شبکه‌های کامپیوتری۲

«بارویکرد حل مساله»

درس سوم:

لایه انتقال(Transport Layer)

رئوس مطالب:

✅ آشنایی با لایه انتقال و سرویس‌های آن

❑ مالتی پلکسینگ و دی مالتی پلکسینگ

❑ انتقال نامطمئن: UDP

❑ اصول انتقال داده قابل اطمینان

❑ انتقال داده اتصال‌گرا: TCP

❑ اصول کنترل ازدحام

❑ کنترل ازدحام درTCP

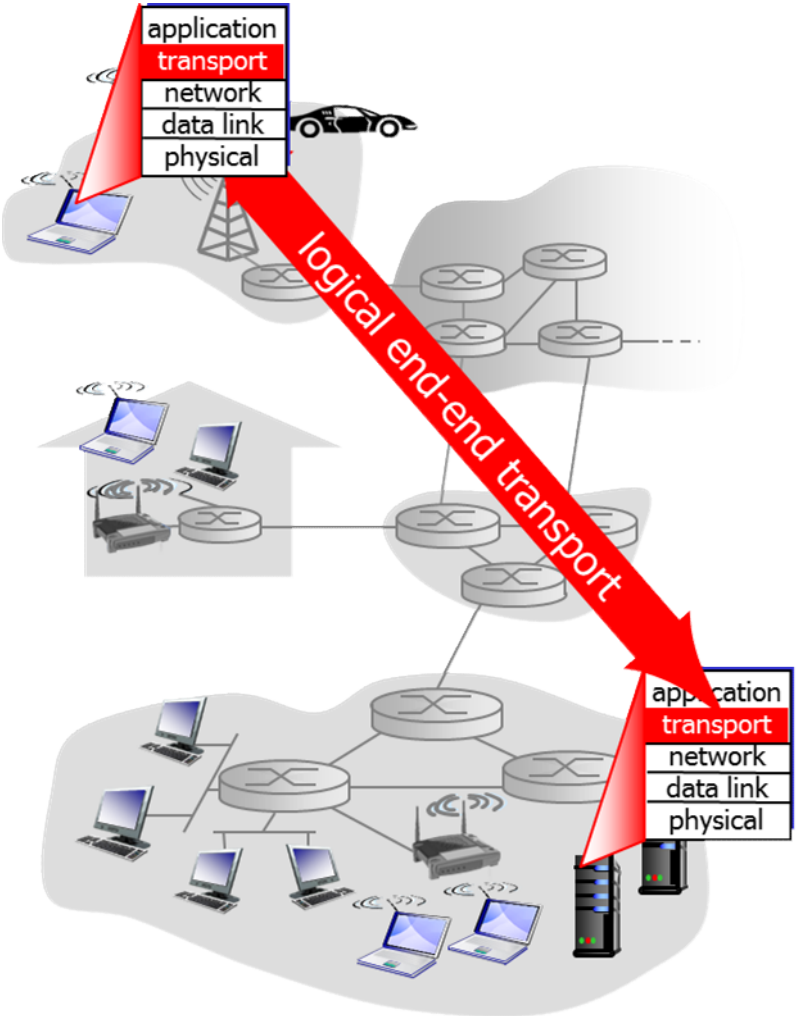
آشنایی با لایه انتقال

- لایه انتقال یک ارتباط منطقی بین فرایندهای درحال اجرا روی میزبان‌های مختلف ایجاد می‌کند.
- پروتکل‌های انتقال روی سیستم‌های انتهایی اجرا می‌شوند.

–درسمت فرستنده پیام‌های لایه کاربرد دریافت و به بسته‌های لایه انتقال تبدیل شده و به لایه شبکه سپرده می‌شود.

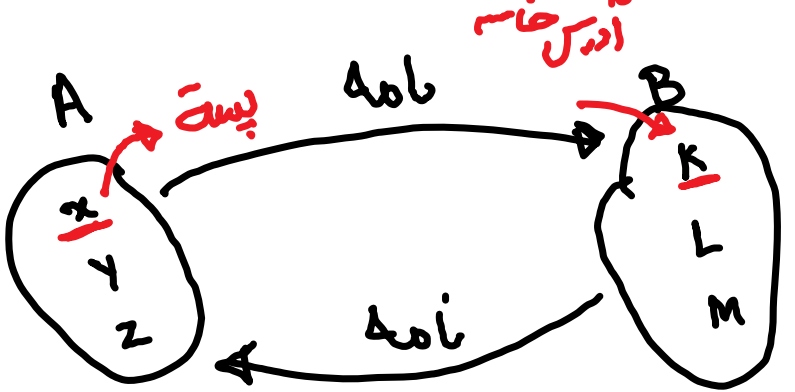
- به بسته‌های لایه انتقال قطعه می‌گویند،هرقطعه‌ای بخشی و یا کل یک پیام را دربرمی‌گیرد که سرایندهایی نیز به ان افزوده می‌شود.

- درسمت گیرنده لایه انتقال قطعات دریافت شده را پردازش و داده موجود در آن را در اختیار لایه بالاتر(کاربرد) قرار می‌دهد.


application
transport
network
data link
physical
logical end-end transport
application
transport
network
data link
physical

مقایسه لایه شبکه و لایه انتقال در ارتباط

- لایه شبکه یک ارتباط منطقی بین دو میزبان ایجاد می‌کند.
- لایه انتقال یک ارتباط منطقی بین دو فرایند در حال اجرا روی میزبان‌ها ایجاد می‌کند.



لایه انتقال در اینترنت

- ارایه سرویس انتقال مطمئن و تحویل بسته ها به ترتیب-TCP
  -کنترل ازدحام در شبکه
  - کنترل جریان
- ارایه سرویس نامطمئن-عدم کنترل روی ترتیب بسته‌ها-UDP

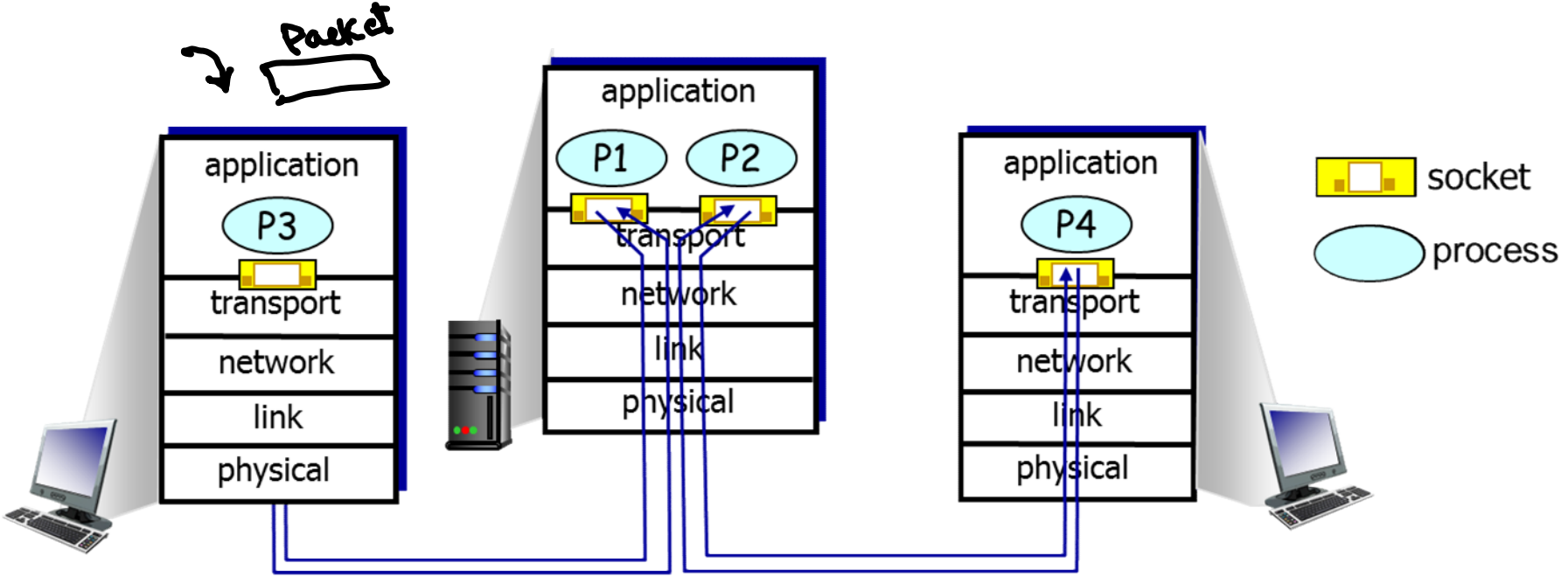
-حداقل خدمات موردنیاز در لایه انتقال یعنی تحویل داده فرایند به فرایند و تشخیص خطا را را انجام می‌دهد.

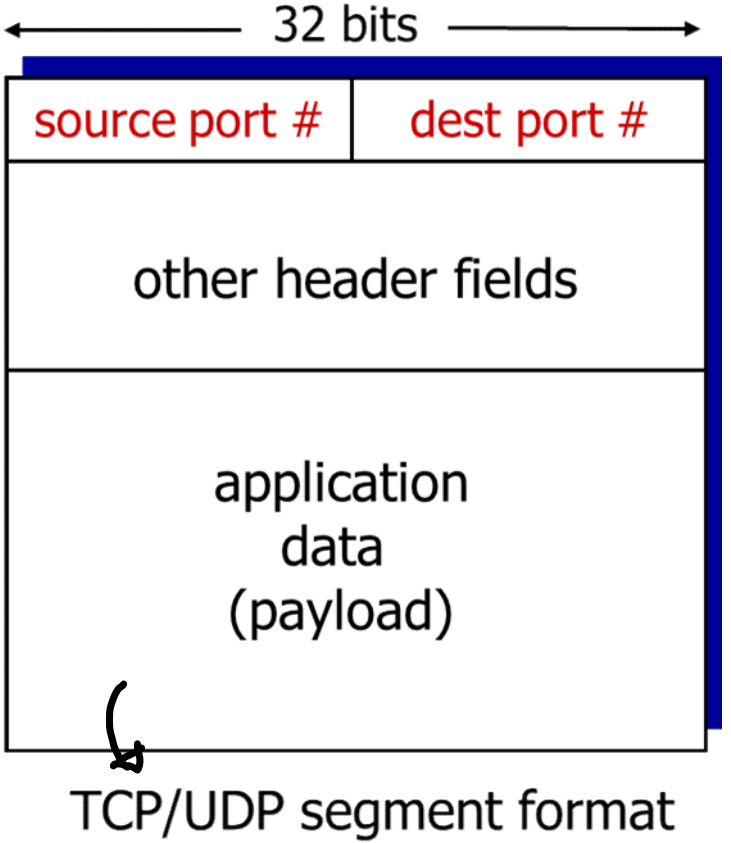
- همانند سرویس ارایه شده توسط IP، یک سرویس غیرقابل اطمینان است.

رئوس مطالب:

- ❑ آشنایی با لایه انتقال و سرویس‌های آن
- ✅ **مالتی پلکسینگ و دی مالتی پلکسینگ**
- ❑ **انتقال نامطمئن: UDP**
- ❑ **اصول انتقال داده قابل اطمینان**
- ❑ **انتقال داده اتصال‌گرا: TCP**
- ❑ **اصول کنترل ازدحام**
- ❑ **کنترل ازدحام درTCP**

Multiplexing - Demultiplexing





TCP/UDP segment format

RFC 1700
پورت های معروف
0 - 1023

Multiplexing-Demultiplexing:UDP

یادآوری : ایجاد یک سوکت UDP در پایتون:

```
ClientSocket= Socket(socket.AF_INET, socket.SOCK_DGRAM);
```

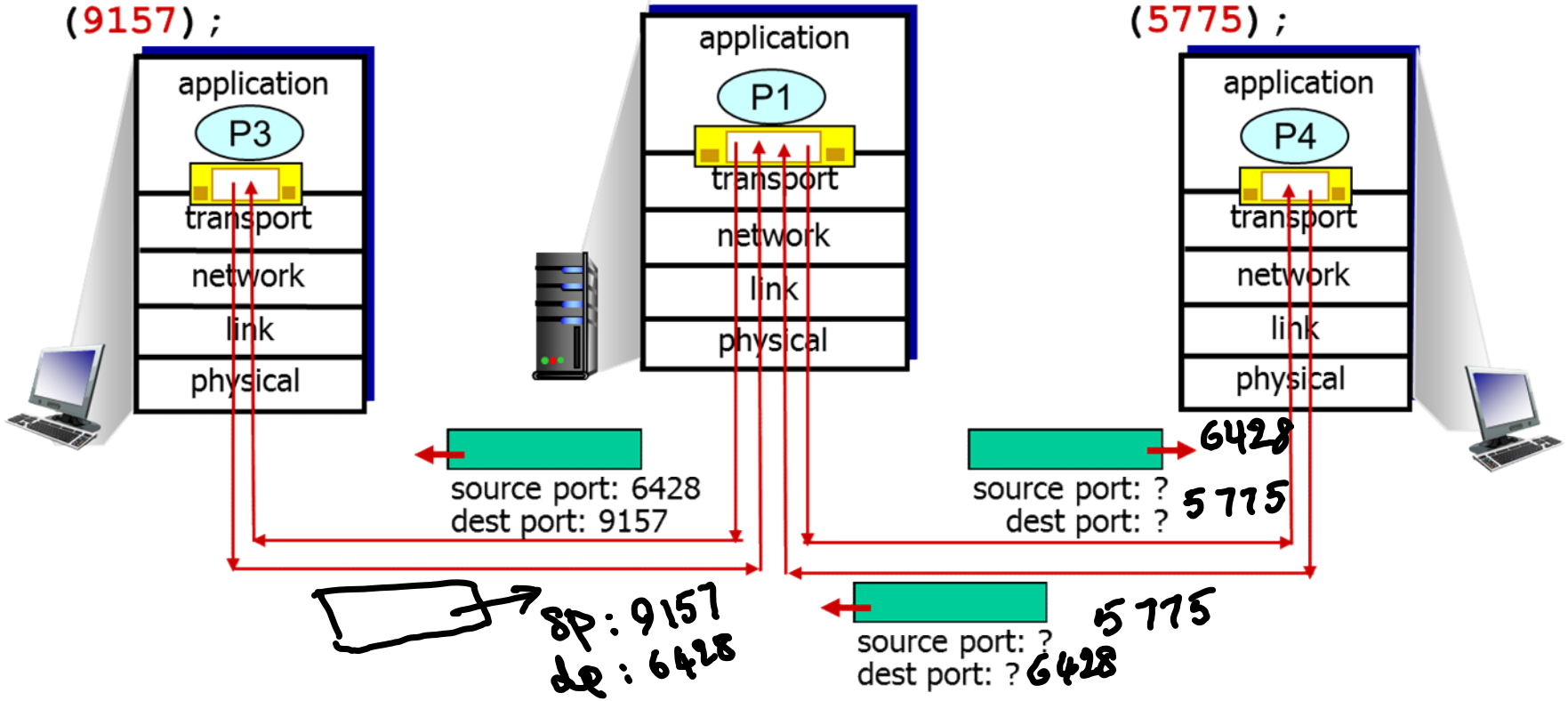
- وقتی‌که یک میزبان یک قطعه UDP را دریافت می‌کند،لایه انتقال این میزبان با بررسی شماره پورت مقصد؛عمل دی مالتی‌پلکسینگ را انجام و آنرا به سوکت مربوطه در میزبان تحویل می‌دهد.

- برای شناسایی یک سوکت UDP در مقصد فقط به دو مولفه آدرسIP و شماره پورت مقصد نیاز داریم.

- قطعات UDP با شماره پورت مقصد یکسان به یک سوکت تحویل داده می‌شوند.

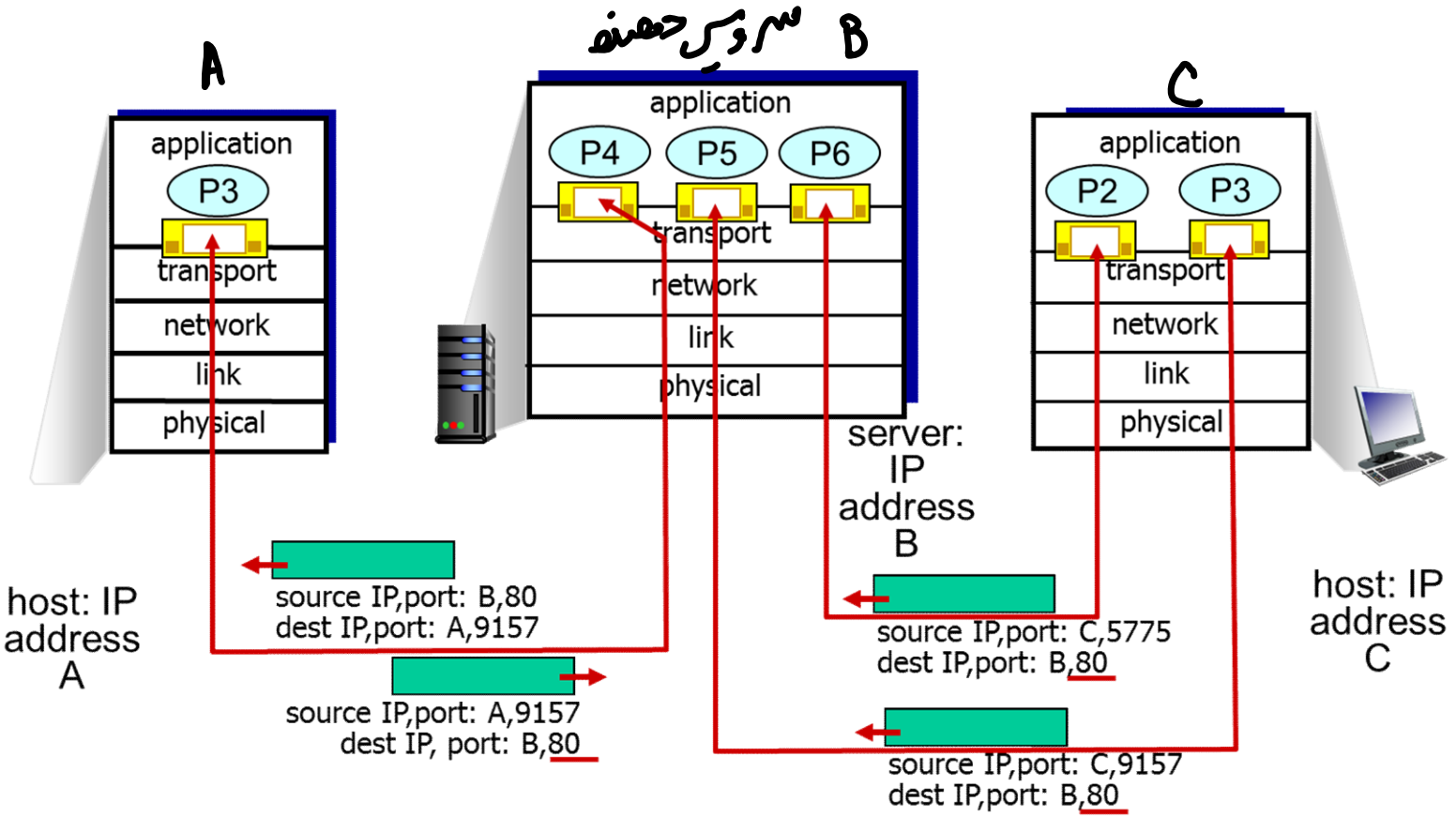
```
DatagramSocket
 mySocket2 = new
 DatagramSocket
 (9157);
```

```
DatagramSocket
 mySocket1 = new
 DatagramSocket
 (5775);
```

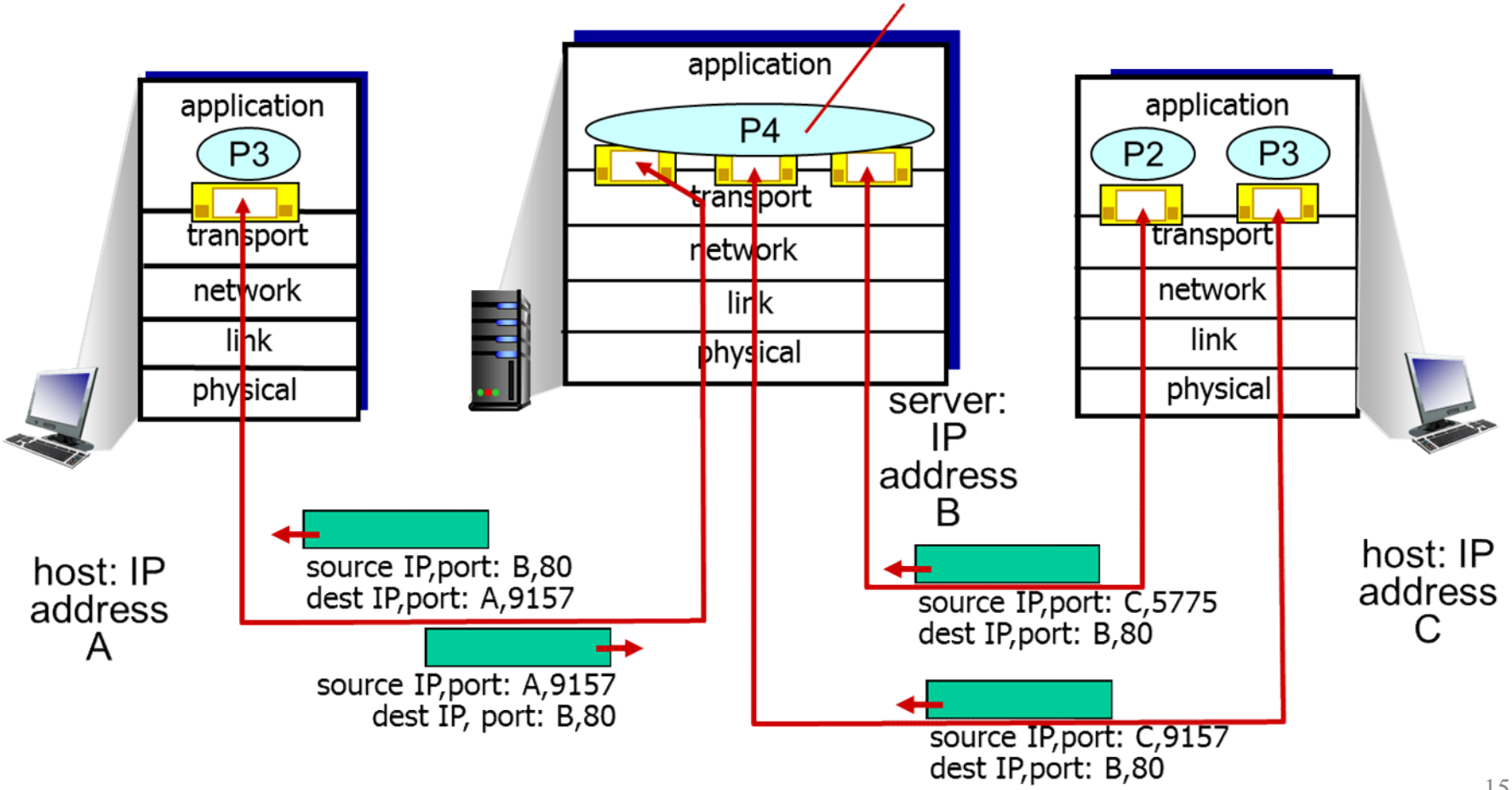


Multiplexing-Demultiplexing:TCP

- سوکت TCPبرخلاف UDP با چهار مشخصه زیر شناسایی می‌شود:
  - شماره پورت مبدا
  - شماره پورت مقصد
  - آدرس IP مبدا
  - آدرس IP مقصد
- در مقصد از هرچهار مولفه بالا برای مشخص نمودن فرایند مقصد استفاده می‌شود.
- برخلاف UDP دو قطعه TCP با شماره پورت یا IP مبدا متفاوت به دو سوکت مختلف هدایت می‌شوند.
- سرویس‌دهنده وب برای هر اتصال TCP یک سوکت باز می‌کند.



three segments, all destined to IP address: B,
dest port: 80 are demultiplexed to *different* sockets


application
P3
transport
network
link
physical
application
P4
transport
network
link
physical
server:
IP
address
B
application
P2
P3
transport
network
link
physical
host: IP
address
A
source IP,port: B,80
dest IP,port: A,9157
source IP,port: A,9157
dest IP, port: B,80
source IP,port: C,5775
dest IP,port: B,80
source IP,port: C,9157
dest IP,port: B,80
host: IP
address
C

رئوس مطالب:

- ❑ آشنایی با لایه انتقال و سرویس‌های آن
- ❑ مالتی پلکسینگ و دی مالتی پلکسینگ
- ✅ **انتقال نامطمئن: UDP**
- ❑ **اصول انتقال داده قابل اطمینان**
- ❑ **انتقال داده اتصال‌گرا: TCP**
- ❑ **اصول کنترل ازدحام**
- ❑ **کنترل ازدحام درTCP**

UDP: User Datagram Protocol [RFC 768]

- یک پروتکل ساده بدون پیچیدگی می‌باشد.و وظایف یک پروتکل انتقال را در کمترین سطح انجام می‌دهد.
- فقط عملیات مالتی پلسینگ و دی مالتی پلکسینگ و کمی بررسی خطا انجام می‌دهد
- UDP یک پروتکل انتقال نامتصل است.لذا قبل از ارسال قطعات هیچ‌گونه دست تکانی بین گیرنده و فرستنده اتفاق نمی‌افتد.
- بدلیل سبک بودن UDP، پروتکل لایه کاربردDNS از آن استفاده می‌کند.
- برنامه‌های کاربردی انتقال ویدئو مانند ویدئوکنفرانس از UDP استفاده می‌کنند

UDP: User Datagram Protocol [RFC 768]

❖ دلایلی که یک برنامه کاربردی UDP را برTCPترجیح می‌دهد:

۱-کنترل برنامه‌کاربردی بر محتویات پیام‌ها و زمان ارسال آنها

۲-عدم نیاز به برقراری اتصال

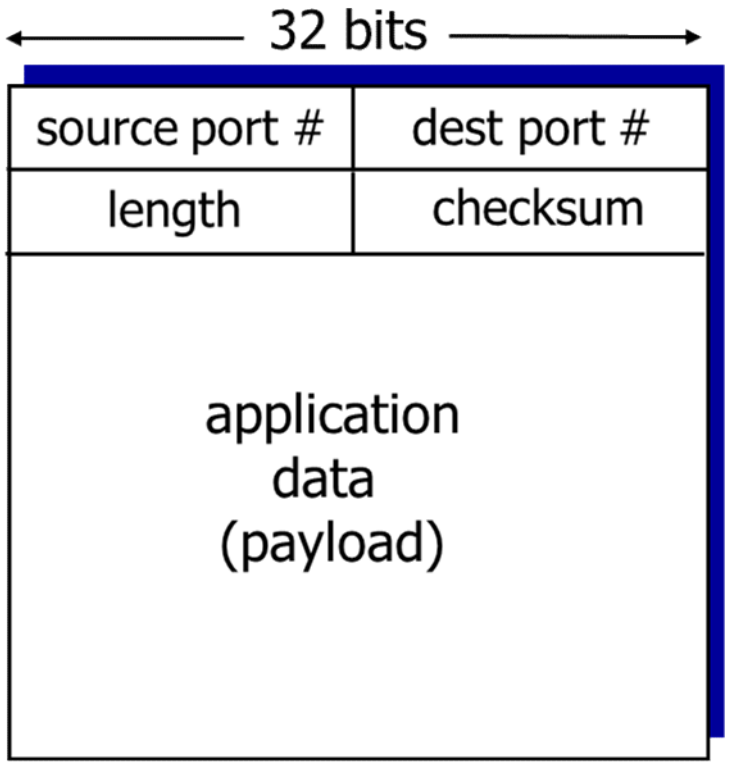
۳-عدم نیازبه حالت اتصال

۴-ناچیز بودن سربارسرایند در هر بسته

سرآیند ۲۰ بایت

سرآیند ۸ بایت

ساختار قطعه UDP



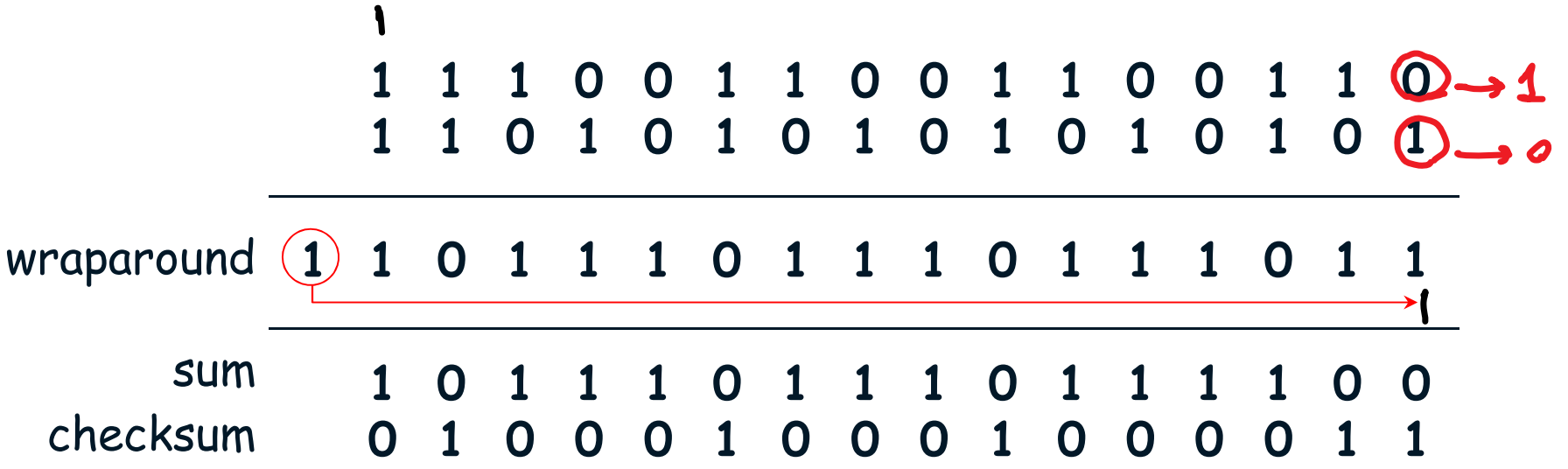
UDP segment format

جمع کنترلی در UDP

- هدف از جمع کنترلی کشف خطا در داده‌های ارسال شده می‌باشد.

- درسمت فرستنده، کلمات ۱۶ بیتی قطعه با هم جمع شده و نهایتا نتیجه مکمل۱ شده و در فیلد Checksum قطعه UDP قرار می‌گیرد.

- درسمت گیرنده نیز کلمات۱۶بیتی باهم جمع می‌شوند، و نتیجه با مقدار فیلدChecksumجمع شده و مکمل ۱ می‌شود، درصورتیکه تمام بیت‌های حاصل یک باشند، خطا رخ نداده و درغیراینصورت خطا رخ داده است.

