لایه انتقال (۲)

پروتکل های انتقال داده

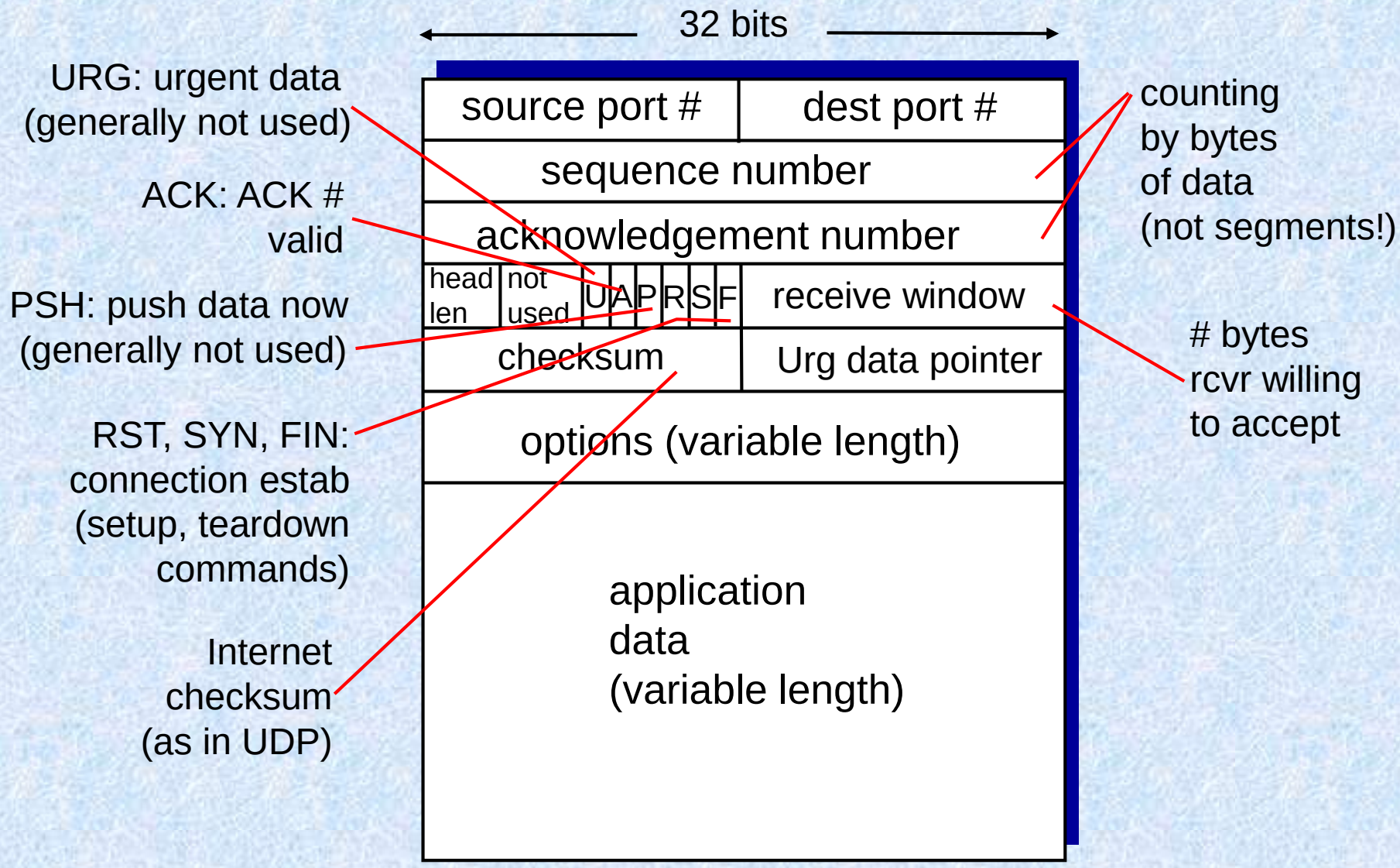
رئوس مطالب

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

TCP: Overview RFCs: 793,1122,1323, 2018, 2581

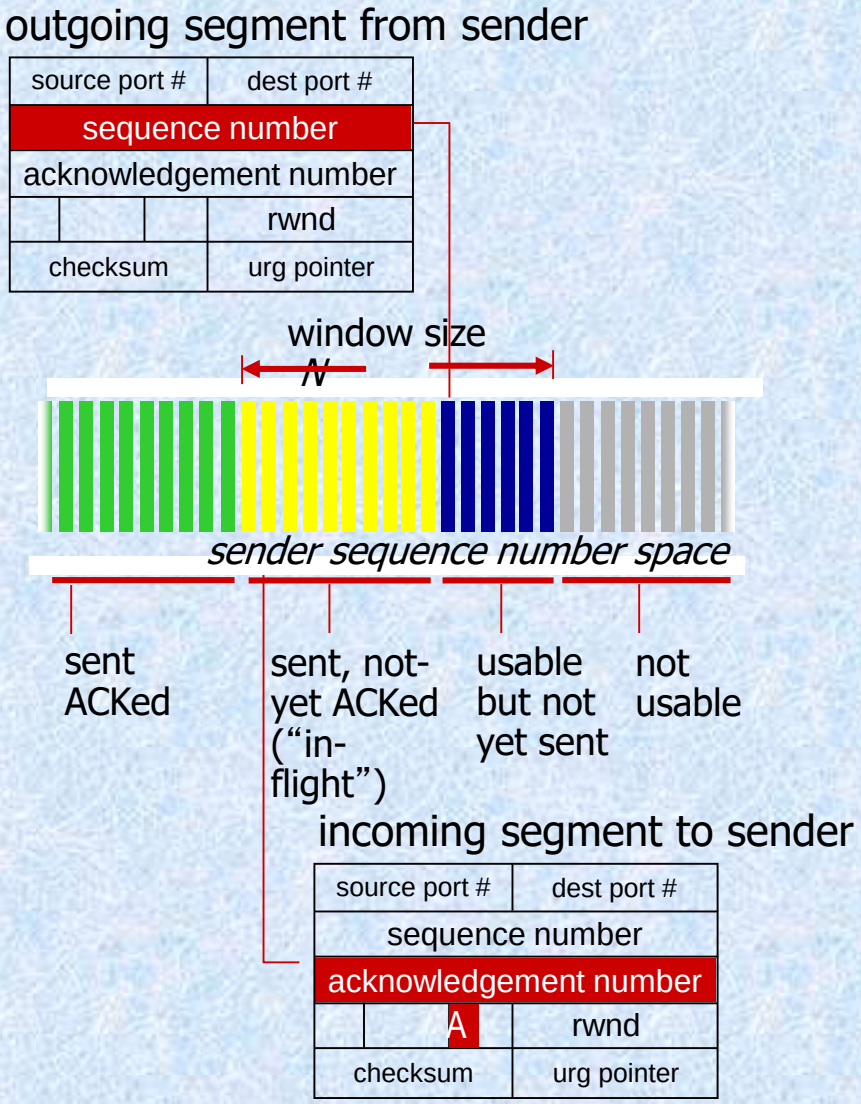
- پروتکل نقطه به نقطه
  - یک فرستنده به یک گیرنده
- جریان بایت دارای ترتیب، قابل اطمینان
- پروتکل خط لوله ای
  - اندازه پنجره با معیار های کنترل ازدحام و کنترل جریان تعیین می شود
- جریان داده دو طرفه کامل
  - مسیر دو طرفه داده
- اتصال گرا
  - نیاز به تبادل داده های کنترلی برای مباحثه و دست تکانی (**Handshaking**) پیش از تبادل و انتقال داده
- کنترل جریان
  - فرستنده در صورت آمادگی گیرنده ارسال می کند

ساختار قطعه TCP

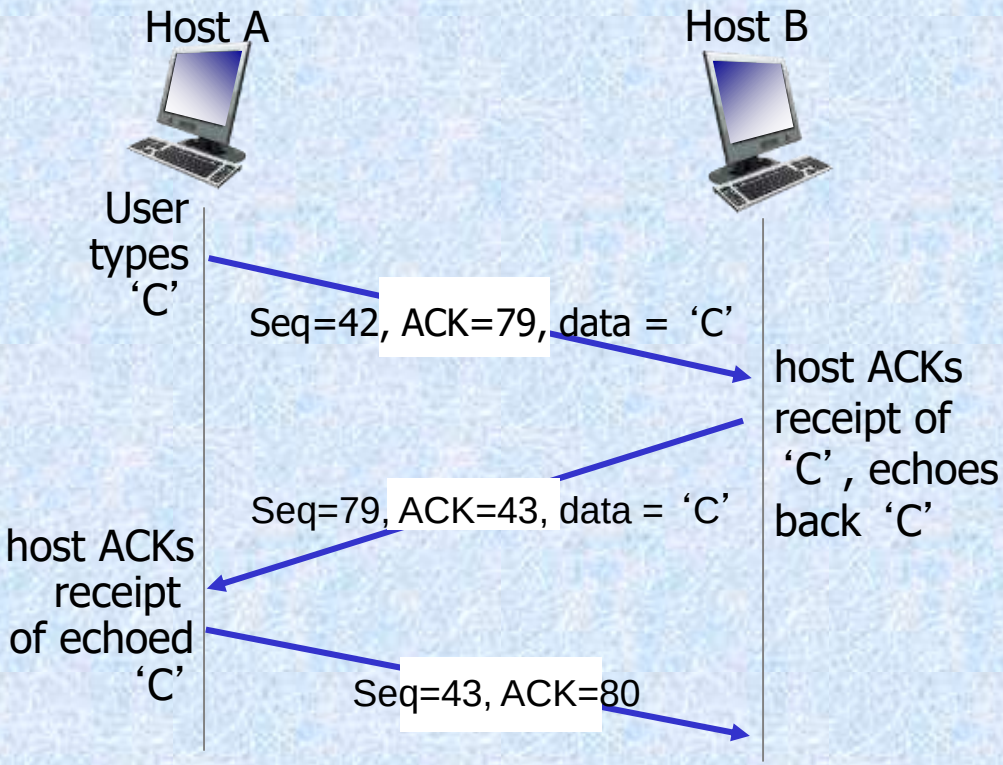


شماره ترتیب قطعه و پاسخ

- شماره ترتیب
  - شماره جریان بایت
  - شماره اولین بایت ارسالی فرستنده
- پاسخ
  - شماره ترتیب بایت بعدی مورد انتظار توسط گیرنده
  - ارسال پاسخ های تجمیع شده



