

www.it98.ir

بخش دریافت FSM برای rdt 2.0 هنوز یک تک حالت دارد. در هنگام رسیدن بسته، گیرنده با پیام ACK یا NAK با توجه به اینکه بسته معیوب دریافت شده یا نه، پاسخ می دهد. در شکل ۱۰.۳، نماد Corrupt(rcvpkt) && rdt_rcv(rcvpkt) x با این رویداد مطابقت دارند که در آن بسته دریافت شده و آن خطا مشاهده شده است.

ممکن است چنین بنظر برسد که پروتکل rdt 2.0 کار می کند، اما متاسفانه، دچار نقص مهلکی است. بطورخاص، ما احتمال اینکه بسته های ACK یا NAK معیوب باشند را به حساب نمی آوریم. (قبل از اینکه جلوتر برویم، باید درباره اینکه چطور مساله را حل کنیم، فکر کنیم). متاسفانه، اشتباه ناچیز ما، آنچنان که ممکن است بنظر برسد، بی ضرر نیست. حداقل، ما باید بیت های checksum را به بسته های ACK یا NAK اضافه کنیم تا چنین خطاهایی را ردیابی کنند. سوال مشکل تر آن است که چطور پروتکل، باید خطاهای بسته های ACK یا NAK را بازیافت کند. مشکل در اینجاست که اگر ACK یا NAK معیوب باشند، فرستنده راهی برای اگاهی از اینکه آیا گیرنده آخرین اطلاعات ارسال شده را درست دریافت کرده یا نه، ندارد.

در انتخاب اول، فکر کنید یک انسال در سناریو دیکته پیام چه کاری ممکن است انجام دهد. اگر گوینده پاسخ "Ok" یا " Please Repeat that " را از گیرنده درک نکند، احتمالا خواهد پرسید : " what did you say " (بنابراین نوع جدیدی از بسته اطلاعاتی فرستنده به گیرنده به پروتکل ما اضافه می شود). سپس گوینده پاسخ را تکرار می کند. اما چه اتفاقی می افتد اگر جمله " what did you say" ناقص دریافت شود (خراب شود) ؟ گیرنده نمی داند که آیا این جمله درهم، بخشی از دیکته بود یا تقاضا برای تکرار آخرین جواب، که احتمالا در جواب خواهد گفت : "what did you say " و البته ممکن است این جواب هم مبهم باشد.

انتخاب دوم، اضافه کردن بیت های کافی checksum است تا به فرستنده امکان دهد نه تنها خطاهای بیتی را ردیابی کند، بلکه آن ها را بازیافت هم کند. این امر، مشکل نوری کانال را که می تواند بسته ها را خراب کند اما آنها را مفقود نمی کند، حل می کند.

راه حل سوم برای فرستنده، این است که وقتی یک بسته اطلاعاتی ACK یا NAK دریافت شد، بسته اطلاعاتی اخیر را مجددا ارسال نماید. این روش، به معرفی بسته های تکراری (المثنی) در کانال فرستنده به گیرنده می پردازد. اشکال اصلی این بسته ها آن است که گیرنده نمی داند آیا آخرین ACK یا NAK که ارسال کرده، بطور صحیح توسط فرستنده دریافت شده است یا نه. بنابراین، نمی داند بسته ای که دریافت کرده شامل اطلاعات جدید است یا یک ارسال مجدد است.

یک راه حل ساده برای این مشکل جدید آن است که یک فیلد جدید به بسته اطلاعاتی اضافه کنیم و با قراردادن (درنظر گرفتن) یک شماره توالی برای این فیلد کاری می کنیم که فرستنده شماره بسته های اطلاعاتی خود را داشته باشد. در این صورت گیرنده برای تعیین اینکه ایا بسته دریافتی یک ارسال مجدد است یا نه، تنها نیاز دارد این عدد را چک کند. در مورد پروتکل ساده Stop and wait، یک عدد توالی ۱ بیتی کافی است، چون این عدد به گیرنده امکان خواهد داد تا بداند آیا گیرنده، بسته ارسال شده قبلی را دوباره فرستاده (در اینصورت شماره توالی بسته رسیده همان شماره بسته ای است که الآن به دست ما رسیده) و یا یک بسته جدید است (شماره فرق می کند). چون ما کانالی را فرض کرده ایم در آن بسته ها گم نمی شوند، پس بسته های ACK یا NAK نیازی به نشان دادن شماره توالی بسته ندارند. فرستنده می داند که بسته ACK یا NAK رسیده در پاسخ به آخرین بسته اطلاعاتی ارسال شده است.

شکل های ۱۱.۳ و ۱۲.۳ تعریف FSM برای rdt 2.1 که نسخه ثابتی از rdt 2.0 است را نشان می دهد. FSM های فرستنده و گیرنده rdt 2.1، هرکدام دو حالت دارند. دلیل این امر، آن است که پروتکل باید بازتاب کننده این مطلب باشد که بسته ای که اخیرا ارسال شده (توسط فرستنده) دریافت شده (توسط گیرنده) باید یک عدد توالی ۰ یا ۱ را بپذیرد، توجه کنید که اعمال در شرایطی که بسته شماره ۰ ارسال یا دریافت شده، اعمالی هستند که در بسته های شماره ۱ ارسالی یا دریافتی مشاهده می شوند نه تنها تفاوت در نحوه استعمال عدد توالی است.

www.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

پروتکل rdt 2.1 از هر دو تایید مثبت و منفی از گیرنده یا فرستنده استفاده می کند. وقتی یک بسته خارج از دستور دریافت شد، گیرنده یک پیام مثبت برای بسته ای که دریافت کرده، می فرستد. وقتی بسته ناقص دریافت شد، پیام منفی می فرستد. اگر بجای ارسال یک NAK، پیام ACK را برای آخرین پیام دریافت شده ناقص بفرستیم، همان تاثیری را دارد که NAK دارد. فرستنده برای یک پیام دو ACK دریافت می کند. (یعنی ACK تکراری دریافت می کند) و می داند که گیرنده، پیام را درست دریافت نکرده است.

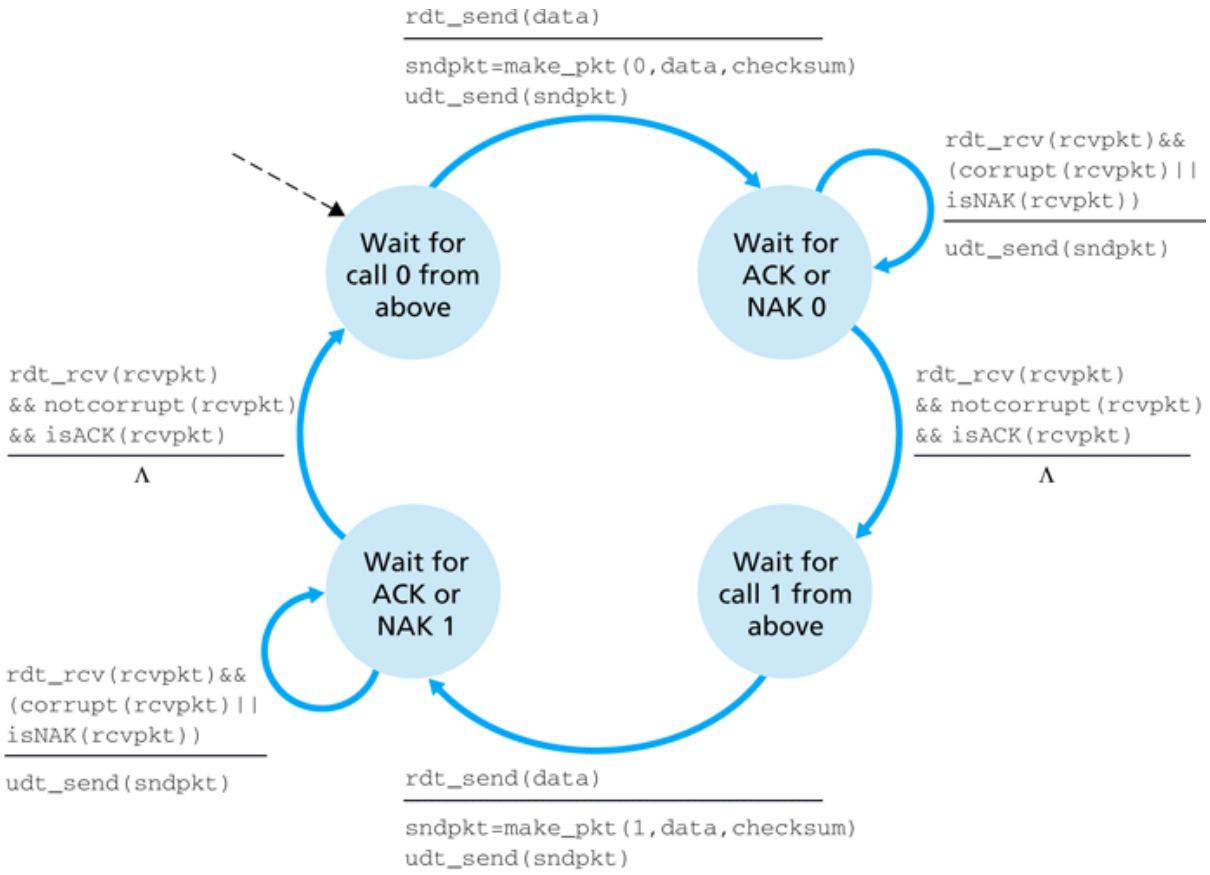


Figure 3.11 ♦ rdt2.1 sender

پروتکل ارسال مطمئن اطلاعات بدون NAK برای یک کانال با خطاهای بیتی در rdt 2.2 در شکل های ۱۳.۳ و ۱۴.۳ نشان داده شده است. یک تفاوت ظریف بین rdt 2.1 و rdt2.2 این است که در اینجا گیرنده باید در عدد توالی بسته تائید شده یک پیام ACK را بگنجاند. (این کار را می توان با گنجاندن ۰ و ACK یا ۱ ACK در FSM گیرنده make_pkt() انجام داد) و فرستنده باید پیام ACK دریافت شده را در عدد توالی بسته تائید شده بگنجاند (با استفاده از گنجاندن ۰ یا ۱ در isACK() در FSM فرستنده).

www.it98.ir

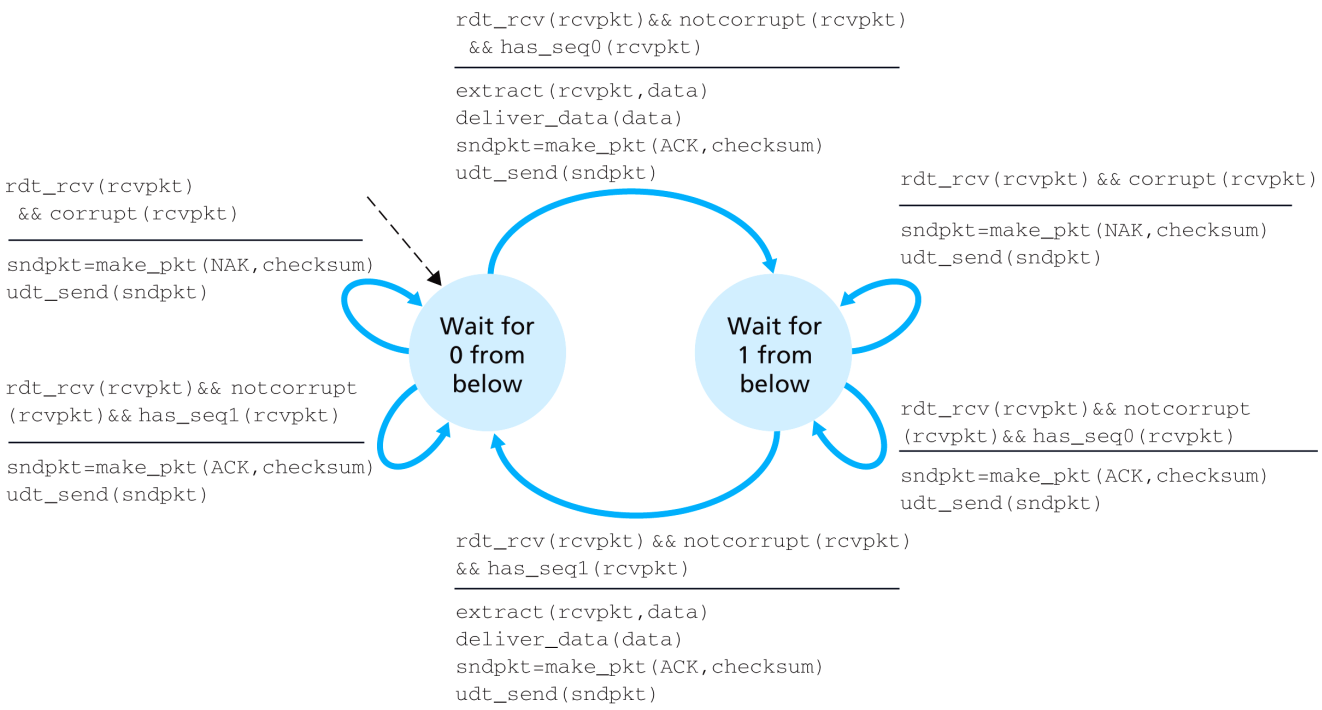


Figure 3.12 ♦ rdt2.1 receiver

انتقال مطمئن اطلاعات از طریق کانال Lossy با خطاهای بیتی : rdt 3.0

فرض کنید که علاوه بر بیت های خراب، کانال اصلی می تواند بسته ها را گم هم کند، یک اتفاق نه چندان غیرعادی در شبکه های امروزی کامپیوتر(از جمله اینترنت) . اکنون پروتکل با دو مسئله جدید روبرو است : چطور بسته گم شده را ردیابی کند و وقتی بسته گم شد چکار کند. برای پاسخ سوال دوم می توانیم از تکنیک های checksum، اعداد توالی، بسته های ACK، و ارسال مجدد که قبلا در rdt 2.2 با ان ها آشنا شدیم، استفاده کنیم. اما برای مورد اول نیاز به اضافه کردن یک مکانیزم جدید پروتکل داریم.

راه های متعددی برای پرداختن به مسئله بسته گم شده وجود دارد. در اینجا با ردیابی و بازیافت بسته های گم شده را بر دوش فرستنده می اندازیم. فرض کنید که فرستنده یک بسته را ارسال می کند و آن بسته سمت ACK گیرنده گم می شود. در هر مورد، پاسخی آماده ارائه از طرف گیرنده برای فرستنده نیست. اگر فرستنده مدت زمان طولانی منتظر باشد مطمئن می شود که بسته گم شده است، و می تواند بسته را مجددا ارسال کند. باید خودتان را متقاعد کنید که این پروتکل واقعا کار می کند.

www.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

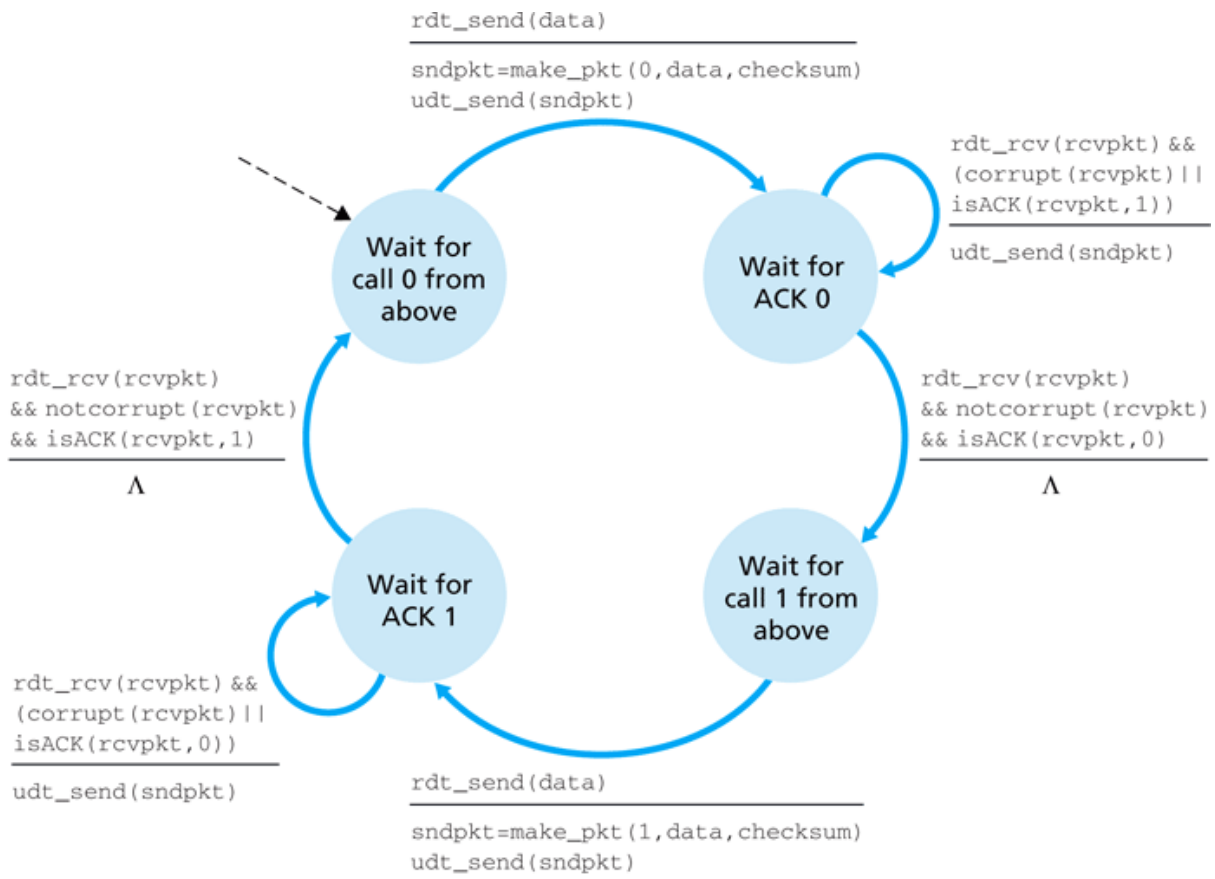


Figure 3.13 ♦ rdt2.2 sender

اما فرستنده چقدر باید منتظر بماند تا مطمئن شود که بسته گم شده است؟ فرستنده باید حداقل به اندازه یک تاخیر دو طرفه بین فرستنده و گیرنده (شامل بافرینگ در مسیریاب های میانجی) بعلاوه مقدار زمان مورد نیاز برای پردازش بسته در گیرنده صبر کند. در بیشتر شبکه ها، برآورد این حداکثر تاخیر بسیار سخت است و کمتر می توان با اطمینان در مورد آن نظر داد. بعلاوه، پروتکل باید بسته را تا حد امکان بازیافت کند بنابراین حداقل زمان انتظار باید شروع بازیابی خطا باشد. به این روش که در عمل پذیرفته شده، فرستنده باید دقت یک مقدار زمانی را انتخاب کند که طی آن گم شدن بسته اتفاق می افتد. توجه کنید که اگر بسته تاخیر زیادی داشته باشد، ممکن است فرستنده بسته را مجددا ارسال کند در صورتی که ممکن است نه بسته و نه پاسخ ACK آن گم نشده باشد. این کار، امکان وجود بسته های اطلاعاتی تکراری را در کانال فرستنده به گیرنده ممکن می سازد. خوشبختانه ۷ پروتکل rdt 2.2، کارکرد لازم برای مدیریت بسته های تکراری را دارد.

www.it98.ir

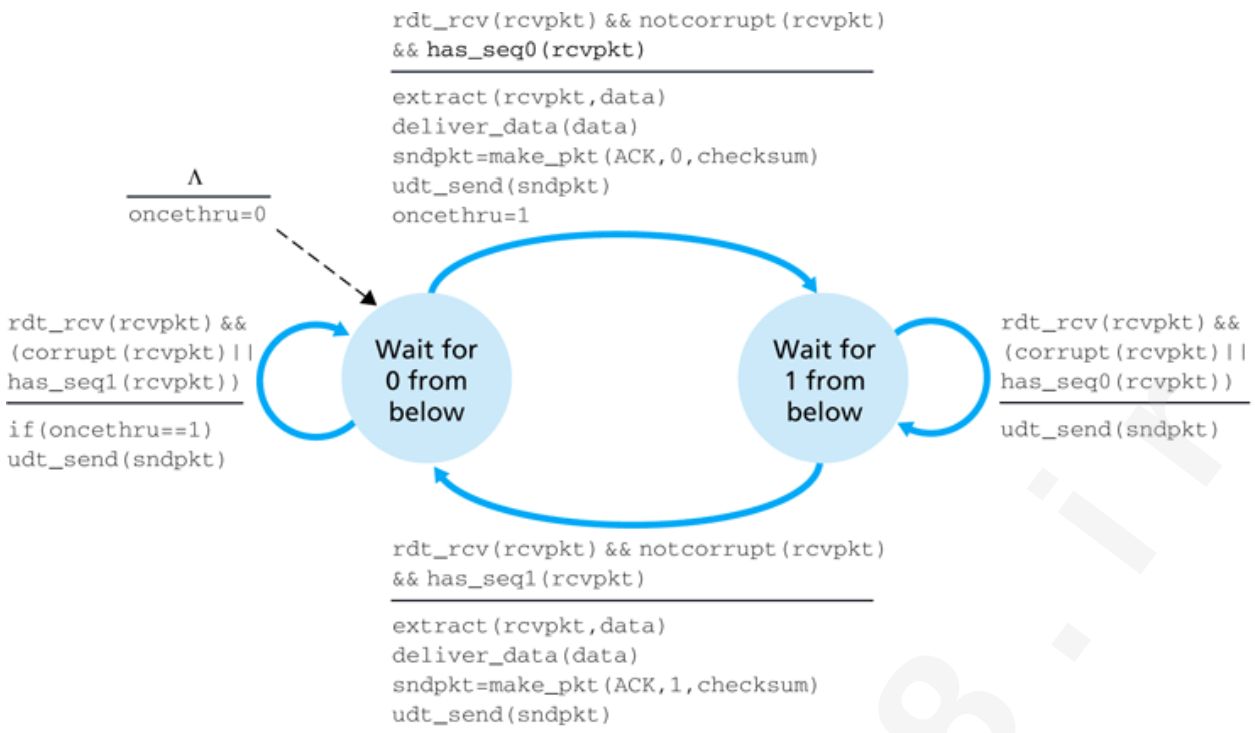


Figure 3.14 ♦ rdt2.2 receiver

از نقطه نظر فرستنده، ارسال مجدد یک نوشدارو است. فرستنده نمی داند بسته گم شده، ACK گم شده یا از فقط بسته و ACK زیادی تاخیر دارند. در تمام موارد، اقدام مشابه است، ارسال مجدد پیاده سازی یک مکانیزم ارسال مجدد متکی به زمان، نیازمند یک تایمر شمارش معکوس (Countdown timer) است که می تواند فرستنده را بعد از اینکه زمان مورد نظر به پایان رسید، متوقف کند. بنابراین فرستنده نیاز دارد بتواند :

۱ : هربار که بسته ارسال شد تایمر را بکار اندازد. (چه بسته ای که برای اولین بار ارسال می شود چه ارسال مجدد)

۲ : به وقفه تایمر پاسخ دهد (با اجرای اعمال مقتضی)

۳ : تایمر را از کار اندازد.

شکل ۱۵.۳، FSM فرستنده برای rdt 3.0 را نشان می دهد، یک پروتکل که انتقال صحیح اطلاعات را از طریق کانال انجام می دهد و این کانال می تواند بسته ها را ناقص یا گم کند، در مسائل خانگی، از شما خواسته می شود که FSM گیرنده را برای rdt 3.0 فراهم کنید. شکل ۱۶.۳ نشان می دهد که چطور پروتکل، بسته ها را بدون تاخیر یا گم شدن اداره می کند و چطور بسته های گم شده را مدیریت می کند.

در شکل ۱۶.۳، زمان از بالای نمودار به پایین آن شروع به جلورفتن می کند، توجه کنید که زمان دریافتی برای یک بسته لزوما بیشتر از زمان ارسال بسته است که علت این امر، تاخیرهای ارسال و انتشار (تکثیر) است. در شکل ۱۶.۳ از b تا d، پرانتزهای بخش ارسال زمانی را که یک تایمر برای شروع و پایان نیاز دارد را نشان می دهد. ظریف ترین ابعاد این پروتکل در تمرین های آخر این فصل بررسی شده اند. چون اعداد توالی بسته بین ۰ و ۱ تغییر می کنند، پروتکل rdt 3.0 بعضی وقت ها پروتکل بیت – جایگزین (alternating-bit Protocol) هم نامیده می شود.

wWw.it98.ir

www.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

۳-۴) اصول انتقال مطمئن داده ها

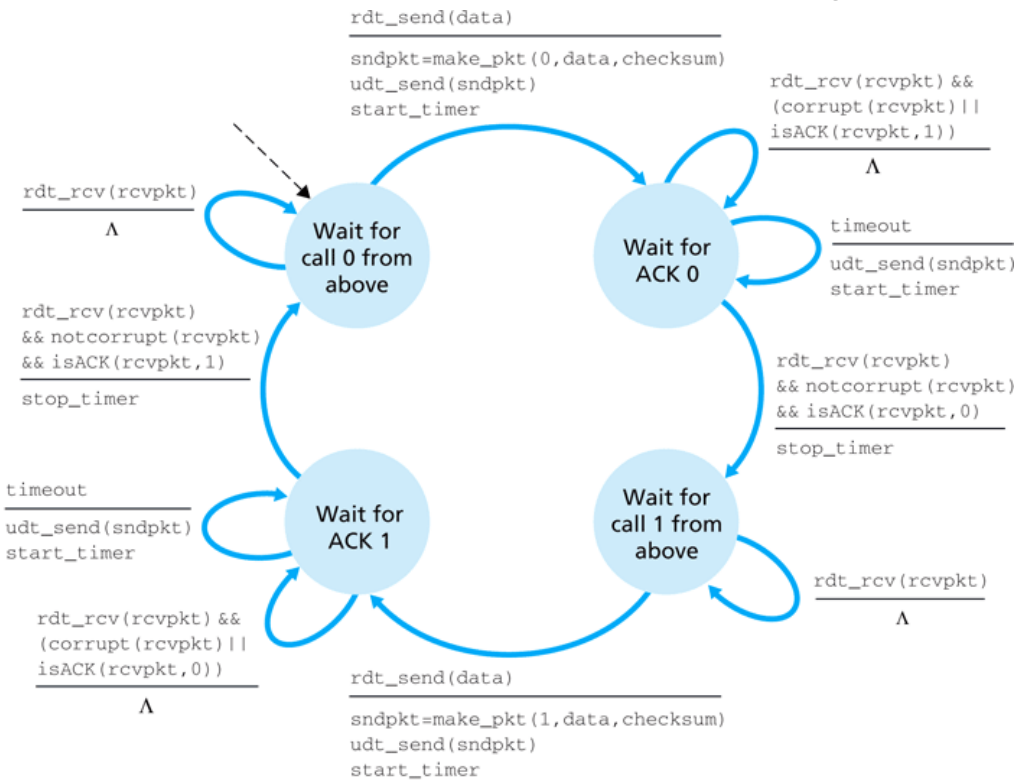


Figure 3.15 ♦ rdt3.0 sender

شکل ۱۵-۳) فرستنده rdt3.0

ما تاکنون عناصر اصلی پروتکل انتقال داده ها را گردآوری کرده ایم. Check sum ها، اعداد ترتیبی، زمانبندی ها و بسته های خبری مثبت و منفی، که هر کدام نقش مهم و لازمی را در عملیات یک پروتکل به عهده دارند . اکنون ما با پروتکل انتقال داده مطمئن روبرو هستیم.

