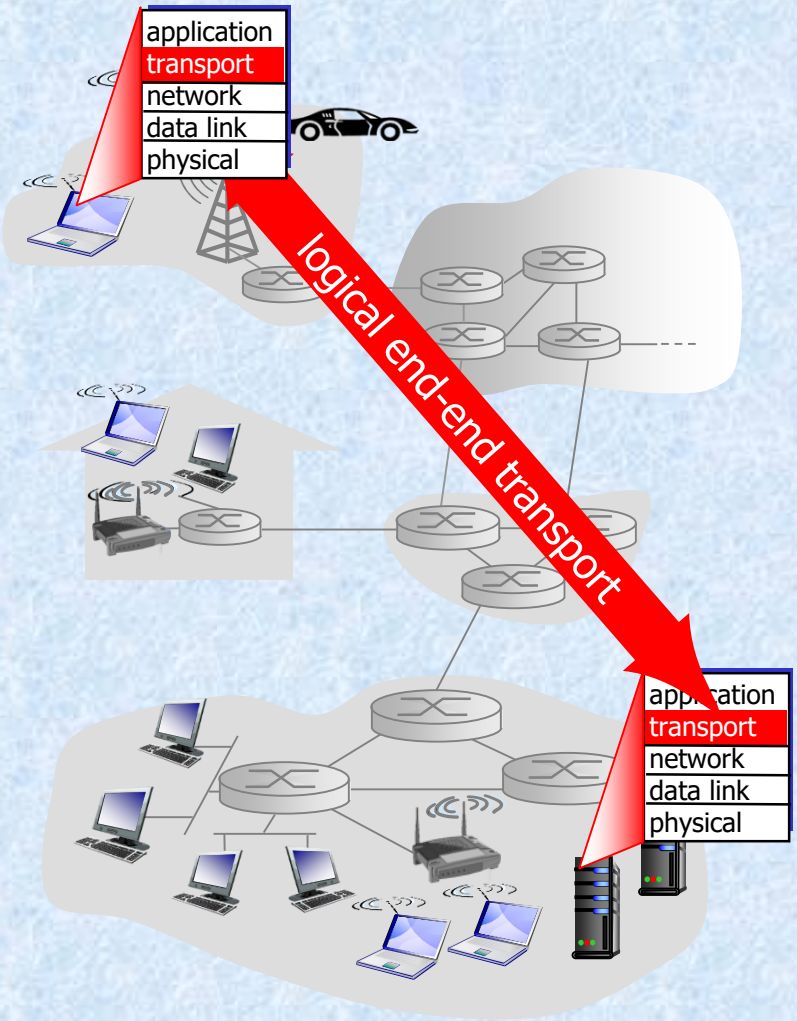
لایه انتقال

ساختار اصولی ارتباط میزبان به میزبان

رئوس مطالب

- خدمات لایه انتقال
- مالتی پلکسینگ
- پروتکل انتقال فایل (FTP)
- خدمات انتقال بدون اتصال
  - معرفی پروتکل UDP
- اصول انتقال قابل اطمینان
- خدمات انتقال اتصال گرا
  - معرفی پروتکل TCP
  - فرآیند کنترل جریان
  - مدیریت اتصال
- کنترل ازدحام

پروتکل ها و خدمات انتقال



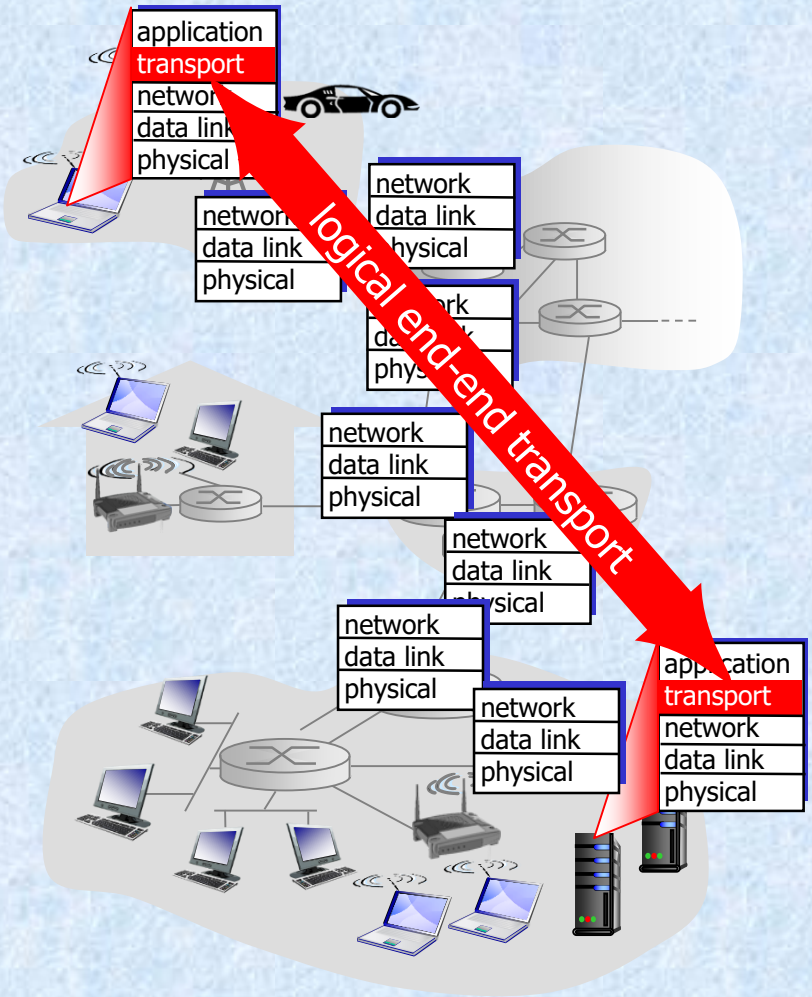
- برقراری ارتباط منطقی بین پردازش ها (برنامه های کاربردی)ی در حال اجرا در میزبان ها
- پروتکل انتقال فرستنده
  - شکستن پیام برنامه کاربردی به قطعات (**Segment**) و انتقال به لایه شبکه
- پروتکل انتقال سمت گیرنده
  - هم بندی قطعات و تحویل به لایه کاربرد
- پروتکل های لایه انتقال
  - اینترنت: **TCP, UDP, SCTP**

لایه های انتقال و شبکه

- لایه شبکه
  - اتصال منطقی میان میزبان ها
- لایه انتقال
  - اتصال منطقی میان پردازش ها
- عملیات لایه انتقال
  - استفاده از خدمات لایه شبکه
  - بهبود خدمات شبکه

- مثال
- ارسال نامه از طرف ۱۲ کارمند مستقر در سازمان ۱ به ۱۲ کارمند در سازمان ۲
  - میزبان ها = سازمان ها
  - پردازش ها = کارمندان
  - پیام های لایه کاربرد = نامه های داخل پاکت
  - پروتکل لایه انتقال = دبیرخانه سازمان ها
  - پروتکل لایه شبکه = نامه رسان

پروتکل های لایه انتقال



- تحویل داده قابل اطمینان – مرتب
  - کنترل ازدحام
  - کنترل جریان
  - برقراری اتصال
- تحویل داده غیر قابل اطمینان – نا مرتب
  - بر مبنای بهترین-تلاش
  - بدون ضمانت (تحویل سالم یا مرتب)
- خدمات غیر قابل دسترس در پروتکل های لایه انتقال
  - ضمانت تاخیر
  - ضمانت پهنای باند

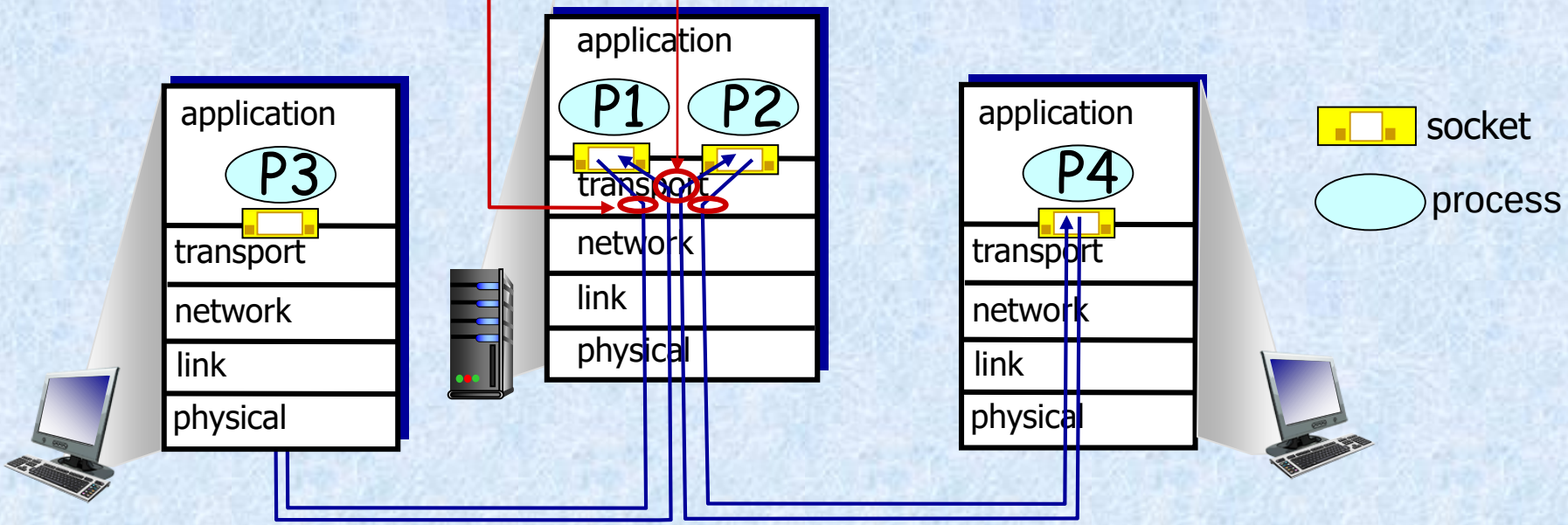
مالتی پلکسینگ

سمت فرستنده

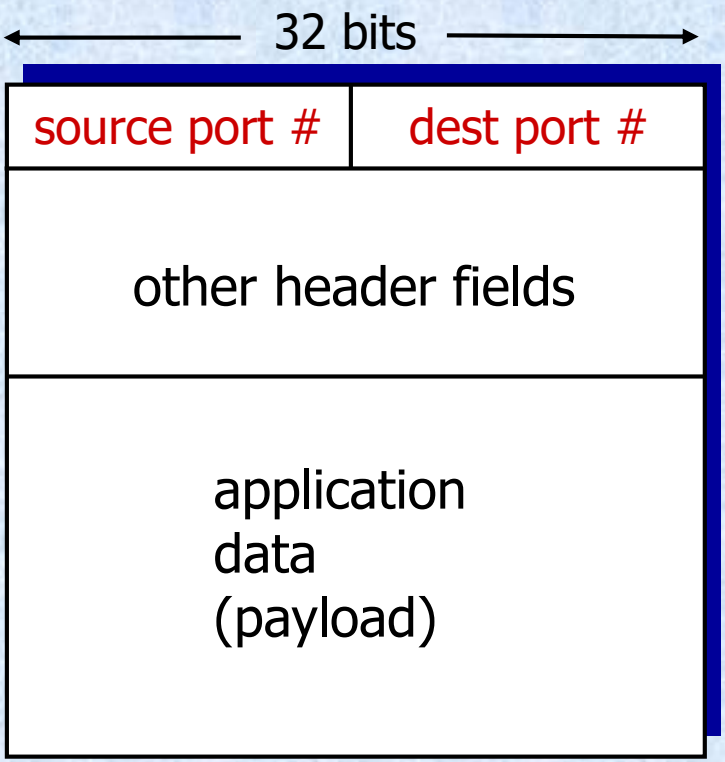
دریافت داده از سوکت های مختلف
اضافه کردن اطلاعات انتقال در سرآیند

