شبکه‌های کامپیوتری

مسعود صبائی

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی امیرکبیر

لایه انتقال

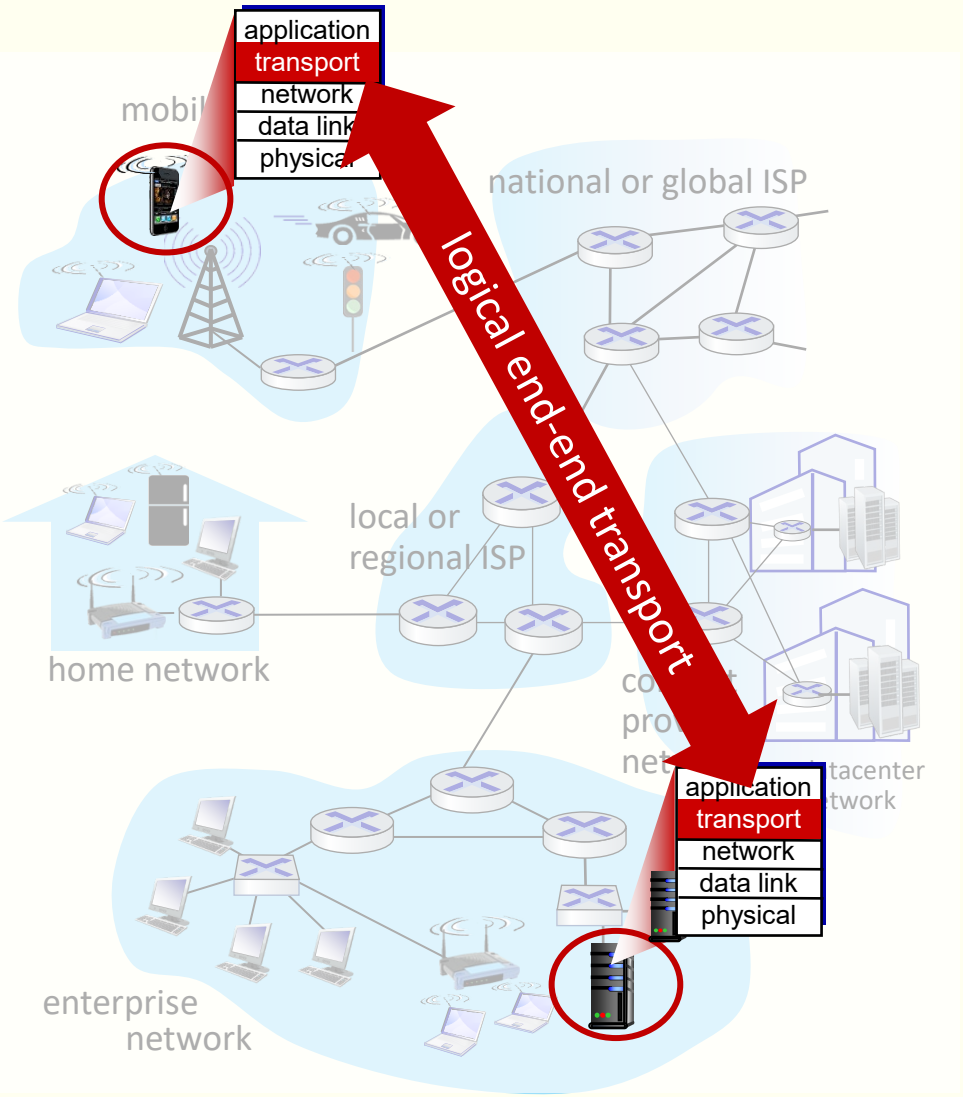
لایه انتقال

فهرست مطالب:

- معرفی سرویس‌های لایه انتقال
- سرویس بدون اتصال لایه انتقال و پروتکل UDP
- اصول انتقال مطمئن داده (پروتکل‌های کنترل خطای ARQ)
- سرویس اتصال‌گرا لایه انتقال و پروتکل TCP
- اصول کنترل ازدحام و کنترل ازدحام در پروتکل TCP

لایه انتقال

پروتکل‌ها و سرویس‌های لایه انتقال



- تبادل داده‌ها به صورت منطقی بین فرایندهای برنامه‌های کاربردی درحال اجرا روی میزبان‌های مختلف
- اقدامات پروتکل‌های لایه انتقال در سیستم‌های انتهایی:
  - فرستنده: شکستن پیام‌های لایه کاربرد درون سگمنت‌ها و دادن آن‌ها به لایه شبکه
  - گیرنده: بازسازی سگمنت‌ها درون پیام‌ها و دادن آن‌ها به لایه کاربرد
- دو پروتکل لایه انتقال دردسترس برای برنامه‌هی کاربردی
  - پروتکل TCP و پروتکل UDP

لایه انتقال

مقایسه پروتکل‌ها و سرویس‌های لایه انتقال و لایه شبکه

قیاس با اهالی یک خانه

۱۲ کودک در خانه آنا برای ۱۲ کودک در خانه بیل نامه ارسال می‌کنند.

- میزبان‌ها = خانه‌ها
- فرایندها = کودکان
- پیام‌های برنامه‌های کاربردی = نامه‌های درون پاکت

لایه انتقال

مقایسه پروتکل‌ها و سرویس‌های لایه انتقال و لایه شبکه

- لایه شبکه: تبادل داده‌ها به صورت منطقی بین میزبان‌ها
- لایه انتقال: تبادل داده‌ها به صورت منطقی بین فرایندها
- متکی به سرویس لایه شبکه و بهبود آن

قیاس با اهالی یک خانه

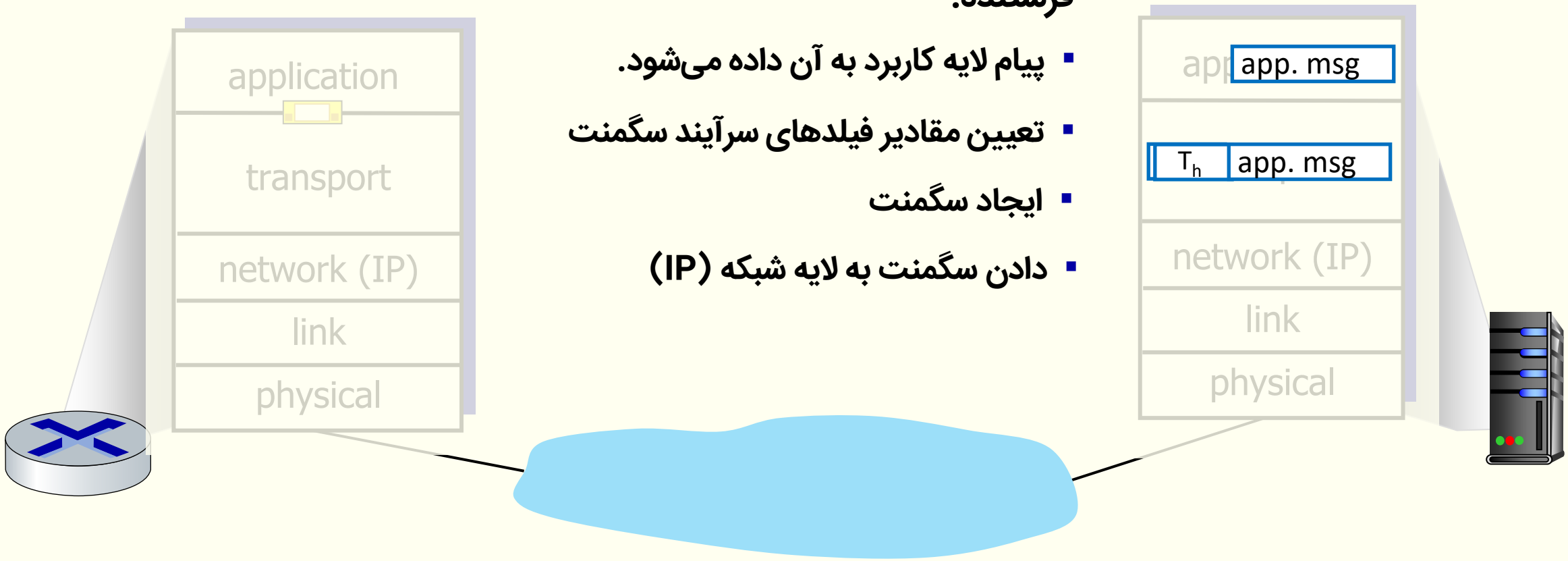
۱۲ کودک در خانه آنا برای ۱۲ کودک در خانه بیل نامه ارسال می‌کنند.

- میزبان‌ها = خانه‌ها
- فرایندها = کودکان
- پیام‌های برنامه‌های کاربردی = نامه‌های درون پاکت

اقدامات لایه انتقال - فرستنده

فرستنده:

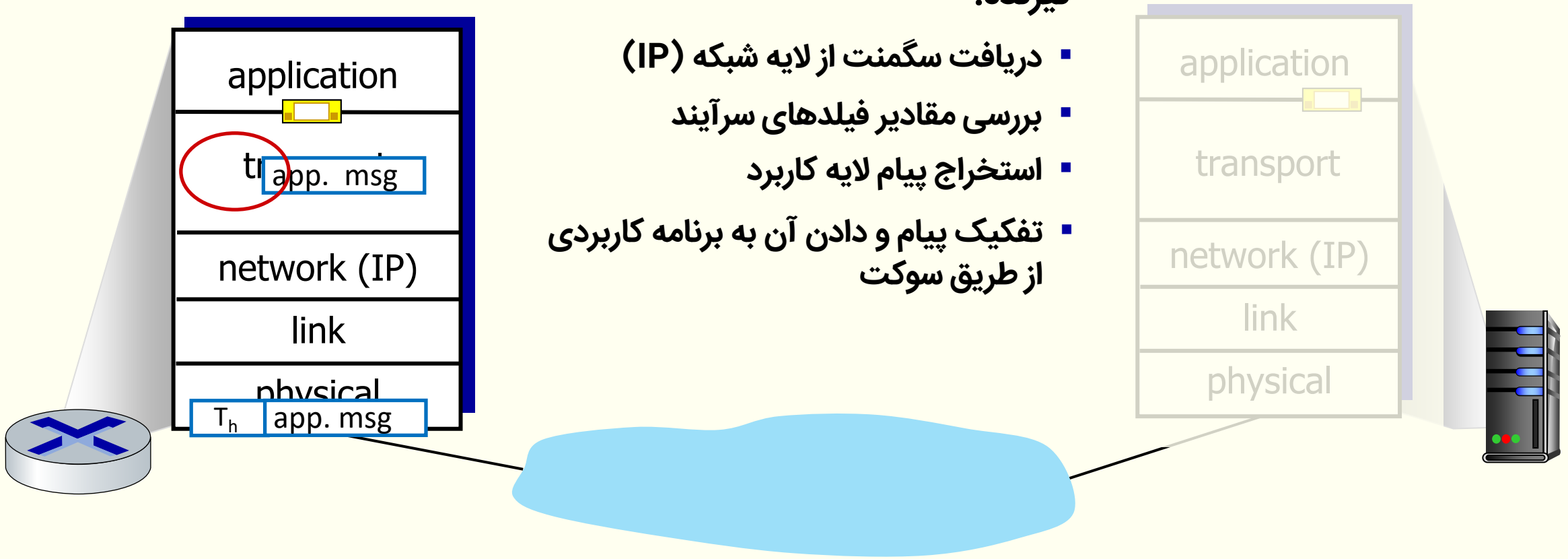
- **پیام لایه کاربرد به آن داده می‌شود.**
- **تعیین مقادیر فیلدهای سرآیند سگمنت**
- **ایجاد سگمنت**
- **دادن سگمنت به لایه شبکه (IP)**



اقدامات لایه انتقال – گیرنده

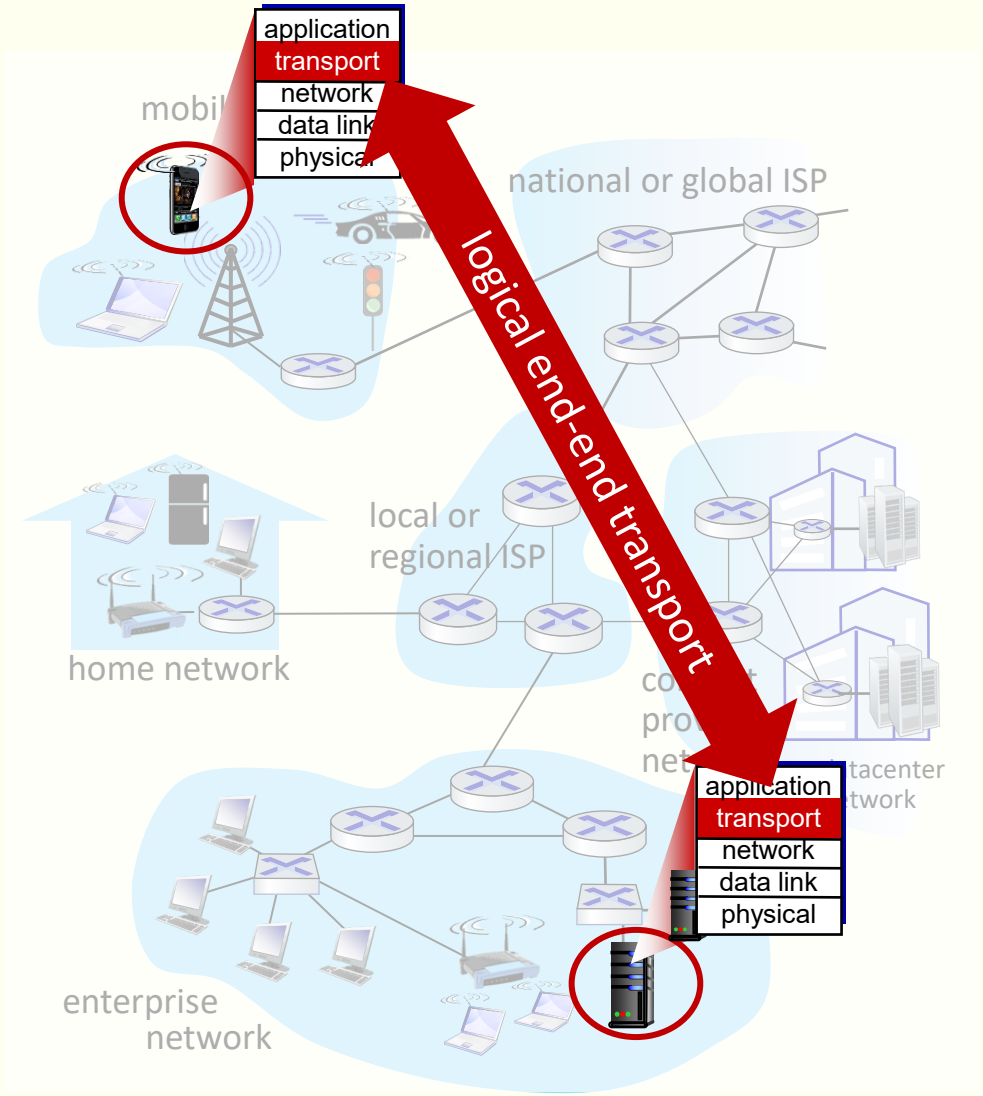
گیرنده:

- **دریافت سگمنت از لایه شبکه (IP)**
- **بررسی مقادیر فیلدهای سرآیند**
- **استخراج پیام لایه کاربرد**
- **تفکیک پیام و دادن آن به برنامه کاربردی از طریق سوکت**



لایه انتقال

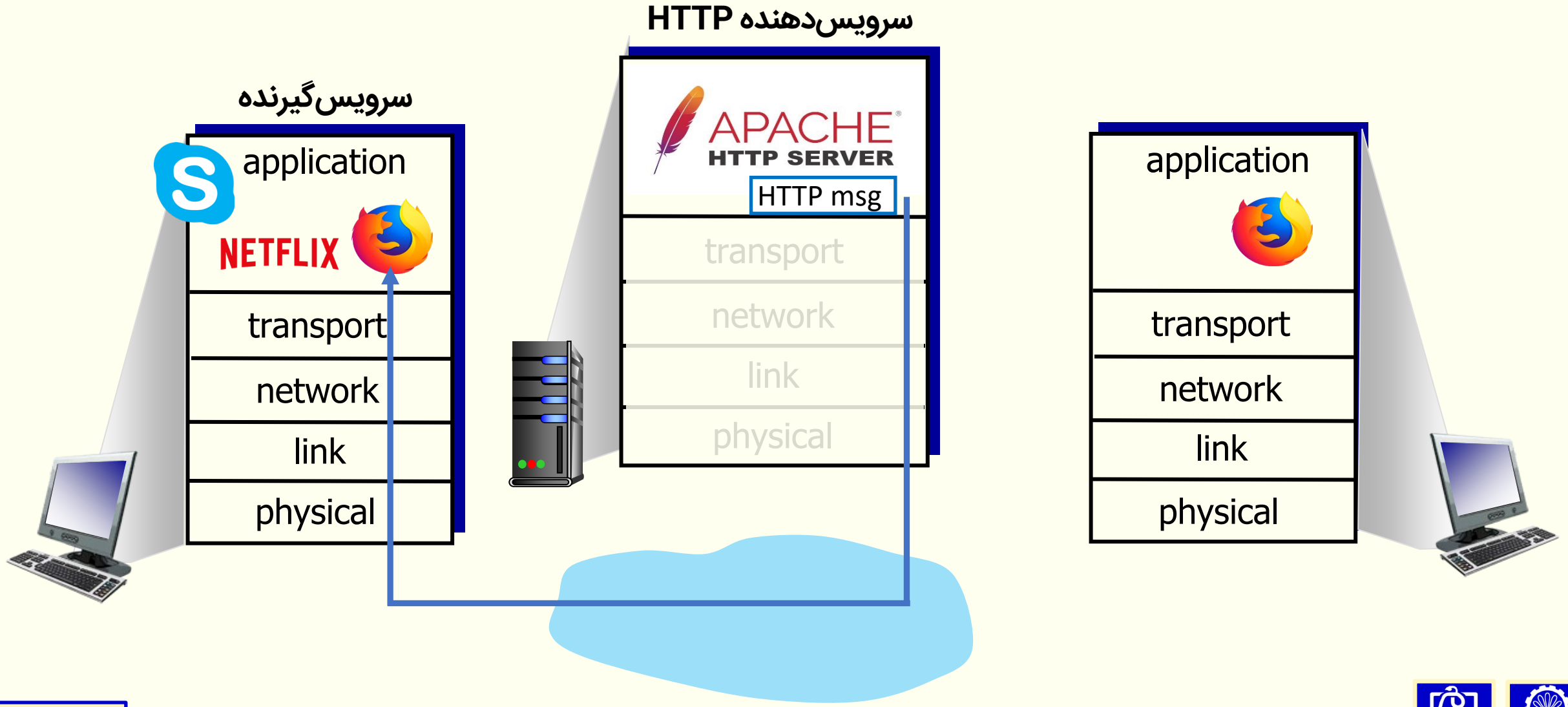
دو پروتکل اصلی لایه انتقال شبکه اینترنت



- TCP: پروتکل کنترل ارسال (Transmission Control Protocol)
  - مطمئن (تحویل بدون خطا و به ترتیب داده‌ها)
  - کنترل ازدحام
  - کنترل جریان
  - برقراری اتصال
- UDP: پروتکل دیتاگرام کاربر (User Datagram Protocol)
  - غیرمطمئن (عدم تضمین تحویل و تحویل به ترتیب داده‌ها)
  - توسعه بدون پیرایه سرویس بیشترین تلاش IP
- سرویس‌هایی که هیچ‌کدام تأمین نمی‌کنند.
  - تضمین تأخیر
  - تضمین پهنای‌باند