۳.۴.۲) پروتکلهای انتقال داده مطمئن لوله ای

پروتکل rdt3.0، درست کار می کند اما به نظر نمی رسد که کسی از عملکرد آن راضی باشد، به خصوص در شبکه های پر سرعت امروزی مشکلی که در قلب عملکرد rdt3.0 وجود دارد این است که این، پروتکل توقف و انتظار است.

برای بهبود تاثیر عملکردی رفتار توقف و انتظار، یک حالت ایده آل از دو میزبان (host) را در نظر بگیرید که یکی در سواحل غربی ایالات متحده قرار گرفته باشد و دیگری در سواحل شرقی، همان طور که در شکل 3.17 نشان داده شده است. تاخیر انتشار در سرعت یک دور سفر نور بین دو سیستم در نقاط نهایی، RTT، حدودا ۳۰ میلی ثانیه است. فرض کنید این دو نقطه با کانالی با سرعت انتقال R از یک GBPs (بیت در ثانیه) به هم بپیوندند. با بسته ی با سایز L شامل ۱۰۰۰ بایت (۸۰۰۰ بیت) به ازای هر بسته، که هم شامل بخش عنوان ها و هم داده ها می شود، زمان مورد نیاز برای انتقال واقعی بسته

www.it98.ir

به یک لینک (link) یک GBPs این گونه محاسبه می شود

dtrans = L = 8000Bit/packet = 8 میکرو ثانیه

10^9 R bits /sec

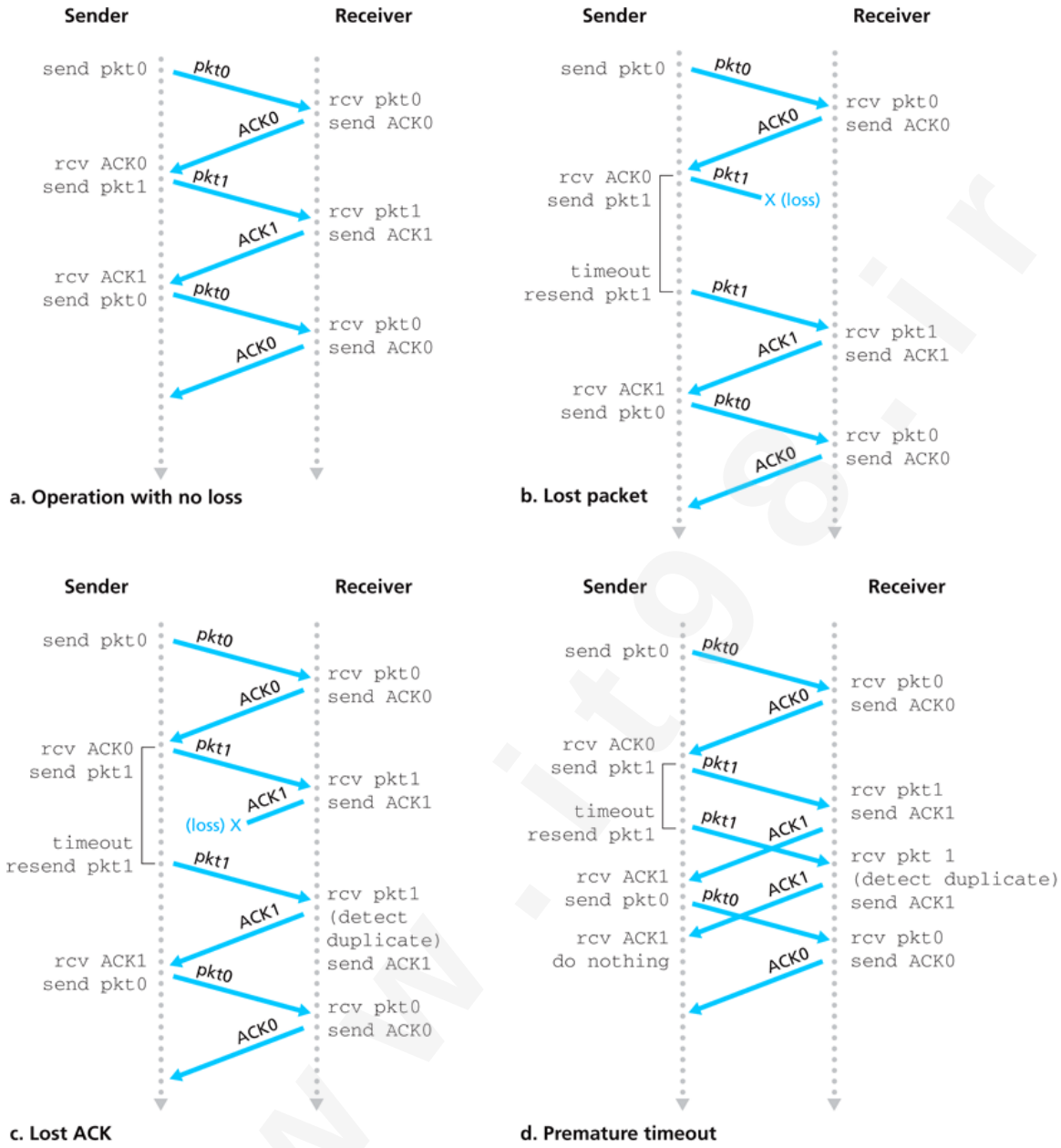


Figure 3.16 ♦ Operation of `rdt3.0`, the alternating-bit protocol

www.it98.ir

wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

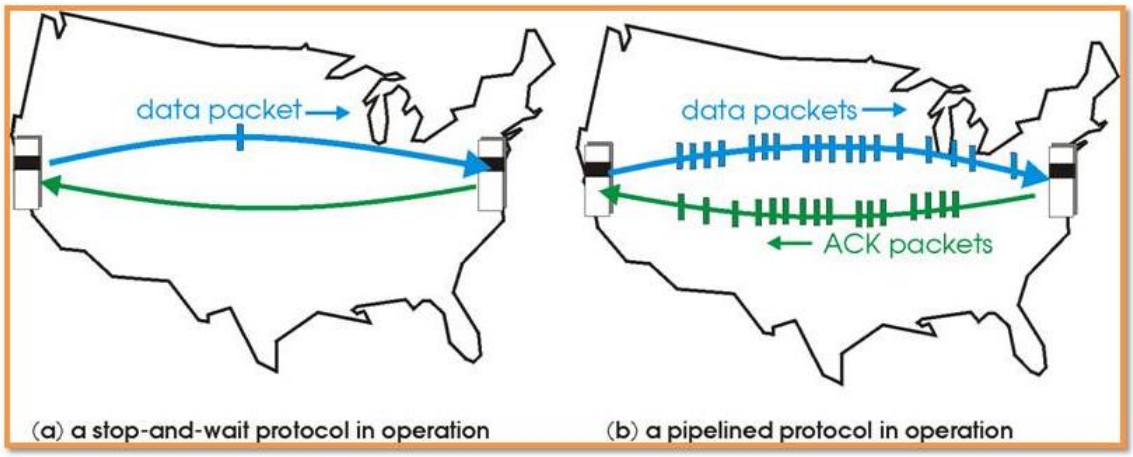
شکل ۱۶-۳) عملیات Rdt3.0، پروتکل بیت جایگزین

عملیات بدون هیچ شکستی

بسته گمشده

ACK گمشده

اتمام زمان قبل از موعد



شکل ۱۷-۳) پروتکل توقف و انتظار در برابر پروتکل لوله ای

پروتکل توقف و انتظار در حال فعالیت

پروتکل لوله ای در حال فعالیت

شکل (a)۱۸-۳ نشان می دهد که با پروتکل توقف و انتظار، اگر فرستنده شروع به ارسال بسته ها در t=0 و سپس در t=L/R=8 میکروثانیه کند، آخرین بیت ورودی به کانال از سمت فرستنده خواهد بود. سپس بسته صفر ۱۵ میکرو ثانیهای خود در عبور از کشور را شروع می کند و آخرین بیت از بسته t= RTT/2+L/R=15.008 میکروثانیه به گیرنده خواهد رسید.

برای سادگی کار فرض می کنیم که بسته های ACK بسیار بسیار کوچکند (و ما می توانیم زمان انتقال آنها را نادیده بگیریم.) و گیرنده می تواند یک ACK را درست موقعی که آخرین بیت از بسته داده ها دریافت می شود، ارسال کند، ACK در t=RTT+L/R=30/008 میکرو ثانیه به فرستنده بر می گردد. در این نقطه، فرستنده می تواند پیام بعدی را انتقال دهد. بنابراین در 30/008 میکروثانیه فرستنده فقط یک پیام 0/008 میکروثانیه ای را فرستاده است. اگر ما بهره وری (utilization) فرستنده (یا کانال) را کسری از زمان در نظر بگیریم که فرستنده کاملا مشغول فرستادن بیتها به کانال است، تحلیل های شکل 3.18(a) نشان می دهند که پروتکل توقف و انتظار، بهره وری رضایت بخشی از سمت فرستنده ندارد و Usender این گونه محاسبه می شود:

$$U\ sender = L/R = 0/008 = 0/00027$$

www.it98.ir

RTT+L/R 30/008

یعنی، فرستنده تنها ۲/۷ از صد در صد زمان مشغول بوده است! به عبارت دیگر، فرستنده تنها قادر به فرستادن ۱/۰۰۰ بایت در ۳۰/۰۰۸ میلی ثانیه بوده است، یعنی ورودی تنها ۲۶۷ کیلوبایت در ثانیه (kbps)، هر چند که یک ارتباط یک Gbps در دسترس بوده است ! مدیر شبکه ی ناراحتی را تصور کنید که مبلغ زیادی را برای ظرفیت ارتباطی یک گیگابایت پرداخته است، اما تنها خروجی ۲۶۷ کیلو بیت در ثانیه می گیرد! این نمونه نمایشی از این است که چطور پروتکل های شبکه ای، قابلیتهای سخت افزاری شبکه ای را محدود می کنند. و نیز ما زمانهای پردازش در سطوح پایین لایه های پروتکل را در سمت فرستنده و گیرنده، و نیز تاخیرهای پردازش و صفی که در هر روتر میانی (Router) بین فرستنده و گیرنده بوجود می آید را هم در نظر نگرفته ایم. در نظر گرفتن این تاثیرها، تنها باعث افزایش بیشتر تاخیر و نمود بیشتر عملرد ضعیف می شود.

راه حل این مشکل عملکردی خاص ساده است: بجای عملکرد به حالت توقف و انتظار، به فرستنده اجازه داده شود تا چندین بسته به خارج برای کسب خبر وصول فرستاده شوند همان طور که در شکل (b) ۱۷-۳ نشان داده شده است. شکل (b) ۱۸-۳ نشان می دهد که اگر فرستنده اجازه فرستادن سه بسته را پیش از انتظار برای دریافت خبر وصول داشته باشد، کارآیی (بهره وری) فرستنده در اصل سه برابر می شود. از آنجا که می توان بسته های زیادی در مسیر فرستنده به گیرنده که در حال انتقالند را مشاهده کرد، انگار که یک لوله را پر کرده باشند، به این تکنیک، فن انتقال لوله ای (pipelining) گفته می شود pipelining دستاوردهای زیر را برای پروتکل انتقال داده ها در بردارد.

تعداد اعداد ترتیبی باید افزایش یابد زیرا هر بسته در حال انتقال (بدون محاسبه انتقال های دوباره) باید عدد ترتیبی خود را داشته باشد، و ممکن است چندین بسته در حال انتقال وجود داشته باشد که هنوز بسته اعلام وصول آنها به سمت فرستنده نرسیده است.

طرف فرستنده و گیرنده ی پروتکل ممکن است بیشتر از یک بسته را ارسال کنند. در کمترین حالت، فرستنده باید دوباره بسته هایی را که انتقال داده است اما هنوز اعلام وصولشان نیامده را دوباره ارسال کند. دوباره ارسال کردن بسته های درست رسیده بی مقصد هم ممکن است از سمت گیرنده دوباره لازم باشد، همان طور که در ادامه بحث خواهیم کرد.

تعداد اعداد ترتیبی مورد نیاز و نیازهای دوباره ارسالی به روش برخورد پروتکل انتقال داده با بسته های زیاد تاخیر کرده دچار نقص شده یا گمشده بستگی دارد. دو رویکرد پایه در برخورد با بازیابی خطاها در pipelined شناسایی شده اند: Selective Repeat, Go-Back- N.

۳.۴.۳ Go-Back- N (GBN)

در پروتکل GBN به فرستنده اجازه داده می شود تا چندین بسته را بدون دریافت اعلام وصول انتقال دهد (هر زمان که بتواند)، ولی محدودیت ارسالی دارد که بیش از از تعداد مشخصی، N، اجازه ارسال بسته های بدون اعلام وصول را به لوله انتقال ندارد. ما در این بخش پروتکل GBN را با جزئیات توضیح می دهیم ولی پیش از از جلوتر رفتن از شما می خواهیم تا با GBN applet (یک applet بسیار حیرت انگیز!) در وب سایت مورد نظر کار کنید.

شکل ۱۹-۳ دیدگاه فرستنده از تعداد اعداد ترتیبی در پروتکل GBN را نشان می دهد. اگر ما پایه را عدد ترتیبی آخرین بسته بدون اعلام وصول در نظر بگیریم، و عدد ترتیبی بعدی (nextseqnum) کوچکترین عدد ترتیبی استفاده نشده (یعنی، عدد ترتیبی بسته بعدی که قرار است ارسال شود) باشد، در این حالت چهار فاصله در ترتیب اعداد ترتیبی مشخص می شود. اعداد ترتیبی در فاصله ی (0,base-1) برای آن دسته از بسته هایی هستند که ارسال شده و اعلام وصولشان دریافت شده است فاصله ی (base-Nextseqnam-1) برای آن دسته از بسته هایی است که ارسال شده اند ولی علام وصولشان نیامده است.

www.it98.ir

wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

اعداد ترتیبی در فاصله ی (nextseqnam,base+N-1) برای بسته هایی استفاده می شود که کم کم ارسال خواهند شد، فقط باید داده ها از لایه های بالایی ارسال شوند. و در آخر، اعداد ترتیبی بزرگتر یا مساوی base+N نمی توانند تا زمانی که بسته های بدون اعلام وصول در حال حاضر در لوله انتقال اعلام وصولشان نیاید ارسال شوند (بخصوص، بسته ی که عدد ترتیبی پایه (Base) را دارد.)

همان طور که در شکل ۱۹-۳ آمده است، تعداد اعداد ترتیبی مجاز برای بسته هایی ارسال شده اما هنوز اعلام وصول نشده با پنجره ای از سایز N از اعداد ترتیبی نمایش داده می شود هنگام عملکرد پروتکل، این پنجره در فضای ترتیب اعداد به جلو رانده می شود. به این دلیل، N همیشه به عنوان اندازه ی پنجره در نظر گرفته شود و پروتکل GBN هم به عنوان پروتکل پنجره در حال حرکت (sliding window) نامیده می شود. ممکن است بپرسید که ما چرا تعداد بسته های منتظر و اعلام وصول نشده را به عدد N در مرحله اول محدود می کنیم. چرا به تعداد نامحدودی از این بسته ها اجازه عبور نمی دهیم؟ در بخش ۳.۵ خواهیم دید که کنترل جریان یکی از دلایل قرار دادن محدودیت از سمت فرستنده است. و در بخش ۳.۷ دلیل دیگری را هنگام مطالعه تعارضات TCP بررسی خواهیم کرد.

www.it98.ir

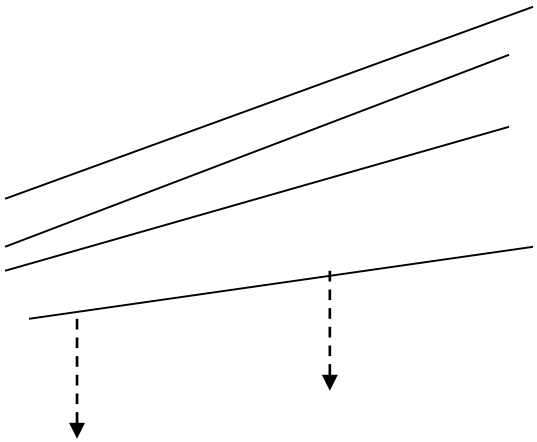
b عملیات (pipeline (خط لوله انتقال

شکل ۱۸-۳) ارسال توقف، انتظار و خط لوله انتقال

Go-Back-Nشکل ۱۹-۳) دیدگاه فرستنده از اعداد ترتیبی در

می رسد، بسته بعدی ارسال ACK

t=RTT+L/Rمی شود



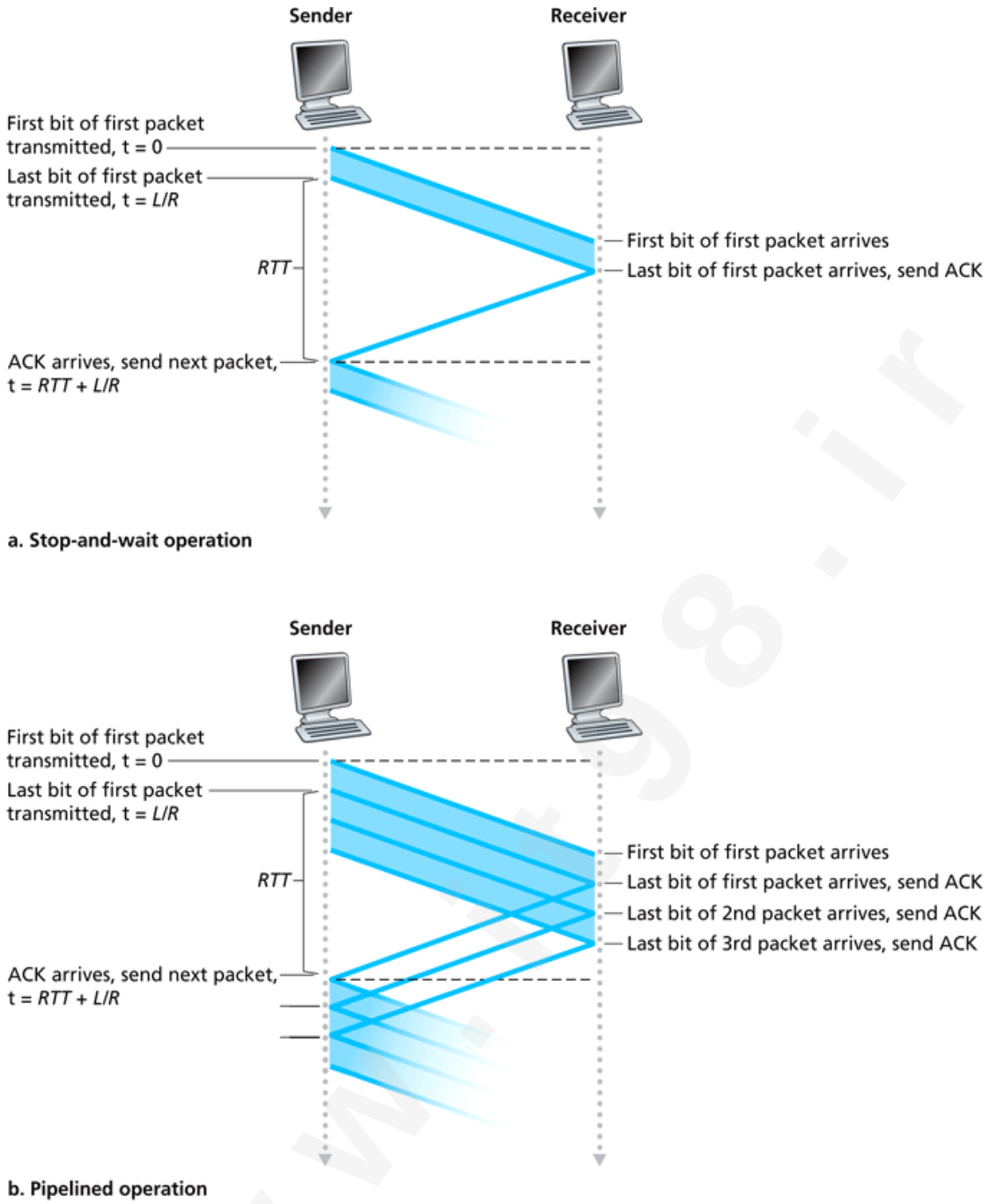


Figure 3.18 ♦ Stop-and-wait and pipelined sending

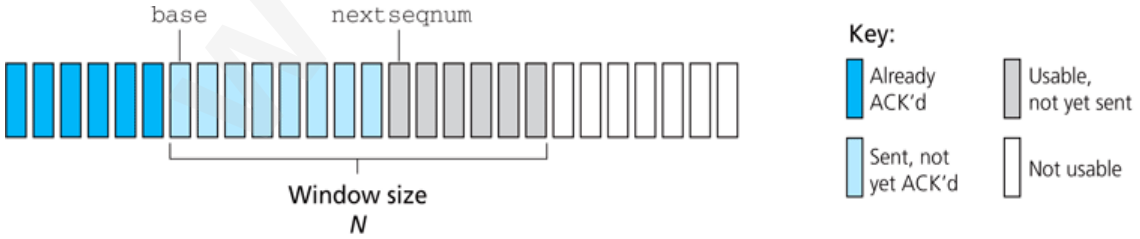


Figure 3.19 ♦ Sender's view of sequence numbers in Go-Back-N

www.it98.ir
wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

در عمل، عدد ترتیبی یک بسته در field با طول ثابتی در عنوان بسته، حمل می شود. اگر تعداد بیتهای موجود در فیلد عدد ترتیبی بسته K باشد تعداد اعداد ترتیبی [0/2k -1] خواهد بود. با تعداد محدودی از اعداد ترتیبی همه حسابها شامل اعداد ترتیبی همه حسابها شامل اعداد ترتیبی باید با استفاده از حساب پیمانه 2k باید محاسبه شوند (یعنی، فضای اعداد ترتیبی را می توان به صورت حلقه ی سایز 2k در نظر گرفت که در آن اعداد ترتیبی 2k -1 به سرعت با عدد ترتیبی صفر دنبال می شوند.) به یاد بیاورید که Rdt3 عدد ترتیبی یک بیتی دارد و ردیف اعداد ترتیبی از [0,1] در آن قرار می گیرند.

برخی از مشکلات در پایان این فصل نتایج تعداد محدود اعداد ترتیبی را مشخص میکنند در بخش ۵-۳ خواهیم دید که TCP فیلد عدد ترتیبی اش ۳۲ بیتی خواهد بود، و در اعداد ترتیبی TCP با بیتها در جریان بایستی شمرده می شوند و نه بسته ها.