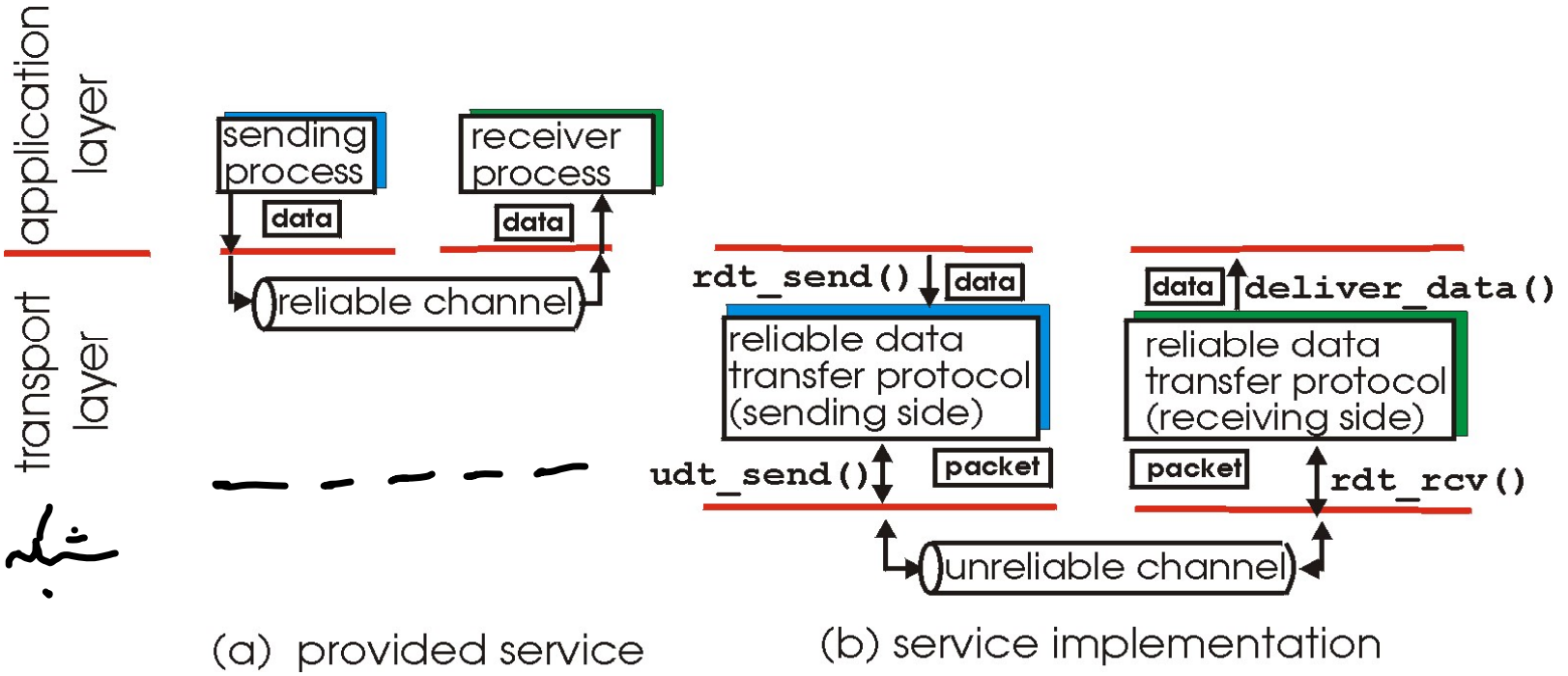

1
1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 → 1
1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 → 0
wraparound
1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1
1
sum
1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0
checksum
0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1

رئوس مطالب:

- ❑ آشنایی با لایه انتقال و سرویس‌های آن
- ❑ مالتی پلکسینگ و دی مالتی پلکسینگ
- ❑ انتقال نامطمئن: UDP
- ✅ **اصول انتقال داده قابل اطمینان**
- ❑ **انتقال داده اتصال‌گرا: TCP**
- ❑ **اصول کنترل ازدحام**
- ❑ **کنترل ازدحام درTCP**

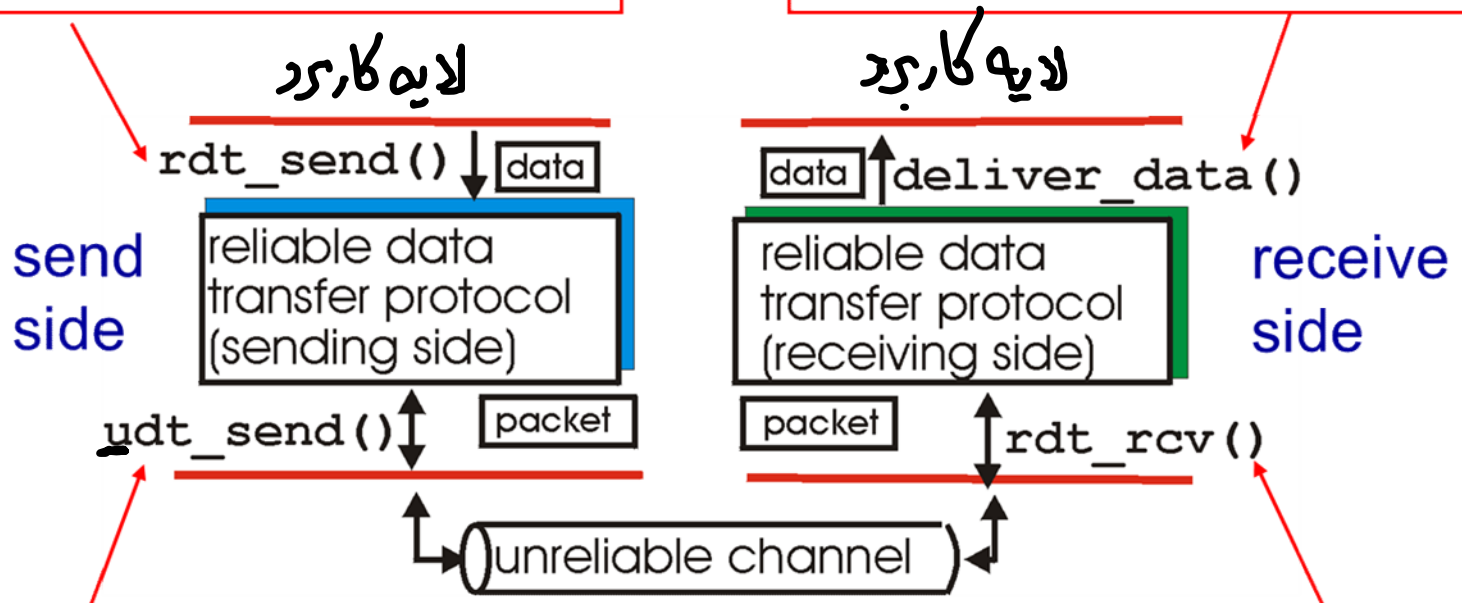
انتقال داده مطمئن (reliable)

- یکی از ده چالش مهم در شبکه‌های کامپیوتری انتقال داده مطمئن است.



rdt_send(): فراخوانی از لایه بالا، داده گرفته شده از لایه بالاتر خود را به سمت گیرنده می‌فرستد.

deliver_data(): بوسیله rdt برای تحویل داده به لایه بالاتر فراخوانی می‌شود



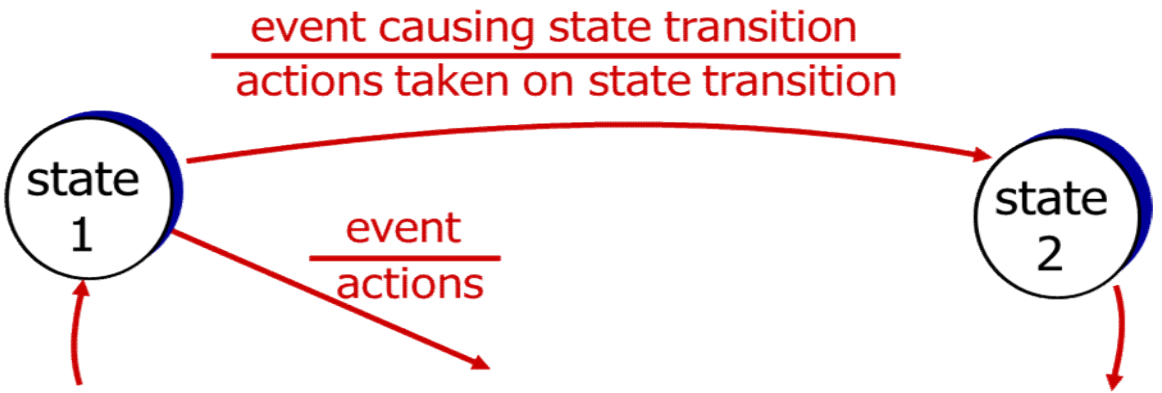
udt_send(): بوسیله rdt برای انتقال داده توسط کانال انتقال نامطمئن (udt) فراخوانی می‌شود

rdt_rcv(): هنگامیکه بسته از لایه پایین می‌رسد این متد فراخوانی می‌شود

ساخت پروتکل انتقال داده مطمئن

- ابتدا ساده‌ترین حالت (کانال زیرین کاملا قابل اطمینان) را درنظر می‌گیریم و به مرور محدودیت‌ها را درباره کانال زیرین افزایش داده تا به مدل واقعی دست پیدا کنیم.
- داده ها فقط در یک سمت ارسال می‌شوند، اما بسته‌های کنترلی دوطرفه ارسال می‌شوند.

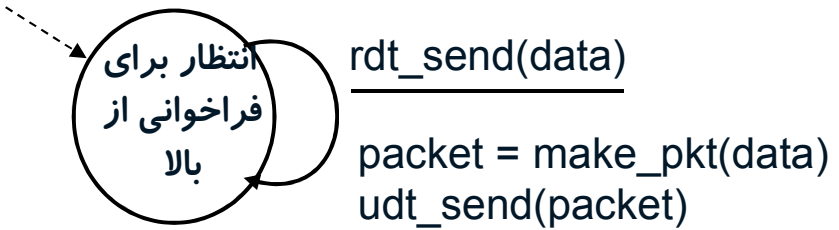
- از ماشین حالت متناهی(FSM) برای نشان دادن عملکرد پروتکل‌ها استفاده می‌کنیم.



rdt1.0: پروتکل انتقال داده مطمئن روی یک کانال زیرین کاملامطمئن

* فرستنده، گیرنده هیچ تعاملی ندارند

نسبت کسرلی



فرستنده

گیرنده

rdt2.0: پروتکل انتقال داده مطمئن روی یک کانال همراه با خطای بیتی

- در کانال زیرین امکان دارد، که بیت‌ها خراب شوند(1←0) یا (0←1) .
- خطای بیتی در هرکدام از مراحل عبور بسته از کانال زیرین(لینک‌های مخابراتی یا هنگام ذخیره‌شدن در مسیریاب‌ها) می‌تواند دچار خطا شود.
- **چگونه فرستنده از خراب شدن بیت‌ها در گیرنده مطلع خواهد شد؟**

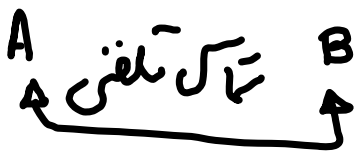
rdt2.0: پروتکل انتقال داده مطمئن روی یک کانال همراه با خطای بیتی

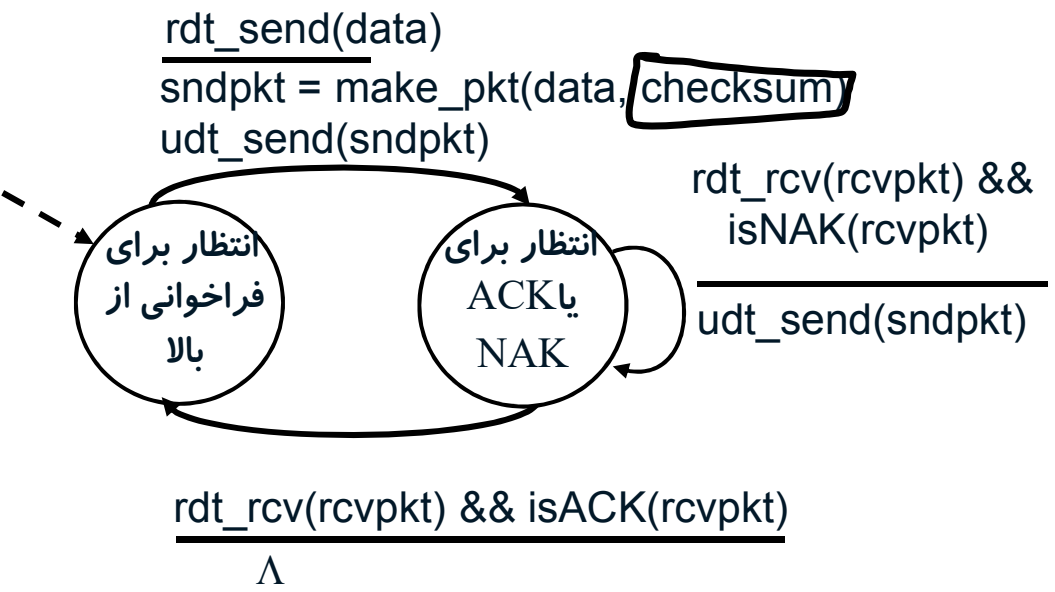
- یک پروتکل درخواست تکرار خودکار (ARQ) برای مقابله با خطای بیتی به سه قابلیت نیاز دارد:

۱- تشخیص خطا با استفاده از جمع کنترلی (Checksum)

۲-فیدبک این خطا به فرستنده از جانب گیرنده

۳-ارسال مجدد بسته از سمت فرستنده

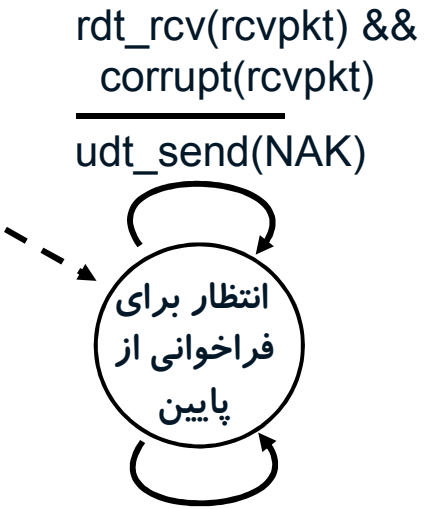


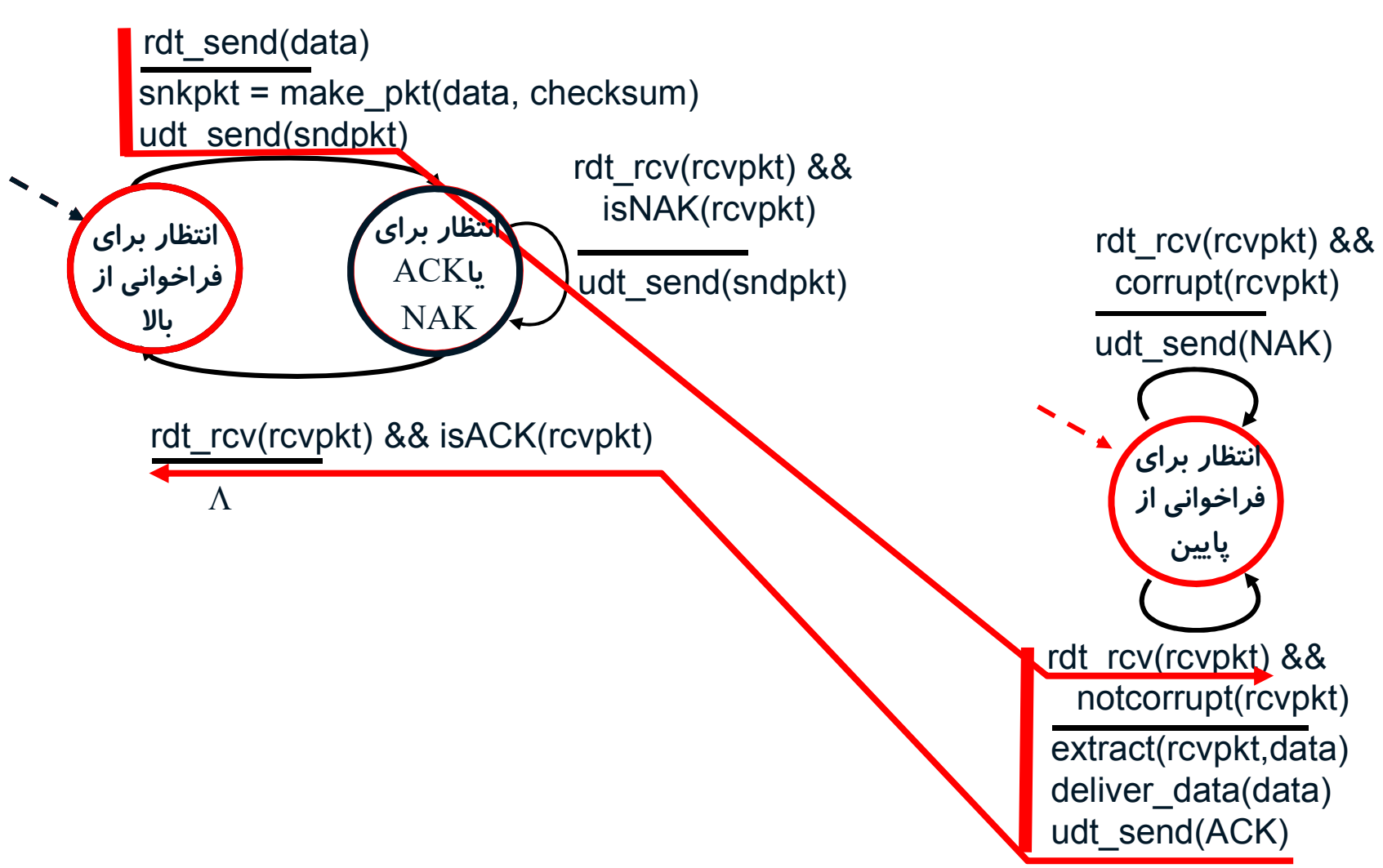


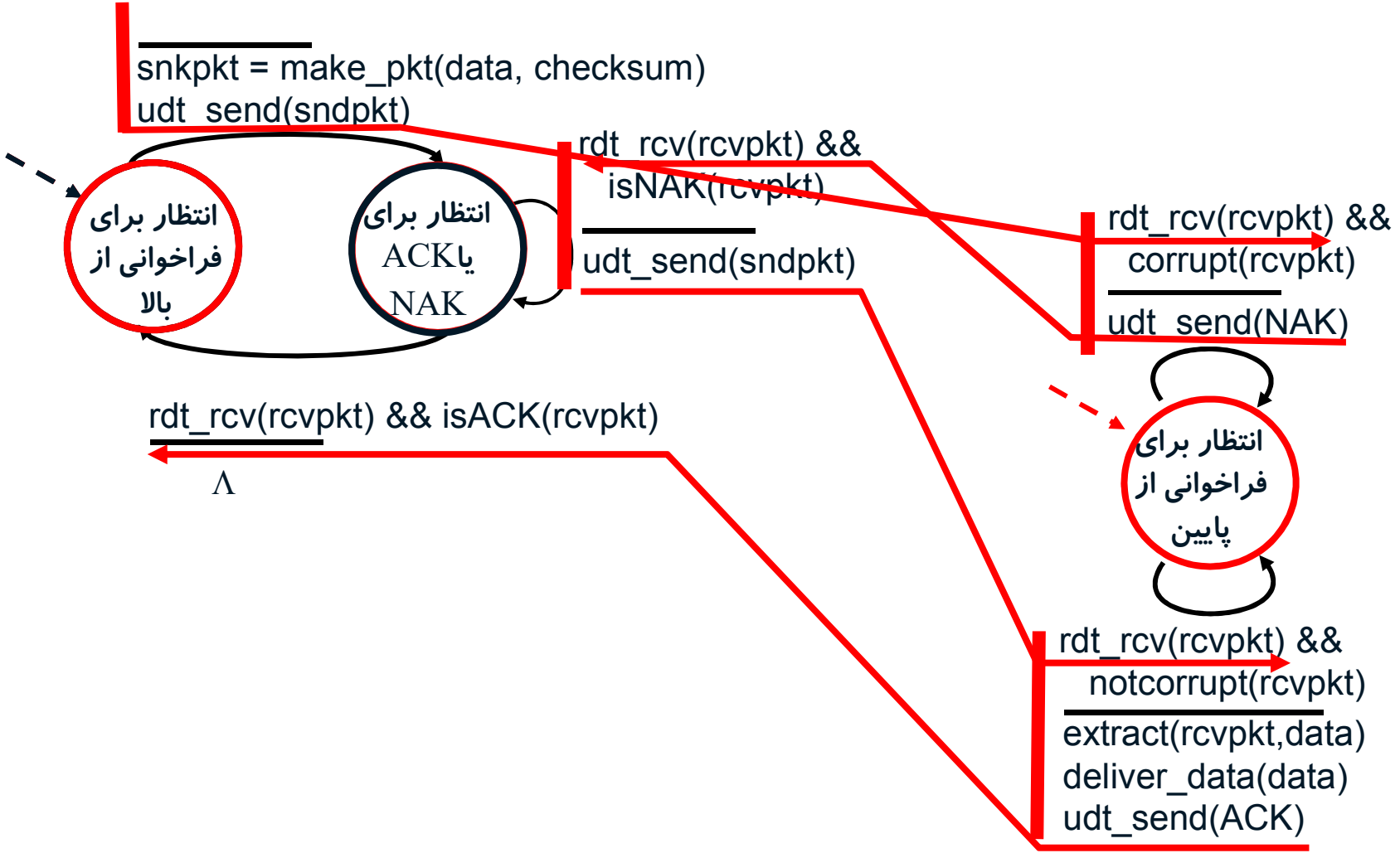
فرستنده-rdt2.0

* فرستنده تا کسب اطمینان از سرنوشت بسته ارسالی هیچ بسته‌ای ارسال نمی‌کند.

گیرنده-rdt2.0




rdt_send(data)
snkpkt = make_pkt(data, checksum)
udt_send(sndpkt)
انتظار برای فراخوانی از بالا
انتظار برای ACK یا NAK
rdt_rcv(rcvpkt) && isNAK(rcvpkt)
udt_send(sndpkt)
rdt_rcv(rcvpkt) && isACK(rcvpkt)
Λ
rdt_rcv(rcvpkt) && corrupt(rcvpkt)
udt_send(NAK)
انتظار برای فراخوانی از پایین
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)
extract(rcvpkt,data)
deliver_data(data)
udt_send(ACK)


snkpkt = make_pkt(data, checksum)
udt_send(sndpkt)
انتظار برای فراخوانی از بالا
انتظار برای ACK یا NAK
rdt_rcv(rcvpkt) && isNAK(rcvpkt)
udt_send(sndpkt)
rdt_rcv(rcvpkt) && corrupt(rcvpkt)
udt_send(NAK)
rdt_rcv(rcvpkt) && isACK(rcvpkt)
Λ
انتظار برای فراخوانی از پایین
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)
extract(rcvpkt,data)
deliver_data(data)
udt_send(ACK)

خراب

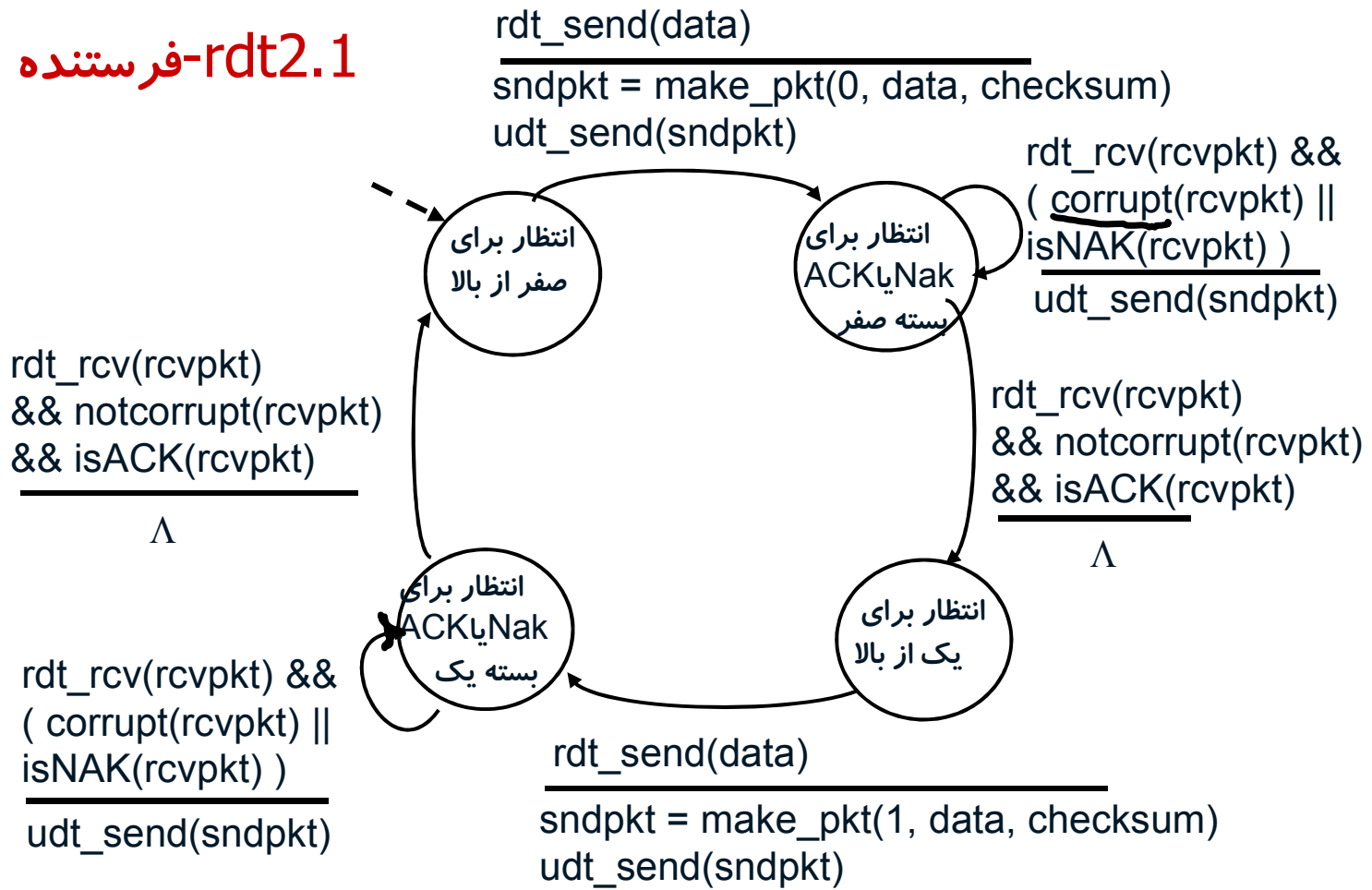
rdt2.0 یک مشکل جدید دارد: اگر بسته‌های پاسخ(ACK,NAK) گم شوند، چه اتفاقی میافتد؟

- فرستنده از سرنوشت بسته ارسال شده، نمی‌تواند مطلع شود.
- راه چاره: درصورتیکه فرستنده یک بسته ACK یا NAK خراب دریافت کند، بسته فعلی را دوباره ارسال می‌کند.اما بازهم یک مشکل داریم، ارسال مجدد بسته‌ها در گیرنده باعث ابهام می‌شود.(بسته دریافتی،جدید است یا تکرار بسته قبلی)
- برای حل مشکل از شماره ترتیب(Sequence number) استفاده می‌کنیم.

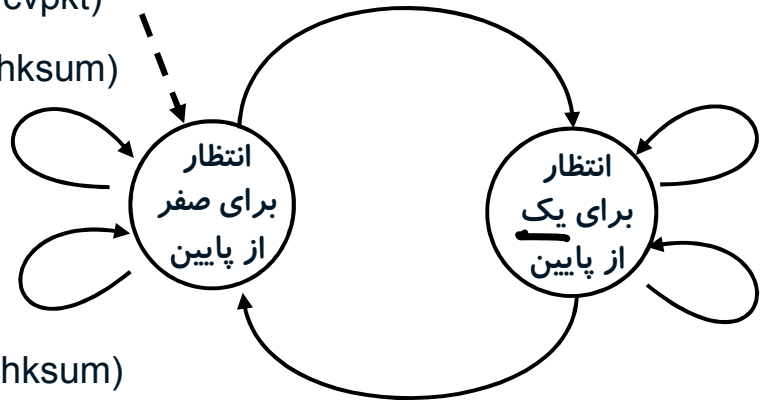
S & W

۰ ـ ۱

فرستنده-rdt2.1



گیرنده-rdt2.1



rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)
&& has_seq0(rcvpkt)

extract(rcvpkt,data)
deliver_data(data)
sndpkt = make_pkt(ACK, chksum)
udt_send(sndpkt)

rdt_rcv(rcvpkt) && (corrupt(rcvpkt)

sndpkt = make_pkt(NAK, chksum)
udt_send(sndpkt)

rdt_rcv(rcvpkt) &&
not corrupt(rcvpkt) &&
has_seq1(rcvpkt)

sndpkt = make_pkt(ACK, chksum)
udt_send(sndpkt)

rdt_rcv(rcvpkt) && (corrupt(rcvpkt)

sndpkt = make_pkt(NAK, chksum)
udt_send(sndpkt)

rdt_rcv(rcvpkt) &&
not corrupt(rcvpkt) &&
has_seq0(rcvpkt)

sndpkt = make_pkt(ACK, chksum)
udt_send(sndpkt)

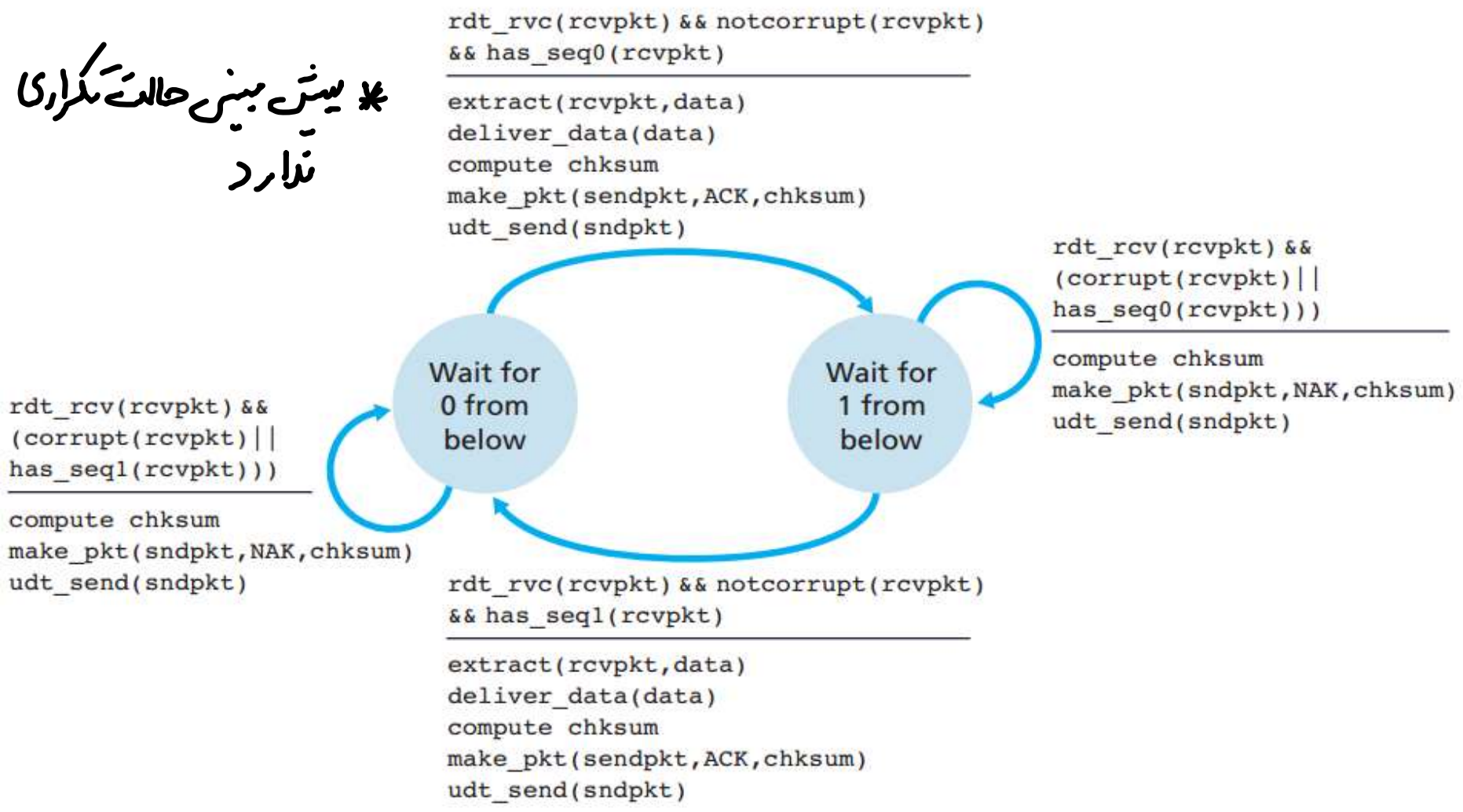
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)
&& has_seq1(rcvpkt)

extract(rcvpkt,data)
deliver_data(data)
sndpkt = make_pkt(ACK, chksum)
udt_send(sndpkt)

- **مثال (۱):اگر در پروتکل rdt2.1 گیرنده بصورت اسلاید بعدی باشد، آیا عملکرد درست خواهد بود؟**

بن بست رخ می دهد.