شماره ترتیب: مثال



simple telnet scenario

زمان رفت–برگشت (RTT) تایمر انقضاء

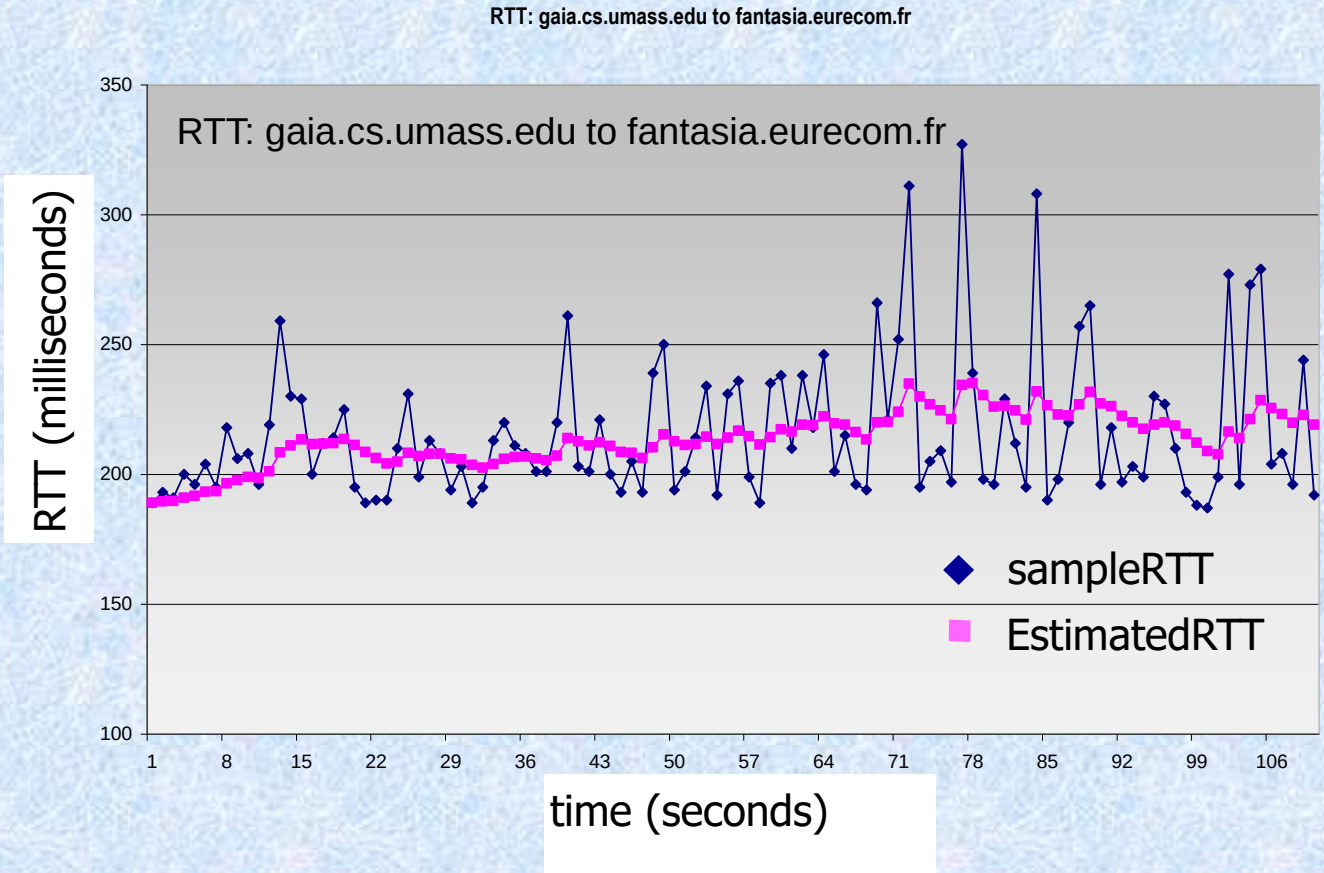
- تایمر انقضاء باید بزرگتر از RTT باشد
- زمان رفت-برگشت نمونه (SampleRTT)
  - مدت زمان بین ارسال قطعه نمونه و دریافت پاسخ آن
- در هر زمان فقط یک نمونه
- نمونه قطعات ارسال مجدد محاسبه نمی شود
- میانگین نمونه بعنوان مقدار تخمینی استفاده می شود

$$EstimatedRTT = (1 - \alpha).EstimatedRTT + \alpha.SampleRTT$$

$$EstimatedRTT = 0.875 \times EstimatedRTT + 0.125 \times SampleRTT$$

ملاحظات RTT و تایمر انقضاء

- میانگین متحرک با وزن نمایی (EWMA)
  - کاهش اثر نمونه های قبلی بصورت نمایی
- مقدار ضریب ترکیب وزنی: 0.125



انحراف RTT و بازه انقضاء تایمر

- بازه انقضاء: عبارت از زمان تخمینی **RTT** و حاشیه اطمینان
  - حاشیه اطمینان با افزایش انحراف **RTT** افزایش می یابد

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

(typically, β = 0.25)

TimeoutInterval = EstimatedRTT + 4*DevRTT

estimated RTT

"safety margin"

طراحی TCP

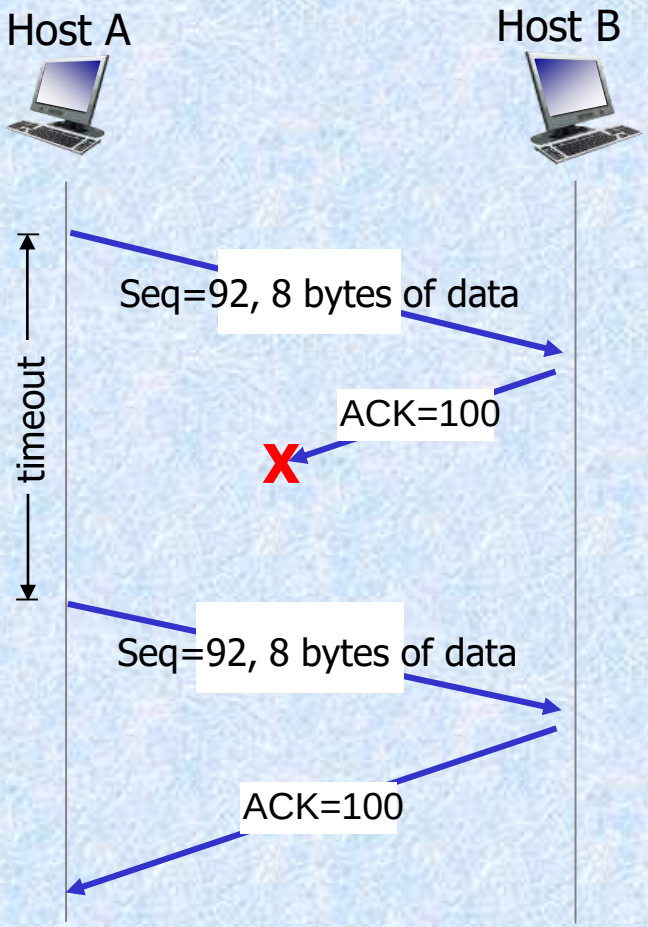
- بهره گیری از سرویس rdt روی کانال غیر قابل اطمینان IP
- استفاده از قطعات خط لوله ای
- استفاده از پاسخ تجمیع شده
- دارای تایمر ارسال مجدد تکی
- رخدادهای ارسال مجدد
  - انقضاء تایمر
  - پاسخ های تکراری

- طراحی اولیه: ساده ترین مدل
- فرض ها
  - بدون پاسخ تکراری
  - بدون کنترل جریان
  - بدن کنترل ازدحام