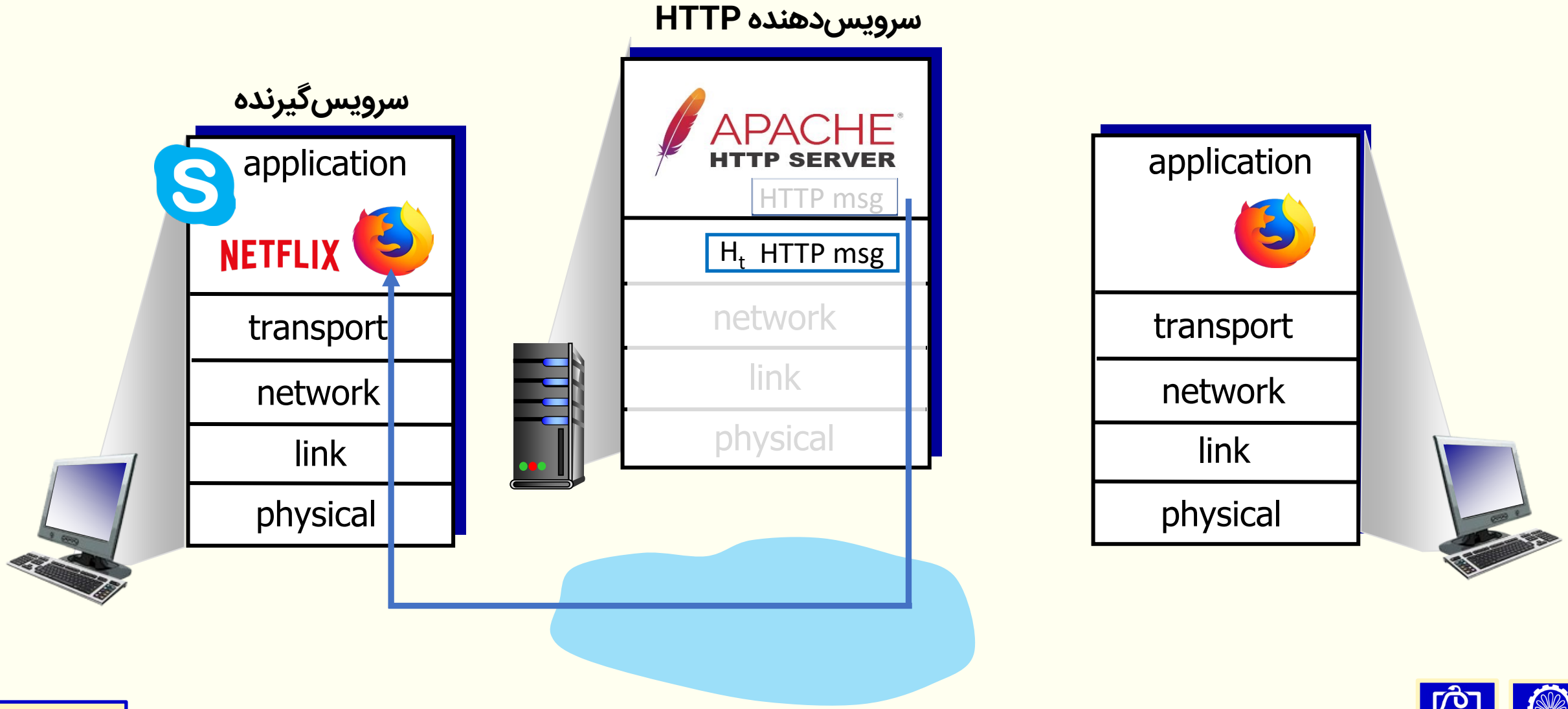
لایه انتقال

مالتی‌پلکسینگ و دی‌مالتی‌پلکسینگ



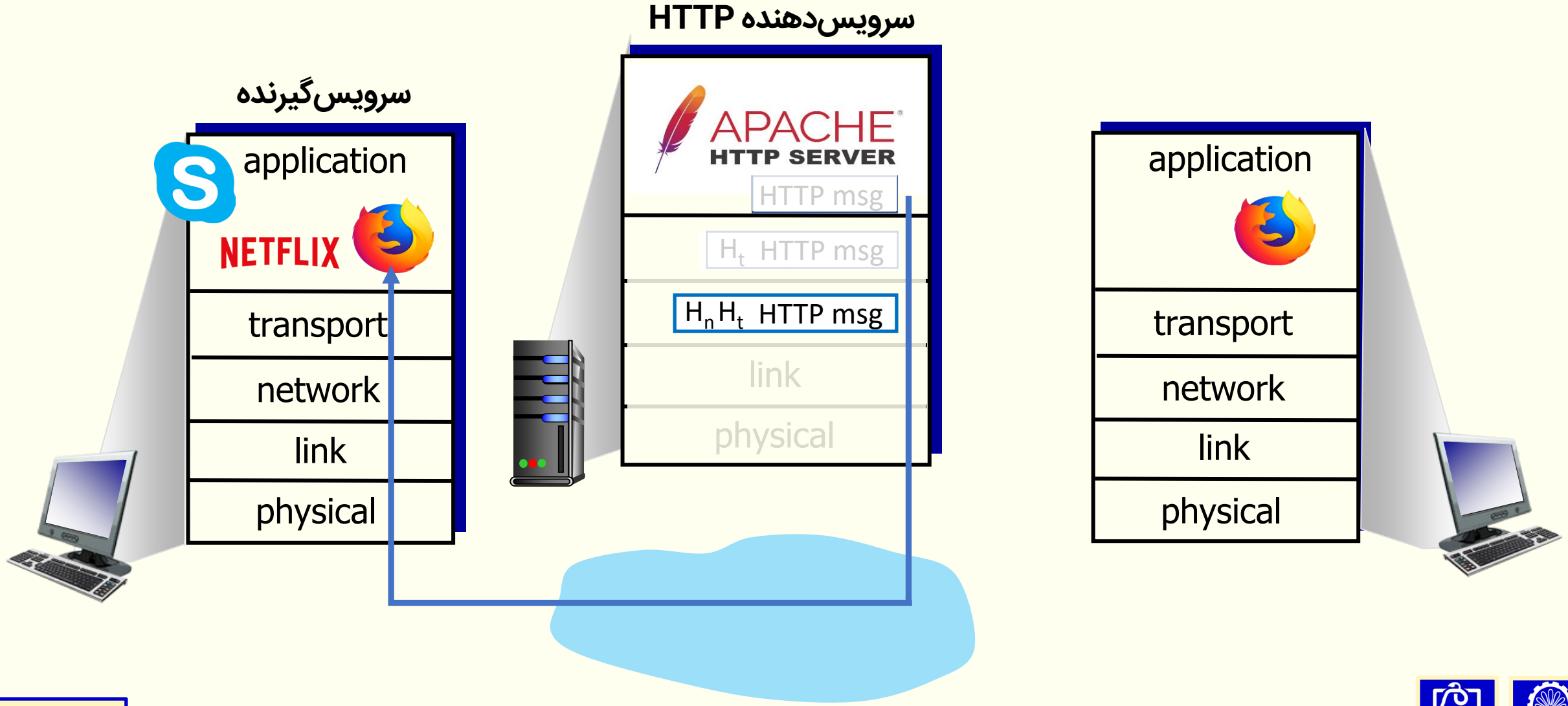
لایه انتقال

مالتی‌پلکسینگ و دی‌مالتی‌پلکسینگ



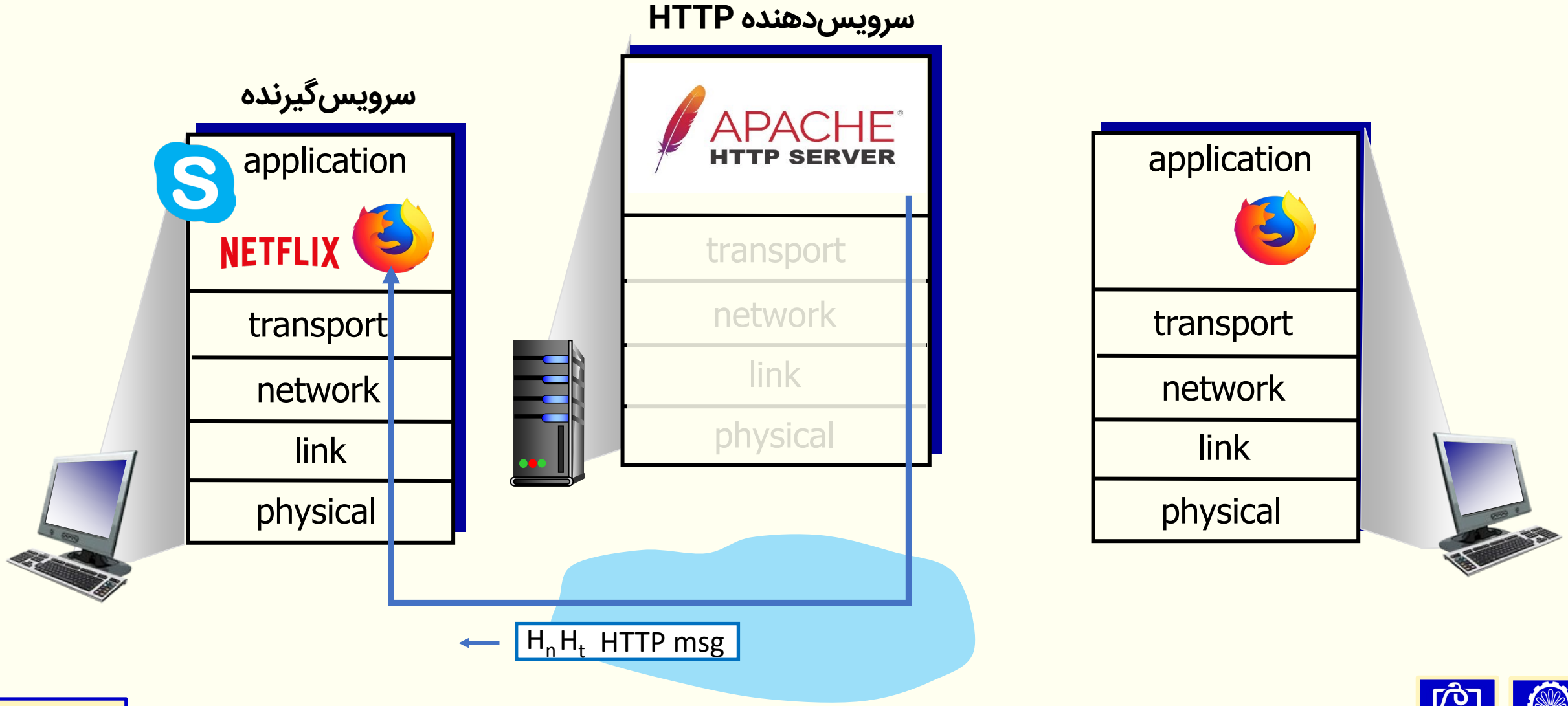
لایه انتقال

مالتی‌پلکسینگ و دی‌مالتی‌پلکسینگ



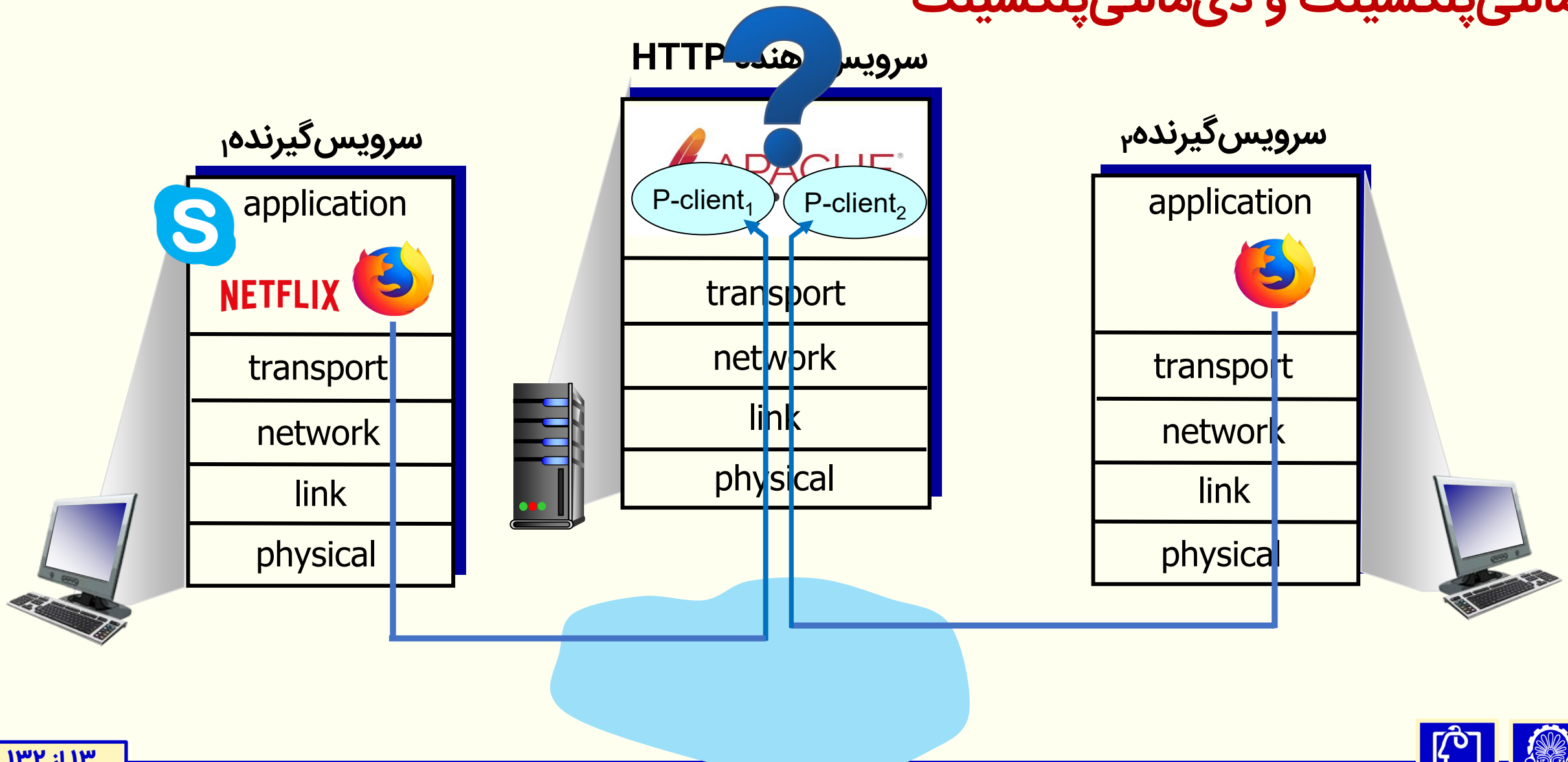
لایه انتقال

مالتی‌پلکسینگ و دی‌مالتی‌پلکسینگ



لایه انتقال

مالتی‌پلکسینگ و دی‌مالتی‌پلکسینگ



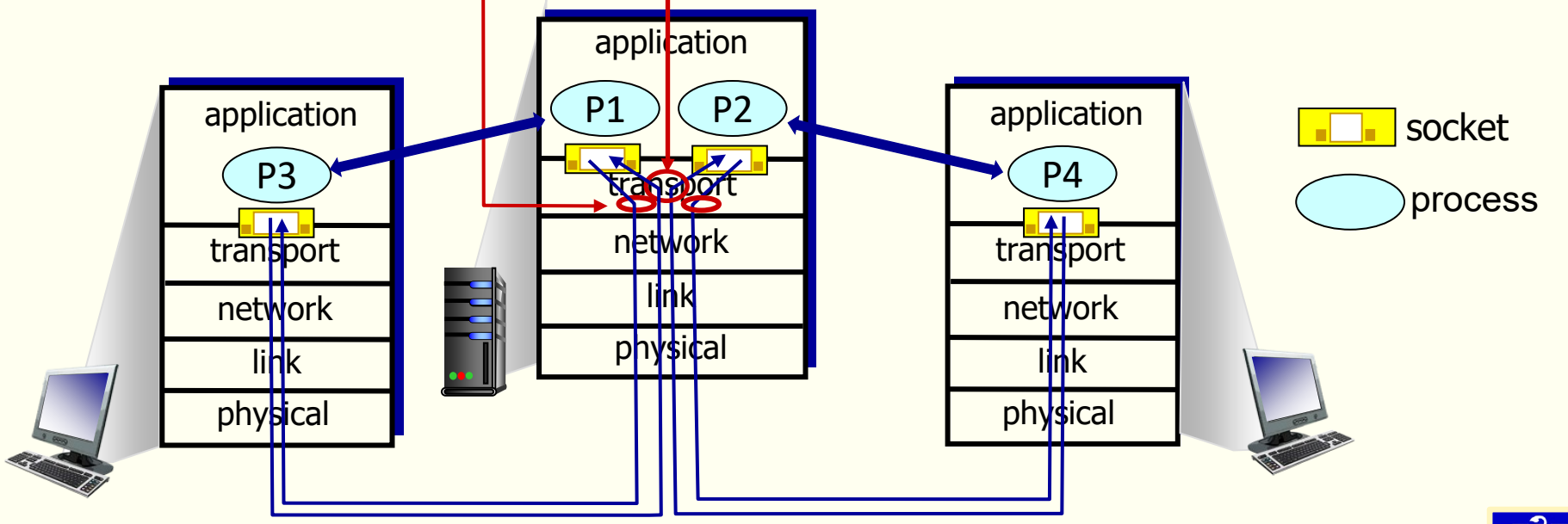
مالتی‌پلکسینگ و دی‌مالتی‌پلکسینگ

مالتی‌پلکسینگ در فرستنده:

رسیدگی کردن به داده‌های چندین سوکت و اضافه کردن سرآیند لایه انتقال
(بعداً برای دی‌مالتی‌پلکسینگ استفاده می‌شود)

دی‌مالتی‌پلکسینگ در گیرنده:

استفاده از اطلاعات سرآیند برای تحویل سگمنت‌های دریافتی به سوکت درست



نحوه انجام دی‌مالتی‌پلکسینگ

فرمت سگمنت‌های UDP و TCP

- دریافت بسته IP توسط میزبان:
  - هر بسته IP آدرس مبدأ و مقصد را در سرآیند خود دارد.
  - هر بسته IP یک سگمنت لایه انتقال (UDP یا TCP) را حمل می‌کند.
  - هر سگمنت (UDP یا TCP) شماره پورت‌های مبدأ و مقصد را در سرآیند خود دارد.
- میزبان برای دادن سگمنت به سوکت مناسب از آدرس‌های IP و شماره پورت‌ها استفاده می‌کند.

لایه انتقال

دی‌مالتی‌پلکسینگ بدون اتصال (UDP)

یادآوری:

- در زمان ایجاد سوکت باید شماره پورت میزبان محلی مشخص شود.

```
DatagramSocket mySocket1
    = new DatagramSocket(12534);
```

- در زمان ایجاد یک دیتاگرام و ارسال آن درون سوکت UDP باید آدرس IP مقصد و شماره پورت مقصد مشخص شود.

- در زمان دریافت یک سگمنت UDP توسط میزبان گیرنده:
  - شماره پورت مقصد در سگمنت دریافتی بررسی می‌شود.
  - سگمنت UDP به سوکتی که شماره پورت مشخص شده، داده می‌شود.

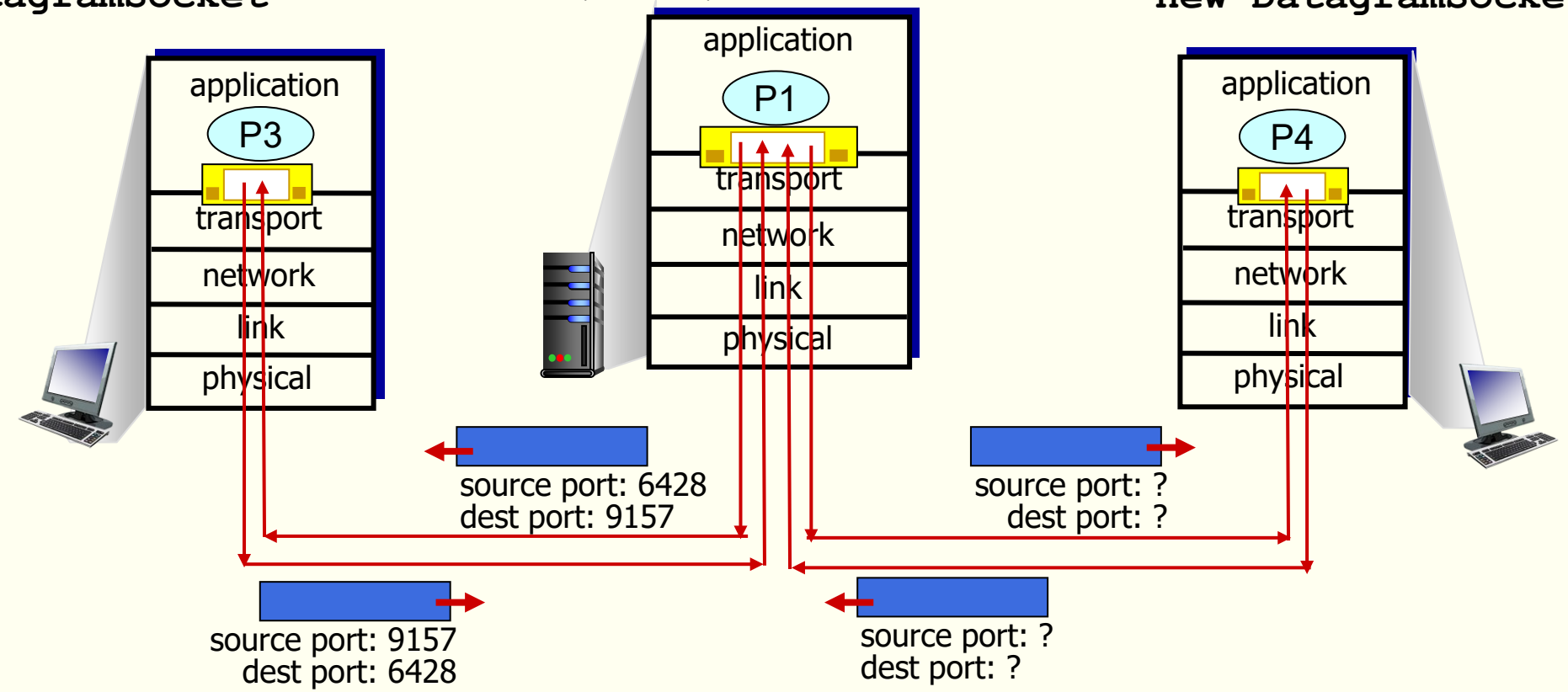
سگمنت‌های دریافتی با آدرس IP و شماره پورت مقصد یکسان ولی با آدرس مبدأ یا شماره پورت مبدأ متفاوت به یک سوکت یکسان در میزبان گیرنده داده می‌شوند.

دی‌مالتی‌پلکسینگ بدون اتصال (UDP) : مثال

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

```
DatagramSocket
 serverSocket = new
 DatagramSocket
 (6428);
```

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

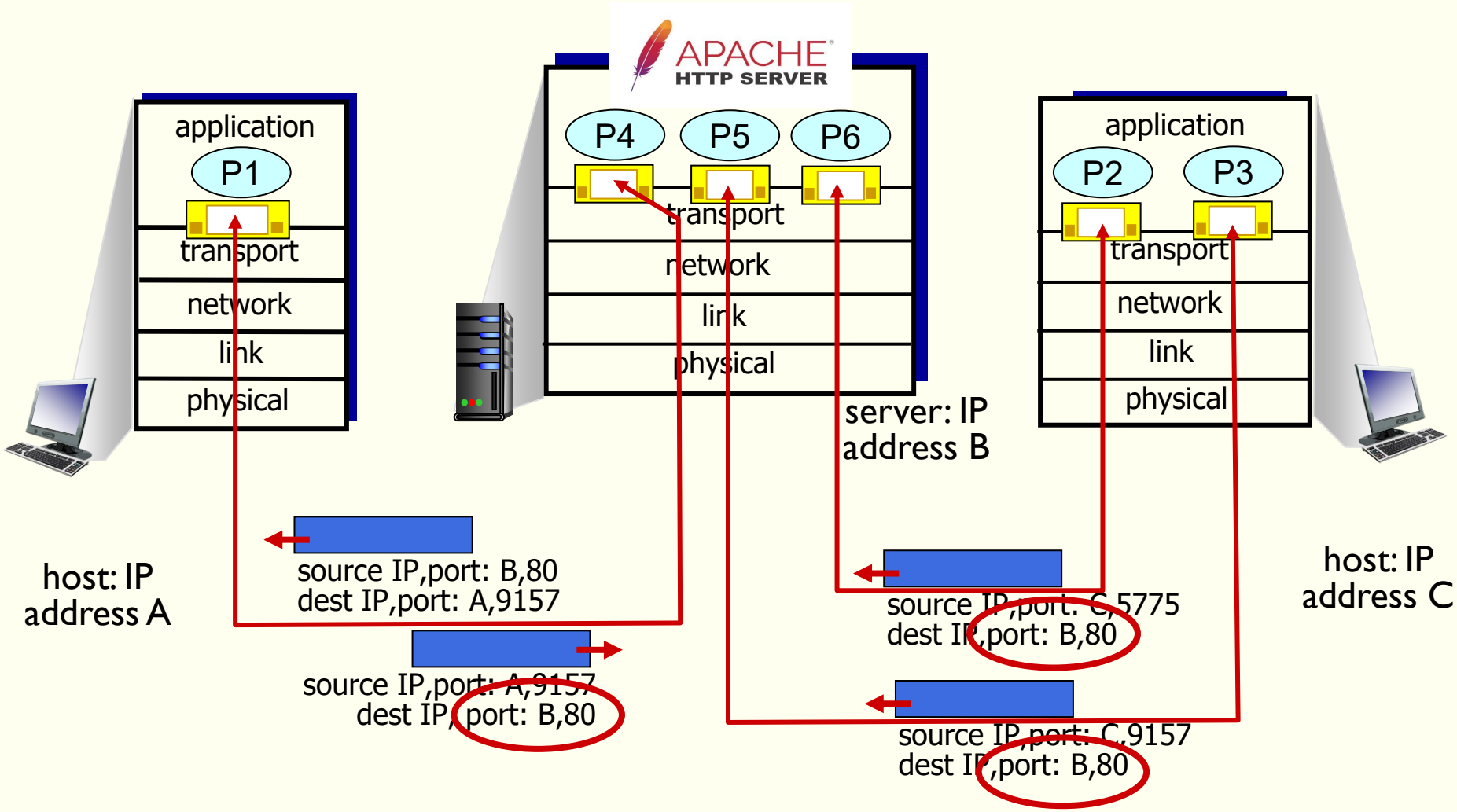


دی‌مالتی‌پلکسینگ اتصال‌گرا (TCP)

- سکوت‌های TCP با یک شناسه چهار جزئی (4-tuple) شناسایی می‌شوند:
  - آدرس IP مبدأ
  - شماره پورت مبدأ
  - آدرس IP مقصد
  - شماره پورت مقصد
- دی‌مالتی‌پلکس‌کردن: گیرنده از تمام چهار مقدار (چهار جزء) برای دادن سگمنت‌ها به سوکت مناسب استفاده می‌کند.

- سرویس‌دهنده‌ها ممکن است از تعدادی سوکت همزمان TCP پشتیبانی کند:
  - هر سوکت با شناسه چهار جزئی خود شناسایی می‌شود.
  - هر سوکت به یک سرویس‌دهنده متفاوت مرتبط است.

دی‌مالتی‌پلکسینگ اتصال‌گرا (TCP): مثال



سه سگمنت با آدرس IP مقصد B و شماره پورت مقصد ۸۰ به سوکت‌های مختلف دی‌مالتی‌پلکس می‌شوند.

لایه انتقال

خلاصه (مالتی‌پلکسینگ و دی‌مالتی‌پلکسینگ)

- مالتی‌پلکسینگ و دی‌مالتی‌پلکسینگ بر مبنای مقادیر فیلدهای سرآیند سگمنت سرویس لایه انتقال (UDP و TCP) و بسته IP انجام می‌شود.
- در پروتکل UDP برای دی‌مالتی‌پلکسینگ فقط از شماره پورت مقصد استفاده می‌شود.
- در پروتکل TCP برای دی‌مالتی‌پلکسینگ از چهار جزء آدرس‌های IP و شماره پورت‌های مبدأ و مقصد استفاده می‌شود.
- مالتی‌پلکسینگ و دی‌مالتی‌پلکسینگ در لایه‌های دیگر شبکه نیز انجام می‌شود.

لایه انتقال

پروتکل دیتاگرام کاربر (UDP: User Datagram Protocol)

- پروتکل "بدون زاوئد" لایه انتقال شبکه اینترنت
- سرویس بیشترین تلاش:
  - احتمال از دست دادن سگمنت‌ها وجود دارد.
  - احتمال تحویل خارج از ترتیب سگمنت‌ها به برنامه کاربردی
- بدون اتصال:
  - بدون دست دادن بین فرستنده و گیرنده UDP
  - هر سگمنت UDP مستقل از بقیه رسیدگی می‌شود.

چرا UDP وجود دارد؟

- بدون برقراری اتصال (برقراری اتصال یک تأخیر اولیه به اندازه RTT ایجاد می‌کند)
- ساده: بدون وضعیت (state) اتصال در فرستنده و گیرنده
- اندازه کوچک سرآیند: سربار پروتکلی کم
- بدون کنترل ازدحام:
  - UDP می‌تواند با سرعت دلخواه ارسال را انجام دهد.
  - در مواجهه با ازدحام کار می‌کند.

لایه انتقال

پروتکل دیتاگرام کاربر (UDP: User Datagram Protocol)

- استفاده‌های UDP:
  - برنامه‌های کاربردی جریان‌سازی چندرسانه‌ای (تحمل‌پذیر از دست‌دادن و حساس به رعایت زمانبندی)
  - DNS
  - SNMP
  - HTTP/3
- در صورت نیاز به انتقال مطمئن روی پروتکل UDP (به عنوان مثال HTTP/۳) :
  - ایجاد قابلیت اطمینان در لایه کاربرد
  - انجام کنترل ازدحام در لایه کاربرد

پروتکل دیتاگرام کاربر (UDP: User Datagram Protocol) [RFC 798]

INTERNET STANDARD

RFC 768

J. Postel
ISI
28 August 1980

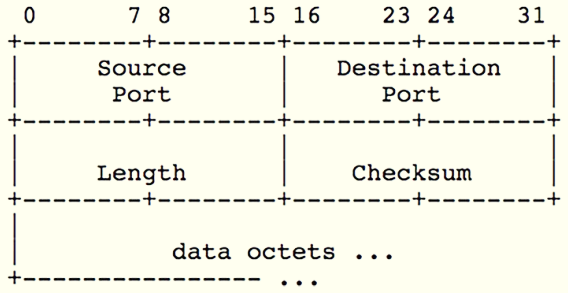
User Datagram Protocol

Introduction

This User Datagram Protocol (UDP) is defined to make available a datagram mode of packet-switched computer communication in the environment of an interconnected set of computer networks. This protocol assumes that the Internet Protocol (IP) [1] is used as the underlying protocol.

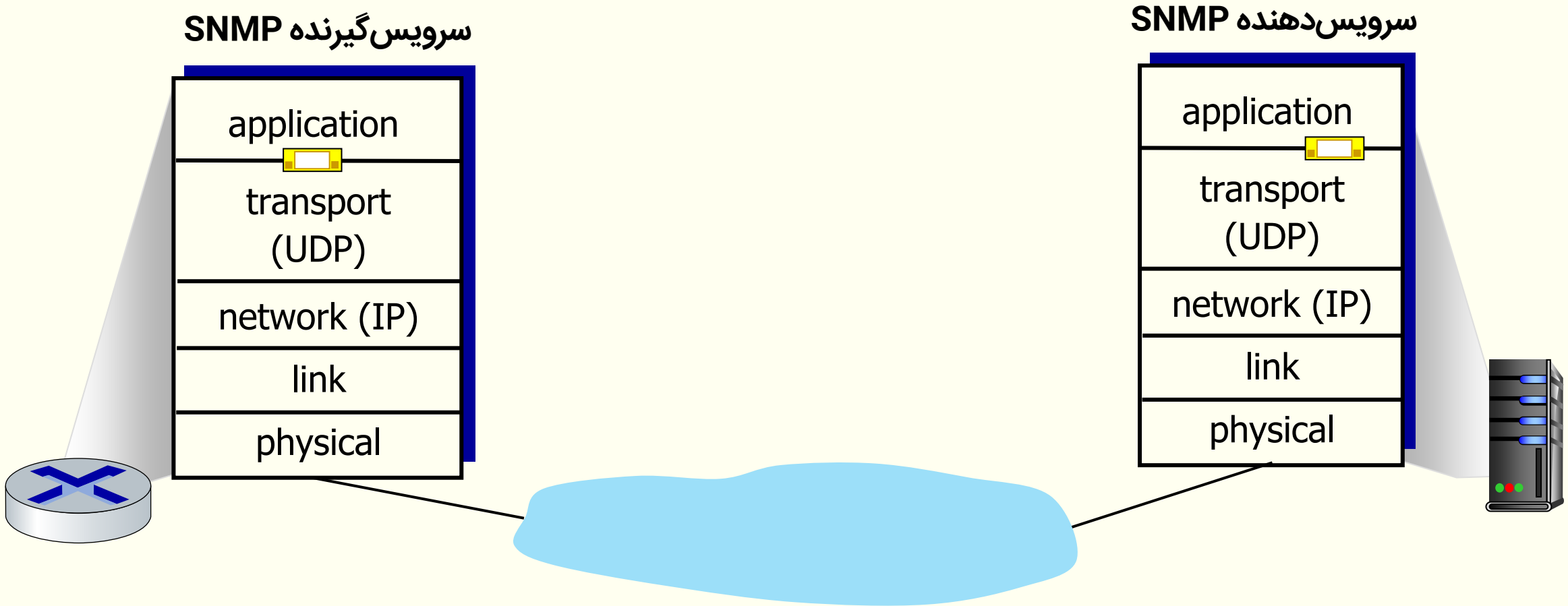
This protocol provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction oriented, and delivery and duplicate protection are not guaranteed. Applications requiring ordered reliable delivery of streams of data should use the Transmission Control Protocol (TCP) [2].

Format



لایه انتقال

UDP: اقدامات لایه انتقال

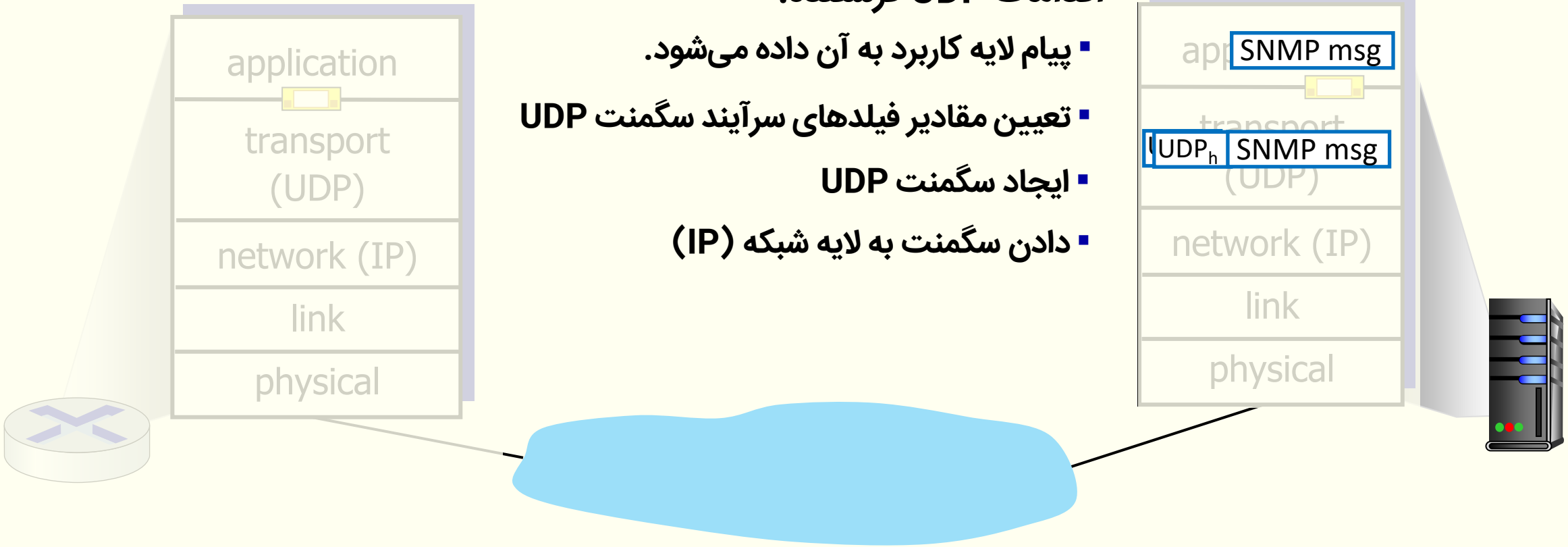


لایه انتقال

UDP: اقدامات لایه انتقال

سرویس‌دهنده SNMP

سرویس‌گیرنده SNMP



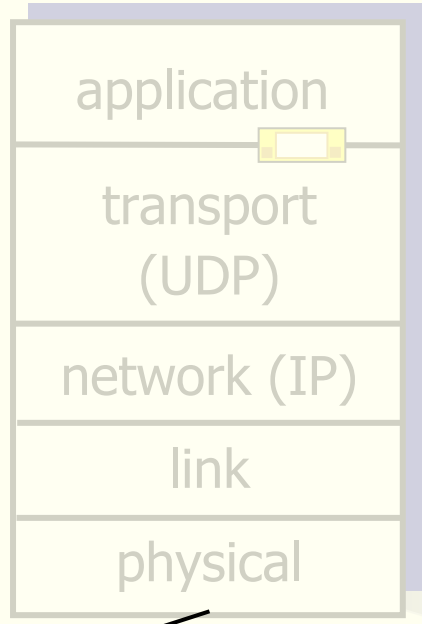
اقدامات UDP فرستنده:

- پیام لایه کاربرد به آن داده می‌شود.
- تعیین مقادیر فیلدهای سرآیند سگمنت UDP
- ایجاد سگمنت UDP
- دادن سگمنت به لایه شبکه (IP)

لایه انتقال

UDP: اقدامات لایه انتقال

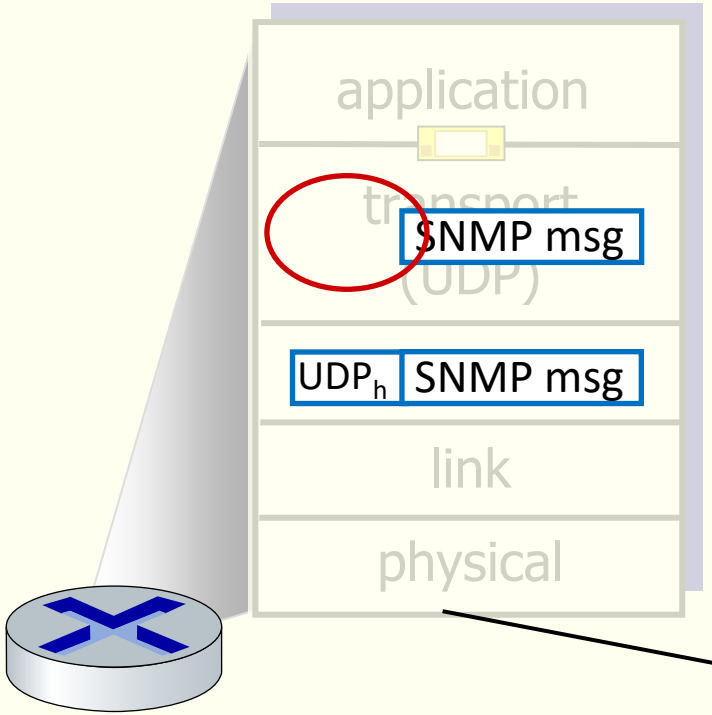
سرویس‌دهنده SNMP



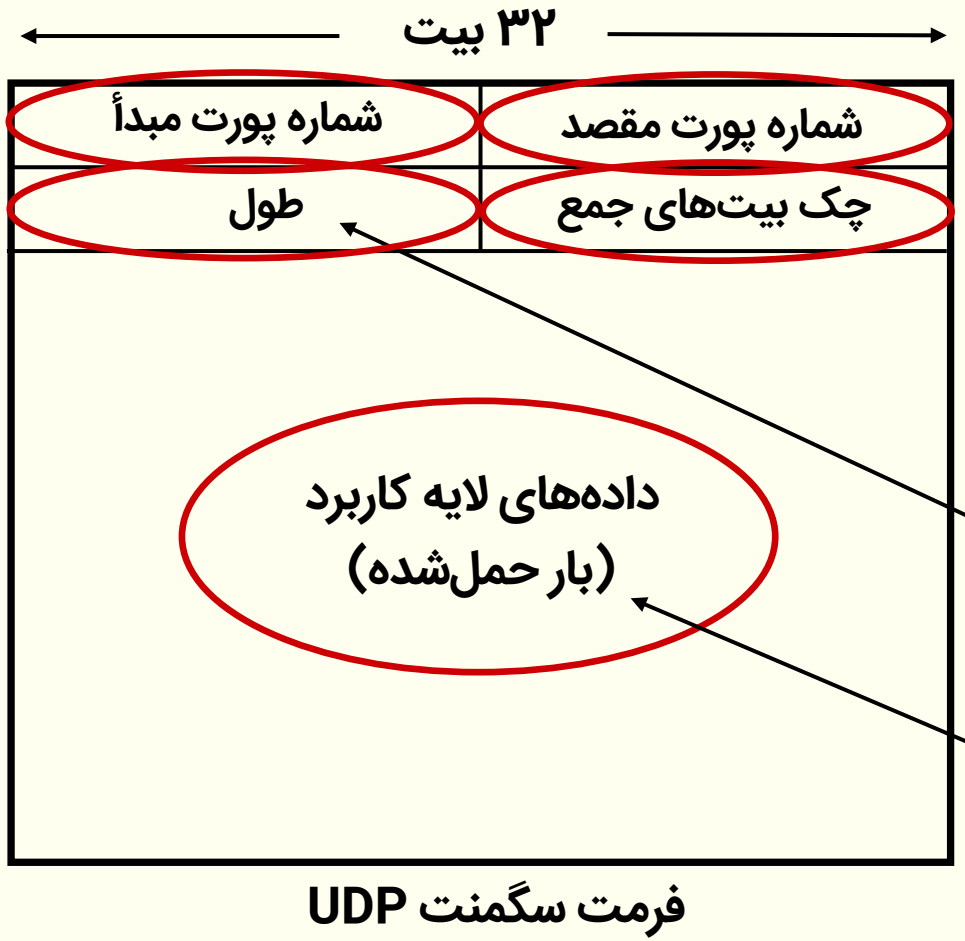
اقدامات گیرنده UDP:

- دریافت سگمنت از لایه شبکه (IP)
- بررسی مقدار چک بیت‌های جمع (checksum) سرآیند UDP
- استخراج پیام لایه کاربرد
- تفکیک پیام و دادن آن به برنامه کاربردی از طریق سوکت

سرویس‌گیرنده SNMP



فرمت سگمنت‌های UDP



فرمت سگمنت UDP

source port number : شماره پورت مبدأ

Destination port number : شماره پورت مقصد

Length : طول

checksum : چک بیت‌های جمع

طول سگمنت UDP شامل سرآیند و داده‌های لایه کاربرد به بایت (حداکثر طول سگمت UDP برابر $2^{16} - 1 = 65535 \approx 64$ KBytes است.

