شبکه های کامپیوتری
فصل 5 - لایه انتقال - قسمت 2

وحید سلوک، سیامک سرمدی

1

فهرست مطالب

- خدمات انتقال اتصال گرا
  - معرفی پروتکل TCP
  - فرآیند کنترل جریان
  - مدیریت اتصال
- کنترل ازدحام

2

پروتکل TCP

- پروتکل TCP در RFC های 793,1122,1323, 2018, 2581 تشریح شده است.

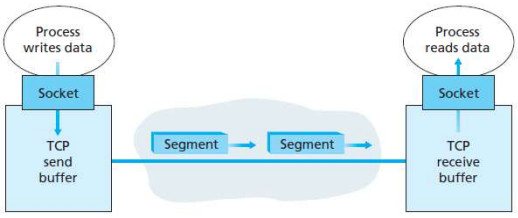
- پروتکل نقطه به نقطه
  - یک پردازش (سوکت) فرستنده به یک گیرنده
- جریان بایت دارای ترتیب، قابل اطمینان
- پروتکل خط لوله ای
  - اندازه پنجره با معیار های کنترل ازدحام و کنترل جریان تعیین می شود

- جریان داده دو طرفه کامل
  - مسیر دو طرفه داده
- اتصال گرا
  - نیاز به تبادل داده های کنترلی برای مباحثه و دست تکانی (Handshaking) پیش از تبادل و انتقال داده
- کنترل جریان
  - فرستنده در صورت آمادگی گیرنده ارسال می کند

3

پروتکل TCP

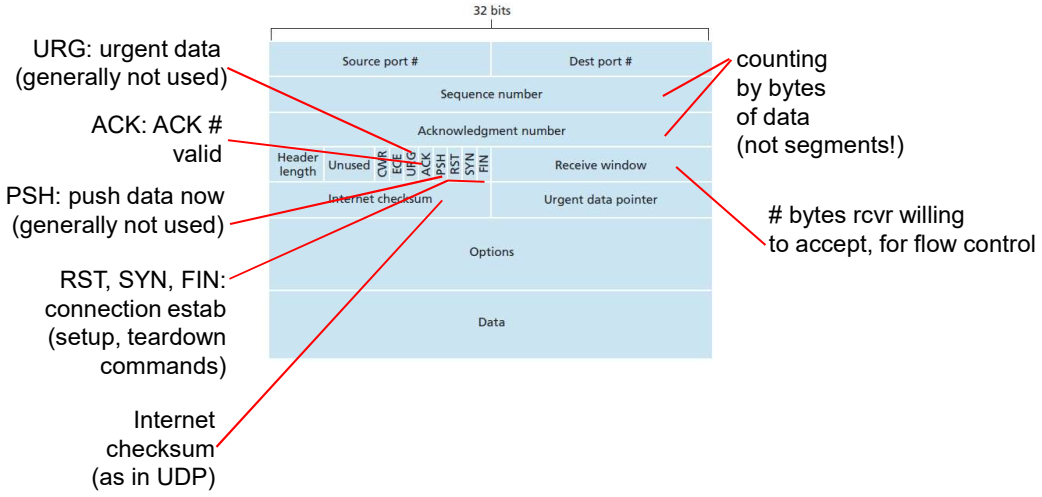
- **بافر ارسال:** وقتی داده ای تحویل TCP میشود، TCP آنرا در این بافر میگذارد. سپس هرچند مدت مقداری از آنرا برداشته و ارسال میکند. مطابق RFC793، TCP مقدار و زمان ارسال داده را مطابق میل خود تنظیم میکند.
  - ماکزیمم سایز داده قابل ارسال در سگمنت یا Maximum Segment Size – MSS (که شامل هدر 20 بایتی TCP نمیشود) با توجه به بزرگترین بسته لایه 2 قابل ارسال از مبداء تا مقصد که لازم به تکه کردن آن نباشد (- MaximumTransmission Unit MTU) تایین میگردد. بطوریکه یک سگمن بتواند در تنها یک فریم لایه 2 قرار گیرد.
  - سایز MTU برای ethernet و PPP معمولا 1500 بایت است که بنابراین سایز معمول MSS برابر با 1460 خواهد بود (40 بایت هدر TCP و IP)
  - برای کشف MTU مبداء تا مقصد روش هایی وجود دارد.
- **بافر دریافت:** در سمت گیرنده قرار دارد و وقتی سگمنتی فرا رسید در آن قرار میگیرد و بعد از مرتب شدن سگمنت ها، برنامه کاربردی داده را از آن میخواند.



4

ساختار قطعه TCP

- ساختار داده سگمنت TCP



5

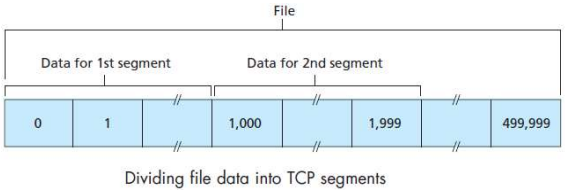
ساختار قطعه TCP

- **Source & Destination port:** آدرس های پورت مبداء و مقصد
- **Sequence # and Acknowledgment #:** برای شماره ترتیب سگمنت های ارسالی و تاییدیه برای سگمنت ها بکار میرود. بجای شماره بسته، بایت های ارسال شده یا تایید شده را مشخص میکند.
- **Receive Window:** تعداد بایتی که گیرنده میخواهد دریافت کند. برای کنترل جریان استفاده می شود.
- **Header Length:** طول هدر که به علت قسمت options متغییر است ولی اگر قسمت options استفاده نشود، طول هدر TCP برابر با 20 است. طول این قسمت هدر، 4 بیت است و تعداد بلوک های 32 بیتی را نشان میدهد (مقدار آن ضرب در 32 بیت میشود).
- **Options:** برای توافق بر سر سایز MSS، یاضریب سایز پنجره (در شبکه های پر سرعت) و کاربردهای دیگر مورد استفاده قرار می گیرد.
- **قسمت فلگ ها:**
  - ACK: نشان میدهد که عدد قرار گرفته در هئر # Acknowledgment معتبر است (برای مشخص کردن سگمنت صحیح دریافت شده)
  - RST، SYN، FIN: برای آغاز و اتمام یک ارتباط TCP مورد استفاده قرار میگیرند.
  - CWR، ECE: برای اطلاعیه ازدحام مورد استفاده قرار میگیرند.
  - PSH: به لایه TCP گیرنده میگوید که باید داده را سریعا به لایه بالا (کاربرد) تحویل دهد
  - URG: نشان میدهد که فرستنده داده ای را در داخل سگمنت گذاشته که فوری میداند. آخرین بایت داده فوری در فیلد urgent data pointer مشخص شده
  - فیلد های URG،PSH ، و urgent data pointer در عمل استفاده نمیشوند.