شکل ۲۰-۳ و ۲۱-۳ توضیح اضافه ای از FSM در طرف فرستنده و گیرنده براساس NAK-Free ، ACK و پروتکل GBN ارائه می دهند. با این توضیح FSM را FSM گسترده می نامیم زیرا ما متغیر های افزوده ای (مثل متغیرهای زبان برنامه نویسی) برای nextseqnum, base داریم و علاوه بر این فعالیتهای افزوده ای بر روی این متغیرها و فعالیتهای شرطی شامل این متغیرها خواهیم داشت توجه کنید مشخص کردن FSM گسترده، اکنون مثل وارد کردن مشخصه ی زبان برنامه نویسی است [Bochman,1984] بوچمن پژهش برجسته ای از افزودنی ها به تکنیک FSM را مثل تکنیک های دیگر براساس زبان برنامه نویسی برای مشخص کردن پروتکل ها را ارائه می دهد

www.it98.ir

شکل ۲۰-۳) توضیح گسترده ی FSM از فرستنده GBN

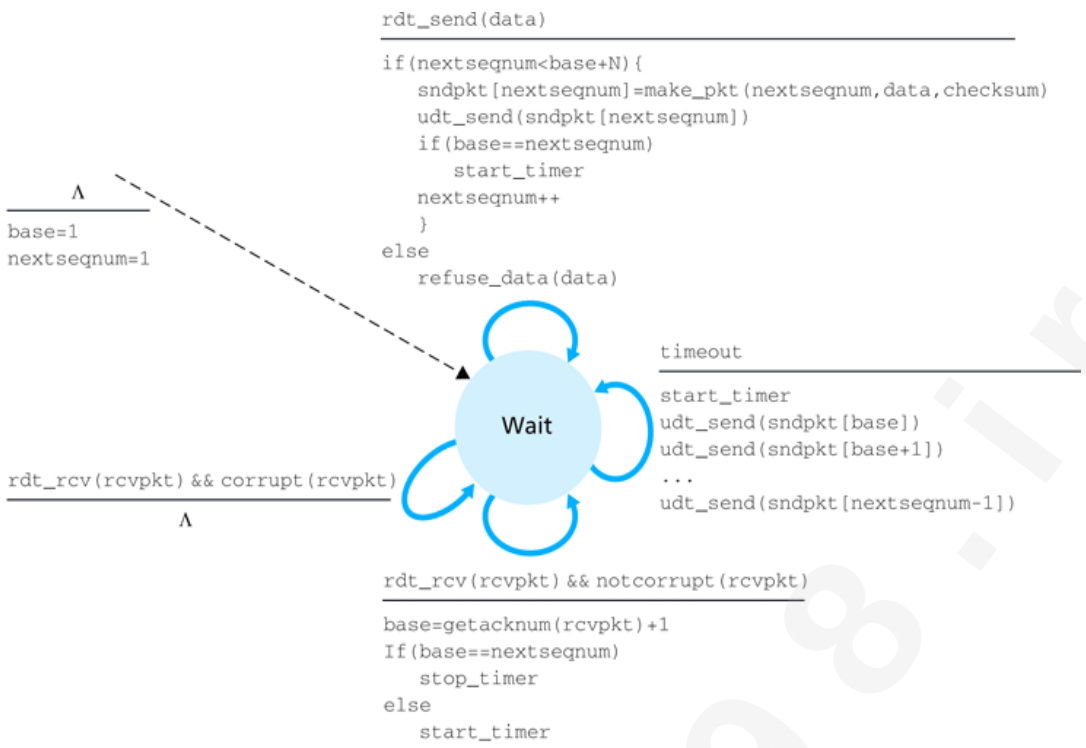


Figure 3.20 ♦ Extended FSM description of GBN sender

شکل ۲۱-۳) توضیح گسترده ی FSM از گیرنده ی GBN

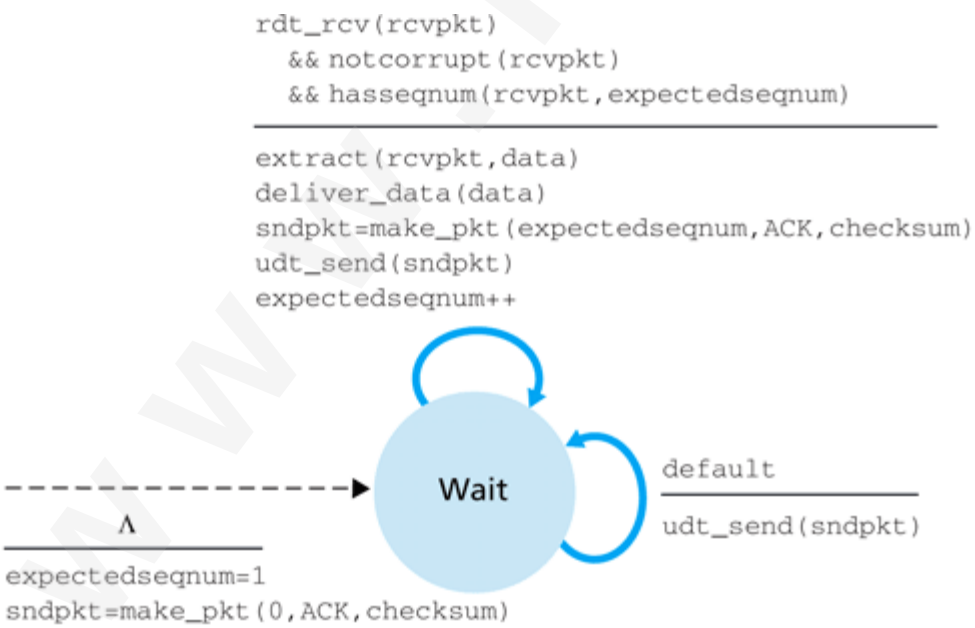


Figure 3.21 ♦ Extended FSM description of GBN receiver

www.it98.ir
wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

فرستنده ی GBN باید به سه نوع رخداد پاسخ دهد:

- درخواست از بالا: وقتی rdt_send( ) از بالا خواسته می شود، ابتدا فرستنده چک می کند که آیا پنجره پر است یا خالی یعنی، آیا N بسته اعلام وصول منتظر وجود دارند یا نه. اگر پنجره پر نباشد، یک بسته ایجاد شده و فرستاده می شود، و متغیرها به همان نسبت به روز می شوند. اگر پنجره پر باشد، فرستنده داده ها را به یک سطح بالاتر می فرستند ، که یک نشانگر ساده از پر بودن پنجره است. لایه ی بالاتر هم باید پس از آن بعدا دوباره تلاش کند. در حالت واقعی، فرستنده بیشتر تمایل دارد این داده ها را منتظر نگه دارد (ولی به سرعت نفرستد)، یا از مکانیزم هم ارزسازی استفاده کند (برای مثال، که باعث می شود لایه ی بالاتر اجازه داشته باشد تنها زمانی rdt-send ( ) را درخواست کند، که پنجره پر نباشد.

دریافت ACK در پروتکل GBN، اعلام وصول یک بسته با عدد ترتیبی n، اعلام وصل تجمعی خواهد بود، یعنی همه بسته ها با اعداد ترتیبی تا عدد n و خود عدد n به درستی به مقصد رسیده اند. ما خیلی کوتاه به این موضوع بر می گردیم وقتی که سمت گیرنده ی GBN را بررسی کردیم.

آگاهی از پایان وقت نام پروتکل Go-back-N از رفتار فرستنده در حالت وجود بسته های گمشده یا تاخیر دارد گرفته شده است. در آن مثل پروتکل توقف و انتظار: از یک زمان سنج برای بازیابی داده های گمشده یا بسته های اعلام وصول نشده استفاده می شود. اگر زمان پایان یابد، فرستنده همه بسته هایی که فرستاده را دوباره می فرستد البته اگر اعلام وصول آن بسته ها نرسیده باشد. فرستنده در شکل ۲۰-۳ تنها از زمان سنج استفاده می کند، که زمان سنج آخرین بسته فرستاده شده ولی تابحال اعلام وصول نشده است. اگر یک ACK برسد ولی باز هم بسته های ارسالی بدون اعلام وصول وجود داشته باشند، زمان سنج از دوباره شروع به کار می کند. اگر هیچ بسته ی باقی نمانده باشد که اعلام وصول نشده باشد، زمان سنج متوقف می شود.

عملکرد گیرنده هم در GBN ساده است. اگر بسته ی با عدد ترتیبی N برسد و در تریب خود قرار گرفته باشد (یعنی، آخرین داده های فرستاده شده به لایه بالاتر، داده های بسته ی باشد که عدد ترتیبی n-1 را داشته باشد) گیرنده ACK بسته n را می فرستد و بخشی از داده های بسته را به لایه بالاتر می فرستد. در دیگر موارد، گیرنده بسته را نگه می دارد و ACK آخرین بسته به ترتیب رسیده را دوباره ارسال می کند. و توجه کنید که بسته های ساده ای را در هر لحظه به سطح بالاتر فرستاده می شوند و اگر بسته K دریافت شده و ACK آن فرستاده شود، یعنی همه بسته ها با عدد ترتیبی کمتر از k تا بحال ACK نشان به مقصد رسیده است. بنابراین، استفاده از اعلام وصول تجمعی انتخابی طبیعی برای GBN است.

در پروتکل GBN گیرنده بسته های خارج از ترتیب را نگه می دارد هر پند به نظر احمقانه و اتلاف وقت می رسد که بسته درست دریافت شده (ولی خارج از نوبت) را نگه دارد ولی دلایلی برای این کار وجود دارد. به یاد بیاورید که گیرنده باید تحویلی داده ها را به لایه بالاتر به ترتیب بدهد. حال فرض کنید که بسته n مورد انتظار است اما بسته n+1 می رسد، از آنجا که داده ها باید به ترتیب تحویل شوند، فرستنده می تواند بسته n+1 را نگه دارد و سپس این بسته را پس از رسیدن و تحویل کردن بسته n، به سطح بالاتر بفرستد. اما اگر بسته n گم شود، هم این بسته و هم بسته n+1 از سمت فرستنده به دلیل قانون دوباره ارسالی GBN، دوباره ارسال می شوند. مزیت این رویکرد، ساده سازی در نگه داری بسته ها از سمت گیرنده است و فرستنده مجبور نیست هیچ بسته خارج از ترتیبی را نگه دارد. بنابراین، وقتی فرستنده باید مرزهای بالاتر و پایین تر پنجره و موقعیت Nextseqnum در این پنجره را هدایت کند، تنها بخشی از اطلاعات که گیرنده باید نگهداری کند: عدد ترتیبی بسته بعدی در ترتیب خود است. این مقدار در متغیر expectedseqnum نگه داری می شود که در ESM گیرنده در شکل ۲۱-۳ نشان داده شده است. البته، اشکال دور انداختن یک بسته درست به مقصد رسیده این است که در ارسال دوباره، آن بسته ممکن است گم شده یا از آن کپی برداری شود و پس از آن شاید یک ارسال دیگر دوباره نیاز باشد.

شکل ۲۲-۳ عملیات پروتکل GBN را با اندازه پنجره ی ۴ بسته نشان می دهد. بدلیل محدودیت ظرفیت پنجره، فرستنده بسته های ۰ تا ۳ را می فرستد اما باید پس از آن صبر کند تا یکی یا چند تا از اعلام وصول آن بسته ها برسد. هر وقت هر ACK

www.it98.ir

موفق (برای مثال ACK1, ACK0) دریافت شدند، پنجره باز می شود و فرستنده می تواند بسته جدید (PKt5, PKt4) را بفرستد. در سمت گیرنده، بسته ۲ گم شده است و سپس بسته های ۳ و ۴و ۵ خارج از ترتیب شناخته می شوند و کنار گذاشته می شوند.

پیش از بستن بحث مان درباره ی GBN، بد نیست بدانیم که کاربران پروتکل در میان توده ای از پروتکل های مختلف، ساختاری شبیه FSM گسترده که در شکل ۲۰-۳ دیدیم دارد. و این کاربرد به شکل روندهای مختلف است که کارهایی را در پاسخ به انواع رخدادهای مختلف که اتفاق می افتد، انجام می دهد. در برنامه نویسی براساس رخدادهای مختلف که اتفاق می افتد انجام می دهد. در برنامه نویسی براساس رخداد به روندهای مختلف خواسته شده (invoked) نامیده می شوند چه با روندهای دیگر در توده ی پروتکل ها و یا در نتیجه ی یک شکست در انجام کارها. از سمت فرستنده، این رخدادها می توانند (۱) درخواستی از سمت موجودیت لایه بالاتر برای احضار ( ) rdt – send باشد. (۲) شکست در کار زمان سنج باشد (۳) درخواستی از سمت لایه پایین برای احضار ( ) rdt-send وقتی که بسته ی دریافت می شود باشد. در انتهای این فصل تمرین های برنامه نویسی به شما شانس اعمال واقعی این روتین ها را در محیط شبکه ای واقعی شبیه سازی شده می دهد.

ما در اینجا نوشتیم که پروتکل GBN، تقریبا از تمامی تکنیک ها کمک می گیرد که ما آنها را هنگام مطالعه ی اجازی قابل اعتماد انتقال داده TCP در بخش ۵۰-۳ بیان خواهیم کرد. این متون شامل استفاده از اعداد ترتیبی، اعلام وصول تجمعی، Checksum ها و فعالیت اتمام وقت / دوباره ارسال، می شود.

۳,۴,۴ تکرار انتخابی (SR)

پروتکل GBN به فرستنده اجازه می دهد که تا جایی که می تواند لوله را از بسته ها پر کند و در شکل ۱۷-۳ نشان داده است، و به این روش از مشکلات پروتکل توقف انتظار که بحث کردیم جلوگیری کند. بهر حال، داستانهایی هم وجود دارند که GBN هم مشکلات مخصوص به خود را دارد. بخصوص، وقتی که سایز پنجره و تاخیر پهنای باند زیاد باشد، در این حالت ممکن است بسته های زیادی در لوله انتقال باشند. و یک خطا برای یک بسته ممکن است باعث دوباره ارسالی تعداد زیادی از بسته ها که خیلی از آنها لازم نیست بشود.

وقتی احتمال خطای کانال افزایش یابد، ممکن است لوله انتقال پر از دوباره انتقالی های نا لازم بشود. فرض کنید در مورد دیکته، پیام، اگر هر بار یک کلمه تحریف شود .... را کلمه ی اطراف آن (برای مثال اندازه ی پنجره ۱,۰۰۰ کلمه باشد) باید تکرار شوند. دیکته با تکرار همه ی کلمات بسیار کند می شود.

همان طور که اسم پروتکل تکرار انتخابی پیداست، این پروتکل ها از دوباره ارسال های نا لازم جلوگیری می کنند به این صورت که فرستنده فقط بسته هایی را می فرستد که گمان می کند از سمت گیرنده با اشکال مواجه شده اند (یعنی، گمشده یا خراب شده اند) برای این باز انتقالی تکی (که هر بار لازم باشد انجام می شود) لازم است که گیرنده برای تک تک بسته های درست به مقصد رسیده اعلام وصول کند. پنجره ای با سایز N بازهم برای محدود کردن تعداد بسته های در حال ارسال و اعلام وصول نشده در لوله انتقال لازم است.

بهر حال، برعکس GBN فرستنده تا بحال ACK بسته هایی را دریافت می کند که تاکنون رسیده اند. شکل ۲۲-۳ دیدگاه فرستنده ی SR را برای برخی بسته ها در پنجره نشان می دهد. شکل ۲۴-۳ کارهایی که فرستنده SR انجام می دهد را با جزئیات بیان می کند.

شکل ۲۲-۳) پروتکل Go-back-N در عمل

www.it98.ir

wWw.it98.ir

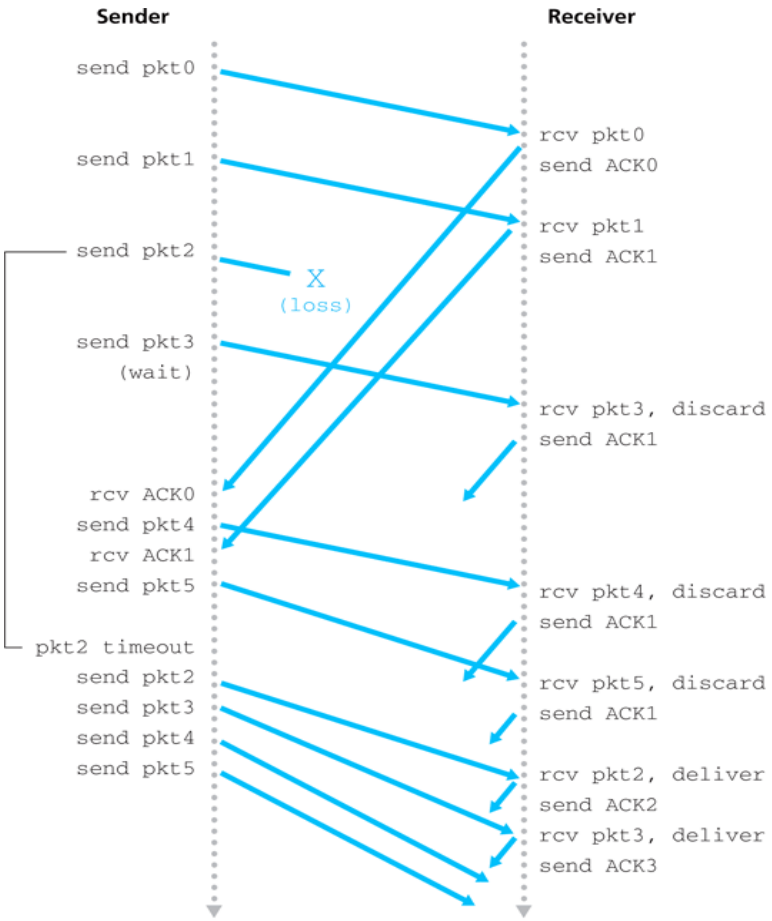


Figure 3.22 ♦ Go-Back-N in operation

www.it98.ir

شکل ۲۳-۳) پروتکل تکرار انتخابی (SR) و دیدگاه گیرنده و فرستنده از فضای اعداد ترتیبی

دیدگاه فرستنده از اعداد ترتیبی

دیدگاه گیرنده از اعداد ترتیبی

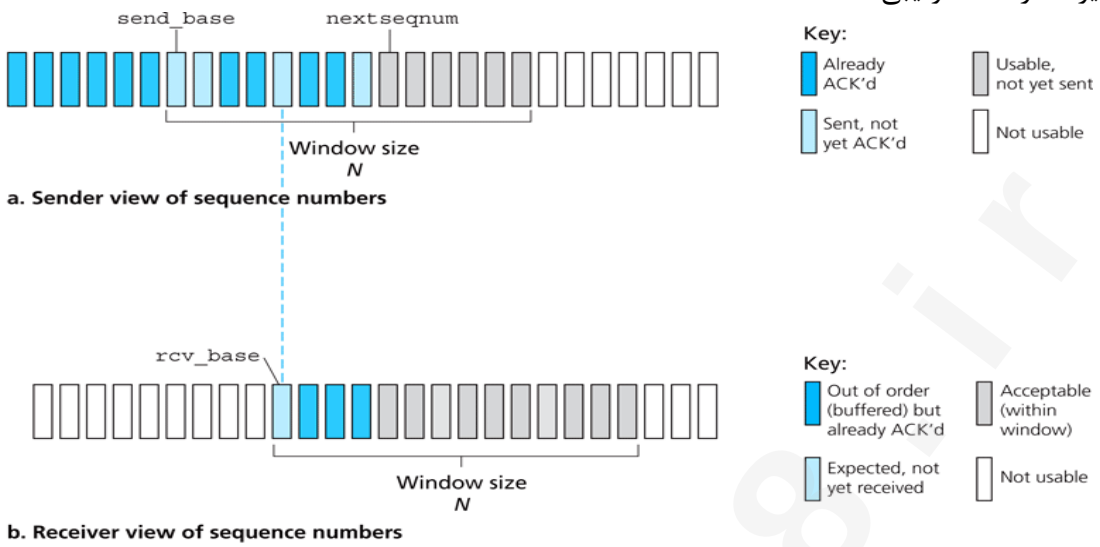


Figure 3.23 ♦ Selective-repeat (SR) sender and receiver views of sequence-number space

گیرنده SR اعلام وصول یک بسته سالم رسیده به مقصد را چه به ترتیب باشد و چه نه می فرستد. بسته های خارج از نوبت تا زمان که همه ی بسته های گمشده برسند نگهداری می شوند (یعنی، بسته هایی که اعداد ترتیبی پایین تر داشته اند) و در این نقطه، یک گروه از بسته ها به لایه بالاتر فرستاه می شوند تا اعلام تحویلی کنند. شکل ۲۵-۳ فعالیتهای مختلف گیرنده ی SR را نشان می دهد. توجه کنید که در شکل ۲۶-۳، گیرنده اول بسته های ۳و۴و۵ را نگهداری می کند و سپس آنها را با بسته ۲ وقتی بالاخره برسد، به لایه بالاتر می فرستد.

باید توجه کنیم که در مرحله ی ۲ در شکل ۲۵-۳، گیرنده دوباره اعلام وصول بسته هایی که رسیده بودند را با اعداد ترتیبی مشخص زیر حد پایه ی پنجره ی فعلی می فرستد (و آنرا نادیده نمی گیرد) و شما باید بدانید که این اعلام وصول دوباره لازم بوده است. با اعطای فضای اعداد ترتیبی در سمت گیرنده و فرستنده در شکل ۲۳-۳ برای مثال، اگر هیچ ACK برای بسته send-base که از سمت گیرنده به فرستنده منتشر شده است نرسد، فرستنده دوباره بسته send-base را می فرستد، هر چند مشخص است که (برای ما نه برای فرستنده) گیرنده آنرا گرفته است.

www.it98.ir

wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

اگر گیرنده از اعلام وصول این بسته خبری ندهد، پنجره ی فرستنده هرگز جلو نمی رود! این مثال یک جنبه ی مهم از پروتکل های SR (و بسیاری از دیگر پروتکل ها) را نشان می دهد. گیرنده و فرستنده دید کاملی از آنچه به درستی به مقصد رسیده و آنچه نرسیده ندارند برای پروتکل های SR، این به این معنی است که پنجره های گیرنده و فرستنده همیشه با هم مطابقت ندارند.

شکل ۲۴-۳) رخدادها و فعالیتهای فرستنده SR

۱. داده ها از بالا می رسند ، وقتی داده ای از بالا می رسد، فرستنده ی SR، عدد ترتیبی بعدی را برای بسته چک می کند. اگر عدد ترتیبی در پنجره ی فرستنده باشد، داه وارد بسته ها شده و فرستاده شده است. اگر نه، این داده یا نگه داشته شده یا به لایه ی بالاتر برگشته مثل GBN.

۲. اتمام وقت ، زمان سنج ها در اینجا هم برای حفاظت در برابر گم شدن بسته ها استفاده می شوند بهر حال، هر بسته باید ساعت منطقی خود را داشته باشد، زیرا فقط یک بسته در زمان اتمام وقت فرستاده می شود. از زمان سنج سخت افزاری می توان برای دنبال کردن چندین زمان سنج منطقی استفاده کرد. [Varghese 1997] .

۳. دریافت ACK ، اگر ACK دریافت شود، فرستنده ی SR آن بسته را رسیده قلمداد می کند و در پنجره هم آنرا مشخص می کند. اگر عدد ترتیبی بسته مساوی send-base باشد، پایه ی پنجره به بسته اعلام وصول نشده بعدی با کمترین عدد ترتیبی می رسد. اگر پنجره جلو برود و بسته های منتقل نشده با اعداد ترتیبی وجود داشته باشند که حالا به پنجره آمده اند، بالاخره این بسته ها هم منتقل می شوند.

شکل ۲۵-۳) رخدادها و وظایف گیرنده SR

بسته با عدد ترتیبی [rcv_base,rcv_base+n-1] به درستی دریافت می شود در این مورد، بسته دریافت شده بین پنجره ی گیرنده می ماند و یک بسته انتخابی ACK به فرستنده فرستاده می شود اگر بسته قبلا نرسیده باشد، نگهداری می شود. اگر بسته، عدد ترتیبی داشته باشد که پایه پنجره دریافتی مساوی باشد (مثل rcv_base در شکل ۲۲-۳) این بسته و هر چیزی که قبلا آماده نگه داشته شده است و اعداد پشت سر هم دارند (با rcv_base شروع می شوند) به لایه بالاتر فرستاده می شوند. پنجره دریافتی پس از آن با تعدا بسته هایی که به لایه بالاتر رفته اند جلوتر می رود. به عنوان مثال، شکل ۲۶-۳ را در نظر بگیرید، وقتی بسته ی با عدد ترتیبی rcv_base=2 دریافت می شود، این بسته و بسته های ۳و۴و۵ می توانند به لایه بعدی بروند.

بسته با عدد ترتیبی [rcv_base-N, rcv_base-1] به درستی دریافت می شود. در این حالت، یک ACK حتما باید ایجاد شود، هر چند که این بسته، بسته ی که گیرنده قبلا اعلام وصول آن را فرستاده است.

در غیر این دو حالت، بسته نادیده گرفته می شود.

شکل ۲۶-۳) عملیات SR

www.it98.ir

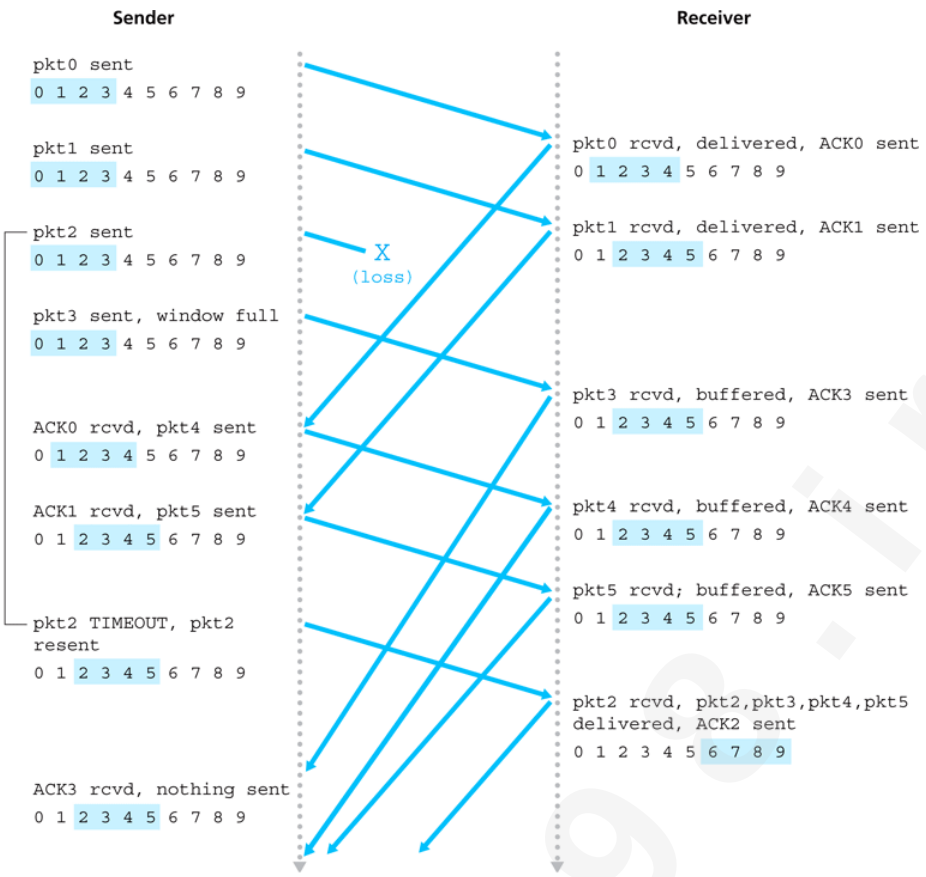


Figure 3.26 ♦ SR operation

فقدان همزمانی بین پنجره های گیرنده و فرستنده، نتایج مهمی هنگام مواجه با مورد واقعی و تعداد محدودی از اعداد ترتیبی در بر دارد. مثلا در نظر بگیرید چه می شود اگر ما تعداد مشخص ۴ بسته و اعداد ترتیبی ۰و۱و۲و۳ و اندازه ی پنجره ۳ داشته باشیم. در این نقطه، پنجره ی گیرنده از بسته های چهارم و پنجم و ششم پر می شود که اعداد ترتیبی ۳و۰ و یک را دارند. حالا دو حالت مختلف را در نظر بگیرید در مورد اول، که در شکل (a) ۲۷-۳ آمده است، ACK سه بسته اول گم می شوند و فرستنده دوباره سه بسته را می فرستد. گیرنده پس از آن بسته ی را با عدد ترتیبی صفر می گیرد که کپی اولین بسته ارسالی است.