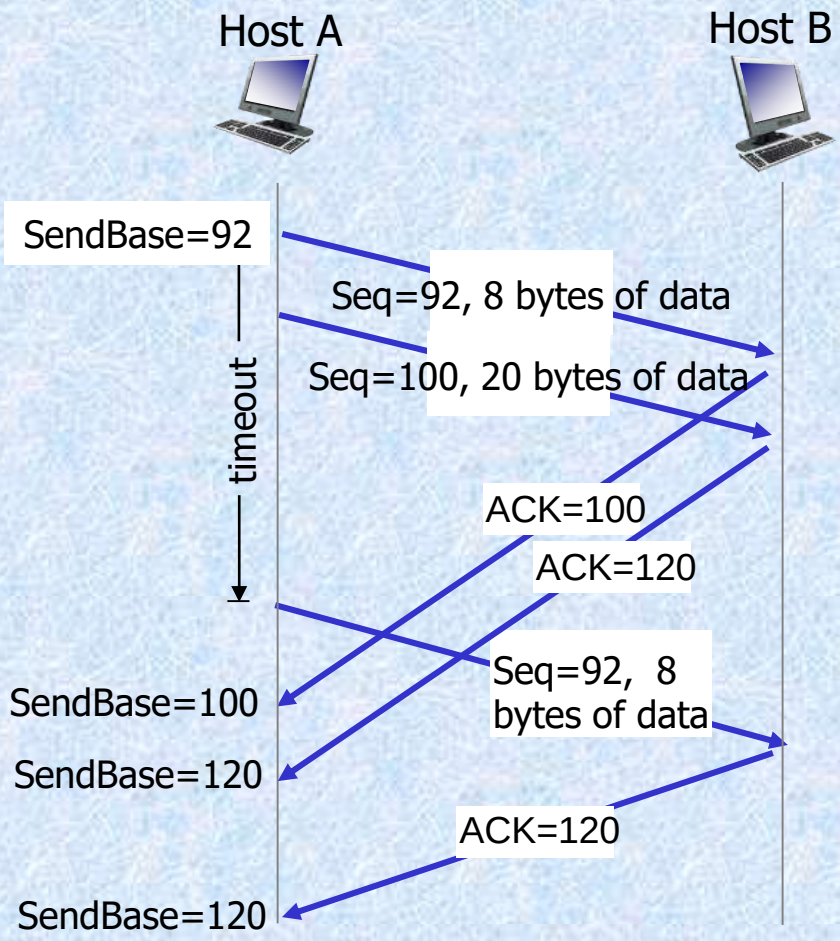
رخدادهای TCP

- درخواست ارسال (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 غیر اینصورت روی قدیمی ترین قطعه بدون پاسخ)
  - 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)

مثال: ارسال مجدد



lost ACK scenario



premature timeout

ارسال مجدد سریع

- بزرگ بودن مقدار تایمر انقضاء باعث تاخیر طولانی قبل از ارسال بسته گم شده می شود
  - تاخیر غیر قابل تحمل در لایه کاربردی
  - نیاز به فضای بافر زیاد برای نگهداری قطعات خارج از ترتیب
- راه کار
  - تشخیص قطعات گم شده توسط پاسخ تکراری
  - در صورت گم شدن قطعه تعداد پاسخ های تکراری بیشتر

- استاندارد
- در صورت دریافت بیش از ۳ 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

ساخت پاسخ TCP

رخدادهای گیرنده

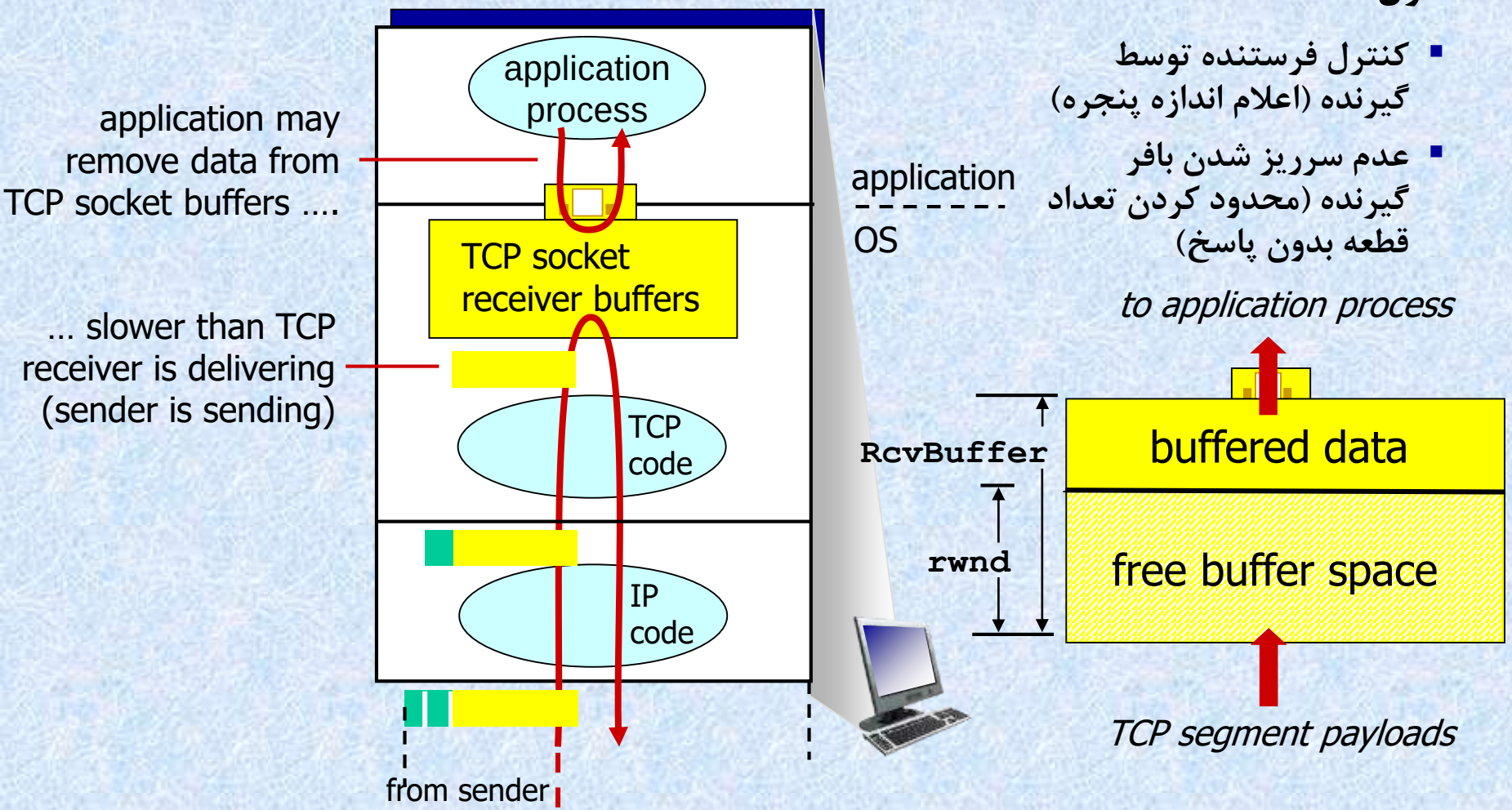
- ورود قطعه ای با شماره ترتیب مورد انتظار (تمامی داده های قبلی پاسخ داده شده اند)
- ورود قطعه ای با شماره ترتیب مورد انتظار (پاسخ یک قطعه قبلی ارسال نشده)
- ورود قطعه خارج از نوبت با شماره ترتیب بزرگتر از مورد انتظار (تشخیص فاصله)
- ورود قطعه میانی (برای پوشش کل یا بخشی از فاصله موجود در پنجره گیرنده)

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

کنترل جریان

- اصول
  - کنترل فرستنده توسط گیرنده (اعلام اندازه پنجره)
  - عدم سرریز شدن بافر گیرنده (محدود کردن تعداد قطعه بدون پاسخ)