سمت گیرنده

استفاده از سرآیند برای تحویل سگمنت ها به سوکت مورد نظر



دی مالتی پلکسینگ



TCP/UDP segment format

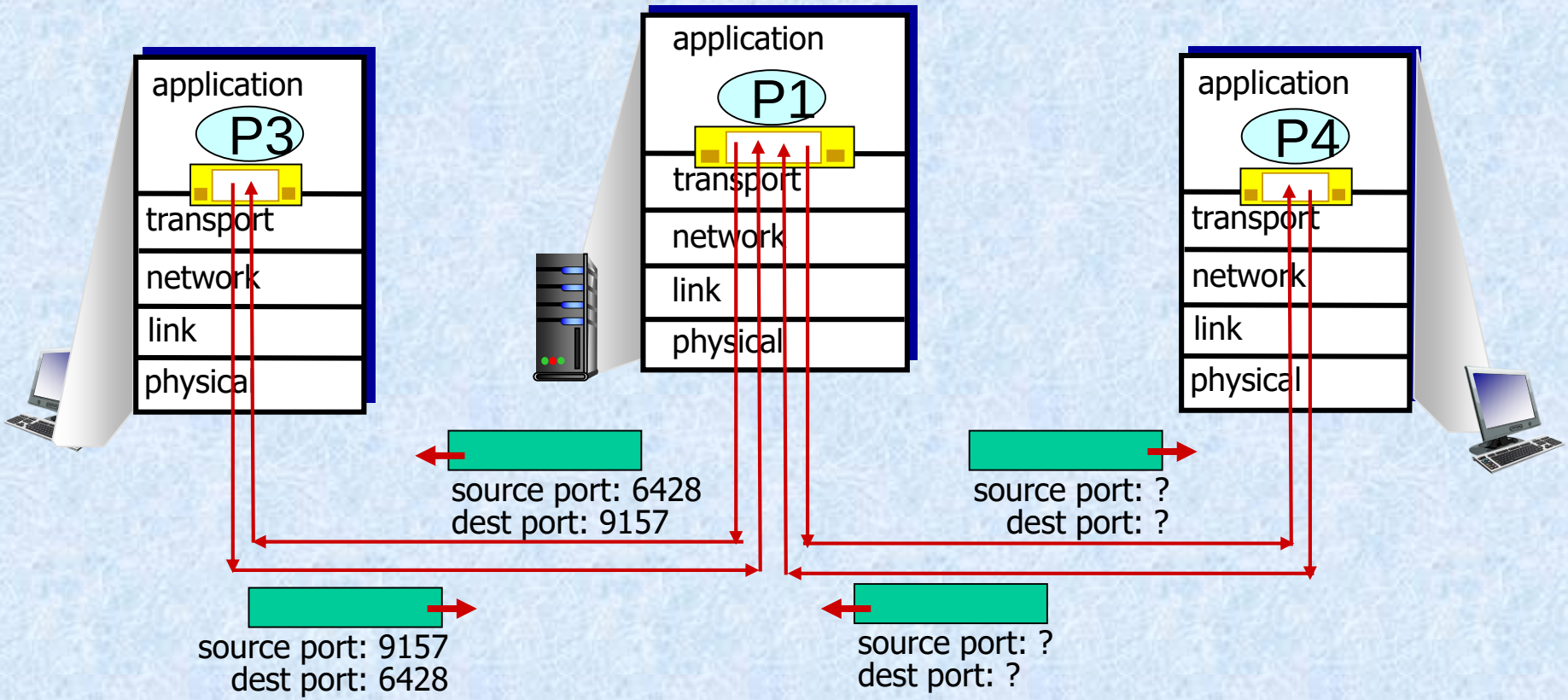
- دریافت بسته های **IP** توسط میزبان گیرنده
  - هر بسته دارای آدرس های (**IP**) فرستنده و گیرنده است
  - هر بسته **IP** دارای یک سگمنت
  - هر سگمنت دارای آدرس های (پورت) مبدا و مقصد
- انتقال سگمنت های دارای آدرس های **IP** و پورت به سوکت مورد نظر

دی مالتی پلکسینگ بدون اتصال (UDP)

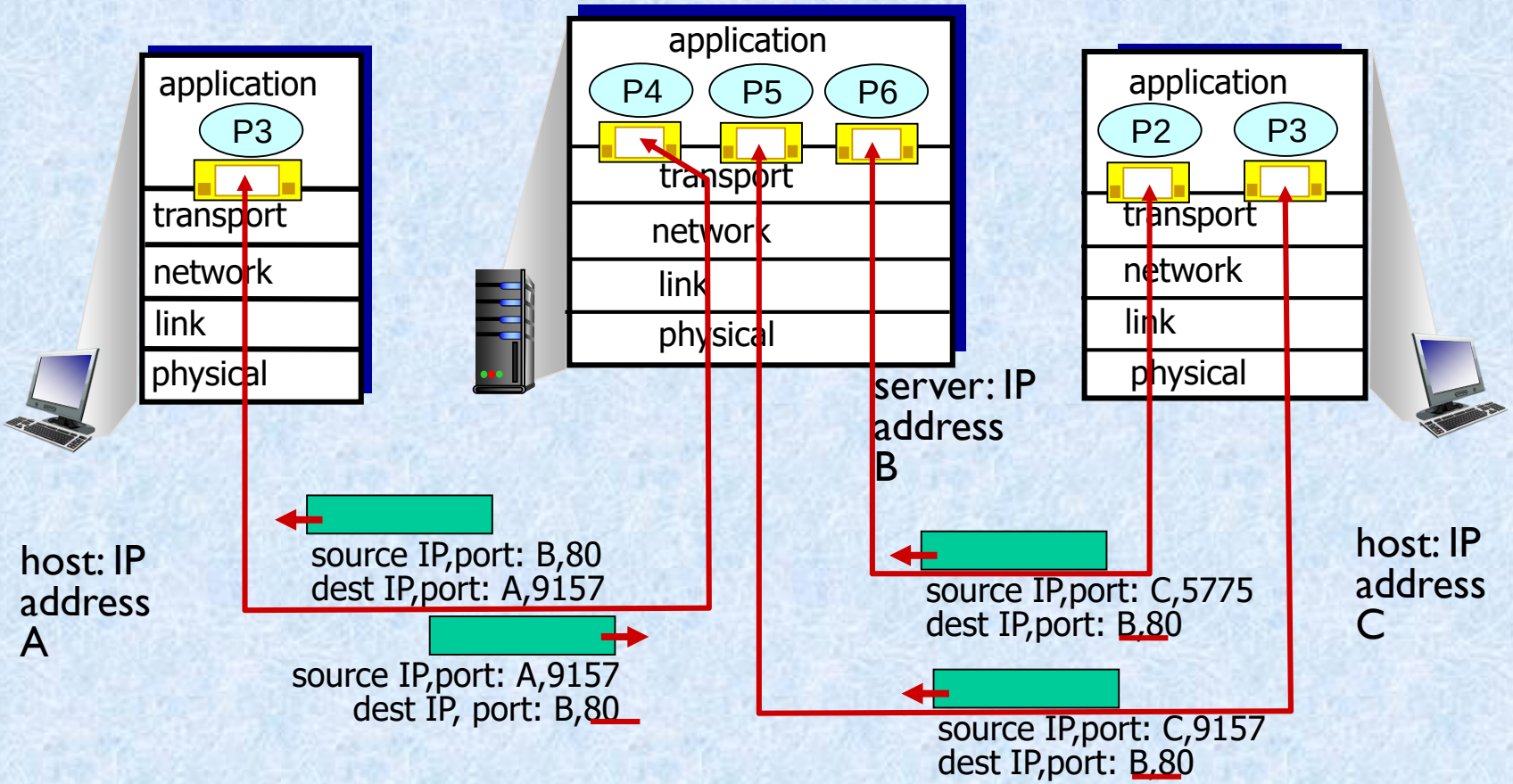
ایجاد یک سوکت روی شماره پورت
خدمات ۶۴۲۸

ایجاد یک سوکت با شماره
پورت تصادفی ۹۱۵۷

ایجاد یک سوکت با شماره پورت
تصادفی ۵۷۷۵



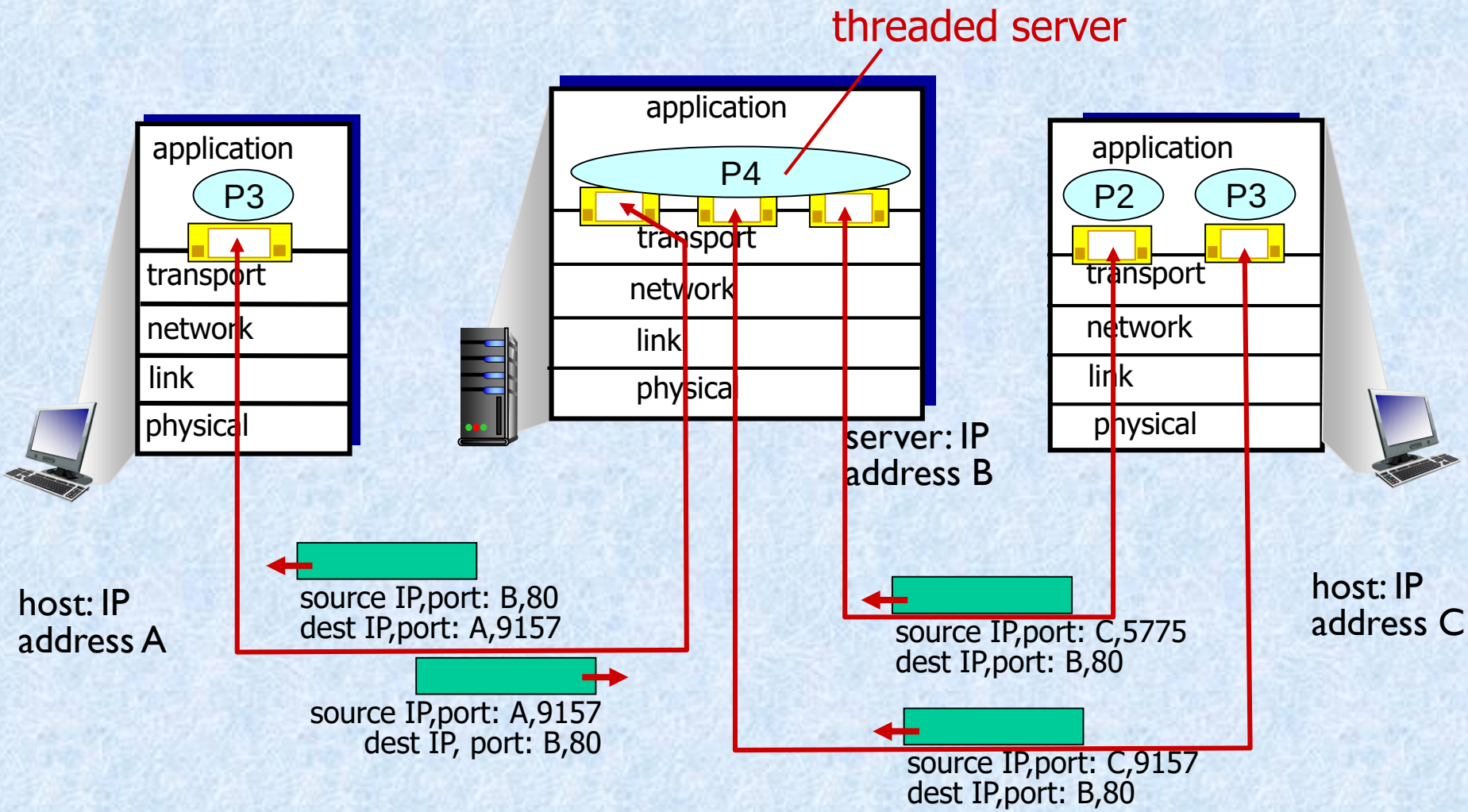
دی مالتی پلکسینگ اتصال گرا (TCP)



یک سوکت با شماره پورت مبدا ۹۱۵۷ به مقصد پورت ۸۰

دو سوکت با شماره پورت های مبدا ۵۷۷۵ و ۹۱۵۷ به مقصد پورت ۸۰

دی مالتی پلکسینگ اتصال گرا: زیر–پردازش

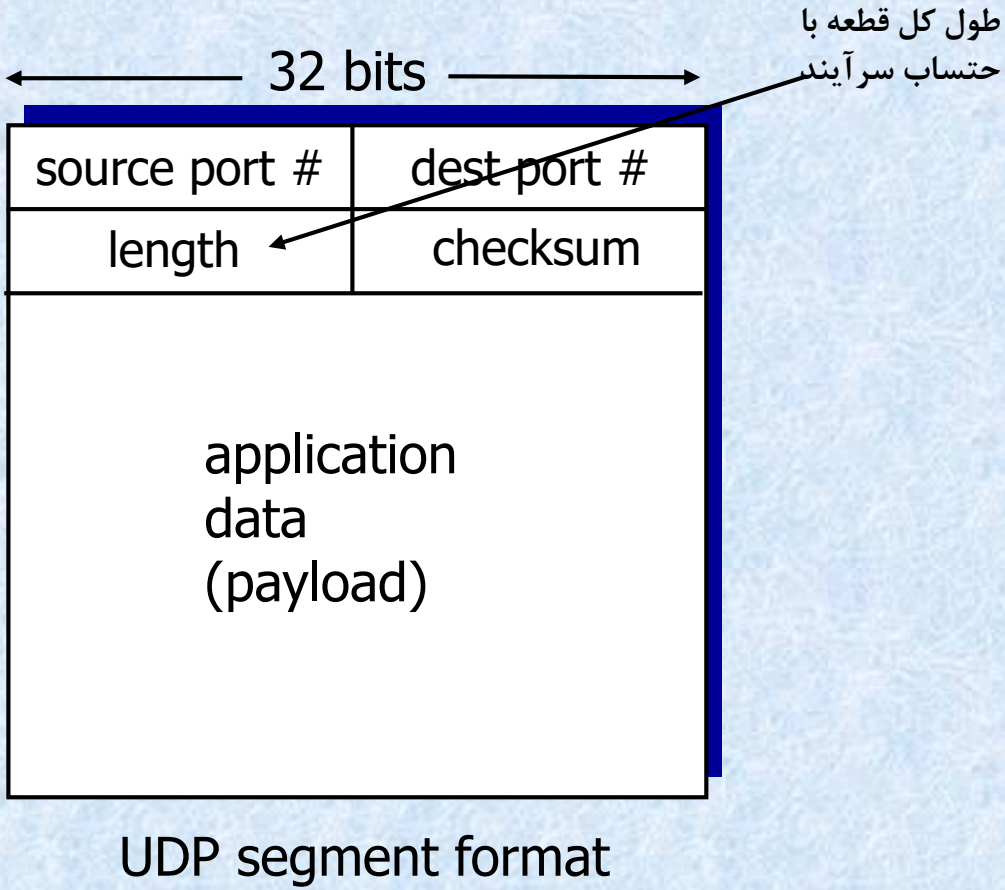


پروتکل دیتاگرام کاربر (UDP)