* پیش بینی حالت تکراری ندارد



در پروتکل rdt2.1:

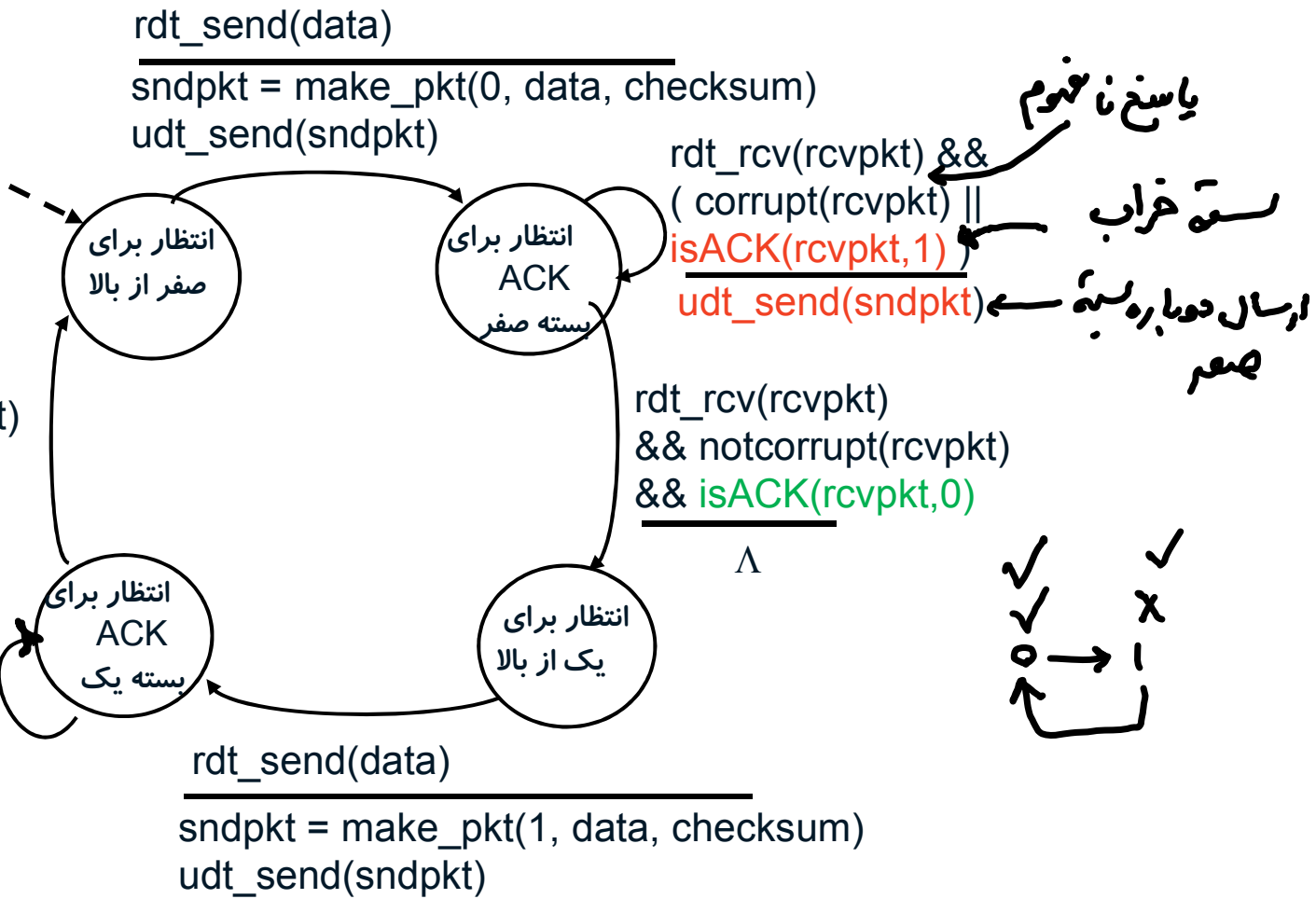
- وقتی بسته‌ای خارج از ترتیب انتظار گیرنده دریافت می‌شود، گیرنده برای آن تایید برمی‌گرداند، تا فرستنده دیگر آنرا ارسال نکند.
- وقتی بسته‌ای خراب به گیرنده می‌رسد؛ گیرنده برای آن بسته NAK برمی‌گرداند.

- اگر بخواهیم این پروتکل را به یک پروتکل تبدیل کنیم که فقط با ACK کار کند، چکار کنیم؟
  - چون پروتکل rdt2.1برای بسته های خراب NAKارسال می‌کند، کافیست با این حالت مثل حالت بسته تکراری رفتار نموده و ACKارسال کنیم.اما ACKرا برای بسته قبلی که درست دریافت شده است، ارسال می‌کنیم.لذا بسته‌های ACK باید شماره ترتیب داشته باشند.

پروتکل rdt2.2: پروتکل بدون NAK

- عملکردی یکسانی با پروتکل rdt2.1 دارد.و فقط از ACK استفاده می‌کند.
- به جای NAK گیرنده یک ACK را برای بسته خراب ارسال می‌کند،که در واقع این ACK ، تایید دوباره بسته قبلی است.(یک تایید تکراری).
- برای اینکه تایید ارسال شده در فرستنده مشخص باشد، که برای کدام بسته ارسال شده است(بسته اخیر یا بسته قبلی) از شماره ترتیب استفاده می‌شود.

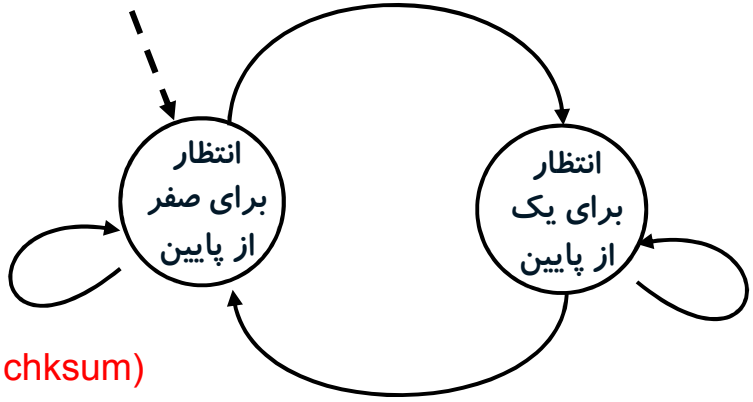
فرستنده-rdt2.2



گیرنده-rdt2.2

rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)
&& has_seq0(rcvpkt)

extract(rcvpkt,data)
deliver_data(data)
sndpkt = make_pkt(ACK, 0,chksum)
udt_send(sndpkt)



(rdt_rcv(rcvpkt) &&
corrupt(rcvpkt))||
has_seq1(rcvpkt)

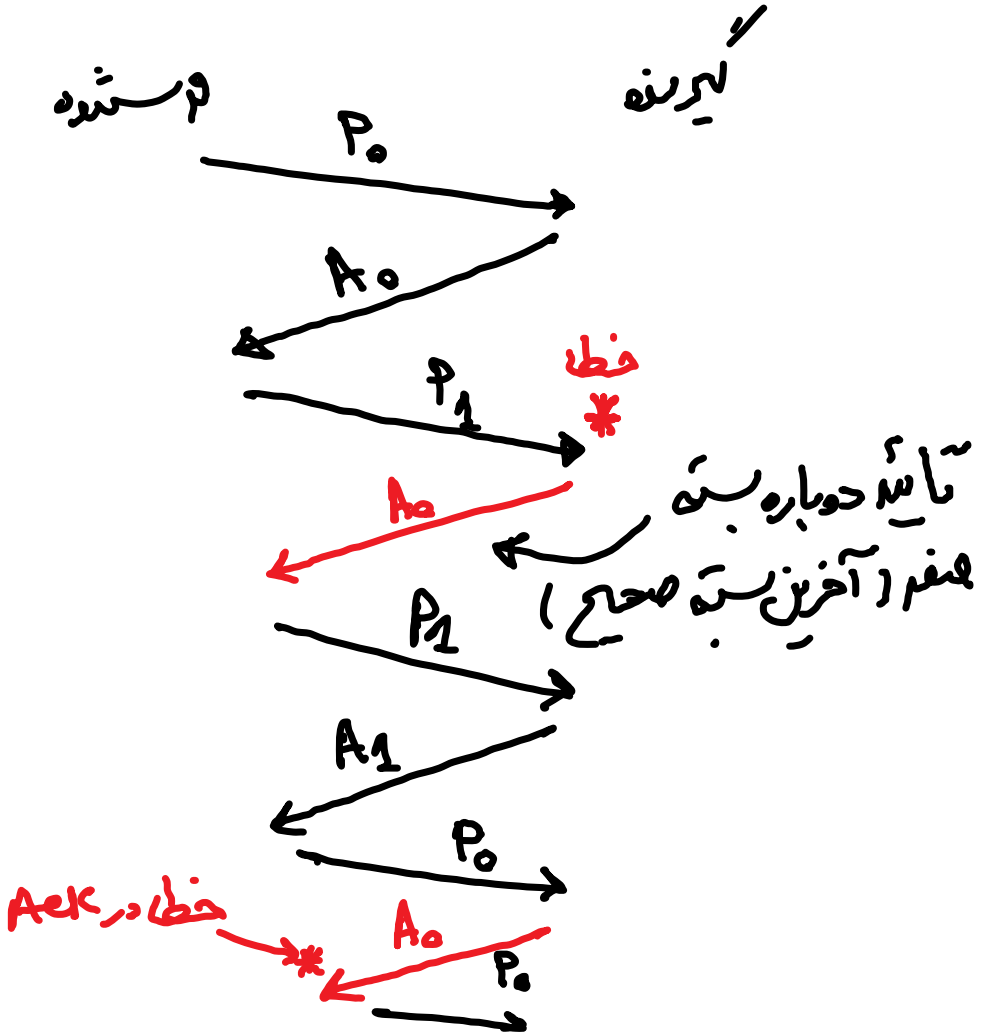
sndpkt = make_pkt(ACK, 1,chksum)
udt_send(sndpkt)

rdt_rcv(rcvpkt) &&
corrupt(rcvpkt) &&
has_seq0(rcvpkt)

sndpkt = make_pkt(ACK,0, chksum)
udt_send(sndpkt)

rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)
&& has_seq1(rcvpkt)

extract(rcvpkt,data)
deliver_data(data)
sndpkt = make_pkt(ACK,1, chksum)
udt_send(sndpkt)

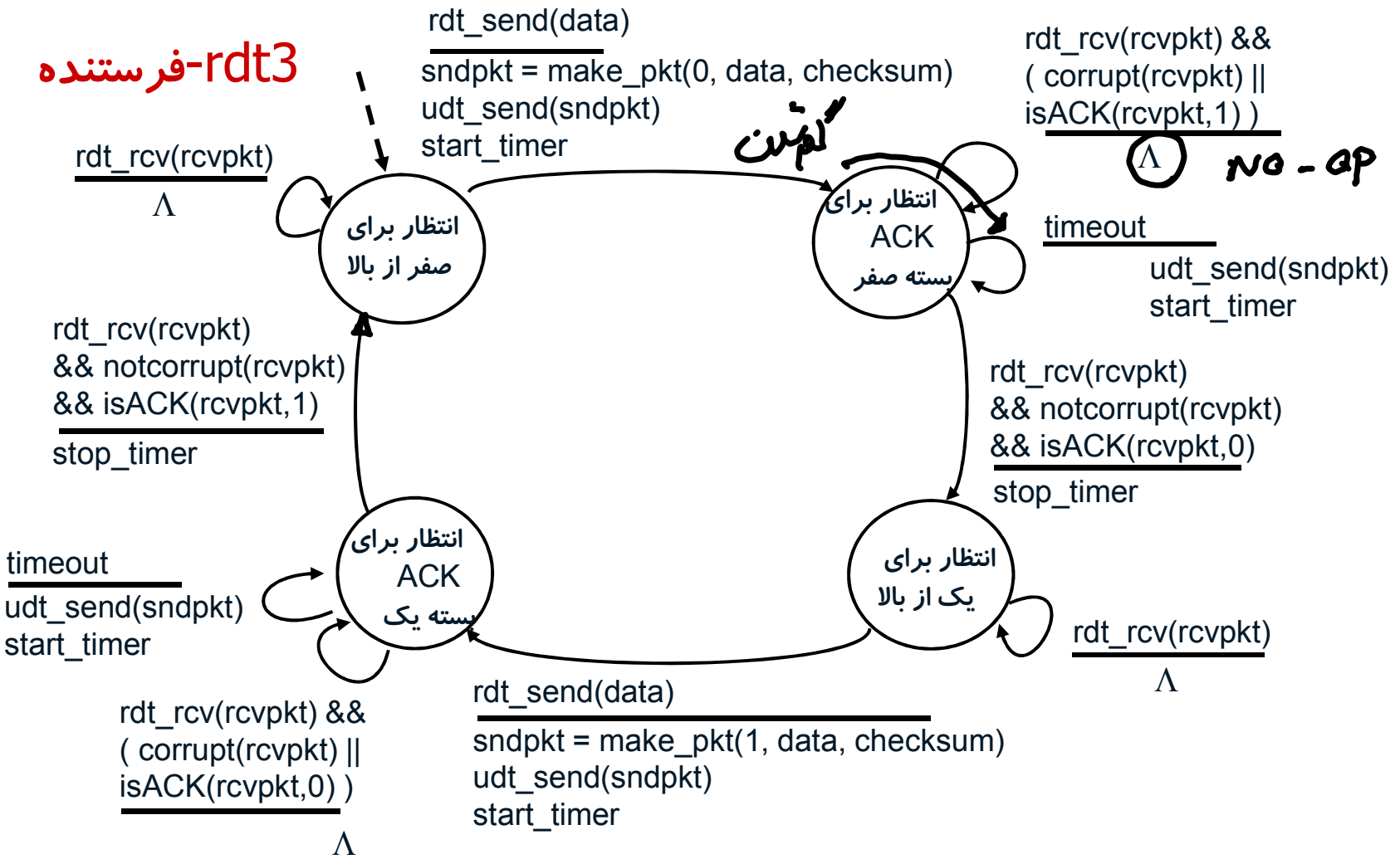

گیرنده
فرستنده
P0
A0
خطا
P1
A0
تأیید دوباره بسته
صفر (آخرین بسته صحیح)
P1
A1
P0
خطا در Ack
A0
P0

پروتکل rdt3.0: پروتکل انتقال داده مطمئن روی کانال با تلفات بسته و خطای بیتی

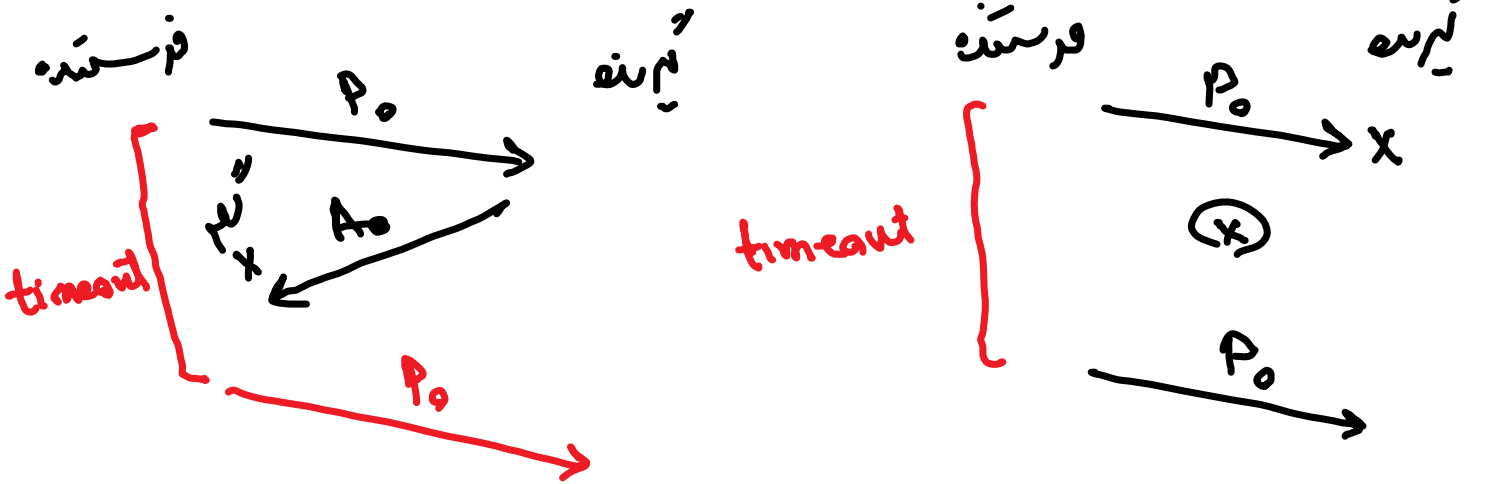
- پروتکل بخاطر احتمال گم شدن بسته‌ها در کانال با دو چالش جدید روبرو است:
  - تشخیص گم شدن بسته
  - واکنش مناسب و اصولی در هنگام گم شدن بسته

* اضافه شدن زمان سنج به فرستنده

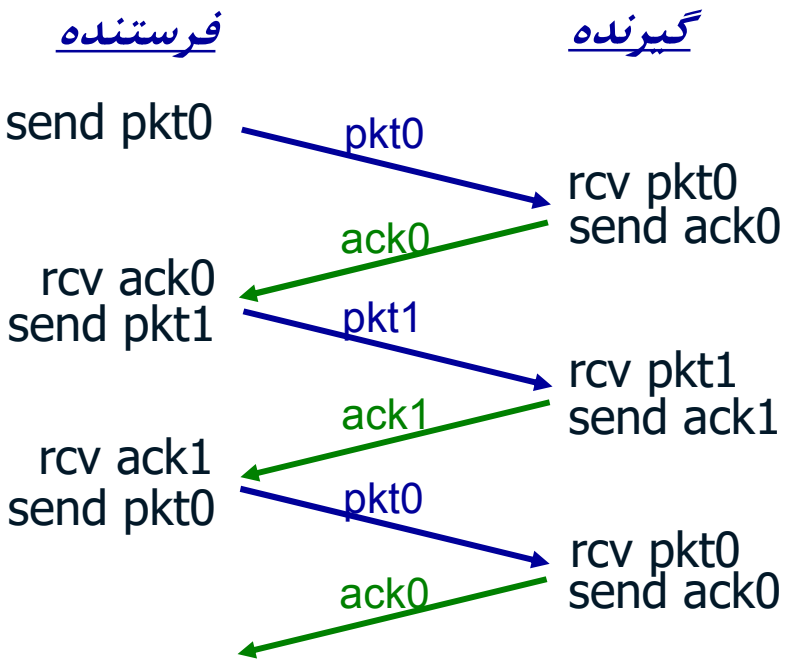
فرستنده-rdt3



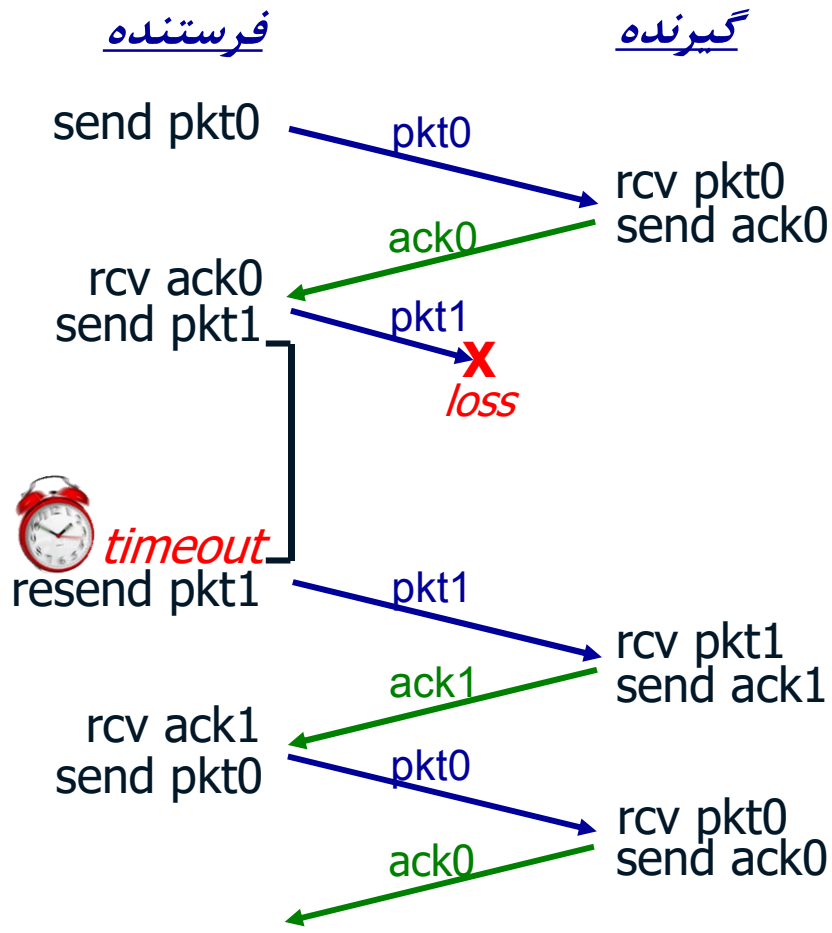
rdt 3.0



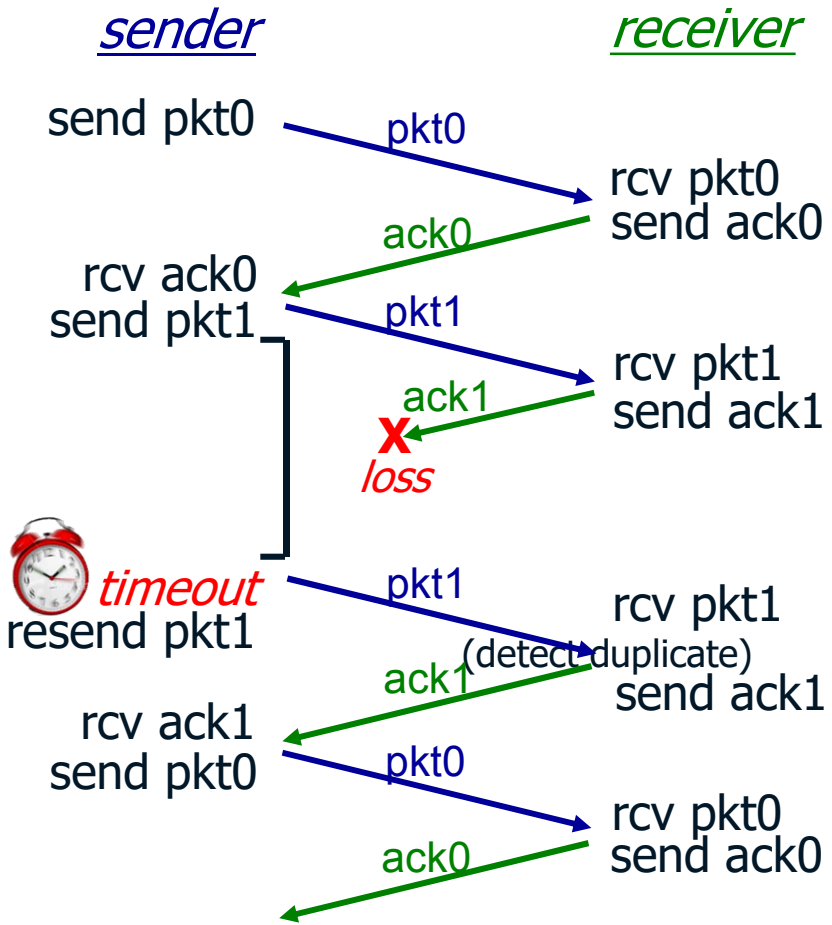
* در گیرنده Rdt 3.0 برای بسته‌های Ack شماره ترتیب نیاز نیست



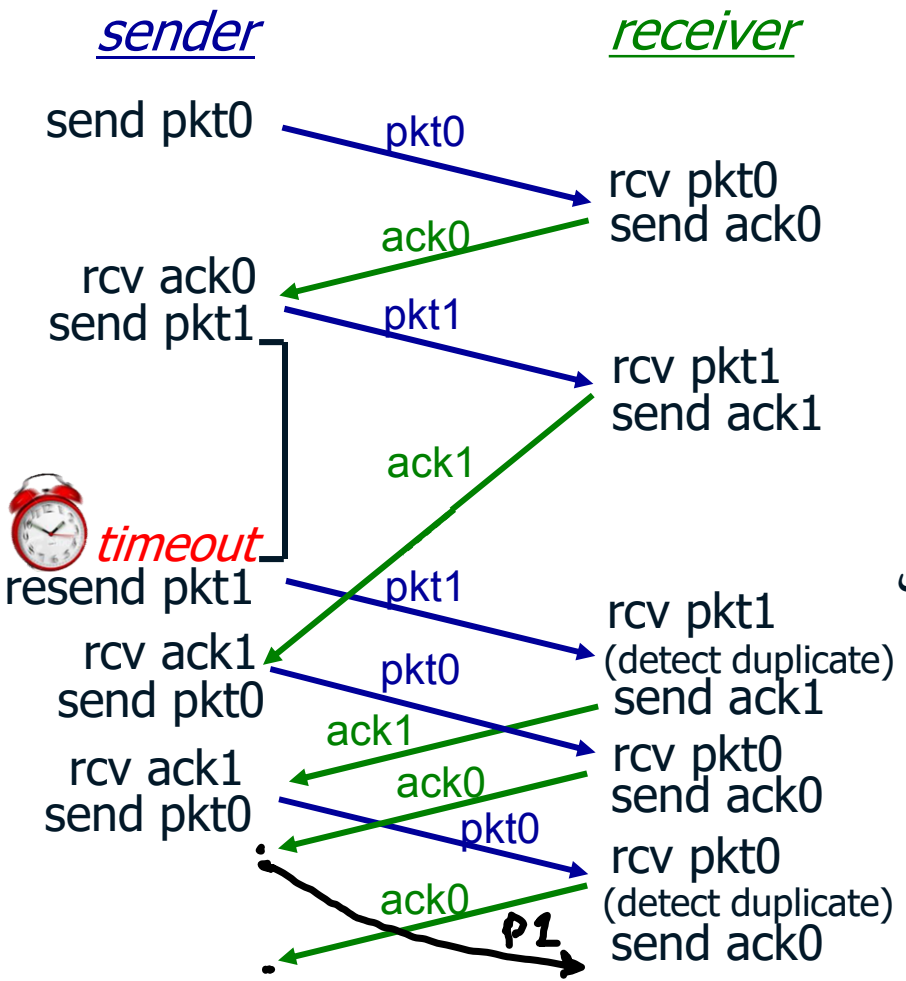
عملکرد درحالت بدون تلفات



عملکرد در حالت گم شدن بسته

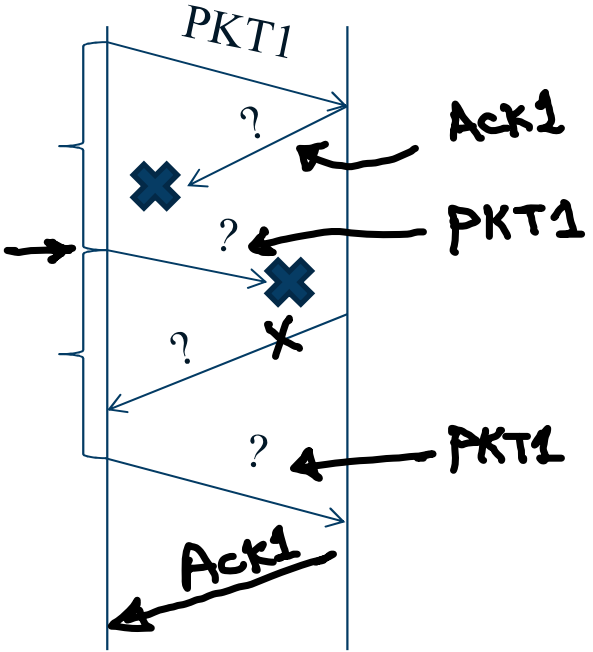


عملکرد در حالت گم شدن ACK



عملکرد در حالت انقضای زودرس

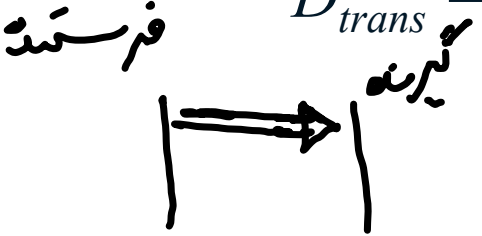
- **مثال (۲):از پروتکل rdt3.0برای ارسال بسته‌ها استفاده شده است.به جای علامت سوال‌ها چه چیزی قرار می‌گیرد؟**

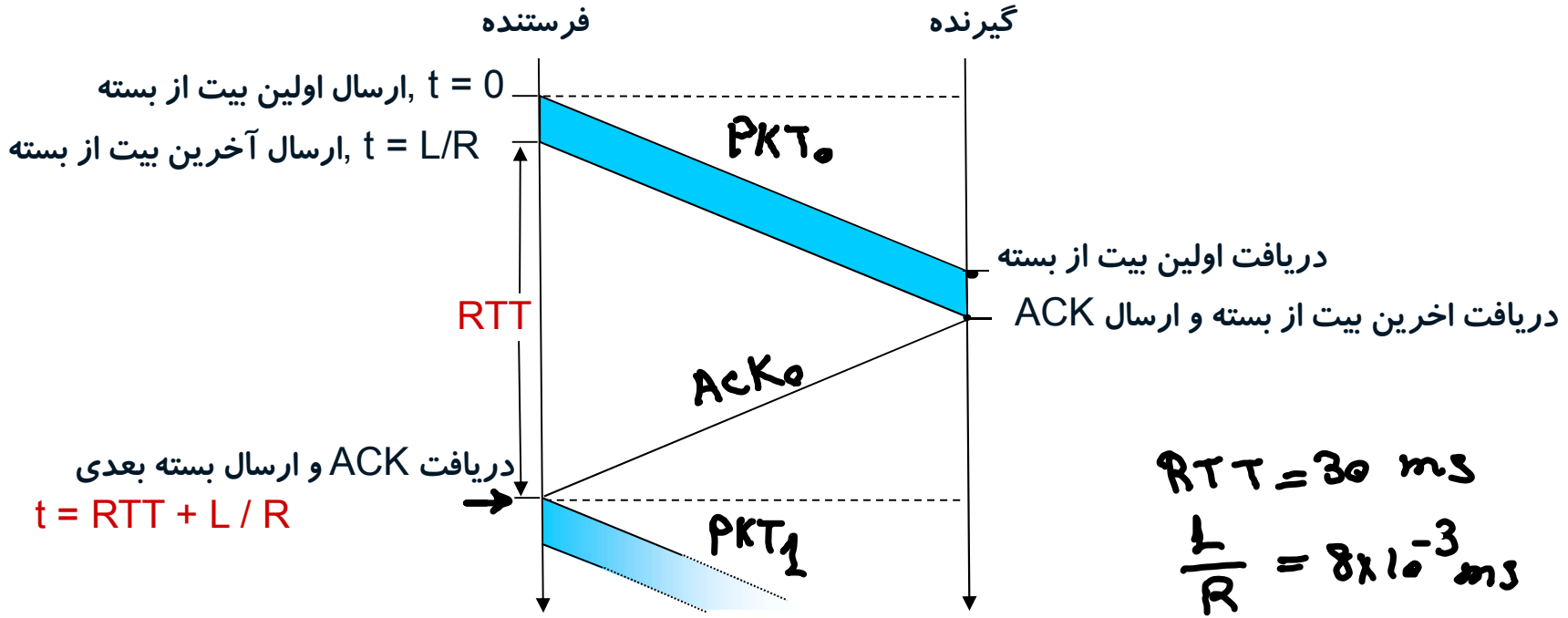


کارایی در پروتکل rdt3.0

- rdt3.0 بدرستی عمل می‌کند، اما بشدت کارایی پایینی دارد.
- اگر یک لینک با نرخ ارسال 1Gbps داشته باشیم.آنگاه زمان ارسال یک بسته با ۸۰۰۰ بیت لصورت زیر است:

$$D_{trans} = \frac{L}{R} = \frac{8000\ bits}{10^9\ bits/sec} = 8\ microsecs$$





$RTT = 30\ ms$

$\frac{L}{R} = 8 \times 10^{-3}\ ms$

$$U_{sender} = \frac{L/R}{RTT + \frac{L}{R}} = \frac{0.008}{30.008} = 27 \times 10^{-5} = \% 27 \times 10^{-3}$$

میلی ثانیه 30.008 — بایت 1000

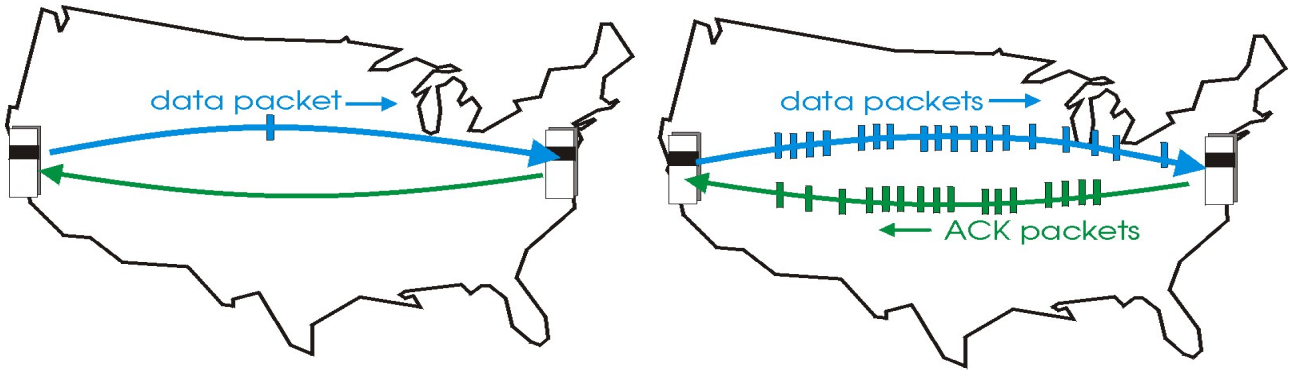
10^3 — x — $\rightarrow x = \frac{10^6}{30.008} = 266.59\ Kbps$

$1\ Gbps \longrightarrow 266.59\ Kbps$

انتقال بسته‌ها بصورت خط لوله

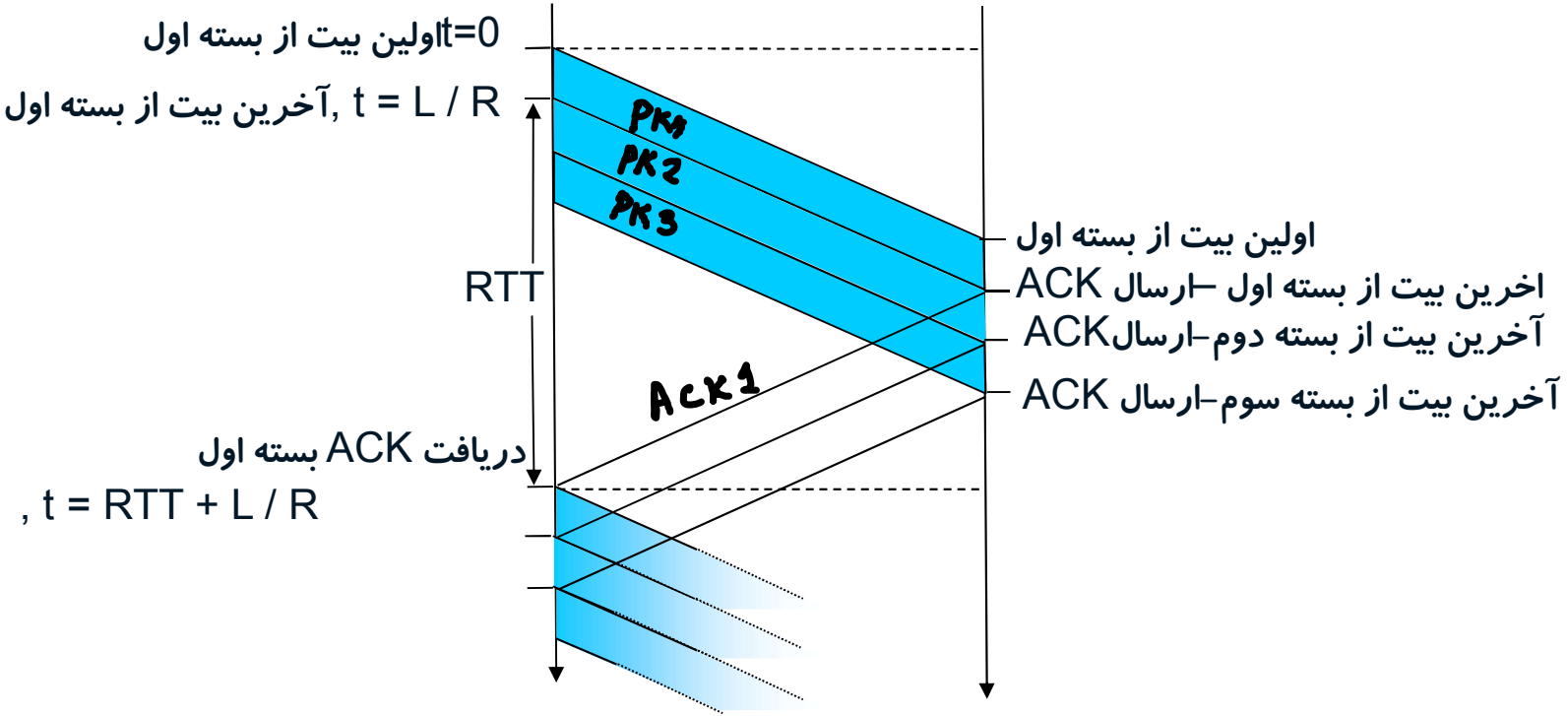
- برای بالا بردن میزان گذردهی فرستنده اجاز دارد،که تعدادی از بسته‌ها را پشت‌سرهم ارسال نموده و منتظر دریافت تایید(ACK) بماند.
- در این حالت چون چندین بسته باهم ارسال می‌شوند، باید تعداد شماره‌های ترتیب بیش از دو باشد.