داده‌های لایه کاربرد

چک بیت‌های جمع UDP (UDP checksum)

هدف: تشخیص خطا تغییر بیت‌ها در سگمنت ارسال شده و تشخیص خطا عملکرد لایه شبکه (مسیریابی)

	1st number	2nd number	sum
Transmitted:	5	6	11
Received:	4	6	11

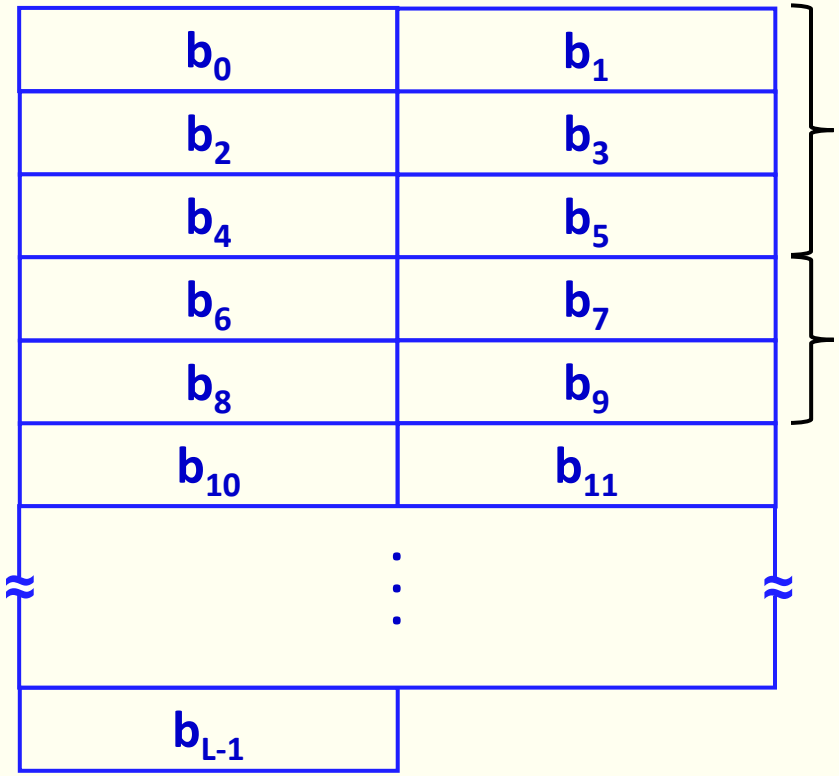
receiver-computed checksum ≠ sender-computed checksum (as received)

چک بیت‌های جمع UDP (UDP checksum)

هدف: تشخیص خطا تغییر بیت‌ها در سگمنت ارسال شده و تشخیص خطا عملکرد لایه شبکه (مسیریابی)

فرستنده:

- یک شبه سرآیند (حاوی آدرس IP مبدأ، آدرس IP مقصد، شناسه پروتکل UDP و طول سگمنت UDP) فقط برای محاسبه چک بیت‌های جمع به در بالای سرآیند سگمنت UDP قرار می‌گیرد.
- محتویات شبه سرآیند و سگمنت UDP به صورت اعداد ۱۶ بیتی در نظر گرفته می‌شوند (در صورت لزوم یک بایت صفر به داده اضافه می‌شود).
- حاصل جمع اعداد ۱۶ بیتی به روش wraparound sum بدست می‌آید.
- مکمل یک حاصل جمع محاسبه شده و مقدار در فیلد چک بیت‌های جمع سرآیند سگمنت قرار داده می‌شود.
- سگمنت UDP (بدون شبه سرآیند) به لایه شبکه (IP) داده می‌شود.



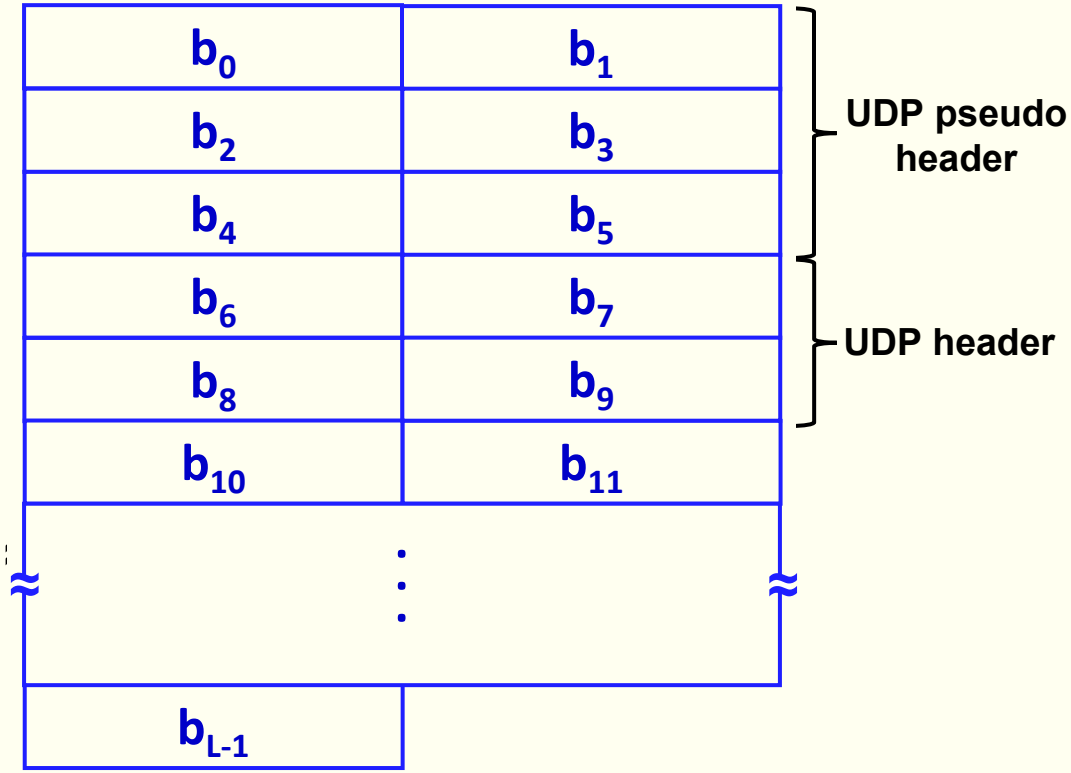
لایه انتقال

چک بیت‌های جمع UDP (UDP checksum)

هدف: تشخیص خطا تغییر بیت‌ها در سگمنت ارسال شده و تشخیص خطا عملکرد لایه شبکه (مسیریابی)

گیرنده:

- دریافت سگمنت از لایه شبکه (IP)
- اضافه کردن شبه سرآیند به سگمنت دریافتی (اطلاعات شبه سرآیند از لایه شبکه دریافت شده است)
- محتویات شبه سرآیند و سگمنت UDP به صورت اعداد ۱۶ بیتی در نظر گرفته می‌شوند (در صورت لزوم یک بایت صفر به داده اضافه می‌شود).
- حاصل جمع اعداد ۱۶ بیتی به روش wraparound sum بدست می‌آید.
- اگر نتیجه حاصل جمع مخالف صفر باشد، سگمنت دریافتی حتماً خطا دارد (تغییر محتوا یا خطای مسیریابی و تحویل اشتباه).
- اگر نتیجه حاصل جمع صفر باشد، خطایی تشخیص داده نمی‌شود!



چک بیت‌های جمع UDP (UDP checksum)

مثال: جمع دو عدد ۱۶ بیتی به روش wraparound sum:

		1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
		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	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

توجه کنید: در زمان جمع اعداد، رقم نقلی باید با حاصل جمع بدست‌آمده جمع شود.

لایه انتقال

خلاصه (UDP)

- پروتکل بدون زوائد
  - احتمال از دست‌دادن سگمنت‌ها یا تحویل خارج از ترتیب
  - سرویس بیشترین تلاش "ارسال سگمنت و امید به موفقیت دریافت آن توسط گیرنده"
- مزایای UDP:
  - بدون نیاز به برقراری اتصال (نداشتن تأخیر برقراری ارتباط)
  - توانایی ارسال در زمانی شبکه دچار مشکل شده است (دارای اختلال است)
  - تشخیص خطا (به کمک چک بیت‌های جمع) و کمک به قابلیت اطمینان
- ایجاد کارکردهای اضافی روی UDP در لایه کاربرد (به عنوان مثال HTTP/3)

لایه انتقال

انتقال مطمئن داده‌ها (reliable data transfer) یا کنترل خطا (Error Control)

کنترل خطا: فراهم‌سازی اطمینان کافی از انتقال صحیح داده‌ها:

- بدون خطا (error-free): تحویل دقیقاً همان بیت‌های یا بسته‌ها ارسال شده
- حفظ ترتیب ارسال (in-sequence): تحویل داده‌ها با همان ترتیب ارسال شده
- بدون تکرار (duplicate-free): تحویل حداکثر یک کپی از داده‌ها
- بدون ازدست‌دادن (loss-free): تحویل حداقل یک کپی از داده‌ها

کنترل خطا (Error Control)

عواملی که منجر به ایجاد خطا در ارسال داده‌ها می‌شوند:

- خطای در محتوا داده‌ها:
  - تغییر محتوا داده‌ها در زمان ارسال، درحین ارسال روی رسانه فیزیکی و در زمان تحویل داده‌ها
- از دست دادن داده‌ها:
  - وقوع ازدحام در مسیر ارسال
  - خطا در مسیریابی و تحویل اشتباه
  - تشخیص خطا در لایه‌های زیرین و حذف داده‌ها
- عدم حفظ ترتیب ارسال:
  - تغییر مسیر در زمان ارسال بسته‌های متوالی
- تکرار داده‌ها
  - ارسال چند کپی از داده‌ها برای افزایش قابلیت اطمینان

کنترل خطا (Error Control)

تعریف کنترل خطا:

- **تشخیص خطا:**
  - تغییر محتوای داده‌های ارسالی
  - ازدست دادن داده‌ها
  - عدم تحویل به ترتیب داده‌ها
  - تکرار داده‌ها
- **تصحیح خطا:**
  - اصلاح خطای تشخیص داده شده

کنترل خطا (Error Control)

دو رویکرد اصلی کنترل خطا:

- کنترل خطا رو به جلو (FEC – Forward Error Control)
  - تشخیص و تصحیح خطا در گیرنده انجام می‌شود.
- درخواست تکرار خودکار (ARQ- Automatic Repeat reQuest)
  - تشخیص خطا درگیرنده انجام می‌شود، اما تصحیح خطا با ارسال مجدد (به صورت خودکار) انجام می‌شود.

کشف خطا (تغییر محتوا)

مثال:

- فرض کنید که فرستنده یک کد n بیتی (۲ بیتی) برای گیرنده می‌فرستد.
- در اثر خطا یک بیت از n بیت تغییر می‌کند (بیت "۰" به "۱" یا برعکس "۱" به "۰" تبدیل می‌شود).

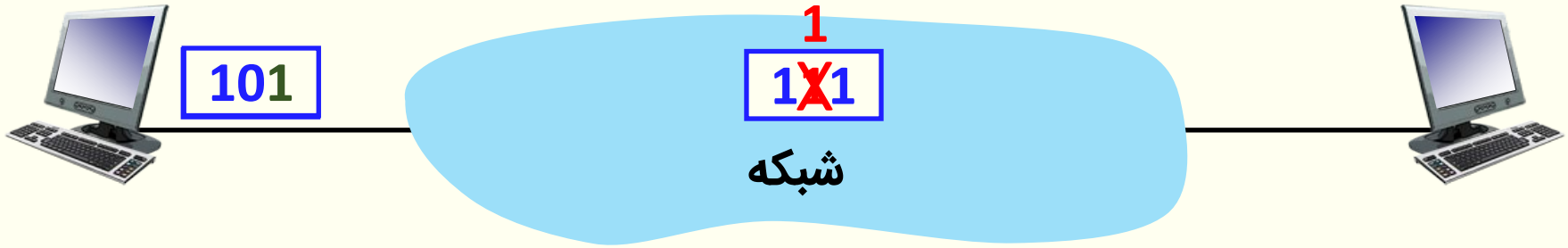
سوال: آیا خطا رخ داده توسط گیرنده قابل تشخیص است؟ چرا؟

پاسخ: خیر! برای گیرنده قابل تشخیص نیست که فرستنده این کد را ارسال کرده یا در اثر خطا کد ارسالی تغییر کرده است.

کشف خطا (تغییر محتوا)

افزدون چک بیت‌های تشخیص خطا:

- چک بیت‌های تشخیص خطا با یک رابطه جبری از روی محتوای داده‌ها تولید می‌شوند. به عنوان بیت توازن زوج



سوال: آیا خطا رخ داده توسط گیرنده قابل تشخیص است؟ چرا؟

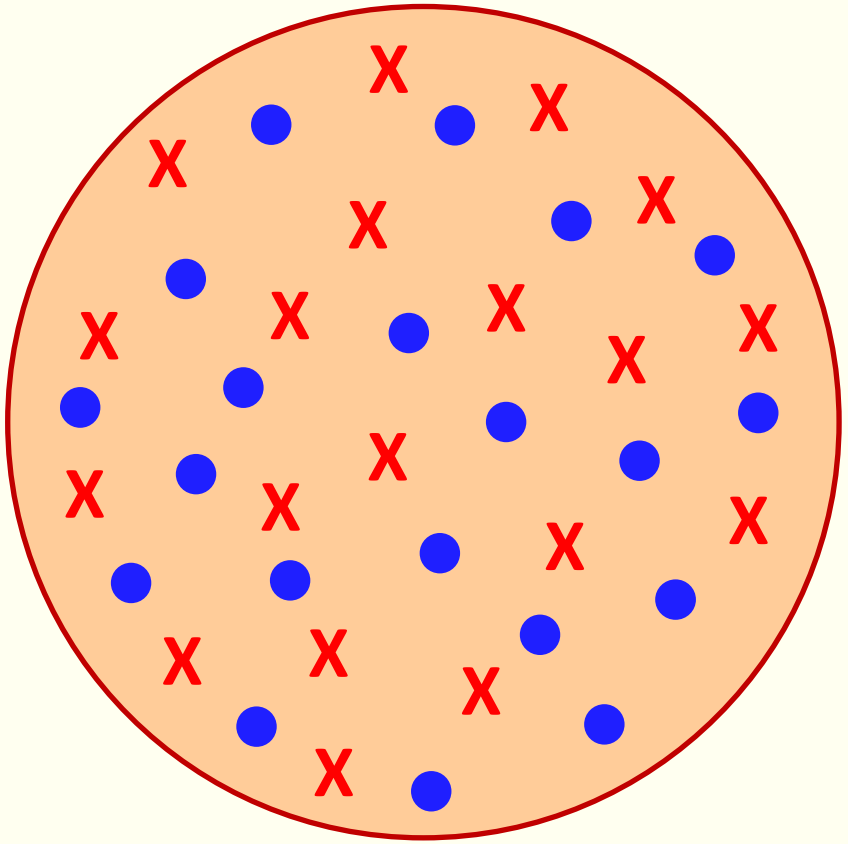
پاسخ: بله! بدلیل اینکه توازن کد دریافتی فرد است و فرستنده کد با توازن فرد ارسال نمی‌کند (کد نامعتبر)

سوال: اگر دو بیت خطا رخ داده بود چطور؟

کشف خطا تغییر محتوا

افزدون چک بیت‌های تشخیص خطا:

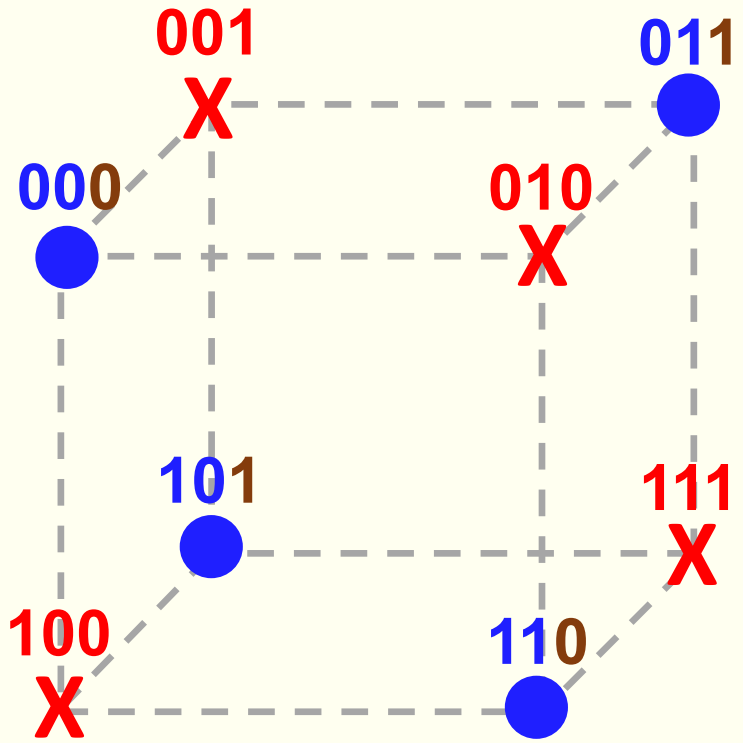
- **کدهای معتبر:**
  - کدهایی هستند که فرستنده تولید و ارسال می‌کند.
- **کدهای غیرمعتبر:**
  - کدهایی هستند که به هیچ‌وجه فرستنده تولید و ارسال نمی‌کند.
- **تشخیص خطا:**
  - دریافت کد غیر معتبر در گیرنده



سوال: آیا امکان دارد گیرنده کد معتبری دریافت کند که خطا داشته باشد؟

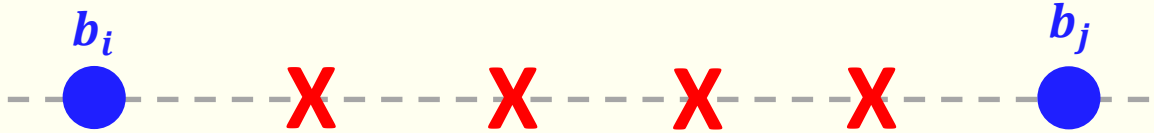
پاسخ: بله

کشف خطا تغییر محتوا



فاصله همینگ (Hamming Distance): فاصله بین کدهای معتبر

- تعداد تغییرات بیتی تا یک کد معتبر به یک کد معتبر دیگر تبدیل شود:



$$HammingDistance(b_i, b_j) = 5$$

- حداقل فاصله همینگ در یک مجموعه کد معتبر

$$D_{min} = \min_{\forall i,j \in B} HammingDistance(b_i, b_j)$$

اگر حداقل فاصله همینگ در یک مجموعه کد D_{min} باشد، آنگاه:

- $D_{min} - 1$ خطا قابل کشف است.
- $\left\lfloor \frac{D_{min}-1}{2} \right\rfloor$ خطای اتفاق افتاده قابل تصحیح است (اگر مطمئن باشیم تعداد بیت‌های دارای خطا از $\left\lfloor \frac{D_{min}-1}{2} \right\rfloor$ کمتر است، خطای اتفاق افتاده قابل تصحیح است).

کشف خطا تغییر محتوا

داده	چک بیت (توازن زوج)
0 0	0
0 1	1
1 0	1
1 1	0

تعداد کل کدها $\#code = 2^n = 8$

تعداد کدهای معتبر $\#valid_{code} = 2^k = 4$

تعداد کدهای غیرمعتبر $\#invalid_{code} = 2^n - 2^k = 4$

مجموعه کدهای غیرمعتبر $S_{valid} = \{000, 011, 101, 110\}$

مجموعه کدهای غیرمعتبر $S_{invalid} = \{001, 010, 100, 111\}$

حداقل فاصله همینگ $D_{min} = 2$

تعداد خطای قابل کشف: ؟

تعداد خطای قابل تصحیح: ؟

کشف خطا تغییر محتوا

یک کلمه کد n بیتی را در نظر بگیرید، اگر احتمال خطای بیتی (BER-Bit Error Rate) برابر p باشد آنگاه:

(دقت کنید: در یک کانال مطمئن $p \ll 1$ است.)

احتمال موفقیت $P_S = Pr[\text{no bit errors}] = (1-p)^n$

احتمال خطا $P_F = 1 - P_S = 1 - (1-p)^n$

احتمال یک بیت خطا $Pr[\text{one bit error}] = \binom{n}{1} p^1 (1-p)^{n-1} = np(1-p)^{n-1}$

احتمال دو بیت خطا $Pr[\text{two bit errors}] = \binom{n}{2} p^2 (1-p)^{n-2} = \frac{n(n-1)}{2} p^2 (1-p)^{n-2}$

احتمال k بیت خطا $Pr[k \text{ bit errors}] = \binom{n}{k} p^k (1-p)^{n-k} = \frac{n!}{k!\,(n-k)!} p^k (1-p)^{n-k}$

مثال: $n = 15$ و $p = 10^{-8}$ =

احتمال یک بیت خطا $Pr[\text{one bit error}] = 15 \times 10^{-8} \times \left(1 - 10^{-8}\right)^{14} = 1.5 \times 10^{-7}$

احتمال دو بیت خطا $Pr[\text{two bit error}] = \frac{15 \times 14}{2} \times 10^{-16} \times \left(1 - 10^{-8}\right)^{13} = 1.05 \times 10^{-14} \approx 0$

لایه انتقال

رویکرد کنترل خطای FEC:

- افزدون چک بیت‌ها به داده‌ها برای کشف و تصحیح خطا
- تعیین تعداد چک بیت‌ها بر اساس اندازه داده و تعداد خطای قابل تصحیح
- سربار بالای چک بیت‌ها (بازدهی پایین)
- پیچیدگی محاسباتی کدبرداری در کدهای درهم‌ساز
- کاربردها:
  - در ارتباطات یک طرفه (پخش رادیو و تلویزیونی)
  - در ارتباطات با تأخیر زیاد (ارتباطات ماهواره‌ای (تصویربرداری و ...)
  - عدم امکان نگهداری و ارسال مجدد (جریان‌سازی ویدیو زنده)
  - عدم امکان استفاده از روش‌های ARQ بدلیل نرخ بالای خطا یا سربار زیاد ارسال مجدد (ارتباطات بی‌سیم در محیط‌های با نویز بالا)
- انواع روش‌های FEC:
  - کدهای بلوکی (block code): Hamming، Reed–Solomon، Golay، BCH، Multidimensional parity
  - کدهای درهم‌ساز (convolutional code): Viterbi، MAP، BCJR

FEC: کد همینگ

- کلمات کد n بیتی هستند (n به گونه‌ای انتخاب می‌شود که احتمال بیش از یک خطا بسیار کم است و تقریباً صفر است
- کد همینگ، فاصله همینگ ۳ ایجاد می‌کند (یعنی یک بیت خطا قابل تصحیح است)
- از این کلمه کد n بیتی m بیت چک بیت است و k آن داده است ($m = \lceil \log_2(n+1) \rceil$ و $k = n - m$)
- نرخ کدگذاری (coding rate) برابر است با k/n
- چک بیت‌ها در لابلای داده قرار داده شده و کلمه کد n بیتی را ارسال می‌کند (بیت‌های با اندیس توان ۲ در کلمه n بیتی، چک بیت هستند).
- چک بیت‌های با روابط زیر محاسبه می‌شوند.

b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}	$\cdots$	b_n
c_1	c_2	d_1	c_4	d_2	d_3	d_4	c_8	d_5	d_6	d_7	d_8	$\cdots$	d_k

$$c_1 = b_3 + b_5 + b_7 + b_9 + b_{11} + b_{13} + \cdots = d_1 + d_2 + d_4 + d_5 + d_7 + d_9 + \cdots$$

$$c_2 = b_3 + b_6 + b_7 + b_{10} + b_{11} + b_{14} + \cdots = d_1 + d_3 + d_4 + d_6 + d_7 + d_{10} + \cdots$$

$$c_4 = b_5 + b_6 + b_7 + b_{12} + b_{13} + b_{14} + \cdots = d_2 + d_3 + d_4 + d_8 + d_9 + d_{10} + \cdots$$

$$c_8 = b_9 + b_{10} + b_{11} + b_{12} + b_{13} + b_{14} + \cdots = d_5 + d_6 + d_7 + d_8 + d_9 + d_{10} + \cdots$$

$$\vdots$$

FEC: کد همینگ

- گیرنده با دریافت کلمه کد n بیتی، یک عدد m بیتی به نام نشانه وجود خطا (syndrome) به صورت زیر محاسبه می‌شود.

r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}	r_{11}	r_{12}	$\cdots$	r_n

$$s_1 = r_1 + r_3 + r_5 + r_7 + r_9 + r_{11} + r_{13} + \cdots$$

$$s_2 = r_2 + r_3 + r_6 + r_7 + r_{10} + r_{11} + r_{14} + \cdots$$

$$s_4 = r_4 + r_5 + r_6 + r_7 + r_{12} + r_{13} + r_{14} + \cdots$$

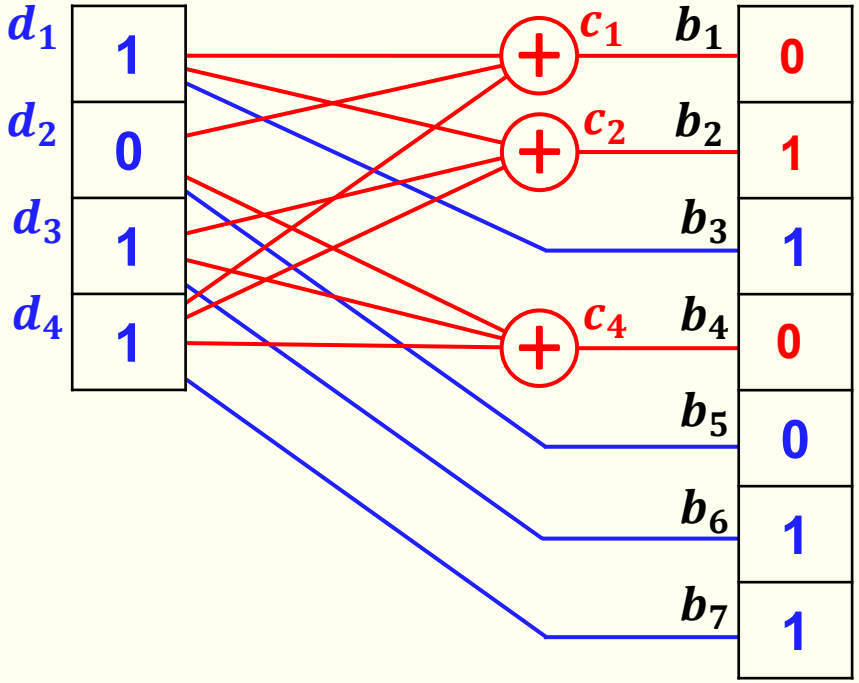
$$s_8 = r_8 + r_9 + r_{10} + r_{11} + r_{12} + r_{13} + r_{14} + \cdots$$

$$\vdots$$

$$S = (\cdots s_8 s_4 s_2 s_1)_{10}$$

- در صورتی که $S = 0$ باشد، در کد دریافتی خطایی تشخیص داده نشده است.
- در غیر اینصورت $(S \neq 0)$ کد دریافتی دارای است و با توجه به اینکه حداکثر تعداد بیت‌های دارای خطا یک است، عدد S اندیس بیت دارای خطاست و گیرنده با تغییر آن بیت، کد دریافتی را تصحیح می‌کند.