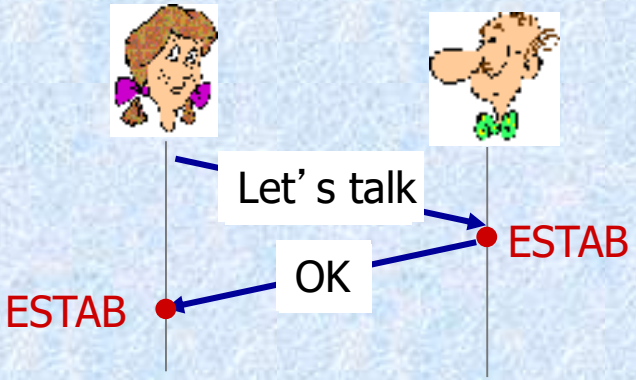


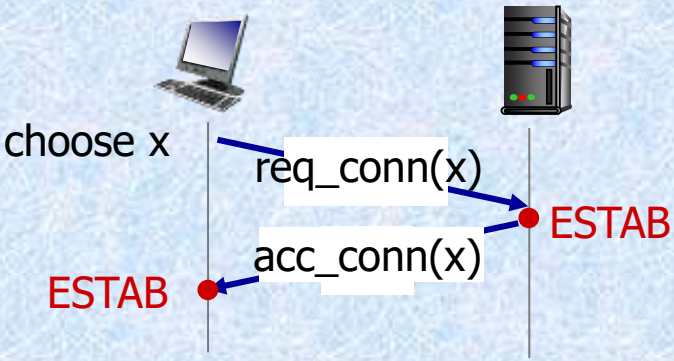
receiver protocol stack

receiver-side buffering

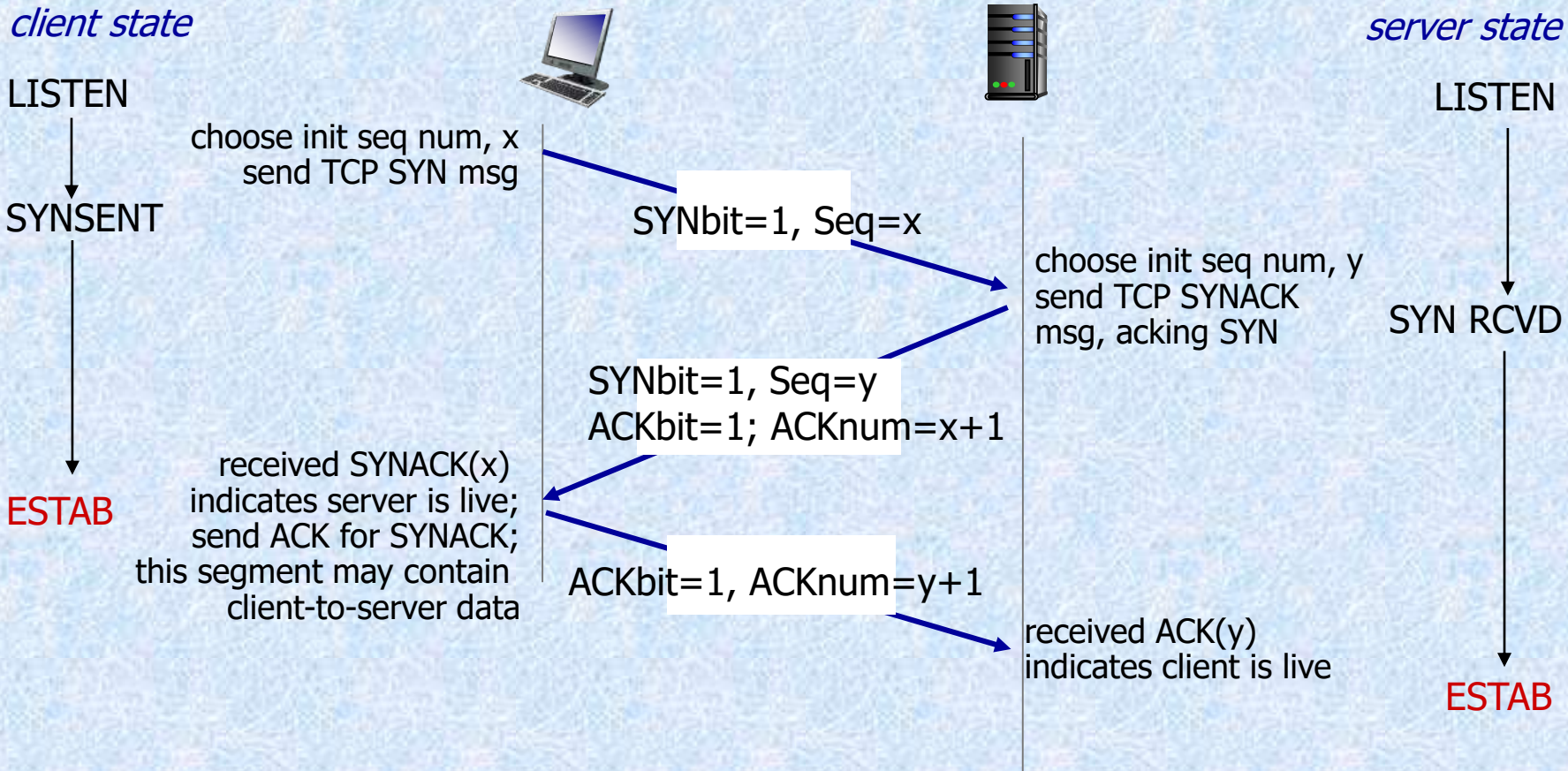
مدیریت اتصال

- سرویس اتصال گرا: نیاز به برقراری اتصال پیش از انتقال داده (Handshake)
  - توافق بر پذیرش اتصال
  - توافق بر پارامترهای اتصال
- نیاز به دست تکانی سه مرحله ای
  - تغییرات تاخیر
  - ارسال مجدد
  - اعلام شماره ترتیب توسط سرور