- در این حالت باید در فرستنده و گیرنده بافر وجود داشته باشد.



(a) a stop-and-wait protocol in operation

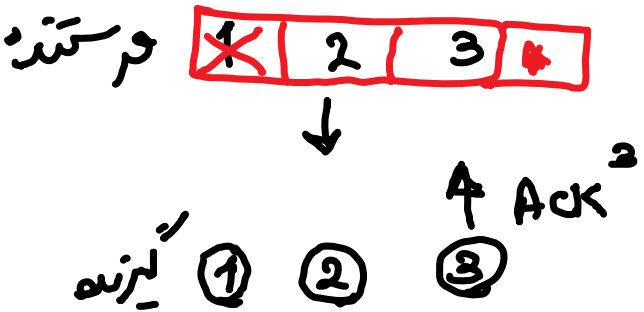
(b) a pipelined protocol in operation



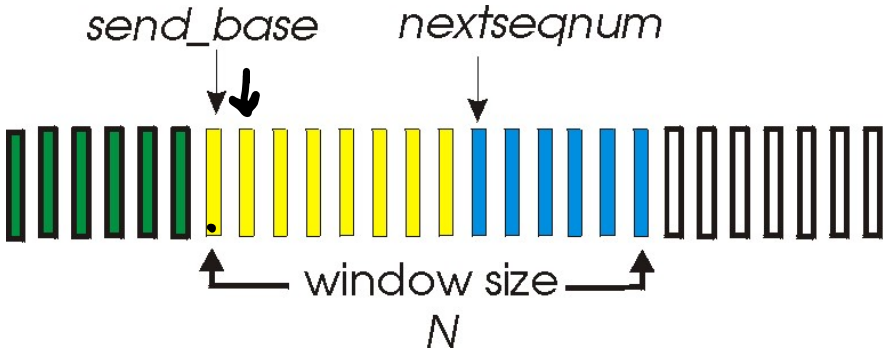
$$U_{sender} = \frac{3L / R}{RTT + L / R} = \frac{.0024}{30.008} = 0.00081$$

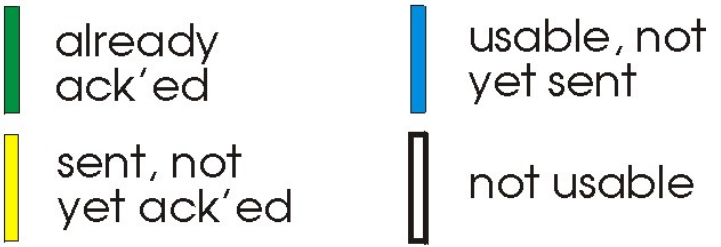
پروتکل خط لوله‌ای GBN
GO-Back-N

- در این روش فرستنده می‌تواند، حداکثرN بسته تایید نشده را وارد پایپ لاین کند.
- در این روش گیرنده از تایید(ACK) تجمعی استفاده می‌کند.
- در این روش زمان‌سنج وجود دارد، و بسته‌های ارسال‌شده و تاییدنشده بعد از انقضای زمان‌سنج دوباره ارسال می‌شوند.



G-BN


send_base
nextseqnum
window size
N


already ack'ed
sent, not yet ack'ed
usable, not yet sent
not usable

rdt_send(data)

if (nextseqnum < base+N) {
  sndpkt[nextseqnum] = make_pkt(nextseqnum,data,chksum)
  udt_send(sndpkt[nextseqnum])
  if (base == nextseqnum)
    start_timer
  nextseqnum++
  }
else
  refuse_data(data)

base+N − nextseqnum > 0

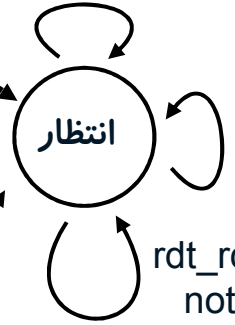
فرستنده-GBN

Λ

base=1
nextseqnum=1

timeout

start_timer
udt_send(sndpkt[base])
udt_send(sndpkt[base+1])
…
udt_send(sndpkt[nextseqnum-1])

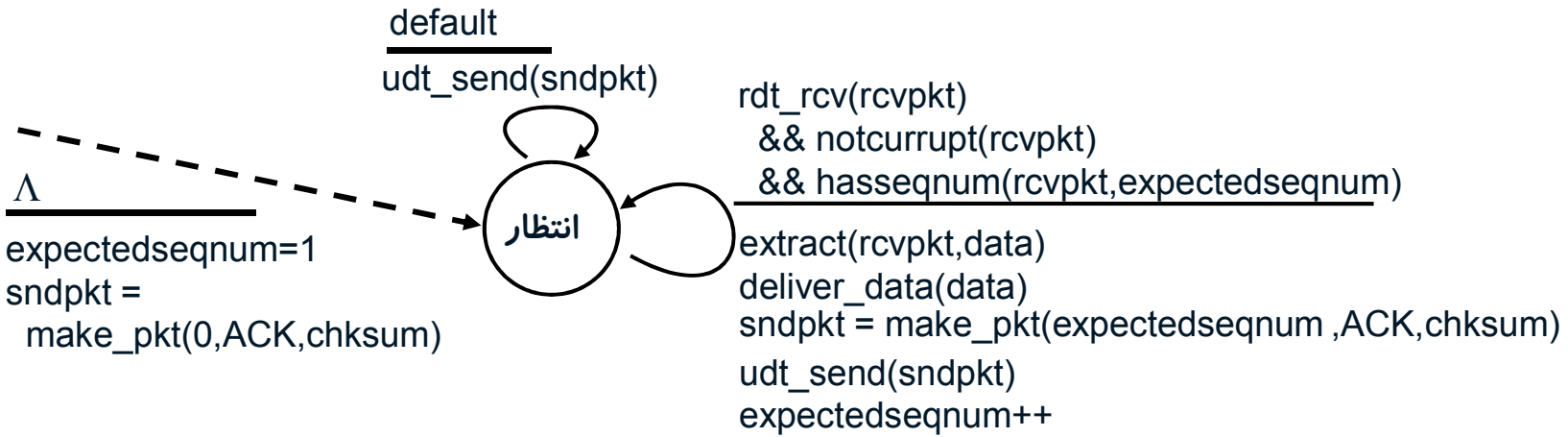


rdt_rcv(rcvpkt)
 && corrupt(rcvpkt)

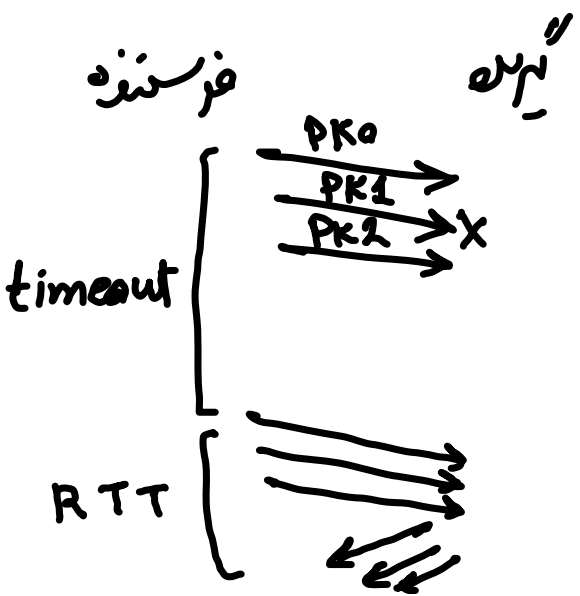
rdt_rcv(rcvpkt) &&
  notcorrupt(rcvpkt)

base = getacknum(rcvpkt)+1
If (base == nextseqnum)
  stop_timer
 else
  start_timer

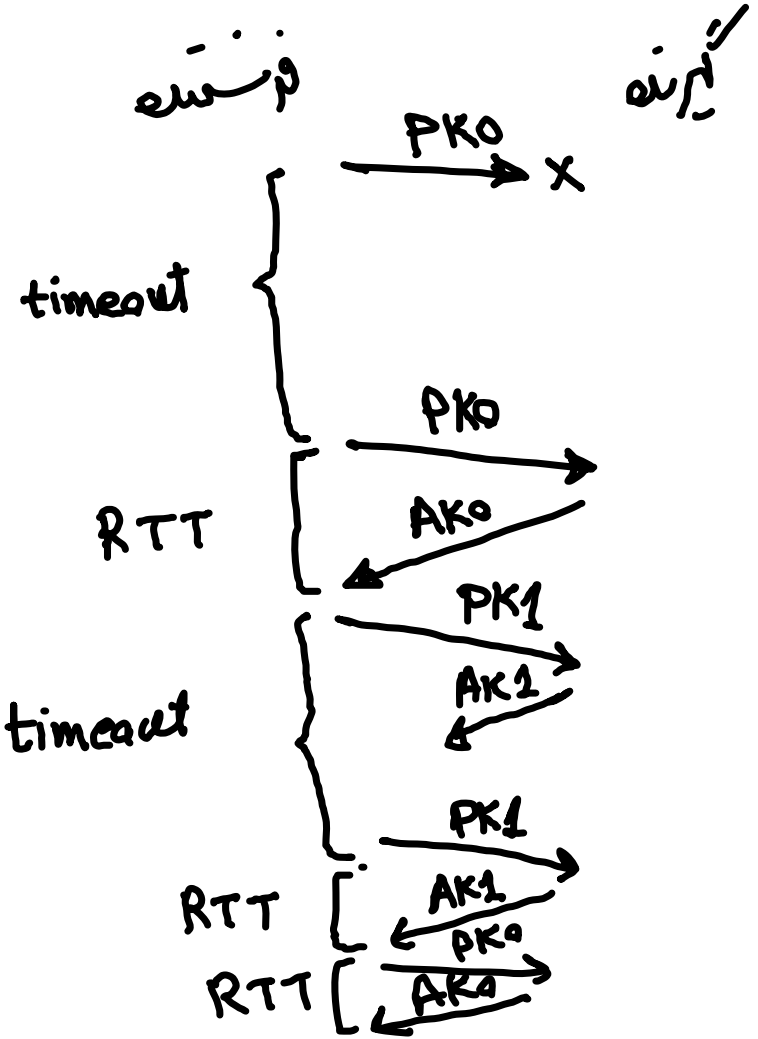
گیرنده-GBN


default
udt_send(sndpkt)
rdt_rcv(rcvpkt)
&& notcurrupt(rcvpkt)
&& hasseqnum(rcvpkt,expectedseqnum)
Λ
انتظار
expectedseqnum=1
sndpkt =
make_pkt(0,ACK,chksum)
extract(rcvpkt,data)
deliver_data(data)
sndpkt = make_pkt(expectedseqnum ,ACK,chksum)
udt_send(sndpkt)
expectedseqnum++

مثال (۳):در پروتکل rdt3.0 ، سه بسته ارسال می‌کنیم.درصورتیکه timeout=1ms باشد.اگر بسته اول در شبکه گم شود.وتایید بسته دوم نیز در برگشت از گیرنده خراب شود.وهر RTT برابر ۰/۷ میلی ثانیه باشد.زمان ارسال این سه بسته چقدر است؟(فرض کنید ارسال‌های مجدد همگی بدون مشکل انجام و تایید می‌شوند)

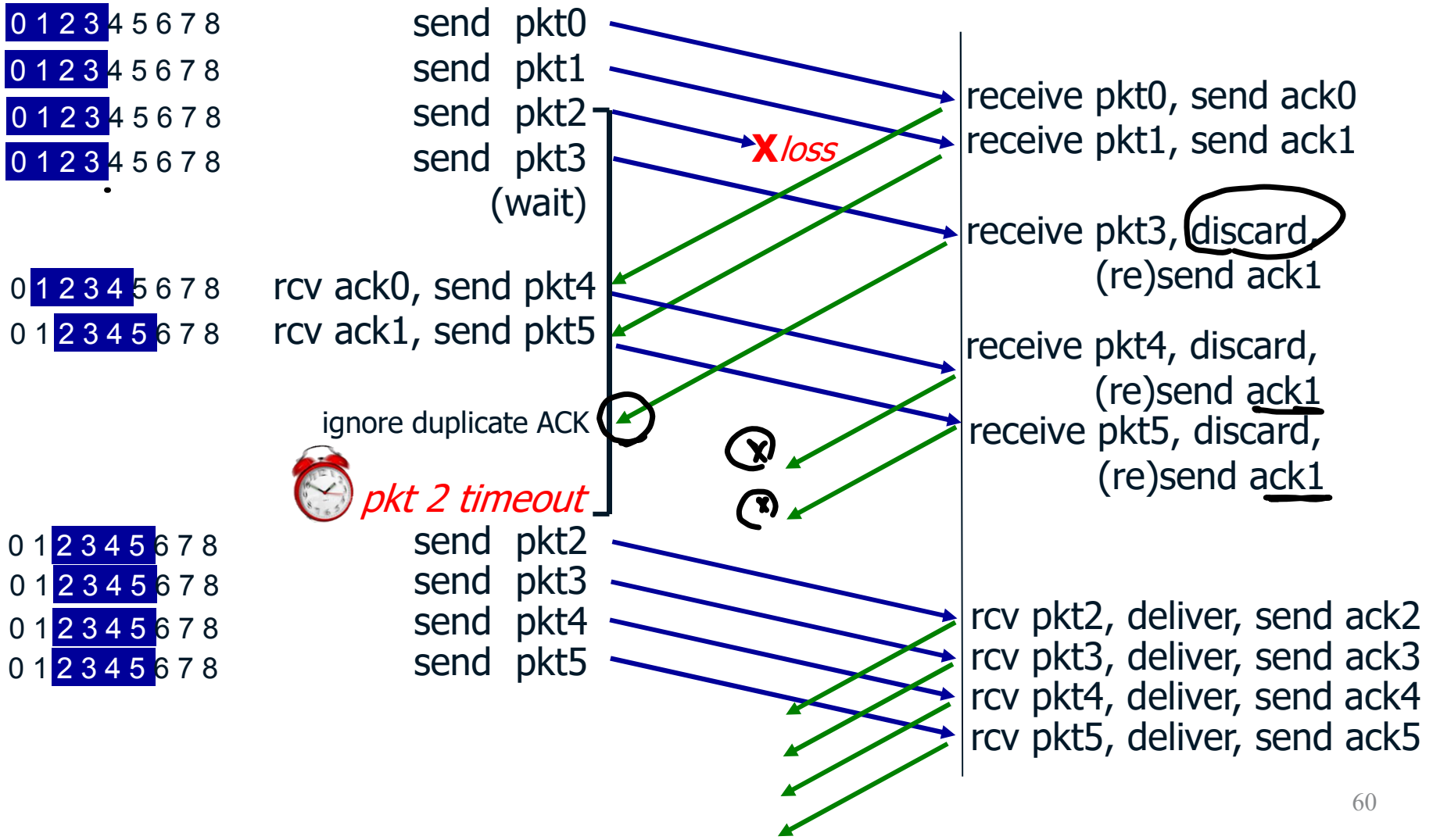


timeout + RTT = 1.7ms

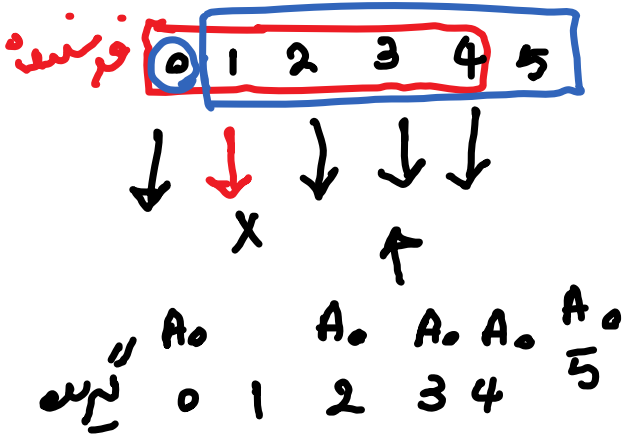


$T_{PK0} = 3\,RTT + 2\,timeout$

$= 3 \times 0.7 + 2 \times 1 = 4.1\ ms$

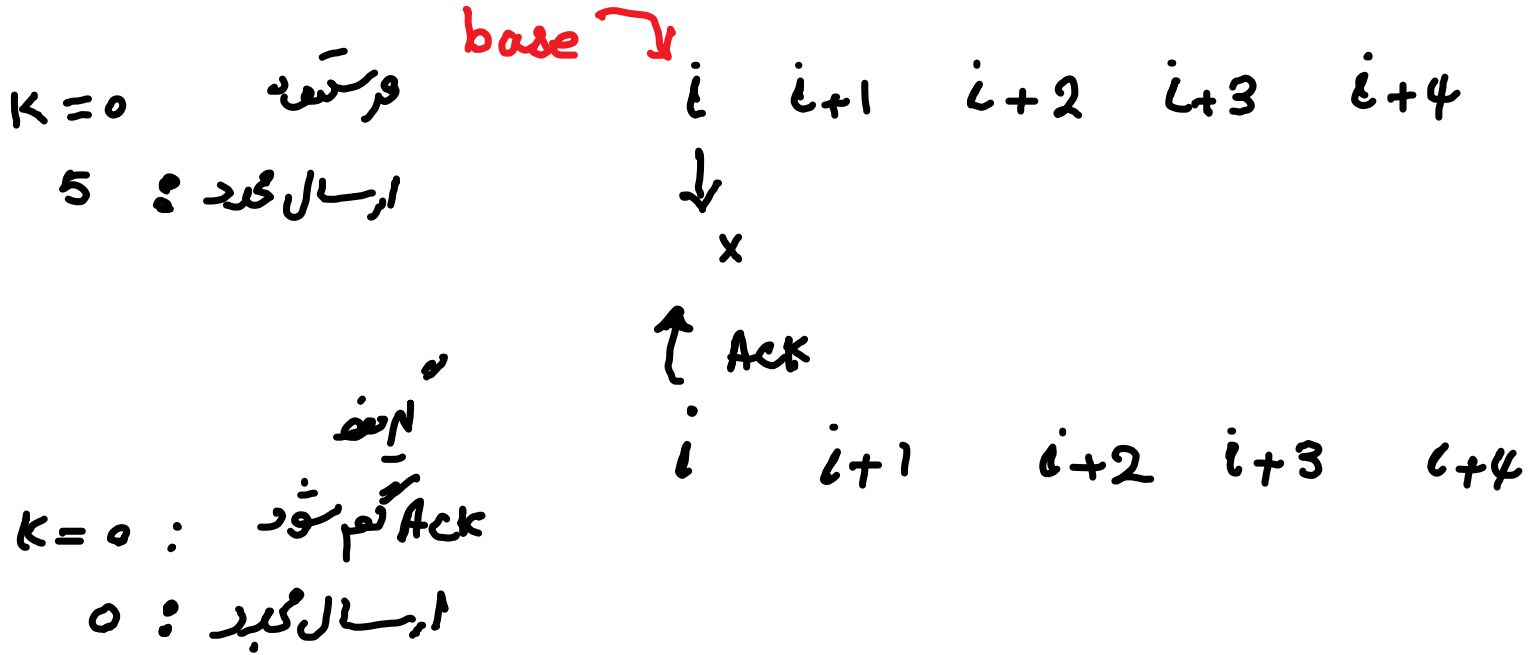

0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
send pkt0
send pkt1
send pkt2
send pkt3
(wait)
X loss
receive pkt0, send ack0
receive pkt1, send ack1
receive pkt3, discard,
(re)send ack1
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
rcv ack0, send pkt4
rcv ack1, send pkt5
receive pkt4, discard,
(re)send ack1
receive pkt5, discard,
(re)send ack1
ignore duplicate ACK
pkt 2 timeout
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
send pkt2
send pkt3
send pkt4
send pkt5
rcv pkt2, deliver, send ack2
rcv pkt3, deliver, send ack3
rcv pkt4, deliver, send ack4
rcv pkt5, deliver, send ack5

مثال (۴):در پروتکل GBN، با اندازه پنجره۵، پنج بسته پشت سرهم ارسال می‌شوند.اگر اعداد توالی[0,9] باشند و بسته با عدد توالی یک در شبکه گم شود.حال وقتیکه ACKبرمی‌گردد،قبل از timeout بسته ششم نیز ارسال و ACKآن برمی‌گردد.از این به به بعد چندبسته ارسال مجدد خواهیم داشت؟



مثال (۵):در پروتکل GBN، پنج بسته با شماره ترتیب‌های i، i+1، i+2، i+3، i+4 ارسال می‌شوند،درصورت گم شدن ACK بسته i+k چند بسته دوباره ارسال خواهد شد؟

k = 0 : 4



$i+K$ بسته با شماره ترتیب گم شود : $5-K$

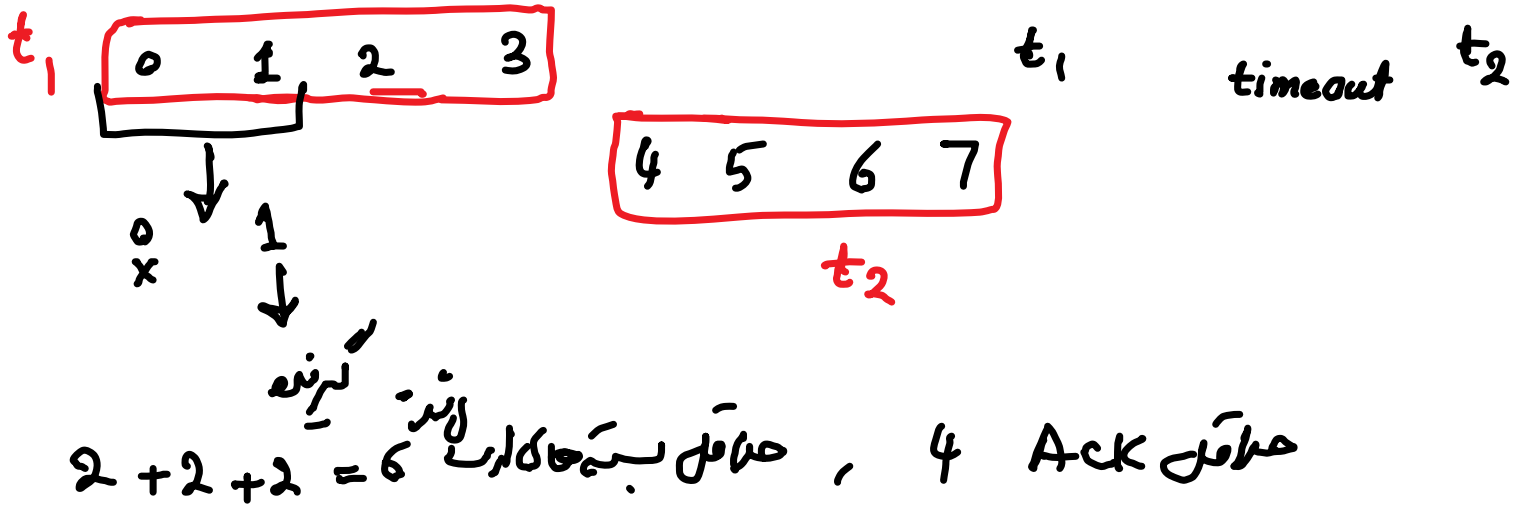
$i+K$ با شماره ترتیب Ack گم شود :

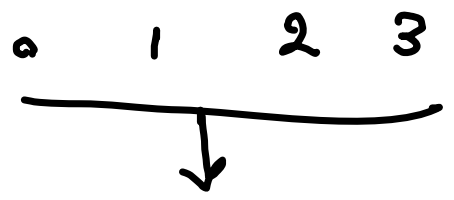
$K=0,1,2,3 \longrightarrow$ ارسال مجدد : a

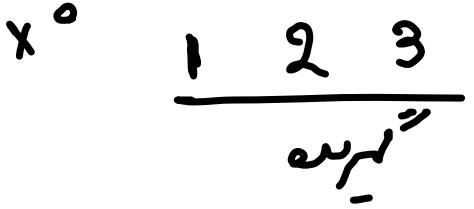
$K=4 \longrightarrow$ ارسال مجدد : 1

$K-\min(3,K)$

مثال (۶): پروتکل GBN، با اندازه پنجره۴ و فضای اعدادتوالی ۱۰۲۴ درنظربگیرید.اگر در لحظهt1پنجره خالی باشد.(بسته ها تا این لحظه همگی تایید شده‌اند).و فرستنده تعدادی بسته ارسال می‌کند،و حداقل یکی از این بسته‌ها سالم به گیرنده می‌رسد.اما گیرنده هیچ پیام ACK برای فرستنده ارسال نمی‌کند.
اگر بعد از زمان t2 بازهم پنجره خالی شود.
حداقل و حداکثر تعداد بسته‌های داده و ACK ارسال شده از لحظه t1 تا t2 چقدر است؟







t_1 timeout t_2

4 + 4 = 8 بسته

4 Ack

N

مثال (۷): پروتکل GBN، با اندازه پنجره۵ و فضای اعدادتوالی ۱۰۲۴ درنظربگیرید.فرض کنید در لحظهt گیرنده منتظر دریافت بسته‌ای با عدد توالی k باشد،همچنین رسانه انتقال ترتیب بسته‌ها را عوض نکند.
الف)در لحظه t ،در مجموعه اعداد توالی در پنجره ارسال چه مقادیری امکان دارد وجود داشته باشد؟
ب)در لحظه t مجموعه همه مقادیر امکانپذیر برای فیلد ACK، درحال بازگشت به فرستنده را بیابید.

K-N ... K-2 K-1

K K+1 .... K+N-1

هنوز نرسیده است

ACK

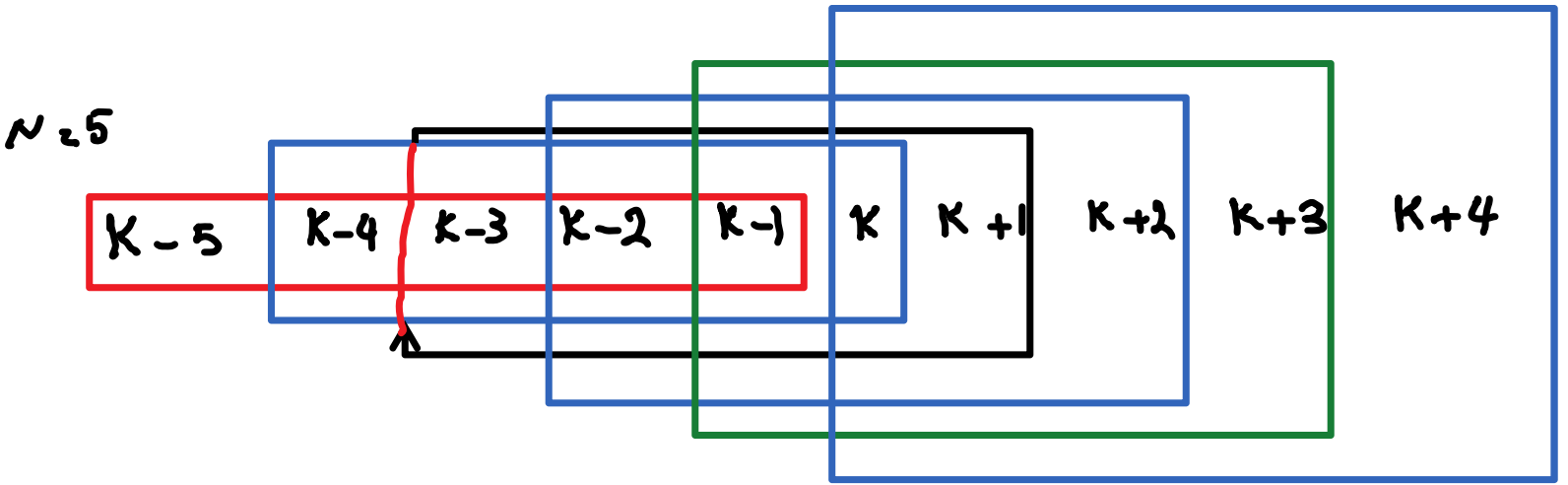
تمام پیام‌ها تأییده فرستنده میرسد

ACK

K-N ... K-3 K-2 K-1 K

K-N ... K-3 .... K-1 K

* اگر N = ۲ باشد، ۲+۱ حالت مختلف برای پنجره ارسال داریم

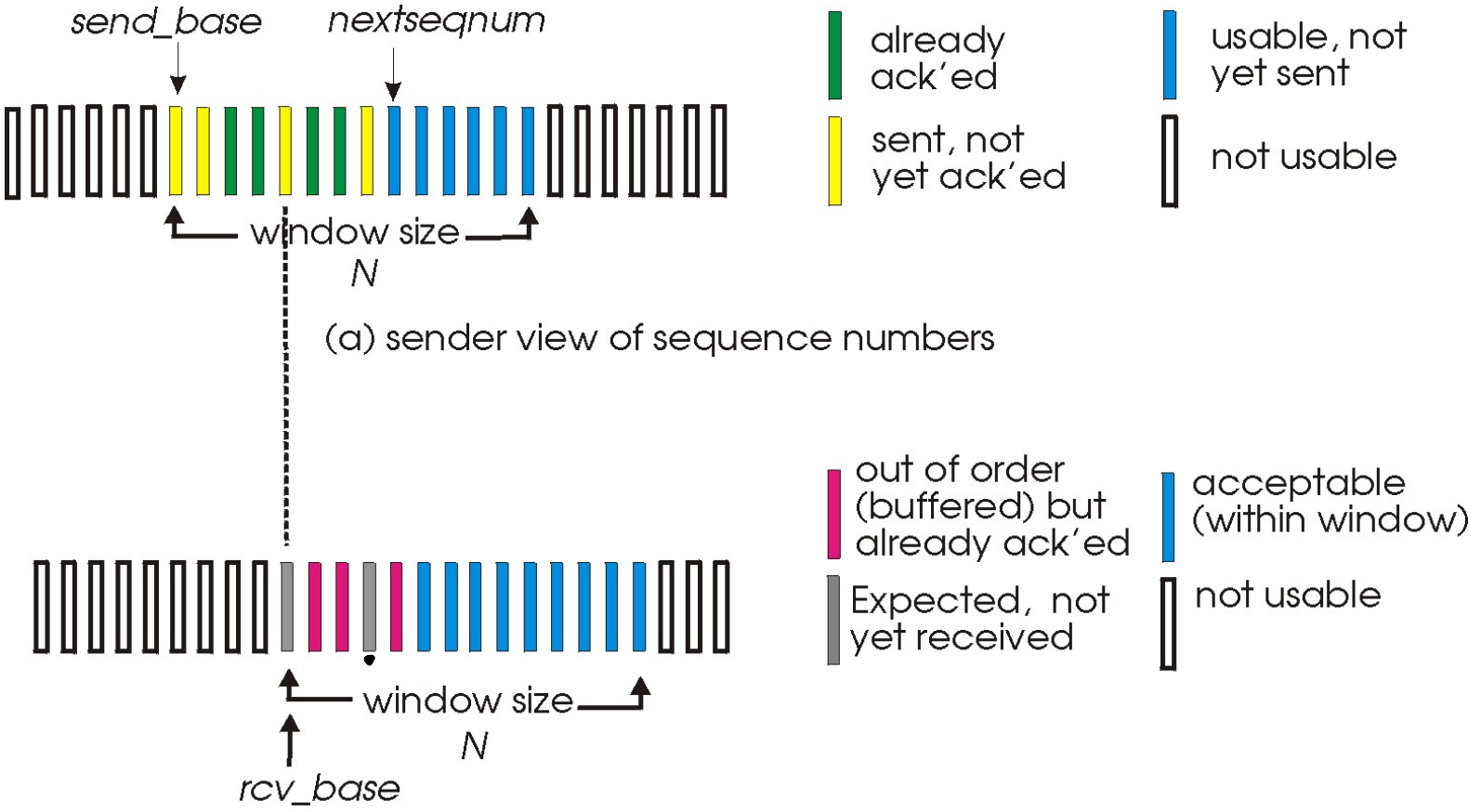


$[K-N, K-1]$

(ب

پروتکل خط لوله‌ای SR
Selective Repeat

- در پروتکلSR، برای هرکدام از بسته‌های صحیح دریافت شده؛ بصورت جداگانه یک ACK ارسال می‌شود.
- فقط بسته‌هایی دوباره ارسال می‌شوند،که ACK انها دریافت نشده است.
- در این روش زمان‌سنج وجود دارد، و بسته‌های ارسال‌شده و تاییدنشده بعد از انقضای زمان‌سنج دوباره ارسال می‌شوند.


send_base
nextseqnum
window size
N
already ack'ed
sent, not yet ack'ed
usable, not yet sent
not usable
(a) sender view of sequence numbers
out of order (buffered) but already ack'ed
Expected, not yet received
acceptable (within window)
not usable
window size
N
rcv_base
(b) receiver view of sequence numbers

فرستنده

1) دریافت داده از لایه بالا : اگر در پنجره جای خالی وجود داشته باشد ارسال انجام می شود .