FEC: کد همینگ - مثال



فرض کنید $n = 7$ باشد در نتیجه $m = 3$ و $k = 4$ خواهد بود.

نرخ کد برابر است با: $4/7$ (FEC $4/7$)

اگر داده (1011) باشد آنگاه:

$$c_1 = b_3 + b_5 + b_7 = d_1 + d_2 + d_4 = 1 + 0 + 1 = 0$$

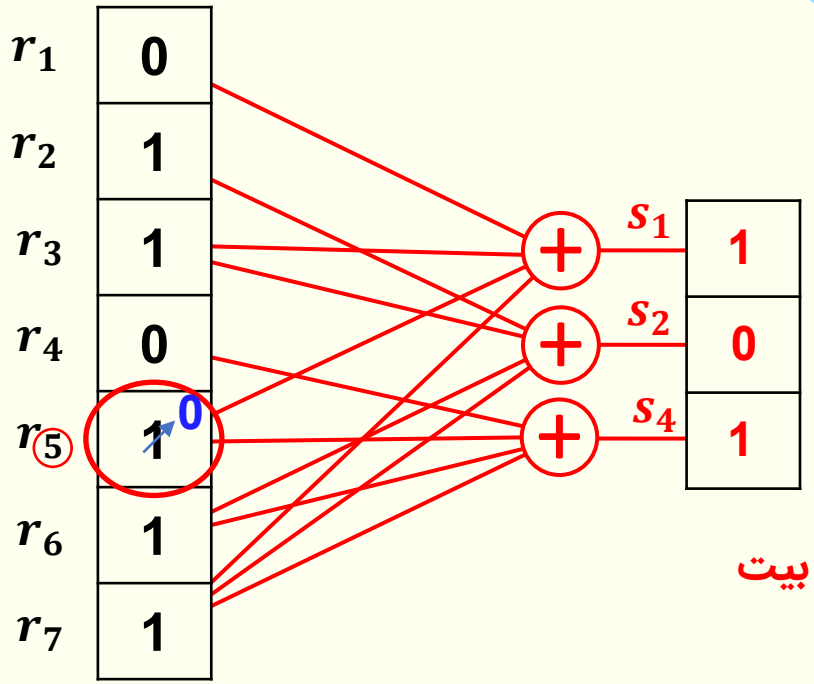
$$c_2 = b_3 + b_6 + b_7 = d_1 + d_3 + d_4 = 1 + 1 + 1 = 1$$

$$c_4 = b_5 + b_6 + b_7 = d_2 + d_3 + d_4 = 0 + 1 + 1 = 0$$

و کلمه کد ارسالی (0110011) خواهد بود.

FEC: کد همینگ - مثال

- دریافت کد (0110111) و محاسبه نشانه وجود خطا (syndrome) و تصحیح خطای احتمالی توسط گیرنده :



$$s_1 = r_1 + r_3 + r_5 + r_7 = 0 + 1 + 1 + 1 = 1$$
$$s_2 = r_2 + r_3 + r_6 + r_7 = 1 + 1 + 1 + 1 = 0$$
$$s_4 = r_4 + r_5 + r_6 + r_7 = 0 + 1 + 1 + 1 = 1$$
$$\Rightarrow S = (101)_{10} = 5$$

- بدلیل اینکه $S \neq 0$ است بنابراین کد دریافتی حتماً خطا دارد و با فرض اولیه یک بیت خطا، بیت پنجم دارای خطا است و کد تصحیح شده (0110011) و داده دریافتی (1011) است.

لایه انتقال

رویکرد کنترل خطای ARQ:

- افزدون چک بیت‌ها و سایر روش‌های دیگر برای کشف خطا در گیرنده
- ارسال تأییده (acknowledgement) به فرستنده در صورت دریافت بدون خطای داده‌ها توسط گیرنده
- اقدام به ارسال مجدد داده‌ها در صورت عدم دریافت تاییده در زمان مقرر (بعضی از روش‌های ARQ از مکانیزیم‌های دیگری نیز جهت تشخیص زودهنگام خطا استفاده می‌کنند)
- سربار چک بیت‌ها پایین است ولی سربار اصلی ارسال مجدد داده‌هاست (بازدهی پایین در شبکه‌های با نرخ خطا یا احتمال ازدست‌دادن بالا)
- حجم حافظه مصرفی و نگهداری وضعیت اتصال از جمله پیچیدگی روش‌های ARQ است.
- کاربردها:
  - فقط در ارتباطات دو طرفه قابل استفاده است (شبکه‌های کامپیوتری)
  - در شبکه‌های که سربار کمتری نسبت به روش‌های FEC دارد (در شبکه‌های با نرخ خطا و احتمال از دست‌دادن پایین).
- انواع روش‌های ARQ:
  - روش توقف و انتظار (Stop-and-Wait ARQ)
  - روش بازگشت به عقب به اندازه N (Go-back-N ARQ)
  - روش تکرار انتخابی (Selective Repeat ARQ)
  - ...

روش‌های ARQ: کلیات

زمانبند
Timer

بسته‌های کنترلی:
تأییده (ACK) و
در بعضی روش‌ها تأییده منفی (NACK)

ACK
Control Packet

بسته‌های داده

H D
Data Packet

- ارسال موفقیت‌آمیز:
  - دریافت بدون خطا
  - دریافت بدون تکرار
  - دریافت با ترتیب مورد انتظار

- ارسال همراه با خطا:
  - عدم دریافت (از دست‌دادن داده‌ها)
  - دریافت با خطای تغییر داده‌ها
  - دریافت تکراری
  - دریافت خارج از ترتیب مورد انتظار

لایه انتقال

روش توقف و انتظار (stop-and-wait ARQ)

فرستنده:

- آماده برای دریافت داده از لایه بالاتر جهت ارسال
- ارسال داده‌ها (فقط یک بسته)، توقف ارسال و انتظار برای نتیجه ارسال:
  - نگهداری یک کپی بسته بعد از ارسال آن
  - روشن کردن زمانبند با زمان مورد انتظار برای گرفتن تاییده دریافت از گیرنده
- دریافت تاییده در زمان تعیین شده:
  - خاموش کردن زمانبند
  - عدم نیاز به نگهداری کپی بسته (دور ریختن کپی بسته)
  - آماده برای ارسال بعدی
- عدم دریافت تاییده در زمان تعیین شده (منقضی‌شدن زمان انتظار):
  - ارسال مجدد بسته
  - روشن کردن مجدد زمانبند

گیرنده:

- آماده برای دریافت بسته
- دریافت بسته مورد انتظار و بدون خطا:
  - تحویل داده دریافتی به لایه بالتر
  - ارسال تاییده دریافت صحیح
  - آماده برای دریافت بعدی
- دریافت بسته خارج از انتظار یا با خطا:
  - دور ریختن بسته دریافتی

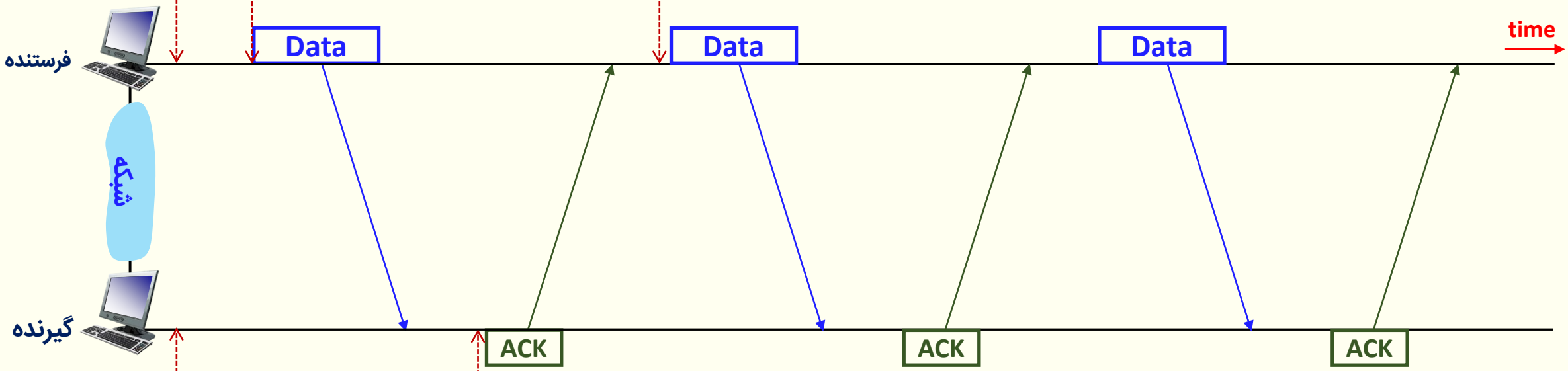
لایه انتقال

روش توقف و انتظار (stop-and-wait ARQ)

دریافت تاییده:
- عدم نگهداری داده‌های تاییده شده
- خاموش کردن زمانبند
- بروزرسانی متغیرها (آماده برای دریافت داده‌ها از لایه بالاتر جهت ارسال)

دریافت داده‌ها از لایه بالاتر:
- تعیین مقادیر فیلدهای سرآیند
- ایجاد بسته و ارسال آن
- نگهداری یک کپی از داده‌های ارسالی
- روشن‌کردن زمانبند با زمان تعیین شده

برقراری اتصال (آماده برای دریافت داده‌ها از لایه بالاتر جهت ارسال):
- تخصیص منابع
- مقداردهی اولیه متغیرها

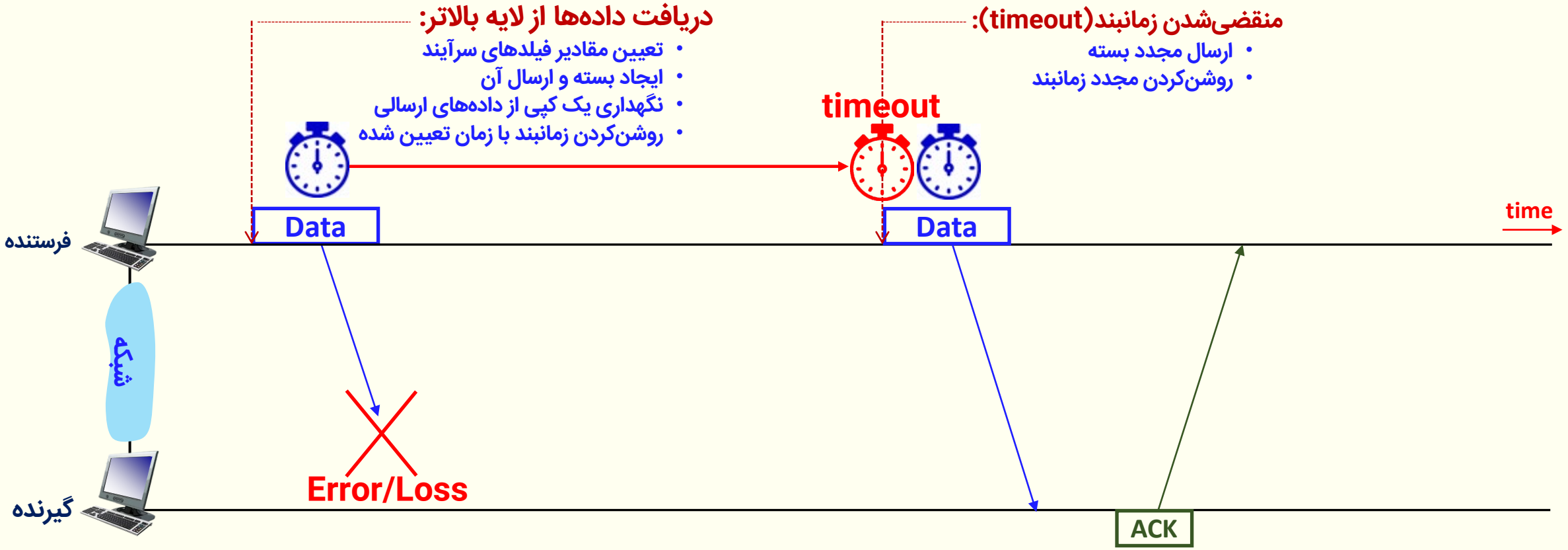


برقراری اتصال (آماده برای دریافت داده‌ها):
- تخصیص منابع
- مقداردهی اولیه متغیرها

دریافت بسته داده‌های بدون خطا:
- تحویل داده‌ها به لایه بالاتر
- ارسال تاییده
- بروزرسانی متغیرها (آماده برای دریافت داده‌های جدید)

لایه انتقال

روش توقف و انتظار (stop-and-wait ARQ)



لایه انتقال

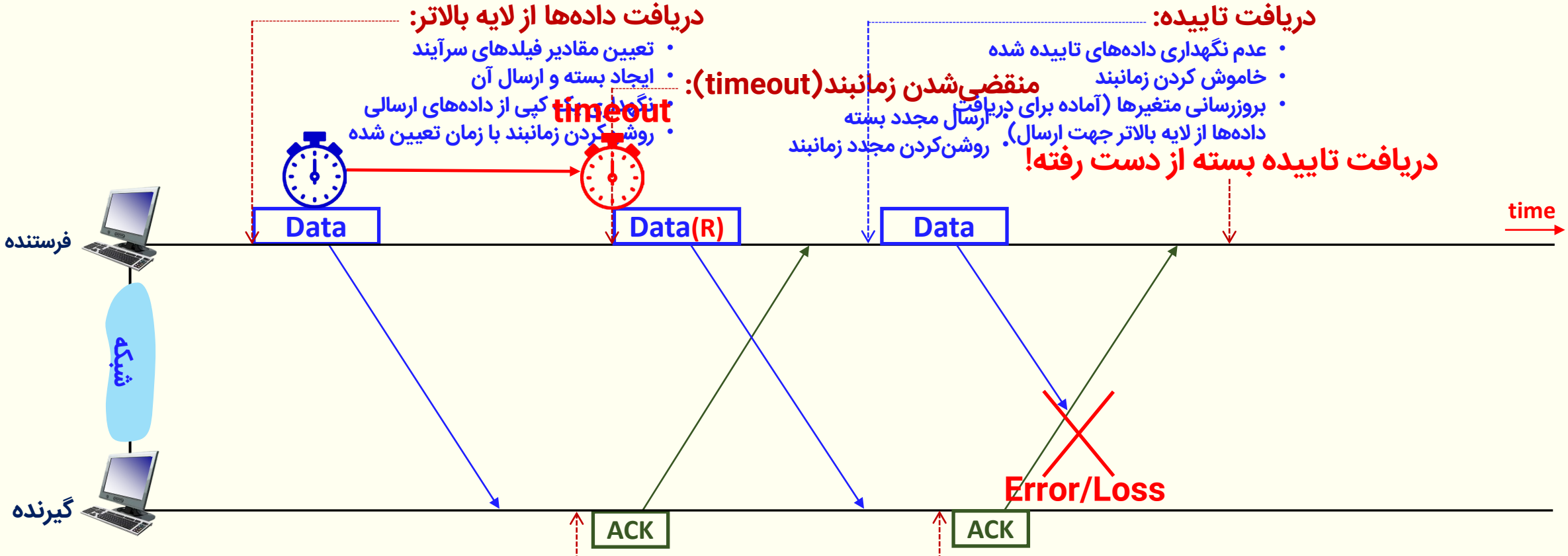
روش توقف و انتظار (stop-and-wait ARQ)

ضرورت داشتن فیلد شماره ترتیب ارسال در سرآیند بسته‌های داده و فیلد شماره تاییده در سرآیند بسته‌های کنترلی ACK

دریافت داده‌ها از لایه بالاتر:
- تعیین مقادیر فیلدهای سرآیند
- ایجاد بسته و ارسال آن
- نگهداری یک کپی از داده‌های ارسالی
- روشن‌کردن زمانبند با زمان تعیین شده

منقضی‌شدن زمانبند(timeout):
- ارسال مجدد بسته
- روشن‌کردن مجدد زمانبند

دریافت تاییده:
- عدم نگهداری داده‌های تاییده شده
- خاموش کردن زمانبند
- بروزرسانی متغیرها (آماده برای دریافت داده‌ها از لایه بالاتر جهت ارسال).



دریافت بسته داده‌های بدون خطا:
- تحویل داده‌ها به لایه بالاتر
- ارسال تاییده
- بروزرسانی متغیرها (آماده برای دریافت داده‌های جدید)

دریافت داده‌های تکراری !

روش توقف و انتظار (stop-and-wait ARQ)

فرستنده:

- در نظر گرفتن یک فیلد در سرآیند بسته‌های داده ارسالی به نام شماره ترتیب (sequence number)
- نگهداری شماره ترتیب ارسال ($N(S)$)
- قرار دادن مقدار شماره ترتیب ارسال ($N(S)$) در فیلد شماره ترتیب در سرآیند بسته‌های داده ارسالی در زمان ارسال هر بسته داده
- افزایش فرستنده شماره ترتیب ارسال ($N(S)$) بعد از هر ارسال موفقیت‌آمیز (دریافت تاییده از گیرنده)

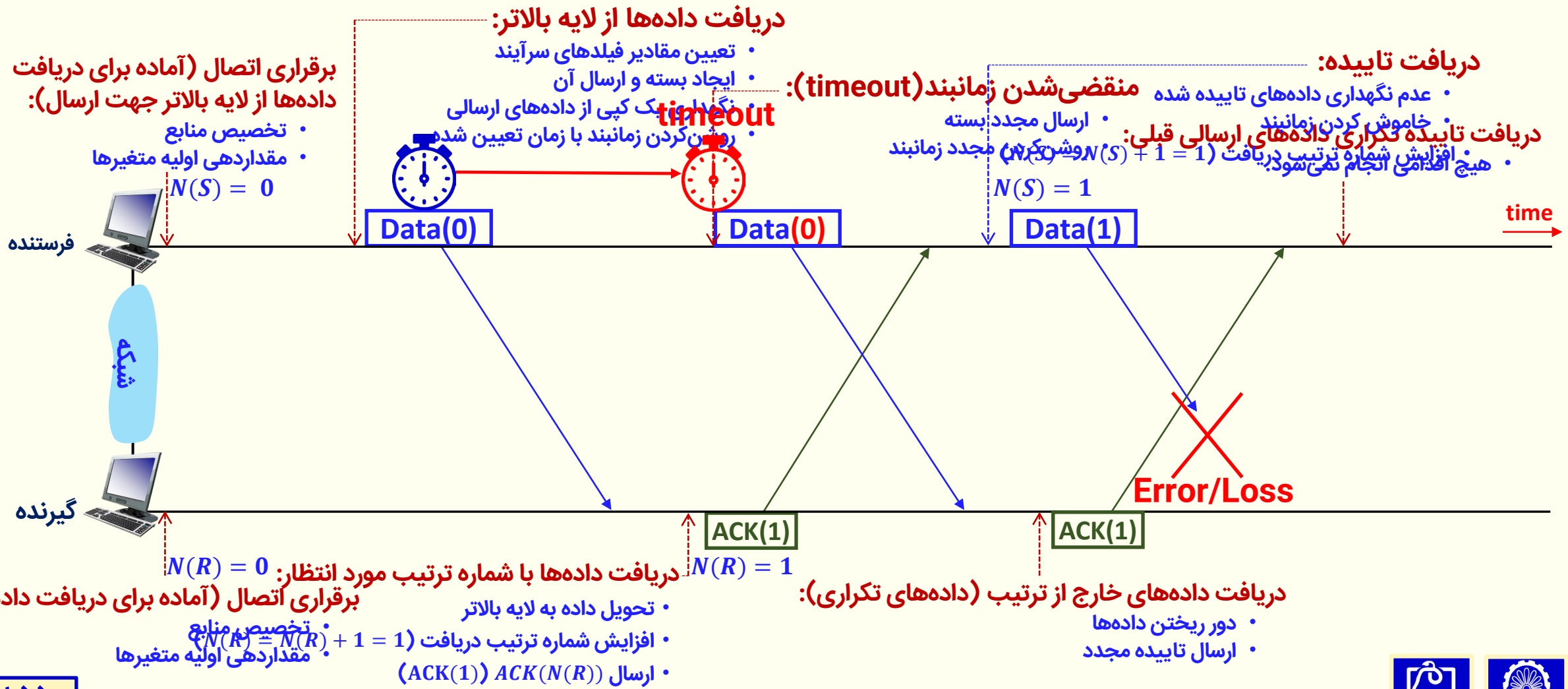
گیرنده:

- در نظر گرفتن یک فیلد در بسته‌های کنترلی تاییده به نام شماره تاییده (acknowledgement number)
- نگهداری شماره ترتیب دریافت بسته مورد انتظار ($N(R)$)
- در صورت دریافت بسته داده با شماره ترتیب مورد انتظار:
  - تحویل داده دریافتی به لایه بالاتر
  - افزایش شماره ترتیب دریافت ($N(R)$)
  - قرار دادن شماره ترتیب دریافت (شماره ترتیب بسته بعدی مورد انتظار) در فیلد شماره تاییده بسته کنترلی تاییده (ACK) و ارسال تاییده

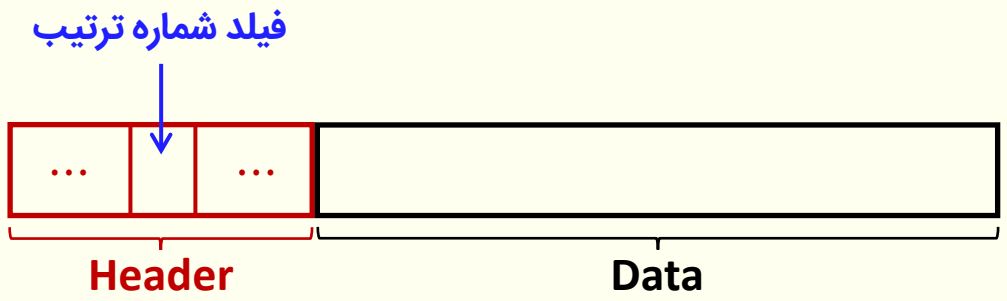
در ابتدای برقراری اتصال متغیرهای حالت $N(S)$ و $N(R)$ با هم برابر هستند (با یک یکسان تنظیم می‌شوند).

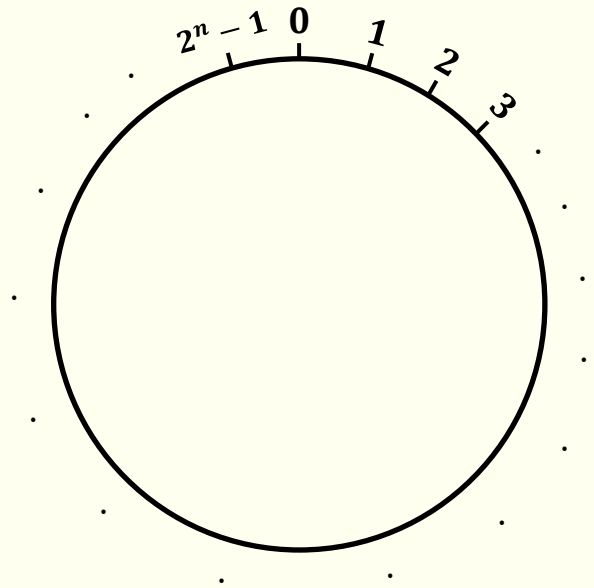
روش توقف و انتظار (stop-and-wait ARQ)

ضرورت داشتن فیلد شماره ترتیب ارسال در سرآیند بسته‌های داده و فیلد شماره تاییده در سرآیند بسته‌های کنترلی ACK



روش توقف و انتظار (stop-and-wait ARQ)





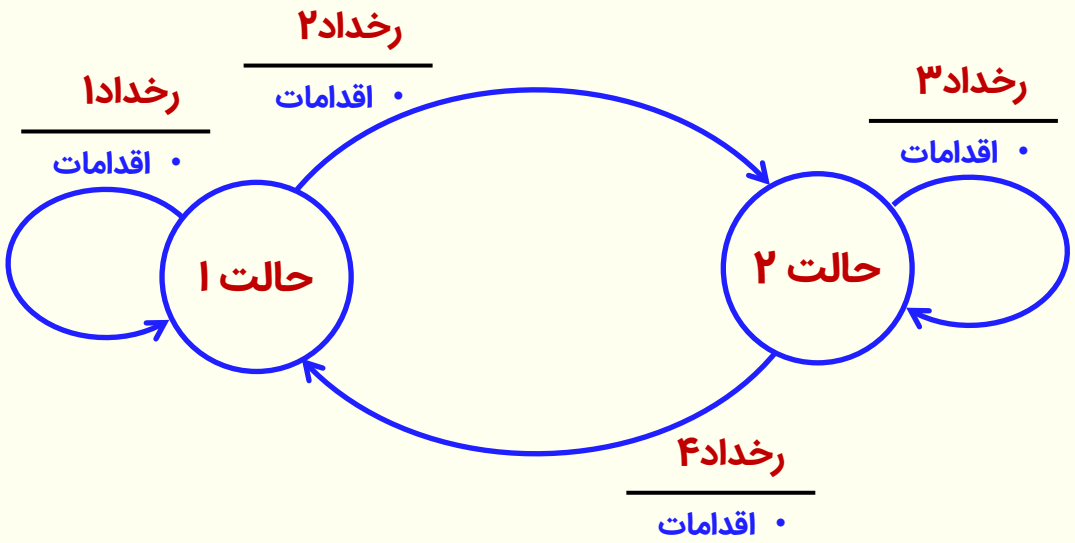
تعیین تعداد بیت‌های شماره ترتیب (و شماره تاییده):

- شماره ترتیب یک فیلد در سرآیند بسته ارسالی است و تعداد بیت‌های آن محدود است (به عنوان مثال: n بیت)
- شماره ترتیب بعد از ارسال تعدادی بسته (2^n) تکرار خواهد شد.
- برای کاهش سربار سرآیند بسته لازم است حداقل تعداد بیت برای شماره ترتیب و شماره تاییده استفاده کرد.
- حداقل تعداد بیت‌های شماره ترتیب برای پروتکل ARQ توقف و انتظار چقدر است؟
- در پروتکل ARQ توقف و انتظار حداقل تعداد بیت شماره ترتیب و شماره تاییده یک بیت است. چرا؟

روش توقف و انتظار (stop-and-wait ARQ)

توصیف (بدون ابهام) کارکرد (function) پروتکل‌های شبکه با استفاده دیاگرام حالت (state diagram):