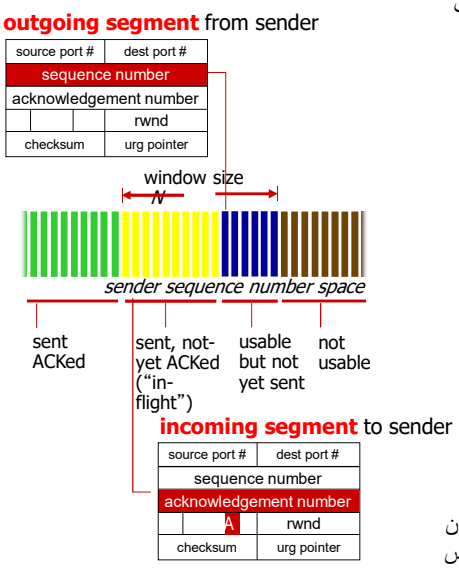
6

شماره ترتیب قطعه و پاسخ

- فرض کنید فایلی به طول 500000 بایت را ارسال کنیم و هر قطعه 1000 بایت است



Dividing file data into TCP segments



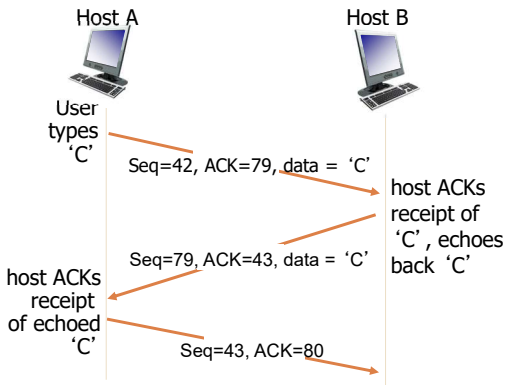
- **شماره ترتیب (در داده ارسالی):**
  - شماره اولین بایت ارسالی در هر سگمنت را مشخص میکند
    - شماره بایت در یک جریان داده :0، 1000، 2000، ...
- **شماره پاسخ (در تاییدیه):**
  - شماره بایت بعدی مورد انتظار توسط گیرنده
    - مثال: اگر سگمنت 0 دریافت شده، سگمنت 1000 مورد انتظار خواهد بود
  - ارسال پاسخ های تجمیع شده
    - مثال: اگر بسته های 0 و 2000 دریافت شده باشند، پاسخ همچنان بسته 1000 را بعنوان بسته مورد انتظار مشخص میکند. اگر بسته 1000 برسد، پاسخ بعدی 3000 را مشخص خواهد کرد.

7

شماره ترتیب – مثال

مثالی از شماره های ارسالی در داده ها و پاسخ ها، در شکل زیر نشان داده شده است.

- شماره ترتیب: در عمل هر کدام از طرفین یک شماره اتفاقی را بعنوان شماره ترتیب شروع انتخاب میکنند
  - (A: 42 و B: 79).
- بسته های پاسخ سوار بر بسته های داده ارسالی از سرور به کلاینت ارسال میشوند.
  - پاسخ B به بسته اول ارسال شده توسط A دو هدف دارد. ارسال تاییدیه که در قسمت هدر TCP انجام میگیرد و همینطور ارسال اکوی (echo) رشته دریافتی که در بدنه سگمنت ارسال میشود.



simple telnet scenario

8

تخمین زدن زمان مناسب برای تایمرانقضاء

- تخمین دقیق زمان مناسب تایمر انقضاء کار دشواری است.
  - تایمر انقضاء باید بزرگتر از RTT باشد (زمان رفت و برگشت به اضافه زمان پردازش و آماده سازی پاسخ در گیرنده).
- زمان رفت-برگشت نمونه (SampleRTT):
  - مدت زمان بین ارسال یک قطعه نمونه و دریافت پاسخ آن
  - در هر زمان فقط یک نمونه از بین قطعات ارسال شده ولی تایید نشده برای این کار انتخاب می شود.
  - "قطعات ارسال مجدد شده" برای محاسبه زمان RTT نمونه مورد استفاده قرار نمی گیرند.
  - این زمان ها برای بسته های متفاوت بر اشاش ازدحام فعلی و مقدار داده داخل قطعه متفاوت خواهد بود.
- TCP برای تخمین RTT یک مقدار میانگین از زمان های RTT نمونه ها با نام EstimatedRTT نگهداری می کند:

$$\text{EstimatedRTT} = (1-\alpha).\,\text{EstimatedRTT} + \alpha.\,\text{SampleRTT}$$

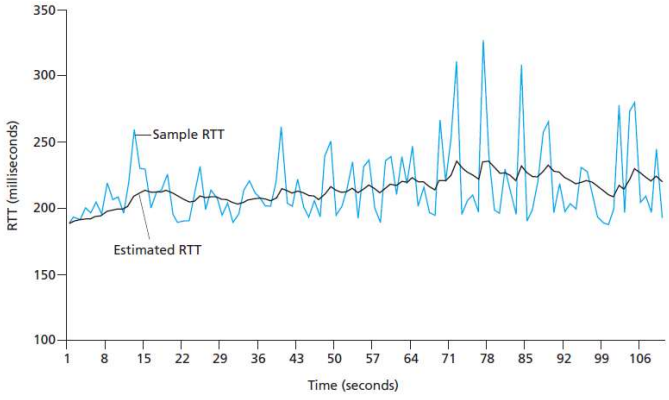
- مقدار پیشنهادی برای α مقدار $1/8$ یا 0.125 است:

$$\text{EstimatedRTT} = 0.875 \times \text{EstimatedRTT} + 0.125 \times \text{SampleRTT}$$