در مورد دوم، که در شکل (b)۲۷-۳ آمده است، ACK سه بسته اول به درستی ارسال می شوند و سپس فرستنده پنجره خود را جلو برده و بسته های چهارم، پنجم و ششم را با اعداد ترتیبی ۳و۰ و یک می فرستد. بسته با عدد ترتیبی ۳ گم می شود ولی بسته با عدد ترتیبی صفر می رسد، بسته ی که داده ی جدید در خود دارد.

www.it98.ir

wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

شکل ۲۷-۳) مشکل گیرنده ی SR با پنجره های با اندازه بزرگ: یک بسته جدید یا ارسال هرباره؟

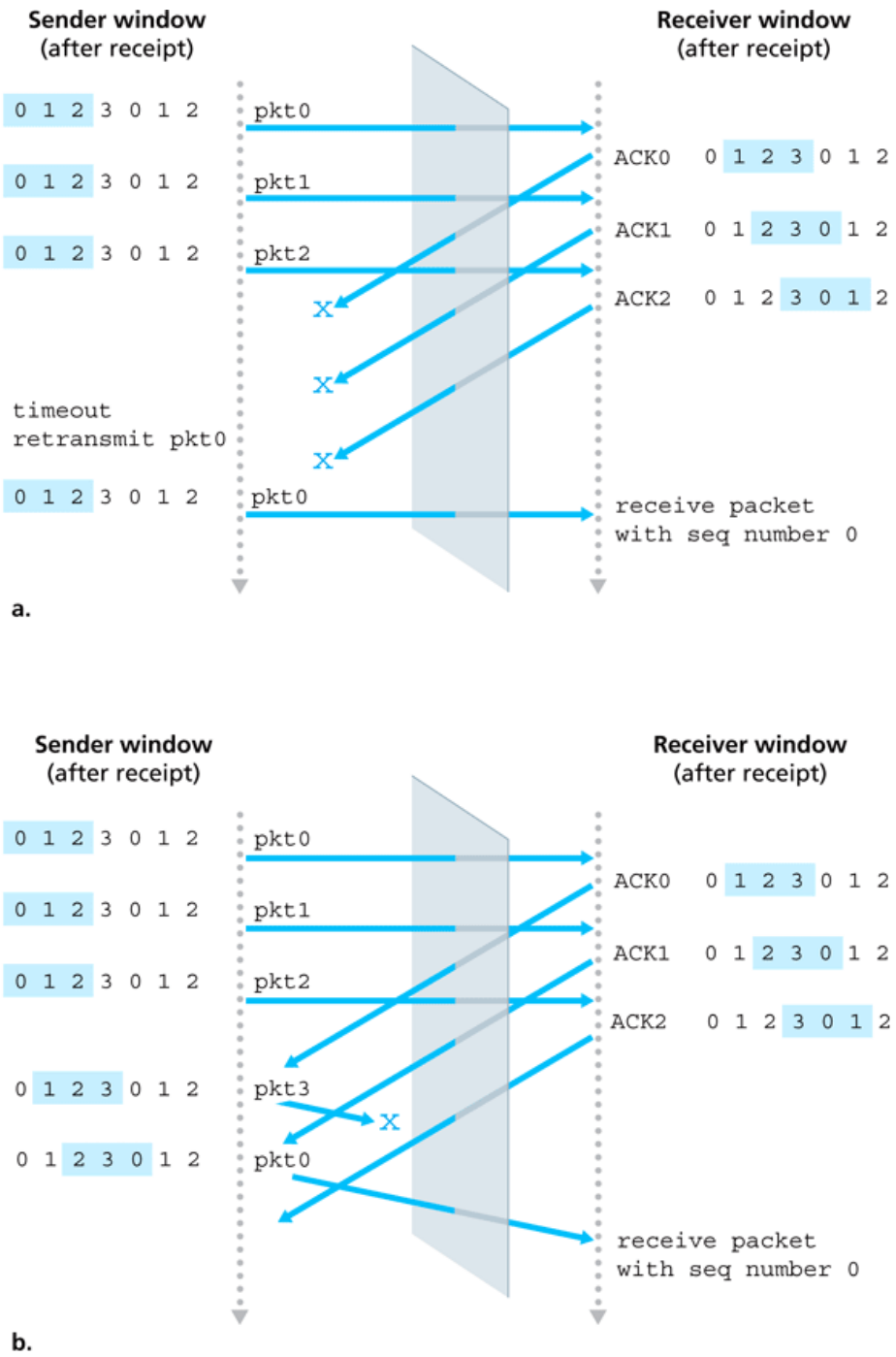


Figure 3.27 ♦ SR receiver dilemma with too-large windows: A new packet or a retransmission?

www.it98.ir

حالا دیدگاه گیرنده را در شکل ۲۷-۳ در نظر بگیرید، که انگار پرده ای بین فرستنده و گیرنده حائل است، وقتی که گیرنده نمی تواند فعالیتهای فرستنده را ببیند ، تنها چیزی که گیرنده می بیند ترتیب پیامهایی است که از کانال دریافت می کند و به کانال می فرستند. تا جایی که مهم است، این حالت در شکل ۲۷-۳ ممکن است رخ دهند. هیچ راه تشخیص دوباره ارسالی اولین بسته از ارسال اصلی پنجمین بسته وجود ندارد. واضح است که، با این تفاسیر اندازه ی پنجره ای با حتی یک عدد کمتر از تعداد اعداد ترتیبی درست کار نمی کند. ولی کوچکی اندازه پنجره چقدر باید باشد؟ یک مسئله در انتهای فصل از شما می خواهد تا نشان دهید که اندازه ی پنجره باید کمتر یا مساوی نصف اندازه فضای اعداد ترتیبی در پروتکل SR باشد.

در وب سایت معرفی شده، یک applet به شما معرفی می شود که فعالیت پروتکل SR را شبیه سازی کرده است. سعی کنید همین کار را که با GBN Applet انجام دادید برای SR هم انجام دهید.

این بحث ما را از پروتکل های انتقال داده تکمیل می کند. ما زمینه های زیادی را پوشش داده ایم و مکانیزم های مختلفی را معرفی کردیم که با هم انتقال مورد اطمینان داده را فراهم می کنند جدول۱-۳ این مکانیزم ها را توضیح می دهد. حالا که ما همه این مکانیزم ها را در عمل دیده ایم و می توانیم "تصویر بزرگ" راببینیم، از شما می خواهیم تا این بخش را دوباره بخوانید تا ببینید که چطور این مکانیزم ها کم کم به پوشش مدلهای پیچیده و (واقعی) ما اضافه شدند، مدلهایی از کانالی که فرستنده و گیرنده را به هم ارتباط می دهد و عملکرد پروتکل را بهبود می بخشد.

۳۲) بیاید از بحث مان درباره پروتکل های انتقال مورد اطمینان با در نظر گرفتن یک فرضیه دیگر در مدل کانال نتیجه بگیریم. بیاد آورید که ما فرض کردیم بسته ها نمی توانند درون کانال بین فرستنده گیرنده دوباره و دوباره وارد شوند. این به طور کل یک فرض منطقی است وقتی که فرستنده و گیرنده، تنها با یک سیم فیزیکی با هم در ارتباط باشند. بهر حال، وقتی که "کانال" ارتباطی دو طرف یک شبکه باشد، عملکرد ترتیب در بسته ها هم می تواند رخ دهد. یک نمونه از دوباره ترتیبی بسته ها این است که نمونه های قدیمی یک بسته با عدد ترتیبی یا اعلام وصول x می تواند دوباره پیدا شود. حتی اگر نه فرستنده و نه گیرنده x را در خود داشته باشند. با دوباره ترتیبی بسته ها می توان تصور کرد که کانال همه بسته ها را نگه می دادرد و خود به خود در هر نقطه از آینده آنها را می فرستد. از آنجا که اعداد ترتیبی ممکن است دوباره استفاده شوند، باید با دوباره ارسالی بسته ها مقابله کرد. رویکردی که در عمل استفاده می شود این است که مطمئن شویم یک عدد ترتیبی تا زمانی که فرستنده مطمئن شود که همه بسته های ارسالی شده قبلی با عدد ترتیبی x دیگر در شبکه وجود ندارند. این کار می توان با این فرض که بسته نمی تواند در شبکه برای بیشتر از مدت مشخصی زنده بماند انجام داد. حداکثر زمان زندگی یک بسته از تقریبا سه دقیقه در گسترش TCP ها در نظر گرفته می شود ....

مورد استفاده ،توضیحات ساز و کار (مکانیزم)

برای تشخیص بیت خطا در بسته منتقل شده استفاده می شود. (checksum) :خطایابی

آن) درون کانال گم شدهACK / ارسال مجدد یک بسته، شاید به این علت که بسته (یا جواب (timeout)تایمر: جهت پایان زمان

پیش از موعد)، یا زمانیtimeoutها می توانند زمانی که بسته تاخیر داشته باشد ولی گم نشده باشد، رخ دهد (timeoutباشد. زیرا

گیرنده به فرستنده از بین رفته باشد، کپی های یک بسته ممکن ACKکه یک بسته توسط دریافت کننده، دریافت شده باشد ولی جواب

است توسط گیرنده دریافت شود.

برای شماره بندی ترتیبی بسته های جریان داده فرستنده به گیرنده استفاده می شود. شکاف- Sequence):

wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

(Numberشماره ترتیب

ها در شماره های ترتیب بسته های دریافت شده به گیرنده امکان می دهد تا یک بسته گم شده را شناسایی کند. بسته ها با شماره های ترتیب تکثیر شده به گیرنده امکان شناسایی کپی های تکثیر شده یک بسته را می دهد.

به وسیله گیرنده جهت بیان اینکه یک بسته یا مجموعه ای از بسته ها به درستی دریافت شده اند،

به(Acknowledgment):تصدیق

به فرستنده صورت می پذیرد. تصدیق ها به طور نمونه شماره ترتیب بسته یا بسته هایی که تصدیق می شوند، حمل خواهند کرد. ممکن است تصدیق ها اختصاصی یا تجمعی باشند که بستگی به پروتکل دارد.

جواب منفی (تصدیق منفی): به وسیله گیرنده برای بیان اینکه بسته به طور صحیح دریافت نشده اس، به فرستنده ارسال می شود. جواب های منفی به طور نمونه شماره ترتیب بسته ای که به درستی دریافت نشده را حمل خواهد کرد.

پنجره، ایجادخط لوله: فرستنده ممکن است ارسال بسته هایی با شماره های ترتیب که در یک محدوده معین رخ می دهند، را محدود کند. از طریق اجازه بسته های چند تایی که منتقل شده اند اما هنوز تصدیق نشده اند، فرستنده می تواند حالت توقف و انتظار عملیات را افزاییش دهد. مشاهده خواهیم کرد که اندازه پنجره ممکن است بر اساس توانایی گیرنده برای دریافت و بافر کردن پیام ها، یا سطح ازدحام در شبکه، و یا هر دو، تنظیم شود.

جدول ۳,۱ : خلاصه سازوکارهای مطمئن انتقال داده و موارد استفاده از آن ها

یک متد برای استفاده از شماره های ترتیب به این ترتیب که از ثبت [RFC 1323] [Sunshine 1978].برای شبکه های سرعت بالا خطاها به طور کامل خودداری می کند را شرح می دهد.

۳,۵TCP انتقال اتصال گرا:

-همان پروتکل لایه انتقال اینترنت اتصال گرا وانتقال مطمئن TCPحال که ما اصول اساسی انتقال مطمئن داده را بررسی کرده ایم،به

مبتنی بر اصول اساسی زیادی که در TCPبازگردیم. در این بخش مشاهده خواهید کرد که به منظور فراهم آوردن انتقال مطمئن داده،

بخش قبل مورد بحث قرار گرفت، مانند شناسایی خطا، ارسال های مجدد، تصدیق های تجمعی، تایمرها و قسمت سرآیند (هدر) برای

تعریف می شود. RFC 2581و RFC 2018,RFC 1323, RFC 1122,RFC 793, درTCPشماره های ترتیب و تصدیق.

۳,۵,۱ اتصال TCP

گفته می شود که TCP اتصال گرا می باشد زیرا قبل از اینکه پردازش یک برنامه بتواند شروع گردد تا داده را به دیگری بفرستد، دو پروسس ابتدا باید با یکدیگر عمل دست تکانی (handshake) را انجام دهند—به عبارت دیگر آنها باید سگمنت های ابتدایی را به همدیگر بفرستند تا پارامترهای انتقال داده ی از پس آمده را ایجاد کنند. بعنوان قسمتی از تاسیسات اتصال TCP، هردو طرف کانکشن متغیرهای بخش TCP بسیاری که با کانکشن TCP مشترک شده اند (بسیاری که در این فصل و در بخش 7.3 بحث خواهند شد) را مقداردهی می کنند.

www.it98.ir

"اتصال" TCP یک جریان TDM یا FDM سرهم پیوسته همانند آنکه در یک شبکه ی راه گزیده (circuit-switched) می باشد، نیست. و نه حتی یک جریان مجازی (فصل ۱ را ببینید) مانند حالت اتصال که بطور کامل در سراسر سیستمهای دو طرفه مستقر می شوند، هم نیست. زیرا پروتکل TCP تنها در سیستمهای انتهایی (end systems) اجرا می شود و نه در عناصر شبکه ی میانی (همانند روترها و سوئیچهای لایه ی لینک)، عناصر شبکه ی میانی حالت اتصال TCP را نگه نمی دارند.

تاریخچه

وینتون سرف، رابرت کان، و TCP/IP

در اوایل دهه ی ۱۹۷۰، شبکه های packet-switched شروع به توسعه کردند، با آرپانت (ARPAnet) –منادی اینترنت- که یکی از بسیاری شبکه های دیگر می باشد. هر کدام از این شبکه ها پروتکل های خاص خود را داشت. دو محقق، وینتون سرف (Vinton Cerf) و رابرت کان (Robert Kahn) به اهمیت بهم پیوستن این شبکه ها پی بردند و یک پروتکل cross-network بنام TCP/IP اختراع نمود که داوطلب کنترل عبور و مرور پروتکل پروتکل/اینترنت بود. اگرچه سرف و کان با نگاه کردن به پروتکل به عنوان یک نهاد واحد شروع کردند، بعدها به دو قسمت تقسیم شد، TCP و IP، که بطور جداگانه فعالیت می کردند. سرف و کان یک مقاله در مورد TCP/IP در می ۱۹۷۴ در IEEE Transactions on Communications Technology منتشر کردند. [Cerf 1974]

پروتکل TCP/IP، که کره مربای اینترنت امروزی است! قبل از PCها و workstationها اختراع شده بود، قبل از ازدیاد اترنت ها (Ethernets) و دیگر تکنولوژیهای شبکه ی محلی، و قبل از وب، استریمهای صوت، و حتی چت. سرف و کان نیاز به یک پروتکل شبکه ای را دیدند که از یک طرف، پشتیبانی برای برنامه های هنوز تعریف نشده ایجاد کند و از طرف دیگر به هاستهای دلخواه و پروتکلهای لایه ی لینک اجازه عملیاتی شدن دهد.

در سال ۲۰۰۴ میلادی، سرف و کان جایزه ACM's Turing را دریافت نمودند که جایزه ی نوبل در رایانه بحساب می آید که بخاطر پیشگامی در گسترش شبکه، از قبیل طراحی و پیاده سازی پروتکل های ارتباطات پایه ای اینترنت، TCP/IP و به خاطر رهبری الهام بخش خود در شبکه سازی می باشد.

در واقع، روتر های میانی نسبت بهTCP کاملا بی توج هستند؛ آن ها دیتاگرام را می بینند نه اتصال ها را.

یک اتصالTCP سرویس کاملا دو طرفه را فراهم می آورد: اگر بین پروسس A روی یک میزبان و پروسس B روی میزبان دیگر اتصال TCP وجود داشته باشد، انگاه داده های لایه کاربرد می توانند از پروسس A به پروسس B سبه طور همزمان، همچنان که جریان های داده لایه کاربرد از پروسس B به پروسس A، جریان می یابند. همچنین اتصال TCP همیشه نقطه به نقطه است که بین یک نک فرستنده و یک تک گیرنده می باشد. بنابراین چند قالبی (Multicasting) نامیده می شود (در بخش ۴,۷ ملاحظه شود)-انتقال داده از یک فرستنده به تعدادی زیادی گیرندهدر یک تک عملیات- ممکن نیست با TCP انجام شود. با TCP با دو میزبان مانند جمعیت کم است و سه میزبان مثل انبوه مردم!

اجازه دهید اکنون نگاهی به چگونگی ایجاد یک اتصال TCP بیندازیم. یک پروسس در حال اجرا در یک ماشین میزبان که می خواهد یک اتصال با پروسس دیگر روی میزبان دیگر را راه بیندازد، در نظر بگیرید. یادآوری می شود که پروسسی که اتصال را راه

www.it98.ir

wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

اندازی می کند، پروسس مشتری نامیده می شود، در حالی که پروسس دیگر، پروسس سرویس دهنده نام دارد. ابتدا پروسس برنامه کاربردی مشتری به لایه انتقال مشتری اطلاع می دهد که می خواهد یک اتصال با یک پروسس در سرویس دهنده، برقرار کند. از بخش ۲,۷ به یاد آورید که، یک برنامه جاوای مشتری این عمل را با صدور این دستور انجام می دهد:

Socket clientSocket = new Socket (“hostname”, portNumber);

که hostname نام سرویس دهنده و portNumber پروسس روی سرویس دهنده را شناسایی می کند. سپس لایه انتقال روی مشتری اقدام به برقراری یک اتصال TCP با TCP سرویس دهنده می کند.در انتهای این بخش درباره بعضی جزئیات پروسس برقراری اتصال بحث می کنیم. برای الآن کفایت می کند که بدانیم مشتری در ابتدا یک سگمنت TCP مخصوص را ارسال می کند؛ سرویس دهنده با یک سگمنت TCP مخصوص ثانیه ای پاسخ می دهد؛ و در نهایت مشتری دوباره با سگمنت مخصوص سوم پاسخ می دهد. دو سگمنت ابتدایی هیچ باری را منتقل نمی کنند، هیچ داده لایه کاربردی وجود ندارد؛ سومین سگمنت ممکن است یک بار مفید حمل کند. زیرا سه سگمنت بین دو میزبان فرستاده می شود، این پروسس برقراری اتصال گاهی به عنوان دست تکانی سه مرحله ای شناخته می شود. یک بار یک اتصال TCP برقرار می شود، دو پروسس برنامه کاربردی داده ها را برای هم می توانند ارسال کنند. اجازه دهید ارسال داده از پروسس مشتری به پروسس سرویس دهنده را مورد بررسی قرار دهیم. پروسس مشتری یک جریان داده را به سوی سوکت (در پروسس) عبور می دهد، همانگونه که در بخش ۲,۷ توضیح داده شده، یک مرتبه داده ها به سوی در هدایت می شوند که اکنون در دستان TCP در حال اجرا روی مشتری می باشد. همانطور که در شکل ۳,۲۸ نشان داده می شود. TCP این داده ها را به بافر ارسال اتصال، که یکی از بافر هایی است که در طول عملیات ابتدایی دست تکانی سه مرحله ای تخصیص داده شده، به پیش می برد. لحظه به لحظه TCP دسته هایی از داده ها را از بافر اتصال می گیرد. با طور قابل توجه خصوصیات TCP [RFC 793] نسبت به تعیین کردن زمانی که TCP واقعا باید داده های بافر شده را ارسال کند، بی تفاوت است، چگونگی این که TCP باید "آن داده های سگمنت ها را در زمان آسودگی خود ارسال کند." حداکثر مقدار داده ای که می تواند گرفته شود و در یک سگمنت قرار داده شودتوسط حداکثر حجم سگمنت (MSS)، محدود می شود. MSS به طور نمونه طول بزرگترین فریم لایه پیوند را با اولین تصمیم گیری، تنظیم می کند که می تواند ارسال شود توسط .....

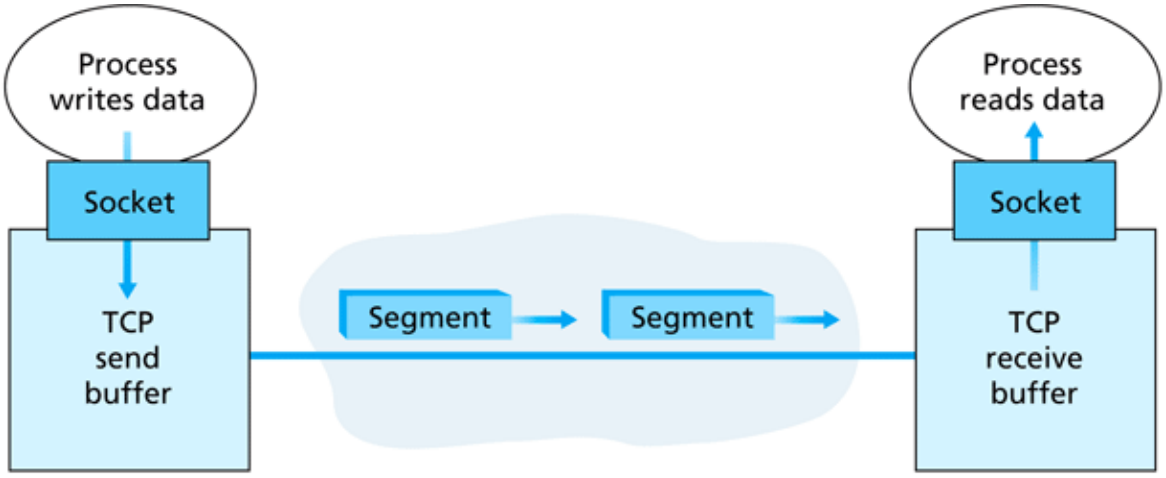


Figure 3.28 ♦ TCP send and receive buffers

و سپس MMS را تنظیم می کنیم تا مطمئن شویم که سگمنت TCP، مناسب فریم لایه ارتباط یک طرفه است. معمول ترین داده ها برای MTU، 1450 بایت، 536 بایت و 512 بایت هستند. روشهایی هم برای کشف مسیر MTU – بزرگترین فریم

www.it98.ir

لایه ارتباط که می توان آنرا از طریق همه لینکها از مبدا تا مقصد فرستاد- و تنظیم MMS بر اساس داده مسیر MTU پیشنهاد شده اند. توجه داشته باشید که MMS ، حداکثر مقدار اطلاعات لایه application در سگمنت است نه حداکثر اندازه سگمنت TCP از جمله هدرها.

TCP مقداری از اطلاعات مشتری را با یک هدر TCP جفت می کند تا سگمنتهای TCP شکل بگیرند. سگمنتها به لایه شبکه انتقال می یابند و در آنجا بطور جداگانه درون دیتاگرام های IP لایه شبکه قرار می گیرند. سپس دیتاگرام های IP به شبکه فرستاده می شوند. همانطور که در شکل 28-3 نشان داده شده وقتی TCP یک سگمنت را در انتهای دیگر دریافت می کند، اطلاعات سگمنت در بافر دریافت TCP قرار می گیرند. Application ، جریان اطلاعات را از این بافر می خواند. هر سمت از ارتباط دارای بافر ارسال و دریافت خاص خود است.(می توانید اپلت کنترل جریان آنلاین را در آدرس hhttp://www.awl.com/kurose-ross مشاهده کنید که تصاویر متحرکی از بافرهای ارسال و دریافت را نشان می دهد.)

با توجه به این مبحث می توانید ببینید که یک ارتباط TCP شامل بافرها، متغیرها و یک سوکت اتصال برای یک پردازش در یک هاست و مجموعه ای دیگر از بافرها، متغیرها و یک سوکت اتصال برای یک پردازش در هاست دیگری است. همانطور که قبلا ذکر شد در عناصر شبکه بین هاست ها هیچ بافر یا متغیری به ارتباط (connection) تخصیص داده نمی شود.

3.5.2 : ساختار سگمنت TCP

بعد از نگاهی مختصر به ارتباط TCP به بررسی ساختار سگمنت TCP می پردازیم. سگمنت TCP شامل فیلدهای هدر و یک فیلد اطلاعات است. فیلد اطلاعات شامل مقداری از اطلاعات application است. همانطور که در بالا ذکر شد MMS ، حداکثر اندازه فیلد اطلاعاتی سگمنت را محدود می کند. وقتی TCP فایل بزرگی مثل یک تصویر را بعنوان بخشی از یک صفحه وب ارسال می کند، فایل را به قطعاتی در اندازه MMS می شکند(بجز آخرین قطعه که اغلب کمتر از MMS است).

application های متقابل اغلب قطعات اطلاعاتی را منتقل می کنند که کوچکتر از MMS هستند. مثلا در مورد applicationهای ارتباط از راه دور مثل Telnet ، فیلد اطلاعات در سگمنت TCP اغلب تنها یک بیت است. چون معمولا هدر TCP 20 بایتی است(12 بایت بیشتر از هدر UDP) ، ممکن است سگمنتهای ارسال شده توسط Telnet فقط 21 بایت طول داشته باشند.