- پروتکل انتقال خلاصه/خالص
- بدون فرآیند کنترل خطا
- تحویل بدون ترتیب
- بدون برقراری اتصال
  - بدون فرآیند دست تکانی
  - انتقال هر قطعه مستقل از قطعات دیگر

- کاربرد
  - پخش چندرسانه ای (تحمل به تلفات، حساس به پهنای باند)
  - DNS
  - SNMP
  - RIP

ساختار سگمنت UDP



UDP segment format

- مزایای UDP
- بدون نیاز به اتصال
  - تاخیر پایین
- ساده
  - بدون نگهداری وضعیت اتصال
- اندازه سرآیند کوچک
- بدون کنترل ازدحام
  - سرعت انتقال بالا

جمع کنترلی در UDP

سمت فرستنده

- تجزیه هر قطعه به رشته (بلوک) های ۱۶ بیتی
  - شامل سرآیند
- جمع دو به دوی بلوک های ۱۶ بیتی به روش مکمل ۱
  - حفظ طول ۱۶ بیتی حاصل جمع
- جمع حاصل با بلوک بعدی
- مکمل گیری نهایی و بدست آوردن جمع کنترلی
- قرار دادن مقدار جمع کنترلی در فیلد مربوطه

سمت گیرنده

- دریافت قطعه و تجزیه به بلوک های ۱۶ بیتی
- محاسبه جمع کنترلی با جمع دو به دوی بلوک ها
- مقایسه حاصل جمع با مقدار جمع کنترلی ارسالی
  - مقدار صفر: بدون خطا
  - غیر صفر: وجود خطا

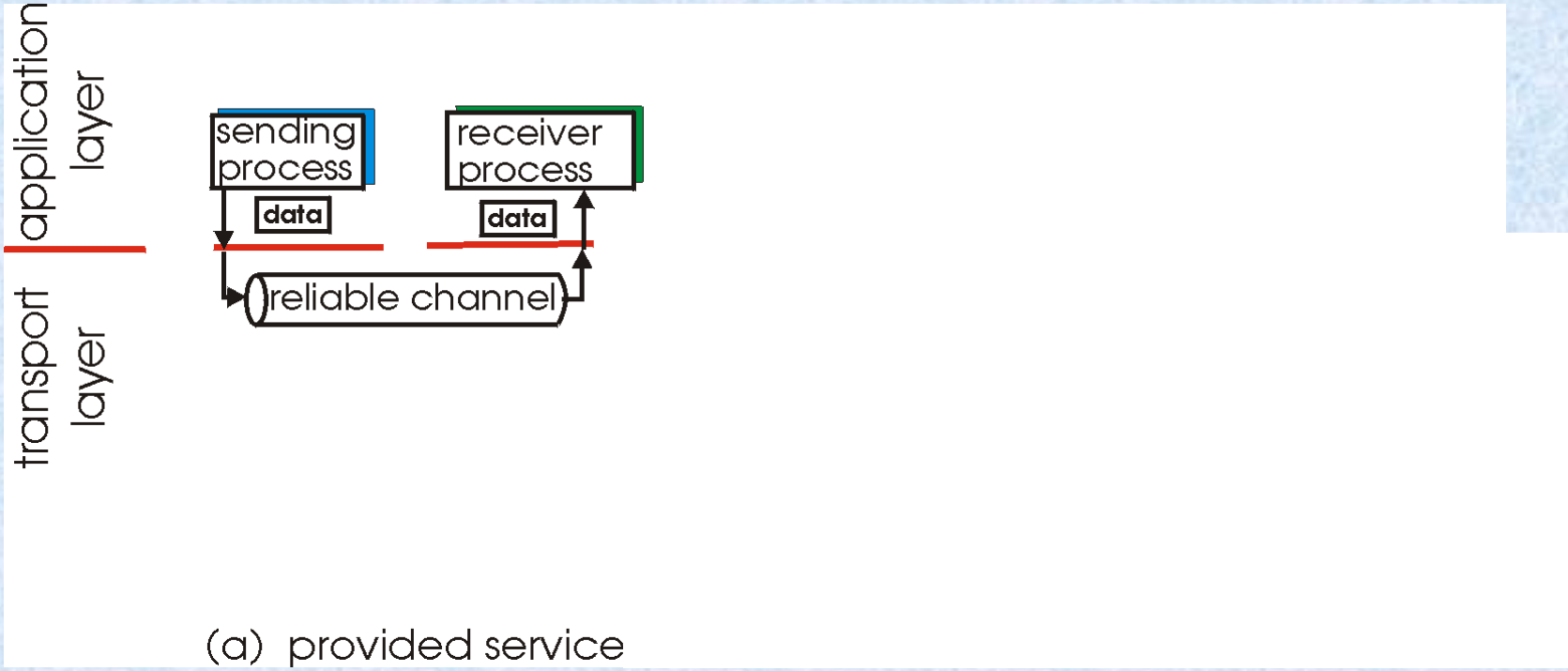
مثال

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

*نکته: بیت انتقالی (**wraparound**) بدست آمده به حاصل جمع اضافه می شود*

اصول انتقال قابل اطمینان (rdt)

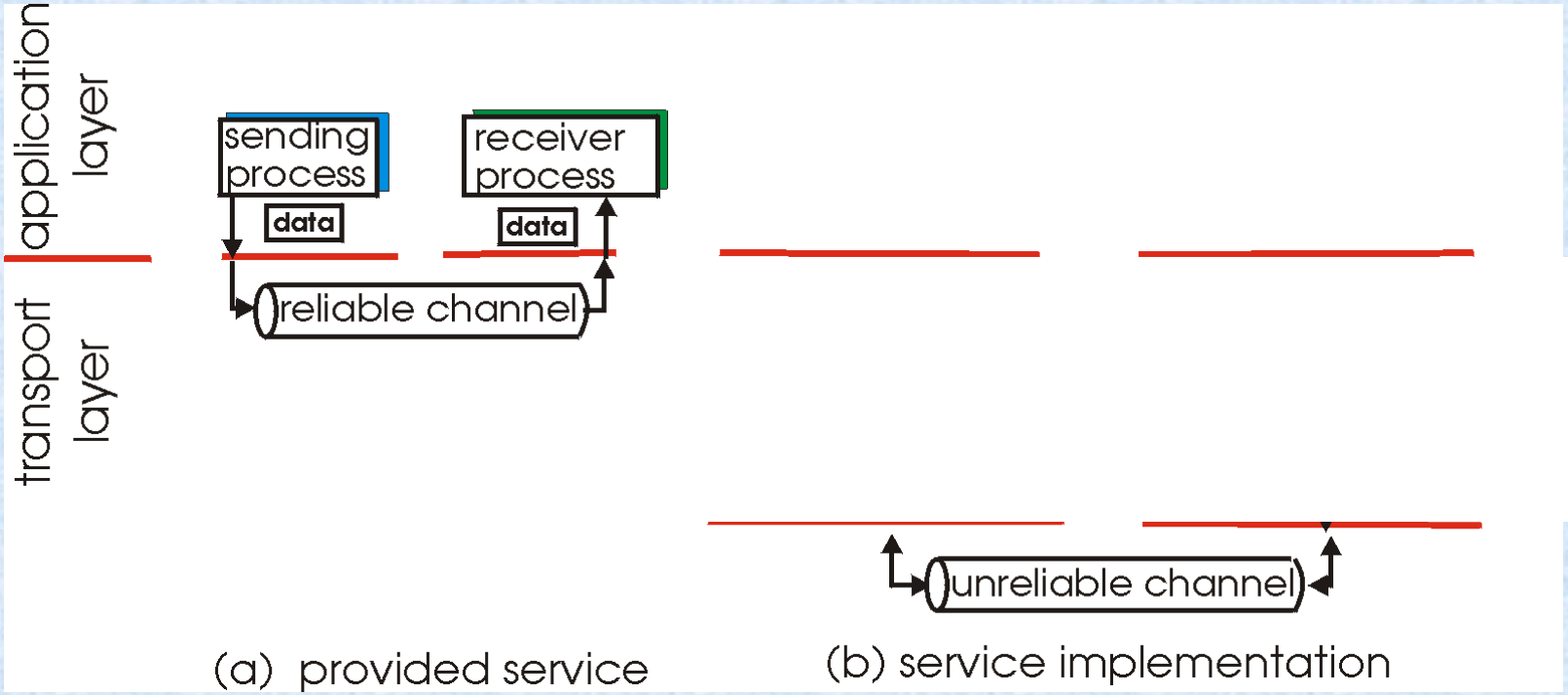
- حائز اهمیت در لایه های کاربرد، انتقال، پیوند داده
  - از مسائل مهم حال حاضر در مطالعات شبکه



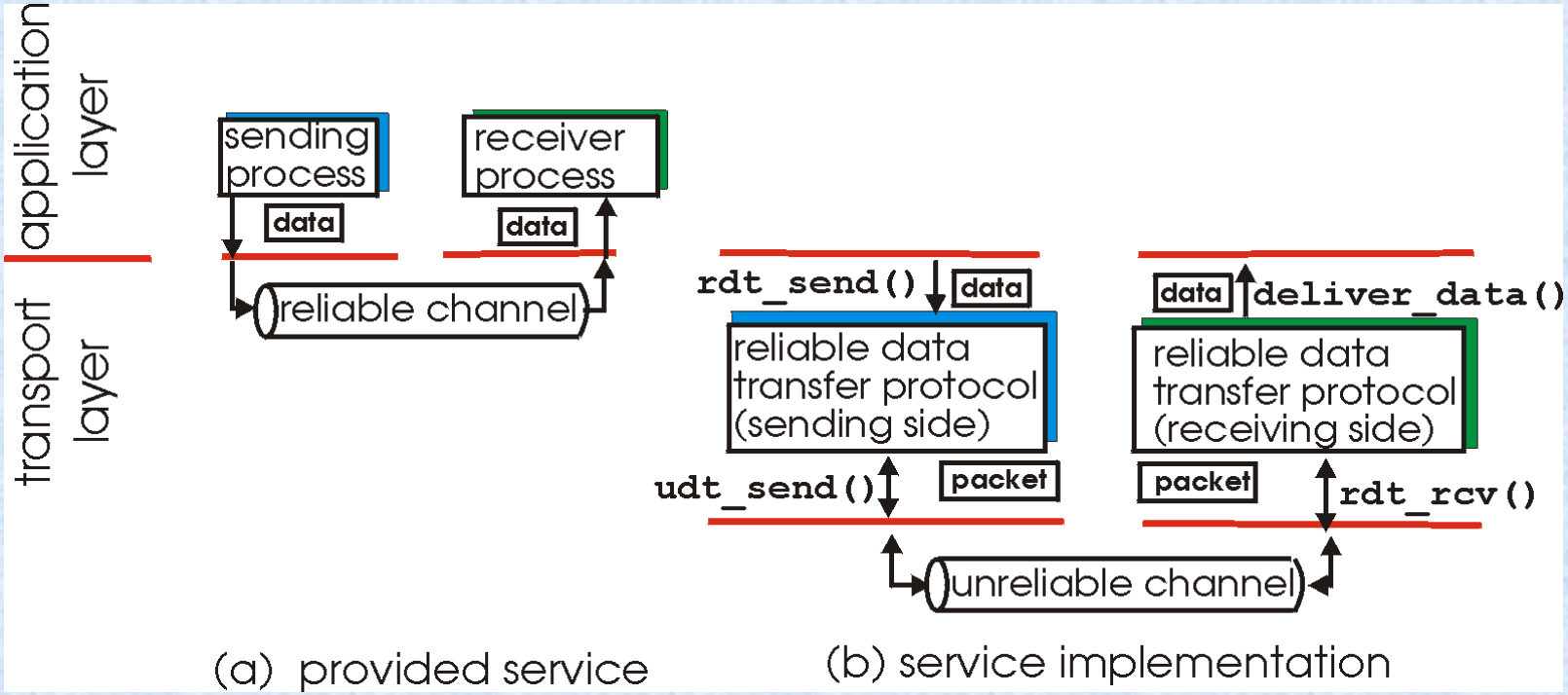
- عامل مهم در پیچیدگی های انتقال قابل اطمینان
  - کانال انتقال غیر قابل اطمینان

سوال: منظور از کانال چیست و چرا غیر قابل اطمینان است؟؟

اصول انتقال قابل اطمینان (rdt)...



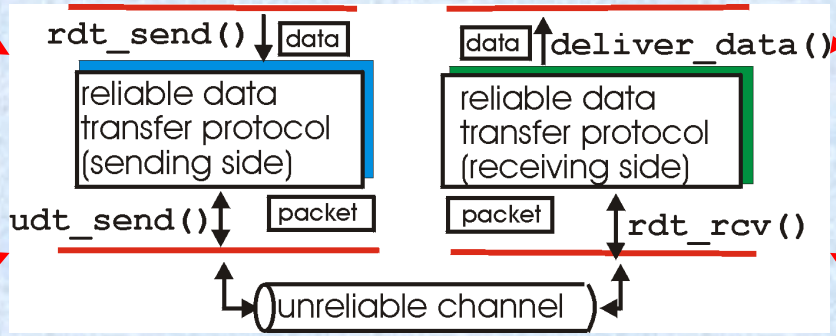
اصول انتقال قابل اطمینان (rdt)...