9

ملاحظات RTT و تایمر انقضاء

- در آمار، میانگین گیری با روش ذکر شده را "میانگین متحرک با وزن نمایی" (EWMA) می نامند.
  - هدف از این روش کاهش اثر نمونه های قبلی (بصورت نمایی) است.
    - نمودار زیر زمان RTT نمونه و تخمینی را برای قطعه های ارسالی بین یک میزبان در فرانسه و میزبانی در آمریکا را نمایش میدهد



RTT samples and RTT estimates

10

انحراف از RTT و بازه انقضاء تایمر

- **انحراف از RTT:** علاوه بر EstimatedRTT، محاسبه انحراف از RTT که با روش زیر محاسبه میشود مفید است:

$$DevRTT = (1 - \beta).DevRTT + \beta.|SampleRTT - EstimatedRTT|$$

(typically, β = 0.25)

- **بازه انقضاء (Timeout):** عبارت از زمان تخمینی RTT و یک حاشیه اطمینان (برای جلوگیری از تکرار ارسال بیمورد) است.
  - حاشیه اطمینان معمولا چهار برابرانحراف RTT در نظر گرفته میشود.
  - حاشیه اطمینان با افزایش انحراف RTT افزایش می یابد
  - در ابتدای ارسال، مقدار TimeoutInterval=1s در RFC6298 پیشنهاد شده است.

TimeoutInterval = EstimatedRTT + 4*DevRTT

11

طراحی TCP

- **مشخصات:**
  - بهره گیری از سرویس rdt روی کانال غیر قابل اطمینان IP
  - استفاده از قطعات خط لوله ای
  - استفاده از پاسخ تجمیع شده (بسته های قبل از شماره ترتیب تایید شده هم تایید می شوند)
  - دارای فقط یک تایمر ارسال مجدد
  - رخدادهای ارسال مجدد
    - انقضاء تایمر
    - پاسخ های تکراری
- **طراحی اولیه:** ساده ترین مدل
  - فرض ها
    - بدون پاسخ تکراری
    - بدون کنترل جریان
    - بدون کنترل ازدحام

12

رخدادهای TCP

- درخواست ارسال (تحویل داده از لایه کاربرد)
  - ساخت قطعه با شماره ترتیب
  - مقدار شماره ترتیب: شماره اولین بایت جریان داده در قطعه
- آغاز تایمر (اگر قبلا در حال کار نبود)
  - در صورت فعال بودن تایمر، تایمر روی قدیمی ترین قطعه بدون پاسخ تنظیم می شود
  - محاسبه بازه انقضاء
- انقضاء
  - ارسال مجدد قطعه ای که باعث انقضاء تایمر شده
  - شروع مجدد تایمر
- دریافت پاسخ
  - اگر شماره پاسخ قبلا دریافت نشده
  - به روز رسانی وضعیت پنجره
  - شروع تایمر در صورت وجود قطعات بدون پاسخ در پنجره (بافر)

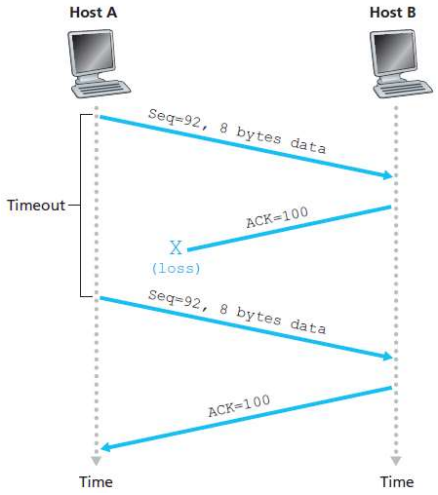
13

TCP Pseudo-code

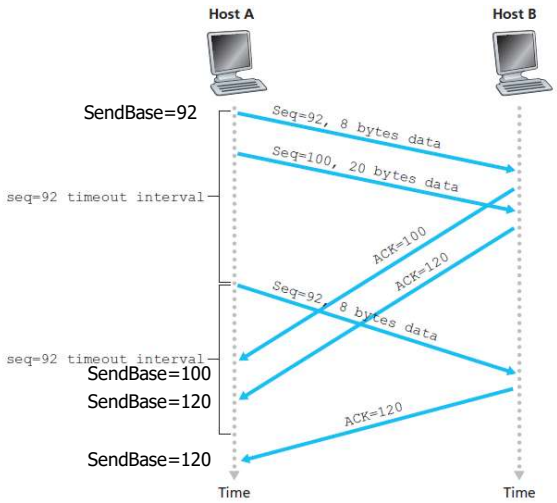
14

سناریو - ارسال مجدد

- پیام های تبادل شده در وضعیت تلف شدن پاسخ و تایم آوت رودرس (قبل از رسیدن تایید)



lost ACK scenario

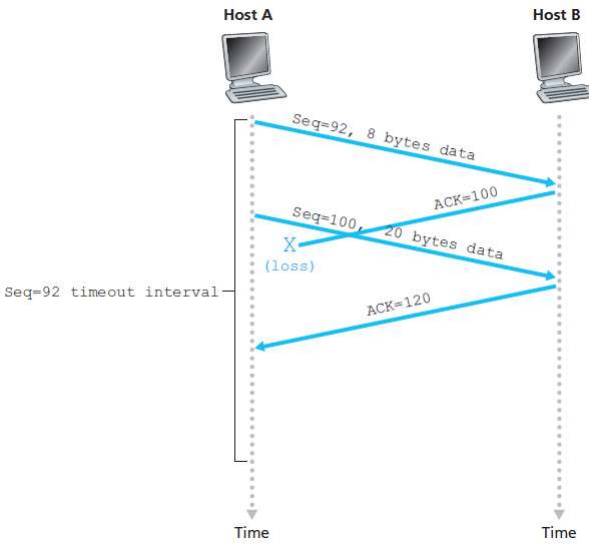


premature timeout

15

سناریو - ارسال مجدد

- پیام های تبادل شده در وضعیت پاسخ تجمیعی



- **نکته:** در هنگام تکرار ارسال زمان تایم آوت از روی فرمول بحث شده انتخاب نمیشود، و 2 برابر زمان ارسال قبل در نظر میگیرد.
  - اینکار یک مکانیزم جلوگیری از ازدحام ایجاد میکند (چرا؟)