شکل 29-3 ساختار سگمنت TCP را نشان می دهد. مثل UDP ، در اینجا هم هدر شامل شماره پورت های مبدا و مقصد است که برای ترکیب/تقسیم اطلاعات از/به application های لایه بالایی استفاده می شوند. همچنین همانند UDP ، هدر شامل یک فیلد checksum نیز هست. هدر سگمنت TCP شامل فیلدهای زیر نیز هست:

www.it98.ir

wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

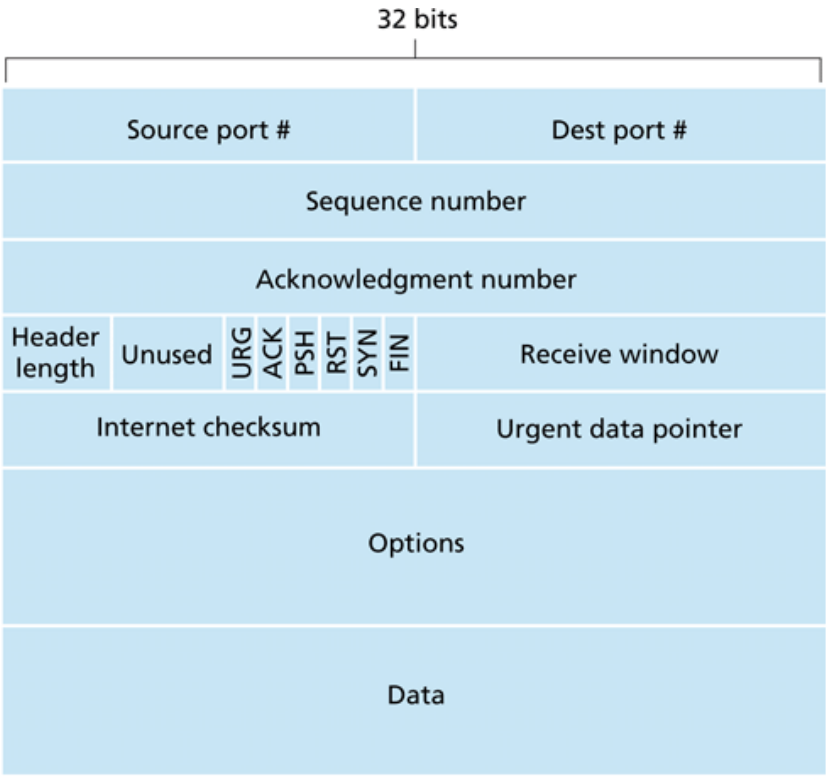


Figure 3.29 ♦ TCP segment structure

فیلد عدد توالی 32 بیتی و فیلد عدد تائید 32 بیتی که توسط فرستنده و گیرنده TCP برای اجرای یک سرویس مطمئن اطلاعات مورد استفاده قرار می گیرند و در زیر مورد بحث قرار می گیرد.

فیلد پنجره دریافت(16 (receive window بیتی که برای کنترل جریان استفاده می شود. به اختصار خواهیم دید که برای نشان دادن شماره بایت هایی که گیرنده مایل به پذیرش است مورد استفاده قرار می گیرد.

فیلد طول هدر(4 (header length بیتی که طول هدر TCP را در قالب کلمات 32 بیتی تعیین می کند. هدر TCP می تواند بسته به فیلد گزینه های TCP دارای طولی متغیر باشد(معمولا فیلد options خالی است بنابراین طول هدر TCP عمدتا 20 بایت است).

فیلد گزینه های (options) اختیاری و با طول متغیر که یا وقتی فرستنده و گیرنده حداکثر اندازه سگمنت(MMS) را مبادله می کند مورد استفاده قرار می گیرد یا بعنوان یک فاکتور اندازه گیری در شبکه های با سرعت بالا استفاده می شود. یک گزینه time-stamping هم تعریف شده است. برای جزئیات بیشتر به RFC854 و RFC1323 مراجعه کنید.

فیلد flag 6 بیتی است. از بیت ACK استفاده می شود تا نشان دهیم که داده انتقال یافته در فیلد تائید معتبر است یعنی سگمنت، سگمنتی که با موفقیت دریافت شده را تائید می کند. همانطور که در پایان این بخش بحث خواهیم کرد بیت های RST، SYN و FIN برای برقراری و قطع ارتباط مورد استفاده قرار می گیرند. تنظیم بیت PSH نشان می دهد که گیرنده باید اطلاعات را فورا به لایه بالایی ارسال کند. در نهایت بیت URG نشان دهنده این مطلب است که اطلاعاتی در این سگمنت وجود دارد که قسمت ارسال لایه بالایی آنرا به عنوان یک ضرورت "urgent" می شناسد. موقعیت آخرین بایت از این اطلاعات ضروری توسط فیلد نشانگر اطلاعات ضروری(urgeny data pointer) نشان داده می شود. وقتی اطلاعات ضروری وجود دارد، TCP باید آنرا به اطلاع قسمت دریافت لایه بالایی برساند و این اطلاعات را به نشانگر واقع در انتهای اطلاعات ضروری ارسال کند(در عمل PSH، URG و نشانگر اطلاعات ضروری استفاده نمی شوند ما فقط برای تکمیل مبحث فیلدها آنها را ذکر

www.it98.ir

کردیم).

اعداد توالی و تائید

دو تا از مهمترین فیلدها در هدر سگمنت TCP ، فیلد عدد توالی و فیلد عدد تائید هستند. این فیلدها بخش حیاتی سرویس مطمئن انتقال اطلاعات به شمار می آیند. اما قبل از اینکه به چگونگی استفاده از این فیلدها برای انتقال مطمئن اطلاعات بپردازیم، اجازه دهید اول توضیح دهیم که TCP دقیقا چه چیزی را درون این فیلدها قرار می دهد.

TCP ، اطلاعات را بصورت جریانی از بایتهای غیر ساختارمند اما منظم نمایش می دهد. استفاده TCP از اعداد توالی انعکاس دهنده این دیدگاه است که اعداد توالی روی جریانی از بایت های انتقالی قرار دارند نه روی مجموعه ای از سگمنت های انتقالی. بنابراین عدد توالی یک سگمنت، عدد گروه بایتی(byte-stream) اولین بایت در سگمنت است. فرض کنید که یک فرآیند در هاست A می خواهد جریانی از اطلاعات را از طریق یک ارتباط TCP به یک فرآیند در هاست B بفرستد. TCP در هاست A به طور ضمنی هر بایت را در جریان اطلاعاتی عددگذاری می کند. فرض کنید که جریان اطلاعاتی شامل یک فایل 500000 بایتی باشد، MMS 1000 بایت باشد و اولین بایت این جریان با عدد 0 مشخص شده باشد. به اولین سگمنت، عدد توالی 0 تعلق می گیرد، دومین سگمنت عدد توالی 1000، سومی 2000 و غیره. هر عدد توالی در فیلد عدد توالی واقع در هدر سگمنت مناسب TCP قرار می گیرد.

حالا به بررسی اعداد تائید می پردازیم. این اعداد کمی نیرنگ بازتر از اعداد توالی هستند. به یاد دارید که TCP کاملا دو طرفه است بنابراین ممکن است هاست A همزمان با ارسال اطلاعات به هاست B ، اطلاعاتی را از هاست B دریافت کند. هر یک از سگمنت های دریافتی از هاست B دارای یک عدد توالی برای جریان دهی اطلاعات از هاست B به A هستند. عدد تائیدی که هاست A در سگمنت خود قرار می دهد، عدد توالی بایت بعدی است که هاست A از هاست B انتظار دارد. خوب است که برای درک اینکه چه اتفاقی در اینجا می افتد چند مثال بزنیم. فرض کنید که هاست A همه بایت هایی با عدد 0 را از طریق بایت 535 از هاست B دریافت کرده و یک سگمنت را به هاست B ارسال کرده است. هاست A منتظر بایت 536 و همه بایت های متعاقب آن در جریان اطلاعاتی هاست B می ماند. بنابراین هاست A، 536 را در فیلد عدد تائید سگمنت ارسالی خود به B قرار می دهد.

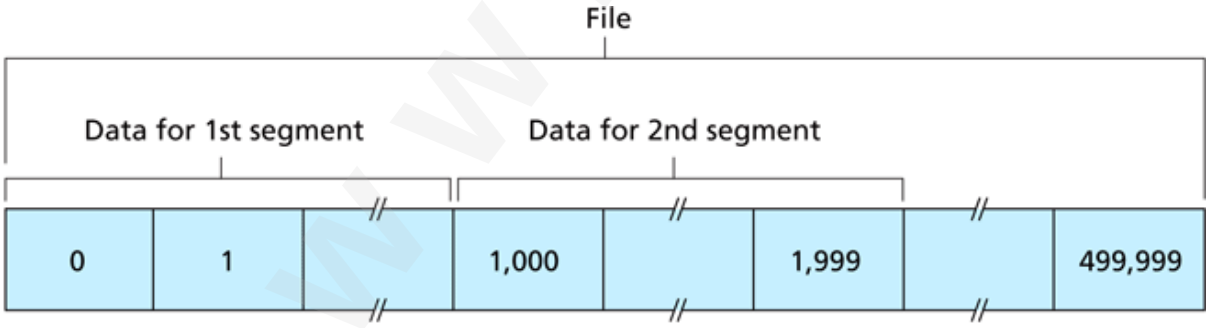


Figure 3.30 ♦ Dividing file data into TCP segments

در یک مثال دیگر، فرض کنید که هاست A یک سگمنت شامل بایت های 0 را توسط بایت 535 و سگمنت دیگری شامل بایت های 900 را توسط بایت 1000 از هاست B دریافت کرده است. به دلایلی هاست A هنوز بایت های 536 را توسط بایت 899 دریافت نکرده است. در این مثال هاست A هنوز منتظر بایت 536 است تا جریان اطلاعاتی B را دوباره ایجاد کند. بنابراین سگمنت بعدی A به B شامل عدد 536 در فیلد تائید خواهد بود. چون TCP فقط بایت هایی تا اولین بایت مفقوده در جریان

www.it98.ir
wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

را تائید می کند گفته می شود که TCP تائیدات متراکم (تجمعی) را فراهم می آورد.

مثال آخر شامل یک نکته مهم اما ظریف است. هاست A قبل از دریافت سگمنت دوم(بایت های 536 از طریق 899) سومین سگمنت را دریافت می کند(بایت های 900 از طریق 1000). بنابراین سگمنت سوم نامنظم دریافت می شود. نکته ظریف این است که : یک هاست وقتی سگمنت های نامنظم را در یک ارتباط TCP دریافت می کند باید چکار کند؟ جالب توجه اینکه TCP RFC ها هیچ قانونی را در اینجا تحمیل نمی کنند و تصمیم گیری را به افرادی واگذار می کنند که کار برنامه ریزی برای یک اجرای TCP را انجام می دهند. اساسا دو انتخاب جلوی روی ماست: 1- گیرنده فورا سگمنت های نامنظم را رد کند(همانطور که قبلا بحث کردیم این امر کار طراحی گیرنده را آسانتر می کند). 2- گیرنده بایت های نامنظم را نگه داشته و منتظر بایت های مفقوده برای پر کردن شکاف ها بماند. واضح است که انتخاب دوم بر حسب پهنای باند شبکه کارایی بیشتری دارد و روشی است که در عمل مورد استفاده قرار می گیرد.

در شکل 3-30 فرض کرده ایم که عدد توالی اولیه صفر بوده است. در واقعیت هر دو سمت یک ارتباط TCP بطور تصادفی یک عدد توالی را انتخاب می کنند. دلیل این کار این است که ممکن است سگمنت باقیمانده در شبکه از ارتباط تمام شده قبلی بین دو هاست اشتباهی به جای سگمنت معتبر در ارتباط بعدی بین این دو هاست گرفته شود و با انجام این کار احتمال وقوع این مسئله به حداقل می رسد.

Telnet : یک مورد مطالعاتی برای اعداد توالی و تائید

Telnet که در RFC854 تعریف شده است، یک پروتکل لایه کاربردی معروف است که برای ارتباط از راه دور استفاده می شود. از طریق TCP کار می کند و طوری طراحی شده که بین هر جفت از هاست ها کار کند. بر خلاف application های انتقال حجیم اطلاعات که در فصل 2 مورد بررسی قرار گرفت، Telnet یک application متقابل است. در اینجا نمونه ای از یک Telnet را بررسی می کنیم که اعداد توالی و تائید را نشان می دهد. می دانیم که اکنون بیشتر کاربران ترجیح می دهند به جای Telnet از پروتکل ssh استفاده کنند چون اطلاعات ارسال شده در ارتباط Telnet رمزدار نیستند که این امر باعث می شود Telnet در برابر حملات استراق سمع آسیب پذیر باشد(همانطور که در بخش 8.7 بحث شد).

فرض کنید که هاست A یک جلسه Telnet را با هاست B راه بیاندازد. چون هاست A شروع کننده جلسه است تحت عنوان مشتری شناخته می شود و هاست B سرور نام می گیرد. هر کاراکتر تایپ شده توسط کاربر(در مشتری) به هاست ریموت ارسال شده و هاست ریموت یک کپی از هر کاراکتر را مجددا بر می گرداند که روی صفحه کاربر نشان داده می شود. این انعکاس به این منظور است که مطمئن شویم کاراکترهای ارسال شده توسط کاربر Telnet ، دریافت شده اند و در سمت ریموت پردازش شده اند. بنابراین هر کاراکتر از زمانی که کاربر کلید را فشار می دهد تا زمانی که روی مانیتور کاربر نشان داده می شود دو بار در شبکه گردش می کند.

حالا فرض کنید که کاربر حرف "C" را تایپ می کند و سپس یک قهوه می نوشد. بگذارید سگمنت های TCP ارسال شده بین مشتری و سرور را بررسی کنیم. همانطور که در شکل 3-31 نشان داده شده فرض می کنیم که اعداد توالی اولیه برای مشتری و سرور به ترتیب 42 و 79 هستند. به یاد دارید که عدد توالی یک سگمنت برابر است با عدد توالی اولین بایت در فیلد اطلاعاتی. بنابراین اولین سگمنت ارسال شده از مشتری دارای عدد توالی 42 است و سگمنت ارسالی سرور حاوی عدد توالی 79 است. همچنین می دانیم که عدد تائید، عدد توالی بایت بعدی است که هاست منتظر آن است. بعد از برقراری ارتباط TCP و قبل از ارسال اطلاعات، مشتری منتظر بایت 79 است و سرور منتظر بایت 42.

با توجه به شکل 3-31 سه سگمنت ارسال شده اند. اولین سگمنت از مشتری به سرور ارسال شده که حاوی ASCII یک بایتی برای نمایش حرف "C" در فیلد اطلاعاتی خود است. این اولین سگمنت دارای عدد توالی 42 در فیلد عدد توالی خود است. همچنین، چون مشتری هنوز هیچ اطلاعاتی از سرور دریافت نکرده عدد تائید آن 79 است.

www.it98.ir

سگمنت دوم از سرور به مشتری ارسال می شود. این کار یک هدف دو جانبه دارد. اول به تائید اطلاعات دریافتی توسط سرور می پردازد. با قرار دادن عدد 43 در فیلد تائید خود، سرور به مشتری می گوید که همه اطلاعات را از طریق بایت 42 با موفقیت دریافت کرده و منتظر بایت 43 است. هدف دوم این سگمنت انعکاس حرف "C" است. بنابراین سگمنت دوم، ASCII نمایش حرف "C" را در فیلد اطلاعات خود قرار می دهد. این سگمنت دارای عدد توالی 79 است، عدد توالی اولیه مربوط به شارش جریان اطلاعات از سرور به مشتری در این ارتباط TCP . چون این اولین بایت اطلاعاتی است که سرور ارسال می کند، توجه کنید که تائید اطلاعات مشتری به سرور توسط سگمنتی صورت می گیرد که حاوی اطلاعات سرور به مشتری است. گفته می شود که این تائید مثل یک کیسه حمل بار(piggy backed) روی سگمنت اطلاعاتی سرور به مشتری است.

لایه انتقال

Mechanism	Use, Comments
Checksum	Used to detect bit errors in a transmitted packet.
Timer	Used to timeout/retransmit a packet, possibly because the packet (or its ACK) was lost within the channel. Because timeouts can occur when a packet is delayed but not lost (premature timeout), or when a packet has been received by the receiver but the receiver-to-sender ACK has been lost, duplicate copies of a packet may be received by a receiver.
Sequence number	Used for sequential numbering of packets of data flowing from sender to receiver. Gaps in the sequence numbers of received packets allow the receiver to detect a lost packet. Packets with duplicate sequence numbers allow the receiver to detect duplicate copies of a packet.
Acknowledgment	Used by the receiver to tell the sender that a packet or set of packets has been received correctly. Acknowledgments will typically carry the sequence number of the packet or packets being acknowledged. Acknowledgments may be individual or cumulative, depending on the protocol.
Negative acknowledgment	Used by the receiver to tell the sender that a packet has not been received correctly. Negative acknowledgments will typically carry the sequence number of the packet that was not received correctly.
Window, pipelining	The sender may be restricted to sending only packets with sequence numbers that fall within a given range. By allowing multiple packets to be transmitted but not yet acknowledged, sender utilization can be increased over a stop-and-wait mode of operation. We'll see shortly that the window size may be set on the basis of the receiver's ability to receive and buffer messages, or the level of congestion in the network, or both.

Table 3.1 ♦ Summary of reliable data transfer mechanisms and their use

تفسیر و کاربرد

Checksum : برای یافتن اشتباه در یک بسته فرستاده شده استفاده می شود .

TIMER:

www.it98.ir

wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

برای وقفه / دوباره فرستادن یک بسته استفاده می شود که احتمالا به خاطر اینکه بسته(یا (ACK داخل کانال انتقال گم شده است یابه خاطرپایان زمان انتظار وقفه ها, زمانی که بسته دچار تاخیر شده ولی گم نشده(وقفه نارس) یا زمانی که بسته به وسیله دریافت کننده دریافت شده است ولی ACK دریافت کننده به فرستنده گم شده است، می توانند اتفاق بیفتند، کپی های المثنی یک بسته به وسیله دریافت کننده ممکن است دریافت شوند.

Seguence number:

برای شماره گذاری ترکیبی بسته های اطلاعات جاری از فرستنده به دریافت شده استفاده می شود.فاصله های متوالی یا رشته اعداد بسته های دریافت شده به دریافت کننده اجازه می دهد که یک بسته گمشده را پیدا کند. بسته هایی بارشته اعداد المثنی به دریافت شده اجازه می دهند تا یک کپی المثنی را از یک بسته تشخیص دهد.

Acknowledgment:

جهت اعلام به فرستنده توسط دریافت کننده استفاده می شود تا اعلام کند یک بسته یا دسته های یک بسته درست دریافت شده است. علائم وصول نوعا حاصل توالی اعداد بسته یا بسته های و اصل شده می باشد. اعلام وصول ممکن است فردی یا جمعی ، بسته به پروتکل (قرار داد)می باشد.

Negative acknowledgment:

دریافت کننده اعلام وصول منفی جهت اینکه یک بسته به طور صحیح دریافت نشده است به فرستنده اعلام میکند اعلام وصول منفی نوعا حاصل رشته اعداد و بسته ای که به درستی دریافت نشده است می باشد

Window pipelining

ممکن است فرستنده به فرستادن بسته های با کانال کشی اطلاعاتی و پنجره که دارای توالی اعداد می باشند، محدود شده باشد که باعث افت رنج داده شده می شود با اجازه انتقال دادن به بسته های چندگانه ولی اعلام وصول نکردن، بهره برداری فرستنده می تواند بالای عملکرد توقف و انتظار افزایش داده شود.

جدول 301: خلاصه سایر کار انتقال اطلاعات قابل اعتماد و استفاده آن برای شبکه های پرسرعت[RFC 1323] و یک شیوه برای توالی اعداد می باشد آن تشریح شده به مرتب سازی مجدد مشکلات کاملا پرهیز می شود.

انتقال- ارتباط اتصال گرا(TCP)

حالا که ما اصول اساسی انتقال مطمئن اطلاعات را فهمیدیم ، به TCP می پردازیم- لایه انتقال اینترنتی ، اتصال گرا، پروتکل انتقالی مطمئن.

در این قسمت می بینیم که جهت فراهم کردن انتقال مطمئن اطلاعات, TCP به بسیاری از اصول اساسی بحث شده در بخش بعدی وابسته است. شامل ، ردیاب خطا، انتقال دوباره، اعلام وصول انباشته ، تایمرها و HEADER FIELD ها برای ترتیب و اعلام وصول اعداد می باشدTCP. درRFC 2018.... تعریف شده

1-5-3: ارتباط TCP

به TCP اتصال گرا گفته می شود زیرا قبل از اینکه یک فرایند برنامه کاربردی بتواند شروع به فرستادن اطلاعات به دیگری کند این دو باید با هم تعامل برقرار کند، آنها باید بعضی بخشهای مقدماتی را به یکدیگر بفرستند تا پارامترهای انتقال اطلاعات پشت سرهم تاسیس شود.

به عنوان قسمتی از تاسیس ارتباط TCP، هر دو سمت ارتباط بسیاری از وضعیت متغیرهای TCP را با کمک ارتباط TCP ارزش دهی اولیه می کند(که بسیاری از آنها در این قسمت و بخش 7-3 مورد بحث قرار می گیرند).

www.it98.ir

ارتباط TCP یک TDM انتها به انتها یا FDM مداری مانند مدار راهگزینی نیست و همچنین مدار مجازی نمی باشد.(به بخش 1 مراجعه کنید)، همانطور که وضعیت ارتباط، به طور کامل در انتهای هر دو سیستم مستقر می شود.به خاطر اینکه پروتکل TCP فقط در انتهای سیستم ها اجرا می شود و نه در عناصر حد واسط شبکه (مسیر یاب ها و سویچ های لایه ارتباطی، عناصر حد واسط شبکه در مرحله ارتباط TCP ذکر نمی شود)

تاریخچه موضوع: وینتون سرف، رابرت کان و TCP/IP

در اوایل دهه سال 1975- شبکه های پیام رسانی با شبکه آرپانت شروع به توسعه یافتن کرد - زیر برنامه اینترنت- فقط یکی از هزاران شرکت بود, هر کدام از این شبکه ها قرار داد مخصوص به خودشان را داشته اند.

دو محقق، وینتون سرف و رابرت کان متوجه اهمیت اتصال این شبکه ها شدند و یک قرار داد شبکه تقاطعی اختراع کردند که TCP/TP نامیده می شوند، که TCP همان قرار داد کنترل اتصال و IP قرار داد اینترنت می باشد. با وجود اینکه سرف وکان با دیدن پروتکل به عنوان نهاد منفرد شروع کردن بعدا این پروتکل به 2 قسمت تصمیم شدند TCP/TP که هر کدام جداگانه عمل می کنند.

سرف وکان یک روزنامه TCP/TP ماه می 1974 در (IEEE منتشر کردند)[سرف1974] پروتکل TCP/TP، که زیر بنای اینترنت است، قبل از کامپیو ترهای شخص اختراع شده اند،

قبل از تکثیر اترنت ها و دیگر تکنولوژی های شبکه های محلی ، وقبل از وب، انتقال صدا، و چت.

سرف وکان نیاز به یک پروتکل شبکه را دیدند که از یک طرف پشتیبانی لازم را برای برنامه های کاربردی تعریف شده را فراهم می کند و از طرف دیگر به میز بانان اختیاری و پروتکل های لایه ارتباطی اجازه فعالیت داخلی را می دهد. در سال 2004 سرف وکان جایزه ACM تورینگ را در یافت کردند که به عنوان جایزه نوبل محاسبه برای کار پیشگامانه برای اینترنت شامل طراحی و پیاده سازی اصول ابتدایی پروتکل های ارتباطی اینترنت، TCP/TP و برای رهبری الهام بخش در شبکه سازی ، محسوب می شود.

در واقع مسیر یاب های حد واسط برای ارتباط های TCP کاملا واضع هستند؛ آنها داده گرام ها را متصل نمی کنند، می بینند. یک ارتباط TCP یک سرویس کامل مضاعف را فراهم می کند: اگر در صورت وجود یک ارتباط TCP بین فرآیند A روی یک میز بان و فرایند B روی میزبان دیگر باشد اطلاعات لایه می تواند از فرآیند B به فرآیند A روی یک میز بان انتقال یابد همانطور که اطلاعات برنامه کاربردی می تواند از فرآیند B به فرآیند A انتقال پیدا کنند. یک ارتباط TCP نقطه به نقطه است که به معنی :بین یک فرستنده تنها و گیرنده تنها می باشد. بنابراین چند بخشی نامیده می شوند(بخش 7-4 مراجعه کنید). انتقال اطلاعات از یک فرستنده به تعداد زیادی گیرنده در یک مرحله فرستادن با TCP امکان پذیر نیست.

با TCP ، دو تا میزبان اطلاعات رد و بدل می کنند ولی 3 تا ازدحام ایجاد می کنند. حالا بیاید به چگونگی ثبت اطلاعات TCP نگاهی بیندازیم. فرض کنید که یک فرآیند روی یک میزبان اجرا می شود که می خواهد با یک فرایند دیگر روی یک میزبان دیگر یک ارتباط را پایه گذاری کند بخاطر داشته باشید که این فرآیند که ارتباط اولیه را صورت می دهد، فرآیند مشتری نامیده می شود. و فرآیند دیگر فرآیند سروری نامیده می شود. فرآیند مشتری ابتدا به لایه انتقال اطلاع می دهد که می خواهد یک ارتباط را به فرآیند سرور پایه گذاری کند, یادآوری از فصل (7-2). یک جا وا client این عمل را به وسیله فرستادن دستور انجام می دهد.

;(Socket clientsocket = new socket("hostname" , portnumber

در جایی که نام میزبان نام سرور است و شماره پورت, فرآیند را روی سرور مشخص می کند. لایه انتقال یک مشتری سپس عملیات ثبت یک ارتباط TCP با TCP داخل سرور را ادامه می دهد. در پایان این مرحله درباره مرحله ثبت ارتباط به تفصیل

wWw.it98.ir

www.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

بحث می کنیم. اما اکنون دانستن این نکته که مشتری ابتدا بخشی از TCP خاصی را می فرستد، سرور با بخش دوم TCP خاص جواب می دهد و بالاخره مشتری با بخش خاص سوم دوباره جواب می دهد ، دو بخش اولیه ظرفیت ترابری ندارد در نتیجه اطلاعات لایه کاربردی را نداریم ، سومین بخش قابلیت نقل و انتقال دارد به دلیل اینکه این سه بخش بین 2 میزبان فرستاده می شود از این مرحله ثبت ارتباط اغلب به عنوان ارتباط سه راه یاد می شود.

زمانی که ارتباط TCP آغاز به کار کرده ، در فرآیند کاربردی می توانند با یکدیگر اطلاعات رد و بدل کنند. بیاید فرستادن اطلاعات از فرآیند مشتری به فرآیند سرورا در نظر بگیریم فرایند مشتری یک جریان از اطلاعات را از طریق سوکت (ورودی مرحله) انتقال می دهد.