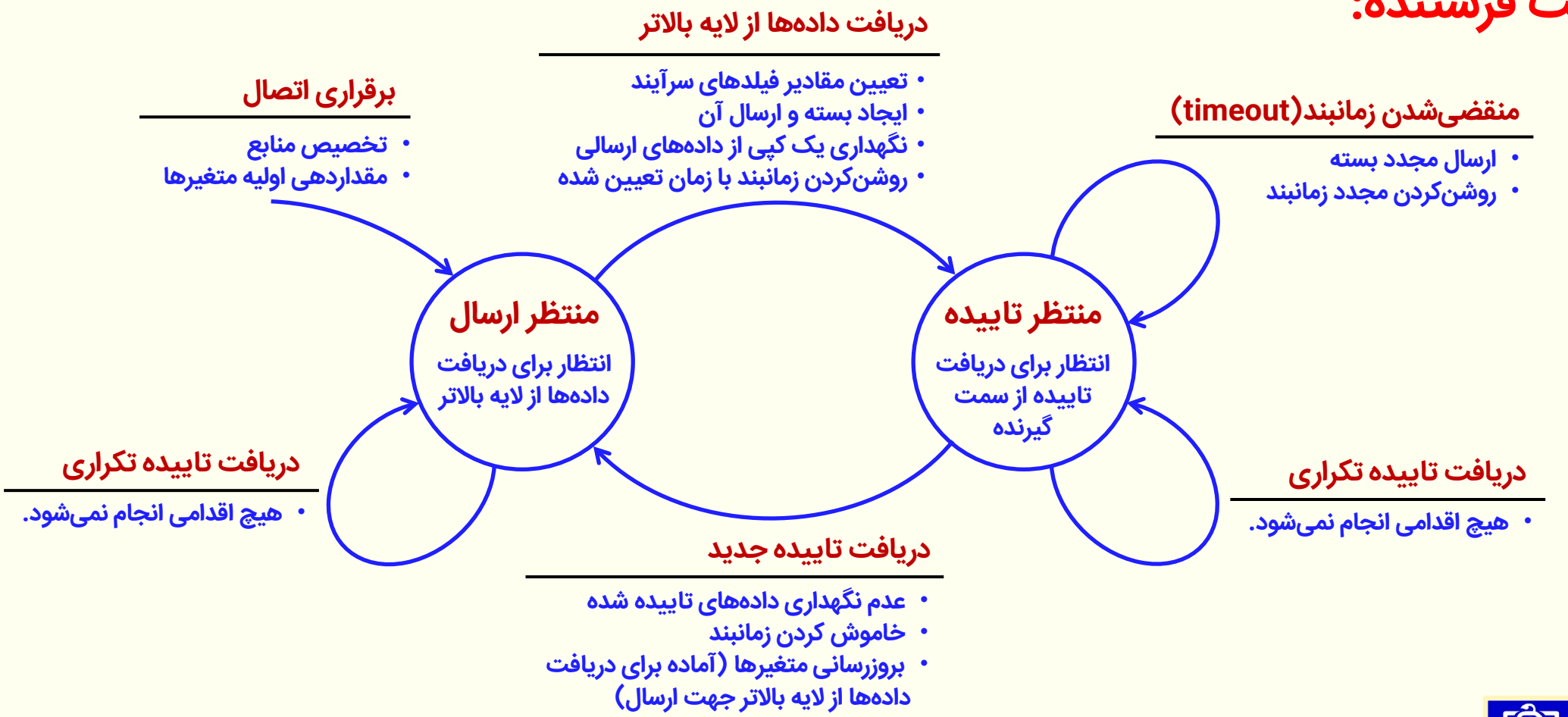
- غالب پروتکل‌های شبکه ماشین‌های با حالت‌های محدود (FSM - finite state machine) هستند.
- توصیف بدون ابهام کارکرد پروتکل با استفاده از دیاگرام حالت
- اجزای دیاگرام حالت:
  - حالت‌ها (atates)
  - رخدادها (events)
  - اقدامات به ازای وقوع رخدادها و حالت فعلی
  - تغییر حالت بر اساس رخداد اتفاق افتاده و حالت فعلی



روش توقف و انتظار (stop-and-wait ARQ)

دیاگرام حالت (state diagram) پروتکل ارسال مجدد خودکار (ARQ) توقف و انتظار (stop-and-wait)

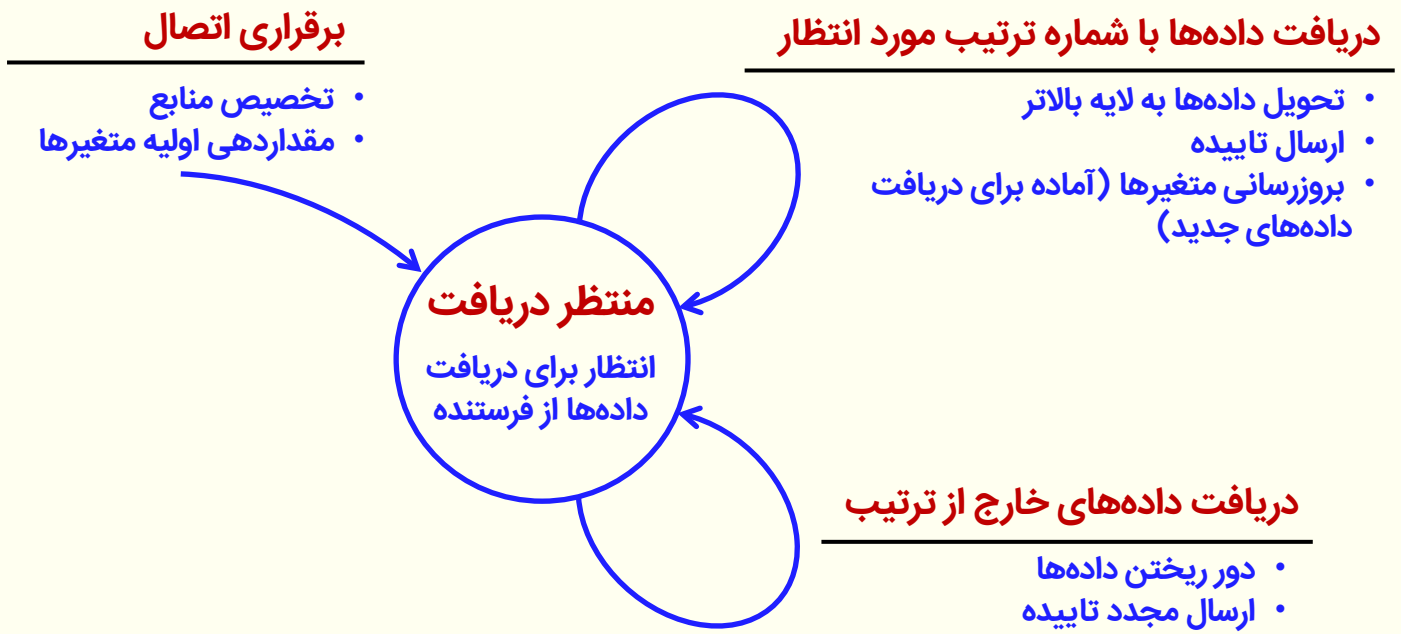
سمت فرستنده:



روش توقف و انتظار (stop-and-wait ARQ)

دیاگرام حالت (state diagram) پروتکل ارسال مجدد خودکار (ARQ) توقف و انتظار (stop-and-wait)

سمت گیرنده:



روش توقف و انتظار (stop-and-wait ARQ)

کارآیی پروتکل ARQ توقف و انتظار:

- بهره‌وری (utilization): درصد استفاده مؤثر (مفید) از پهنای باند

$$utilization = u = \frac{R_{eff}}{R}$$

R: گذردهی (پهنای باند)

R_{eff}: گذردهی مؤثر (effective throughput)

$$R_{eff} = \frac{Average\ data\ sent\ successfully\ (bit)}{Average\ sent\ time\ (second)}$$

روش توقف و انتظار (stop-and-wait ARQ)

کارآیی پروتکل ARQ توقف و انتظار:

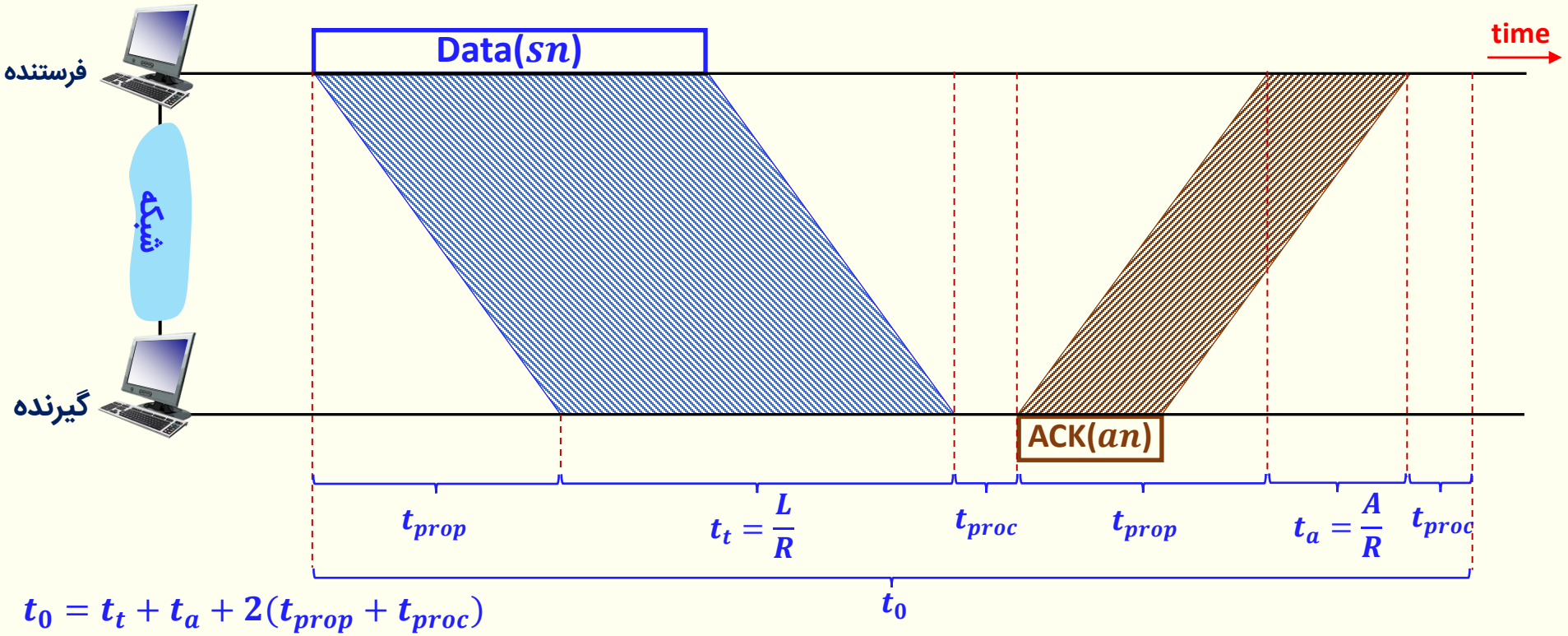
- تعریف نمادها:

نماد	تعریف
L	اندازه بسته‌های داده (شامل سرآیند)
H	اندازه سرآیند بسته‌های داده
D	اندازه داده‌های حمل شده توسط یک بسته $(D = L - H)$
A	اندازه بسته‌های کنترلی تاییده $(A \approx H)$
R	نرخ ارسال (پهنای‌باند)
P_F	احتمال خطا یا ازدست‌دادن بسته (احتمال عدم دریافت تاییده)
R_{eff}^0	نرخ ارسال مؤثر در حالت (ایده‌آل) بدون خطا
R_{eff}	نرخ ارسال مؤثر (با وجود احتمال خطا یا ازدست‌دادن بسته‌ها)
t_t	زمان ارسال بسته داده $(t_t = \frac{L}{R})$
t_a	زمان ارسال بسته کنترلی تاییده $(t_a = \frac{A}{R})$

نماد	تعریف
t_{proc}	زمان پردازش سرآیند بسته (داده یا کنترلی)
t_{prop}	تأخیر انتشار انتها به انتها یکطرفه
RTT	تأخیر رفت و برگشت $(RTT = 2t_{prop} + 2t_{proc})$
t_0	زمان ارسال و دریافت تاییده در حالتی خطا رخ ندهد.
t_{out}	زمان مقضی شدن زمانبند ارسال مجدد
t_{total}	کل زمان صرف شده برای یک ارسال موفقیت‌آمیز
$E[t_{total}]$	متوسط کل زمان صرف شده برای یک ارسال موفقیت‌آمیز
$u_{s\&w}^0$	بهره‌وری پروتکل ARQ stop-and-wait در حالت (ایده‌آل) بدون خطا
$u_{s\&w}$	بهره‌وری پروتکل ARQ stop-and-wait (با وجود احتمال خطا یا ازدست‌دادن بسته‌ها)

روش توقف و انتظار (stop-and-wait ARQ)

بهره‌وری پروتکل ARQ توقف و انتظار در حالت بدون خطا

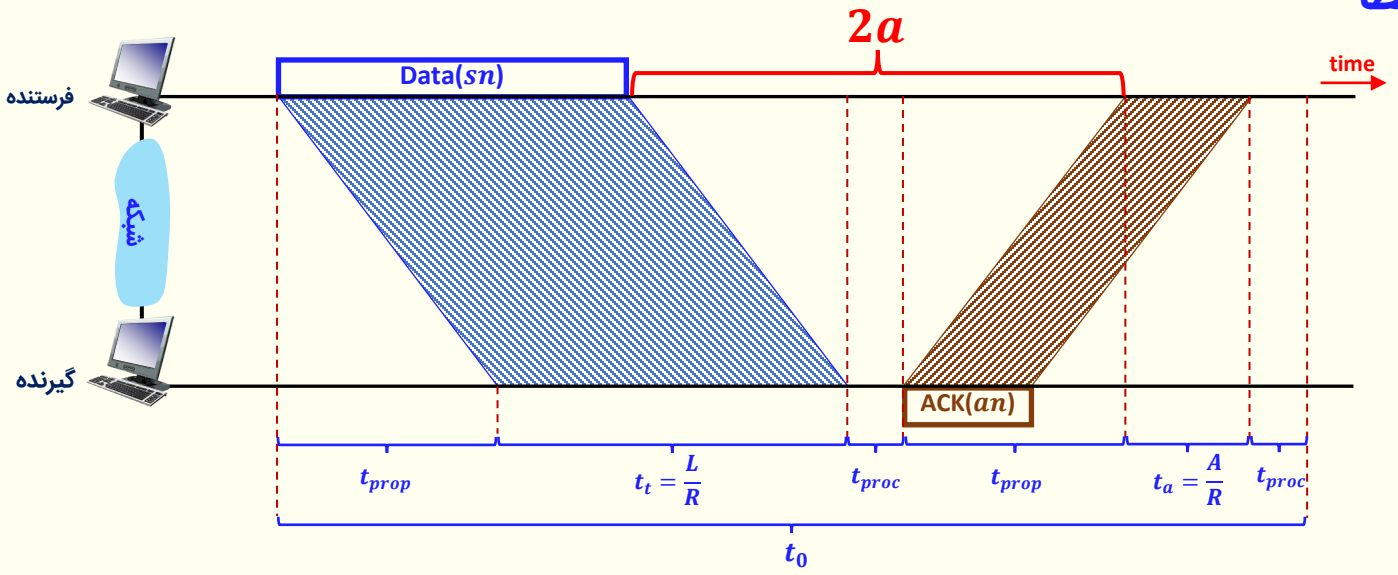


$$t_0 = t_t + t_a + 2(t_{prop} + t_{proc})$$

$$R_{eff}^0 = \frac{D}{t_0} = \frac{L-H}{t_t + t_a + 2(t_{prop} + t_{proc})} = \frac{L-H}{\frac{L}{R} + \frac{A}{R} + 2(t_{prop} + t_{proc})} = \frac{\left(1 - \frac{H}{L}\right)R}{1 + \frac{A}{L} + \frac{2(t_{prop} + t_{proc})R}{L}}$$

روش توقف و انتظار (stop-and-wait ARQ)

بهره‌وری پروتکل ARQ توقف و انتظار در حالت بدون خطا



$$R_{eff}^0 = \frac{\left(1 - \frac{H}{L}\right)R}{1 + \frac{A}{L} + \frac{2(t_{prop} + t_{proc})R}{L}}$$

عامل سربار سرآیند

عامل حاصلضرب تأخیر در پهنای‌باند $(2a)$

$$u_{s\&w}^0 = \frac{R_{eff}^0}{R} = \frac{1 - \frac{H}{L}}{1 + \frac{A}{L} + \frac{2(t_{prop} + t_{proc})R}{L}}$$

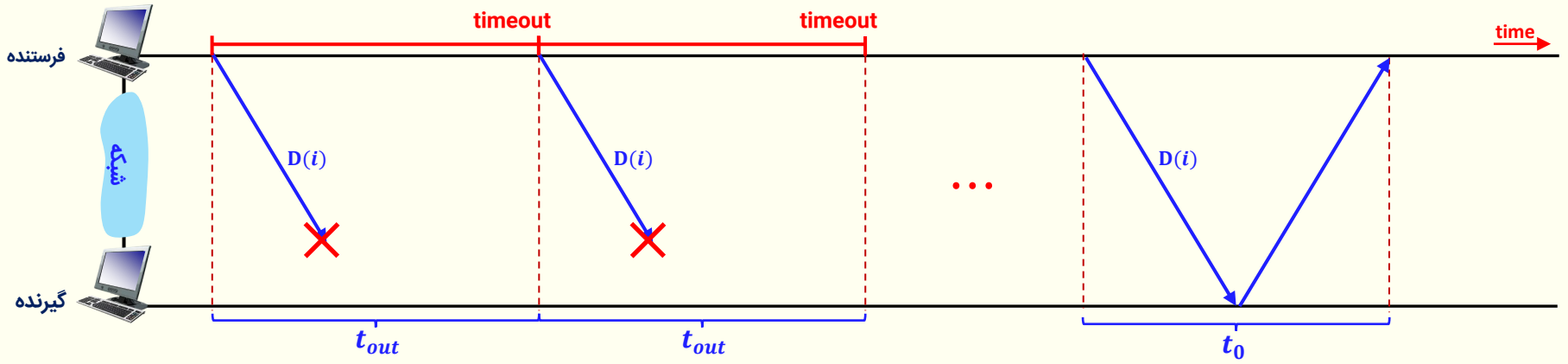
عامل سربار بسته کنترلی تاییده

$$a = \frac{t_{prop}}{t_t} = \frac{t_{prop}R}{L}$$

a: تعداد بسته‌هایی که می‌توان در مدت زمان تأخیر انتشار ارسال کرد.

روش توقف و انتظار (stop-and-wait ARQ)

بهره‌وری پروتکل ARQ توقف و انتظار در حالت واقعی (با وجود احتمال خطا)



ارسال موفقیت‌آمیز بعداز k بار ارسال:

$$t_{total} = (k-1) \times t_{out} + t_0$$

احتمال موفقیت بعداز k بار ارسال:

$$Pr[K = k] = P_F^{(k-1)}(1 - P_F)$$

$$E[t_{total}] = \sum_{k=1}^{\infty} \left(((k-1) \times t_{out} + t_0) \times P_F^{(k-1)}(1 - P_F) \right)$$

بهترین زمان زمانبند $t_{out} = t_0$ است، بنابر این:

$$E[t_{total}] = t_0 \times \sum_{k=1}^{\infty} k \times P_F^{(k-1)}(1 - P_F) = \frac{t_0}{1 - P_F}$$

روش توقف و انتظار (stop-and-wait ARQ)

بهره‌وری پروتکل ARQ توقف و انتظار در حالت واقعی (با وجود احتمال خطا)

$$E[t_{total}] = \frac{t_0}{1 - P_F}$$

$$R_{eff} = \frac{D}{E[t_{total}]} = \frac{L - H}{t_0}(1 - P_F) = R_{eff}^0(1 - P_F)$$

$$u_{s\&w} = \frac{R_{eff}}{R} = \frac{R_{eff}^0}{R}(1 - P_F) = u_{s\&w}^0(1 - P_F)$$

$$u_{s\&w} = \frac{1 - \frac{H}{L}}{1 + \frac{A}{L} + \frac{2(t_{prop} + t_{proc})R}{L}}(1 - P_F)$$

عامل سربار سرآیند

عامل خطا

عامل سربار بسته کنترلی تاییده

عامل حاصلضرب تأخیر در پهنای‌باند $(2a)$

روش توقف و انتظار (stop-and-wait ARQ)

بهره‌وری پروتکل stop-and-wait ARQ درحالت بدون خطا:

مثال: فرض کنید $L = 1500$ Bytes ، $H = A = 20$ Bytes ، $RTT = 0.1,1,10,100$ msec و $R = 1, 10, 100, 1000$ Mbps باشد،

آنگاه بهره‌وری پروتکل ARQ توقف و انتظار در حالت بدون خطا برابر است با:

$u^0_{s\&w}$				
R \ RTT	0.1 msec	1 msec	10 msec	100 msec
1 Mbps	96.57%	89.97%	53.43%	10.56%
10 Mbps	89.97%	53.43%	10.56%	1.17%
100 Mbps	53.43%	10.56%	1.17%	0.12%
1000 Mbps	10.56%	1.17%	0.12%	0.01%

روش توقف و انتظار (stop-and-wait ARQ)

بهره‌وری پروتکل stop-and-wait ARQ درحالت بدون خطا:

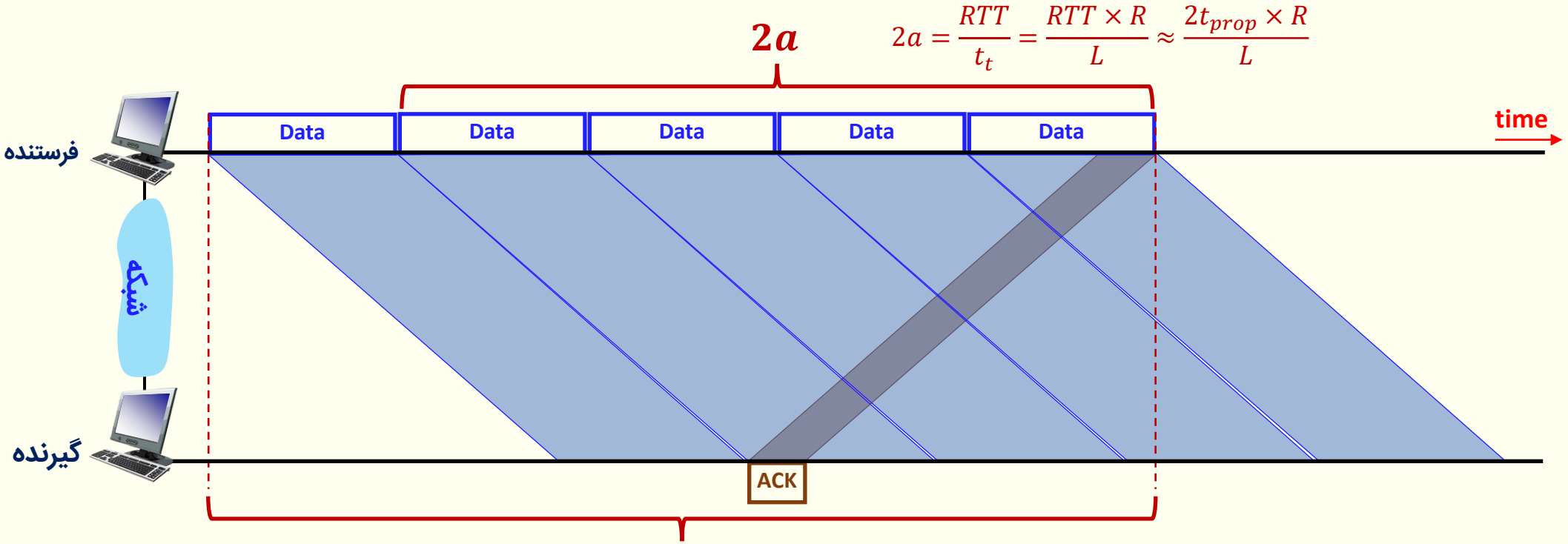
مثال: فرض کنید $L = 1500$ Bytes ، $H = A = 20$ Bytes ، $RTT = 0.1$ msec ، $R = 1, 10, 100, 1000$ Mbps و $P_F = 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}$ باشد، آنگاه بهره‌وری پروتکل ARQ توقف و انتظار در حالت بدون خطا برابر است با:

$u^0_{s\&w}$				
R \ P_F	10^{-6}	10^{-5}	10^{-4}	10^{-3}
1 Mbps	96.57%	96.57%	96.56%	96.48%
10 Mbps	89.97%	89.97%	89.96%	89.88%
100 Mbps	53.43%	53.43%	53.42%	53.38%
1000 Mbps	10.56%	10.56%	10.56%	10.55%

روش‌های ARQ خط لوله (pipeline)

عامل اصلی کاهش بهره‌وری پروتکل ARQ توقف و انتظار: عدم ارسال بسته در زمان انتظار برای دریافت تاییده

$$2a = \frac{RTT}{t_t} = \frac{RTT \times R}{L} \approx \frac{2t_{prop} \times R}{L}$$



$$t_0 = t_t + t_a + RTT$$

$$t_0 \approx (2a + 1)t_t$$

پروتکل ARQ بازگشت به عقب به اندازه N (Go-back-N ARQ)

فرستنده:

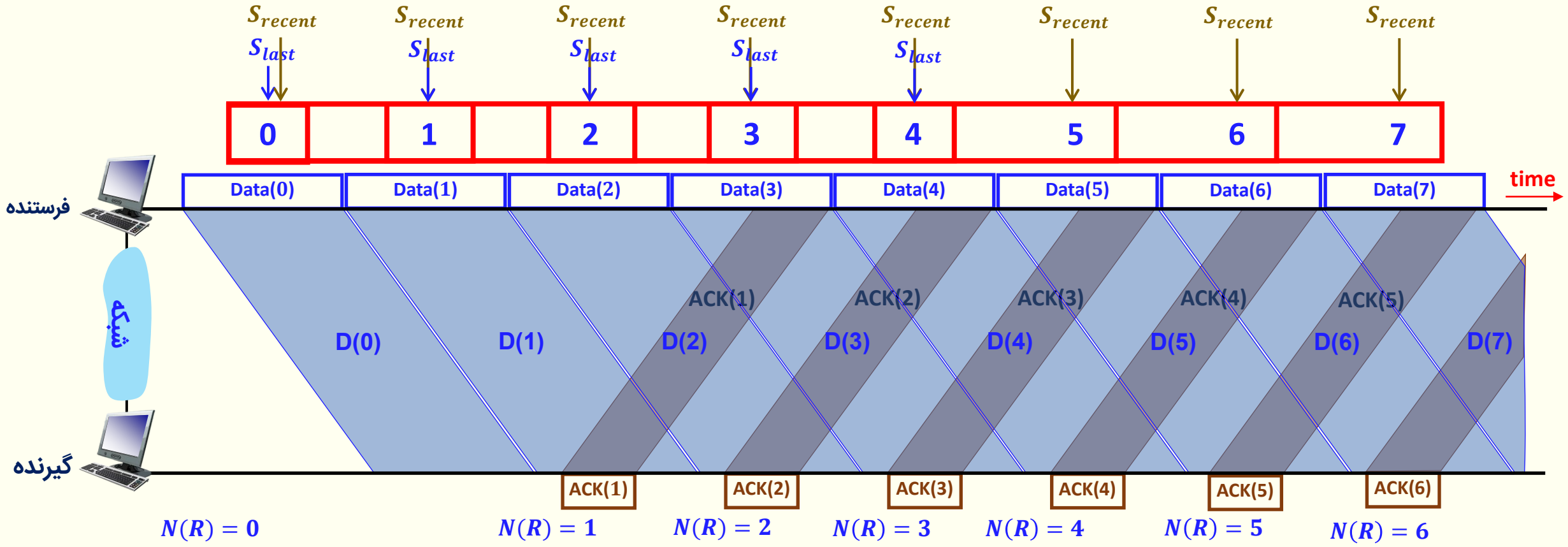
- عدم توقف ارسال بسته‌ها در زمان انتظار برای دریافت تاییده
- تخصیص بافر ارسال با ظرفیت N بسته و نگهداری همه بسته‌های ارسال شده تا دریافت تاییده آن‌ها (حداکثر امکان ارسال N بسته بدون دریافت تاییده وجود دارد)
- در صورت دریافت تاییده بسته تایید شده از بافر حذف و امکان ارسال بسته جدید فراهم می‌شود.
- با توجه به اینکه گیرنده بسته‌ها را به ترتیب دریافت می‌کند. در صورت عدم دریافت تاییده یک بسته در زمان مقرر، آن بسته و تمام بسته‌های ارسال شده بعد از آن مجدداً ارسال می‌شوند.
- نیاز به دو شماره ترتیب:
  - S_{last}: قدیمی‌ترین بسته ارسال شده و هنور تایید نشده
  - S_{recent}: جدیدترین بسته ارسال شده

گیرنده:

- منتظر دریافت بسته با شماره ترتیب مورد انتظار ($N(R)$)
- در صورت دریافت صحیح ارسال تاییده و بروزرسانی شماره ترتیب مورد انتظار
- درصورت دریافت بسته با شماره خارج از ترتیب، بسته دریافتی حذف می‌شود.

پروتکل ARQ بازگشت به عقب به اندازه N (Go-back-N ARQ)

سناریو بدون خطا: پنجره ارسال برابر ۴ است ($W_s = 4$)



پروتکل ARQ بازگشت به عقب به اندازه N (Go-back-N ARQ)

بهره‌وری پروتکل Go-back-N ARQ در حالت بدون خطا

- پنجره ارسال به اندازه کافی بزرگ است که ارسال قطع نمی‌شود ($W_s \geq 2a + 1$ یا $W_s \geq \frac{t_0}{t_t}$).
- گیرنده در هر t_t ثانیه یک بسته دریافت می‌کند.