2) timeout : برای اطلاع فرستنده از بسته های گم شده از زمان سنج استفاده می شود و با timeout بسته ای که ACK نگرفته است دوباره ارسال می شود

3) دریافت ACK : در فرستنده اگر بسته ای ACK دریافت کند و در لبه پنجره باشد ، آن گاه پنجره ارسال یک واحد جلو خواهد رفت .

گیرنده

۱- دریافت بسته سالم در محدوده [rcv_base, rcv_base+N-1]

محدوده پنجره دریافت

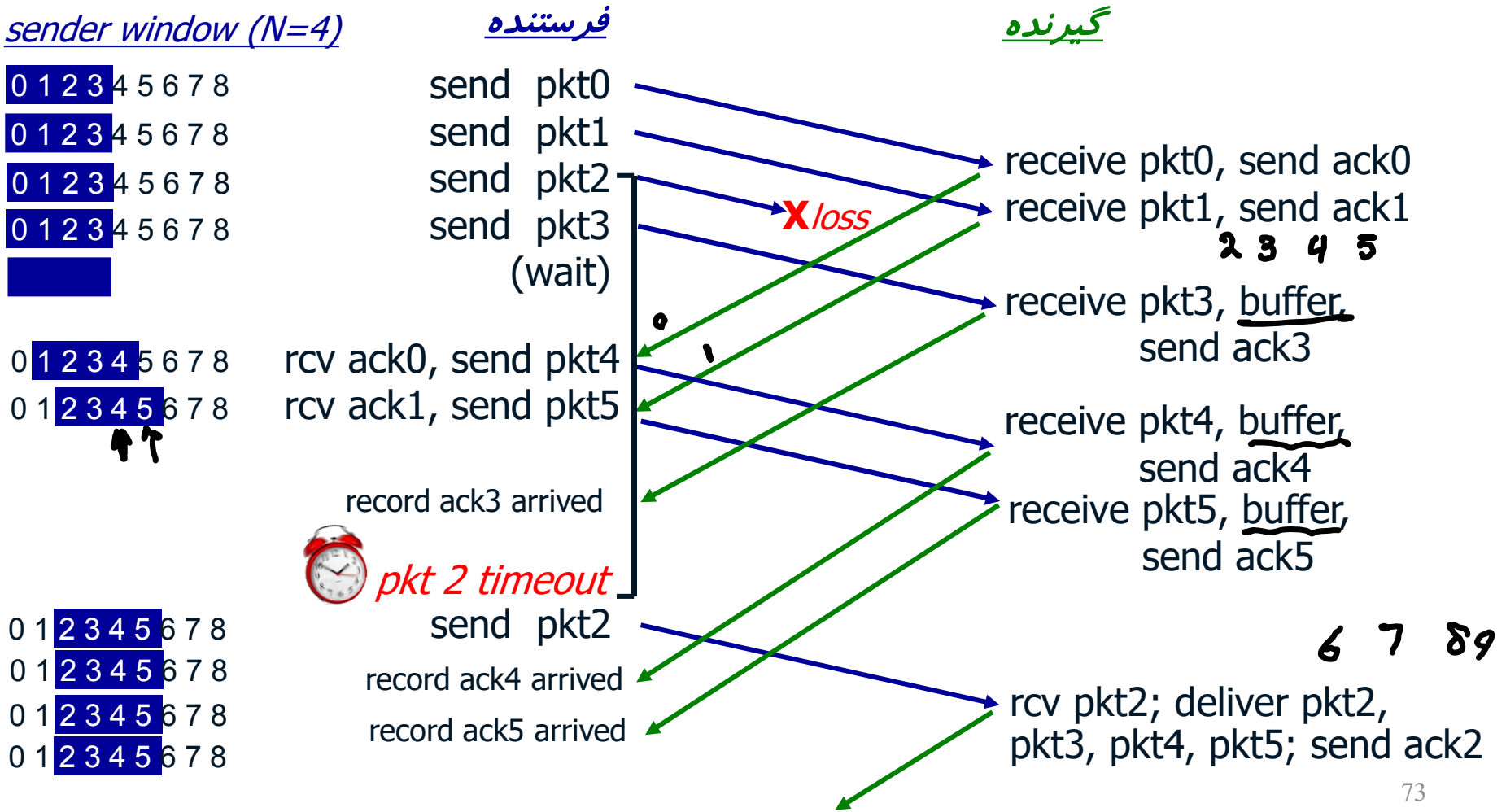
برای بسته مورد نظر یک ACK ارسال می‌شود و در صورتی که برای اولین دریافت نشده باشد در بافر قرار می‌گیرد.

اگر عدد توالی بسته دریافت شده با rcv_base برابر باشد آن گاه پنجره دریافت ۱ واحد جلو خواهد رفت و تمام بسته‌های دریافت شده بعد از آن تا اولین بسته دریافت نشده به لایه بالاتر تحویل داده می‌شوند و پنجره دریافت همان اندازه جلو می‌رود.

2- دریافت بسته سالم با عدد توالی [rcv_base-N , rcv_base-1]

در این حالت گیرنده باز هم برای بسته‌های دریافتی ACK برمی‌گرداند

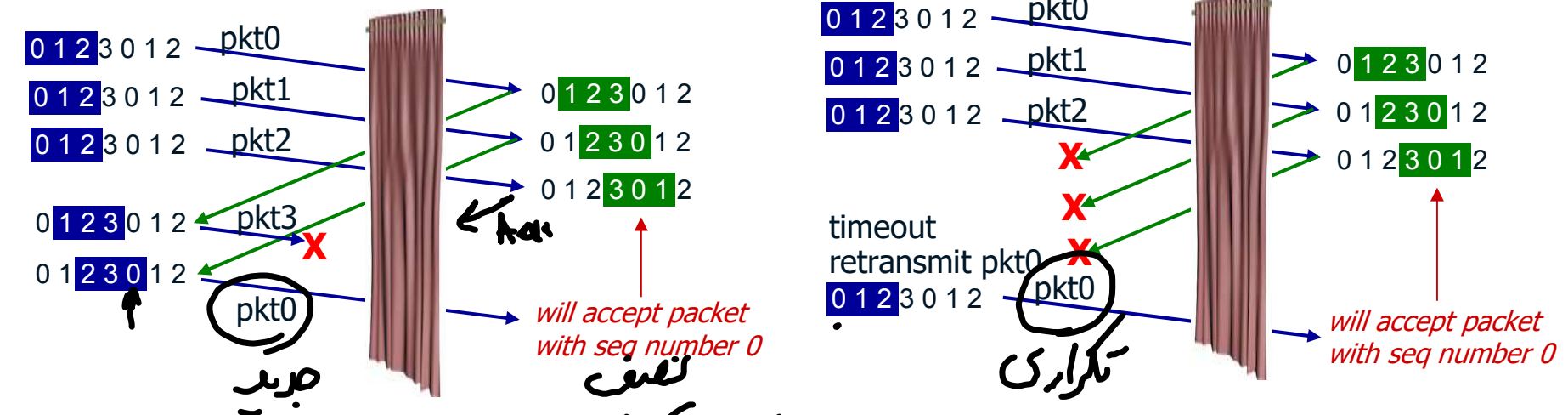
3- در غیر این صورت : بسته توسط گیرنده نادیده گرفته می‌شود


sender window (N=4)
فرستنده
گیرنده
0 1 2 3 4 5 6 7 8
send pkt0
send pkt1
send pkt2
send pkt3
(wait)
X loss
receive pkt0, send ack0
receive pkt1, send ack1
2 3 4 5
receive pkt3, buffer,
send ack3
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
rcv ack0, send pkt4
rcv ack1, send pkt5
receive pkt4, buffer,
send ack4
receive pkt5, buffer,
send ack5
record ack3 arrived
pkt 2 timeout
send pkt2
6 7 8 9
record ack4 arrived
record ack5 arrived
rcv pkt2; deliver pkt2,
pkt3, pkt4, pkt5; send ack2

چه رابطه‌ای بین پنجره ارسال و فضای اعداد توالی باشد؟

N=3 0,1,2,3

یکسان



* اندازه پنجره باید مساوی و کوچکتر از نصف فضای اعداد توالی باشد

اندازه پنجره : W $\quad [m, m+W-1]$

- پنجره گیرنده

- گیرنده منتظر سمت m است.

پنجره فرستنده : $m-W \quad \ldots \quad m-2 \quad m-1$: $[m-w, m-1]$

ابتدای پنجره فرستنده − ابتدای پنجره گیرنده $= (m+w-1) - (m-w) + 1$

$= 2w$

فضای اعداد توالی $\geq 2w$

رئوس مطالب:

- ❑ آشنایی با لایه انتقال و سرویس‌های آن
- ❑ مالتی پلکسینگ و دی مالتی پلکسینگ
- ❑ انتقال نامطمئن: UDP
- ❑ اصول انتقال داده قابل اطمینان

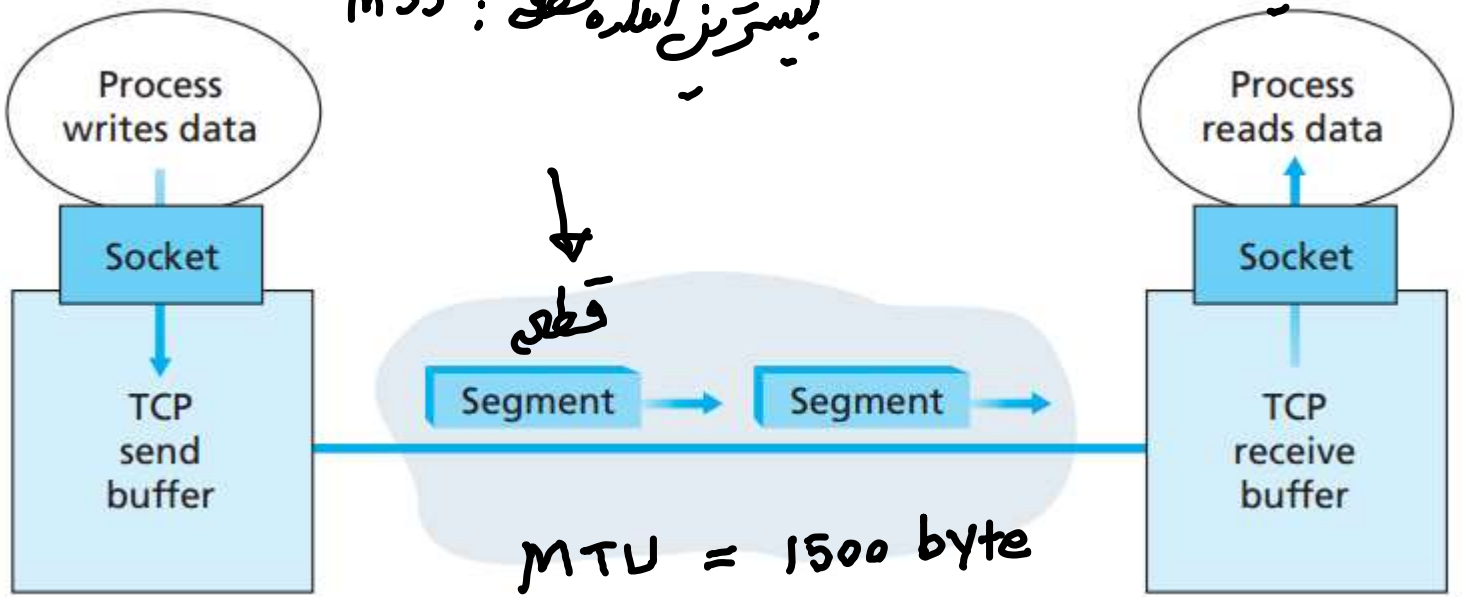
✅ **انتقال داده اتصال‌گرا: TCP**

- ❑ **اصول کنترل ازدحام**
- ❑ **کنترل ازدحام درTCP**

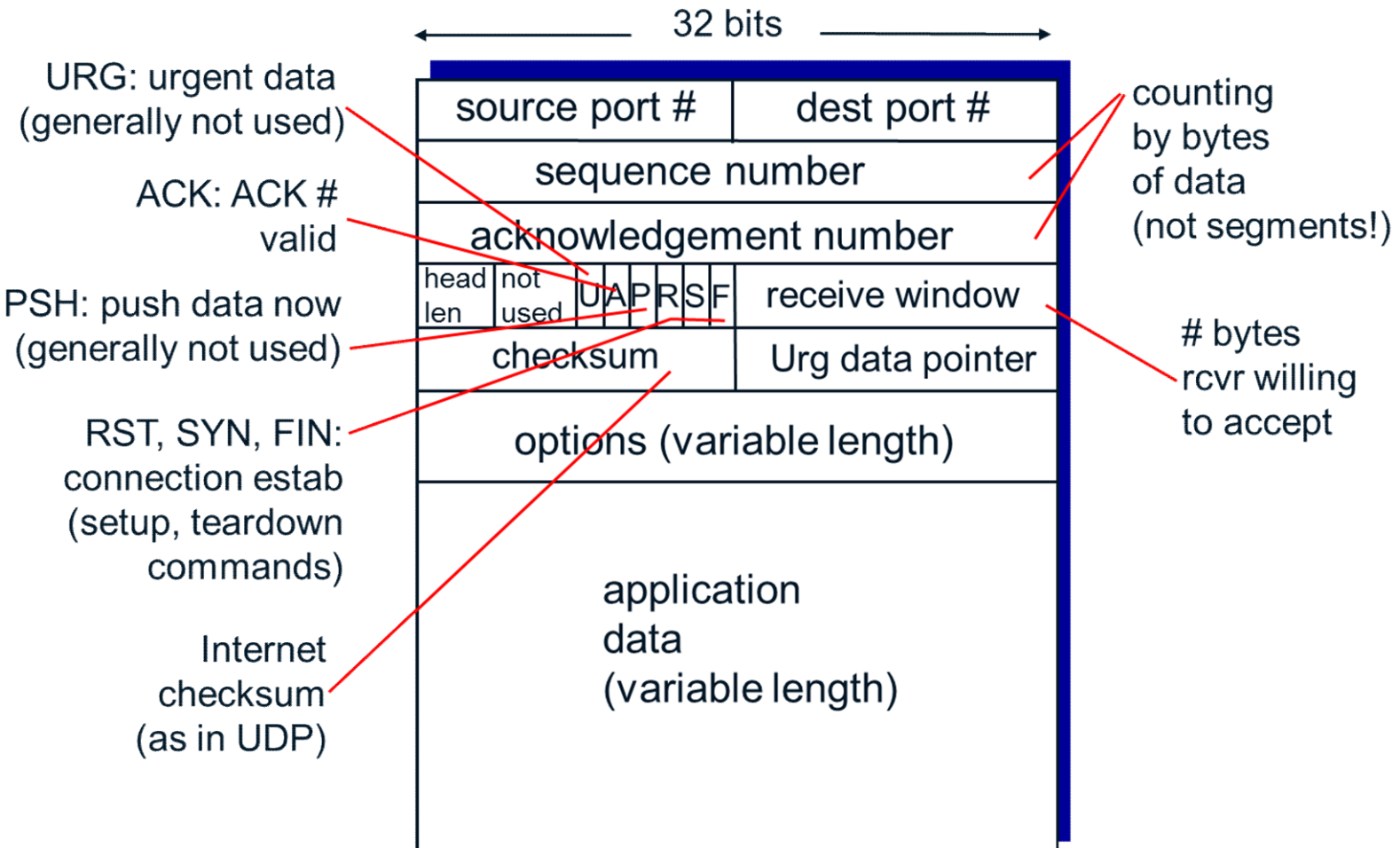
انتقال اتصال‌گرا:TCP

- TCP پروتکل انتقال داده اتصال‌گرا و قابل اطمینان در لایه انتقال اینترنت است.
- درTCP دو فرایند قبل از شروع مبادله داده‌ها، دست تکانی دارند.
- اتصال TCP یک سرویس دوطرفه کامل است.و همچنین همیشه نقطه به نقطه است.
- برای برقراری اتصال TCPبین دو میزبان، سه قطعه ردوبدل می‌شود.لذا روال برقراری اتصال را Three-way handshaking گویند.

تبادل داده بین فرستنده و گیرنده در TCP

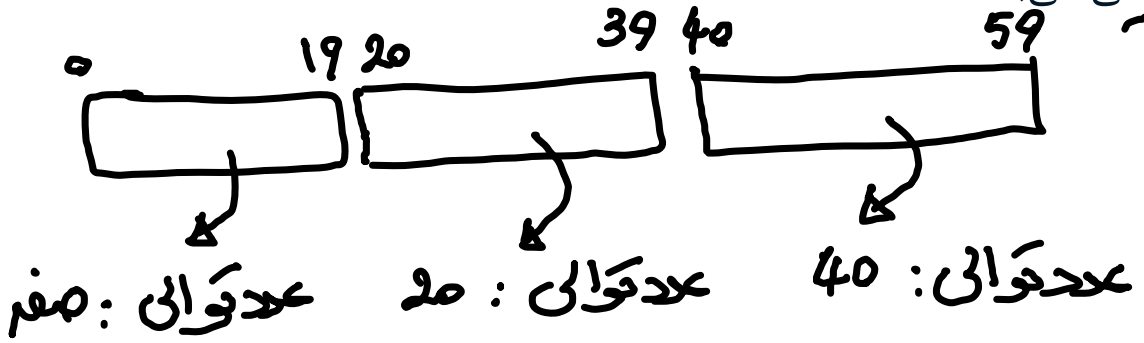


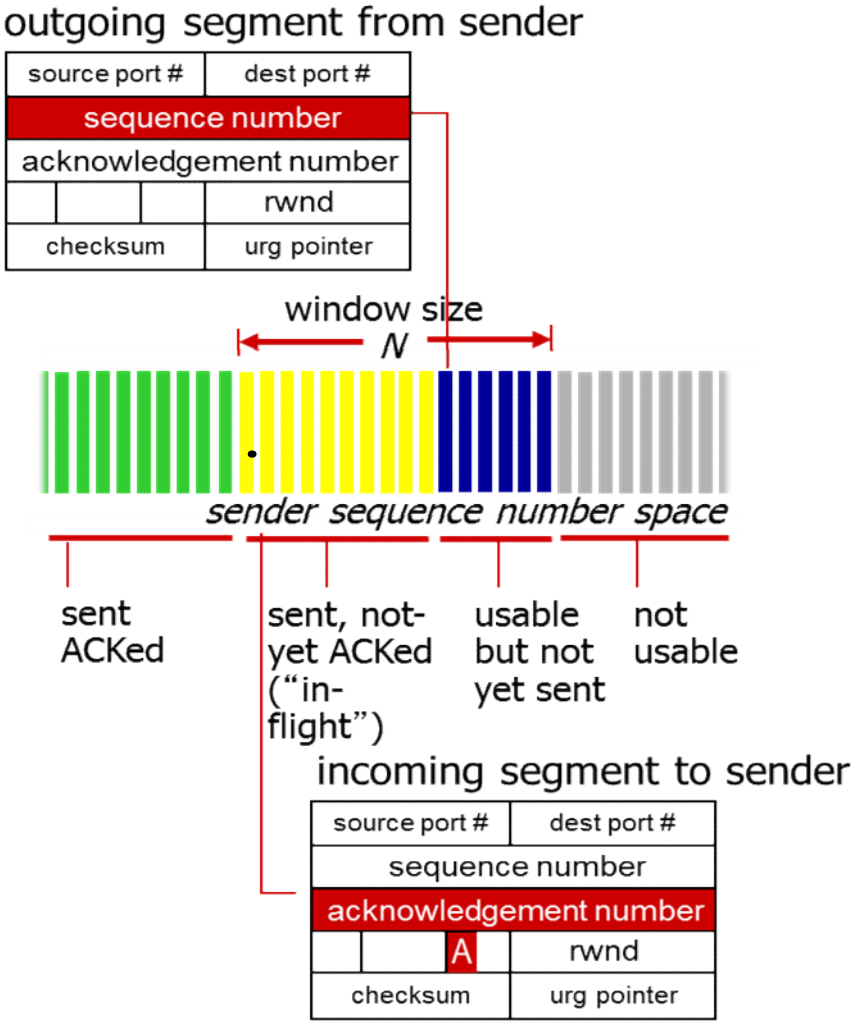
ساختار یک قطعهTCP



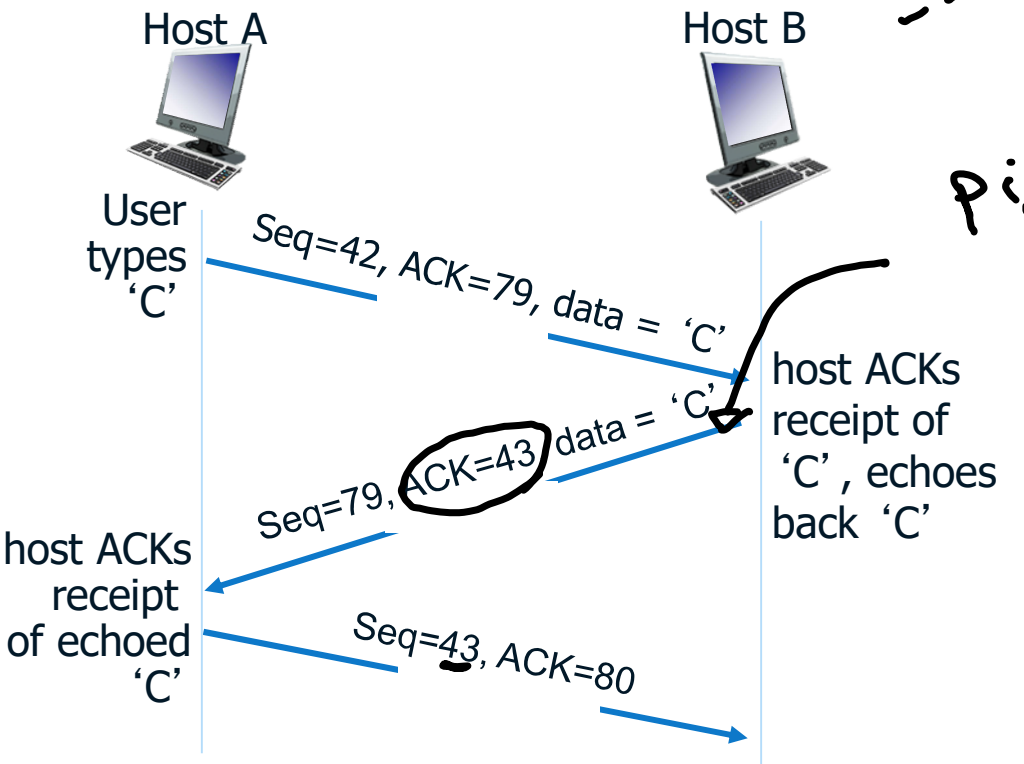
عددتوالی و عددتصدیق(ACK)درقطعاتTCP

- عدد توالی شماره اولین بایت از قطعه است.
- عددتصدیق شماره اولین بایت بعدی است که گیرنده انتظار دریافت آنرا می‌کشد.
- درTCP تصدیق بصورت تجمعی می‌باشد.




outgoing segment from sender
source port #
dest port #
sequence number
acknowledgement number
rwnd
checksum
urg pointer
window size
N
sender sequence number space
sent ACKed
sent, not-yet ACKed ("in-flight")
usable but not yet sent
not usable
incoming segment to sender
source port #
dest port #
sequence number
acknowledgement number
A
rwnd
checksum
urg pointer

استفاده از Telnet در TCP



تخمین زمان رفت و برگشت(RTT) –زمان Timeout

timeout = RTT + ε

- زمان انقضای زمانسنج(Timeout) باید RTT بزرگتر باشد.اما چقدر بزرگتر؟

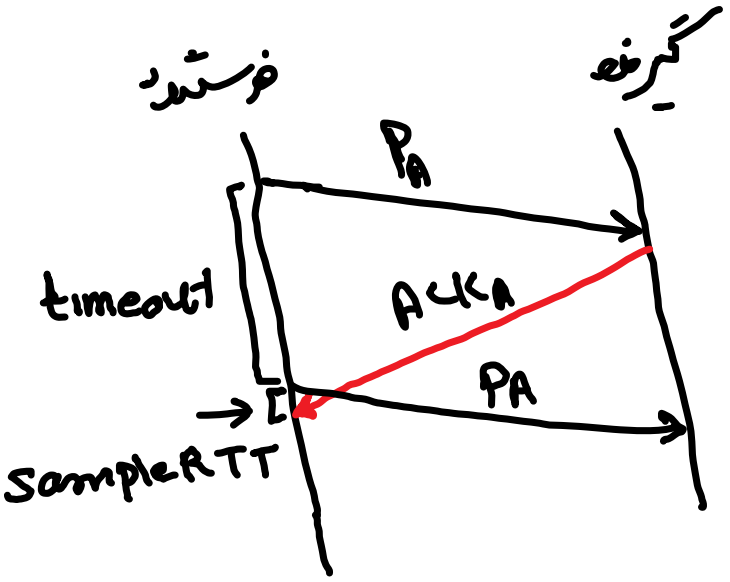
- تخمین زمان RTT:

-محاسبه SampleRTTدر هرزمان برای یک قطعه که ACKدریافت نموده است.

-هیچگاه SampleRTTبر مبنای قطعات ارسال مجدد محاسبه نمی‌شود.چرا؟

- باتوجه به یکسان نبودن وضعیت شبکه در موقعیت‌های مختلف، مقدارSampleRTT درطول زمان نوسان دارد.لذا از میانگین مقادیر اخیر استفاده می‌شود.

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


فرستنده
گیرنده
PA
timeout
ACKA
PA
SampleRTT

S(1) S(2) S(3) S(4) ....

قدیمی‌تر ⟶ جدیدتر

$0 < \alpha < 1$

$$Es(1) = \alpha S(1)$$

$$ES(2) = \alpha S(2) + (1-\alpha)\,\mathrm{Es}(1)$$

$$\mathrm{Es}(3) = \alpha S(3) + (1-\alpha)\,\mathrm{Es}(2)$$

$$\mathrm{Es}(4) = \alpha S(4) + (1-\alpha)\,\mathrm{Es}(3)$$

$$\mathrm{Es}(4) = \alpha S(4) + (1-\alpha)\Big(\alpha S(3) + (1-\alpha)\Big(\alpha S(2) + (1-\alpha)\big(\alpha S(1)\big)\Big)\Big)$$

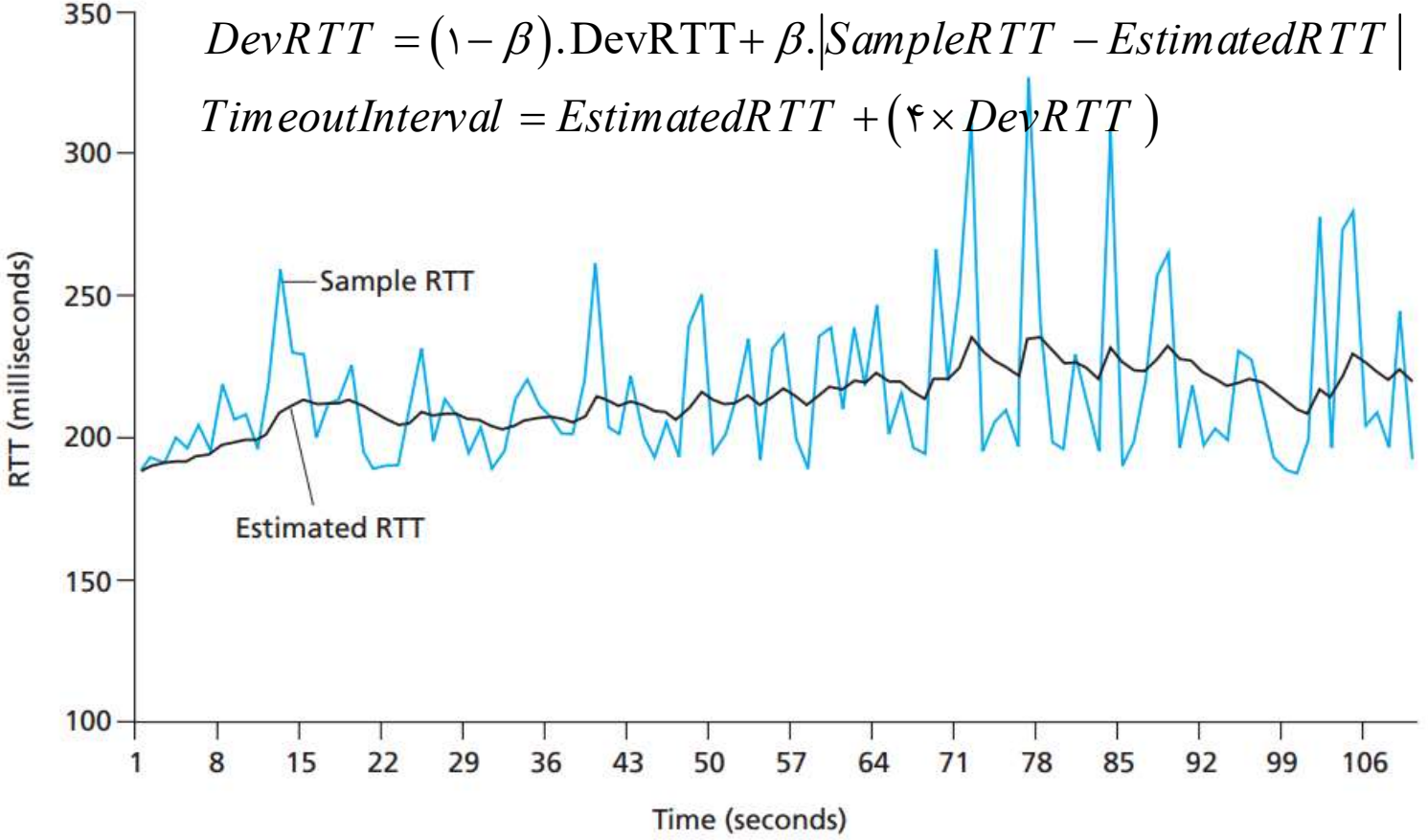
0.2 0.2 0.8

$$= \alpha S(4) + \alpha(1-\alpha) S(3) + \alpha(1-\alpha)^2 S(2) + \alpha(1-\alpha)^3 S(1)$$

$$\mathrm{Es(n)} = \alpha\left[\sum_{i=1}^{n-1}(1-\alpha)^{i-1} S(n-i+1) + (1-\alpha)^{n-1} S(1)\right]$$

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

$$TimeoutInterval = EstimatedRTT + (4 \times DevRTT)$$



انتقال داده قابل اطمینان

- TCPروی سرویس غیرقابل اطمینانIP یک سرویس انتقال داده مطمئن ایجاد می‌کند.
- سرویس انتقال دادهTCP بایتها را بدون خطا و به ترتیب و بدون تکرار به گیرنده تحویل می‌دهد.
- درTCPبرای جلوگیری از سربار اضافی برای تمام قطعات ارسال شده و هنوز تایید نشده فقط یک زمان‌سنج در نظر گرفته می‌شود.