همان طور که در بخش 7-2 توضیح داده شد. وقتی که اطلاعات از طریق در انتقاد داده می شوند , اطلاعات در دست TCP اجرا شده داخل مشتری می باشد. همانطور که در شکل 302 نشان داده شده است. TCP این اطلاعات را داخل با فر ارسال ذخیره می کند که یکی از بافرهایی است در کنار ارتباط سه راهی اولیه جایگزاری شده است. در هر زمان TCPقسمت بزرگی از اطلاعات را از روی send بافر بر می دارد. جالب توجه است که TCP برای تعیین, بسیار ساده عمل می کند. در واقع باید اطلاعات ذخیره شده را بفرستد، مشخص می کند که TCPباید اطلاعات را در بخش هایی در حد تسهیلات خودش بفرستد حداکثر میزان اطلاعات که می تواند برداشته شود و در یک بخش قرار گیرد به وسیله حداکثر ظرفیت بخش Mss محدود می کند، mss،maximum segment size به طور نمونه ابتدا به وسیله طول و حجم قابل لایه ارتباطی تعیین می شود که می تواند به وسیله میزبان فرستنده محلی فرستاده شود. (حداکثر ارسال واحد MTU نامیده می شود.)

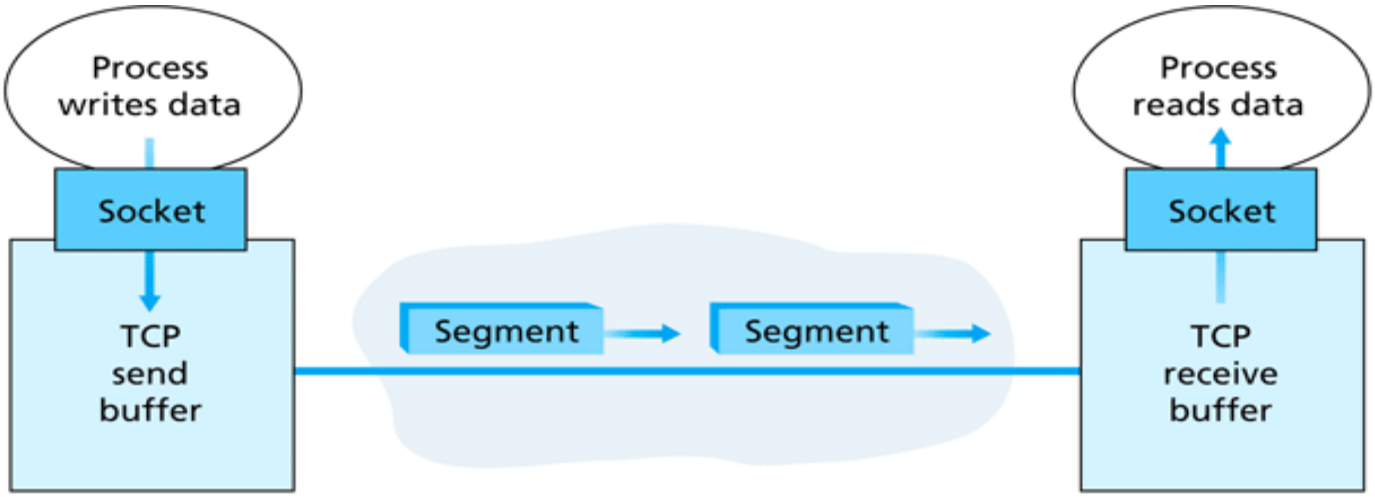


Figure 3.28 ♦ TCP send and receive buffers

و سپس mss جهت تضمین یک بخش TCP نصب می شود(وقتی که داخل یک IP داده گرام جاگذاری می شود) به صورت یک قالب لایه ای ارتباط منفرد در می آید. حجم ها مشترک برای 1/460MTU بایت، 536 بایت و 512 بایت هستند دسترسی ها همچنین برای شناسایی مسیر MTU پیشنهاد شده است- بزرگترین قالب لایه ارتباطی که می تواند در تمام ارتباطات از مبدا به مقصد فرستاده شود به [RFC 1191 مراجعه نماید] و نصب Mss بر پایه مسیر مقدار MTU می باشد .

توجه کنید که Mss حداکثر میزان اطلاعات لایه کاربردی در بخش می باشند حداکثر سایز بخش TCP شامل HEADERها نه حداکثر اندازه TCP در برگیرنده HEADERها

این مجموعات اطلاعات گیج کننده می باشد و لی ما مجبوریم با آنها زندگی کنیم. مانند اینکه آنها به خوبی ثبت شده اند)

www.it98.ir

جفتهای TCP هر یک شامل قسمت بزرگی از اطلاعات مشتری با یک TCP- HEADER می باشد.
در نتیجه یک بخش TCP شکل می گیرد. بخش ها به طرف لایه شبکه وارد می شود جایی که آنها به طور جداگانه در داده گرام IP لایه شبکه جایگذاری می شود. سپس داده گرام IP به شبکه فرستاده می شود.وقتی که TCP یک بخش را در یک انتها دریافت می کند، اطلاعات بخش داخل بافر TCPارتباطی گیرنده جایگذاری می شود همانطور که در شکل 28-3 نمایش داده شده است. برنامه کاربردی جریان اطلاعات از روی این بافر می خواند. هر سمت ارتباط بافرفرستنده مربوطه به خودش را دارد(شما می توانید جریان کنترل آن لاین اپلت را در http://www.awl.com/kurose-ross که یک انیمیشن از ارسال و دریافت بافرها را نمایش می دهد ببینند.)
ما از این بحث نتیجه می گیریم که ارتباط TCP مرکب از بافرها ، متغیرها و سوکتهای اتباطی برای یک فرآیند در یک میزبان، و یک مجموعه دیگر از بافرها، متغیرها و یک سوکت ارتباطی به یک فرآیند در یک میزبان دیگر است همانگونه که قبلا ذکر شد هیچگونه بافر یا متغیری به ارتباط عناصر شبکه بین دو میزبان تشخیص داده نشده(ریشه ها، سویچ ها وری پیترها)

3-5-2ساختار بخش TCP

با یک نگاه کوتاه به ارتباط TCP،بیایند ساختار بخش TCP را آزمایش کنیم یک بخش TCP از میدان HEADER ها و میدان اطلاعات تشکیل شده است. میدان اطلاعات شامل قسمت بزرگی از اطلاعات کاربردی می باشد.

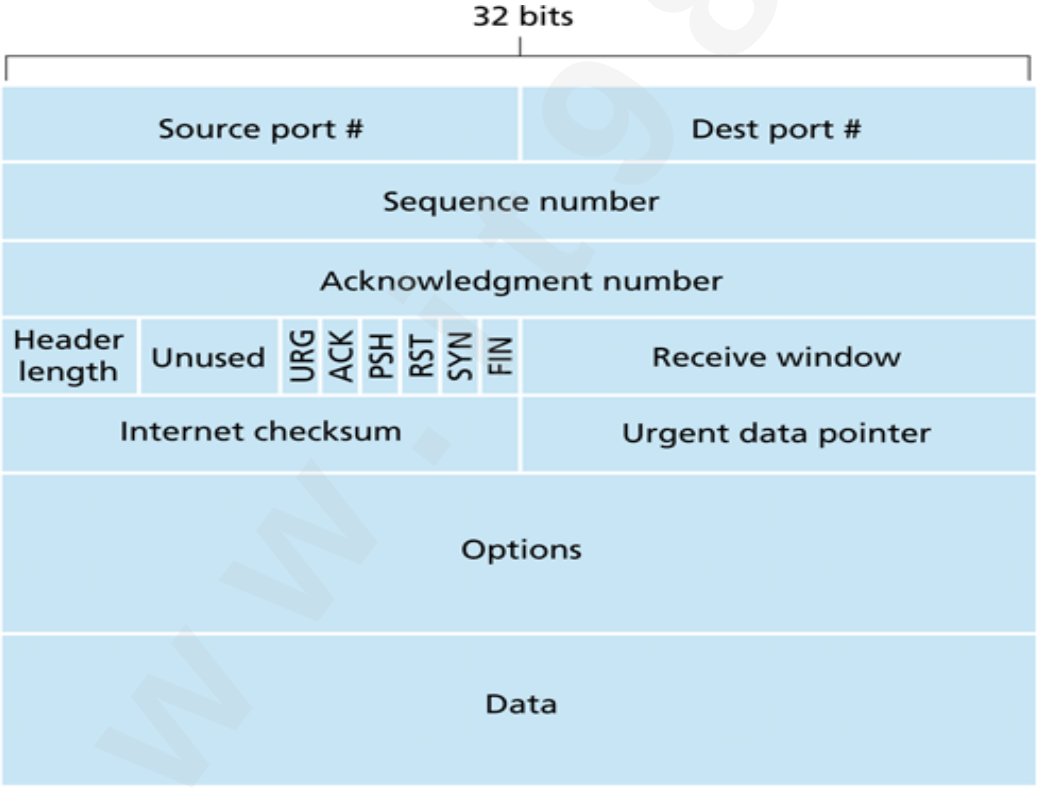


Figure 3.29 ♦ TCP segment structure

همانگونه که در بالا ذکر شد Mss حداکثر اندازه یک بخش میدان اطلاعات را محدود می کند.
وقتی که TCP یک فایل بزرگ می فرستد مثل یک تصویر که قسمتی از یک صفحه وب است، آن را به فایلهایی در حد اندازه Mss می شکند (به غیر از آخرین تکه که معمولا کمتر از حجم Mss است) برنامه کاربردی تعاملی هر چند اغلب تکه های اطلاعات که کوچکتر از Mss است انتقال می دهند. برای مثال با ورود از دور به برنامه کاربردی مانند Telnet ، میدان اطلاعات

wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

در بخش TCPاغلب تنها یک بایت است. به دلیل اینکه HEADER، TCP عموما 20 بایت است 12 بایت بیشتر از UDP-header بخش فرستاده شده به وسیله Telnet ممکن است تنها به طول 21 بایت باشد.

در شکل 29-3 ساختار بخش TCP سگمنت نشان داده شده است. مانند UDP، header شامل مبدا و مقصد شماره پورت می باشد که برای تسهیل سازی و عدم تسهیل سازی اطلاعات از / به لایه بالاتر,لایه کاربردی استفاده می شود. همچنین با UDP-header شامل CHECKSUM FIELD جمع کنترلی می باشد . یک header بخش TCP همچنین در برگیرنده زیرمجموعه های زیر است.

32 بیت میدان اعداد ترتیبی و 32 بیت میدان اعداد تصدیقی که به وسیله فرستنده و گیرنده TCP در اجرا یک سرویس انتقال مطمئن اطلاعات استفاده می شود، همانطور که در پایین توضیح داده شده است.

Receive window 16 بیتی که برای کنترل جریان استفاده می شود. ما به صورت خلاصه می بینیم که این برای نشان داده شماره با بیتهایی استفاده می شود که یک دریافت شده می تواند قبول کند.

Header length field به طول 4 بیت که طول header، TCP در 32 بیت کلمه را مشخص می کند. Header، TCP می تواند طول متغیری بنابر میدان انتخاب TCP داشته باشد.

طول متغیر و اختیاری که زمانی که یک فرستنده و گیرنده در حداکثر اندازه بخش Mss مراوده می کنند یا به عنوان فاکتور عامل اندازه گیری بدوه برای استفاده در شبکه های پرسرعت استفاده می شود.

یک اختیار قالب بندی زمانی نیز تعریف شده است.

Flag field شامل 6 بیت. Ack Bit برای نشان دادن این که ارزش رسیده در میدان اعلام وصول معتبر است استفاده می شود. این بخش شامل یک اعلام وصول برای یک سگمنت که به صورت موفق دریافت شده می باشد.

بیتهای FIN,SYN,RST برای برقراری و متلاشی کردن ارتباط استفاده می شود همان طور که در پایان این بخش توضیح داده شده تنظیم بیت PSH مشخص می کند که گیرنده باید اطلاعات را سریعا به لایه بالاتر بفرستند در نهایت بیت Urg مشخص می کند.اطلاعاتی در این بخش وجود دارد.

فرستادن به هویت لایه بالایی ضروری خوانده شده است موقعیت آخرین با بیت اطلاعات ضرورتی با میدان 16 بیتی نشان دهنده اطلاعات ضروری مشخص می شود.

Sequence number and Acknowledgment Numbers

اعداد ترتیبی و اعداد اعلام وصولی

2 تا از مهمترین فیلدها در header بخش TCP فیلد اعداد ترتیبی و فیلد اعداد تصدیقی می باشد این فیلدها یک بخش حساس سرویس انتقال مطمئن TCP می باشد.ولی قبل از اینکه در مورد چگونگی استفاده این فیلدها برای به وجود آوردن انتقال مطمئن دیتا بحث کنیم بیائید ابتدا دقیقا چگونگی جایگذاری TCP در این فیلدها را توضیح دهیم.

TCP داده ها را بدون شکل ولی مرتب می بیند، جریانی از بایت ها، استفاده TCP از اعداد ترتیبی این نگرش را در مورد اعداد ترتیبی نشان می دهد.

www.it98.ir

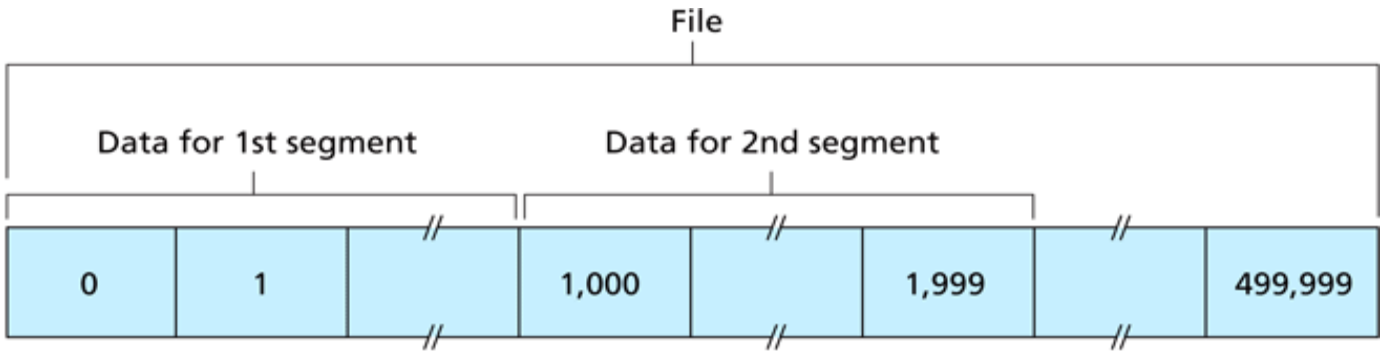


Figure 3.30 ♦ Dividing file data into TCP segments

عدد ترتیبی برای یک بخش برای شماره جریان بایت از اولین بایت درون بخش می باشد. بیائید به یک مثال نگاه کنیم. فرض کنید یک فرآیند در میزبان A می خواهد یک جریان اطلاعات را برای یک فرآیند در میزبان B از طریق ارتباط TCP بفرستد. TCP میزبان A هر بایت در جریان اطلاعات شماره ای اختصاص می دهد. فرض کنید که این جریان اطلاعات مرکب از 000/500 بایت ترکیب شده است که Mss آن 1000 بایت باشد و به اولین بایت جریان اطلاعات شماره صفر داده شود همانطور که در شکل 3020 نشان داده شده است، ساختار TCP 500 بخشی خارج از جریان داده است، اولین بخش با عدد ترتیبی صفر مشخص شده است، بخش دوم شماره ترتیبی 000/1 را می گیرد.

سومین بخش عدد ترتیبی 000/2 و مثل این هر عدد ترتیبی در فیلد اعداد ترتیبی هدر TCP درج می شود.

حالا بیاید اعداد تصدیقی را در نظر بگیریم این ها کمی بیشتر از اعداد ترتیبی گیج کننده هستند به خاطر بیارید که TCP مضاعف کامل است بنابر این میزبان A ممکن است اطلاعات را از میزبان B در ضمن ارسال به اطلاعات به میزبان B دریافت کند.(به عنوان قسمتی از ارتباط TCP یکنواخت هر کدام از بخش ها که از میزبان B می رسند یکه عدد ترتیبی برای جریان اطلاعات از B به A دارند. عدد تصدیقی که میزبان A در بخش خودش می گذارد عدد ترتیبی بایت بعدی میزبان A است که از میزبان B انتظار دارد. خوب است که نگاهی به چند مثال بیندازیم تا بفهیم در این جا چه اتفاقی می افتد. فرض کنید که میزبان A تمام بیت های شماره صفر را از طریق 535 از B دریافت کرده است و فرض کنید در حال فرستادن یک بخش به میزبان B می باشد. میزبان A منتظر بایت 536 و تمام متعاقب در جریان اطلاعات میزبان B می باشد. بنابراین میزبان 536 A را در فیلد اعداد تصدیقی بخش برای B می فرستد. به عنوان مثالی دیگر فرض کنید که میزبان A یک بخش از میزبان B شامل بایت های صفر از طریق 535 و بخش دیگر شامل با بیتهای 900 از طریق 000/1 دریافت کرده است و به دلیل مشابهه میزبان A هنوز بایت 536 را از طریق 899 دریافت نکرده است. در این مثال میزبان A هنوز منتظر بایت 536 جهت دوباره سازی جریان اطلاعات B می باشد. بنابراین بخش بعدی A به B شامل 536 در فیلد عدد تصدیقی می باشد برای این که بایت هایی تصدیقی TCP فقط برای اولین بایت های گم شده در جریان است به TCP گفته می شود اعلام وصول متراکم آماده کند. در شکل 3-30 ما فرض می کنیم که عدد ترتیبی اولیه صفر باشد در حقیقت هر دو طرف ارتباط TCP یک عدد ترتیبی اولیه انتخاب می کند این عمل جهت کم کردن این احتمال است که یک بخش که هنوز در شبکه از قبل وجود دارد و در حال پایان دادن به ارتباط بین 2 میزبان است با یک بخش در دسترس در ارتباط بعدی بین همین 2 میزبان اشتباه نشود.

Telnet یک مورد مطالعه برای اعداد ترتیبی و تصدیقی Telnet در RFC 854تعریف شده است ، این یک پروتکل لایه کاربردی محسوب مورد استفاده برای ارتباط از راه دور می باشد آن در TCP اجرا می شود و برای کاربین هر جفت میزبانی طراحی شده است. متفاوت با کاربردهای انتقالی اطلاعات حجمی بحث شده در بخش 2 می باشد. Telnet یک فرآیند تعاملی می باشد. در این جا یک مثال Telnet را توضیح می دهیم.همانطورکه به خوبی دراعداد ترتیبی وترکیبی TCP به تصدیق

www.it98.ir
wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

کشیده شده است.

ما در نظر می گیریم بسیاری از USER ها استفاده از پروتکل SSH را به Telnet ترجیح می دهند. زمانی که به اطلاعات با یک ارتباط Telnet فرستاده شد (شامل پسوردها) به صورت رمز نشده است. Telnet را در مقابل خطر استراق سمع آسیب پذیر می کند. همانطور که در بخش 7-8 توضیح داده شد. فرض کنید که میزبان A یک جلسه Telnet را با میزبان B برگزار می کند به خاطر اینکه میزبان A جلسه را پایه گذاری کرده است.

آن را مشتری قلمداد می کند و مشتری B را سرور قلمداد می کند هر شخصیتی به وسیله کاربر TYPE شود برای مشتری به میزبان راه دور فرستاده می شود میزبان راه دور فرستاده می شود میزبان راه دور یک کپی از هر حذف را پس می فرستد که روی صحفه کاربر Telnet بکار برده می شود این برگشت موج برای تضمین این که شخصیت ها به وسیله کاربر Telnet دیده شده و همچنین دریافت شده اند و در سایت راه دور پردازش شده اند استفاده می شوند هر کارکتر شبکه را فاصله زمانی که کاربر یک کلید را فشار می دهد تا زمانی که کارکتر روی مانیتور کاربر به نمایش در بیاید 2 بار می پیماید حالا فرض کنید کاربر C را تایپ می نماید و سپس قهوه اش را بر می دارد. بیایید بخش TCP را که بین مشتری و سرور رد و بدل می شود را آزمایش کنیم.

همان طور که در شکل 1-2-3 نشان داده شده است ما فرض می کینم که عدد ترتیبی شروع کننده 42 و 79 برای مشتری و سرور می باشد. به خاطر داشته باشید که عدد ترتیبی یک بخش عدد ترتیبی اولین بایت فیلد اطلاعات می باشد بنابراین اولین بخش که از طرف مشتری فرستاده می شود شماره ترتیبی 42 را دارد، اولین بخش فرستاده شده توسط سرور عدد 79 را دارد به خاطر داشته باشید که اعداد تصدیقی اعداد ترتیبی بایت بعدی اطلاعاتی هستند که میزبان منتظر آنهاست بعد از تثبیت ارتباط TCP ولی قبل از فرستادن هر اطلاعاتی مشتری منتظر بایت 79 و سرور منتظر بایت 42 می باشد. همان طور که در شکل 1-3-3 نشان داده شده است 3 بخش فرستاده شده است اولین بخش از طرف مشتری به سرور شامل یک بایت ASCII که نماینده حرف C در فیلد داده های خودش می باشد این بخش هم چنین شماره 42 را در اعداد ترتیبی فیلد خودش دارد همانطور که تا حالا توضیح دادیم هم چنین به دلیل اینکه مشتری هنوز هیچ اطلاعاتی از سرور دریافت نکرده است این اولین بخش شماره 79 را در فیلد اعداد تصدیقی خودش می گیردبخش دوم از سرور به مشتری فرستاده می شود که هدف دو گانه این را دارد اول اینکه اعلام وصول اطلاعات ی که سرور دریافت کرده است به وسیله قرار دادن عدد 42 در فیلد اعلام وصول سرور به مشتری اعلام می کند که هر چیزی را که درون بایت 42 بایت بدست خودش می باشد بنابراین بخش ثانویه نماینده ASCII در فیلد اطلاعات خودش می باشد. بخش دوم شماره ترتیبی 79 را دارد و شماره ترتیبی اولیه جریان اطلاعات سرور به مشتری می باشد و به خاطر اینکه این اولین بایت اطلاعاتی است که سرور می فرستد توجه داشته باشید که اعلام وصول برای اطلاعات مشتری به سرور در بخش حاصل ارتباط سرور به مشتری منتقل می شود به این اعلام وصول Piggybacked گفته می شود که بخش اطلاعات سرور به مشتری می باشد. بخش سوم از طرف مشتری به سرور فرستاده می شود این هدف انحصاری اعلام وصول اطلاعات که از سرور دریافت شده می باشد(به خاطر داشته باشید که بخش دوم حاصل اطلاعات... حرفC... از سرور به مشتری می باشد.

این بخش یک فیلد اطلاعات خالی دارد (که این اعلام وصول به وسیله هیچ اطلاعات مشتری به سرور Piggybacked شده است) این بخش شماره 80 را در فیلد اعدا تصدیقی دارد و بخاطر اینکه مشتری جریان بایت ها را از طریق بایت شماره 79 دریافت کرده و اکنون منتظر بایت 80 است شما ممکن است که فکر کنید که عجیب است که این بخش هم چنین یک عد ترتیبی دارد و شامل هیچ اطلاعاتی باشد ولی به دلیل اینکه TCP فیلد اعدا ترتیبی را دارد این بخش باید یک عدد ترتیبی داده شده باشد.

www.it98.ir

INFORMATION TECHNOLOGY
FORUM
انجمن فناوری اطلاعات
www.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

۱.۶.۳ دلایل و هزینه های ازدحام

بیایید موضوع اصلی مطالعه ی کنترل ازدحام با احتساب ۳ سناریو پیچیدگی در حال رشد که در هر کدام ازدحام اتفاق می افتد آغاز کنیم. در هر مورد، در مرحله ی اول ما به دنبال این خواهیم گشت که چرا ازدحام اتفاق می افتد؟ و هزینه های ازدحام را بررسی می کنیم(در اصطلاح منابع به طور کامل مورد استفاده قرار نگرفته و عملکرد ضعیفی بوسیله سیستم نهایی می رسد). ما نمی خواهیم (هنوز) روی اینکه چطور عکس العمل نشان دهیم، یا چطور از ازدحام اجتناب کنیم، تمرکز کنیم، اما بجای آن روی موضوعی که فهمیدنش ساده تر است، یعنی چه اتفاقی می افتد که میزبان ها، نرخ انتقالات شان افزایش پیدا کرده و شبکه پرازدحام و شلوغ می شود، تمرکز کنیم.

سناریو ۱ : ۲ ارسال کننده، یک مسیریاب با بافرهای نامحدود

ما با احتساب شاید سناریوی ممکن ازدحام آغاز می کنیم. ۲ میزبان (A , B) که هرکدام یک اتصال دارند که یک هاب مابین مبدا و مقصد به اشتراک گذاشته شده است. همانگونه که در شکل ۴۳.۳ نشان داده شده است، بیایید فرض کنیم که برنامه ای که در کامپیوتر میزبان A در حال ارسال داده به مسیر اتصال است (برای مثال فرستادن داده به پروتکل سطح یا لایه ی انتقال بوسیله ی یک سوکت) در یک نرخ متوسط لانداى ورودی بایت/ثانیه. این داده ها در منطق اصیل هستند زیرا هر واحدی از داده ها فقط یک بار به درون سوکت فرستاده شده است. مبنای پروتکل لایه ی انتقال یک چیز ساده است. داده ها کپسوله شده و ارسال شده است، بدون خطای بازگردانی (برای مثال انتقال دوباره) کنترل سرریز، یا کنترل ازدحام انجام شده است. صرف نظر از کنترل سرریز اضافه شده در طول انتقال و اطلاعات سرآیند لایه های پایین تر، نرخ هرکدام از میزبان A خواستار ترافیک مسیریاب در اولین سناریو است. بنابراین لانداى ورودی بایت/ثانیه میزبان B در مسیر مشابه عمل می کند و ما برای سادگی فرض می کنیم که هردو در نرخ لانداى ورودی بایت/ثانیه ارسال می کنند. بسته ها از میزبان A و B از طریق مسیریاب و در طول یک لینک ارتباطی به اشتراک گذاشته شده با ظرفیت R عبور می کنند.