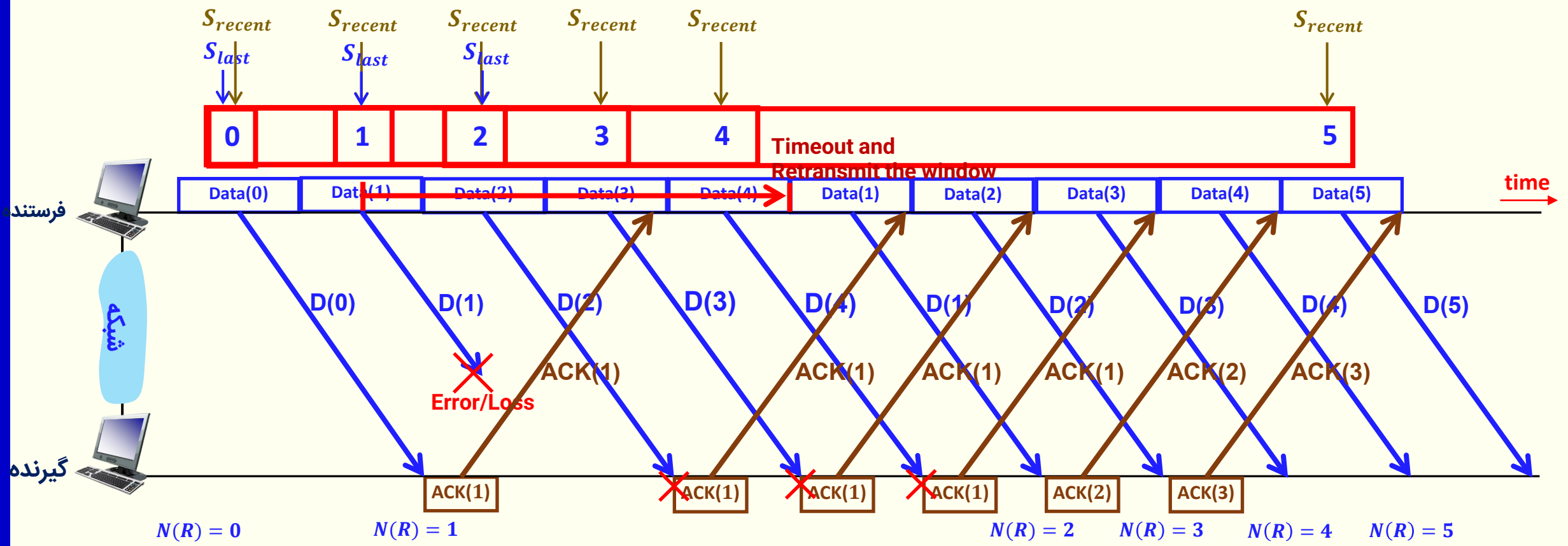
$$R_{eff}^{0} = \frac{D}{t_t} = \frac{L - H}{\frac{L}{R}} = \left(1 - \frac{H}{L}\right) R$$

$$u_{GBN}^{0} = \frac{R_{eff}^{0}}{R} = \left(1 - \frac{H}{L}\right)$$

عامل سربار سرآیند

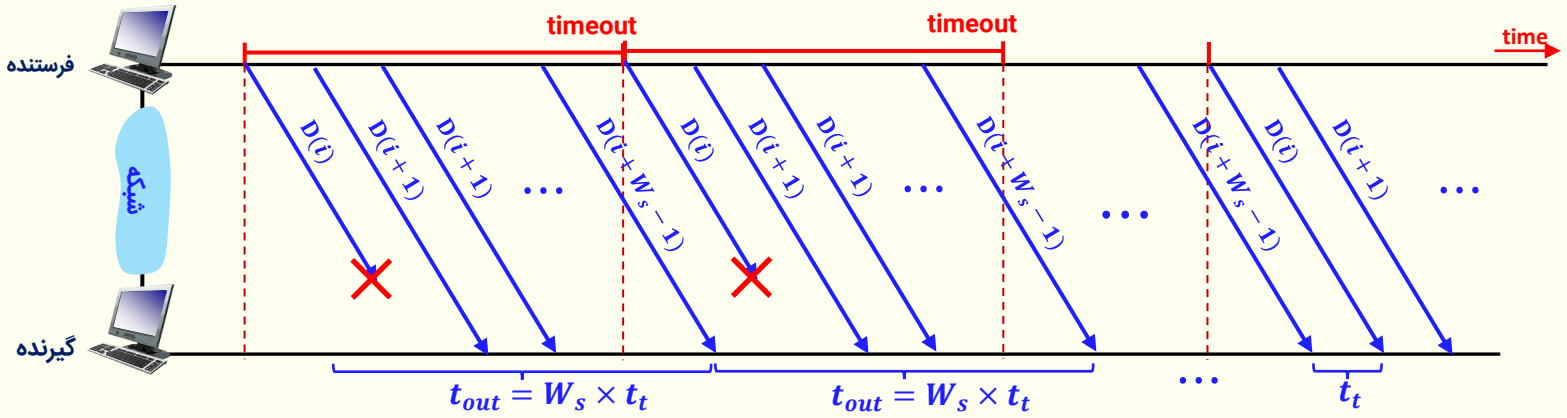
پروتکل ARQ بازگشت به عقب به اندازه N (Go-back-N ARQ)

سناریو با وجود احتمال خطا:



پروتکل ARQ بازگشت به عقب به اندازه N (Go-back-N ARQ)

بهره‌وری پروتکل Go-back-N ARQ در حالت واقعی (با وجود احتمال خطا)



- پنجره ارسال به اندازه کافی بزرگ است که ارسال قطع نمی‌شود ($W_s \geq 2a + 1$ یا $W_s \geq \frac{t_0}{t_t}$).
- درصورت خطا فرستنده مجبور است W_s بسته را مجدداً ارسال کند.

$$t_{GBN} = E[t_{total}] = t_t(1 - P_F) + \left(t_t + \frac{W_s t_t}{1 - P_F}\right) P_F = \left(\frac{1 + (W_s - 1)P_F}{1 - P_F}\right) t_t$$

$$R_{eff} = \frac{D}{t_{GBN}} = \frac{L - H}{\left(\frac{1 + (W_s - 1)P_F}{1 - P_F}\right) t_t}(1 - P_F) = \frac{1 - \frac{H}{L}}{(1 + (W_s - 1)P_F)}(1 - P_F)R$$

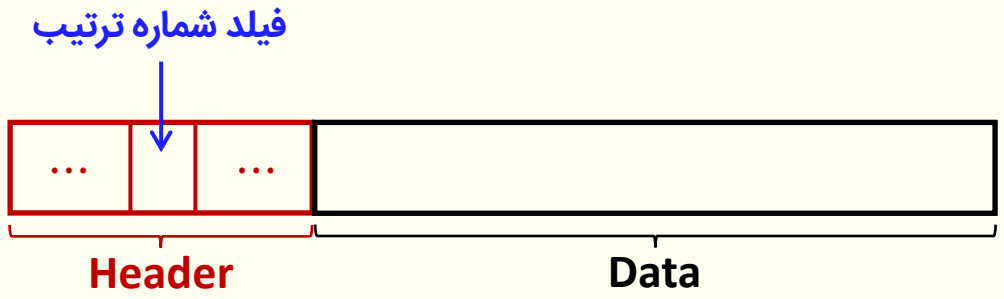
$$u_{GBN} = \frac{R_{eff}}{R} = \frac{1 - \frac{H}{L}}{(1 + (W_s - 1)P_F)}(1 - P_F)$$

عامل سربار سرآیند

عامل حاصلضرب تأخیر در پهنای‌باند ($2a$)

عامل خطا

پروتکل ARQ بازگشت به عقب به اندازه N (Go-back-N ARQ)



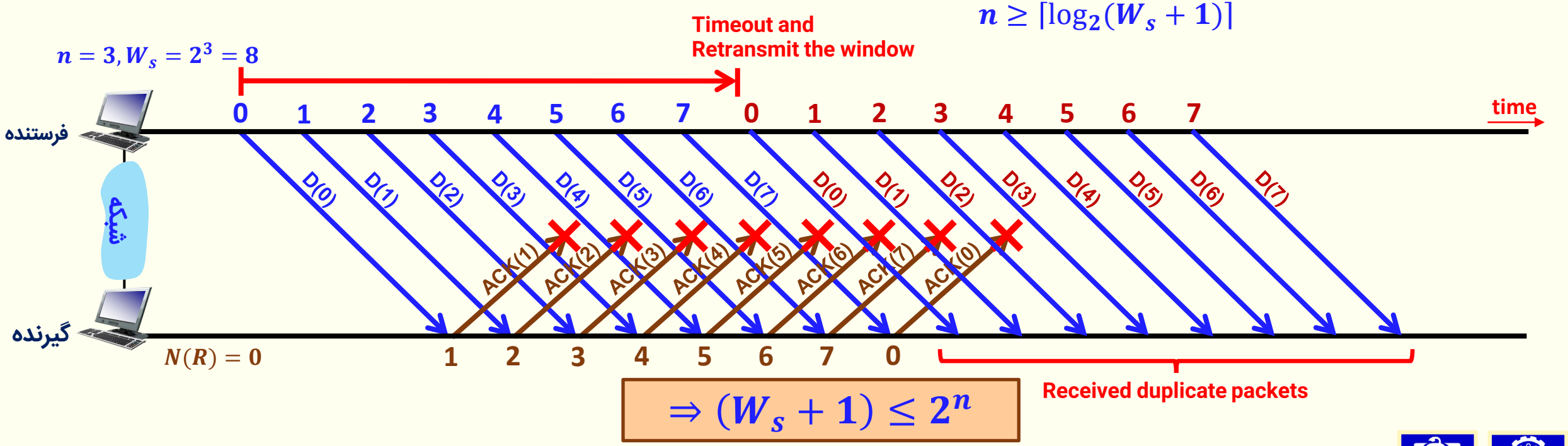
تعیین تعداد بیت‌های شماره ترتیب (و شماره تاییده):

- اگر پنجره ارسال W_s و تعداد بیت‌های شماره ترتیب n باشد، آنگاه

$$(W_s + 1) \leq 2^n$$

$$W_s \leq (2^n - 1)$$

$$n \geq \lceil \log_2(W_s + 1) \rceil$$



$$\Rightarrow (W_s + 1) \leq 2^n$$

لایه انتقال

پروتکل ARQ تکرار انتخابی (selective repeat ARQ)

پروتکل Go-back-N ARQ:

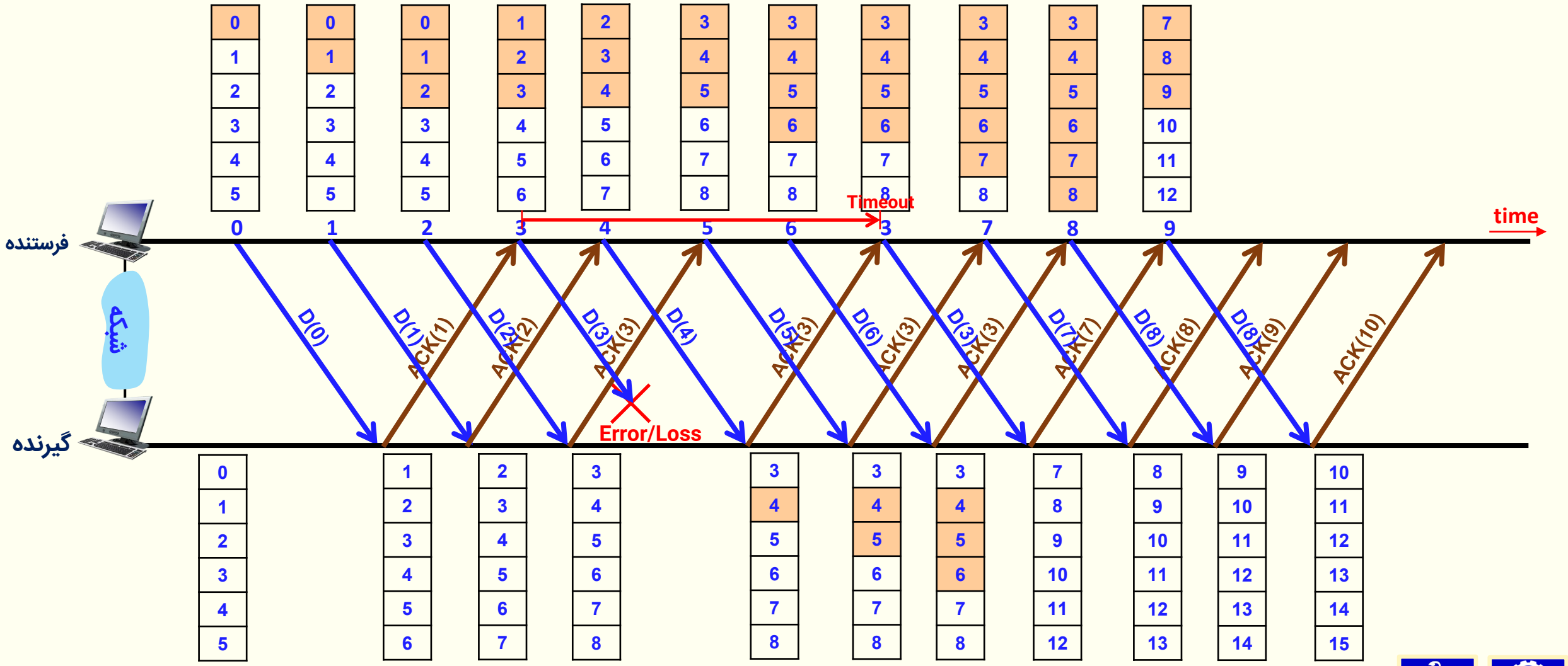
- مزیت: فرستنده بدون توقف همواره در حال ارسال است (با انتخاب پنجره ارسال مناسب)
- معایب: به ازای هر خطا کل پنجره ارسال باید مجدداً ارسال شود (تمام بسته‌ها بعداز بسته از دست رفته باید مجدداً ارسال شوند.
  - گیرنده فقط به ترتیب دریافت می‌کند.

پروتکل selective repeat ARQ:

- همانند پروتکل Go-Back-N ARQ فرستنده پنجره ارسال دارد و بدون وقفه به اندازه پنجره ارسالش داده‌ها را ارسال می‌کند.
- گیرنده قادر است بسته‌های بعداز که بسته از دست رفته را دریافت و نگهداری کند (داشتن بافر یا پنجره دریافت در گیرنده)
- اگر بسته‌ای از بین برود، فقط همان بسته مجدداً ارسال می‌شود (تکرار انتخابی).
- گیرنده به محض دریافت بسته از دست رفته (پر شدن گپ) می‌تواند داده‌ها (آن بسته و بسته‌های نگهداری شده در پنجره دریافت) را به لایه بالاتر تحویل دهد.

لایه انتقال

پروتکل ARQ تکرار انتخابی (selective repeat ARQ)



پروتکل ARQ تکرار انتخابی (selective repeat ARQ)

بهره‌وری پروتکل selective repeat ARQ در حالت بدون خطا:

- پنجره ارسال به اندازه کافی بزرگ است که ارسال قطع نمی‌شود.
- گیرنده در هر t_t ثانیه یک بسته دریافت می‌کند.

$$R_{eff}^{0} = \frac{D}{t_t} = \frac{L-H}{\frac{L}{R}} = \left(1-\frac{H}{L}\right)R$$

$$u_{SR}^{0} = \frac{R_{eff}^{0}}{R} = \left(1-\frac{H}{L}\right)$$

بهره‌وری پروتکل selective repeat ARQ در حالت واقعی (با وجود احتمال خطا):

- پنجره ارسال به اندازه کافی بزرگ است که ارسال قطع نمی‌شود.
- گیرنده در هر t_t ثانیه یک بسته دریافت می‌کند.

$$t_{SR} = E[t_{total}] = E[Kt_t] = E[K]t_t = \frac{t_t}{1-P_F}$$

$$R_{eff} = \frac{D}{t_{SR}} = \frac{L-H}{\frac{t_t}{1-P_F}} = \left(1-\frac{H}{L}\right)(1-P_F)R$$

$$u_{SR} = \frac{R_{eff}}{R} = \left(1-\frac{H}{L}\right)(1-P_F)$$

عامل سربار سرآیند

عامل خطا

پروتکل ARQ تکرار انتخابی (selective repeat ARQ)

تعیین تعداد بیت‌های شماره ترتیب (و شماره تاییده):

- اگر پنجره ارسال W_S ، پنجره دریافت W_R و تعداد بیت‌های شماره ترتیب n باشد،

آنگاه:

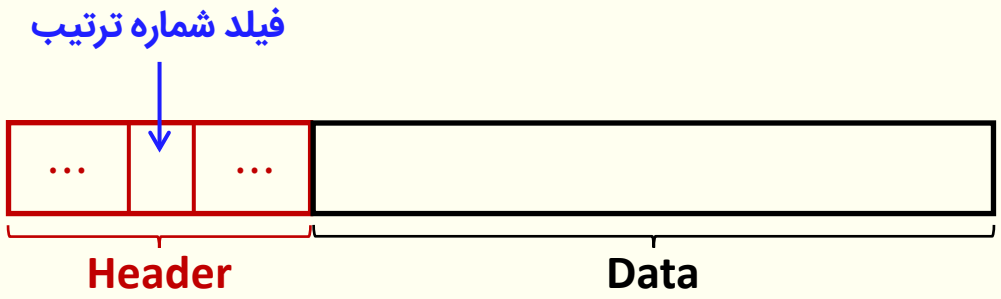
$$(W_s + W_R) \leq 2^n$$

اگر

$$W_s = W_R$$

آنگاه

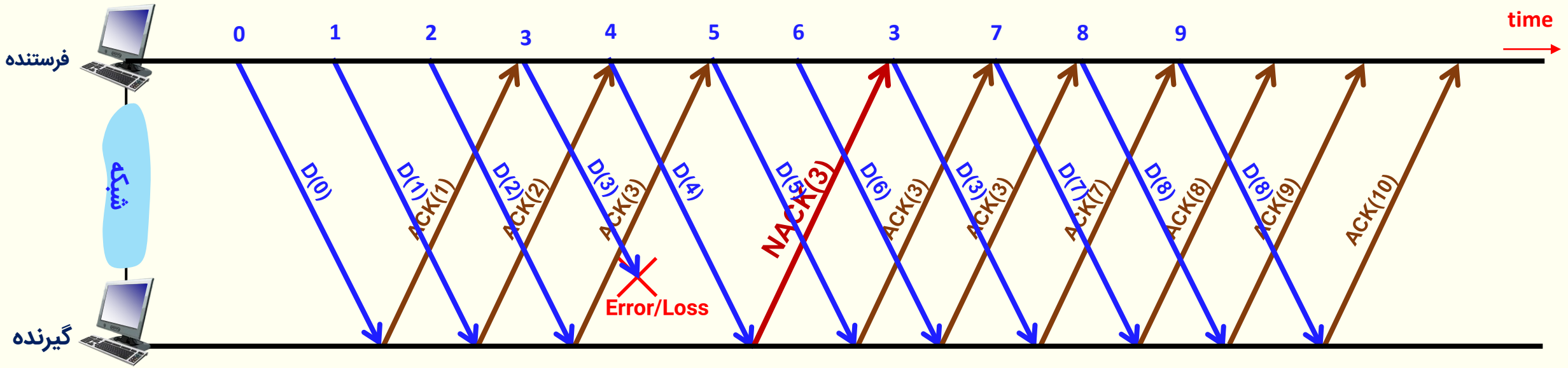
$$W_s \leq 2^{n-1}$$



پروتکل ARQ تکرار انتخابی (selective repeat ARQ)

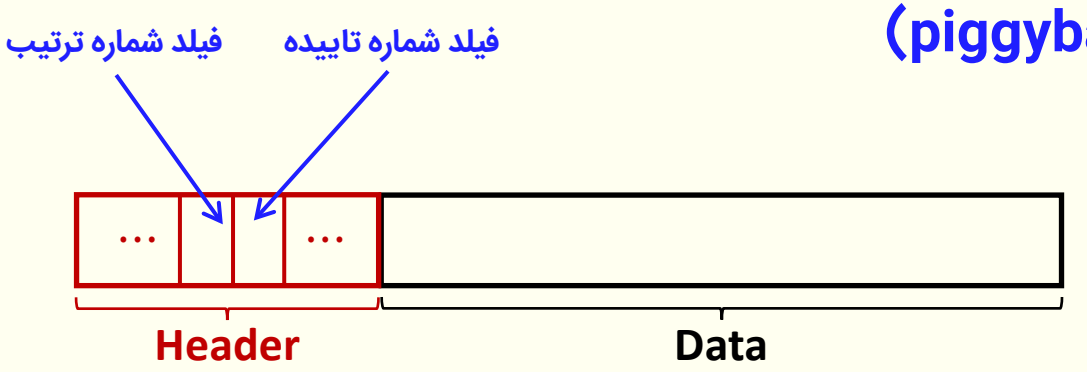
ارسال سریع (Fast Retransmission):

- تشخیص زودهنگام قبل از منقضی‌شدن زمانبند
- دریافت تاییده منفی (گیرنده مطمئن است که بسته از بین رفته است)
- دریافت سه تاییده تکراری در روش کنترل خطای پروتکل TCP

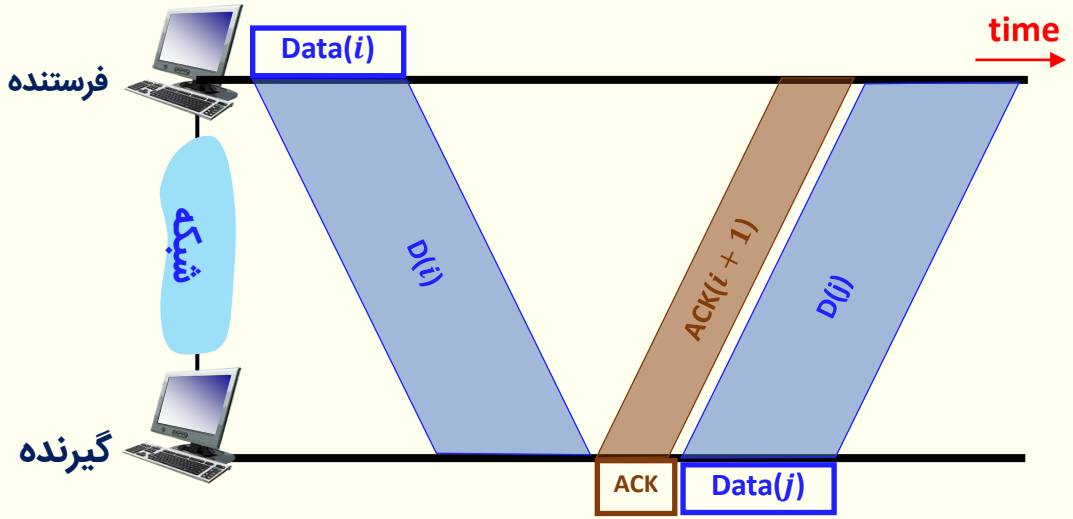


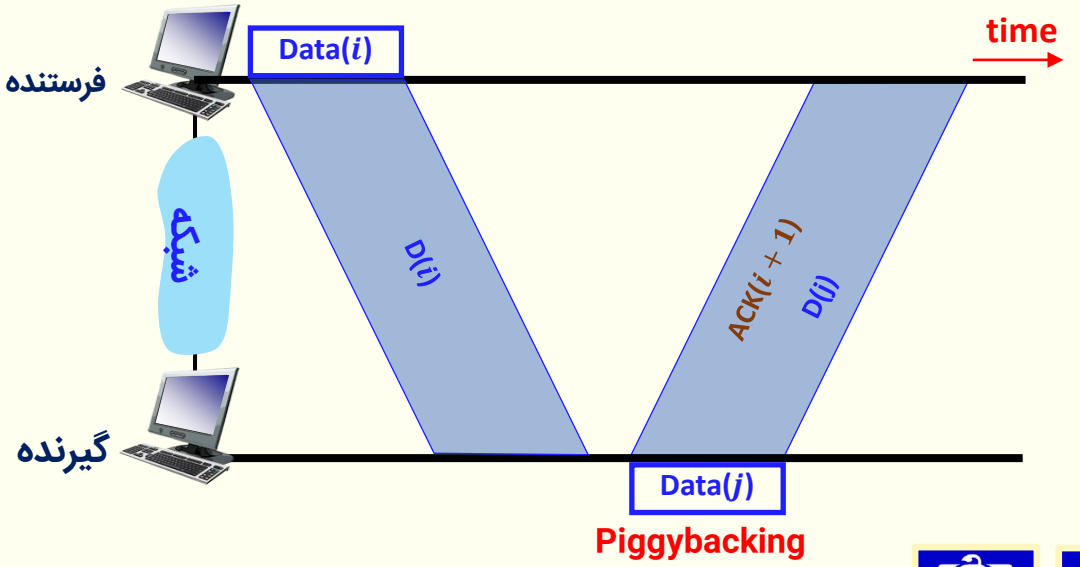
پروتکل ARQ تکرار انتخابی (selective repeat ARQ)

ارسال اطلاعات کنترلی همراه با داده‌ها (سواری مجانی یا piggybacking)



- ارتباطات دو طرفه:
  - ارسال داده‌ها از فرستنده به گیرنده (ارسال درخواست)
  - دریافت داده‌ها از گیرنده توسط فرستنده (دریافت پاسخ)
- Piggybacking:
  - ارسال اطلاعات کنترلی همراه با داده‌ها (نپرداختن هزینه برای ارسال اطلاعات کنترلی)



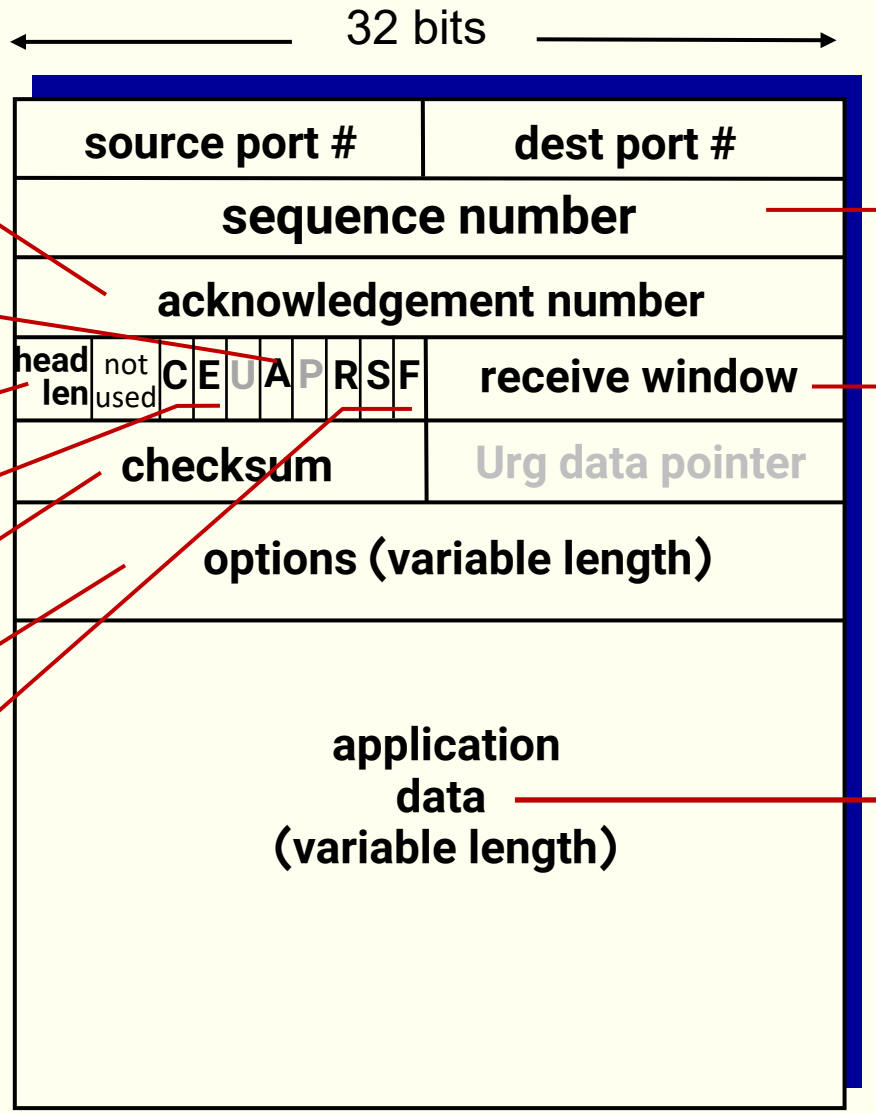


پروتکل TCP: کلیات (پروتکل کنترل ارسال - Transmission Control Protocol)

- پروتکل کنترل ارسال (Transmission Control Protocol):
  - RFCهای شماره ۷۹۳، ۱۱۲۲، ۲۰۱۸، ۵۶۸۱، ۷۳۲۳
- نقطه به نقطه (point-to-point):
  - یک فرستنده، یک گیرنده
- کنترل خطا (ارسال مطمئن):
  - ارسال بدون خطا و حفظ ترتیب رشته بایت‌ها
  - مرزبندی پیام‌ها توسط لایه کاربرد باید انجام شود.
- ارتباط دوطرفه کامل:
  - ایجاد دو اتصال همزمان:
    - از سمت سرویس‌گیرنده به سرویس‌دهنده
    - از سمت سرویس‌دهنده به سرویس‌گیرنده
- تعیین حداکثر اندازه سگمنت‌ها (MSS: Maximum Segment Size)
- اتصال‌گرا:
  - انجام دست‌دهی (تبادل سگمنت‌های کنترلی برای برقراری اتصال) قبل از تبادل داده‌ها
  - تبادل داده‌ها
  - تبادل سگمنت‌های کنترلی و بستن اتصال در خاتمه تبادل داده‌ها
- ارسال تاییده تجمعی
- کنترل نرخ ارسال با تنظیم پنجره ارسال به منظور:
  - کنترل جریان
  - کنترل ازدحام

لایه انتقال

پروتکل TCP: فرمت (ساختار) سگمنت‌های TCP



شماره تاییده: شماره بایت مورد انتظار که به صورت piggybacking منتقل شده است.