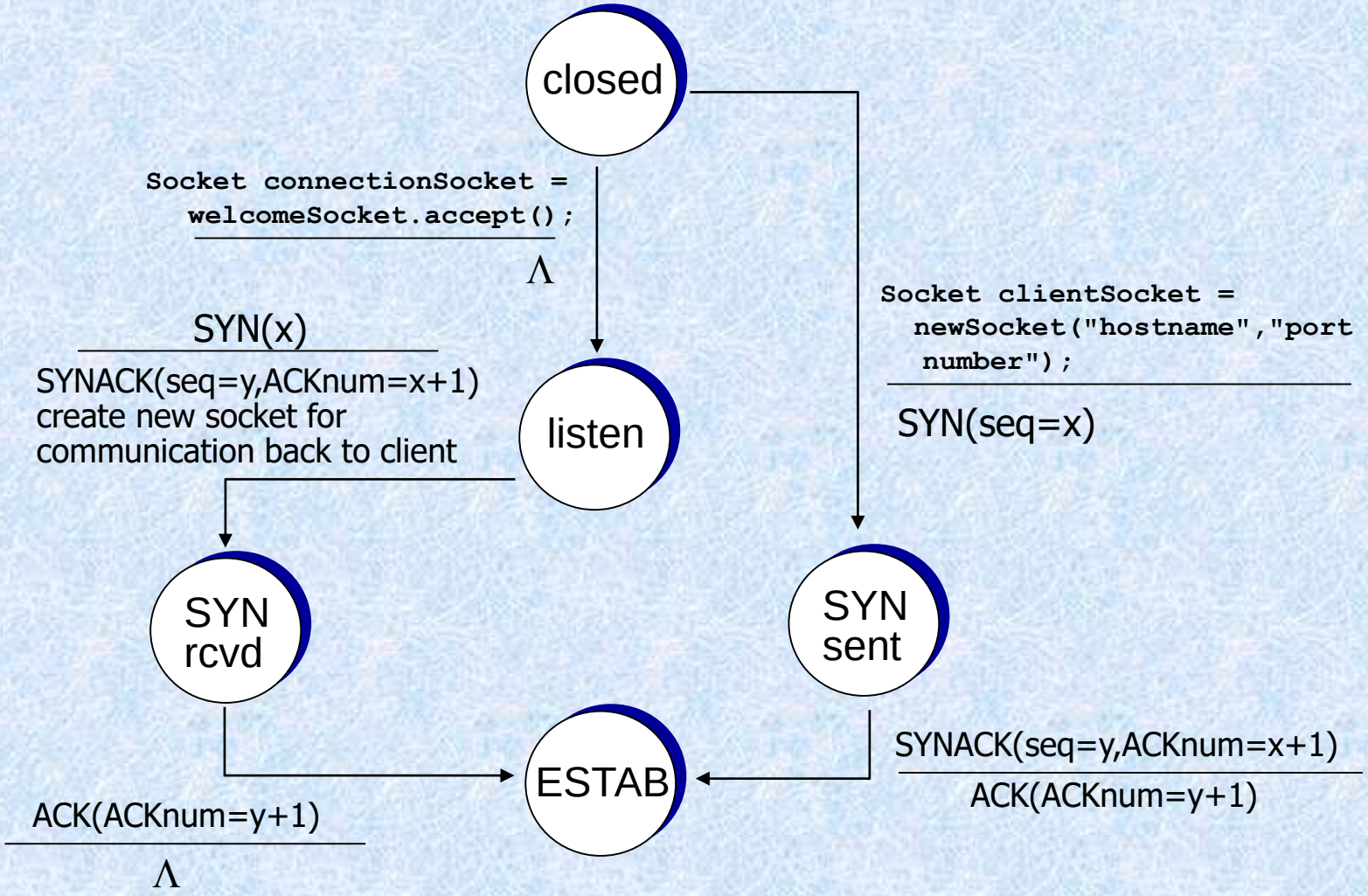




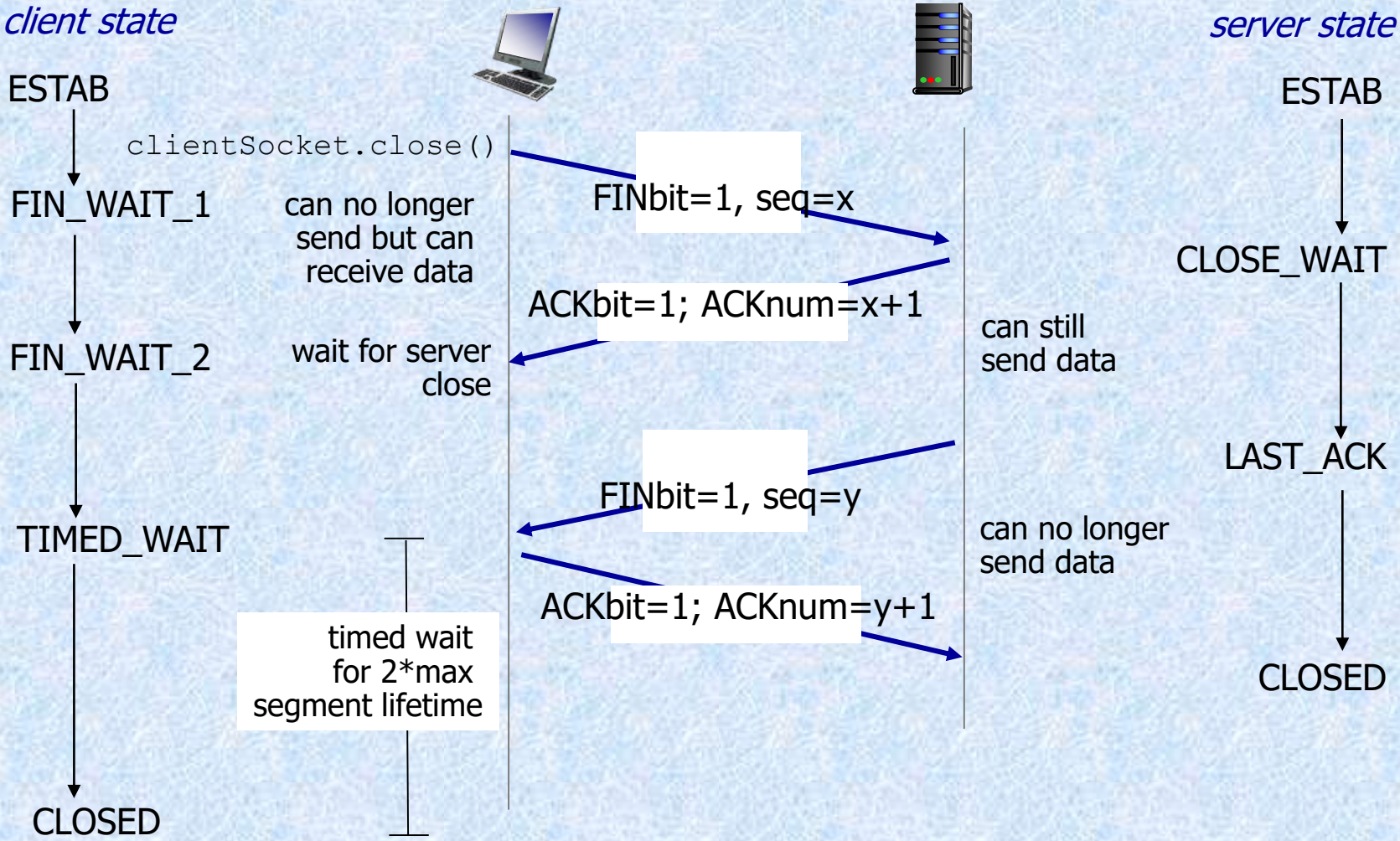
دست تکانی سه مرحله ای



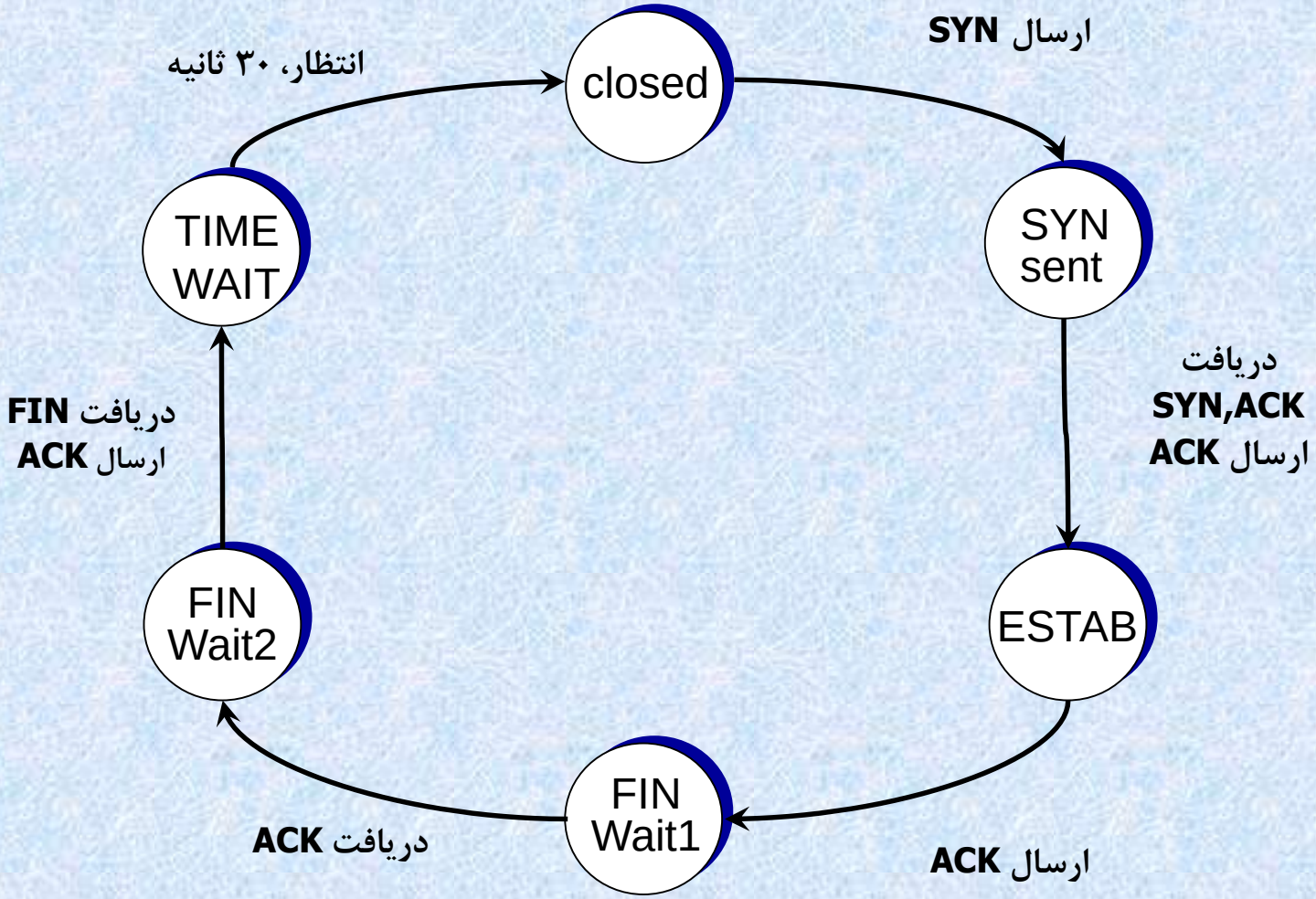
دست تکانی سه مرحله ای: FSM



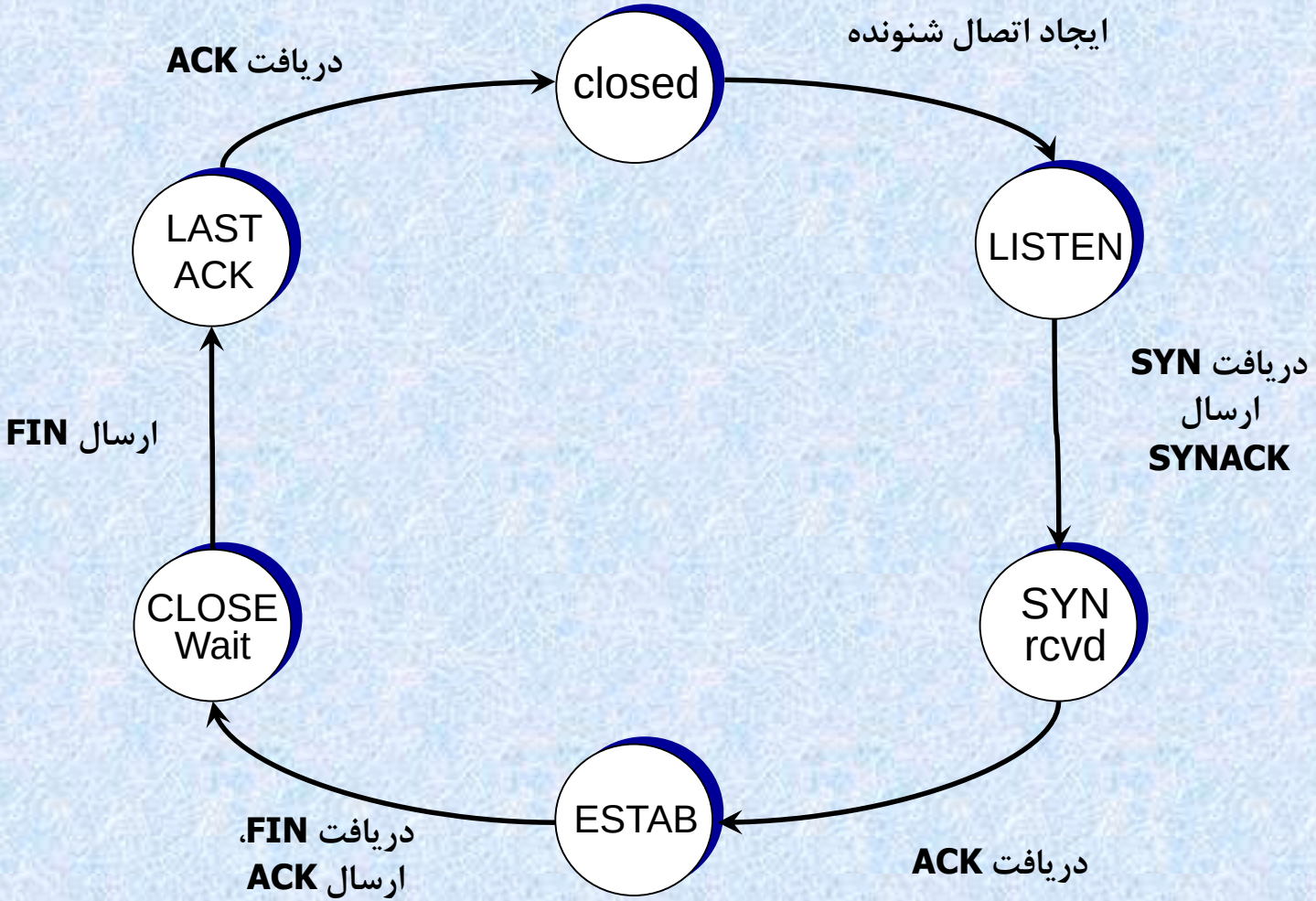
قطع اتصال


client state
server state
ESTAB
ESTAB
clientSocket.close()
FIN_WAIT_1
can no longer send but can receive data
FINbit=1, seq=x
CLOSE_WAIT
ACKbit=1; ACKnum=x+1
FIN_WAIT_2
wait for server close
can still send data
FINbit=1, seq=y
LAST_ACK
TIMED_WAIT
can no longer send data
ACKbit=1; ACKnum=y+1
timed wait for 2*max segment lifetime
CLOSED
CLOSED