انتقال قابل اطمینان (*rdt*)

rdt_send()
فراخوانی توسط برنامه، ارسال داده به گیرنده

deliver_data()
فراخوانی توسط **rdt** برای تحویل داده به برنامه



فرستنده

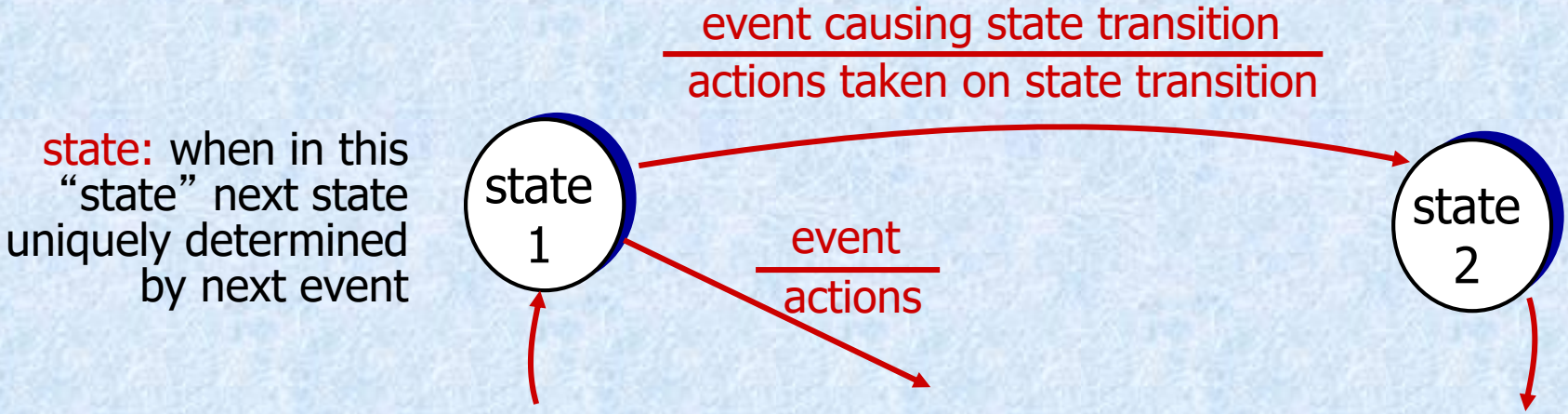
گیرنده

udt_send()
فراخوانی توسط **rdt** برای انتقال بسته به گیرنده

rdt_rcv()
فراخوانی در زمان دریافت بسته

طراحی rdt1.0

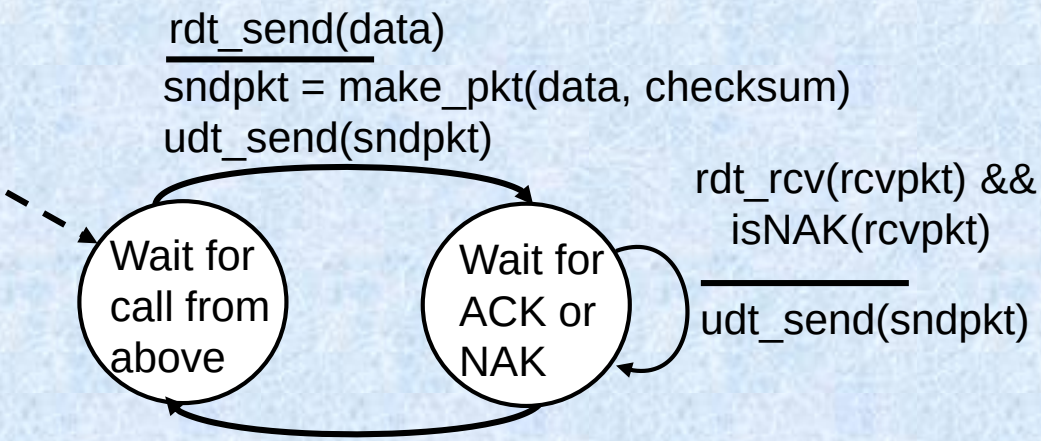
- مبتنی بر FSM
- طراحی گام به گام
- فرض های اولیه
  - لینک قابل اطمینان (عدم وجود خطا و تلف بسته)
  - مسیر یک طرفه (یک سمت فرستنده-یک سمت گیرنده)
    - فرستنده: ارسال داده روی کانال (انتظار برای درخواست برنامه کاربردی)
    - گیرنده: دریافت داده از کانال (انتظار برای ورود داده از کانال)



rdt2.0: کانال با خطا

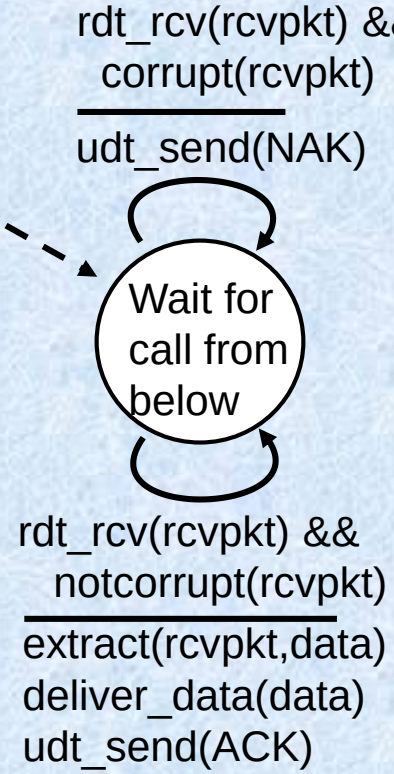
- کانال غیر قابل اطمینان: تغییر بیت ها در بسته
  - تشخیص خطای بیت (جمع کنترلی)
- *نحوه کنترل خطا؟*
  - *استفاده از پیام پاسخ مثبت (ACK): اطلاع به فرستنده از دریافت صحیح بسته توسط گیرنده*
  - *پیام پاسخ منفی (NAK):اطلاع به فرستنده مبنی بر عدم دریافت صحیح بسته*
  - ارسال مجدد بسته توسط فرستنده با دریافت NAK
  - حذف بسته ذخیره شده در فرستنده با دریافت ACK

rdt2.0

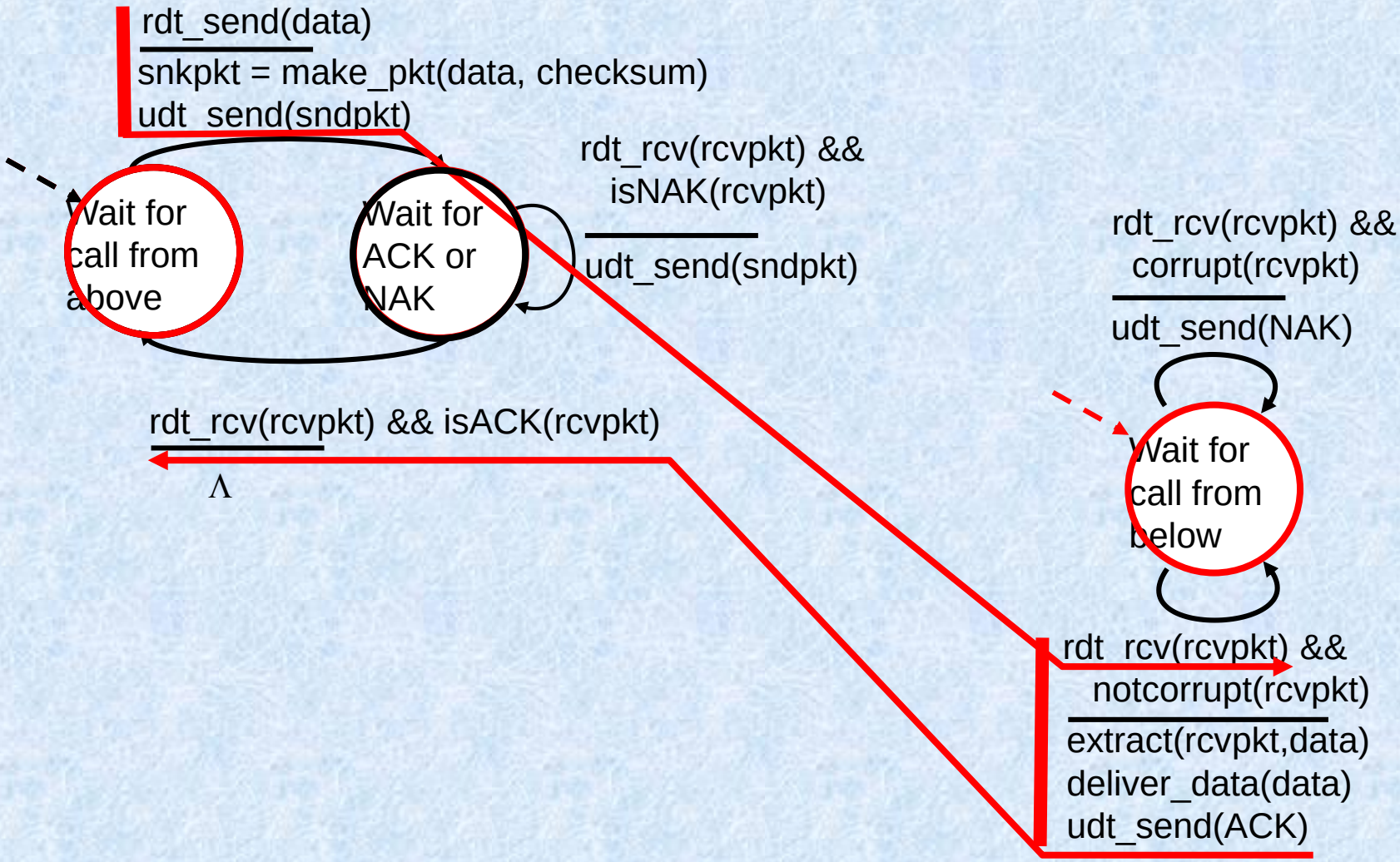


فرستنده

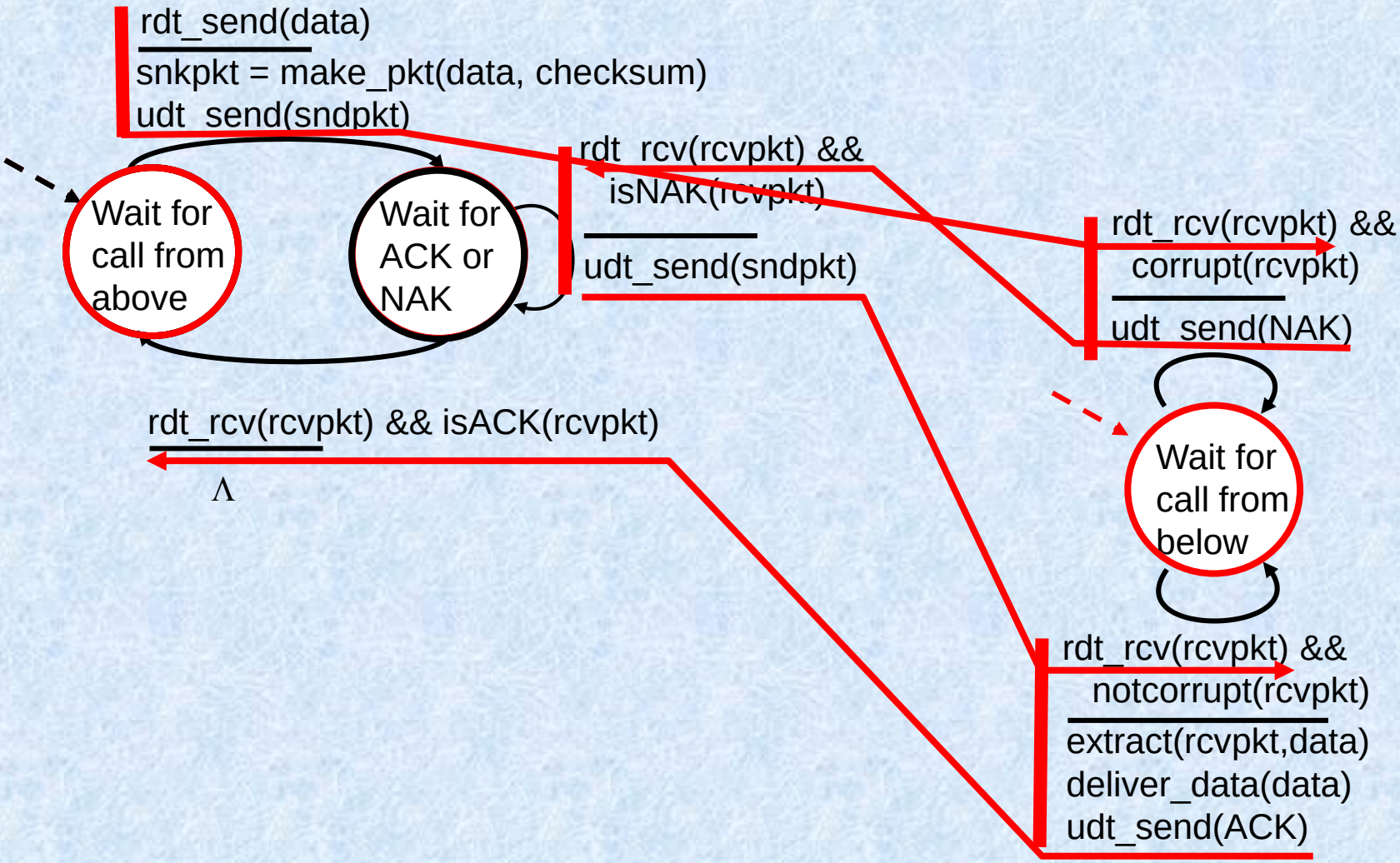
گیرنده



سناریوی بدون خطا :rdt2.0



rdt2.0: رخ دادن خطا



rdt2.0: اشکالات اساسی

بروز خطا در پیام های پاسخ (ACK/NAK)