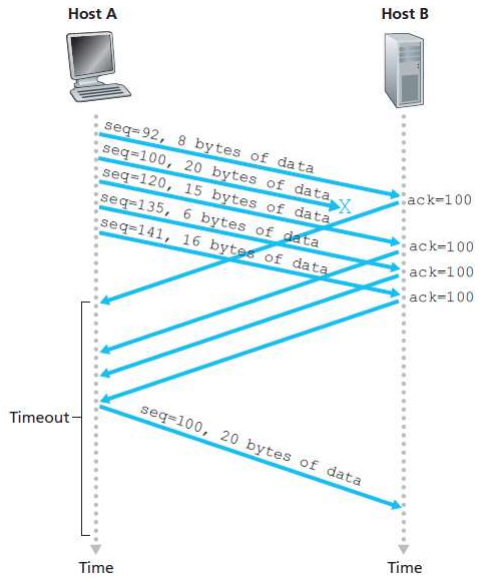
16

ارسال مجدد سریع

- مقدار تایمر انقضاء ممکن است طولانی باشد و باعث تاخیر طولانی قبل از ارسال مجدد بسته گم شده بشود.
  - این تاخیر در لایه کاربردی، غیر قابل تحمل خواهد بود.
  - نیاز به فضای بافر زیاد برای نگهداری قطعات خارج از ترتیب خواهیم داشت.
- راه کار
  - تشخیص قطعات گم شده توسط پاسخ تکراری
    - تایید تکراری برای بسته قبلی که صحیح دریافت شده و تایید هم شده بود
  - در صورت گم شدن با رسیدن بسته های جدید خارج از ترتیب، تعداد پاسخ های تکراری بیشتری برای بسته "سالم و مرتب رسیده" قبلی ارسال می شود.

17

ارسال مجدد سریع



- استاندارد
  - در صورت دریافت بیش از 3 پاسخ تکراری در فرستنده ← ارسال مجدد قطعه با کوچکترین شماره ترتیب
  - فرض: قدیمی ترین ارسال از یک قطعه گم شده ← عدم انتظار برای انقضاء تایمر

18

ساخت پاسخ TCP

- رخدادهای گیرنده
  - ورود قطعه ای با شماره ترتیب مورد انتظار (تمامی داده های قبلی پاسخ داده شده اند)
  - ورود قطعه ای با شماره ترتیب مورد انتظار (پاسخ یک قطعه قبلی ارسال نشده)
  - ورود قطعه خارج از نوبت با شماره ترتیب بزرگتر از مورد انتظار (تشخیص فاصله)
  - ورود قطعه میانی (برای پوشش کل یا بخشی از فاصله موجود در پنجره گیرنده)
- عملیات گیرنده
  - توقف پاسخ، انتظار بمدت 500 ms برای قطعه بعدی. ارسال پاسخ در صورت عدم دریافت قطعه
  - ارسال بلافاصله پاسخ تجمیع شده، برای پاسخ هر دو قطعه
  - ارسال بلافاصله پاسخ تکراری، با شماره بایت بعدی مورد انتظار
  - در صورتیکه شروع قطعه از ابتدای فاصله باشد، ارسال بلافاصله پاسخ

19

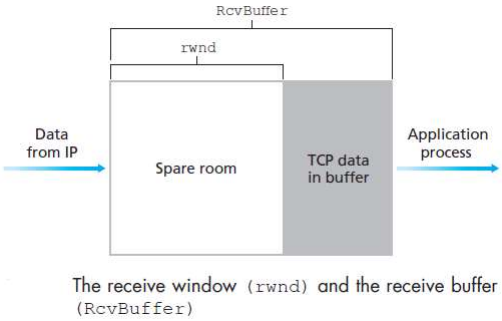
کنترل جریان و کنترل ازدحام

- هر سمت از یک ارتباط TCP یک بافر دریافت (receive buffer) دارند که داده های رسیده درآن قرار میگیرند.
  - نرم افزار لایه کاربرد هر از چندگاه داده ها را از این بافر میخواند.
  - ولی ممکن است به علت مشغول بودن به یک کار دیگر، برنامه کاربردی مدت طولانی، داده ای را نخواند.
  - درصورتیکه فرستنده به ارسال اطلاعات ادامه دهد، بافر دریافت به سرعت سر ریز (overflow) میشود.
- **کنترل جریان:** برای جلوگیری از سر ریز شدن بافر دریافت از یک مکانیزم کنترل جریان استفاده میشود.
  - در این مکانیزم، سمت گیرنده با هر پیام تایید، مقدار فضای باقیمانده در بافر خود را (مقدار داده ای که حاضر است قلول کند) به اطلاع فرستنده میرساند. فرستنده سایز پنجره ارسال خود را به نحوی تنظیم میکند که داده ای بیش از مقدار قابل ذخیره در سمت گیرنده را ارسال نکند.
- **کنترل ازدحام:** مکانیزم مشابهی است که سرعت ارسال فرستنده را کند میکند.
  - در این مکانیزم، سایز پنجره اجازه ارسال تعداد محدودی قطعه قبل از دریافت تایید برای آنها را میدهد. درصورتیکه تعداد زیادی بسته تایید نشده در سمت فرستنده بافر شده باشد، بسته جدیدی ارسال نمیشود.
- **مقایسه:** مکانیزم های فوق شباهت هایی دارند (هردو مقدار ارسال داده را کنترل میکنند) ولی فرق اساسی آنها در هدف کنترل است. یکی برای جلوگیری از پر شدن بافر گیرنده، و دیگری برای جلوگیری از ازدحام روی بافر فرستنده (و کانال) بکار میرود.

20

کنترل جریان

- اصول
  - **اعلام مقدار داده قابل قبول توسط گیرنده:** تصویر، بافر سمت گیرنده را نشان میدهد که به اندازه RcvBuffer جا دارد. مقدار فضای باقیمانده آن، rwnd است که در پاسخ ها به سمت گیرنده اعلام میشود.
    - در صورتیکه rwnd=0 باشد، سمت فرستنده میفهمد که نباید داده ارسال کند. ولی مشکل این است که چون بسته ای ارسال نمیکند و پاسخی هم دریافت نمیکند، نخواهد توانست هنگامی که بافر گیرنده خالی شد، اطلاع یابد.
    - برای همین فرستنده به ارسال بسته های خالی، ادامه میدهد و گیرنده نیز به آنها پاسخ میدهد. درصورت خالی شدن مجدد بافر گیرنده، فرستنده خواهد توانست ارسال انجام دهد.
  - **محدود کردن تعداد قطعه بدون پاسخ در فرستنده:** علاوه بر ارسال داده به اندازه درخواستی توسط گیرنده، این مکانیزم نیز که بیشتر برای کنترل ازدحام است، نیز به کنترل جریان کمک میکند.