فرستنده ساده TCP –فقط تشخیص گم‌شدن قطعه با timeout

دریافت داده از برنامه کاربردی لایه بالاتر

۱)ایجاد یک قطعه با شماره ترتیبNextSeqNum

۲)اگر زمان‌سنج غیرفعال باشد،آنرا فعال می‌کند.(زمان سنج همواره برای قدیمی‌ترین قطعه تایید نشده در نظر گرفته می‌شود.و زمان انقضای آن از طریق متغیر TimeoutInterval مشخص می‌شود

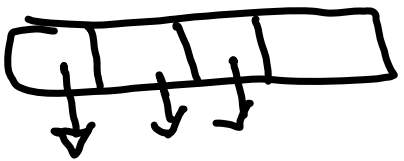
۳)قطعه به لایه شبکه تحویل داده می‌شود.

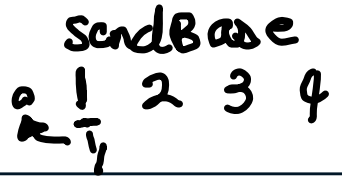
Lencdata)

۴)NextSeqNum=NextSeqNum+1

رخداد timeout

۱) قطعه با کوچکترین شماره ترتیب(تایید نشده) که باعث انقضای زمان‌سنج شده است دوباره ارسال می‌شود. و زمان‌سنج دوباره فعال می‌شود.

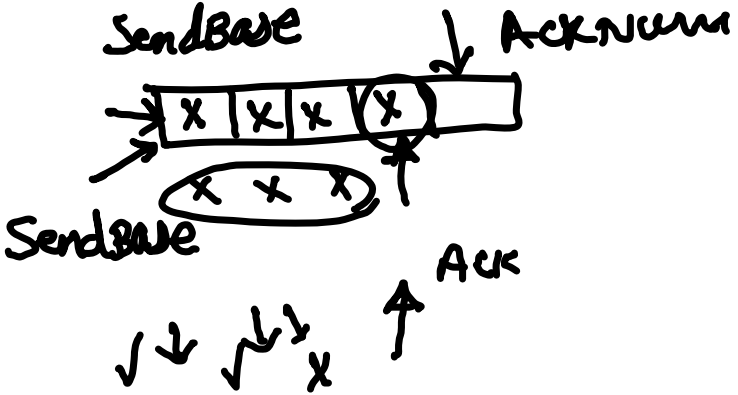


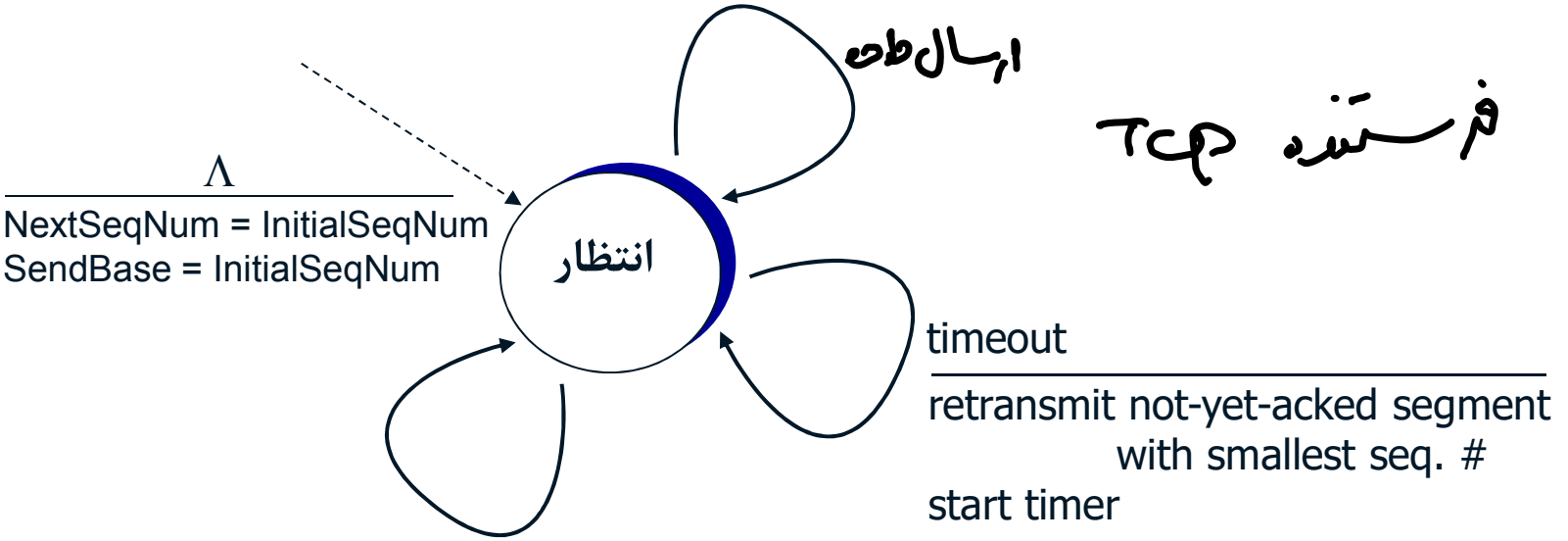


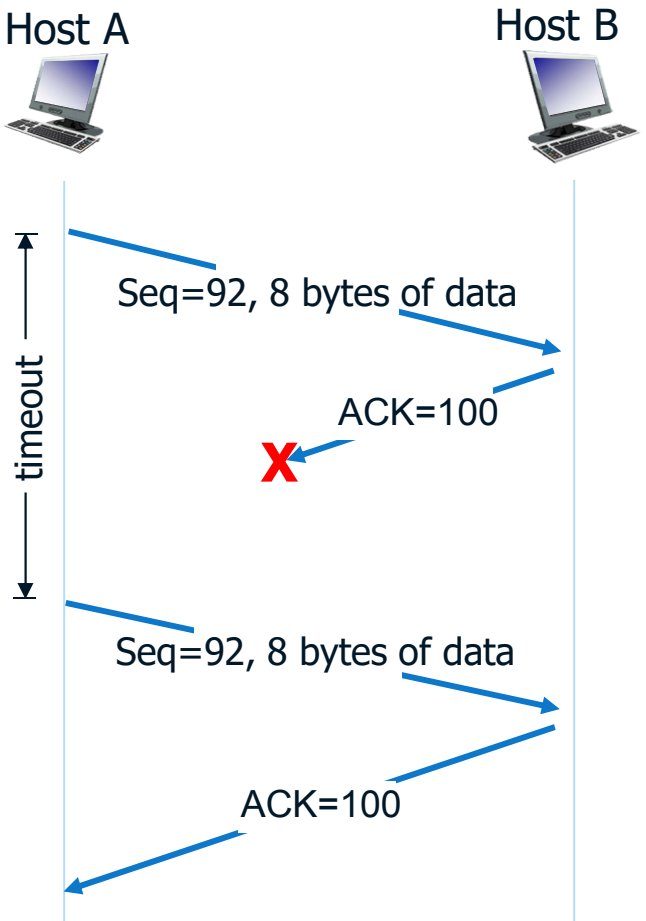
دریافت قطعه تایید(ACK) از گیرنده

۱)TCP شماره تایید(ACKNUM) را با متغیرSendBase مقایسه می‌کند.(همواره شماره ترتیب قدیمی‌ترین بایت تایید نشده در SendBaseقرار دارد).اگر ACKNUM>SendBaseباشد، آنگاه با دریافت این تایید تمام قطعات تایید نشده قبل از ACKNUMتایید می‌شوند.لذا این مقدار در SendBaseقرار می‌گیرد.

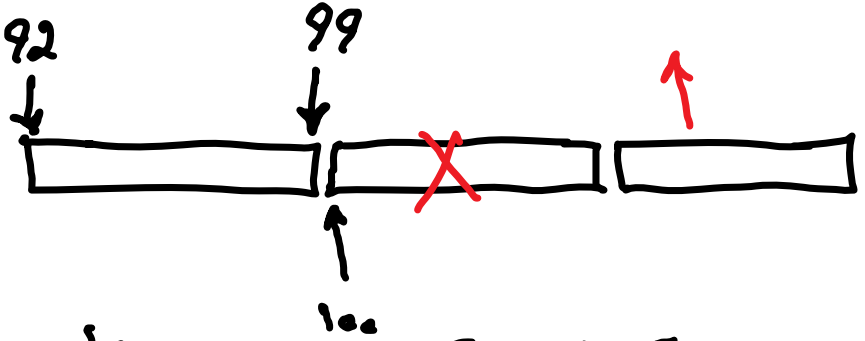
۲)اگر قطعه تاییدنشده‌ای وجود داشته باشد، آنگاه TCPزمان‌سنج را دوباره راه‌اندازی می‌کند.



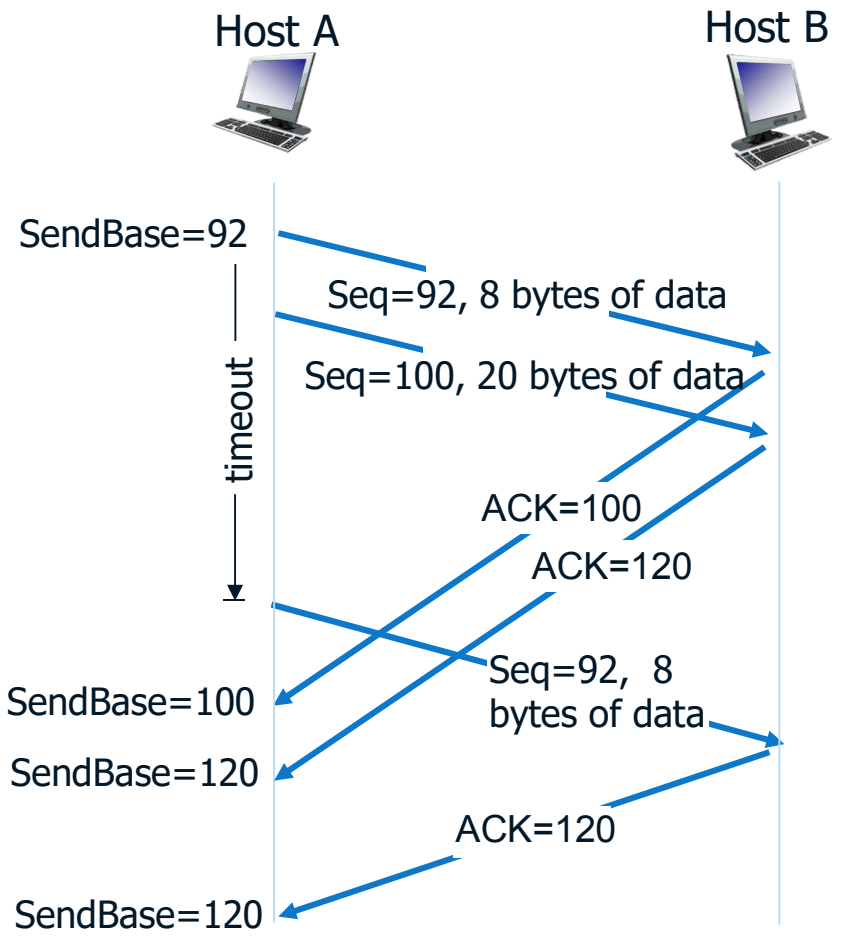

ارسال طی
فرستنده TCP
Λ
NextSeqNum = InitialSeqNum
SendBase = InitialSeqNum
انتظار
timeout
retransmit not-yet-acked segment
with smallest seq. #
start timer
ACK received, with ACK field value y
if (y > SendBase) {
SendBase = y
/* SendBase–1: last cumulatively ACKed byte */
if (there are currently not-yet-acked segments)
start timer
else stop timer
}



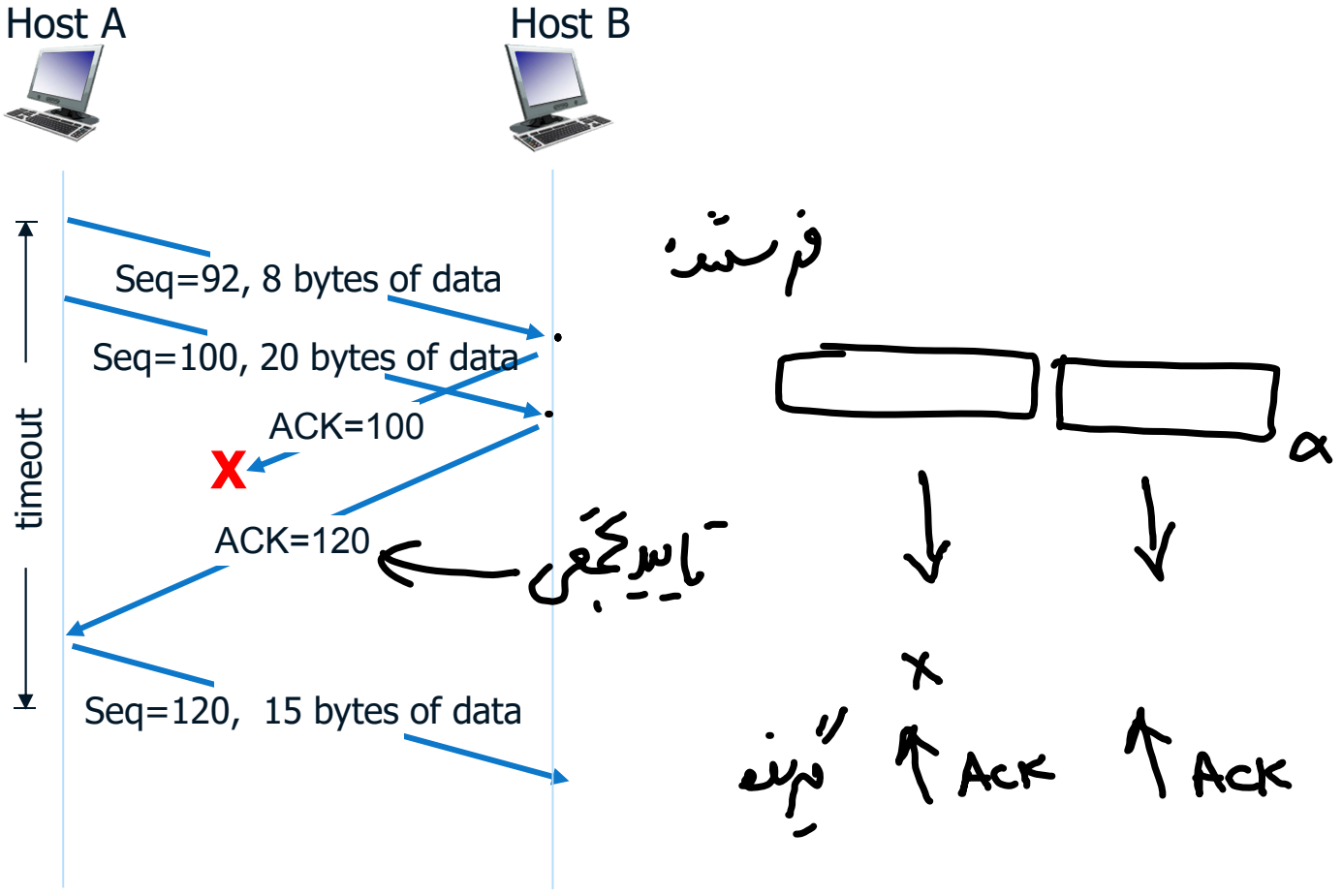
گم شدن قطعه تایید



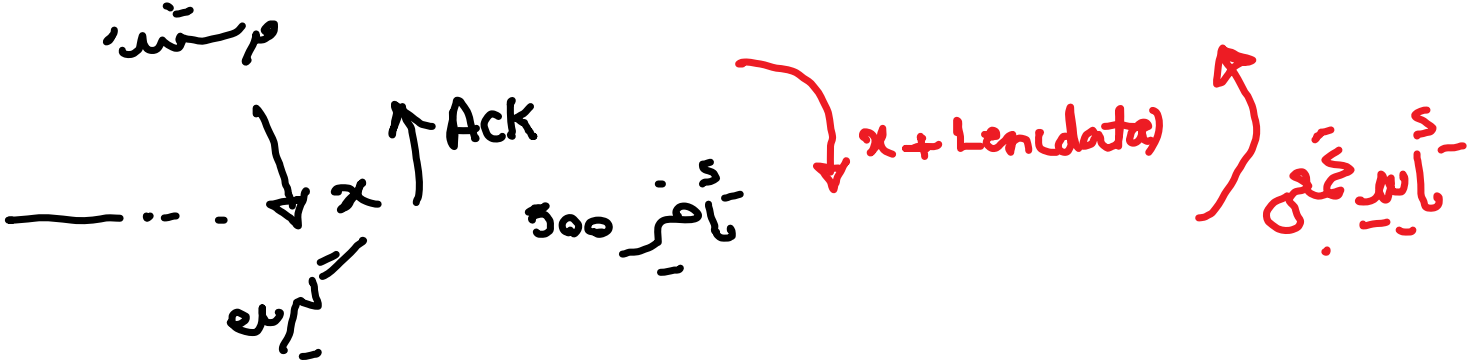
*در TCP، تأیید یک قطعه دارای شماره N
برابر اولین بایت هوردانتظار است



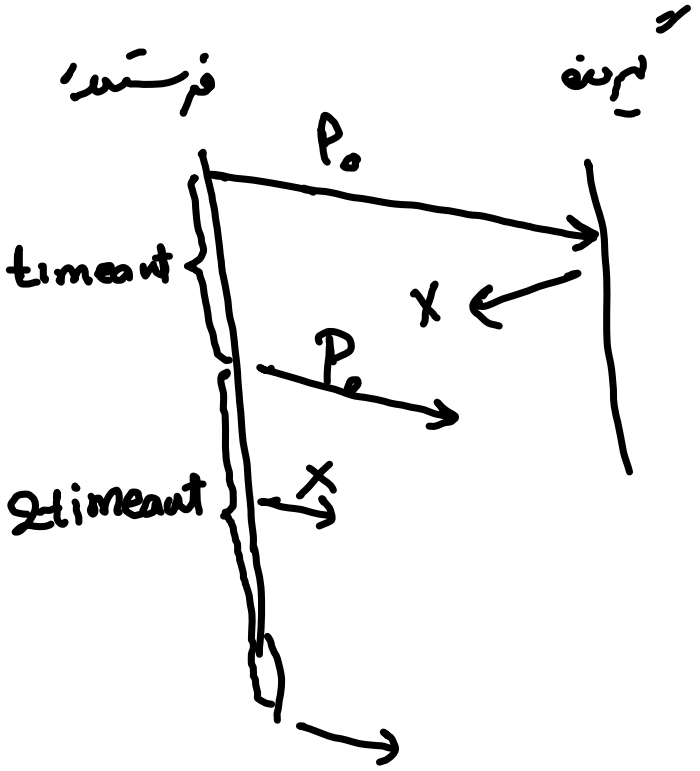
انقضای پیش‌رس زمان‌سنج


Host A
Host B
Seq=92, 8 bytes of data
Seq=100, 20 bytes of data
ACK=100
X
ACK=120
timeout
Seq=120, 15 bytes of data
فرستنده
گیرنده
ACK
ACK
تصدیق تجمعی

رویداد گیرنده	عکس العمل گیرنده
دریافت به نوبت قطعه با شماره ترتیب مورد انتظار، تمام قطعات تا این شماره قبلا تصدیق شده‌اند.	به‌تاخیرانداختن ACK، ۵۰۰ میلی ثانیه انتظار (شاید قطعه دیگری در نوبت وارد شود).
دریافت یک قطعه با شماره ترتیب موردانتظار، قطعه دیگری قبلا دریافت شده و هنوز ACK نشده است.	ارسال یک تایید تجمعی، که دریافت هردو قطعه را تایید می‌کند
دریافت خارج از نوبت قطعه با شماره ترتیب بزرگتر از عدد مورد انتظار و بروز فاصله بین قطعه ها	ارسال تایید تکراری (در واقع بایت بعد از کران پایین در فاصله بین بسته‌ها را تایید می‌کند)
دریافت یک قطعه که فاصله بین قطعه‌های دریافتی از کران پایین را پر می‌کند.	برای قطعه دریافتی ACK ارسال می‌شود.


فرستنده
x
Ack
تأخیر 500
x + Len(data)
تأیید تجمعی
گیرنده

دوبار شدن انقضای تایمر:

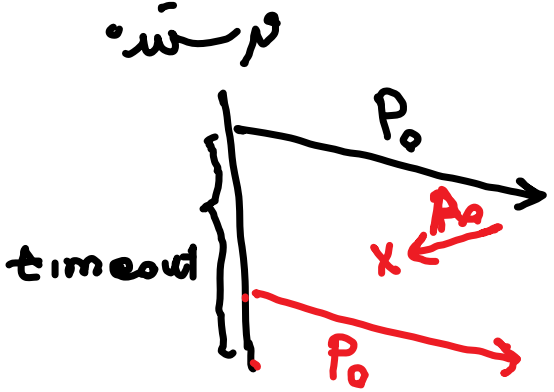


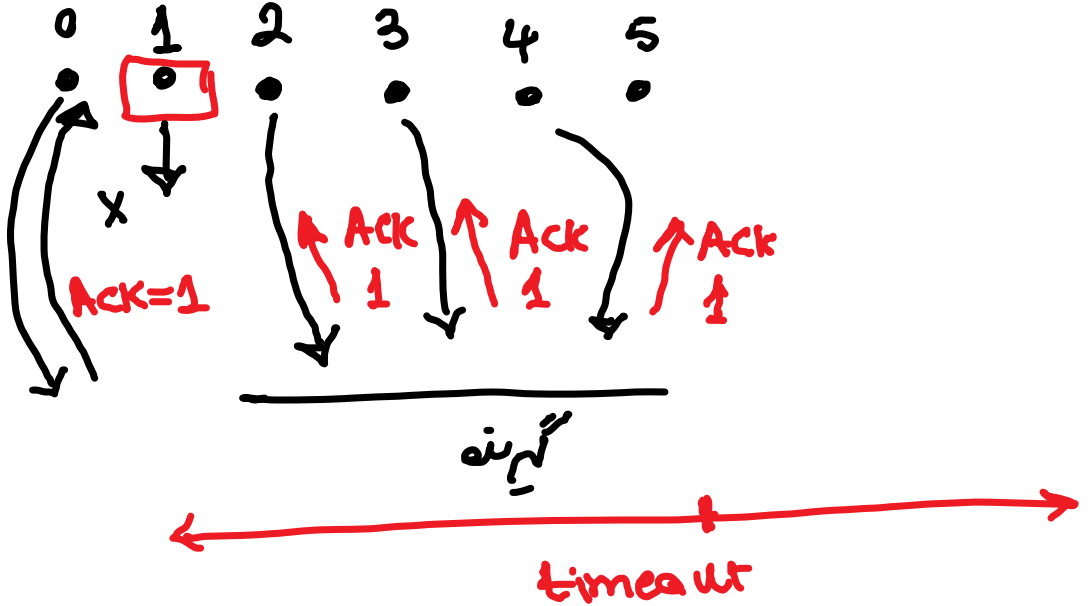
برای جلوگیری از بهتر شدن اندازه در شبکه

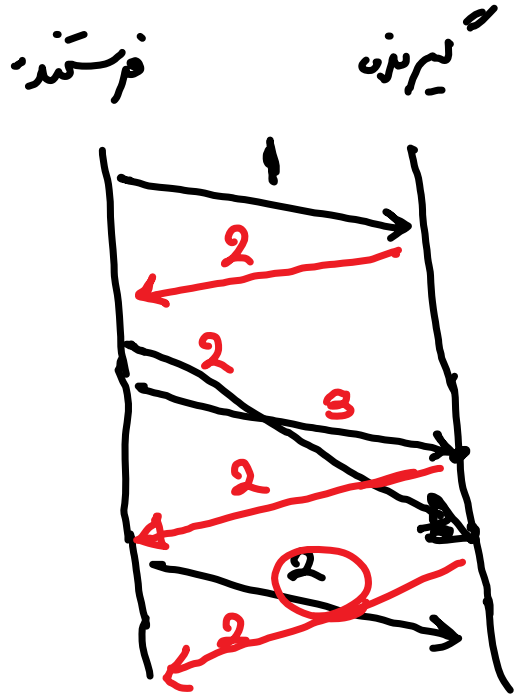
ارسال مجدد سریع

- دوره زمانی timeoutممکن است طولانی شود:
  - برای بسته‌هایی که گم می‌شوند، زمان ارسال مجدد آنها به تاخیر می‌افتد.که منجر به تاخیر انتها به انتهای زیاد می‌شود
- فرستنده با تاییدهای تکراری گم‌شدن پیش از موعد را کشف می‌کند.

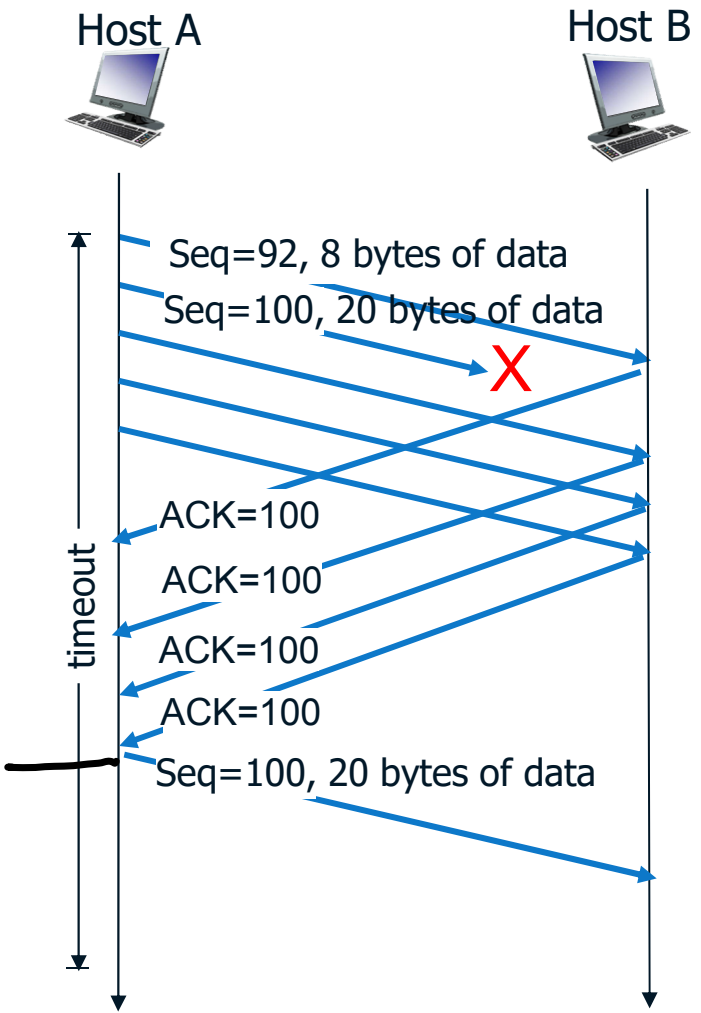
- درصورتیکه فرستنده سه ACKتکراری دریافت کند منتظر timeout نمی‌ماند و بسته گم شده را دوباره ارسال می‌کند.


فرستنده
P_0
A_0
X
timeout
P_0


0
1
2
3
4
5
X
ACK=1
Ack
1
Ack
1
Ack
1
گیرنده
timeout



1 ✓ 2 ✗ 3 ✓

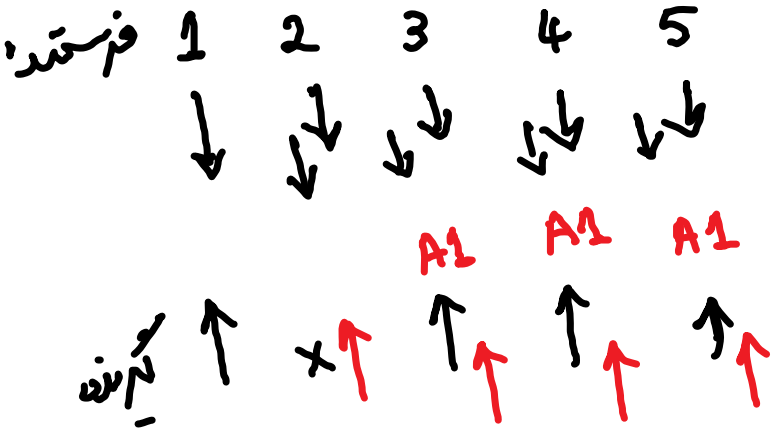

Host A
Host B
Seq=92, 8 bytes of data
Seq=100, 20 bytes of data
X
ACK=100
ACK=100
ACK=100
ACK=100
timeout
Seq=100, 20 bytes of data

مثال (۸): فرض کنید که میزبان A پنج قطعه به میزبان B ارسال می‌کند و اندازه timeout در هرسه پروتکل زیاد باشد.(گیرنده و فرستنده در این مدت می‌توانند بسته ها را ارسال و تایید کنند)،حال اگر قطعه دوم در شبکه گم شود.در پایان که میزبان B همه قطعات را بدرستی دریافت نموده است.به سوالات زیر برای پروتکل‌های SR،GBN وTCP پاسخ دهید.