- عدم اطلاع فرستنده از وضعیت بسته
- عدم دریافت وضعیت از گیرنده
- ارسال مجدد ← تکرار بسته

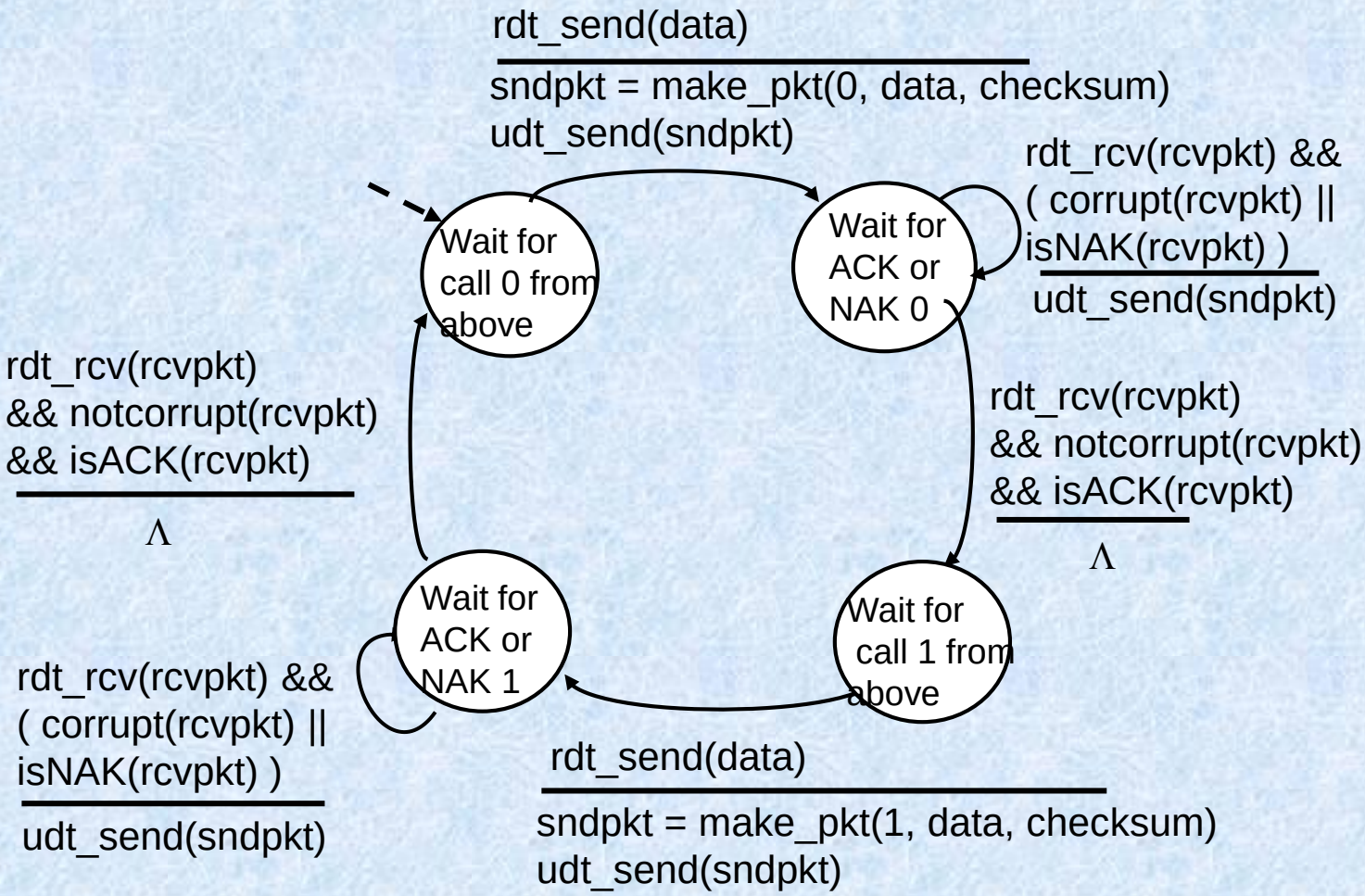
برخورد با تکرار بسته

- ارسال مجدد صرفا در صورت دریافت پیام پاسخ منفی
- حذف بسته در صورت دریافت پیام پاسخ مثبت
- اضافه کردن شماره ترتیب به هر بسته
- کنترل بسته های تکرای در گیرنده با شماره ترتیب

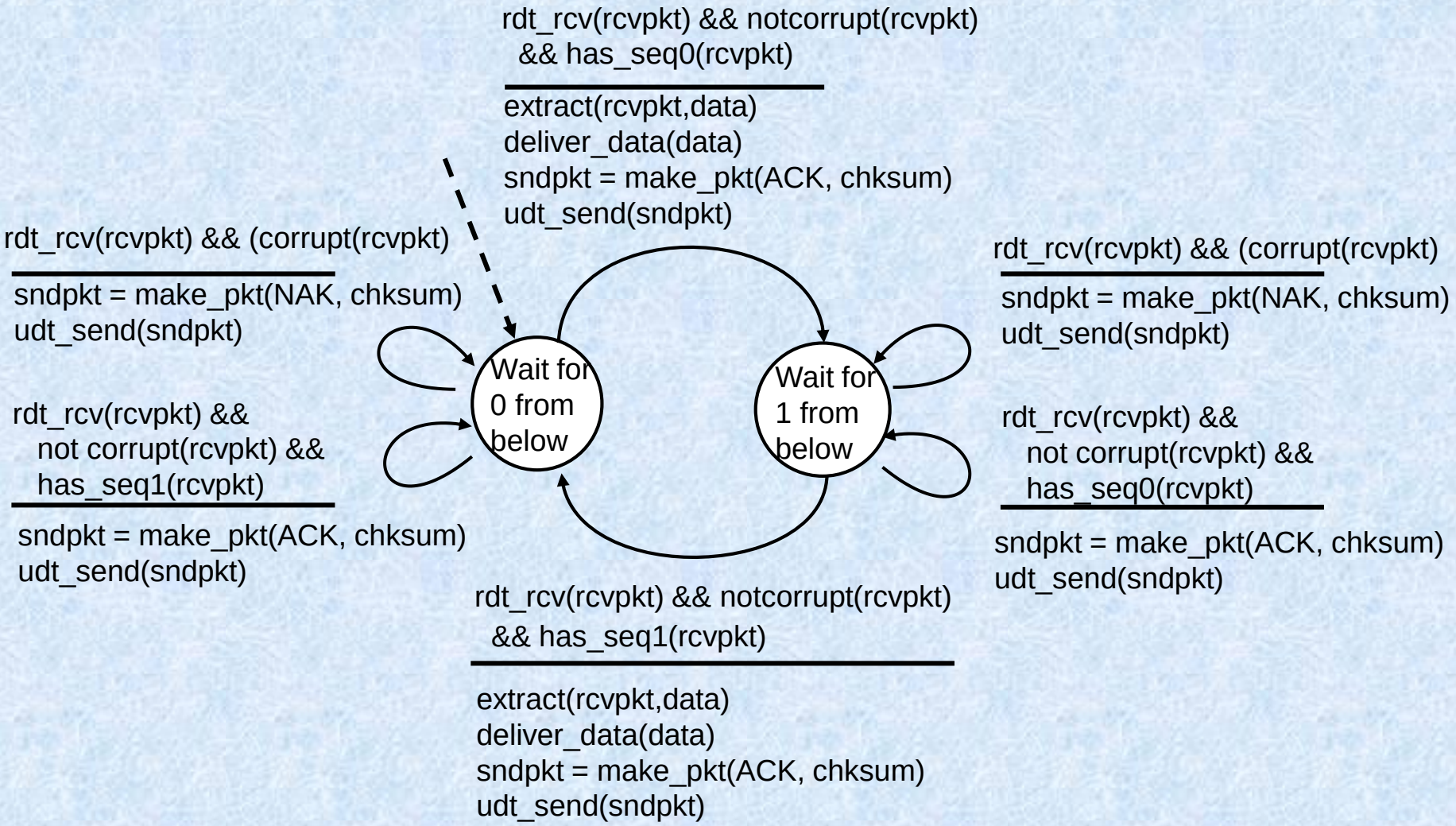
توقف و انتظار

- در فرستنده:
  - ارسال یک بسته
  - توقف ارسال
  - انتظار برای دریافت پاسخ

rdt2.1: (فرستنده)شماره ترتیب برای پیام های پاسخ



گیرنده :rdt2.1



rdt2.1: بررسی

فرستنده

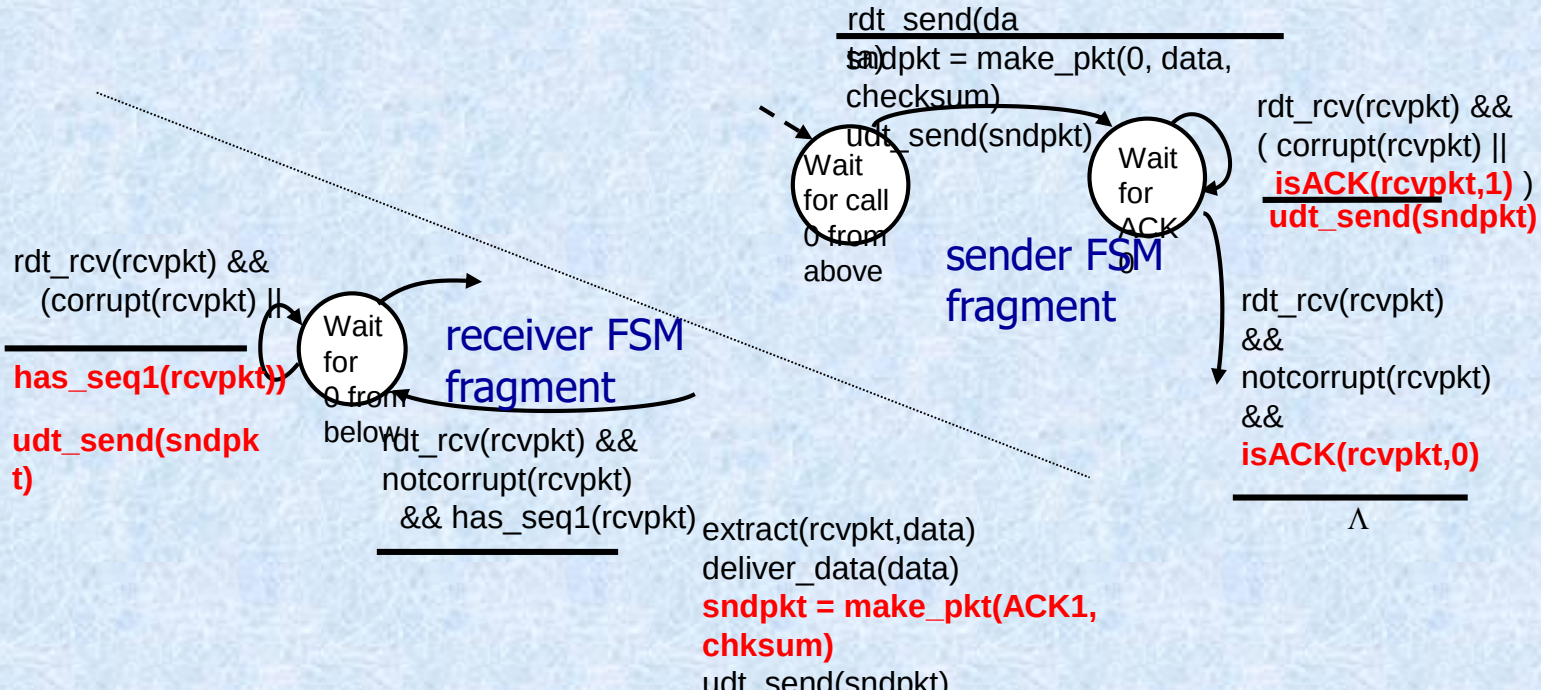
- اضافه کردن شماره ترتیب به بسته
- تعداد شماره ترتیب: ۲ (0,1)
- کنترل خطا روی پیام های پاسخ
- افزودن یک حالت بیشتر
  - برای تشخیص شماره ترتیب

گیرنده

- بررسی تکراری بودن بسته
- افزودن یک حالت برای اعلام شماره ترتیب مورد انتظار
- نکته
  - گیرنده امکان تشخیص دریافت صحیح پاسخ در فرستنده را ندارد