بیت A اعتبار فیلد شماره تاییده را مشخص می‌کند.

طول سرآیند سگمنت TCP

C و B برای اعلان ازدحام استفاده می‌شوند.

چک بیت‌های جمع

فیلدهای اختیاری

RST، SYN و FIN برای مدیریت اتصال استفاده می‌شوند.

شماره ترتیب ارسال: شماره ترتیب بر روی بایت‌ها است (نه بر روی سگمنت‌ها)

این فیلد برای کنترل جریان استفاده می‌شود. تعداد بایت‌هایی که گیرنده می‌تواند دریافت کند.

داده‌های دریافتی از لایه بالاتر برای ارسال از طریق سوکت

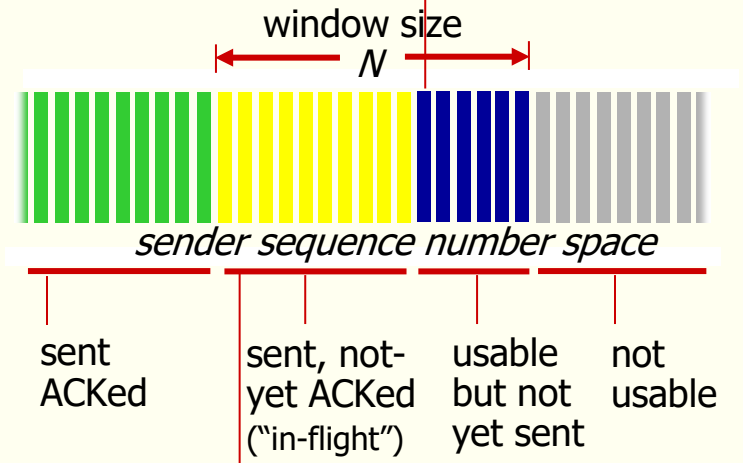
پروتکل TCP: شماره ترتیب و شماره تاییده

- **شماره ترتیب:**
  - شماره ترتیب بر روی بایت‌ها است.
  - شماره ترتیب در سرآیند سگمنت TCP، شماره اولین بایت از داده‌های حمل شده توسط آن سگمنت است.

- **شماره تاییده:**
  - شماره ترتیب بایتی است که گیرنده منتظر دریافت آن است.
  - تاییده تجمعی است. یعنی گیرنده تا بایت با شماره یکی قبل از شماره تاییده را دریافت کرده است.
  - شماره تاییده در صورتی اعتبار دارد که بیت A یک باشد.

سگمنت ارسالی توسط فرستنده

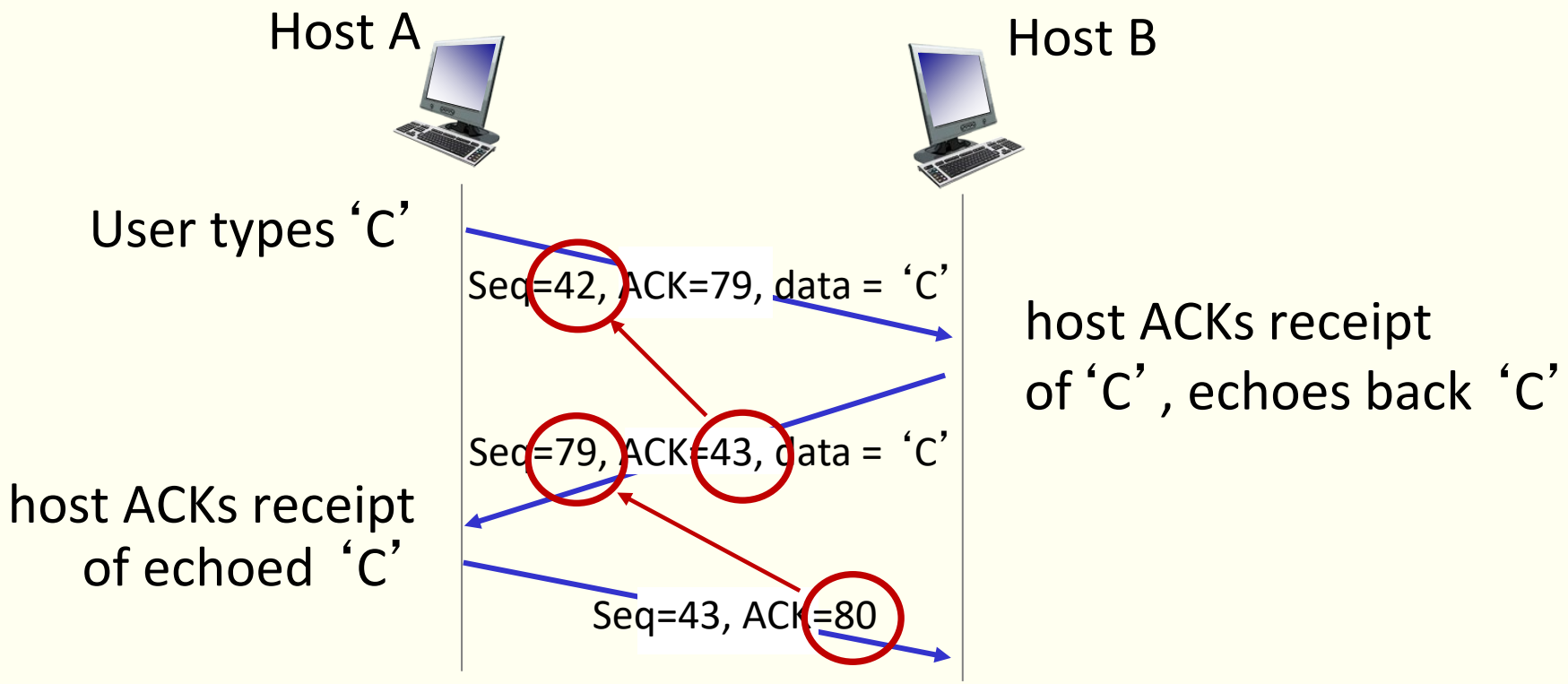
source port #	dest port #
sequence number	
acknowledgement number	
	rwnd
checksum	urg pointer



سگمنت ارسالی توسط گیرنده

source port #	dest port #
sequence number	
acknowledgement number	
A	rwnd
checksum	urg pointer

پروتکل TCP: شماره ترتیب و شماره تاییده



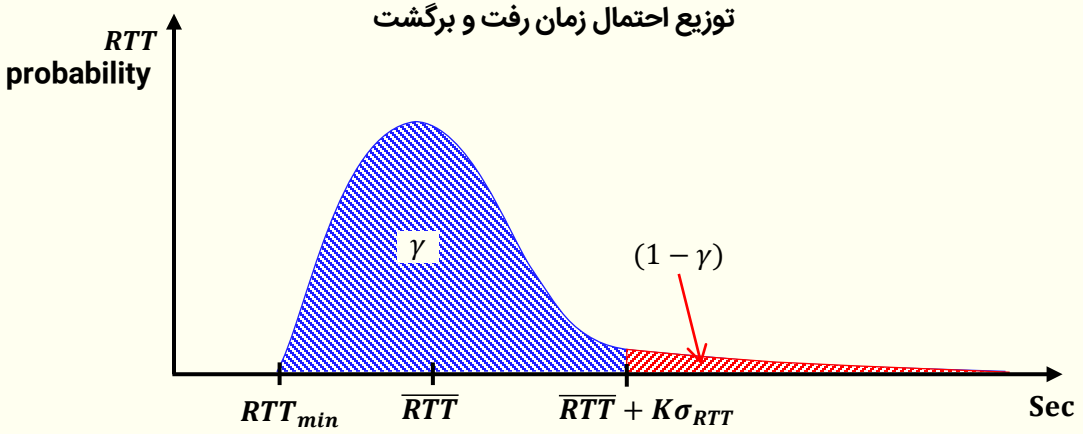
simple telnet scenario

پروتکل TCP: کنترل خطا

- **روش کنترل خطای TCP:**
  - حالت خاصی از پروتکل ARQ تکرار انتخابی (selective repeat ARQ)
- **شماره ترتیب بر روی بایت‌ها**
- **فقط دارای یک زمانبند:**
  - برای قدیمی‌ترین سگمنت که هنوز تاییده آن نیامده است.
- **تنظیم پویای زمان t_{out}:**
  - تخمین زمان رفت و برگشت با استفاده از روش میانگین متحرک وزن‌دار نمایی را روی نمونه‌های زمان رفت و برگشت اندازه‌گیری شده
  - استفاده از الگوریتم Karn
- **تشخیص از دست رفتن سگمنت و ارسال مجدد:**
  - منقضی شدن زمانبند
  - دریافت ۳ تاییده اضافه (۳ تاییده تکراری)
  - ارسال سریع (fast retransmission) قبل از منقضی شدن زمانبند
- **ارسال تاییده تجمعی:**
  - شماره تاییده بایت مورد انتظار دریافت گیرنده را مشخص می‌کند. یعنی تا یکی کمتر از شماره ترتیب مورد تایید است.
- **ارسال تاییده به صورت piggybacking**
- **خط مشی ارسال تاییده براساس وضعیت داده‌های دریافت شده**
  - ارسال سریع تاییده درصورت دریافت داده‌های خارج از ترتیب

پروتکل TCP: تنظیم زمان t_{out}

- **اتصال TCP:**
  - اتصال بین سرویس‌گیرنده و سرویس‌دهنده
  - بی‌اطلاعی TCP از تأخیر رفت و برگشت بین سرویس‌گیرنده و سرویس‌دهنده
  - کمتر از ۱ میلی ثانیه، درصورتی که سرویس‌گیرنده و سرویس‌دهنده روی یک شبکه محلی باشند.
  - تا چند صدثانیه، درصورتی که سرویس‌گیرنده و سرویس‌دهنده روی شبکه گسترده باشند.
- **تنظیم زمان t_{out}:**
  - زمان t_{out} باید متناسب با زمان رفت و برگشت تنظیم شود.
    - اگر زمان t_{out} بسیار بزرگ تنظیم شود، در صورت از دست دادن سگمنت داده فرستنده دیرهنگام متوجه شده و پهنای باند ارسال را از دست می‌دهد.
    - اگر زمان t_{out} بسیار کوچک تنظیم شود، قبل از دریافت تاییده زمانبد منقضی شده و ارسال مجدد بیهوده انجام می‌شود.



بهترین زمان t_{out}:

$$TimeoutInterval = \overline{RTT} + K\sigma_{RTT}$$

سوال: چگونه $\overline{RTT}$ و σ_{RTT} را محاسبه کنیم؟

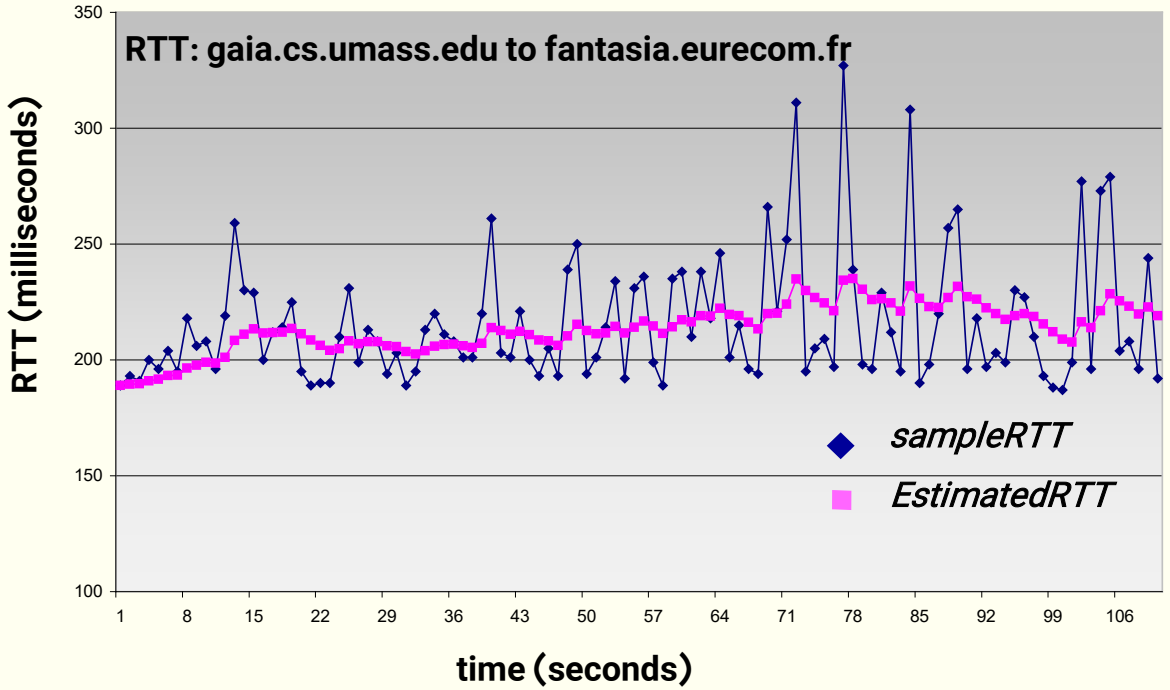
پروتکل TCP: تنظیم زمان t_{out}

تخمین میانگین زمان رفت و برگشت:

- تخمین بر اساس نمونه‌های زمان رفت و برگشت اندازه‌گیری شده:
  - مدت زمان اندازه‌گیری شده از زمان ارسال سگمنت و دریافت تاییده ($SampleRTT$) (برای ارسال مجددها اندازه گیری نمی‌شود)
- استفاده از روش میانگین متحرک وزن‌دار نمایی (EWMA) (Exponential Weighted Moving Average)
  - تأثیر نمونه‌های قدیمی به صورت نمایی کاهش می‌یابد.

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

$$\alpha = 1/8 = 0.125$$



پروتکل TCP: تنظیم زمان t_{out}

تخمین میانگین زمان رفت و برگشت:

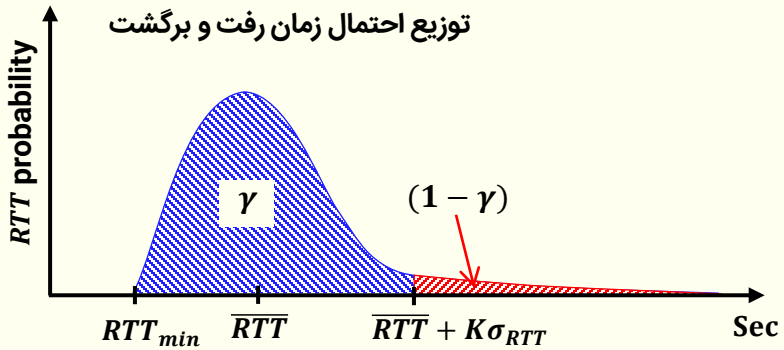
$$EstimatedRTT = (1 - \alpha) \times EstimatedRTT + \alpha \times SampleRTT$$

(Typically $\alpha = 1/8 = 0.125$)

تخمین انحراف معیار زمان رفت و برگشت:

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

(Typically $\beta = 1/4 = 0.25$)



$$TimeoutInterval = \overline{RTT} + K\sigma_{RTT}$$

تنظیم زمان t_{out}:

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

پروتکل TCP: تنظیم زمان t_{out}

الگوریتم Karn برای تنظیم زمان t_{out}:

- بعداز هر منقضی شدن زمانبند (timeout) مقدار زمان زمانبند دو برابر می‌شود.
  - منقضی شدن زمانبند مترداف است با از دست دادن بسته در یکی از گره‌ها شبکه. از دست دادن بسته مترداف است با وجود ازدحام در شبکه. وجود ازدحام در شبکه مترداف است با افزایش تأخیر شبکه. با افزایش تأخیر شبکه، زمان t_{out} نیز باید افزایش یابد.
- اندازه‌گیری زمان ارسال سگمنت و دریافت تاییده ($SampleRTT$) برای سگمنت‌های ارسال مجدد شده انجام نمی‌شود.
  - اگر سگمنتی ارسال مجدد شود زمان اندازگیری شده بین ارسال سگمنت و دریافت تاییده ابهام دارد.
- بعداز هر نمونه زمان رفت و برگشت اندازه‌گیری شده زمان t_{out} برای اساس تخمین‌های بروز شده میانگین زمان رفت و برگشت و انحراف معیار تنظیم می‌شود.

لایه انتقال

پروتکل TCP:

TCP فرستنده:

- رخداد: دریافت داده از برنامه کاربردی (لایه بالاتر):
  - ایجاد سگمنت با شماره ترتیب (بعدی)
  - شماره ترتیب بر حسب بایت است و شماره ترتیب سگمنت شماره اولین بایت داده‌های سگمنت است.
  - اگر زمانبند روشن نیست، زمانبند روشن شود.
    - زمانبند مربوط به قدیمی‌ترین سگمنت هنوز تایید نشده است.
    - زمان زمانبند با مقدار t_{out} تنظیم شود ($TimeoutInterval$).
- رخداد: منقضی زمانبند (timeout):
  - ارسال مجدد سگمنتی که زمانبند برای آن روشن شده بود.
  - روشن کردن مجدد زمانبند (با زمان $TimeoutInterval$ محاسبه شده با الگوریتم Karn).
- رخداد: دریافت تاییده:
  - اگر تاییده سگمنتی را تایید می‌کند که هنوز تایید نشده است:
    - بروز رسانی نشانگر داده‌های تایید شده
    - اگر هنوز سگمنت ارسال شده‌ای که هنوز تایید نشده وجود دارد، زمانبند با مقدار t_{out} مجدداً روشن شود.
  - اگر تاییده سگمنتی را تایید می‌کند که قبلاً تایید شده است:
    - مقدار شمارنده دریافت تاییده تکراری یک واحد افزایش یابد و اگر مقدار این شمارنده برابر ۳ شده است:
      - قدیمی‌ترین سگمنتی که هنوز تایید نشده است مجدداً ارسال شود.
      - زمانبند با مقدار t_{out} مجدداً روشن شود.

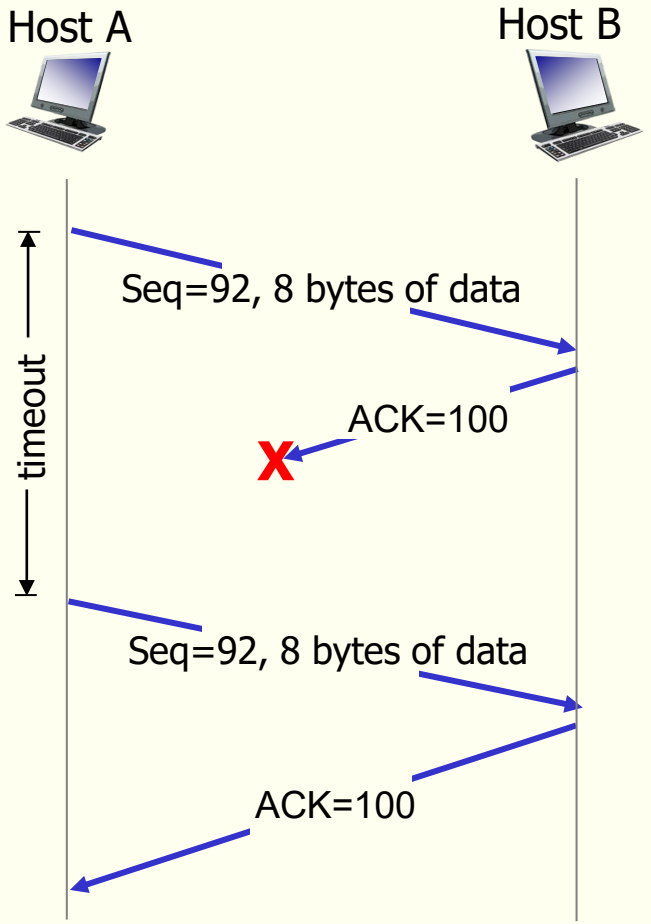
پروتکل TCP

فرستنده TCP:

```
NextSeqNum = InitialSeqNumber
SendBase = InitialSeqNumber
loop (forever) {
        switch (event)
                event: data received from application above
                        create TCP segment with sequence number NextSeqNum
                        if (timer currently not running)
                                start timer
                        pass segment to IP
                        NextSeqNum = NextSeqNum + length(data)
                        break;
                event: timer timeout
                        retransmit not-yet-acknowledged segment withmsmallest sequence number
                        start timer
                        break;
                event: ACK received, with ACK field value of y
                        if (y > SendBase) {
                                SendBase=y
                        if (there are currently any not-yet-acknowledged segments)
                                start timer
                        }
                        else {/* a duplicate ACK for already ACKed segment */
                                increment number of duplicate ACKs received for y
                                if (number of duplicate ACKS received for y==3)
                                /* TCP fast retransmit */
                                resend segment with sequence number y
                        }
                        break;
} /* end of loop forever */
```

پروتکل TCP

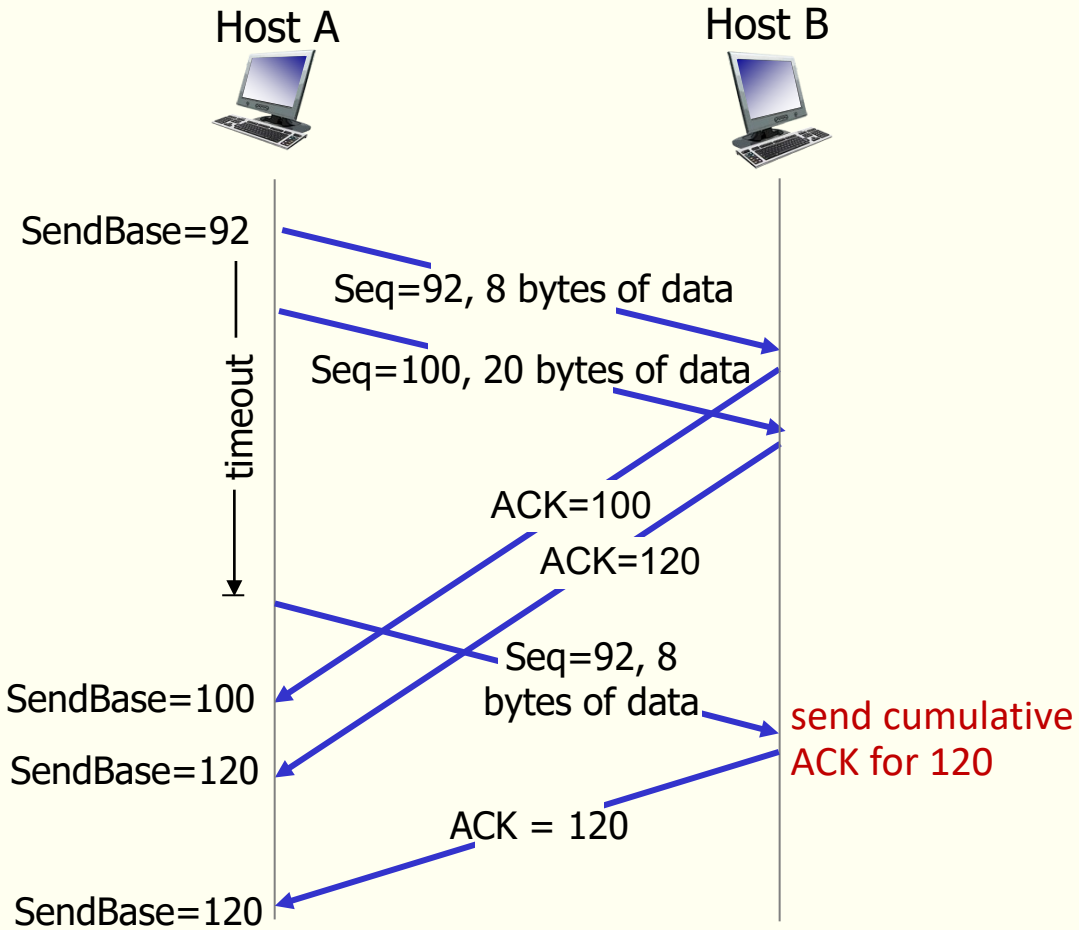
سناریورهای ارسال مجدد:



سناریو از دست‌دادن تاییده

پروتکل TCP

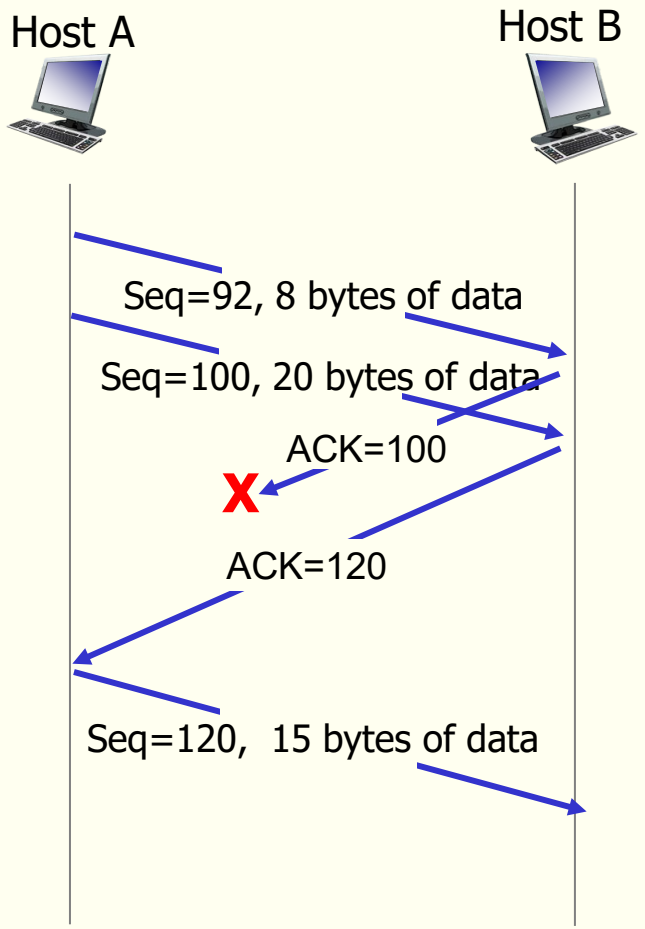
سناریورهای ارسال مجدد:



سناریو منقضی‌شدن زمان زمانبند (timeout)

پروتکل TCP

سناریورهای ارسال مجدد:



تاییده تجمعی ازدست‌رفتن تاییده قبلی را جبران می‌کند.

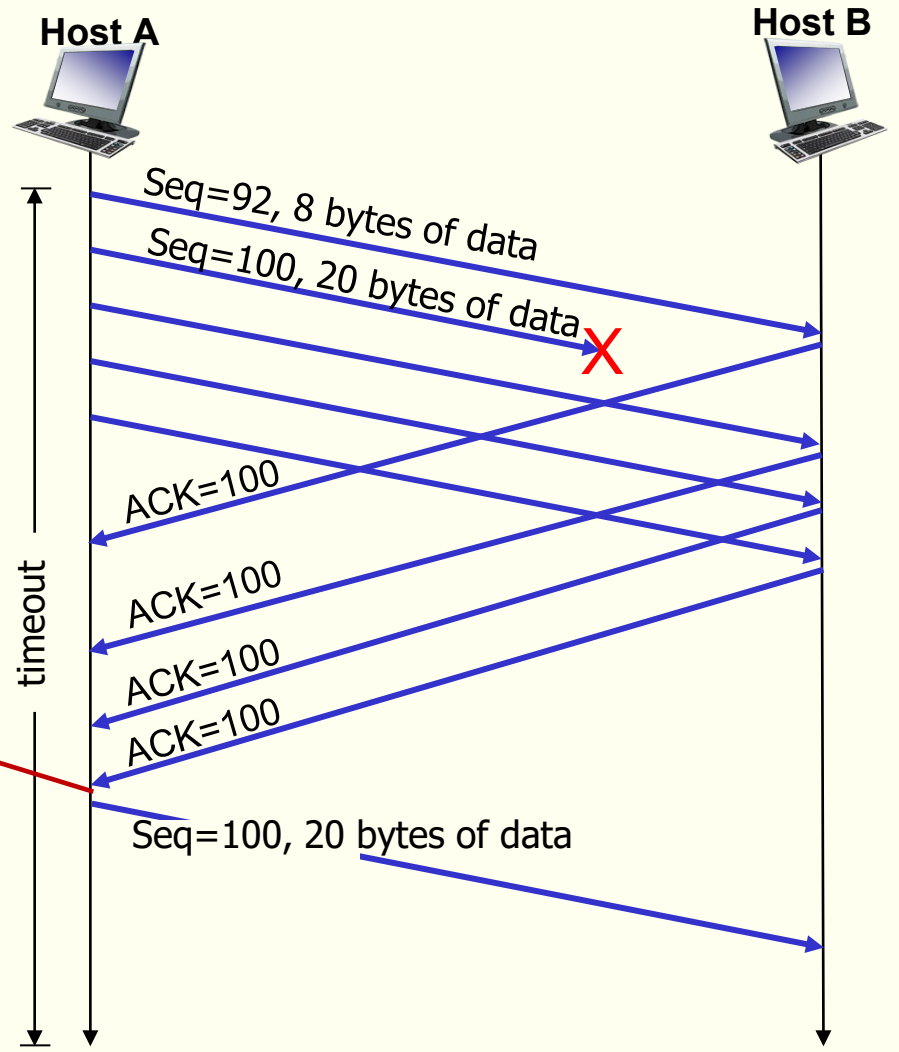
پروتکل TCP

سناریوهای ارسال مجدد:

ارسال سریع TCP

درصورتی که فرستنده ۳ تاییده تکراری برای یک داده یکسان دریافت کند ("triple duplicate ACKs") سگمنت با کوچک‌ترین شماره ترتیب تاییدنشده را مجدداً ارسال می‌کند.