FSM: وضعیت مشتری



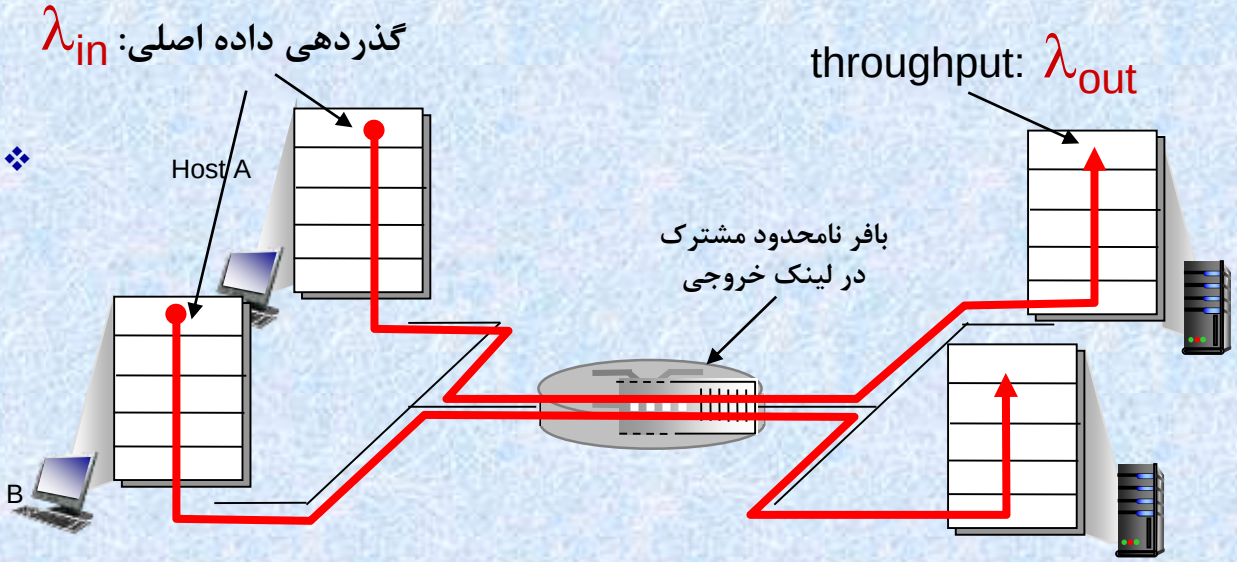
FSM: وضعیت سرویس دهنده



کنترل ازدحام

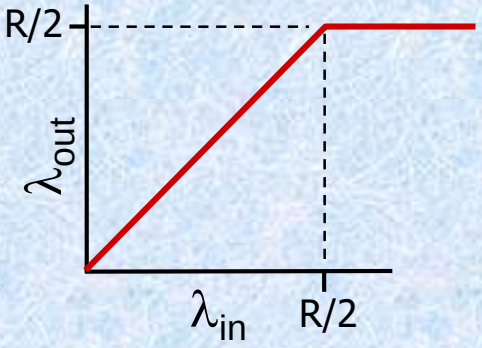
- منشاء عمده تلفات بسته
  - سرریز شدن بافر در مسیریاب ها به دلیل ازدحام
- راه کار تلفات بسته
  - ارسال مجدد
- سوال: آیا فرآیند کنترل جریان قادر به حل مشکل ازدحام است؟
- لازمه کنترل ازدحام
  - فرآیندی برای کنترل ارسال فرستنده

سناریوی ۱: دو فرستنده، یک مسیریاب

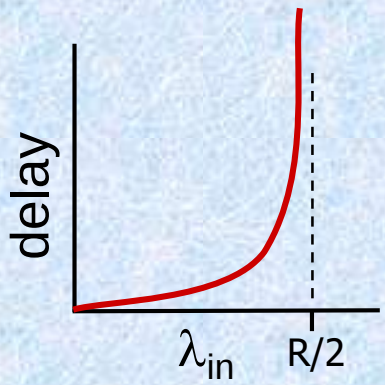


❖ فرض ها

- دو فرستنده، دو گیرنده
- فضای بافر مسیریاب نامحدود
- ظرفیت لینک خروجی: **R**
- پروتکل انتقال ساده (بدون کنترل ها)