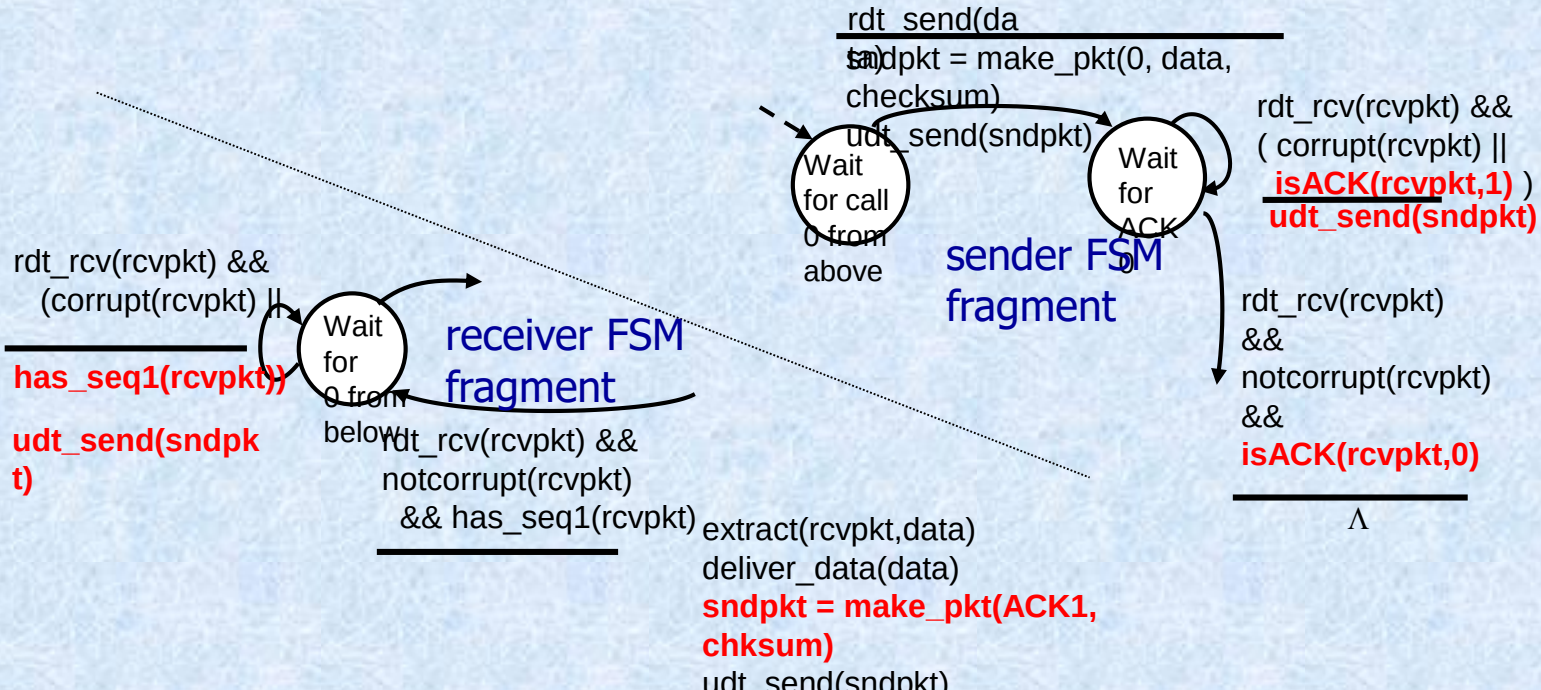
rdt2.2: پروتکل بدون NAK

- گیرنده: ارسال ACK مجدد در صورت دریافت بسته با خطا
  - شماره ترتیب پاسخ برابر شماره آخرین بسته دریافت شده سالم
- فرستنده: بررسی تکراری بودن پاسخ
  - ارسال مجدد آخریت بسته در صورت دریافت ACK تکراری



rdt2.2: پروتکل بدون NAK

- گیرنده: ارسال ACK مجدد در صورت دریافت بسته با خطا
  - شماره ترتیب پاسخ برابر شماره آخرین بسته دریافت شده سالم
- فرستنده: بررسی تکراری بودن پاسخ
  - ارسال مجدد آخریت بسته در صورت دریافت ACK تکراری



rdt3.0: کانال با خطا و تلف بسته

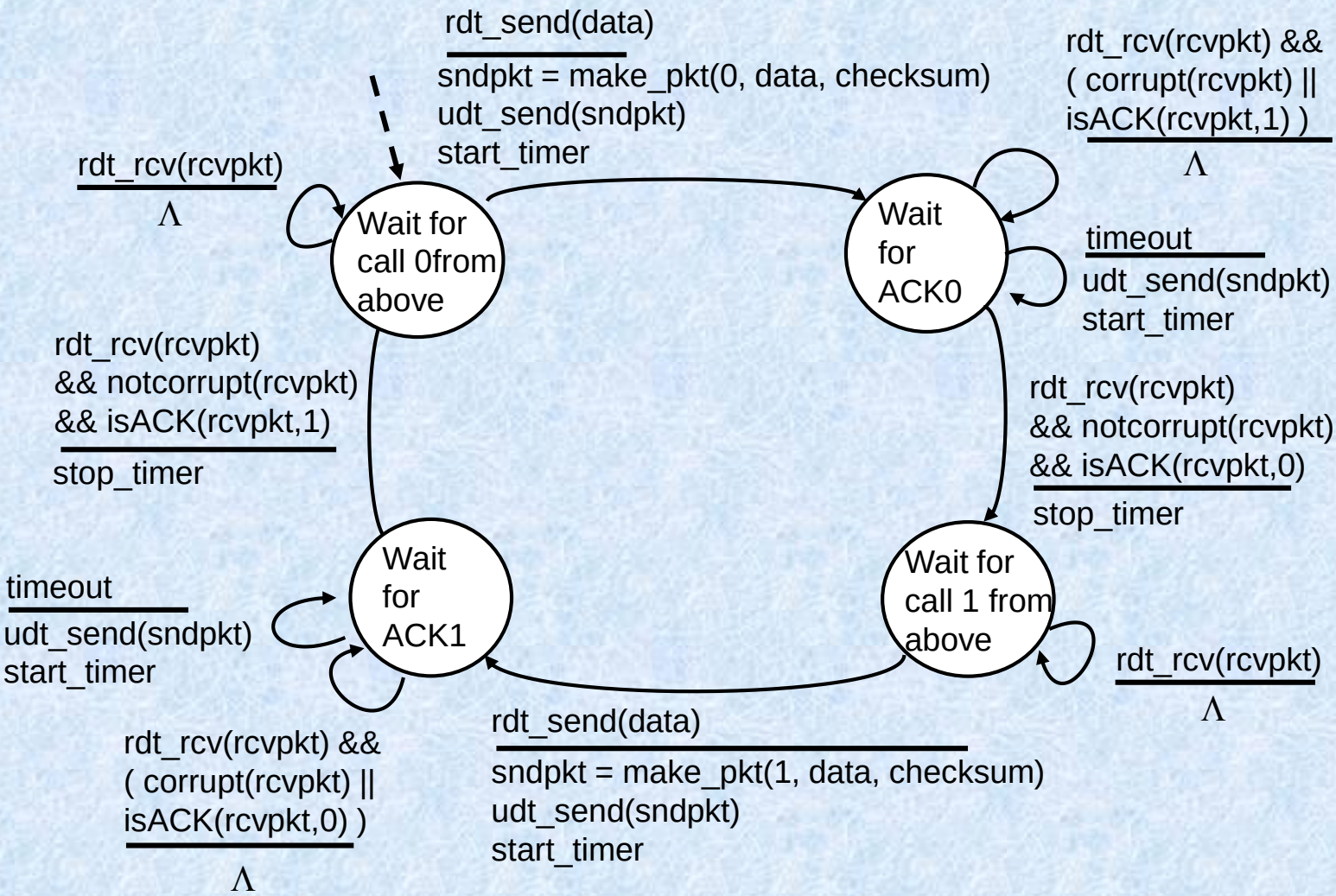
مسئله

- احتمال گم شدن (تلف) بسته در کانال حین ارسال
  - شامل هر دو بسته داده و پاسخ
- استفاده از جمع کنترلی، شماره ترتیب، ارسال مجدد برای تلف کارساز نیست

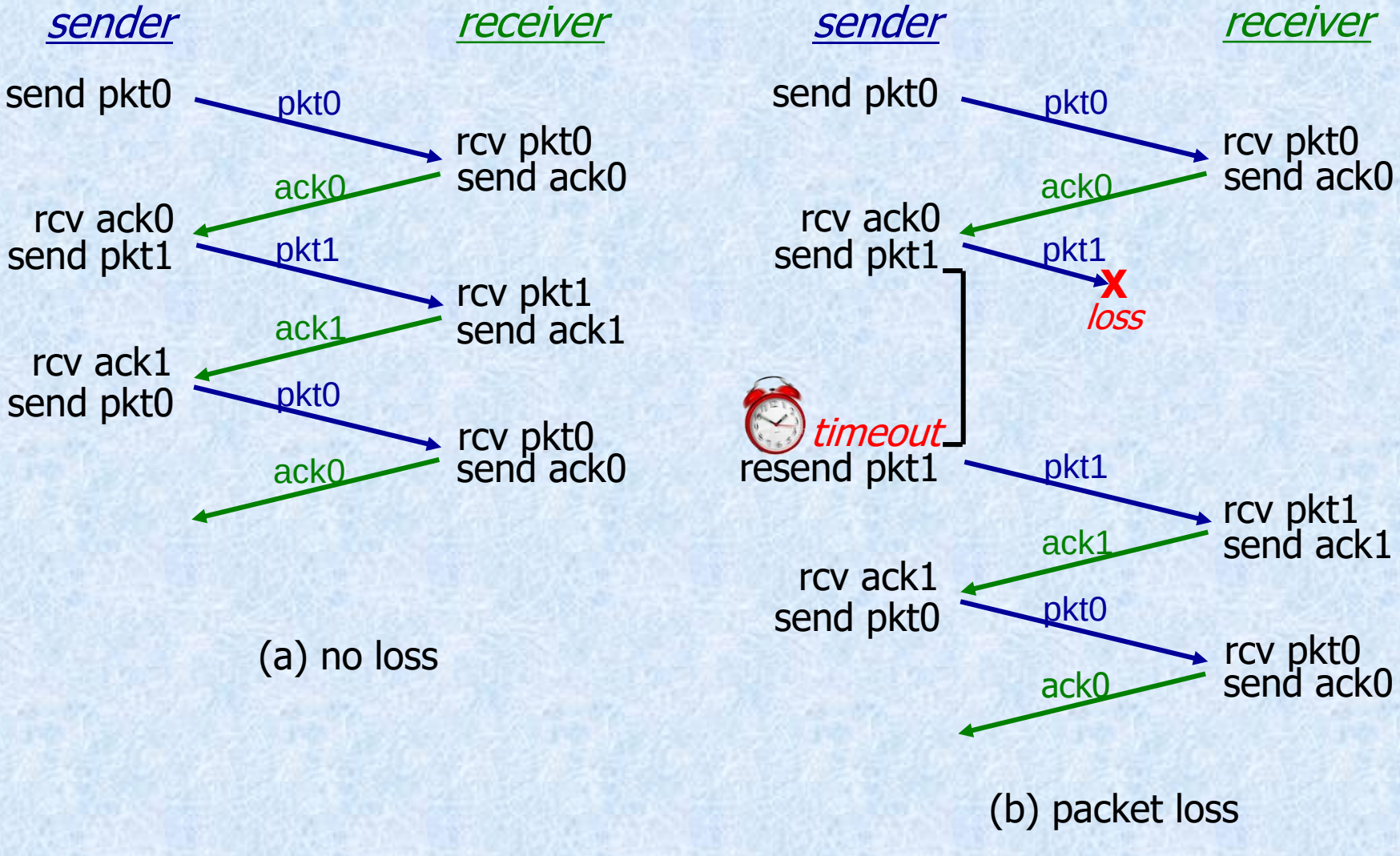
پیشنهاد

- انتظار برای پاسخ در فرستنده به مدت مشخص
- ارسال مجدد در صورت عدم دریافت پاسخ مثبت
- در صورتی که بسته (داده/پاسخ) فقط دارای تاخیر باشد
  - ارسال مجدد باعث تکرار می شود ولی با شماره ترتیب قابل تشخیص است
  - گیرنده: مشخص کردن شماره ترتیب بسته دریافت شده در پاسخ
  - تایمر شمارش معکوس