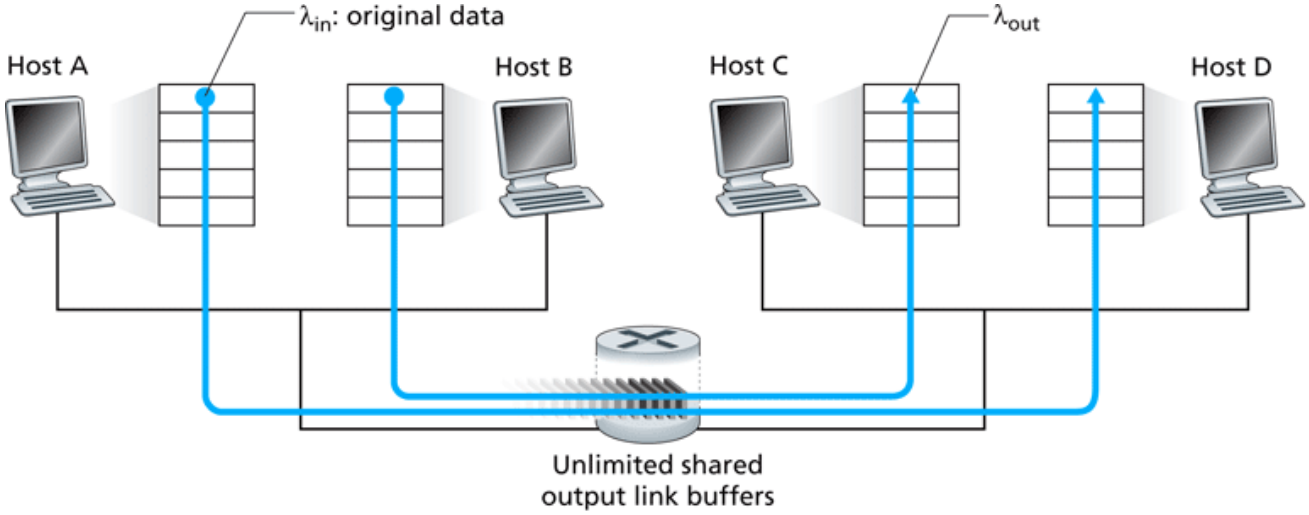


Figure 3.43 ♦ Congestion scenario 1: Two connections sharing a single hop with infinite buffers

مسیریاب بافرهایی دارد که به او اجازه می دهد که بسته های ورودی را زمانی که نرخ بسته های رسیده فراتر از از حد ظرفیت لینک های عبور باشد، ذخیره کند. در این اولین سناریو، ما فرض کرده ایم که مسیریاب یک مقدار نامحدودی از فضای بافر را در اختیار دارد. شکل ۴۴.۳ عملکرد اتصال A را تحت شرایط اولین سناریو رسم می کند. نمودار سمت چپ ظرفیت پذیرش هر

www.it98.ir

اتصال (تعداد بایت در هر ثانیه در گیرنده) به عنوان یک تابع از نرخ اتصال – ارسال را رسم می کند. برای یک نرخ ارسال بین ۰ تا R/2 توانش زمانی گیرنده، معادل است با نرخ ارسال فرستنده، هر چیز فرستاده شده بوسیله ی ارسال کننده در گیرنده با یک تاخیر محدود دریافت شده است.

زمانی که نرخ ارسال از R/2 بالاتر است، با این وجود، ظرفیت پذیرش فقط R/2 است. این محدودیت بالاتر روی ظرفیت پذیرش پیامد به اشتراک گذاشتن ظرفیت لینک ها بین دو اتصال است. لینک به سادگی نمی تواند بسته ها را به یک گیرنده در یک مقدار حالت ثابت که فراتر از R/2 است برساند. مهم نیست که میزبان های A , B میزان ارسال ان ها چقدر بالا تنظیم شده است، آن ها هرگز یک توانش زمانی بالاتر از R/2 نخواهند رسید.دست یافتن به توانش زمانی R/2 در هر اتصال ممکن است در واقع یک چیز خوب بنظر برسد، بدلیل اینحکه لینک کاملا مورد استفاده واقع شده است، در تحویل بسته ها به مقصدشان.

نمودار دست راست در شکل ۴۴.۳،با این حال، پیامد عملکرد نزدیک به ظرفیت لینک را نشان می دهد. به محض اینکه ارسال اطلاعات به R/2 نزدیک می شود (از سمت چپ) متوسط تاخیر بزرگتر و بزرگتر می شود. زمانی که میزان ارسال از R/2 تجاوز کند، تعداد متوسط بسته های صف شده در مسیریاب از محدوده خارج شده و تاخیر متوسط بین مبدا و مقصد نامحدود می شود(فرض کنید که اتصالات در این میزان نرخ ارسال عمل می کنند برای یک برهه ی زمانی نامحدود و مقدار نامحدود از بافر نیز در دسترس است و وجود دارد. بنابراین تا زمانی که عمل کردن در مجموع ظرفیت پذیرش نزدیک به R ممکن است از دید ظرفیت پذیرش ایده آل باشد، از دید تاخیر بسیار دور است. حتی در این(بیش ازحد) سناریوی ایده آل شده، ما هم اکنون یک هزینه ی شبکه ی شلوغ شده یا مزدحم شده را یافتیم، تاخیر صف کرده های بزرگ برای میزان بسته های تازه آمده نزدیک به ظرفیت لینک تجربه شده است.

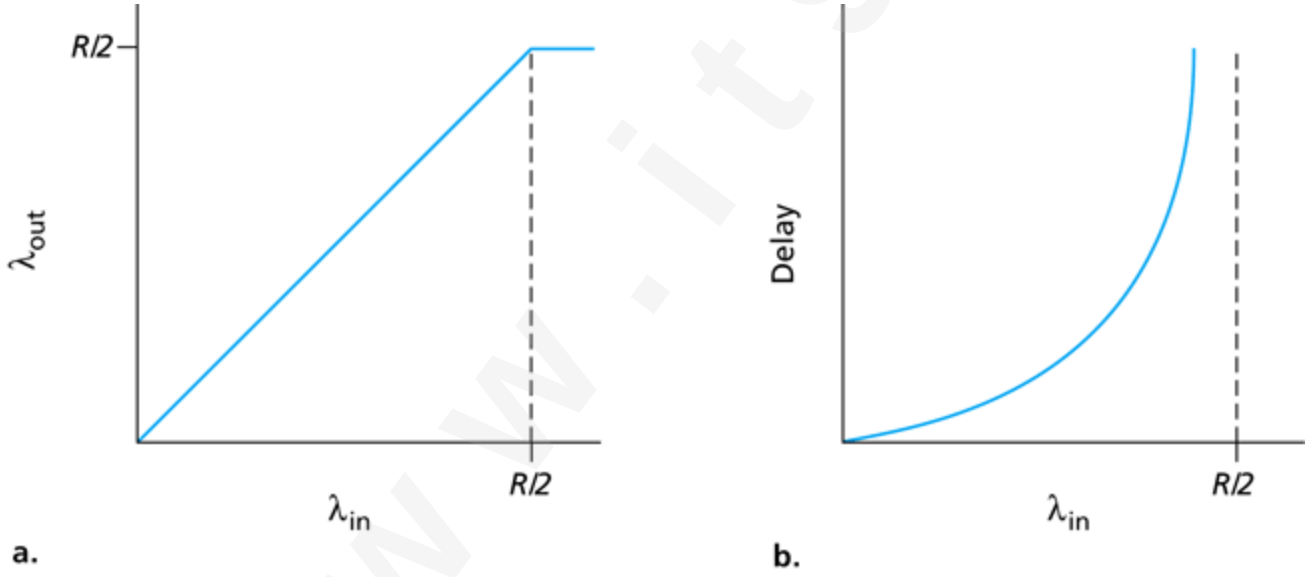


Figure 3.44 ♦ Congestion scenario 1: Throughput and delay as a function of host sending rate

سناریوی ۲ : ۲ فرستنده و یک مسیریاب با بافر محدود

بگذارید ما حالا یک ذره سناریوی ۱ را در ۲ مسیر پیش رو تغییر دهیم (شکل ۴۵.۳ را ببینید). اول، مقدار بافرینگ مسیریاب

www.it98.ir
wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

محدود فرض شده است. یکی از نتایج این فرض جهان واقعی این است که بسته ها زمانی که به بافر پر برسند رها خواهند شد.اگر یک بسته که شامل سرآیند لایه ی انتقال است در مسیریاب افتاده باشد، ارسال کننده معمولا دوباره آن را منتقل خواهد کرد. بخاطر اینکه بسته ها می توانند دوباره منتقل شوند، ما باید حالا بیشتر مراقب استفاده مان از واژه ی نرخ ارسال باشیم مخصوصا، بگذارید ما دوباره تاکید کنیم که میزان هر برنامه ای که داده های اصلی را به سوکت ارسال می کند بوسیله لاندای ورودی بایت/ثانیه مشخص می شود. میزانی که لایه ی انتقال سرآیندها را ارسال می کند (شامل داده های اصلی و داده های دوباره انتقال یافته) درون شبکه بصورت لاندا پریم ورودی بایت/ثانیه موکد خواهد شد. لاندا پریم ورودی بایت/ثانیه بعضی اوقات بعنوان بارگذاری درخواست شده ی شبکه ارجاع داده می شود. درک عملکرد طبق سناریوی ۲. حالا قویا به اینکه چگونه انتقال دوباره انجام خواهد شد بستگی دارد.

نخست، یک مورد غیرواقعی را محسوب می کنیم که میزبان A قادر است بگونه ای (جادویی!) مشخص کند که یک بافر در مسیریاب آزاد است یا نه. بنابراین یک بسته را فقط زمانی که بافر آزاد(خالی) است ارسال می کند. در این مورد، از بین رفتن اتفاق نخواهد افتاد.لانداری ورودی بایت بر ثانیه برابر با لاندا پریم ورودی بایت/ثانیه خواهد بود و توانش زمانی یا ظرفیت پذیرش انتقال ها متناسب با لاندا پریم ورودی بایت/ثانیه خواهد بود. این مورد در شکل ۴۴.۳ نشان داده شده است. از منظر ظرفیت پذیرش، عملکرد ایده آل است، هرچیزی که ارسال شده است دریافت شده است. توجه کنید که میزان متوسط ارسال میزبان نمی تواند بیش از R/2 باشد تحت این سناریو، تا زمانی که از دست رفتن بسته ها فرض شده است که هرگز رخ نخواهد داد.

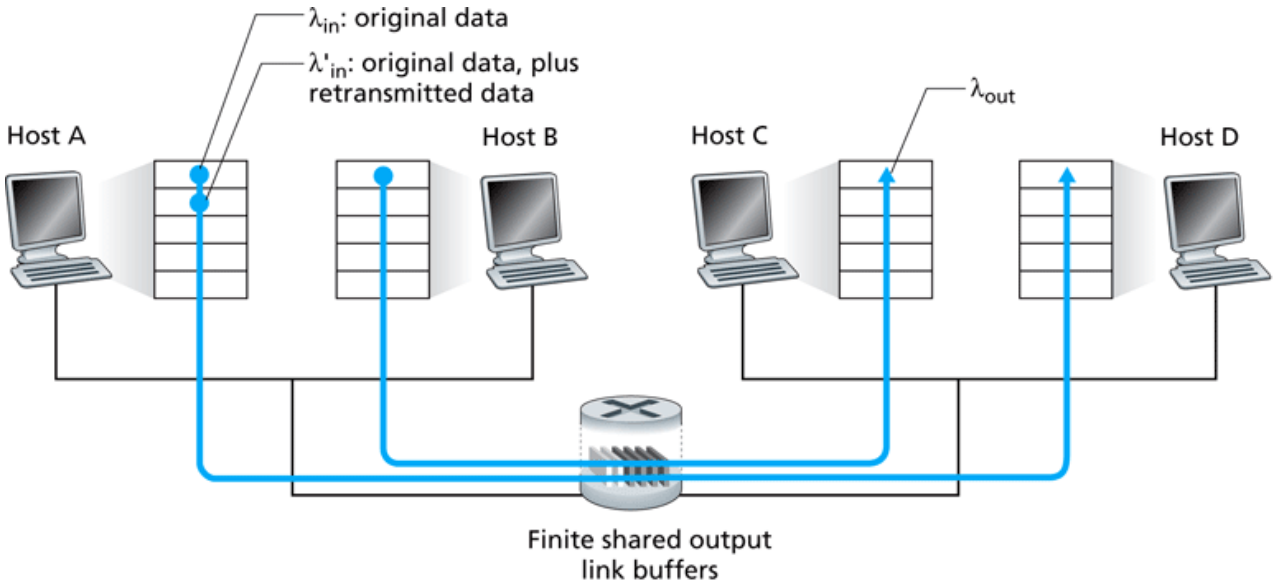


Figure 3.45 ♦ Scenario 2: Two hosts (with retransmissions) and a router with finite buffers

مورد بعد را که کمی بیشتر به واقعیت نزدیک است فرض کنید که ارسال کننده فقط زمانی که بسته ی گمشده برای یک بخش خاص شناخته شده است آن را دوباره انتقال می دهد(دوباره این فرض مقداری انعطاف دارد، با این وجود، این امکان پذیر است که میزبان فرستنده ممکن است زمان مابین ارال هایش را به اندازه کافی بزرگ تنظیم کرده باشد تا بصورت مجازی تضمین دهد که بسته هایی که تصدیق نخورده اند از بین رفته اند). در این مورد ممکن است شبیه چیزی برسد که در شکل ۴۶.۳ نشان داده شده است. برای درک کردن اینکه چه اتفاقی اینجا رخ داده است، شرایط را طوری فرض کنید که بارگذاری درخواست شده لاندا پریم ورودی بایت/ثانیه باشد (میزان نرخ انتقال داده های اصلی بعلاوه ی انتقال دوباره) مساوی R/2. طبق شکل ۴۶.۳

www.it98.ir

بارگذاری درخواست شده در این مقدار هرکدام از داده هایی که به برنامه گیرنده می رسد R/3 است. بنابراین بیش از R 0/5 واحد از داده های منتقل شده، R 0/33 بایت/ثانیه (بطور متوسط) داده های اصلی هستند و R 0/16 بایت/ثانیه (بطور متوسط) داده های دوباره انتقال یافته هستند. ما اینجا یکی دیگر از هزینه های شبکه پرازدحام را می بینیم – فرستنده باید عمل انتقال دوباره را انجام دهد تا برای بسته های رها شده بخاطر سرریز بافر را جبران کند.

در نهایت، بیایید شرایطی را فرض کنیم که فرستنده ممکن است بیش از موقع Timeout شود و بسته ای را دوباره ارسال کند که در صف تاخیر دارد اما هنوز از بین نرفته است. در این شرایط، هردو بسته های شامل داده های اصلی و دوباره انتقال یافته ممکن است به گیرنده برسند. البته، گیرنده به یک کپی از بسته احتیاج دارد و بسته را نخواهد پذیرفت. در این مورد کار بوسیله ی مسیریاب در ارسال کپی دوباره انتقال یافته از بسته ی داده ای اصلی که تلف شده بودند انجام می شود، تا به گیرنده هم اکنون یک کپی اصلی از این بسته رسیده باشد. مسیریاب بهتر است تا به اندازه ظرفیت انتقال لینک بجای ارسال بسته های متفاوت استفاده کند. پس اینجا نیز یکی دیگر از هزینه های شبکه ی پرازدحام است، انتقال دوباره ی ناخواسته بوسیله ارسال کننده در صورت تاخیرهای بزرگ ممکن است باعث شود تا یک مسیریاب از پهنای باند لینک خود برای ارسال کپی های ناخواسته از بسته استفاده کند. شکل ۴۶.۳ نشان می دهد ظرفیت پذیرش ذکر شده را، زمانی که فرض شده است هر بسته دوبار بوسیله ی مسیریاب (بطور متوسط) ارسال شده است. تا زمانی که هر بسته دوبار ارسال شود، توانش زمانی(ظرفیت پذیرش) مقداری نشانه ی R/4 از بارگذاری خواسته شده تا R/2 را خواهد داشت.

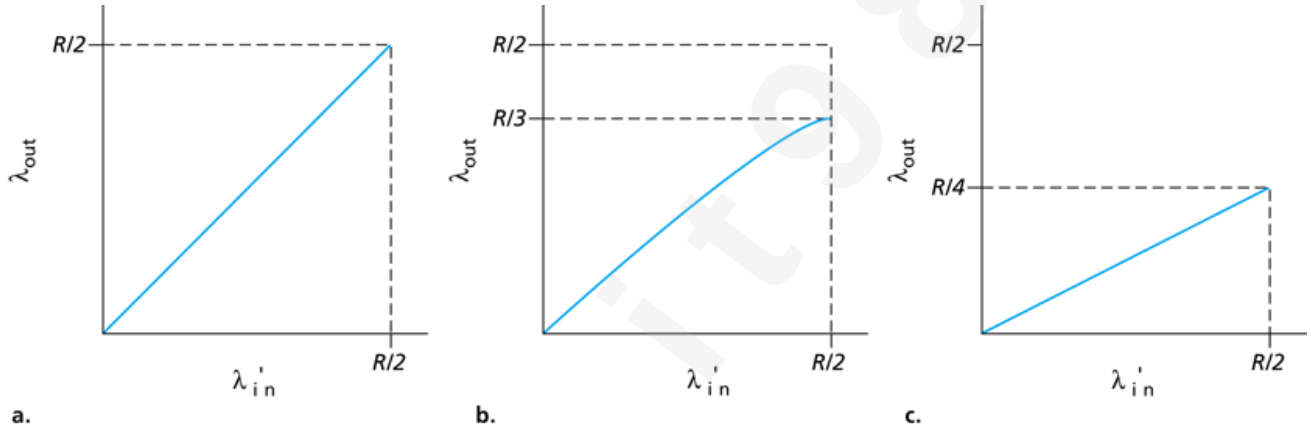


Figure 3.46 ♦ Scenario 2 performance with finite buffers

سناریو ۳ : ۴ فرستنده، مسیریاب با بافر محدود و مسیرهای چندهاب

در آخرین سناریوی ازدحام ما، ۴ میزبان بسته ها را انتقال می دهند، هرکدام روی لبه ی مسیرهای دوهاب، همانطور که در شکل ۴۷.۳ نشان داده شده است. ما دوباره فرض می کنیم که هر میزان از یک مکانیزم پایان زمان/انتقال دوباره برای پیاده سازی یک سرویس انتقال داده ی قابل اعتماد استفاده می کند، که همه ی میزبان ها مقدار مشابه لاندای ورودی بایت/ثانیه و همه ی لینک های مسیریاب ها ظرفیت R بایت/ثانیه را دارا خواهند بود. بیایید اتصال بین میزبان A و میزبان B را فرض کنیم، که از بین مسیریاب های R1 , R2 می گذرد. اتصال A-C و مسیریاب R1 را با اتصال B-D به اشتراک می گذارد و مسیریاب R2 را با یک اتصال B-D برای مقادیر بیش از حد کوچک لاندا ورودی بایت/ثانیه سرریز بافر نادر و کمیاب است (مثل سناریوی ازدحام ۱ و ۲) و ظرفیت پذیرش تقریبا معادل بارگذاری خواسته شده است. برای مقادیر کمی بزرگتر از لاندا ورودی بایت/ثانیه ظرفیت پذیرش نیز همچنین بزرگتر است. تا زمانی که بیشتر داده های اصلی منتقل شده اند در شبکه و به مقصد رسیده اند، و سرریز هنوز کمیاب و نادر است. بنابراین برای مقادیر کوچک لاندای ورودی بایت/ثانیه یک افزایش در لاندای ورودی بایت/ثانیه

wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

نتیجه اش افزایش در لاندا خروجی بایت/ثانیه است.

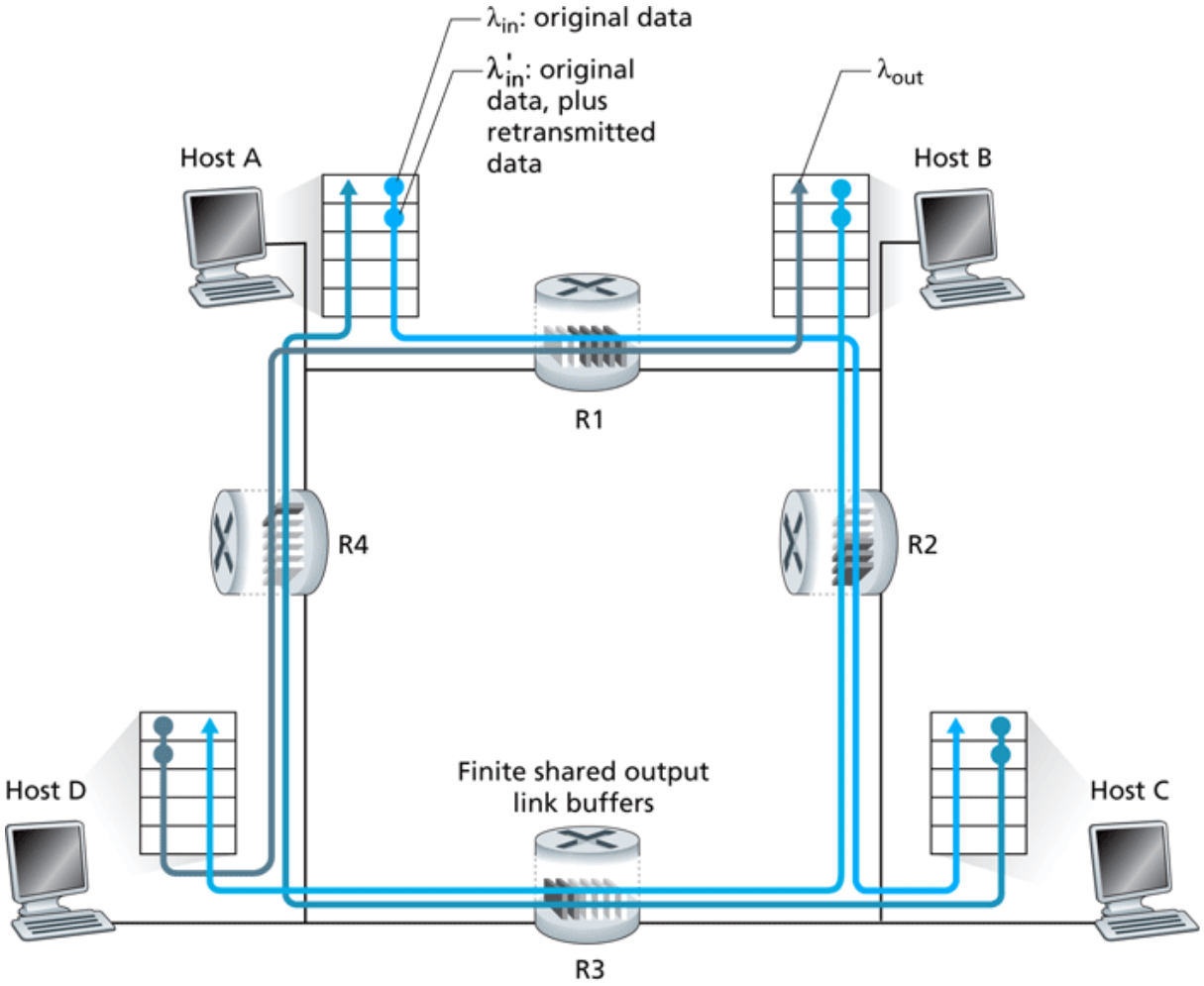


Figure 3.47 ♦ Four senders, routers with finite buffers, and multihop paths

مورد ترافیک بیش از حد پایین را بررسی کردیم، بیایید مورد بعدی را که لاندا ورودی بایت/ثانیه (و متقابلا لاندا پریم ورودی بایت/ثانیه) بیش از اندازه بزرگ است را بررسی کنیم. مسیریاب R2 را در نظر بگیرید. ترافیک رسیدن A-C به مسیریاب R2 (که به R2 می رسد بعد از اینکه از R1 ارسال می شود) می تواند میزان رسیدن به R2 را داشته باشد که تقریبا R است، ظرفیت لینک از R1 تا R2، علی رغم وجو مقدار لاندا ورودی بایت/ثانیه اگر لاندا پریم ورودی بایت/ثانیه برای تمام اتصالات بیش از حد بزرگ باشد (شامل اتصال B-D) سپس مقدار رسیدن ترافیک B-D به R2 می تواند خیلی بزرگتر از ترافیک A-C باشد. بدلیل اینکه A-C و B-D باید بخاطر مقدار محدود فضای بافر در مسیریاب R2 با یکدیگر رقابت کنند، مقدار ترافیک A-C که با موفقیت به R2 رسیده است (یعنی دچار سرریز بافر نشده است) کوچکتر و کوچکتر شده همانگونه که بارگذاری خواسته شده ی از B-D بزرگتر و بزرگتر می شود. در آن محدوده، بارگذاری درخواست شده به فضای لایتنهای نزدیک می شود، بافر خالی R2 فورا بوسیله ی یک بسته ی B-D پر می شود، و ظرفیت پذیرش اتصال A-C در R2 به صفر می رسد. این در حقیقت، تاکید می کند که ظرفیت پذیرش انتها به انتهای A-C در محدوده ی ترافیک سنگین به صفر می رسد.

دلیل کاهش متعاقب در ظرفیت پذیرش همراه با افزایش بارگذاری خواسته شده بدیهی، آشکار و قابل درک است. زمانی که مقدار کار تلف شده ای که بوسیله ی شبکه انجام شده است. در سناریوی ترافیک بالا که در بالا شرح داده شد، هروقت یک بسته

www.it98.ir

در دومین هاب مسیریاب رها شده باشد، کار انجام شده بوسیله ی اولین هاب / مسیریاب در ارسال بسته به دومین هاب/مسیریاب انجام می شود و شبکه نسبتا از کار می افتد.