❖ حداکثر نرخ ارسال هر فرستنده: **R/2**

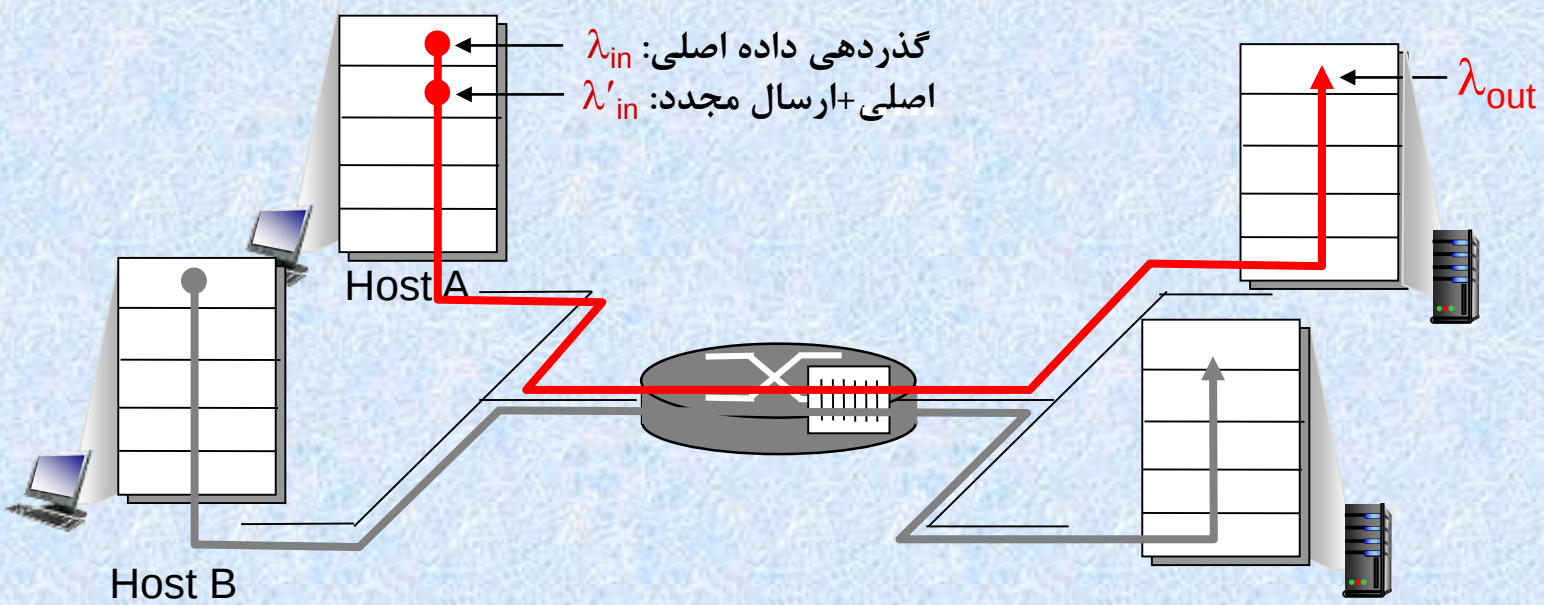


❖ افزایش تاخیر با افزایش نرخ دریافت داده به نرخ ارسال λ_{in}

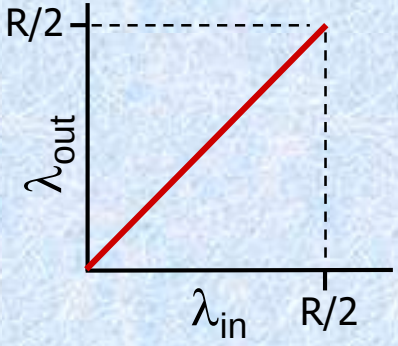
سناریوی ۲: بافر محدود، ارسال مجدد

❖ فرض ها

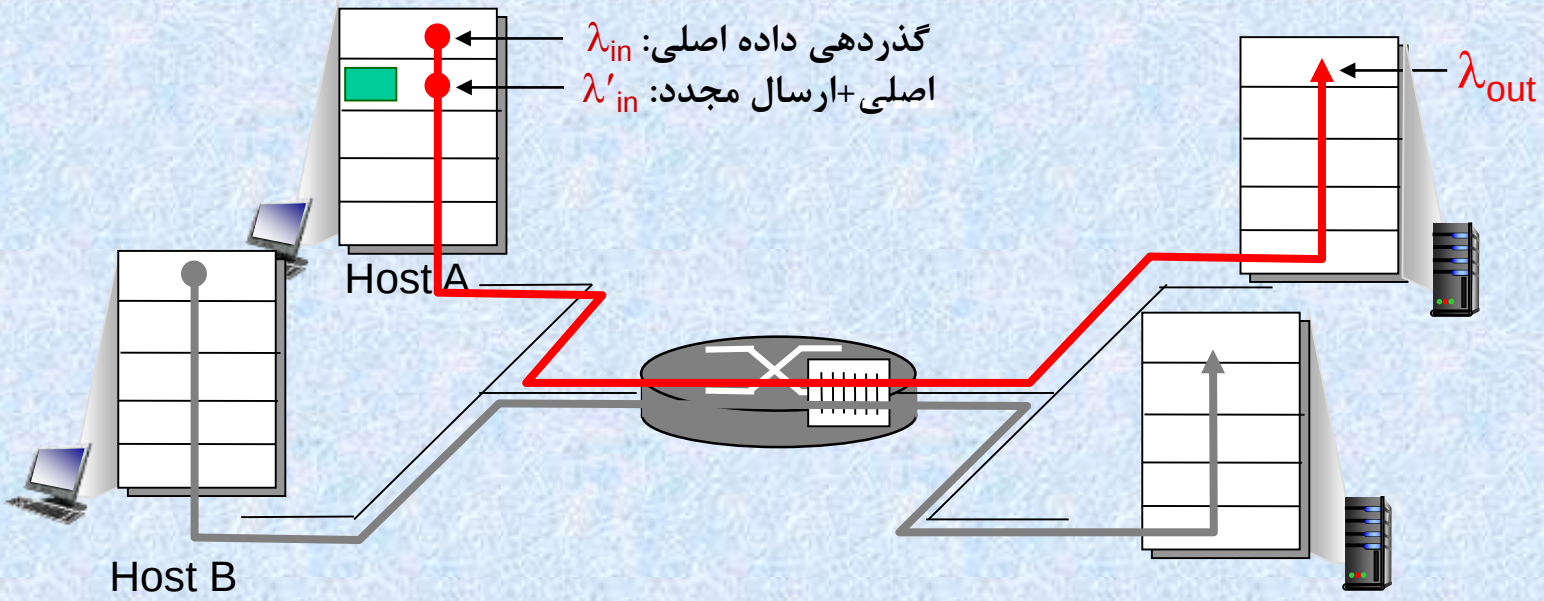
- فضای بافر مسیریاب محدود
- ارسال مجدد بسته های منقضی شده
- نرخ لایه کاربرد: ورودی = خروجی
- نرخ لایه انتقال: ورودی ≤ خروجی



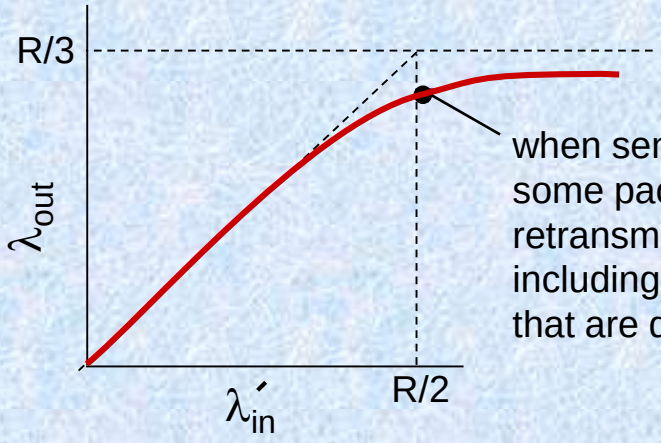
سناریوی ۲: دانش فرستنده



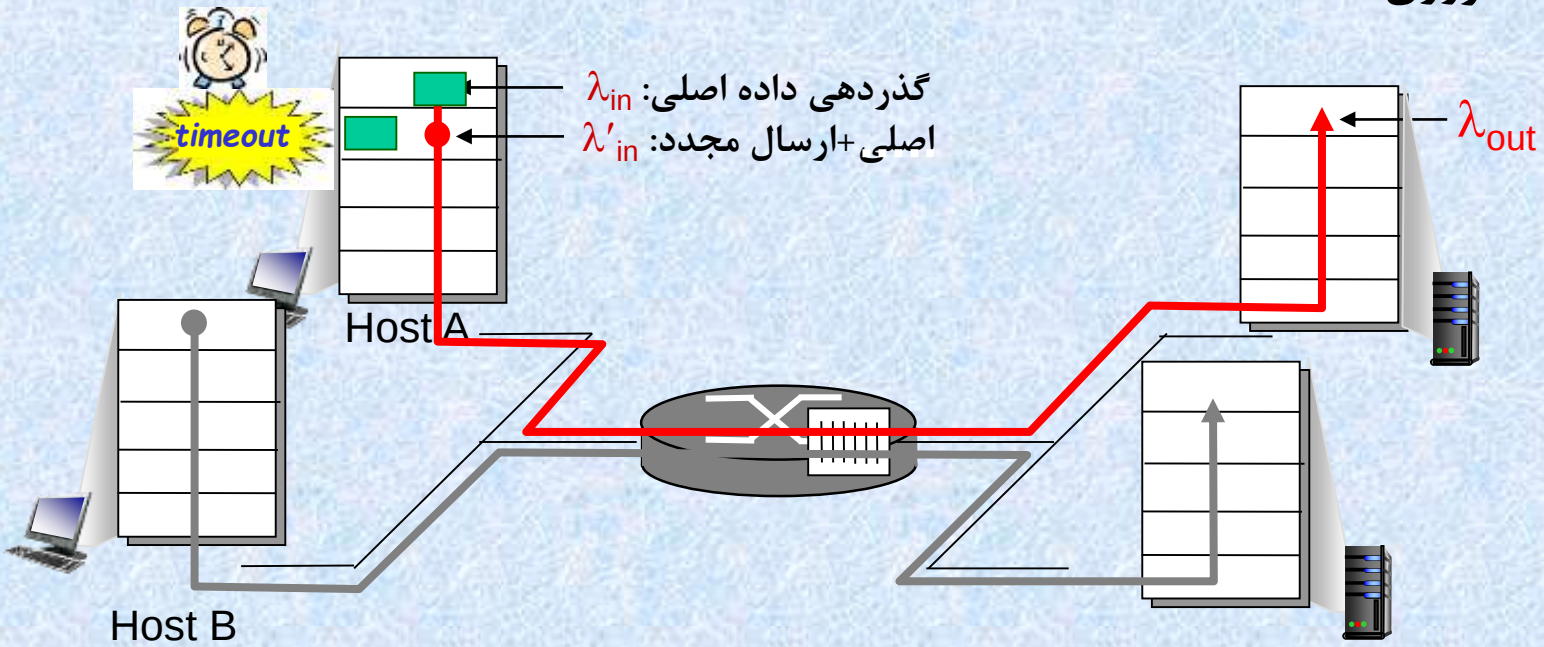
- اطلاع فرستنده از فضای بافر موجود در مسیریاب
- ارسال تنها در صورت خالی بودن بافر مسیریاب
- هزینه ازدحام: کاهش بار به دلیل ارسال مجدد