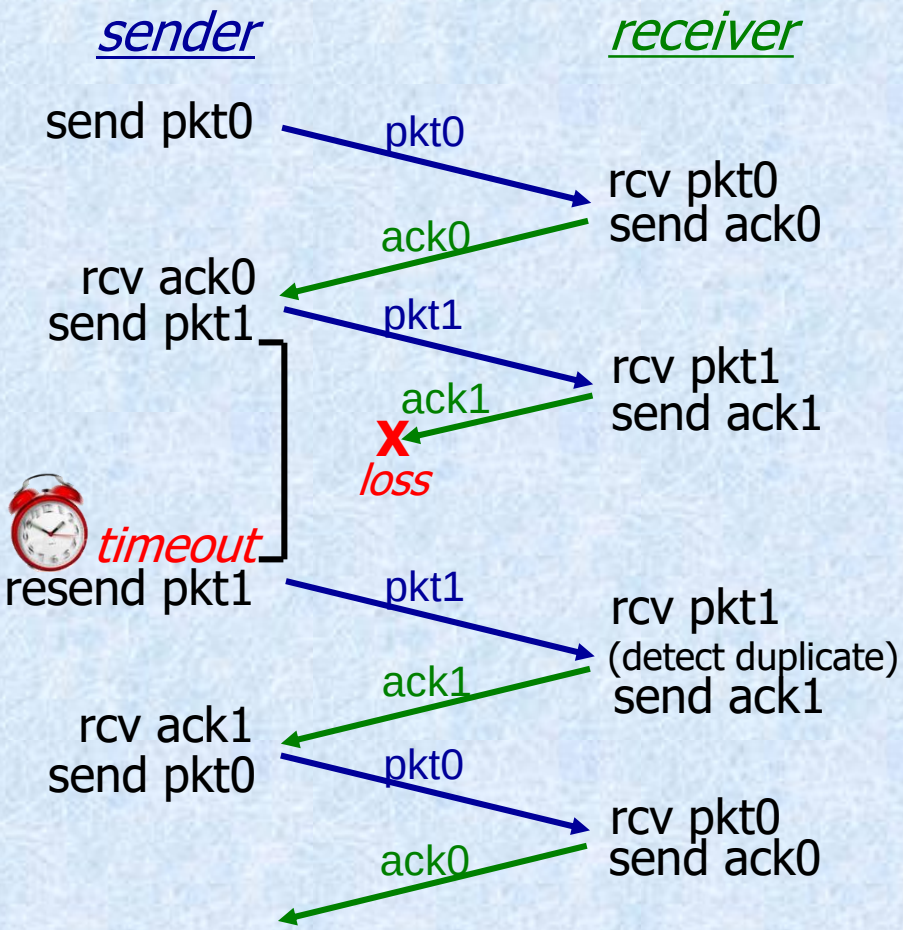
فرستنده :rdt3.0



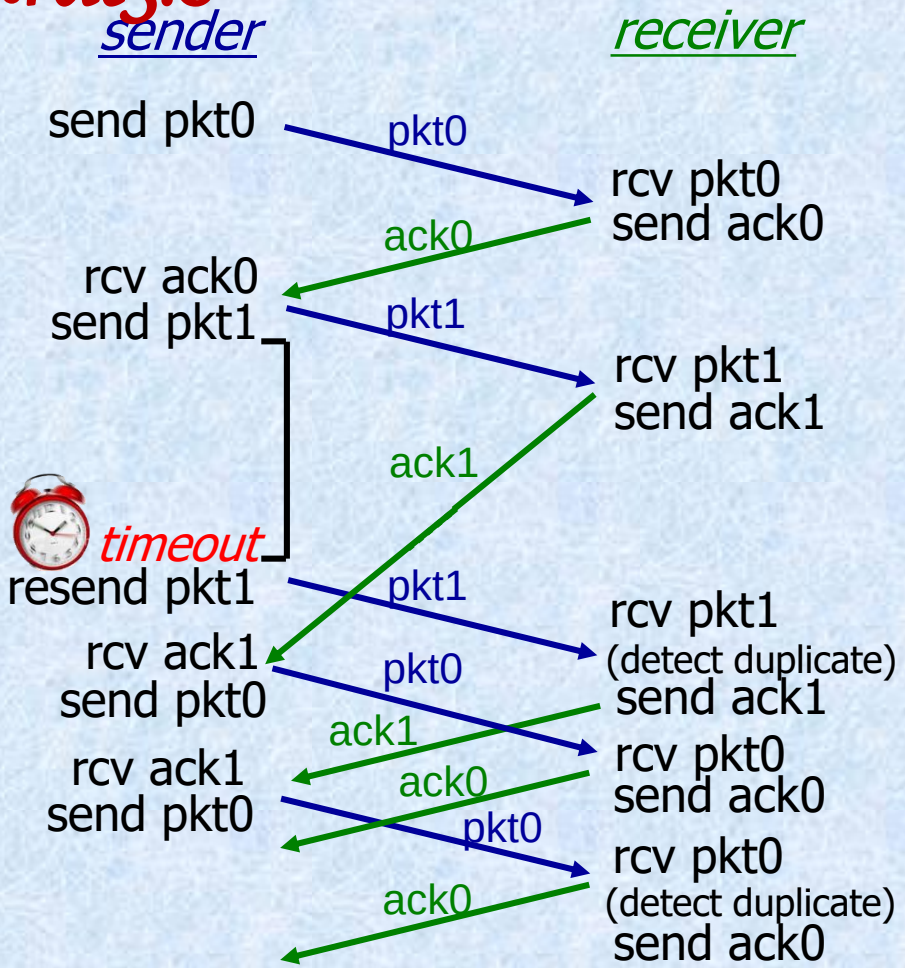
rdt3.0: تلف بسته



rdt3.0: تلف پاسخ



(c) ACK loss



(d) premature timeout/ delayed ACK

rdt3.0: کارایی

- مثال: برای پهنای باند **1 Gbps** با تاخیر انتشار یک طرفه **15 ms** و اندازه بسته های **8000 bit** داریم:

$$D_{trans} = \frac{L}{R} = \frac{8000\ bits}{10^9 bps} = 8\ \mu s$$

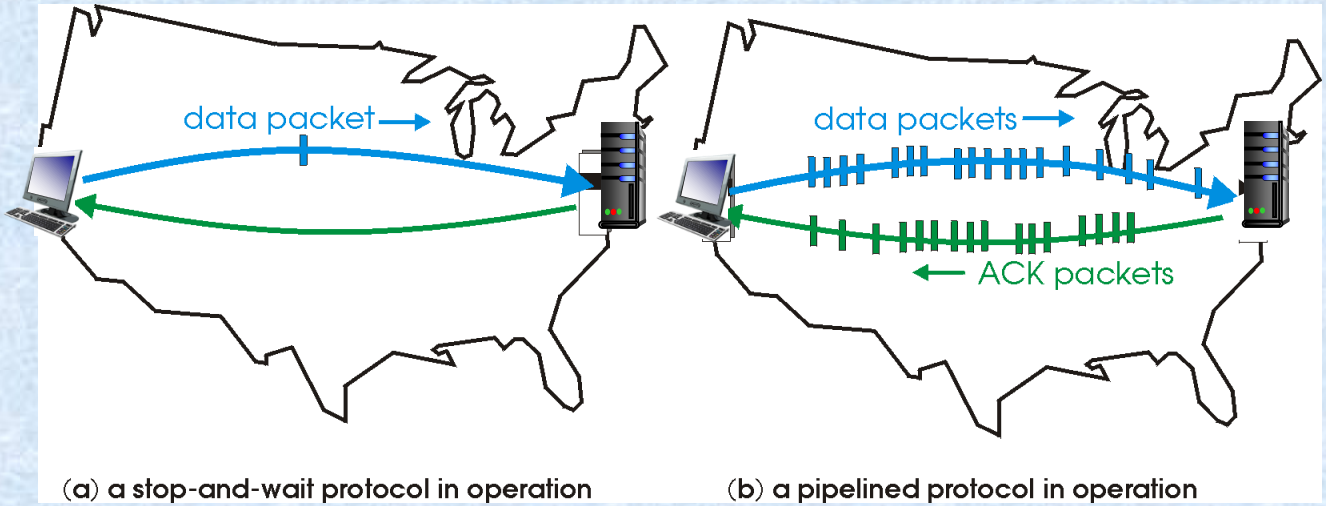
- U_{sender}: *utilization* – fraction of time sender busy sending

$$U_{sender} = \frac{L/R}{RTT + L/R} = \frac{0.008}{30.008} = 0.00027$$

- تاخیر انتشار بسیار پر رنگتر از تاخیر انتقال ← در هر ۳۰ میلی ثانیه تنها ۱ کیلوبایت ارسال می شود
- پروتکل دست و پا گیر!

پروتکل های خط لوله ای

- راه حل پیشنهادی: امکان ارسال بسته های بیشتر در مدت زمان انتظار برای پاسخ
  - افزایش تعداد شماره ترتیب ها
  - امکان ذخیره بسته ها در فرستنده و گیرنده



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

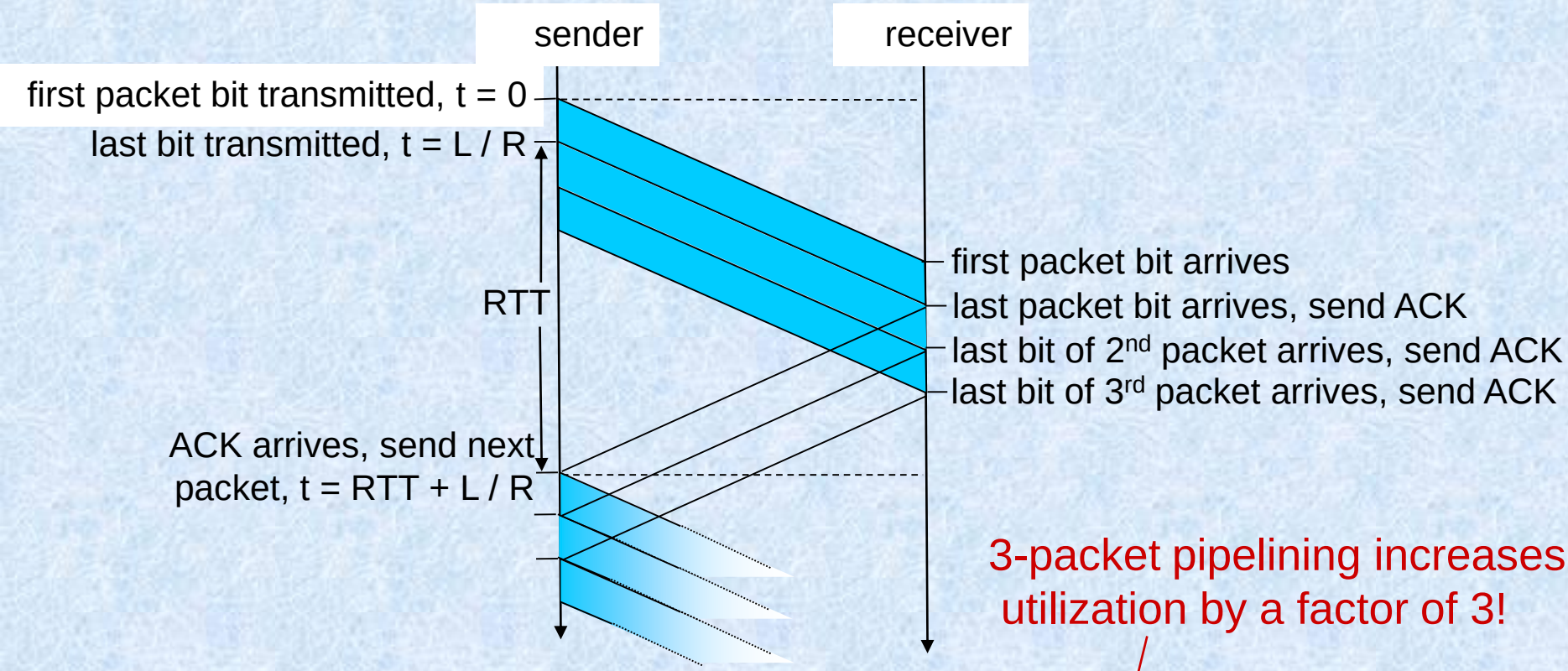
(b) a pipelined protocol in operation

- دو پروتکل پایه مبتنی بر خط لوله:

Go-Back-N

Selective Repeat

افزایش کارایی



3-packet pipelining increases utilization by a factor of 3!

$$U_{sender} = \frac{L/R}{RTT + L/R} = \frac{0.024}{30.008} = 0.00081$$

پروتکل های خط لوله ای

Go-Back-N

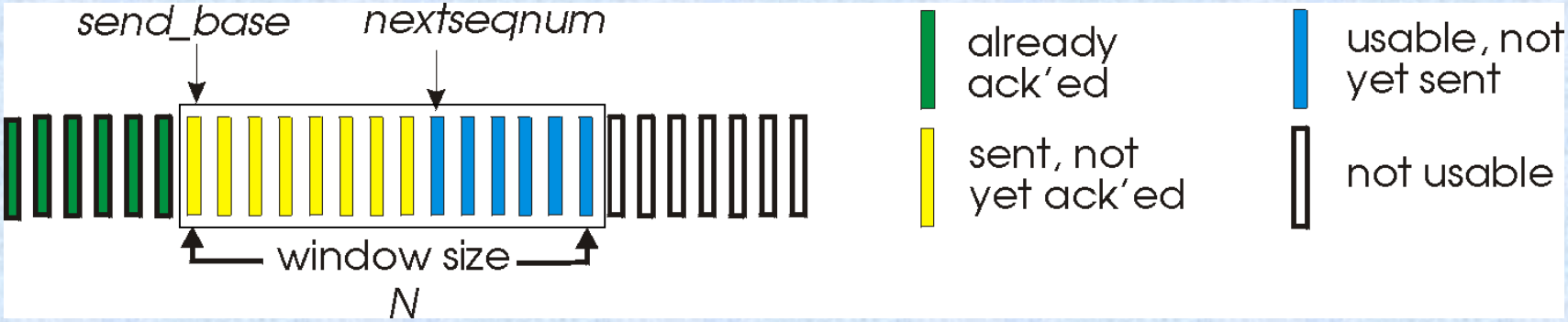
- امکان نگهداری (بافر) حداکثر **N** بسته در انتظار پاسخ
- گیرنده فقط پاسخ های تجمیع شده ارسال می کند
- ارسال پاسخ فقط در صورت وجود توالی در بسته ها
- فرستنده برای اولین بسته بدون پاسخ تایمر دارد
- در صورت انقضای تایمر ← ارسال مجدد تمامی بسته ها