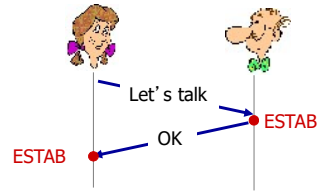


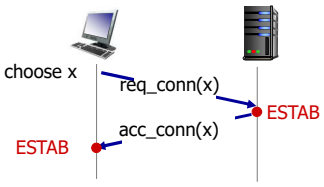
The receive window (rwnd) and the receive buffer (RcvBuffer)

21

مدیریت اتصال

- سرویس اتصال گرا: نیاز به برقراری اتصال پیش از انتقال داده (Handshake) است
  - توافق بر پذیرش اتصال
  - توافق بر پارامترهای اتصال
- نیاز به دست تکانی سه مرحله ای داریم





22

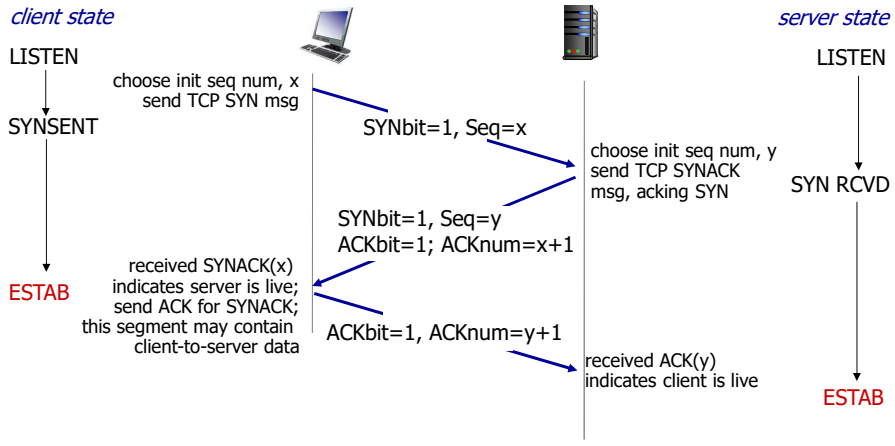
دست تکانی سه مرحله ای

- **مرحله 1:** مشتری یک بسته TCP بدون داده که بیت SYN آن 1 است به سرویس دهنده ارسال میکند.
  - مشتری همچنین یک شماره ترتیب ابتدایی انتخاب کرده و در قطعه میگذارد
  - این قطعه به قطعه SYN معروف است.
- **مرحله 2:** سرویس دهنده قطعه را دریافت کرده و سپس:
  - بافر و متغییر های لازم را به ارتباط اختصاص میدهد
  - یک سگمنت SYN که مقدار ACK آن نیز به اندازه "شماره ترتیب مشتری + 1" تنظیم شده می سازد.
  - یک شماره ترتیب تصادفی در آن بعنوان شماره ترتیب ابتدایی سرویس دهنده در سگمنت قرار میدهد
  - سگمنت را به فرستنده میفرستد. این قطعه به نام قطعه SYNACK شناخته میشود.
- **مرحله 3:** با دریافت سگمنت SYNACK مشتری:
  - بافر و متغییر های لازم را ایجاد میکند
  - سگمنت دیگری را به سرویس دهنده ارسال میکند که درآن SYN=0 و مقدار #Acknwoledgment به اندازه "شماره ترتیب سرور + 1" تنظیم شده.
  - این بسته میتواند شامل داده مشتری به سرور نیز باشد.

23

دست تکانی سه مرحله ای

- فرآیند سه مرحله ای و پیامهای تبادل شده برای برقراری ارتباط (Connection Establishment)



24

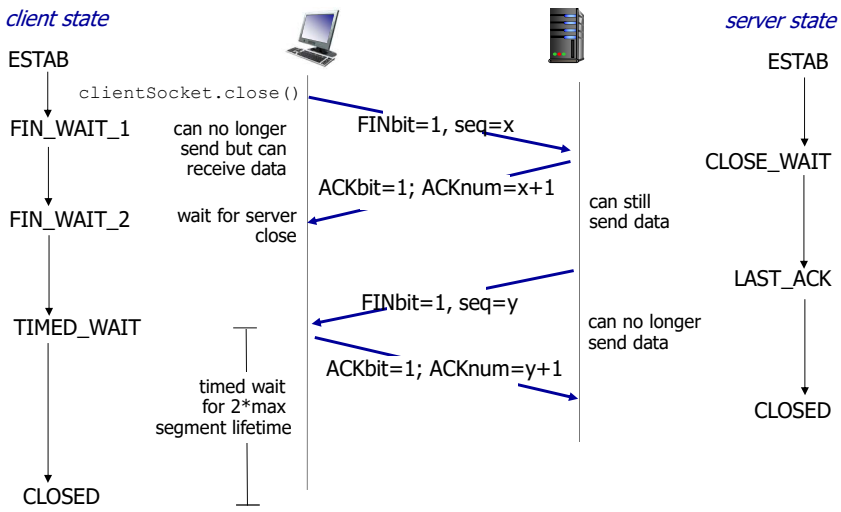
مراحل قطع اتصال

- **مرحله 1:** مشتری یک بسته TCP بدون داده که بیت FIN آن 1 شده به سرویس دهنده ارسال میکند.
- **مرحله 2:** سرویس دهنده قطعه را دریافت کرده و پاسخی با یک سگمنت ACK برای آن ارسال میکند.
- **مرحله 3:** سپس سرور نیز یک قطعه FIN به مشترس ارسال میکند.
- مرحله 4: مشتری نیز دریافت قطعه FIN را با پاسخی بصورت قطعه ACK جواب میدهد.
- در انتها هر دو سمت، بافرها و متغییر ها را آزاد میکنند.