سناریوی ۲: ارسال تکراری

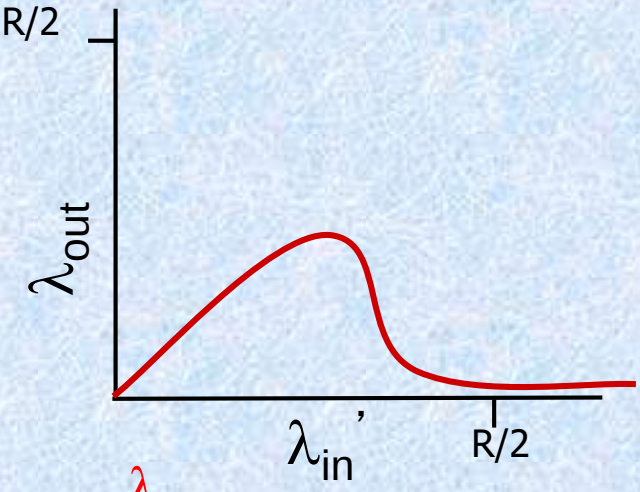


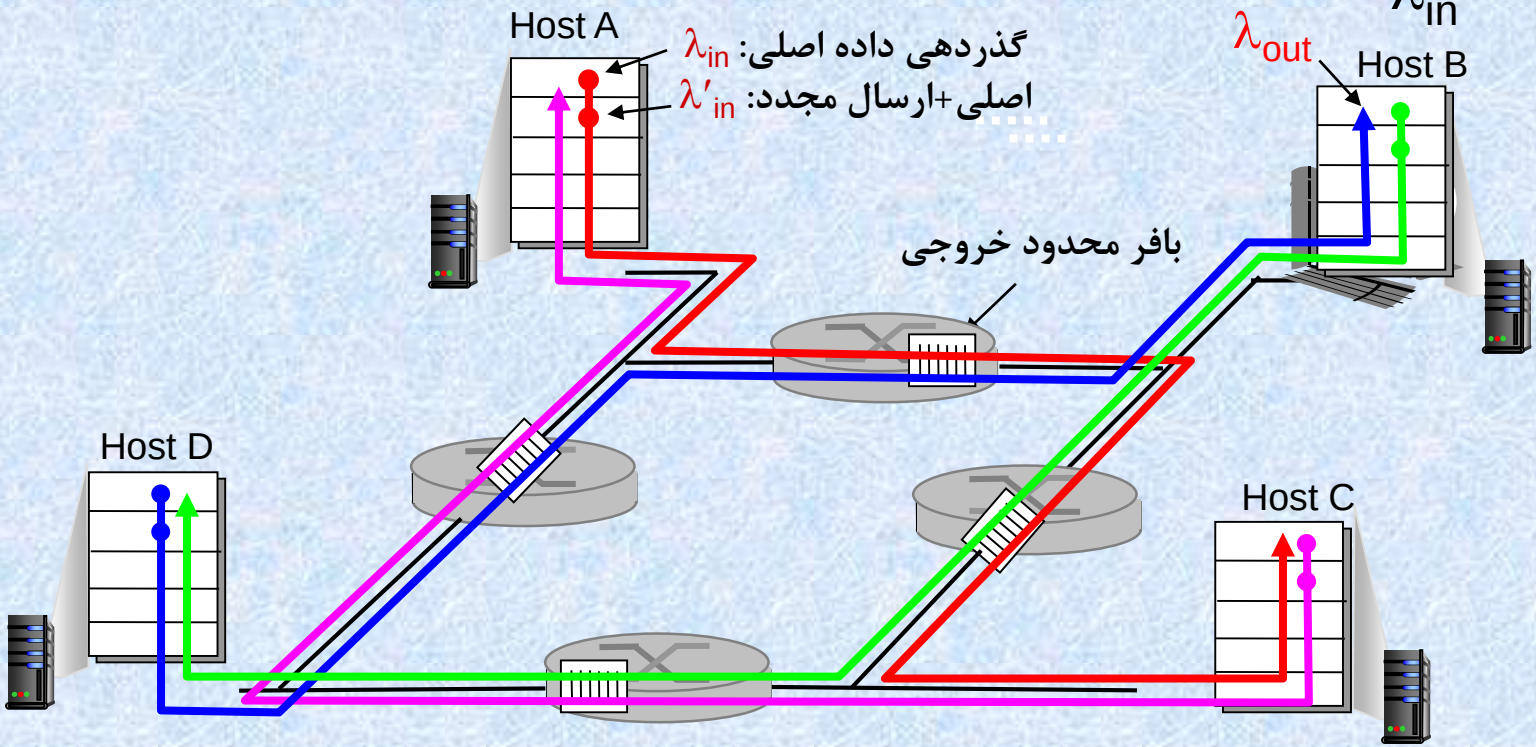
- انقضای پیش از موعد تایمر فرستنده
- ارسال مجدد زودهنگام و دریافت بسته تکراری در گیرنده
- هزینه ازدحام: ارسال مجدد غیر ضروری



سناریوی ۳: ۴ فرستنده، چند گام

- دارای تایمر/ارسال مجدد
- بار تحویلی ناچیز ← عدم سرریزی بافر
- بار تحویلی زیاد ← بار عبوری (گذردهی) از مسیریاب کاهش



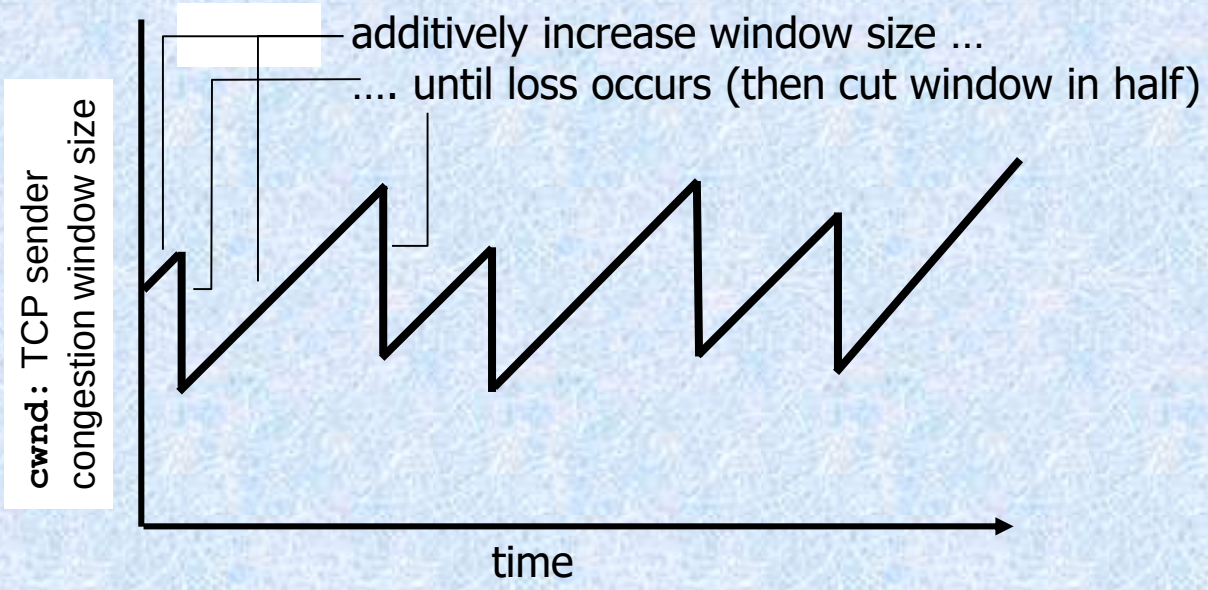


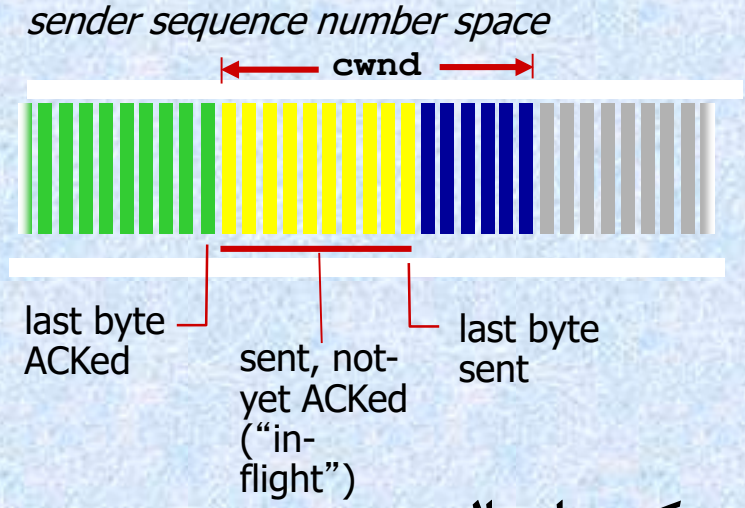
راه کارهای کنترل ازدحام

- سیاست:
  - افزایش نرخ ارسال با تحویل سالم بسته ها: افزایش **cwnd** (افزایش **MSS** با هر **RTT**)
  - کاهش نسبتا سریع نرخ با رخداد خطا: نصف **cwnd**

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

AIMD saw tooth behavior: probing for bandwidth





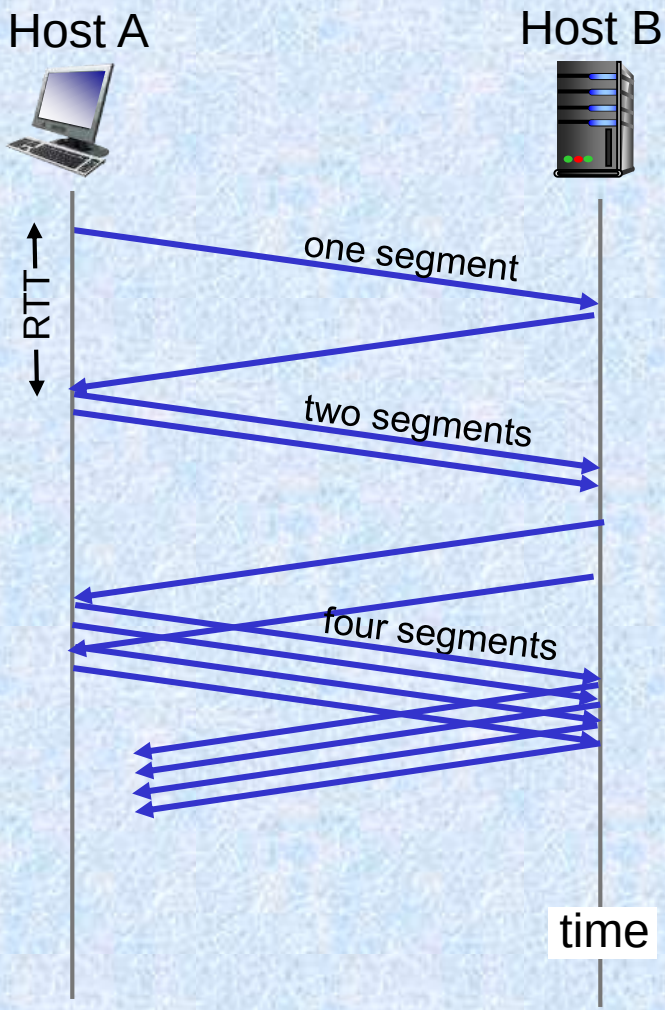
- نرخ ارسال
- انتظار به مدت **RTT** برای **ACK** پس از کل پنجره (**cwnd**)

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

- محدود کردن ارسال

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

شروع آهسته



- شروع ارسال در ابتدا با حداقل نرخ
  - یک **MSS** در مدت **RTT**
- افزایش دو برابری (نمایی) در هر بار
- روش ۱: کاهش سریع پنجره به ۱ در صورت انقضای تایمر (شروع مجدد آهسته)
- روش ۲: کاهش پنجره به نصف آخرین مقدار
- اجتناب از ازدحام (روش ۱): افزایش ۱ واحدی بجای نمایی
- روش ۲: افزایش **MSS/cwnd**