Selective Repeat

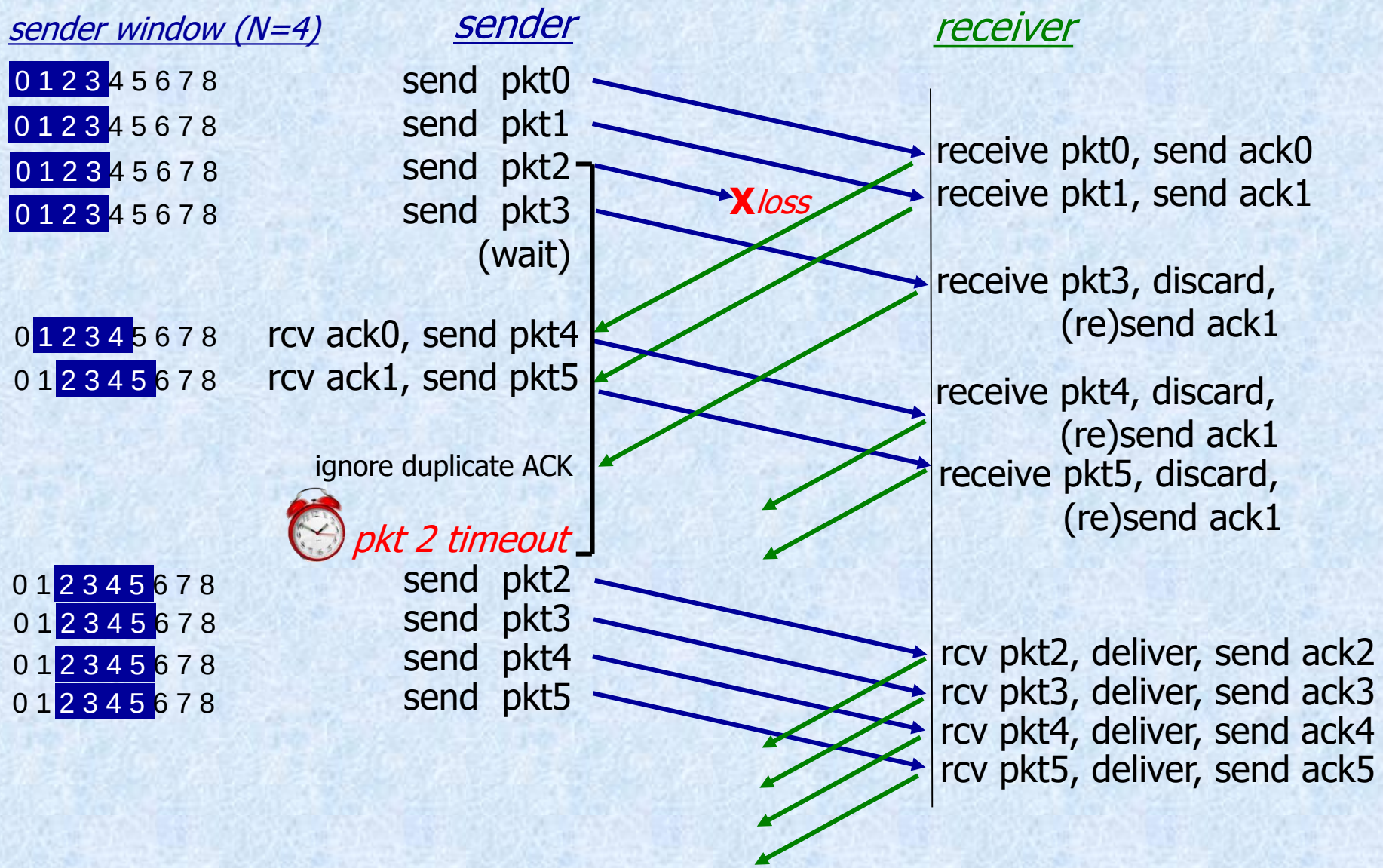
- امکان نگهداری حداکثر **N** بسته در انتظار پاسخ
- گیرنده برای هر بسته دریافتی پاسخ مجزا می فرستد
- فرستنده برای هر بسته منتظر پاسخ یک تایمر دارد
- ارسال مجدد صرفا بسته های با تایمر منقضی شده

GBN: پنجره ارسال

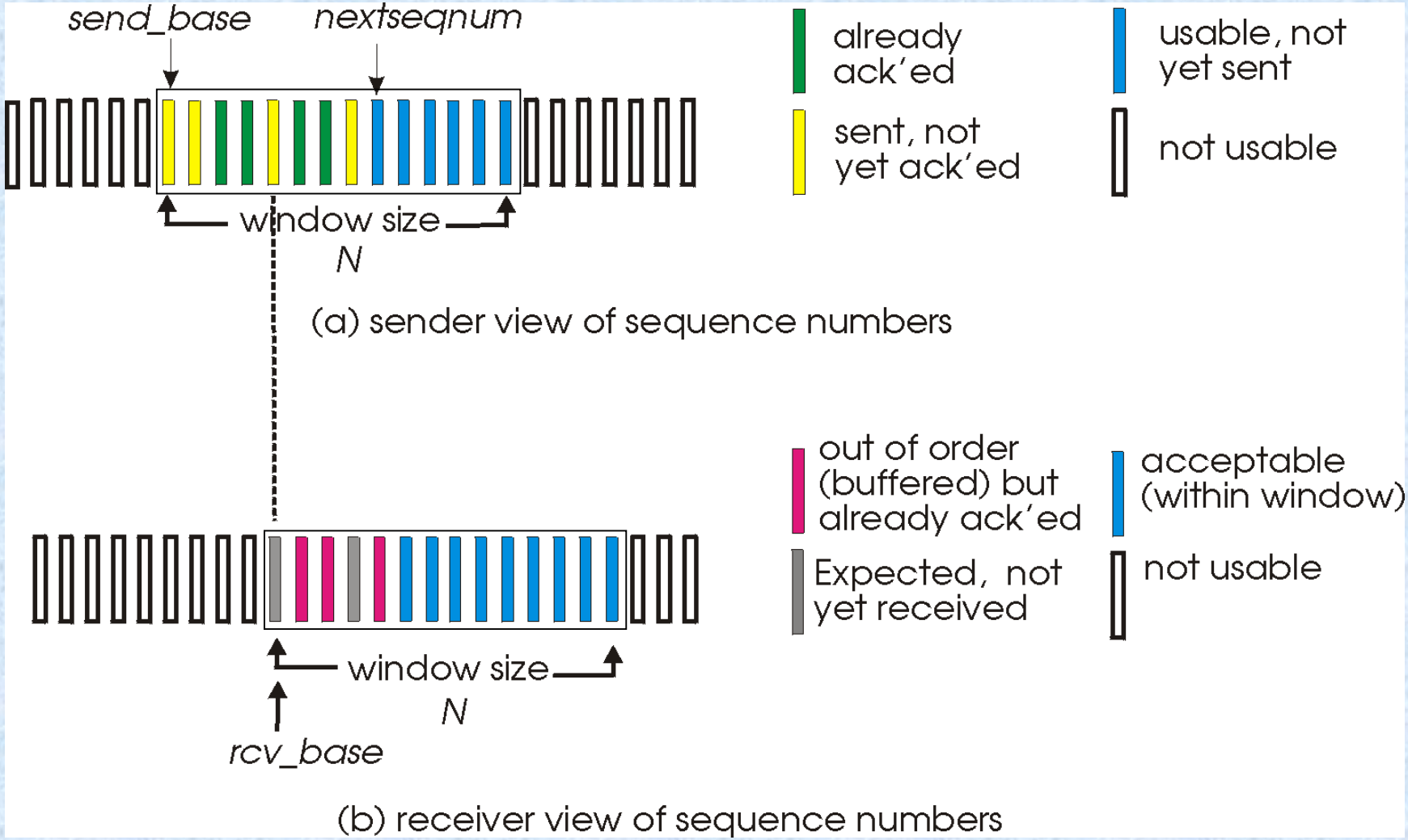
- تعداد شماره ترتیب برای K بیت: 2^k-1
- بسته های متوالی منتظر پاسخ
- شماره پاسخ باید در پنجره وجود داشته باشد
- شماره پاسخی بزرگتر از شماره مورد انتظار ← لغزش پنجره به تعداد شماره
- یک تایمر برای کل پنجره روی اولین بسته بدون پاسخ



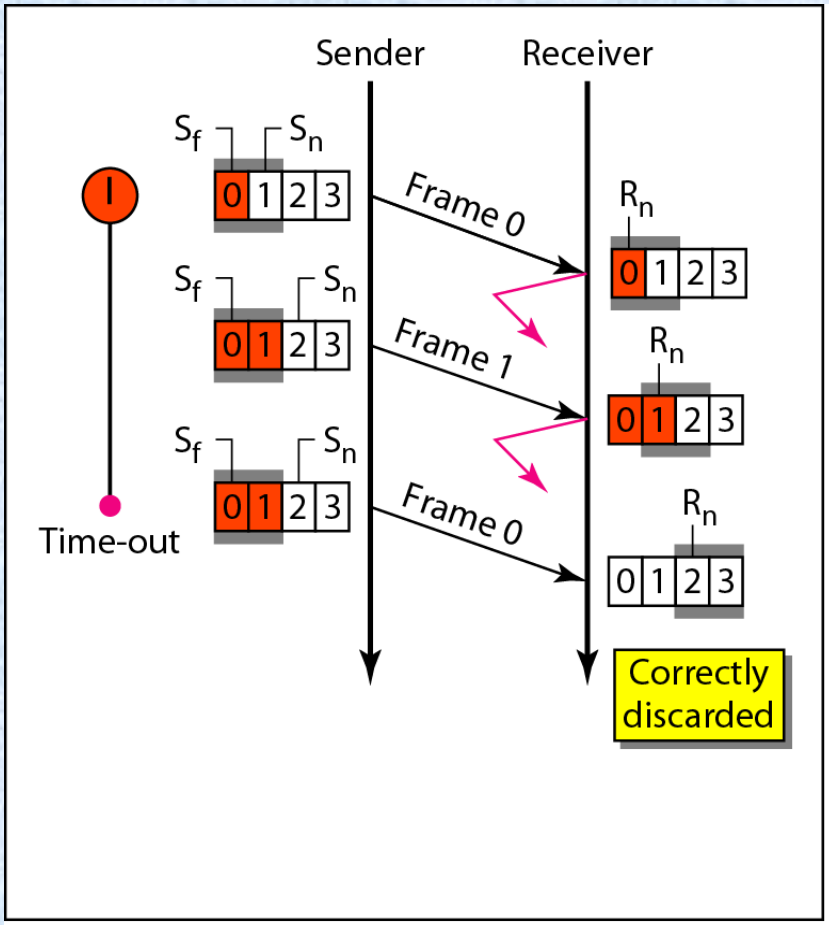
GBN


sender window (N=4)
sender
receiver
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
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
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

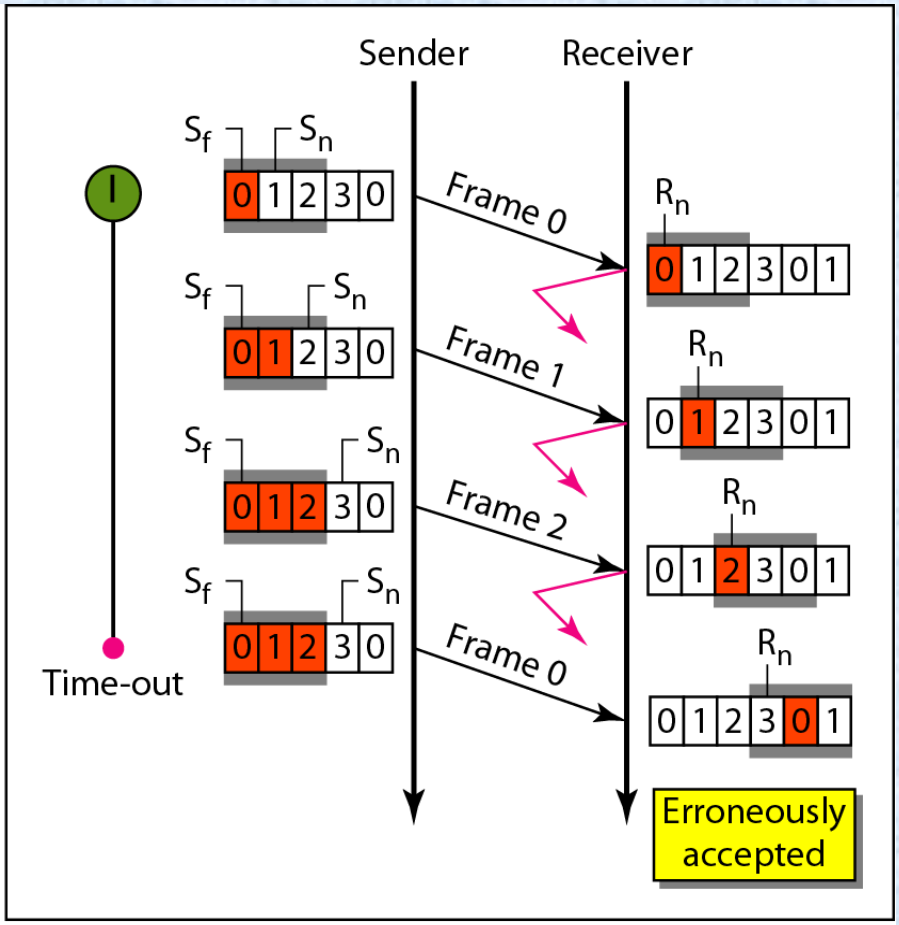
پنجره های ارسال و دریافت :Selective Repeat



اندازه پنجره: تکرار انتخابی (Selective Repeat)



a. Window size = 2^{m-1}



b. Window size > 2^{m-1}

خطای ارسال: تکرار انتخابی