25

مراحل قطع اتصال

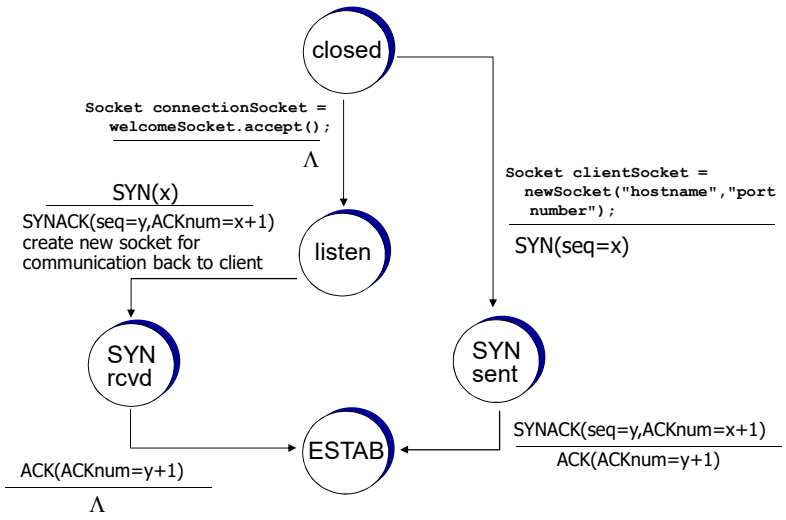
- فرآیند و پیامهای رد و بدل شده برای قطع ارتباط



26

دست تکانی سه مرحله ای: FSM

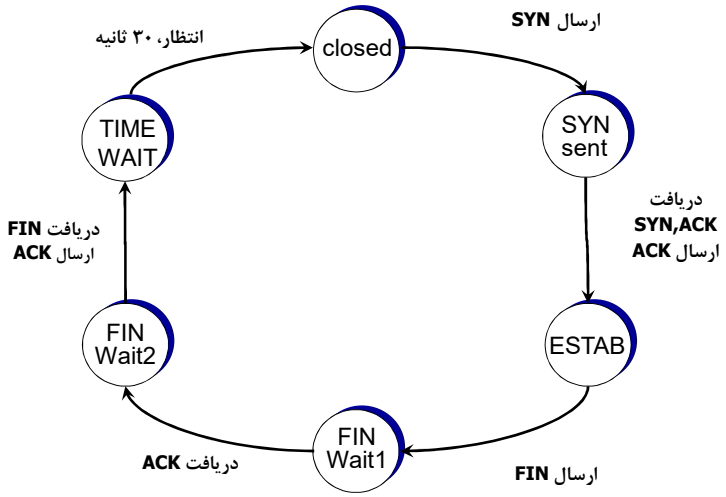
- نمودار FSM ایجاد ارتباط



27

FSM: وضعیت مشتری

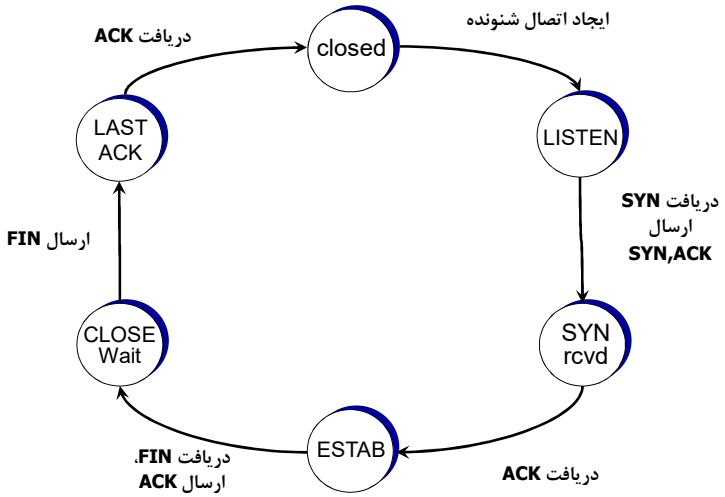
- نمودار FSM در سمت مشتری



28

FSM: وضعیت سرویس دهنده

- نمودار FSM در سمت سرویس دهنده



29

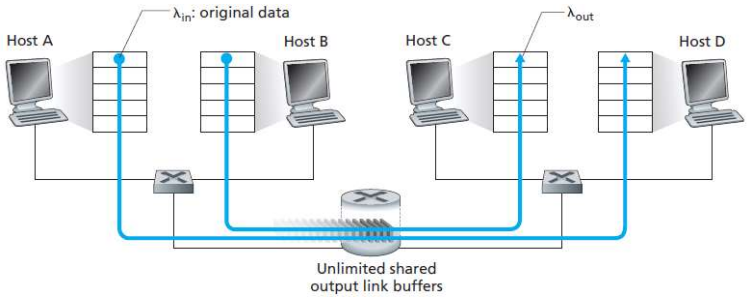
کنترل ازدحام

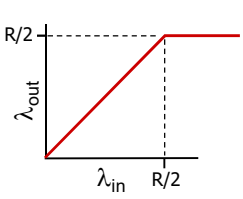
- منشاء عمده تلفات بسته
  - سرریز شدن بافر در مسیریاب ها به دلیل ازدحام در بافرهای ارسال آنها
- راه کار برای جبران تلفات بسته
  - ارسال مجدد
- لازمه کنترل ازدحام
  - فرآیندی برای کنترل ارسال فرستنده
- سوال: آیا فرآیند کنترل جریان قادر به حل مشکل ازدحام نیز هست هست؟

30

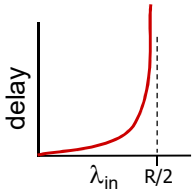
سناریوی 1: دو فرستنده، یک مسیریاب و بافر نامحدود

- فرض ها
  - دو فرستنده به دو گیرنده پیام ارسال میکنند و نرخ داده خروجی هر دو λ_{in} است.
  - فضای بافر مسیریاب نامحدود
  - ظرفیت لینک خروجی فیزیکی R است
  - پروتکل انتقال ساده (بدون کنترل ها)
  - از اورهد، سرآمد لایه ها چشم پوشی میکنیم.
- اگر هر میزبان با نرخ R/2 داده ارسال کند، ظرفیت لینک تمام میشود و داده بیشتری ارسال نمیشود
- به علت نامحدود بودن فضای بافر، انتظار و تاخیر بسته ها نیز به سمت بی نهایت میل میکند.