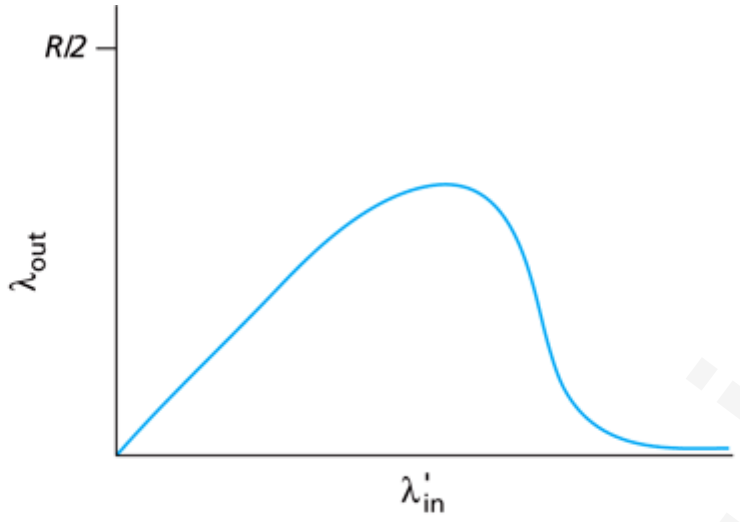


Figure 3.48 ♦ Scenario 3 performance with finite buffers and multihop paths

رویکرد دست یابی به کنترل ازدحام

در بخش سوم فصل 7، به صورت مفصل شیوه های خاص دست یابی Tcp به کنترل ازدحام را برسی خواهیم کرد. در اینجا ما هر دو شیوه گسترده دست یابی به کنترل ازدحام را که به کار بردن(امتحان می شوند) و تکنیک های خاص معماری و پروتوکل های ازدحام (تراکم) کنترل متضمن این شیوه ها را می شناسیم (تشخیص می دهیم). بر گسترده ترین سطح می توانیم بین شیوه ای دست یابی به کنترل ازدحام(کنترل) بواسط اینکه آیا طبقه(سطوح شبکه) هیچگونه پشتیبانی آشکاری از سطح انتقال به هدف کنترل ازدحام فراهم می کنند:

کنترل ازدحام سر به سر. در رویکرد کنترل سر به سر سطح شبکه هیچگونه پشتیبانی آشکاری از سطح از سطح انتقال به هدف کنترل ازدحام فراهم نمی کند. حتی حضور تراکم در شبکه باید با سیستم سر (انتهایی) مبنی بر وضوح مشهود شبکه (برای مثال اتلاف بسته ای و تاخیر) استنباط شود. در بخش 3.7 خواهیم دید که Tcp لزوما" باید این شیوه سر به سر را در برابر کنترل تراکم به کار گیرد، چون سطح Ip هیچگونه بازخوردی به سیستم آخر (سر) راجع به تراکم شبکه فراهم نمی کند. فقدان قسمت(قطعه) Tcp (همچنانکه با یک خبر وصول با وقفه یا سه جزئی تکراری نشان داده شده است) به عنوان نشانه کنترل تراکم گرفته شده است و از ایزو Tcp از اندازه پنجره اش می کاهد.

نیز طرح جدید تری برای تراکم Tcp خواهیم دید که از سفر رفت و برگشتی افزاینده مقدار تاخیر به عنوان شاخص تراکم افزاینده شبکه استفاده می کند.

کنترل تراکم پشتیبانی شبکه . با پشتیبانی کنترل تراکم شبکه (اجرای)،مولفه سطوح شبکه(که تعیین کننده مسیر است) باز خورد آشکاری به فرستنده راجع به چگونگی تراکم در شبکه فراهم می کند. ممکن است این بازخورد به سادگی ذره ای منفرد باشد که نشان دهنده تراکم در اتصال است.

این شیوه از اوایل {IBM SNA{Schwartz 1982 و شبکه معماران DEC DEC {jain 1989 . romarkishham

wWw.it98.ir

www.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

1990} گرفته شده بود اخیرا" برای شبکه های TCPLIP مطرح شده بود.{ Floyed TCP 1994,RFC 3168} ،و در کنترل تراکم سرعت ذره ای(مفید)موجود ABR) ATM) همانطور که در زیر مطرح شده نیز استفاده شده است. بازخورد شبکه پیچیده تر نیز موجود است. برای مثال یک نوع کنترل تراکم ATM ABR که به اختصار آن را برسی خواهیم کرد. به یک جهت نما اجازه می دهد که فرستنده را آشکارا از انتقال سرعت که آن (جهت نما) می تواند اتصال باقی مانده پشتیبانی کند آگاه کند.پروتکل {xcp katabi:2002} برای هر مبدا محاسبه بازخورد جهت نما را فراهم کند که در بسته سر ساز حمل می شوند را جع به اینکه چگونه آن مبدا باید سرعت انتقالش را افزایش یا کاهش دهد. برای کنترل تراکم پشتیبانی شبکه بطور نمونه اطلاعات تراکم بازخورد می شوند ممکن است بازخورد مستقیم از شبکه جهت نما به فرستنده فرستاده شود بطور نمونه این نوع اختار شکل بسته مسدود به خود می گیرد دومین شکل این اختار وقتی رخ می دهد که جهت نما مسیر یاب رشته ای را در بسته علامت گذاری و بخ شکل جدید در می آورد

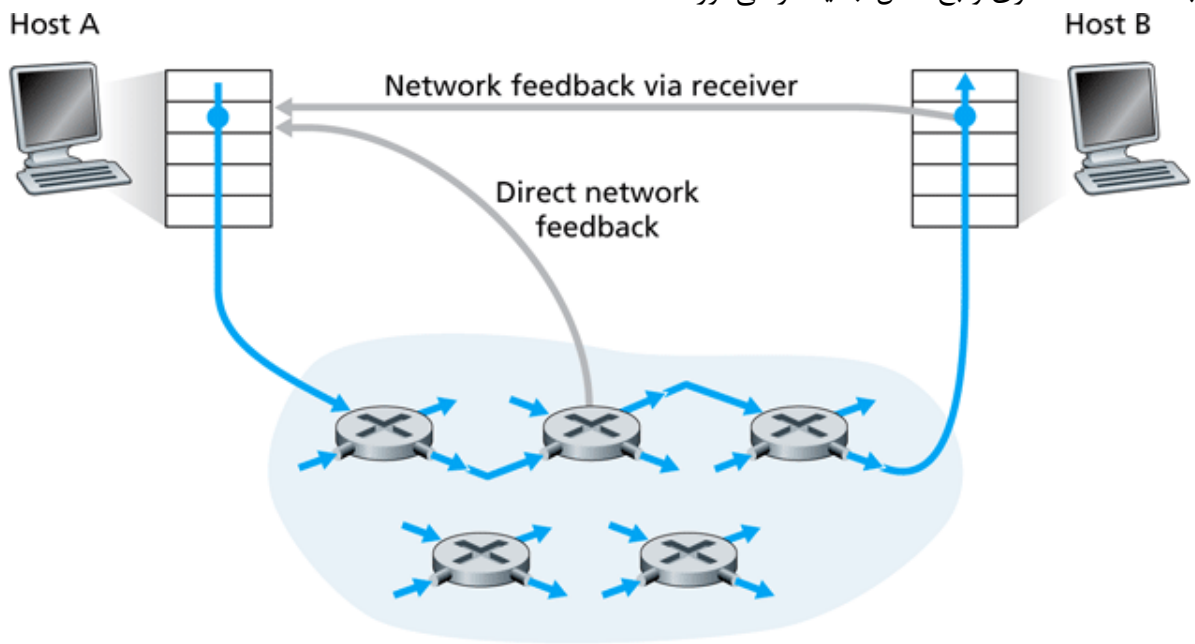


Figure 3.49 ♦ Two feedback pathways for network-indicated congestion information

دو مسیر بازخورد برای اطلاعات تراکم شبکه خبر دهنده(اخباری) شکل 3049
(درگردش از) فرستنده به گیرنده تا تراکم را نشان دهد .گیرنده به مجرد بسته ای نشان دار فرستنده را از نشانه های تراکم آگاه می کند به خاطر داشته باشید که شکل دومی اختار حداقل وقت زمان رفت برگشت کاملی می گیرد

3.6.3: مثال های شبکه پشتیبانی کنترل تراکم:

کنترل تراکمATM ABR این قسمت را با مطالعه مختصری از الگریتم کنترل تراکم در ATM ABR بروتکلی است که شیوه شبکه پشتیبانی را در مقابل کنترل تراکم به کار می گیرد به پایان می رسانیم

تاکید می کنیم که در اینجا هدف ما شرح مفصل معماری ATM نیست و هدفمان بیشتر نشان دادن پروتوکلی است که شیوه متفاوت خاصی را در مقابل کنترل تراکم از پروتکل TPC اینترنت به کار می گیرد در واقع در زیر فقط آن دسته از ویژگی های معماری ATM را که برای داشتن کنترل تراکم ABR لازم است مصرف می کنیم.

اساسا atm شیوه مدار مجازی راهنمایی (vc) در مقابل بسته های گزینشی به کار می رود

www.it98.ir

یاد آوری در فصل1: بدین معناست که هر گزینه (تعویض جریان) در مسیر مبدا به مقصد وضعیت را پیرامون موارد مجازی مبدا به مقصد نگه خواهد داشت این وضعیت مدار مجازی به تعویض جریان اجازه می دهد که حرکت منحصر به فرد فرستنده را دنبال کند (مثال): دنبال کردن سرعت انتقال میانگین وبرای به کار گرفتن مبدا مشخص جریان کنترل تراکم (مثل اخطار دادن آشکار به فرستنده برای کاهش سرعت وقتی تغییر جریان متراکم می شود.)

این وضعیت موارد مجازی در تغییر جریانات شبکه atmرا به صورت ایده آل برای اجرا کردن کنترل تراکم پشتیبانی شبکه مناسب میکند

ABRمانند خدمات انتقال اطلا عات داده های قابل ار تجاع (کشسان) طراحی شده اند به شیوه ای که یاد آور TCPاست .وقتی شبکه در حال بار گذاری است خدمات abr باید بتواند مزیت پهنای باند یدکی موجود را بکار گیرد وقتی که شبکه شلوغ است ، خدمات ABRباید سرعت انتقالش را کم وبیش در سرعتی از پیش تعیین شده به کنترل در آورد آموزش جامع گنترل تراکم ATM ABRومدیریت مخابره در ژوئن1996فراهم شده است شکل.3.50چارچوب کنترل تراکم ATM ABR را نشان می دهد . در این بررسی اصطلا حات ATM را اتخاذ می کنیم (به عنوان مثال کاربرد واژه سوئیچ به جای جهت نما وواژه سلول به جای بسته) خدمات ATM ABR باعث میشود سلول داده ها از طریق تغییر جریان از مبدا به مقصد منتقل شوند سلول های مدیریت منبع با داده ها INTER SPRET شده اند این سلول های RMمیتوانند برای انتقال اطلا عات مربوطه به تراکم در میان پذی رش گر وتغییر جریان به کار روند هنگامی که سلول RMبه مقصد می رسد تغییر جهت میدهد وبه فرستنده بازط می گردند (احتملا بعد از اینکه مقصد محتوی سلول RMرا تغییر میدهد

خود تغیر جریان همچنین می تواند یک سلول RM تولید کند و این سلول RMرا مستقیما" به یک مبدا بفرستد به این ترتیب سلول های RM می توانند هم بازخورد مستقیم شبکه و هم بازخورد شبکه از طریق دریافت شده به کار رود کنترل تراکمATM ABR رویکرد سرعت مناسب به این معنا که فرستنده

به وضوح سرعت ماکسیمی را محاسبه می کند که بر طبق آن می تواند خودش را در همان سرعت تنظیم کرده و بفرستد ABR سه مکانیزم برای نمایش دادن اطلاعات مربوت به تراکم از تغییر جریان به گیرنده به کار می برند

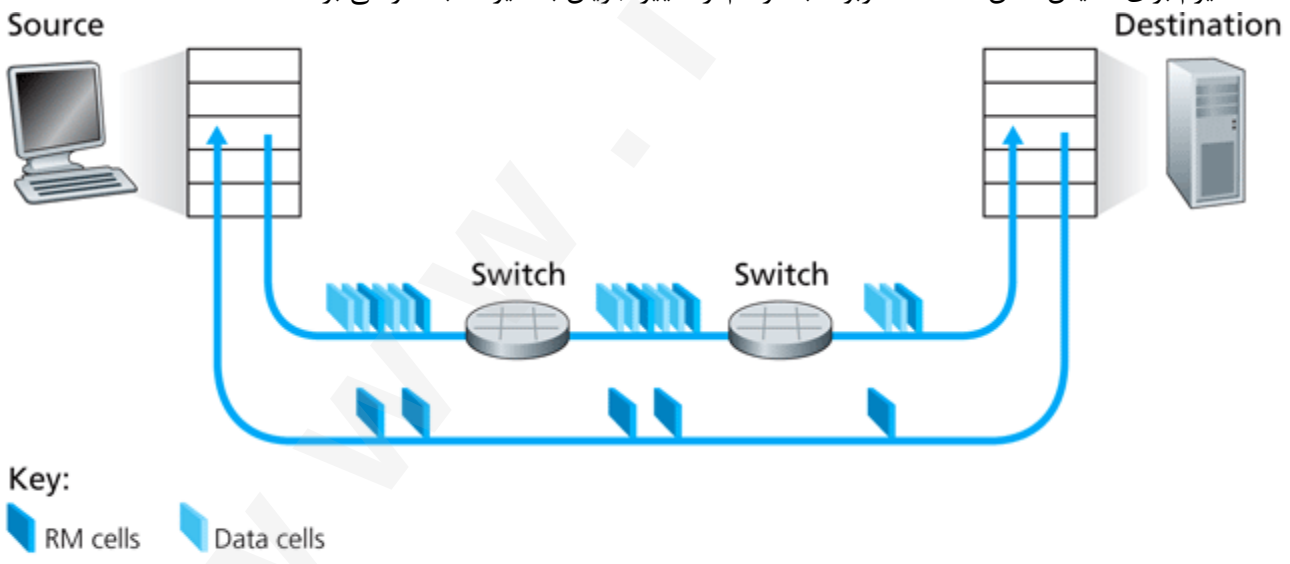


Figure 3.50 ♦ Congestion-control framework for ATM ABR service

ذره EFCI-

wWw.it98.ir

انجمن فناوری اطلاعات
www.it98.ir

هر سلول داده شامل نمایش آشکار ارسال تراکم ذره ای efciاست تغیر جریان شبکه تراکم می تواند ذره EFCI را یک سلول داده قرار داده تا تراکم را به پذیرشگر مقصد نشان دهد. مقصد باید ذره ی EFCI را در همه ی اطلا عات دریافتی سلول بررسی کند. هنگامی که سلول RM به مقصد می رسد اگر آخرین سلول داده دریافتی ذره EFCI را در 1 قرار داده باشد، آنگاه مقصد علامت تراکم ذره ای (ذرهء CI) سلول RM را به 1 می دهد و سلول RMرا به فرستنده ارسال می کند. بدین گونه با استفاده ازEFCI در سلول های داده و ذره CI در سلول های RM فرستنده می تواند متوجه تراکمی در تغیر جریان شبکه شود.
ذرات CI و NI همانطور که در بالا ذکر شد سلول های RM فرستنده به دریافت کننده با سلول های داده پراکنده می شوند.
سرعت پراکنش سلول RM یک پرامتر منظم است که با قرار داد اعتباری از هر 32 سلول داده یک سلول RM ساخته می شود.
این سلول های RM یک نمایش تراکم ذره ای و یک ذره افزایش ناپذیر دارند که می توانند با یک تغییر جریان شبکه متراکم ایجاد شود . به ویژه تغیر جریان می تواند تحت تراکم ضعیفی ذرهء NIرا در سلول RM گذرنده به ذره CI را تحت حالت های تراکمی شدیدی در 1 قرار دهد. وقتی پذیرشگر مقصد یک سلول RM دریافت می کند سلول RM را همراه با ذرات CIو NI سالمش به فرستنده باز می گرداند(به جز اینکه ممکن است CI در نتیجه مکانیزم EFCI که در بالا شرح داده شد توسط مقصد در 1 قرار داده شود).
تنظیمات ER همچنین هر سلول RM شامل یک میدان 2بایتی با سرعت مشخص است. یک تغییر جریان متراکم ممکن است از اعتبار میدان ER در سلول RM گذرنده بکاهد. در این روش ، میدان ERبر پایین ترین سرعت پشتیبانی همه تغییر جریانات برمسیر مبدا به مقصد قرار می گیرد.

کنترل تراکم TCP

7-3 در این قسمت دوباره به مطالعه TCP می پردازیم همانطور که در بخش 5-3 آموختیم ،TPC خدمات انتقال معتبری بین دو پروسه جاری در پذیرشگرهای مختلف را میسر می سازد.
یکی دیگر از اجزای TPC مکانیزم کنترل تراکم است . همانطور که در بخش قبل گفته شد TPC به جای کنترل تراکم شبکه پشتیبانی باید از کنترل تراکم آنها استفاده کند زیرا لایه IP هیچ بازخورد آشکاری راجع به کنترل تراکم سیستم های پایان فراهم نمی کند.
رویکرد به کار گرفته شده توسط TCP این است که هر فرستنده ای به عنوان عملکرد تراکم شبکه دریافت کننده سرعتی را که در آن اطلاعات را به رابطش ارسال می کند معین کننده اگر یک فرستندهTPC این اطلاعات را دریافت کند تراکم جزئی در مسیر بین خودش و مقصد به وجود می آید سپس فرستنده TPC سرعت ارسالش را افزایش می دهد و اگر فرستنده آن را دریافت کند در طول مسیر تراکم به وجود می آید سپس فرستنده سرعت ارسالش را کاهش می دهد . اما این رویکرد 3 پرسش ایجاد می کند: نخست اینکه چگونه یک فرستنده TPC سرعتی را که در آن اطلاعات را به رابطش ارسال می کند معین می کند؟ دوم اینکه چگونه یک فرستنده TPC متوجه می شود که در مسیر بین خودش و مقصد تراکم وجود دارد؟ و سوم اینکه فرستنده باید چه الگوریتمی را برای تغییر سرعتش به عنوان عملکرد تراکم انتها به انتهای دریافت به کار گیرد؟ این 3 موضوع را در ضمینه الگریتم کنترل تراکمTCP Reno که در بیشتر سیستم عامل های مدرن به کار گرفته شده اند . برای روشن شدن این موضوع فرض می کنیم که فرستنده TPC پرونده کاملی فرستاده است. بگذارید نخست این را برسی کنیم که چگونه یک فرستنده TPC سرعتی که در آن اطلاعات به رابطش ارسال می کند معین می کند.
در بخش 5-3 دیدیم که هر قسمت از TPC از یک محافظ دریافت و یک محافظ ارسال و چندین متغیر(last byte read، Rcv window و غیره) تشکیل شده است. مکانیزم کنترل تراکم TPC در هر قسمت اتصال بر مسیر یک متغیر اضافی ادامه می دهد. پنجره تراکم که به معنی congwinاست بر سرعتی که یک فرستنده TCP می تواند اطلاعات را به شبکه ارسال کند

www.it98.ir

شار اعمال می کند . بویژه اینکه ممکن است ، میزان داده ای وصول نشده در یک فرستنده از مینیمم congwinو Rcvwindow تجاوز نکند بدین معنی که:

$$\{\text{Last Byte sent_ last Bytencked} \leq \min\{\text{congwin Rcuwindow}\}$$

بگذارید از این پس برای کنترل تراکم (به عنوان مخالف روند کنترل) فرض کنیم که محافظ دریافت TCP به قدری کامل است که پنجره دیافت فشار می تواند نا دیده گرفته شود پس میزان داده های تعیین نشده در فرستنده منحصرا" توسط congwin محدود شده است.

فشار ذکر شده در بالامیزان داده های فرستنده را محدود می کند وبنا بر این به صورت غیر مستقیم سرعت ار سال فرستنده را محدود می کند برای روشن شدن ای موضوعرابطه ای را در نظر بگیرید که در آن فقدان وتاخیر انتقال بسته بی اهمیتند آنگاه فاز ، در ابتدای هر rtt، صریحا فرستنده را مجاز به ارسال conwin بایت دادهها به رابط میکند ودر پایان rtt فرستنده غیر وصول داده ها را در یافت می کند پس سرعت ارسال فرستنده اکیدا congwin/rtt بایت ثانوی است

از اینرو با تنظیم اغتبار congwir ، فرستنده می تواند سرعتی که در آن داده را به رابطش ارسال می کند تنظیم کند.

سپس بگذارید ملاحظه کنیم چگونه یک فرستنده TCP متوجه می شود که در مسیر بین مقصد و خودش تراکم وجود دارد.بگذارید یک اتلاف در یک فرستنده TCP هم به عنوان وقوع وقفه و هم دریافت کنیم ACK تکراری از گیرنده را تعریف کنیم.

(بحث گفت و گویمان را در بخش 3.5.4 وقوع وقفه در شکل 3.33و تغیر بعدی آن که باز فرستادن فوری پس از دیافت ACK تکراری را در بر دارد به یاد آورد)

هنگامی که تراکم بیش از اندازه است یک(سانتیمتر) در طول مسیر طغیان می کند و یک دیاگرام ایجاد می کند(شامل یک قطعه TPC) که افت می کند.

دیتا گرامی که افت کرده به نوبه خود یک رویداد اتلاف را در فرستنده موجب می شود هم وقفه وهم در یافت ack3 تکراری – که توسط فرستنده به عنوان نشانه یتراکم در مسیر فرستنده به گیرنده بکار گرفته می شود.

با در نظر گرفتن اینکه تراکم چگونه کشف شده است پس بگذاری د خوشبینانه ترین وضعیت را در نظر بگیریم که شبکه تراکم مطلق است یعنی وقتی رویداد اتلاف رخ نمیدهد در این وضعیت خبر وصول بخش های وصول نشده ی قبلی در فرستنده tcp در یافت خواهد شد.

همانطور که خواهیم دید TCP ورود این وصول را به عنوان نشانه ای بر صحت همه چیز به کار میگیرد آن بخش هایی که به شبکه منتقل شده اند با موفقیت به مقصد تحویل داده شده اند واز این خبر های وصول برای افزایش تراکنش واز این رو سرعت انتقالش استفاده می کند توجه کنید که اگر خبر های وصول با سرعتی نسبتا آهسته برسند اگر مسیر انتها به انتها تاخیر زیادی دارد یا شامل زنجیره های کوتاهی از اطلاعات است سپس پنجره تراکم با سرعتی نسبتا آهسته افزایش خواهد یافت از طرف دیگر اگر خبر های وصول با سر عت بالایی برسد پنجره تراکم سریعتر افزایش می یابد tcp را خود سنجش گر می گویند زیرا از خبر های وصول برای شروع افزایش اندازه ی پنجره تراکم

ما اکنون در موقعیتی هستیم که باید جزییات الگریتمی را که یک فرستنده tcp برای تنظیم سرعت ارسالش به عنوان عملکرد تراکم در یافتی استفاده می کند در نظر بگیریم . این الگریتم معروف ، الگریتم کنترل تراکم tcp است .

الگریتم 3جزء عمده دارد :

افزایش مفرط کاهش خبری

شروع آهسته

عکس العمل به رویدادهای وقفه انداز

wWw.it98.ir

www.it98.ir

افزایش مفرط وکاهش خبری :آگاهی اصلی کنترل تراکم tcp برای فرستنده است برای اینکه وقتی رویداد اتلاف رخ می دهد سرعت ارسالش را با کاهش اندازهی پنجرهی تراکم ، که همان congwin است کاهش دهد

از آنجا ئیکه که دیگر رابط های tcp که از مسیر های تراکم مشابه می گذرد هم متمایلند رویداد های اتلاف را تجزیه کنند ، آنها نیز متمایلند که سرعت ار سالشان را با افزایش congwin نشان دهند .

پس تاثیر کمی برای مبدا هایی است که برای کاهش سرعتی که در آن اطلا عات را به شبکه ارسال می کنند مسیر هایی به مسیر یاب های تراکم دارند که موقع تراکمن را در مسیر یاب های متراکم کاهش می دهد . اما وقتی یک رویداد اتلاف رخ می دهد یک فرستنده tpc تاچه اندازه در سایز پنجره تراکمش را کاهش می دهد ؟ TCP رویکردی که به اصتلاح کاهش ضربی نامیده می شود را به کار می برد و اعتبار متداول conhwin را بعد از رویداد اتلاف نصف می کند . بنابراین اگر اعتبار congwin در یک فرستنده Tpc به طور متداول 20 کیلوبایت است و وقفه پیدا شود تا congwin به دو نیمه 10 کیلوبایتی تبدیل می شود. اگر وقفه دیگر رخ دهد، congwin مجدادا" 5 کیلو بایت کاهش می یابد . ممکن است اعتبار congwin افت کردن را ادامه دهد اما مجاز به افت زیر 1mms نیست.

با تشریح اینکه چگونه یک فرستنده tcp سرعت ارسالش را به شکل تراکم آشکار کاهش می دهد طبیعی است که باید ملاحظه کنیم که وقتی tcp هیچ تراکمی دریافت نمی کند بدین معنا که وقتی Ackها در برابر داده ها در حال وصول پیشین می رسندtcp چگونه باید سرعت ارسالش را افزایش دهد. مبانی افزایش سرعت این است که اگر هیچ تراکم آشکاری وجود ندارد پس این احتمال وجود دارد که اطلاعاتی موجود است(و به کار نگرفته) که می تواند به صورت مضاعف توسط tcpاستفاده شود. در چنین شرایطی tcpآهسته پنجره تراکمش را افزایش می دهد که محتاطانه در جست و جوی اطلاعات اضافی موجود در مسیر انتها به انتهاست.

فرستنده tcp هر بار که خبر وصول دریافت می کند با توسعه ی اندک conhwin به منظور توسعه congwin به اندازه 1mms در هر زمان رفت و برگشت این کار را انجام می دهد.{ Rfc 2581} این می تواند به چندین طریق انجام شود : یک رویکرد مشترک برای فرستنده tcp این است که congwin را هر وقت که خبر وصول جدیدی می رسد به اندازه (Mss/congwin) Mssبایت ها افزایش دهد.

برای مثال اگر MMs 460 و 1بایت و congwin 600 و 14 بایت باشد آنگاه 10 قسمت بر RTT فرستاده می شوند هر ACK ورودی (فرض کنیم یکAck در هر قطعه) باعث افزایش اندازه پنجره تراکم به اندازه 1.10MMS می شود و از این رو پس از خبر وصول در هر 10 قسمت اعتبار پنجره تراکم به اندازه MMS افزایش می یابد.

www.it۹۸.ir
انجمن فناوری اطلاعات پیام نور