الف) درمجموع، میزبانAچندقطعه داده و میزبانBچندقطعه تایید ارسال کرده‌اند.شماره ترتیب آنها چیست؟

ب)اگر زمان انقضا در هرسه پروتکل از 5RTTبزرگتر باشد.کدام پروتکل این پنج قطعه را در کوتاهترین زمان به مقصد تحویل می‌دهد؟

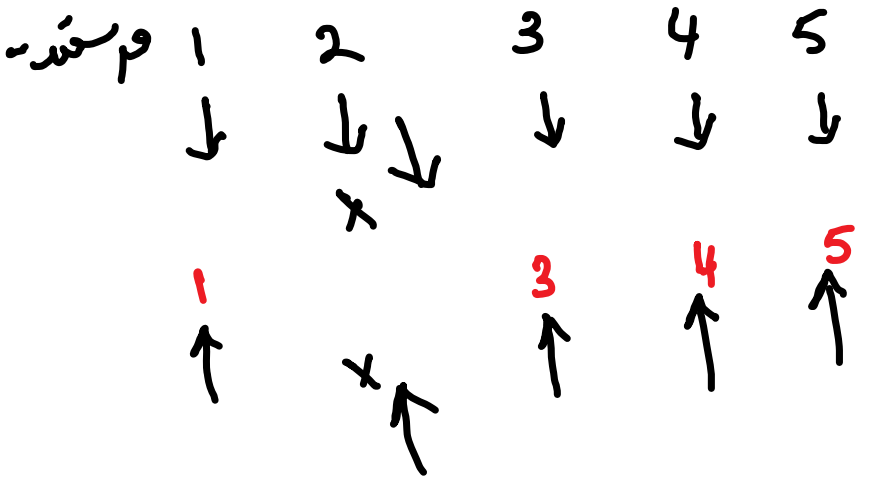
GBN:



داده ارسال = 5 + 4 = 9

Ack ارسال = 4 + 4 = 8

SR



ارسال داده = 5 + 1 = 6

Ack ارسال = 4 + 1 = 5

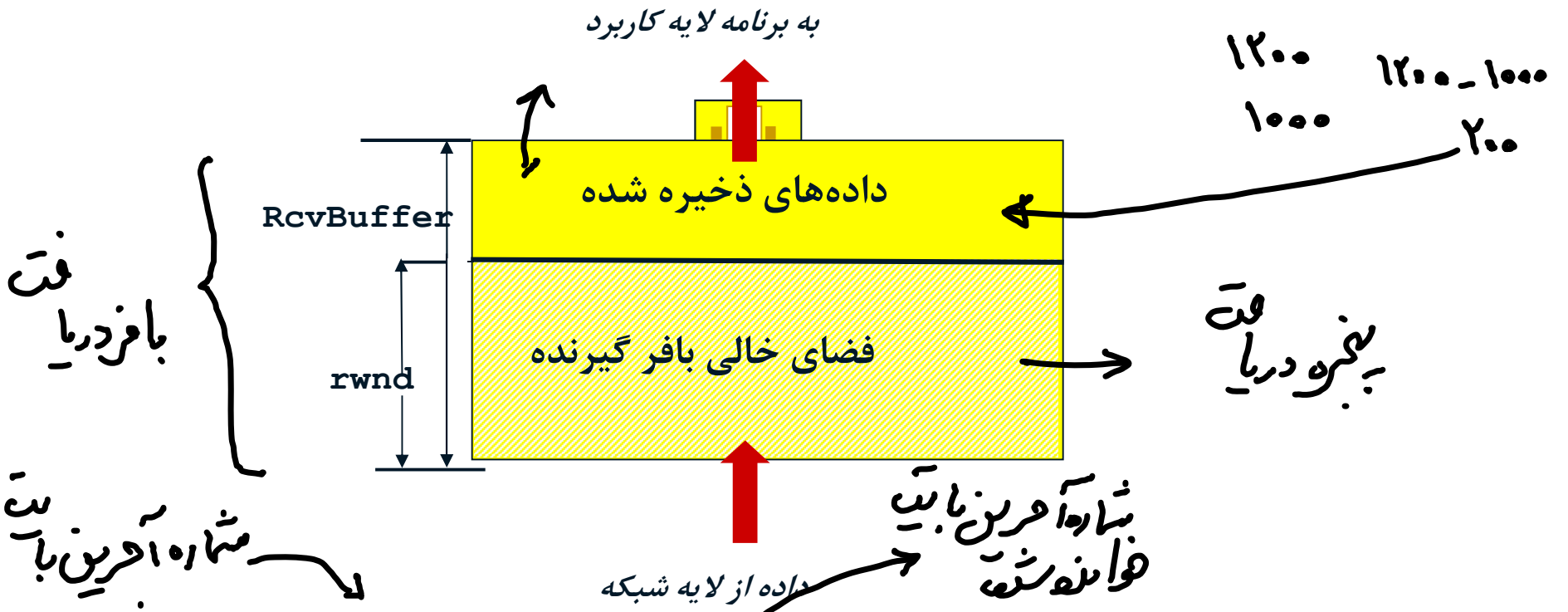
Tcp :

ارسال داده = 6

Ack ارسال = 5

کنترل جریان

- برای جلوگیری از احتمال پرشدن بافر گیرنده از مکانسیم کنترل جریان استفاده می‌شود.
- کنترل جریان نرخ ارسال در فرستنده را با نرخ دریافت گیرنده از بافر تنظیم می‌کند.
- برای مکانسیم کنترل جریان، TCP فرستنده را مجبور می‌کند تا متغیری تحت عنوان پنجره دریافت (rwnd) نزد خود نگه دارد.



$$LastByteRcvd - LastByte\,\mathrm{Read} \le RcvBuffer$$

$$rwnd = RcvBuffer - [LastByteRcvd - LastByte\,\mathrm{Read}]$$

$$LastByteSent - LastByteAcked \le rwnd$$

مدیریت اتصال TCP

client state

LISTEN

SYNSENT

ESTAB

choose init seq num, x
send TCP SYN msg

SYNbit=1, Seq=x

server state

LISTEN

choose init seq num, y
send TCP SYNACK
msg, acking SYN

SYN RCVD

SYNbit=1, Seq=y
ACKbit=1; ACKnum=x+1

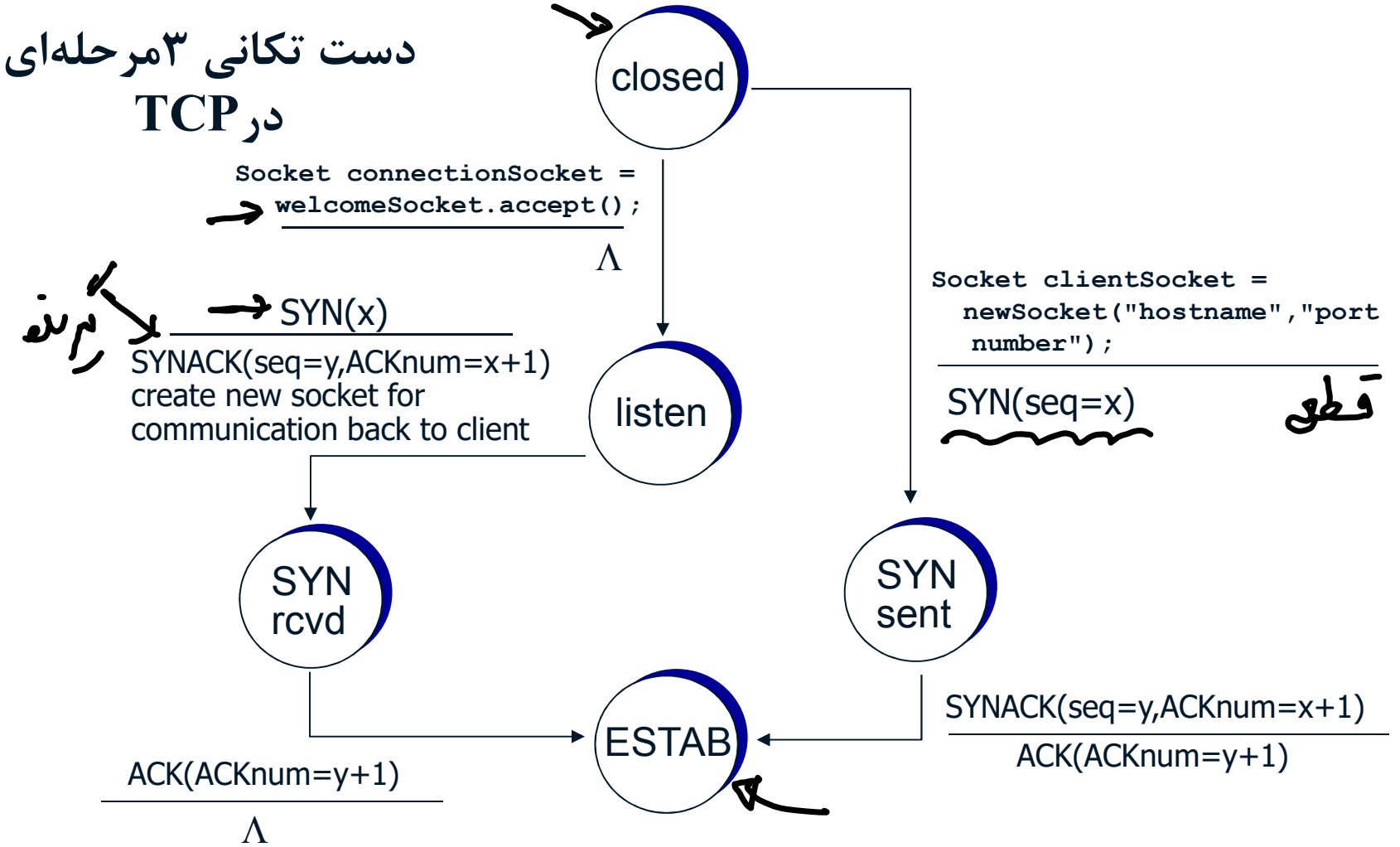
received SYNACK(x)
indicates server is live;
send ACK for SYNACK;
this segment may contain
client-to-server data

ACKbit=1, ACKnum=y+1

received ACK(y)
indicates client is live

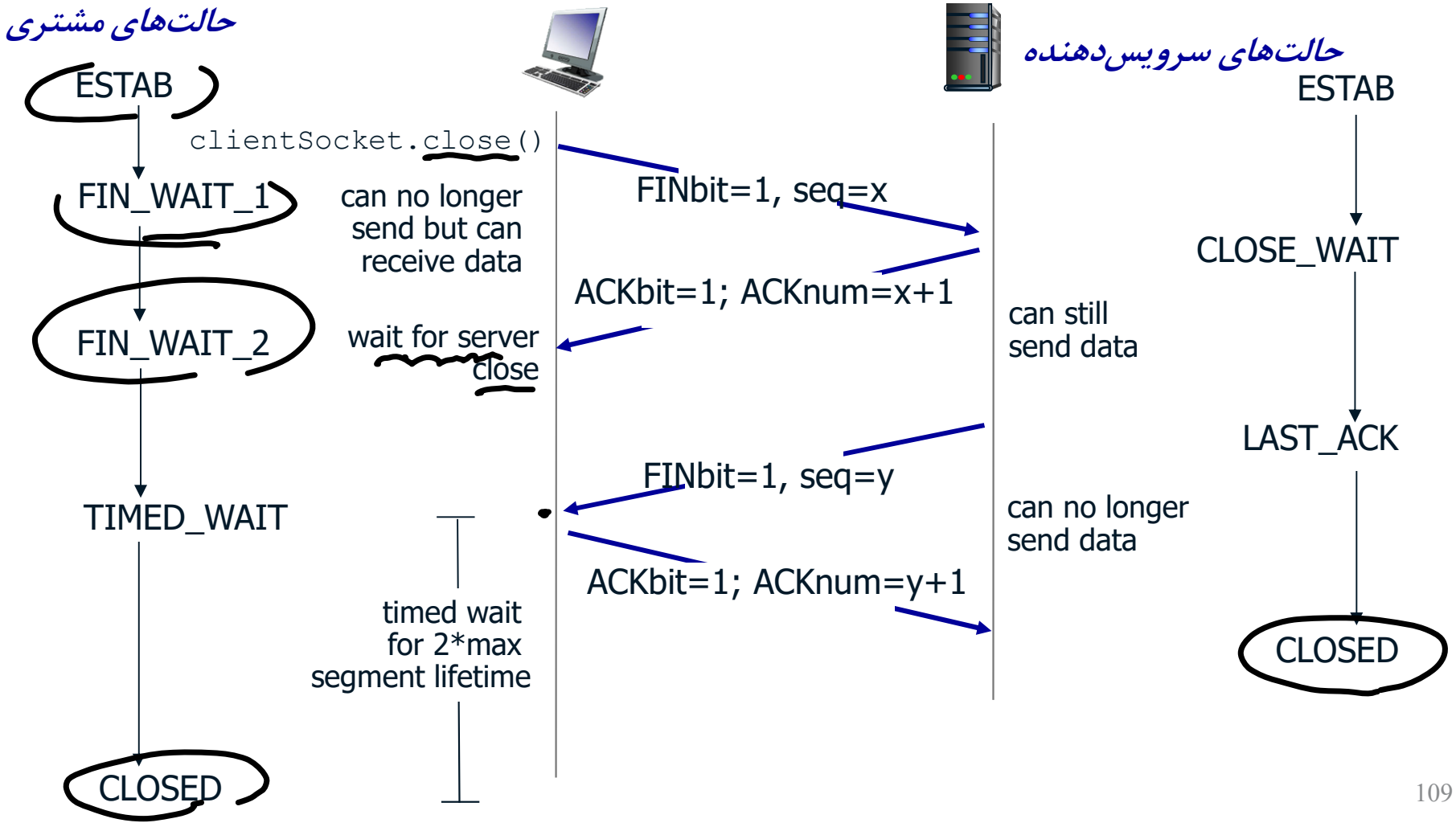
ESTAB

دست تکانی ۳مرحله‌ای در TCP



قطع ارتباط TCP

- هرکدام از سرویس‌دهنده یا مشتری می‌توانند ارتباط TCPایجاد شده را قطع کنند.
- مشتری یک قطعه FIN، (قطعه‌ای که بیت FINدر سرایند آن یک است) ارسال می‌کند.و منتظر دریافت تایید از سمت سرویس‌دهنده می‌ماند.با دریافت ACKارتباط از طرف مشتری پایان یافته است.
- سرویس‌دهنده نیز همانند مشتری با ارسال یک قطعه FIN و دریافت ACK از مشتری ارتباط خود را قطع می‌کند.


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

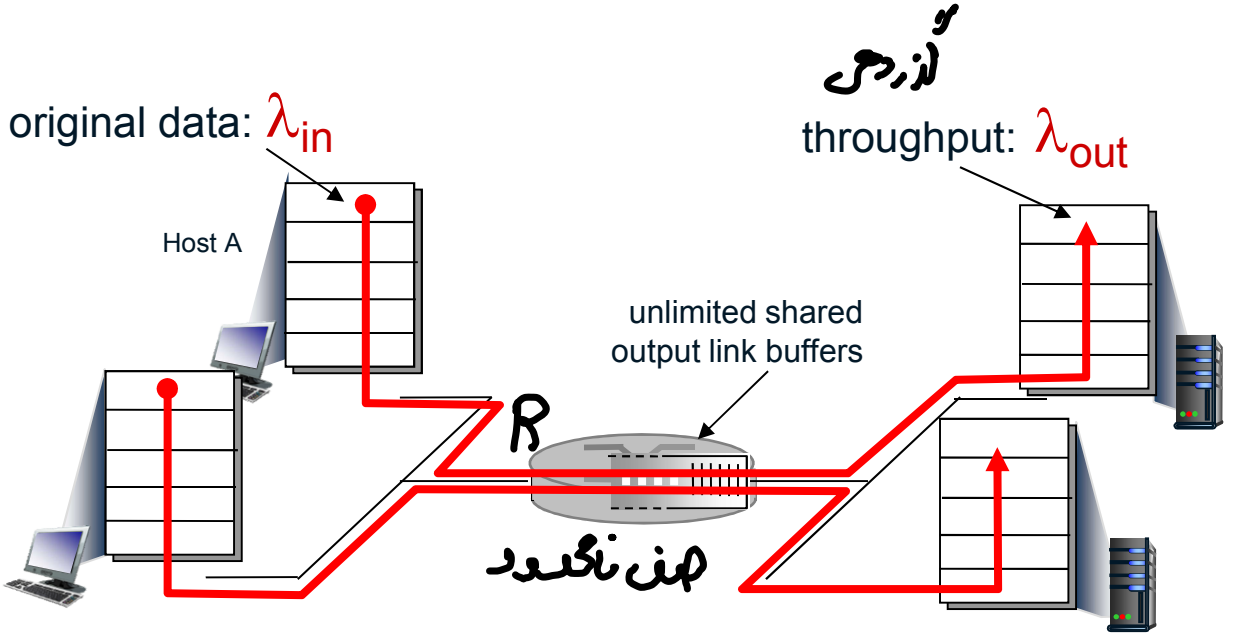
رئوس مطالب:

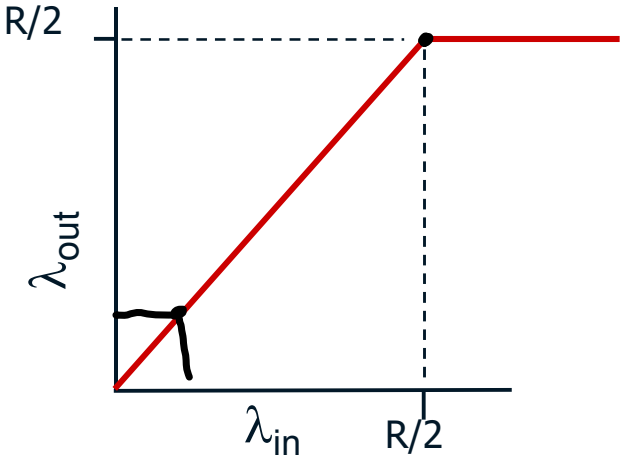
- ❑ آشنایی با لایه انتقال و سرویس‌های آن
- ❑ مالتی پلکسینگ و دی مالتی پلکسینگ
- ❑ انتقال نامطمئن: UDP
- ❑ اصول انتقال داده قابل اطمینان
- ❑ انتقال داده اتصال‌گرا: TCP
- ✅ اصول کنترل ازدحام
- ❑ کنترل ازدحام در TCP

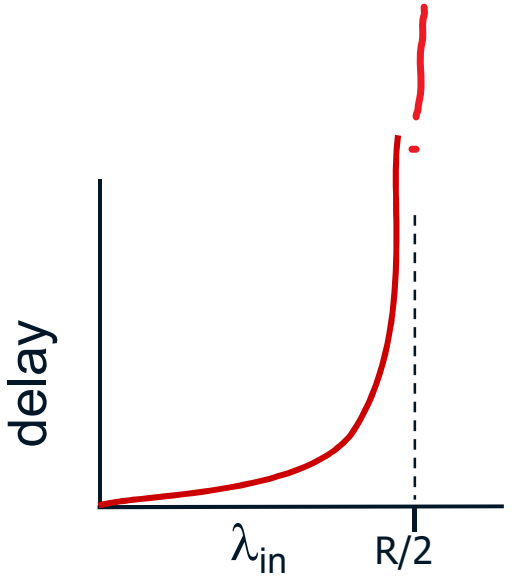
اصول کنترل ازدحام

- ازدحام در شبکه باعث سریز شدن بافر مسیریاب‌ها در شبکه شده، و تلفات بسته بالا می‌رود.
- تلفات بسته که محصول ازدحام است،باعث استفاده بیشتر از سازوکار ارسال مجدد خواهدشد، که کارایی را پایین می‌آورد.

بررسی کنترل ازدحام: سناریو۱

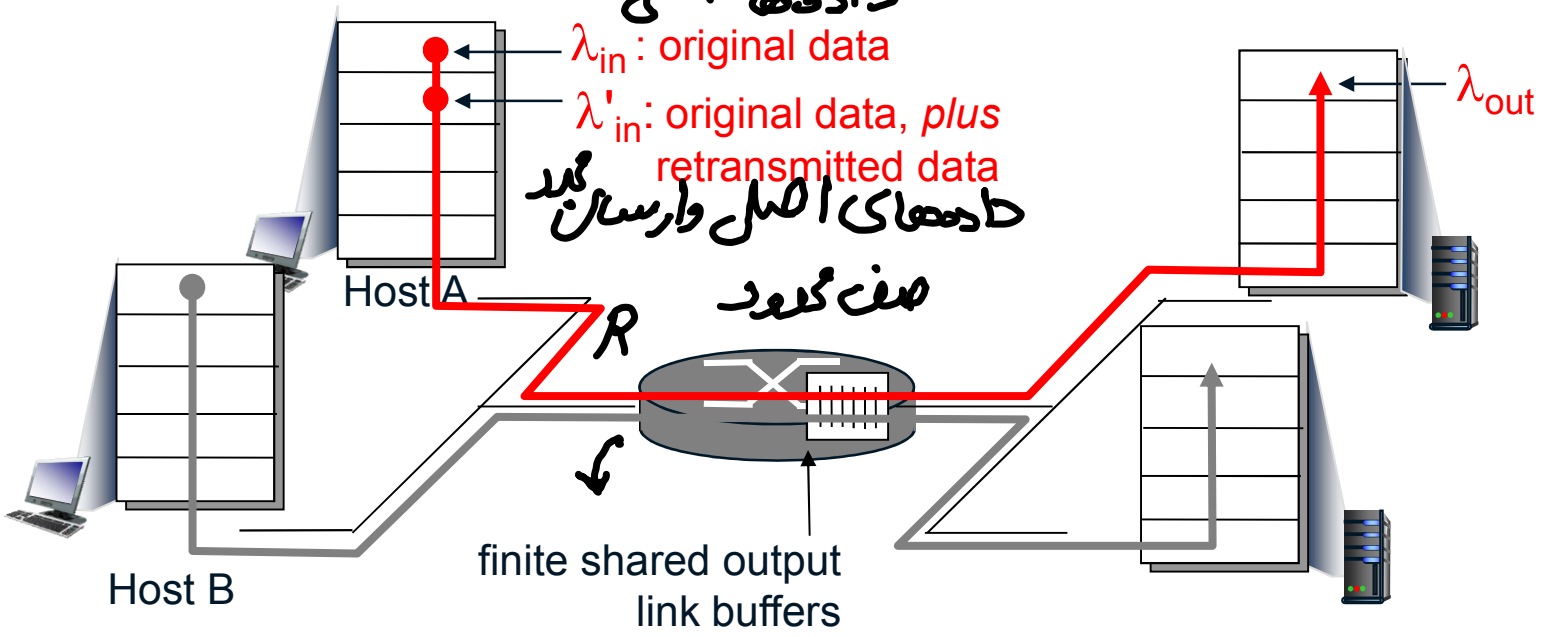






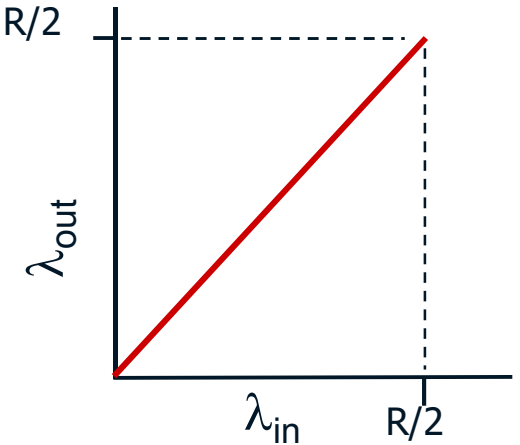
$\lambda_{in} \rightarrow \frac{R}{2} \Rightarrow \text{delay} \rightarrow \infty$

بررسی کنترل ازدحام: سناریو۲

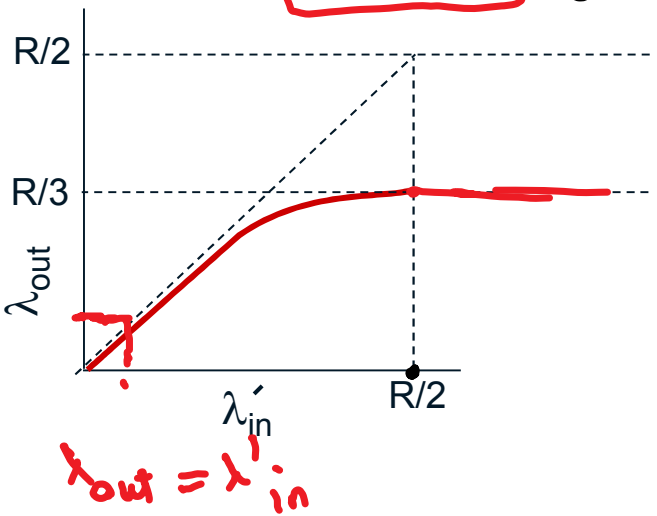


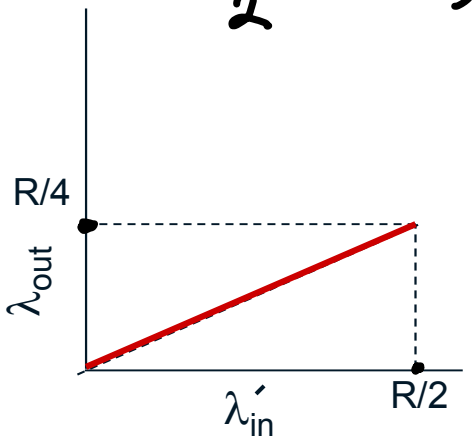
λ'_{in} : offered load بازفرستاده شده

❖ فرض ایده آل: فرستنده از وضعیت بافرمطلع است، فقط هنگامیکه بافر فضای خالی داشته باشد، فرستنده ارسال انجام می‌دهد.

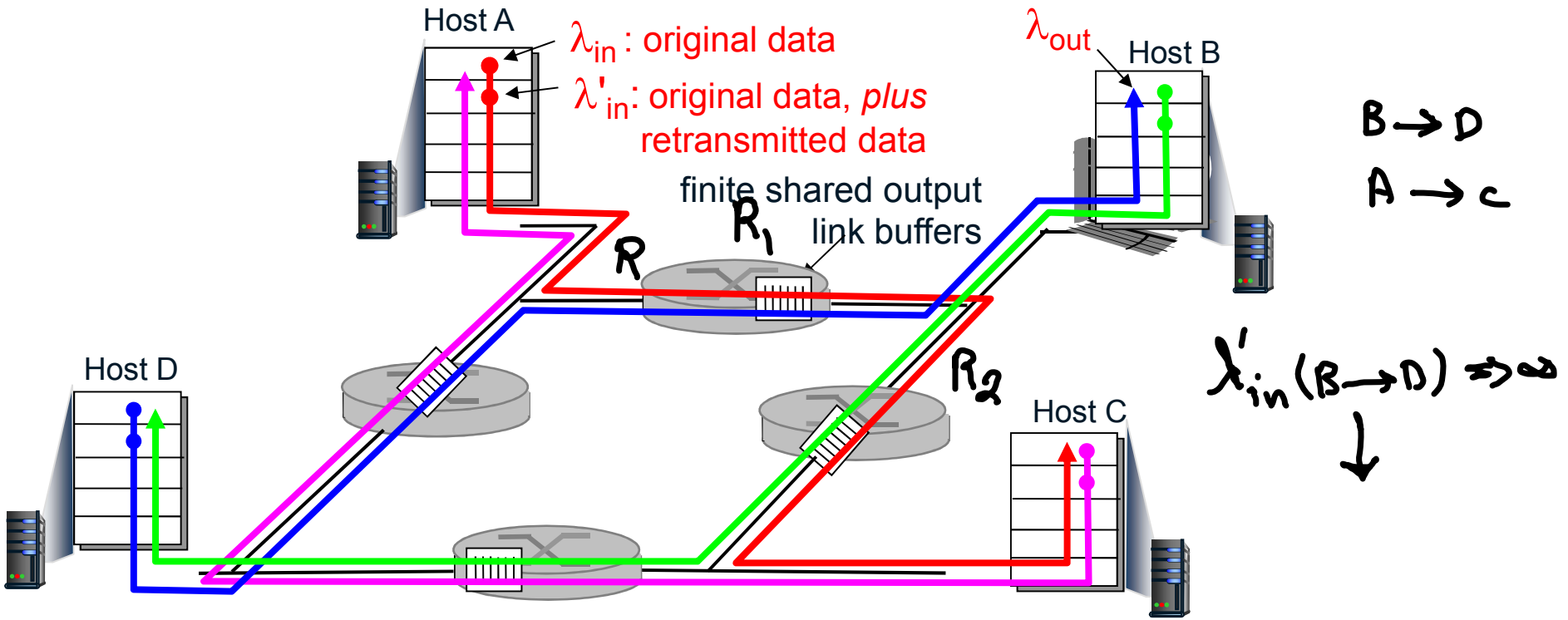


❖ فرستنده بسته‌هایی را دوباره ارسال می‌کند، که از گم شدن آن بطور یقین مطمئن است.

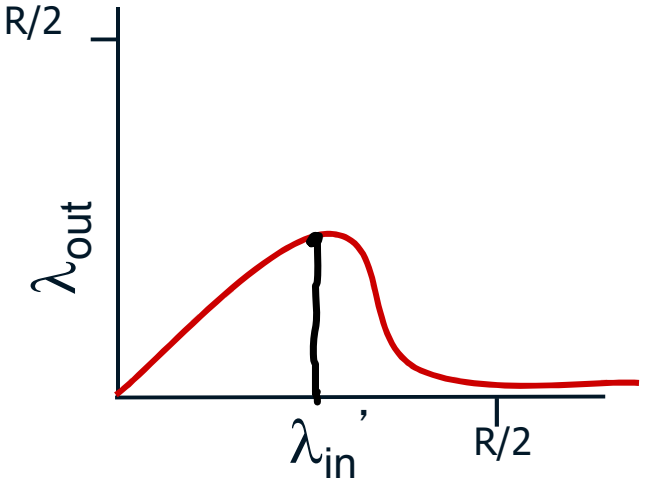




بررسی کنترل ازدحام: سناریو۳



❖ در سناریوی تعریف شده، هنگامیکه یک بسته به علت ازدحام در یک گام بالاتر گم می‌شود، کار تمامی مسریاب ها در گام‌های پایینتر ضایع خواهد شد.و مثل این است که آن مسیریاب‌ها هیچ کاری در جهت انتقال بسته انجام نداده‌اند.



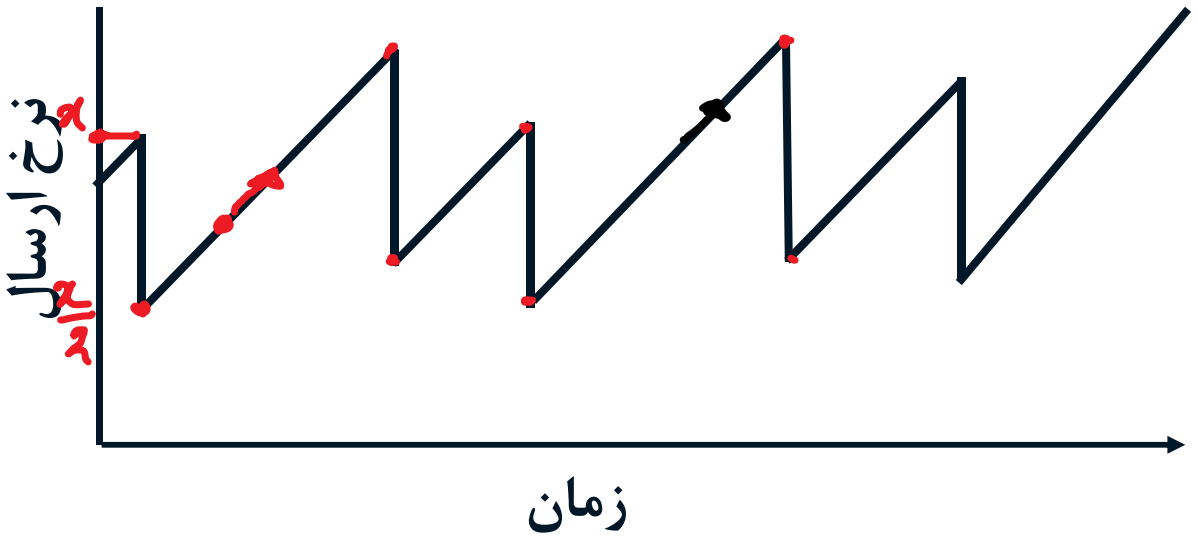
رئوس مطالب:

- ❑ آشنایی با لایه انتقال و سرویس‌های آن
- ❑ مالتی پلکسینگ و دی مالتی پلکسینگ
- ❑ انتقال نامطمئن: UDP
- ❑ اصول انتقال داده قابل اطمینان
- ❑ انتقال داده اتصال‌گرا: TCP
- ❑ اصول کنترل ازدحام

✅ کنترل ازدحام در TCP

کنترل ازدحام در TCP

- درکنترل ازدحام TCPهر فرستنده بر اساس اطلاعاتی که از ازدحام شبکه دارد، نرخ ارسال خود را تنظیم می‌کند.
- درصورتیکه در شبکه ازدحام وجود نداشته باشد، فرستنده نرخ ارسال خود را در طول زمان افزایش می‌دهد.
- درصورتیکه ازدحام در شبکه بوجود بیاید، و بسته‌ها گم شوند، فرستنده نرخ ارسال خود را کاهش می‌دهد.

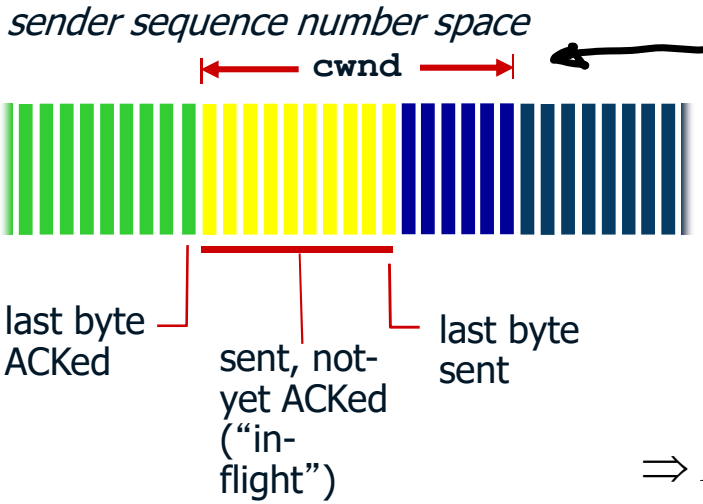

نرخ ارسال
زمان

کنترل ازدحام در TCP....

- برای کنترل ازدحام در TCPباید به این سه سوال پاسخ دهیم؟

۱-فرستنده چگونه نرخ ارسال خود را کنترل می‌کند؟
۲-فرستنده چگونه وجود ازدحام در مسیر خود تا گیرنده را کشف می‌کند؟
۳-با توجه به شدت ازدحام انتها به انتها فرستنده از چه الگوریتمی برای تغییر نرخ ارسال استفاده می‌کند؟

۱-فرستنده چگونه نرخ ارسال خود را کنترل می‌کند؟



Congestion window

۵

$$LastByteSent - LastByteAcked \le rwnd$$

$$LastByteSent - LastByteAcked \le cwnd$$

۴

$$\Rightarrow LastByteSent - LastByteAcked \le \min(rwnd, cwnd)$$

❖ در بهترین حالت تقریبا: در هر **RTT** یک پنجره ازدحام(**cwnd**) ارسال و تایید می‌شود، پس نرخ ارسال در فرستنده **cwnd/RTT** است.

۲-فرستنده چگونه وجود ازدحام در مسیر بین خود تا گیرنده را کشف می‌کند؟

- رخداد timeout یا دریافت سه ACKتکراری نشان دهنده گم شدن بسته می‌باشد.
- گم شدن بسته‌ها نشانه‌ای از سرریزشدن بافر در مسیریاب‌ها است؛ که در واقع نشان از ازدحام در شبکه است.
- درصورتیکه در شبکه تلفات بسته وجود نداشته باشد، و تایید بسته‌های ارسالی با موفقیت به گیرنده برسند آنگاه نوید عدم وجود ازدحام در شبکه است.
- سرعت افزایش اندازه پنجره ازدحام(Cwnd) تابع نرخ دریافت بسته‌های تایید خواهد بود.
- در واقع دریافت تاییدها از گیرنده بعنوان یک عامل محرک برای افزایش پنجره ازدحام می‌باشد.

– فرستنده چگونه می‌تواند ضمن اجتناب از ازدحام در شبکه، از حداکثر پهنای باند در دسترس استفاده کند؟

- ❖ گم شدن بسته خبر از ازدحام در شبکه می‌دهد.لذا بعد از اطلاع از گم شدن بسته فرستنده نرخ ارسال را کاهش می‌دهد.
- ❖ دریافت تایید برای یک قطعه نشان‌دهنده عادی بودن اوضاع در شبکه است.در این حالت فرستنده برای استفاده بیشتر از پهنای باند نرخ ارسال خود را افزایش می‌دهد.
- ❖ لذا فرستنده TCP بطور صریح وجود یا عدم وجود و درجه ازدحام را کشف نمی‌کند، با یک رفتار افزایشی تا تا آستانه ازدحام پیش‌رفته و در صورت رخداد ازدحام و کشف ان از طریق تلفات بسته عقب نشینی کرده و نرخ ارسال را کاهش می‌دهد.اما بازهم رفتار افزایشی را تکرار می‌کند.