حداکثر نرخ ارسال هر فرستنده: **R/2**

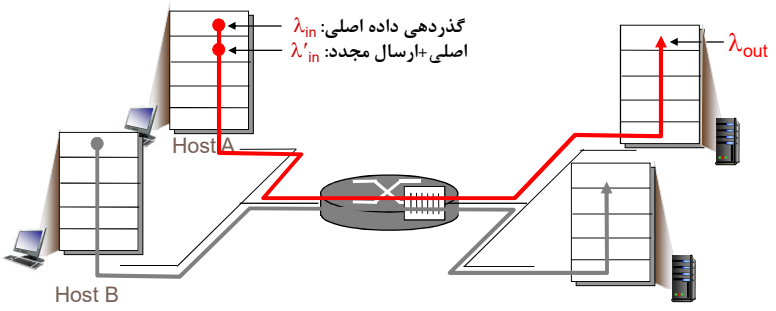


افزایش تاخیر با افزایش نرخ دریافت داده به نرخ ارسال λ_{in}

31

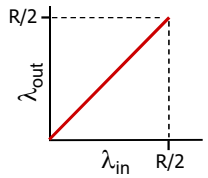
سناریوی 2: بافر محدود، ارسال مجدد

- فرض ها
  - فضای بافر مسیریاب محدود: بسته های جا نگرفته در بافر ارسال تلف میشوند
  - ارسال مجدد بسته های منقضی شده (تلف شده بخاطر پر شدن بافر ارسال)
  - با توجه به تکرار بسته ها، داده ارسالی شامل داده های تکراری است. پس نرخ ارسال داده واقعی (غیر از تکراری ها) کمتر از داده ای است که روی کانال ارسال میشود.
  - نرخ لایه کاربرد: ورودی = خروجی
  - نرخ لایه انتقال: ورودی ≤ خروجی
  - Offered load: λ'_{in}
  - Original Data: λ_{in}

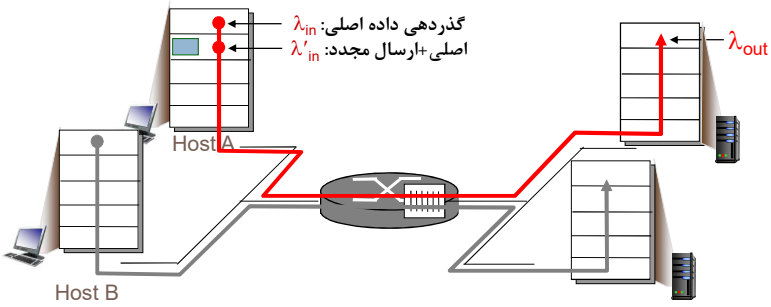


32

سناریوی 2: با دانش فرستنده از وضعیت بافر

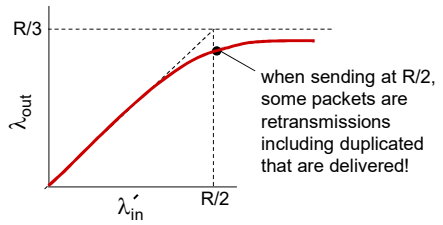


- **اطلاع فرستنده از فضای بافر موجود در مسیریاب:** فرستنده توان این را دارد که بداند آیا بافر پر است یا خالی! (وضعیت غیر ممکن فرضی)
- ارسال تنها در صورت خالی بودن بافر مسیریاب انجام میگیرد
- در اینصورت $\lambda_{in}=\lambda'_{in}$ (چون تلفاتی نداریم که ارسال مجدد صورت بگیرد)
- و گذردهی برابر با λ_{in} خواهد بود
- هزینه ازدحام: کاهش بار به دلیل عدم ارسال مجدد
- چون فرض کردیم تلفات نداریم، نرخ ارسال هر میزبان از R/2 نباید فراتر رود.

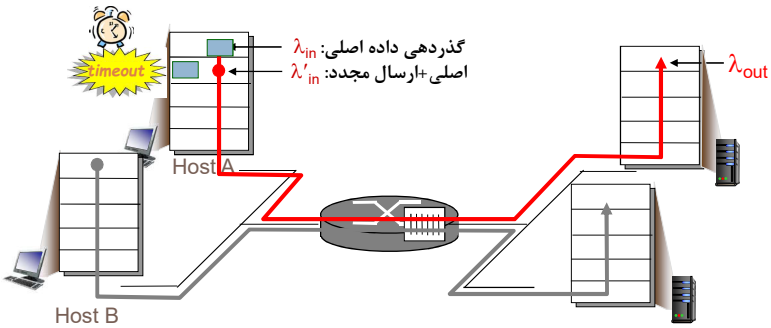


33

سناریوی 2: با ارسال تکراری



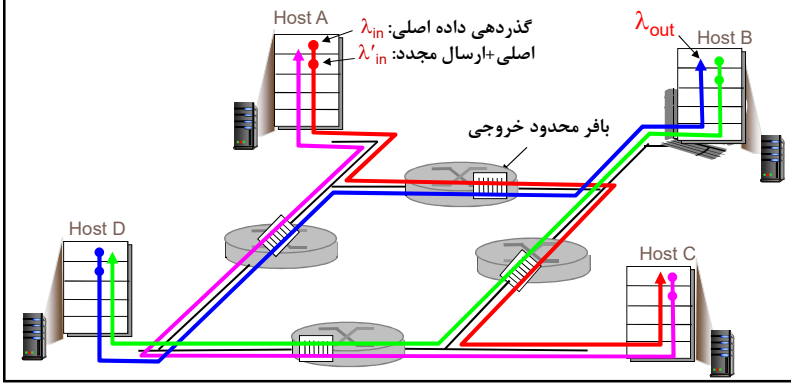
- **سناریوی کمی واقعی تر:** تکرار فقط در صورت اطمینان از تلف شدن بسته (باید زمان تایم آوت خیلی طولانی باشد تا اطمینان یابیم پیام پاسخ نیامده است)
- انقضای پیش از موعد تایمر فرستنده
- ارسال مجدد زودهنگام و دریافت بسته تکراری در گیرنده
- هزینه ازدحام: ارسال مجدد غیر ضروری
- قبل از رسیدن به نرخ R/2 ، به علت اشغال قسمتی از گذردهی خط برای ارسال تکراری، خروجی واقعی خط افت میکند