الگوریتم کنترل ازدحام در TCP :

1- شروع آهسته

2- اجتناب از ازدحام

3 - بازیابی سریع

حالت شروع آهسته

- وقتی‌که یک اتصال TCP شروع می‌شود، اندازه پنجره ازدحام(Cwnd) برابر یک MSS خواهد بود.
- به ازای هر تایید دریافت شده اندازه پنجره یک واحدMSS افزایش می‌یابد.
- در هر RTT ، اندازه پنجره در TCP دوبرابر می‌شود.

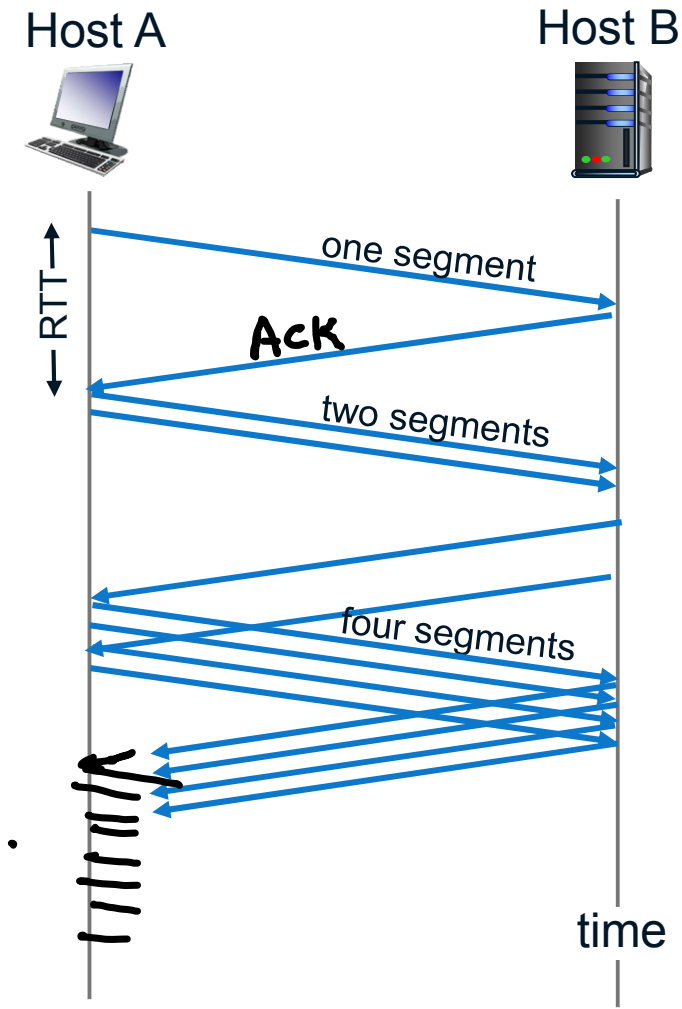
$MSS = 500\ byte$

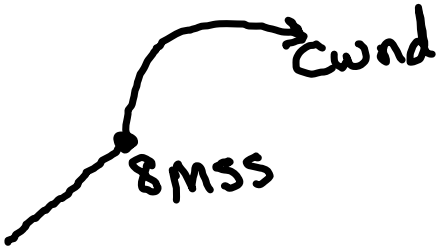
$RTT = 200\ ms$

$RTT_n \longrightarrow cwnd = 2^{n-1} MSS$

$\frac{200 \times 10^{-3}}{1} \quad \frac{500}{x} \longrightarrow x = \frac{500 \times 8}{2} = 20 \times 10^{3}\ s$

$= 20\ Kbps$


Host A
Host B
RTT
one segment
Ack
two segments
four segments
time



حالت شروع آهسته...

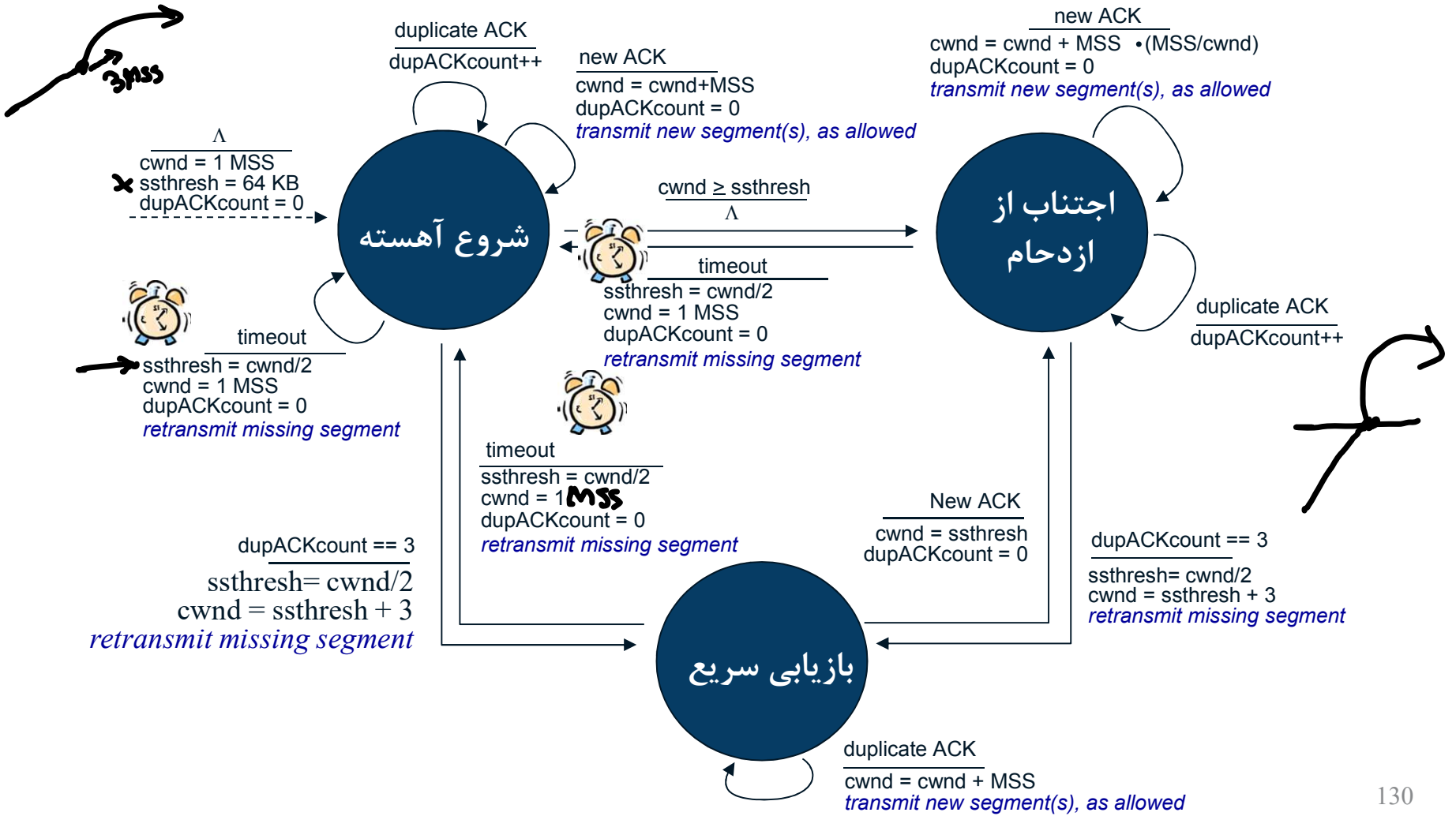
❖ درحالت شروع آهسته، اگر در اثر timeout ، گم شدن بسته کشف شود:

-مقدار ssthresh=Cwnd/2 و مقدار Cwnd=1MSS.

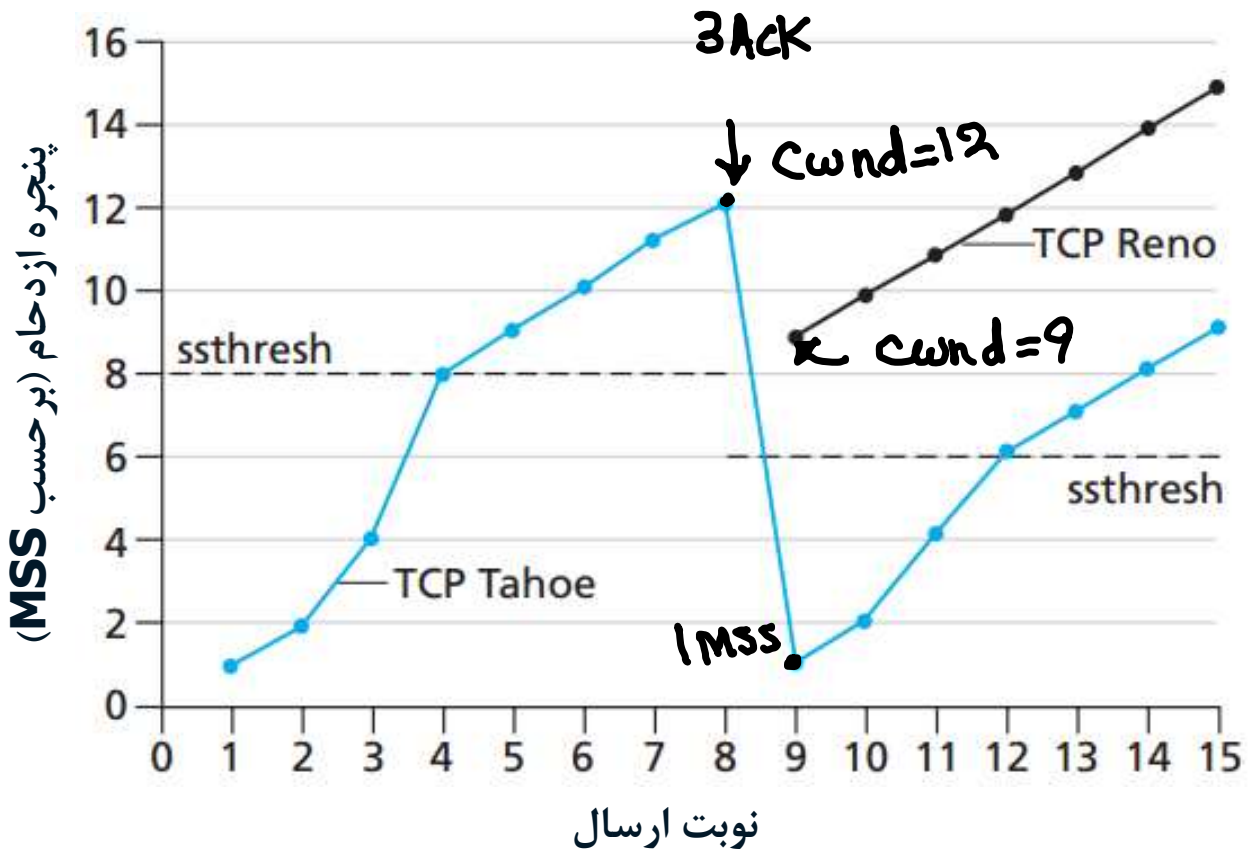
* وقتیکه TCP، دوباره به میزان آستانه (ssthresh) برسد دیگر محتاط عمل کرده و مقدار پنجره را دوبرابر نمی‌کند.و وارد حالت اجتناب از ازدحام می‌رود.

❖ درحالت شروع آهسته، اگر دراثر دریافت سه ACK تکراری، گم شدن بسته کشف شود:

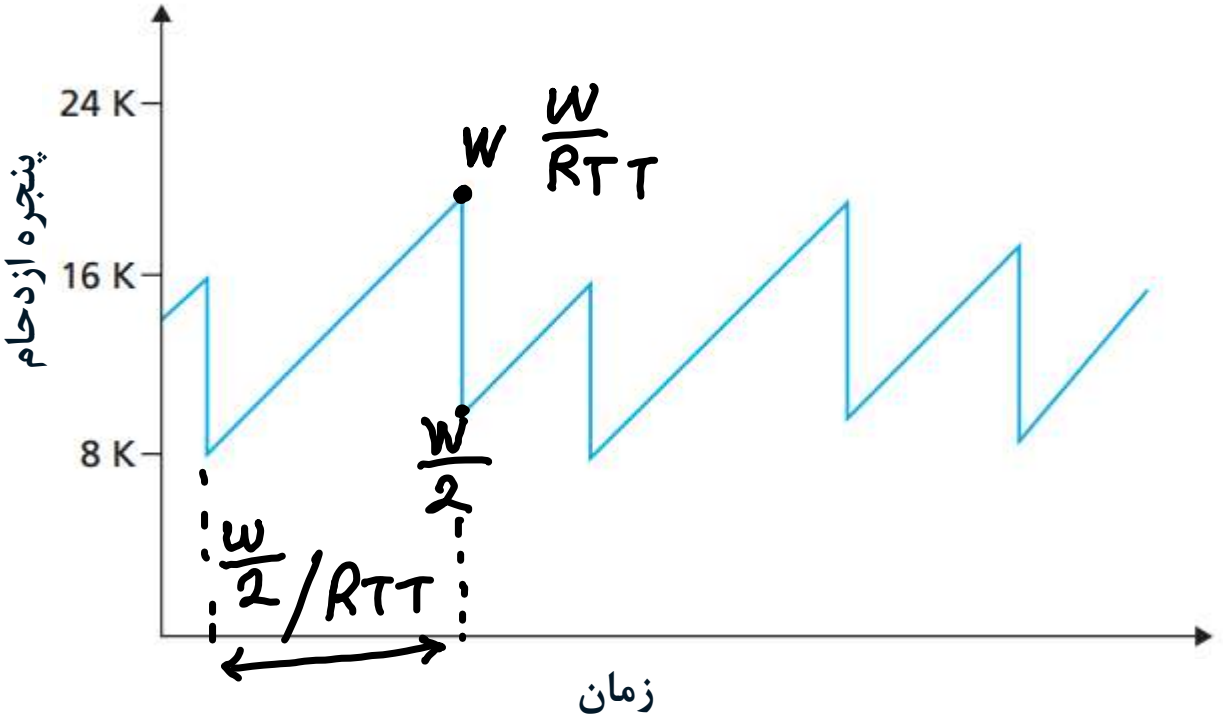
- مقدار ssthresh=Cwnd/2، و مقدار Cwndجدید برابرssthresh+3MSS . و وارد حالت بازیابی سریع می‌شود.


duplicate ACK
dupACKcount++
new ACK
cwnd = cwnd+MSS
dupACKcount = 0
transmit new segment(s), as allowed
new ACK
cwnd = cwnd + MSS •(MSS/cwnd)
dupACKcount = 0
transmit new segment(s), as allowed
Λ
cwnd = 1 MSS
ssthresh = 64 KB
dupACKcount = 0
شروع آهسته
cwnd ≥ ssthresh
Λ
timeout
ssthresh = cwnd/2
cwnd = 1 MSS
dupACKcount = 0
retransmit missing segment
اجتناب از ازدحام
duplicate ACK
dupACKcount++
timeout
ssthresh = cwnd/2
cwnd = 1 MSS
dupACKcount = 0
retransmit missing segment
timeout
ssthresh = cwnd/2
cwnd = 1 MSS
dupACKcount = 0
retransmit missing segment
New ACK
cwnd = ssthresh
dupACKcount = 0
dupACKcount == 3
ssthresh= cwnd/2
cwnd = ssthresh + 3
retransmit missing segment
dupACKcount == 3
ssthresh= cwnd/2
cwnd = ssthresh + 3
retransmit missing segment
بازیابی سریع
duplicate ACK
cwnd = cwnd + MSS
transmit new segment(s), as allowed

ssthresh = 8/6


3ACK
cwnd=12
TCP Reno
cwnd=9
ssthresh
ssthresh
TCP Tahoe
1MSS
پنجره ازدحام (برحسب MSS)
نوبت ارسال
0
2
4
6
8
10
12
14
16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

گذردهی در TCP



✗ شروع آهسته

افزایش خطی (جمعی)

کاهش ضرب

(AIMD)

❖ اندازه پنجره در زمان گم شدن بسته:W ، آنگاه نرخ ارسال از W/2RTT تا W/RTT تغییر خواهد کرد.

نرخ میانگین در اتصال TCP $= \frac{w}{RTT} + \frac{w/2}{RTT} = \frac{w}{RTT} + \frac{w}{2RTT}$

$= \frac{3w}{2RTT} / 2 = \frac{3}{4} \frac{w}{RTT}$

درصد تلف بسته $= \frac{\text{تعداد بسته‌های گم‌شده}}{\text{کل بسته‌های ارسال‌شده}}$

کل بسته‌های ارسال شده $= \frac{w}{2} + (\frac{w}{2}+1) + (\frac{w}{2}+2) + \cdots + w$

$= \frac{(w/2+1)}{2}\left(2(\frac{w}{2}) + \frac{w}{2}\right) = \frac{3}{8}w^2 + \frac{3}{4}w$

مجموع دنباله حسابی با n جمله و قدرنسبت d $= \frac{n}{2}(2a + (n-1)d)$

نرخ تلفات بسته $= \frac{1}{\frac{3}{8}w^2 + \frac{3}{4}w} = L$

W: بزرگ $\quad L \rightarrow \frac{8}{3W^2} \longrightarrow W = \sqrt{\frac{8}{3L}}$

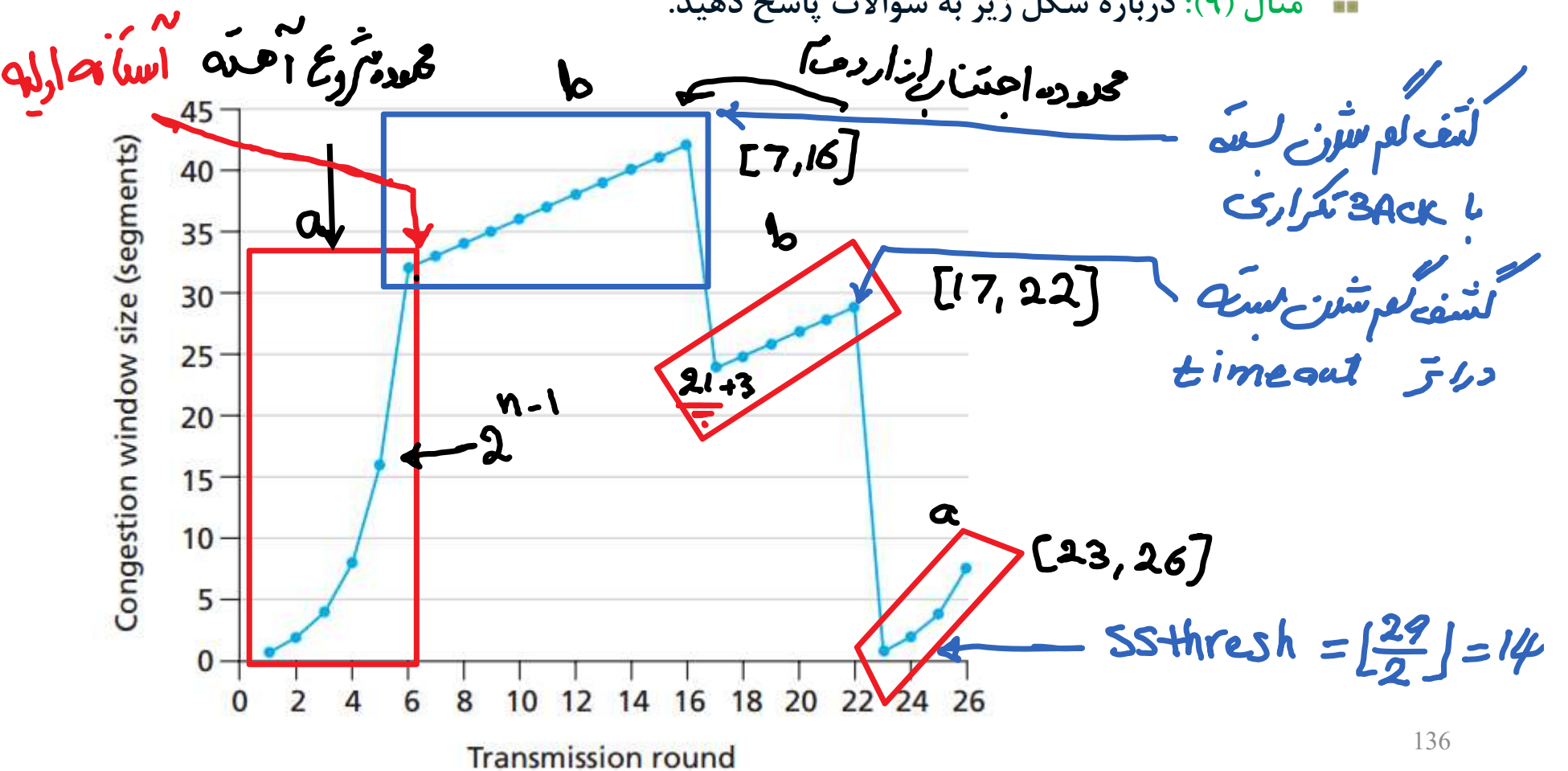
میانگین گذردهی $= \frac{3}{4}\frac{W}{RTT} = \frac{3}{4}\frac{\sqrt{\frac{8}{3L}}\,MSS}{RTT}$

$= \frac{3}{4}\sqrt{\frac{8}{3}}\,\frac{MSS}{\sqrt{L}\,RTT}$

$= \sqrt{\frac{3}{2}}\,\frac{MSS}{\sqrt{L}\,RTT}$

$= \frac{1.22\,MSS}{\sqrt{L}\,RTT}$

مثال (۹): درباره شکل زیر به سوالات پاسخ دهید.



کشف گم شدن بسته با تکراری 3ACK

کشف گم شدن بسته در اثر timeout

$ssthresh = \left\lfloor \frac{29}{2} \right\rfloor = 14$

a) بازه زمانی فاز شروع آهسته را مشخص کنید

b)بازه زمانی فاز اجتناب از ازدحام را مشخص کنید

c)بعد از نوبت ارسال ۱۶و۲۲، رویداد گم شدن بسته چگونه کشف شده است؟

d) مقدار اولیهssthresh در اولین نوبت ارسال چیست؟ 32 MSS

e) مقدار ssthreshدر نوبت ارسال ۱۸ چیست؟ 21 MSS

f) مقدارssthresh در نوبت ارسال ۲۴ چیست؟ 14 MSS

g) هفتادمین قطعه در کدام نوبت ارسال فرستاده می‌شود؟ 7

h) اگر در نوبت ارسال ۲۶ در اثر دریافت سه ACKتکراری، یک رویداد گم شدن اتفاق بیافتد، اندازه پنجره ازدحام و مقدار ssthresh برابر چند است؟

26 → cwnd = 8 —3ACK→ ssthresh = 8/2 = 4 , cwnd = 4 + 3

دور	1	2	3	4	5	6	7 ← اجتناب از ازدحام
cwnd	1	2	4	8	16	32	33
کل قطعات ارسالی	1	3	7	15	31	63	96

قطع هفتم ارسال می شود.

مثال (۱۰): **TCP فایلی را به ۳۲ قطعه تبدیل می‌کند و برای مقصد ارسال می‌کند.اگر بسته بیست و هفتم به مقصد نرسد.در هرکدام از حالت‌های زیر چندRTT زمان صرف ارسال فایل خواهد شد؟**

***مقدارssthreshبرای شروع اهسته برابر۴ است.**

TCP Reno

الف)استفاده از مکانیزم GBN ، کشف گم شدن بسته با سه ACKتکراری

ب)استفاده از مکانیزمGBN، کشف گم شدن بسته با timeout

ج)استفاده از مکانیزم SR، کشف گم شدن بسته با سه ACKتکراری : 8RTT

د)استفاده از مکانیزمSR، کشف گم شدن بسته با timeout : 8RTT

الف

حد	1	2	3	4	5	6	7	8
cwnd	1	2	4	5	6	7	8	4+3
مجموع	1	3	7	12	18	25	33	27-28 33

{26,27,28,29,30,31,32}

8RTT : پاسخ

ب

دور	1	2	3	4	5	6	7	8	9	10
cwnd	1	2	4	5	6	7	8	1	2	4
مجموع	1	3	7	12	18	25	33	27	29	33

timeout

10RTT : الجواب

ج و د : چون فقط نیاز به ارسال مجدد بسته 27 می‌باشد لذا در همان RTT هشتم ارسال شده و کار خاتمه می‌یابد.

مثال (۱۱): دریک ارتباط TCP، ۷۸ بسته برای ارسال وجود دارد.ارسال بسته‌ها از فاز شروع آهسته شروع خواهد شد. چند RTT برای ارسال بسته‌ها لازم خواهد بود؟

الف)اندازه بسته‌ها و پهنای باند ارسال طوری باشد که در هر RTT فقط امکان ارسال ۶ بسته وجود داشته باشد.و درصورت ارسال بسته‌های بیشتر timeout رخ بدهد. 22RTT

ب)اندازه بسته‌ها و پهنای باند ارسال طوری باشد که در هر RTT فقط امکان ارسال ۱۰ بسته وجود داشته باشد و تلفات بسته نداشته باشیم. مقدار آستانه اولیه ۸ است. 11RTT

الف

شروع آهسته

اجتناب ازدحام

دور	1	2	3	4	5	6	7	8	9	10
cwnd	1	2	4	8 6	1	2	4	5	6	7 10
مجموع	1	3	7	13	14	16	20	25	31	37

timeout
ssthresh = 4

timeout
ssthresh = 3

دور	11	12	13	14	15	16	17	18	19	20	21	22
cwnd	1	2	3	4	5	6	7 6	1	2	4	8 6	1
مجموع	38	40	43	47	52	58	64	65	67	71	77	78

شروع آهسته

دور	1	2	3	4	5	6	7	8	9
cwnd	1	2	4	8	9	10	10	10	10
مجموع	1	3	7	15	24	34	44	54	64

ب

دور	10	11
cwnd	10	10
مجموع	74	84

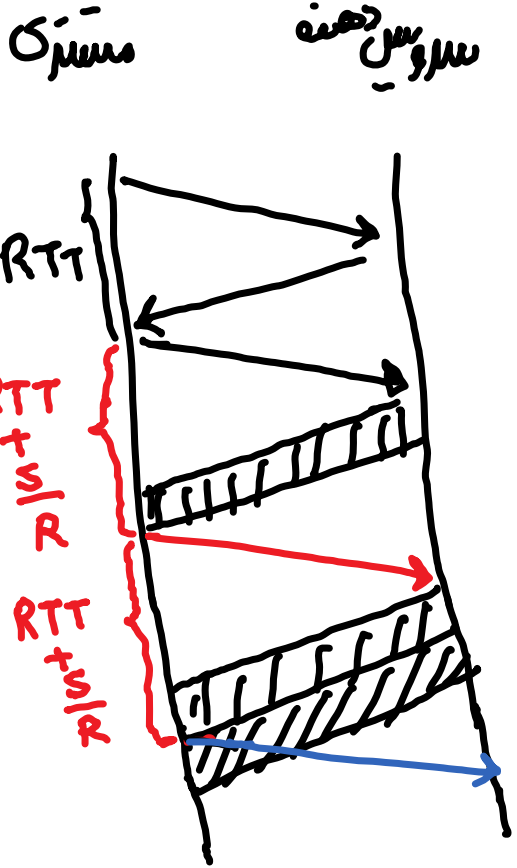
مثال (۱۲): یک مشتری و سرویس‌دهنده در نظر بگیرید، که با یک لینک **R** به هم متصل شده‌اند.اگر مشتری بخواهد یک شی را از سرویس‌دهنده بگیرد که اندازه برابر **15MSS** است.در هرکدام از حالت‌های زیر تاخیر ناشی از مرحله شروع آهسته **TCP** چقدر است؟

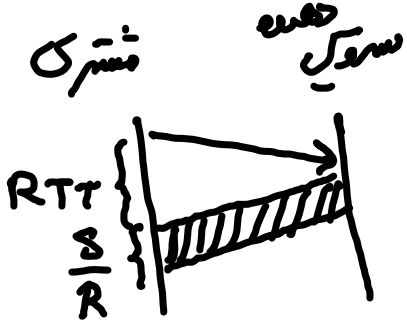
الف) **2MSS/R<MSS/R+RTT<4MSS/R**

ب) **MSS/R>RTT**

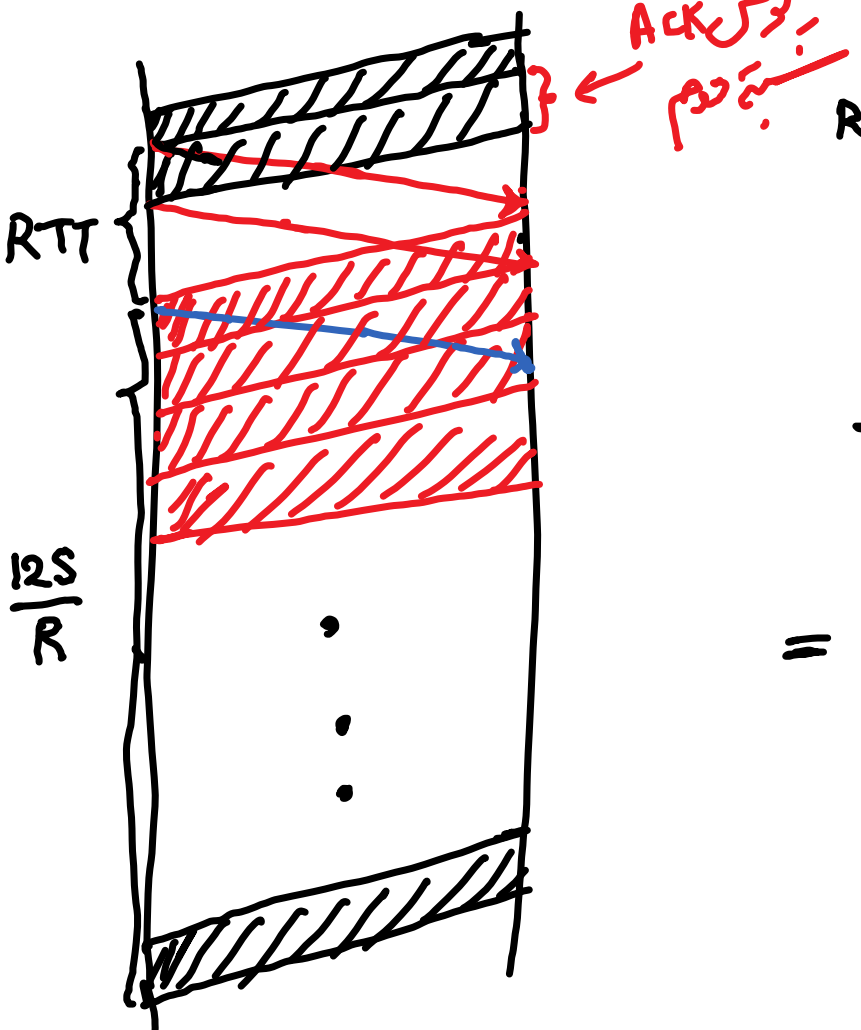
الف)

$$\frac{S}{R} < RTT < 3\frac{S}{R}$$





دور	1	2
cwnd	MSS	2MSS

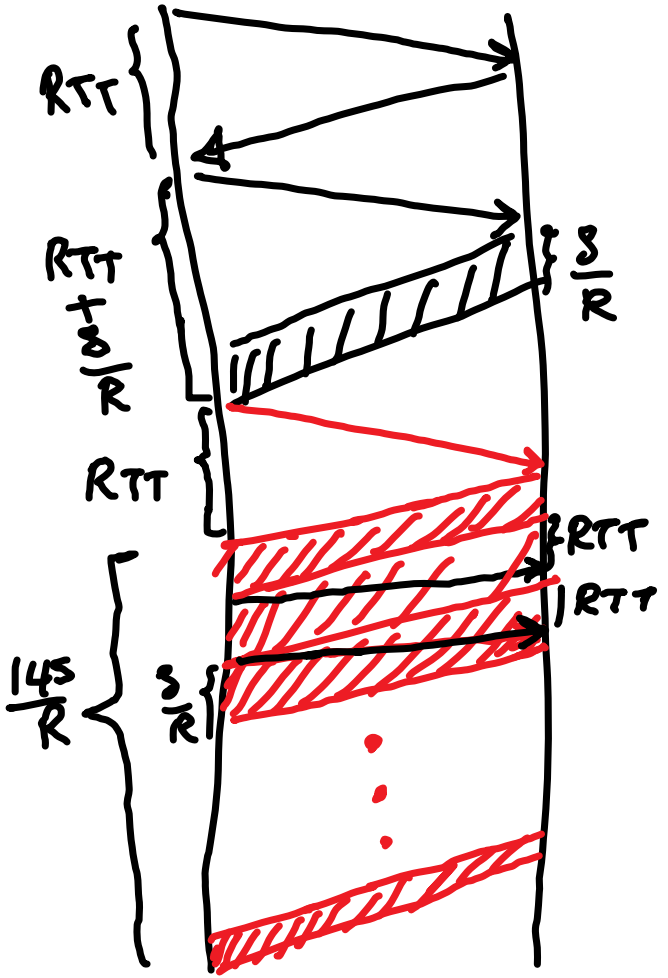


Ack بیت دوم

$$RTT + \underbrace{RTT + \frac{S}{R}}_{\text{قطعه اول}} + \underbrace{RTT + \frac{S}{R}}_{\text{قطعه دوم}} \cdot$$

$$+RTT + 12\frac{S}{R}$$

$$= 4RTT + 14\frac{S}{R}$$



$$RTT + RTT + \frac{S}{R} + RTT + \frac{14S}{R} \quad (\text{ack})$$

$$= 3RTT + \frac{15S}{R}$$

$$\frac{S}{R} > RTT$$