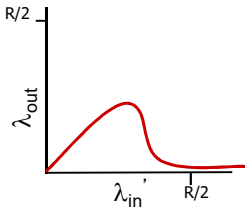


34

سناریوی 3: 4 فرستنده، چند گام

- در این سناریو دو میزبان از مسیرهای دارای همپوشانی به دو میزبان دیگر داده ارسال و دریافت میکنند. لینک ها همه ظرفیت R دارند.
  - هر 4 میزبان دارای مکانیزم تایمر و ارسال مجدد هستند.
  - بار تحویلی λ_{in} ناچیز ← عدم سرریزی بافر (افزایش مختصر λ_{in} به افزایش مشابه λ_{out} منجر می شود)
  - بار تحویلی زیاد ← بار عبوری (گذردهی) از مسیریاب کاهش می یابد (به علت ایجاد ازدحام روی لینک های مورد استفاده مشترک)
  - تکرار زیاد موجب کم و کمتر شدن داده واقعی ارسالی میشود.





35

راه کارهای کنترل ازدحام در TCP

- از آنجائیکه لایه شبکه (IP)، فیدبکی در مورد ازدحام در شبکه برای میزبانها ارائه نمیکند، وظیفه تشخیص ازدحام به عهده لایه TCP گذاشته شده که اینکار را بصورت end-to-end انجام میدهد. یعنی با توجه به وضعیت رسیدن بسته ها به نقطه مقصد، مقدار داده ارسالی را تنظیم میکند.
  - درصورتیکه مبداء احساس کند که ازدحامی در مسیر ارتباطی وجود ندارد، نرخ ارسال را افزایش میدهد.
  - در صورتیکه علائمی دال بر ازدحام دیده شود نرخ ارسال کاهش داده می شود.

- در مورد نحوه کار مکانیزم کنترل ازدحام TCP سه سوال مطرح است:
  - TCP چگونه نرخ ارسال ترافیک به یک اتصال (TCP Connection) را محدود میکند.
  - TCP چگونه از وجود ازدحام بین مبداء و مقصد مطلع میشود.
  - چه الگوریتمی باید برای تنظیم ترافیک ارسالی بعنوان تابعی از مقدار ازدحام مشاهده شده استفاده شود.

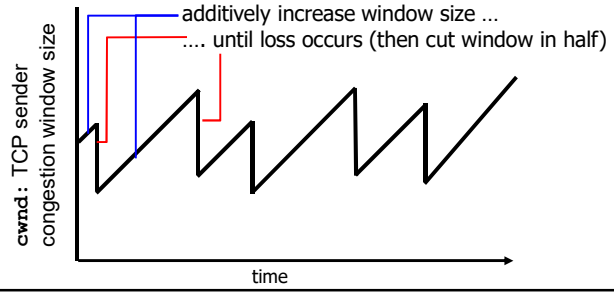
36

راه کارهای کنترل ازدحام در TCP – کنترل نرخ ارسال

- در هر سمت اتصال، بافرهای ارسال و دریافت و متغییر هایی (از قبیل LastByteRead، rwnd) هست.
  - **پنجره ازدحام:** مکانیزم کنترل ازدحام متغییر دیگری به نام پنجره ازدحام (congestion window) یا cwnd را نیز نگه میدارد.
  - **تحویل سالم بسته ها:** افزایش نرخ ارسال با افزایش cwnd
  - **رخداد خطا:** کاهش نسبتا سریع نرخ با نصف کردن اندازه cwnd
  - **در سمت فرستنده:** اندازه داده ارسالی (به نحوی اندازه پنجره ارسال) با توجه به سایز rwnd و cwnd تعیین میشود.

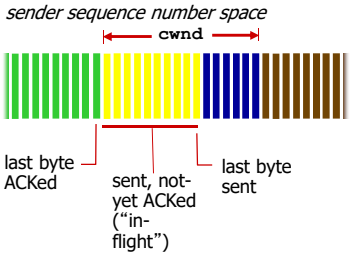
$$LastByteSent - LastByteAcked \leq \min\{rwnd, cwnd\}$$

AIMD saw tooth behavior: probing for bandwidth



37

راه کارهای کنترل ازدحام



- برای تمرکز روی کنترل ازدحام و درک آن، فرض کنید که بافر گیرنده آنقدر بزرگ است که محدودیت receive-window قابل چشم پوشی باشد (rwnd). بنابراین محدودیت در مقدار داده تایید نشده، در سمت فرستنده تنها بر اساس cwnd تنظیم میشود.

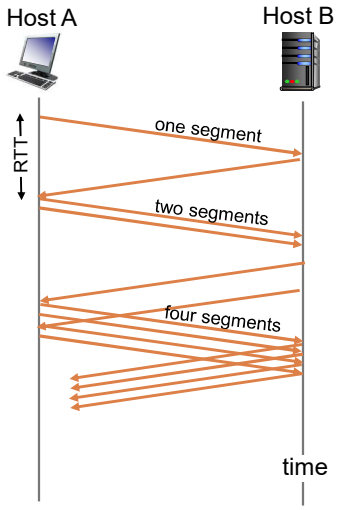
$$\mathbf{LastByteSent - LastByteAcked \leq cwnd}$$

- در این حالت پس از ارسال کل پنجره (به اندازه cwnd) به مدت RTT برای ACK انتظار کشیده میشود و درصورت رسیدن ACK ارسال مجدد ممکن میشود.

$$\mathbf{rate} \approx \frac{\mathbf{cwnd}}{\mathbf{RTT}}$$

38

شروع آهسته



- شروع ارسال در ابتدا با حداقل نرخ
  - یک MSS در مدت RTT
- افزایش دو برابری (نمایی) در هر بار

- **کاهش نرخ در صورت ازدحام:**
  - **روش 1:** کاهش سریع پنجره به 1 در صورت انقضای تایمر (شروع مجدد آهسته)
  - **روش 2:** کاهش پنجره به نصف آخرین مقدار

- **اجتناب از ازدحام:**
  - **روش 1:** افزایش 1 واحدی بجای نمایی
  - **روش 2:** افزایش MSS/cwnd

39