- فرض بر این است که سگمنت تایید نشده ازدست‌رفته و منتظر منقضی‌شدن زمانبند نمی‌ماند.



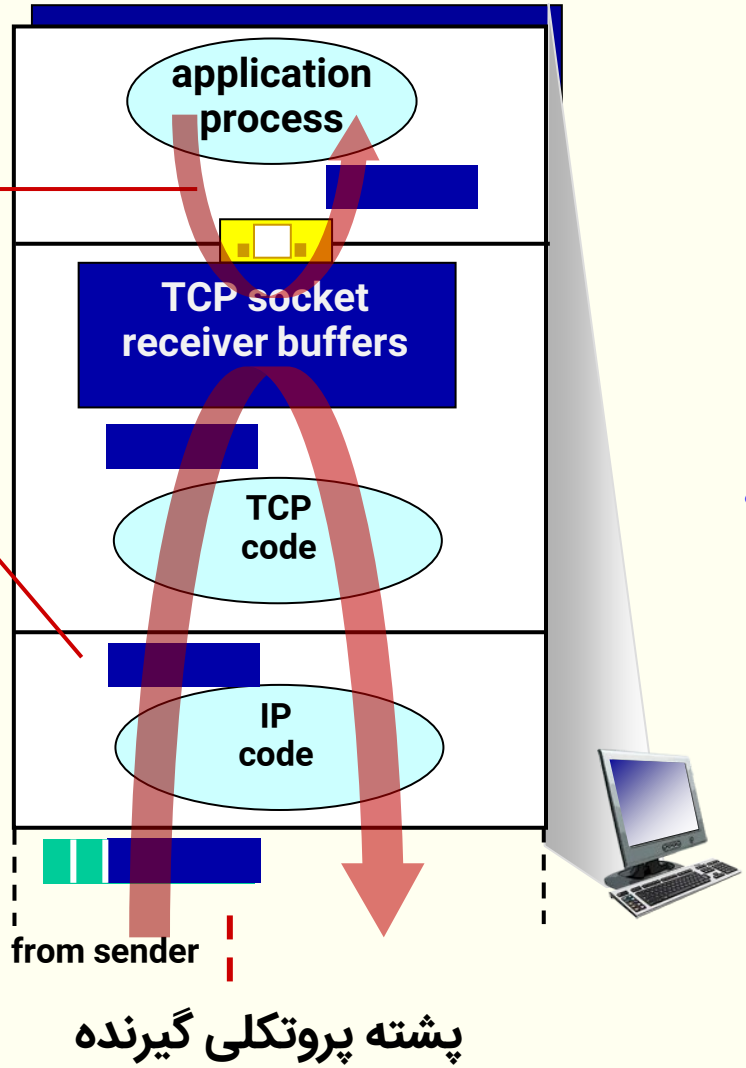
دریافت ۳ تاییده تکراری نشان دهنده این است که گیرنده ۳ سگمنت بعداز سگمنتی که هنوز تایید نشده را دریافت کرده است. این بدان معنی است که سگمنت تایید نشده از دست‌رفته است و باید مجدداً ارسال شود.

پروتکل TCP

TCP سمت گیرنده: تولید و ارسال تاییده (RFC 5681)

رخداد در گیرنده	اقدام گیرنده TCP
دریافت یک سگمنت با شماره ترتیب مورد انتظار که تاییده تمام داده‌های قبل از این سگمنت ارسال شده است.	ارسال با تأخیر تاییده درصورتی که تا ۵۰۰ میلی‌ثانیه تاییده به صورت piggybacking ارسال نشد، تاییده ارسال می‌شود.
دریافت یک سگمنت با شماره ترتیب مورد انتظار که تاییده سگمنت قبلی هنوز ارسال نشده است.	ارسال فوری تاییده ارسال یک تاییده تجمعی که هر دو سگمنت را تایید می‌کند.
دریافت یک سگمنت خارج از ترتیب بعد از شماره ترتیب مور انتظار که گپ بین داده‌ها تشخیص داده شده است.	ارسال فوری تاییده شماره تاییده شماره ترتیب بایت بعدی مورد انتظار است.
دریافت یک سگمنت که کل یا بخشی از گپ موجود بین داده‌های دریافتی در بافر دریافت را پر می‌کند.	ارسال فوری تاییده شماره تاییده شماره ترتیب اولین بایت بعدی گپ است که مورد انتظار برای دریافت است.

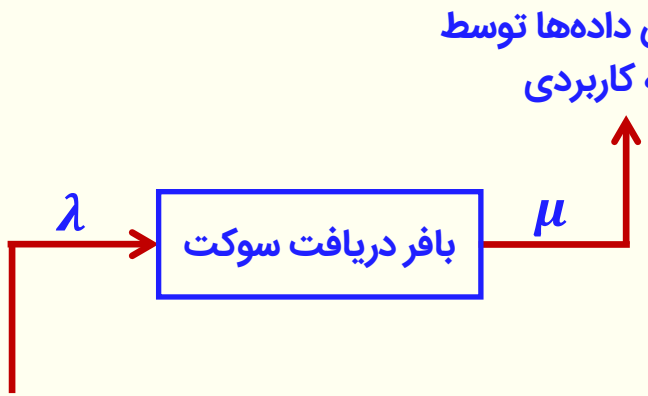
پروتکل TCP: کنترل جریان



برنامه کاربردی داده‌ها را از بافر سوکت TCP بر می‌دارد.

لایه شبکه با دریافت بسته، سگمنت را به TCP می‌دهد و TCP هم داده‌های حمل شده توسط سگمنت را در بافر سوکت قرار می‌دهد.

پشته پروتکلی گیرنده



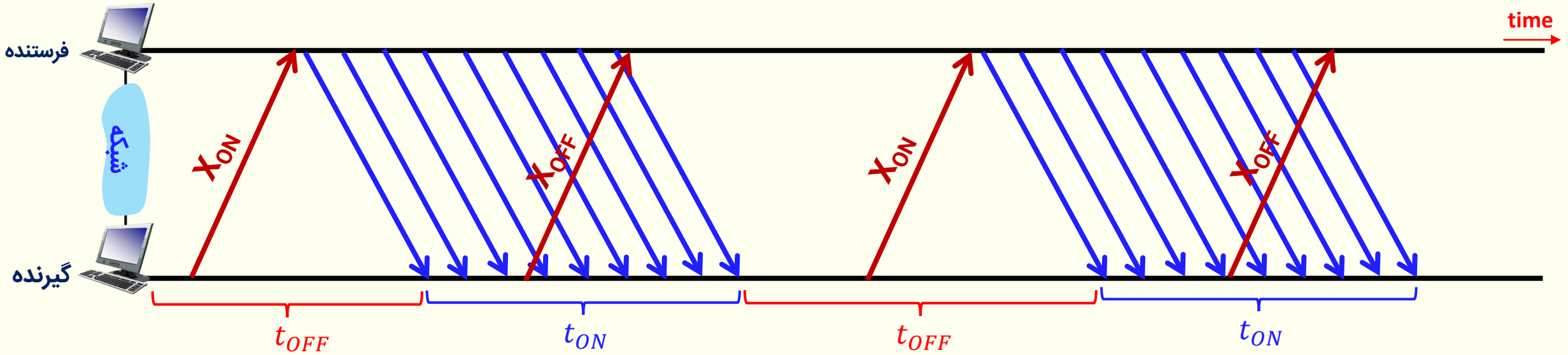
سوال: چه اتفاقی می‌افتد اگر نرخ پر شدن بافر (نرخ ارسال فرستنده) از نرخ خالی شدن بافر (نرخ پردازش داده‌ها توسط برنامه‌کاربردی) بیشتر باشد؟

$$\lambda \overset{?}{>} \mu$$

پروتکل TCP: کنترل جریان

کنترل جریان: کنترل نرخ ارسال فرستنده توسط گیرنده

- عموماً برای کنترل نرخ ارسال از پروتکل‌های پنجره لغزان (sliding window) استفاده می‌کنند:
  - استفاده از اجازه ارسال (X_{ON}) و عدم اجازه ارسال (X_{OFF})
  - کنترل اندازه پنجره ارسال

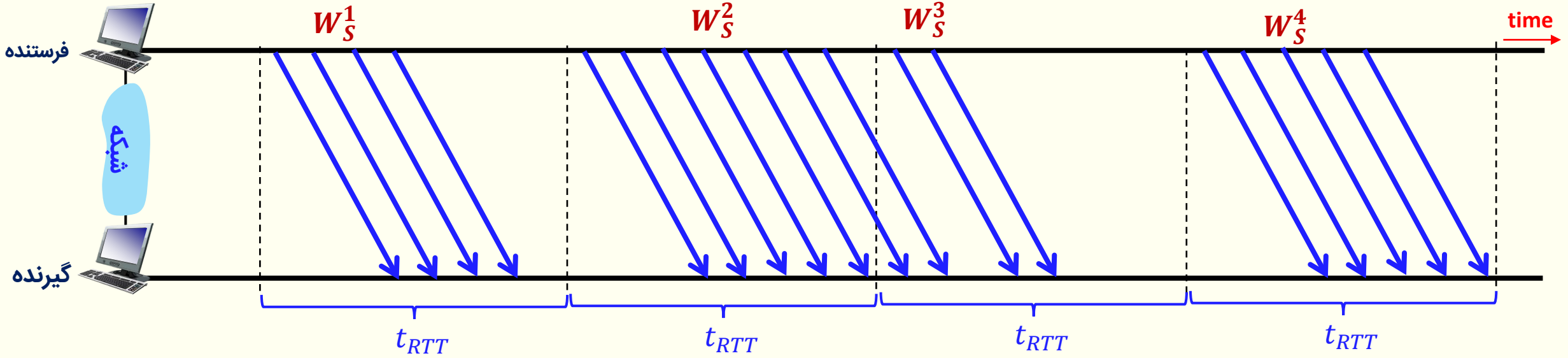


$$\lambda = R_{eff} = \frac{\overline{t_{ON}}}{\overline{t_{ON}} + \overline{t_{OFF}}} \times R$$

پروتکل TCP: کنترل جریان

کنترل جریان: کنترل نرخ ارسال فرستنده توسط گیرنده

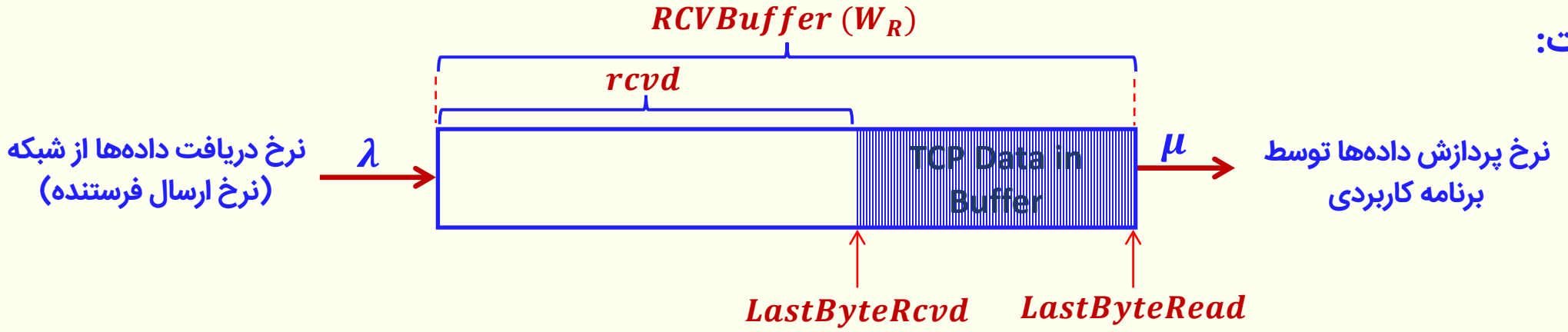
- عموماً برای کنترل نرخ ارسال از پروتکل‌های پنجره لغزان (sliding window) استفاده می‌کنند:
  - استفاده از اجازه ارسال (X_{ON}) و عدم اجازه ارسال (X_{OFF})
  - کنترل اندازه پنجره ارسال

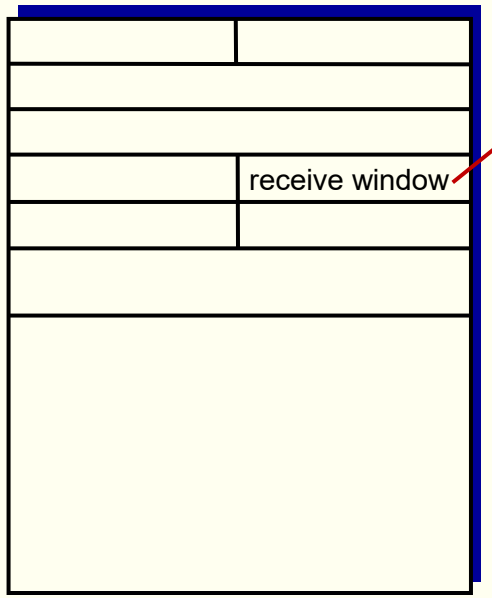


$$\lambda = R_{eff} = \frac{\overline{W_S}}{t_{RTT}}$$

پروتکل TCP: کنترل جریان

بافر دریافت:





کنترل جریان:
تعداد بایت‌هایی که گیرنده
می‌تواند دریافت کند.

$RcvdBuffer\ (W_R)$: بافر دریافت (پنجره دریافت)

$LastByteRead$: آخرین بایت خوانده شده توسط برنامه کاربردی

$LastByteRcvd$: آخرین بایت دریافتی

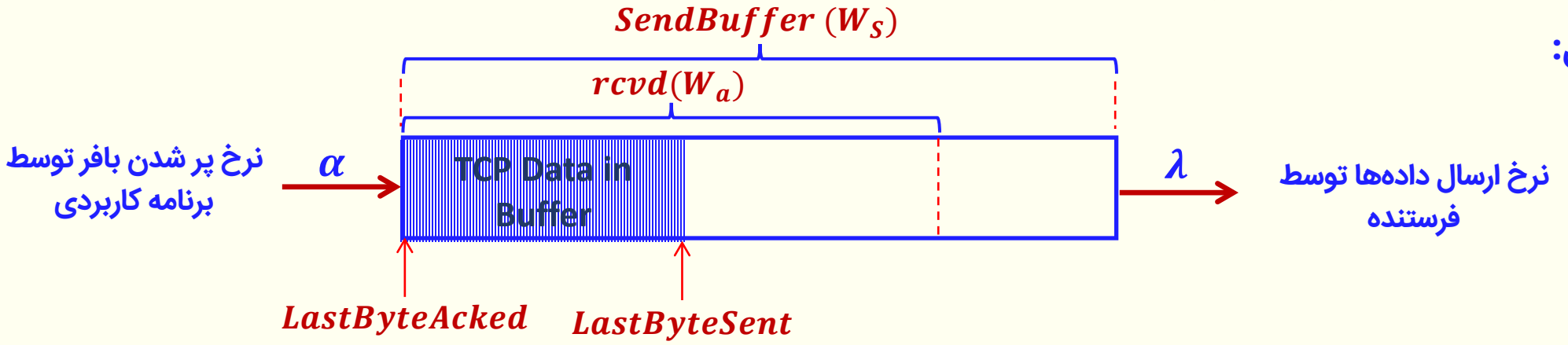
$rwnd\ (W_a)$: بافر دریافت (پنجره دریافت یا پنجره اعلان)

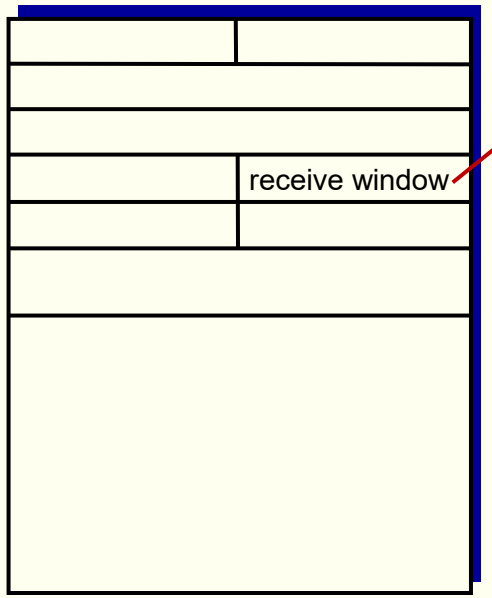
$$rwnd = RcvBuffer - [LastByteRcvd - LastByteRead]$$

مقیاس پنجره (Window Scaling): ۱، ۲، ۴، ۸ و ...

پروتکل TCP: کنترل جریان

بافر ارسال:





کنترل جریان:
تعداد بایت‌هایی که گیرنده می‌تواند دریافت کند.

$SendBuffer\ (W_S)$: بافر ارسال (پنجره ارسال)

$LastByteAcked$: آخرین بایت تایید شده

$LastByteSent$: آخرین بایت ارسالی

$rwnd\ (W_a)$: بافر دریافت (پنجره دریافت یا پنجره اعلان)

$$LastByteSent - LastByteAcked \leq W_S$$
$$LastByteSent - LastByteAcked \leq W_a$$

$$\Rightarrow LastByteSent - LastByteAcked \leq \min(W_S, W_a)$$

پروتکل TCP: مدیریت اتصال

مدیریت اتصال:

- **فاز برقراری اتصال (connection management):**
  - ایجاد دو اتصال همزمان قبل از ارسال داده‌ها
    - یک اتصال از سرویس‌گیرنده به سرویس‌دهنده (برای ارسال درخواست)
    - یک اتصال از سرویس‌دهنده به سرویس‌گیرنده (برای ارسال پاسخ)
  - اطمینان از کنترل خطا بین اتصال‌های متوالی:
    - تغییر شماره ترتیب اولیه (Initial Sequence Number) از یک اتصال به اتصال بعدی
- **فاز بستن اتصال (connection close):**
  - بستن اتصال‌ها بعداز ارسال داده‌ها
    - بر خلاف باز شدن اتصال‌ها که باید همزمان باشند، بسته شدن آن‌ها می‌تواند همزمان نباشد.
    - اتصال سرویس‌گیرنده به سرویس‌دهنده زودتر بسته می‌شود.
    - پس از بسته‌شدن اتصال سرویس‌گیرنده به سرویس‌دهنده، اتصال سرویس‌دهنده به سرویس‌گیرنده می‌تواند همچنان برای ارسال پاسخ باز بماند.
  - اطمینان از دریافت همه سگمنت‌های ارسالی سرویس‌دهنده توسط سرویس‌گیرنده:
    - سرویس‌گیرنده پس از دریافت درخواست بستن اتصال از سرویس‌دهنده و ارسال تاییده آن، به اندازه ۲ برابر حداکثر طول عمر سگمنت (Maximum Segment Lifetime) در شبکه صبر می‌کند و سپس اتصال را می‌بندد.

لایه انتقال

پروتکل TCP: برقراری اتصال (دست‌دهی سه‌گانه - three-way handshaking)

Client state

```
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
```

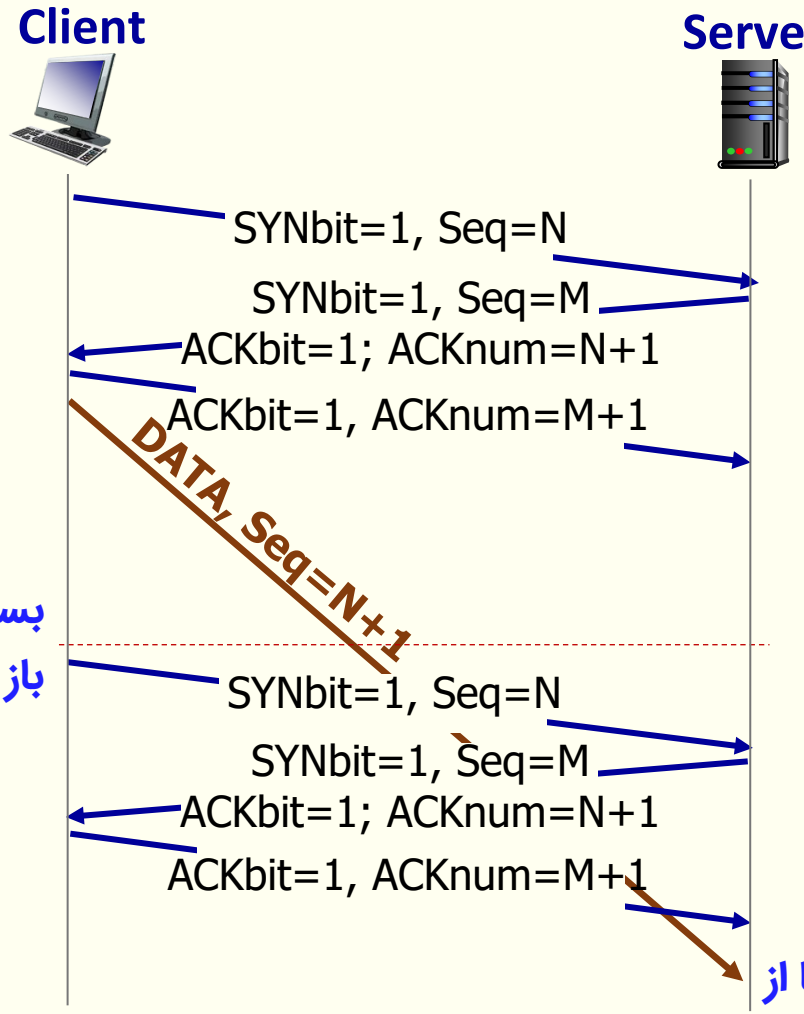
Server state

```
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
connectionSocket, addr = serverSocket.accept()
```

لایه انتقال

پروتکل TCP: برقراری اتصال (تغییر شماره ترتیب اولیه)

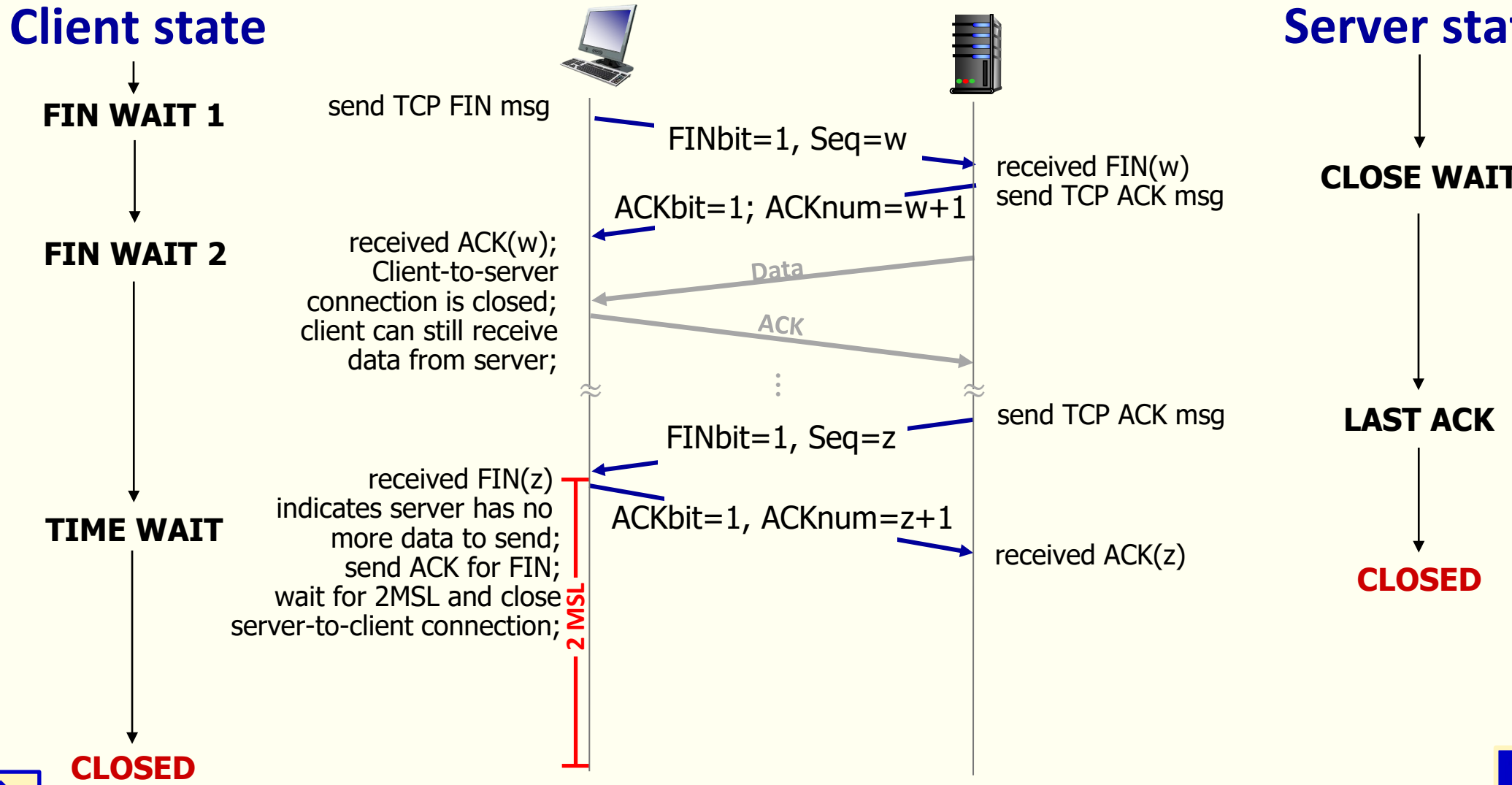
مثال نقض: شروع از یک شماره ترتیب ثابت (N)



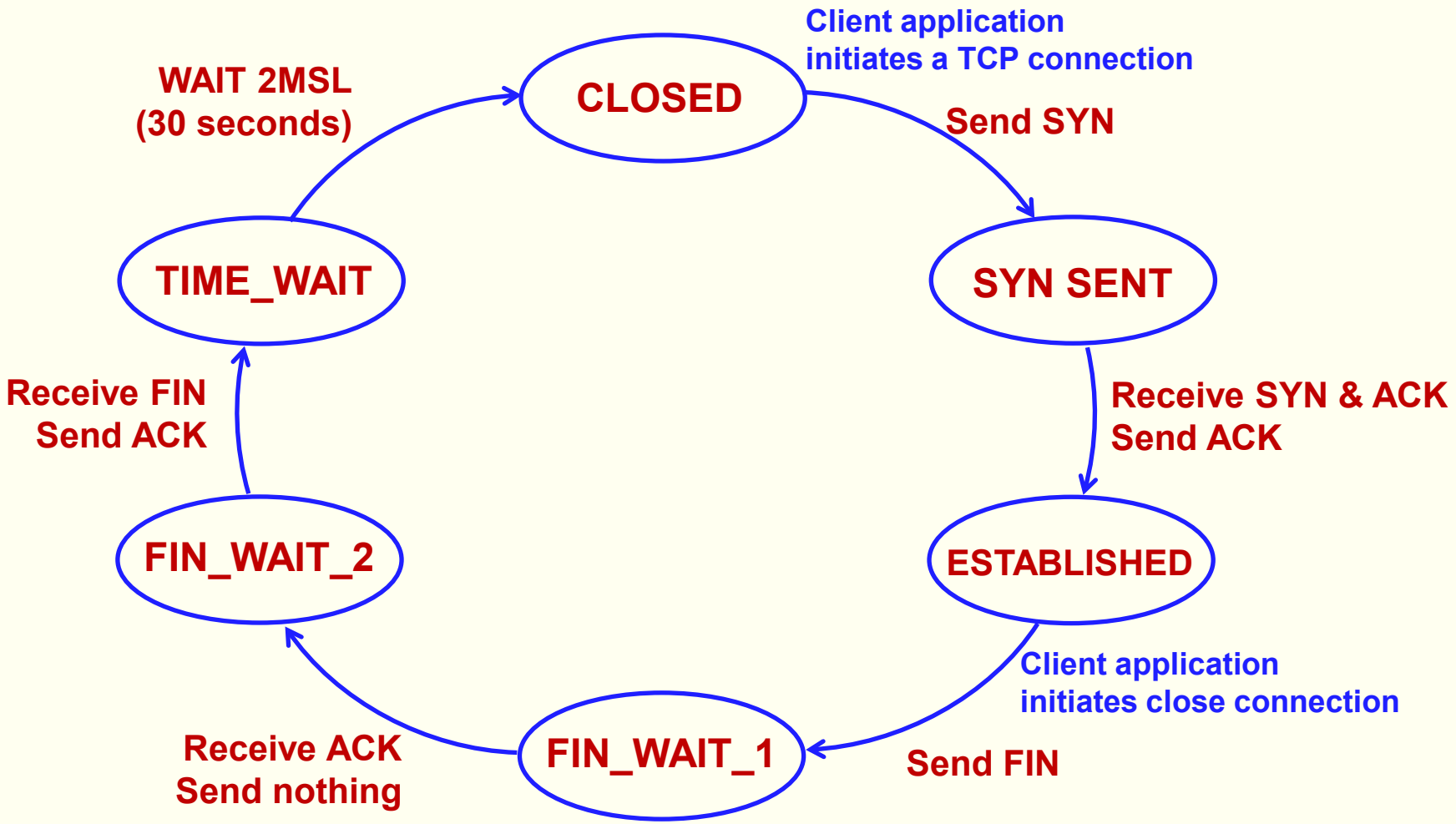
بسته شدن غیر نرمال اتصال و باز شدن مجدد آن

دریافت اطلاعات با شماره ترتیب مورد انتظار، اما از داده‌های اتصال قبلی (دریافت داده‌های اشتباه)

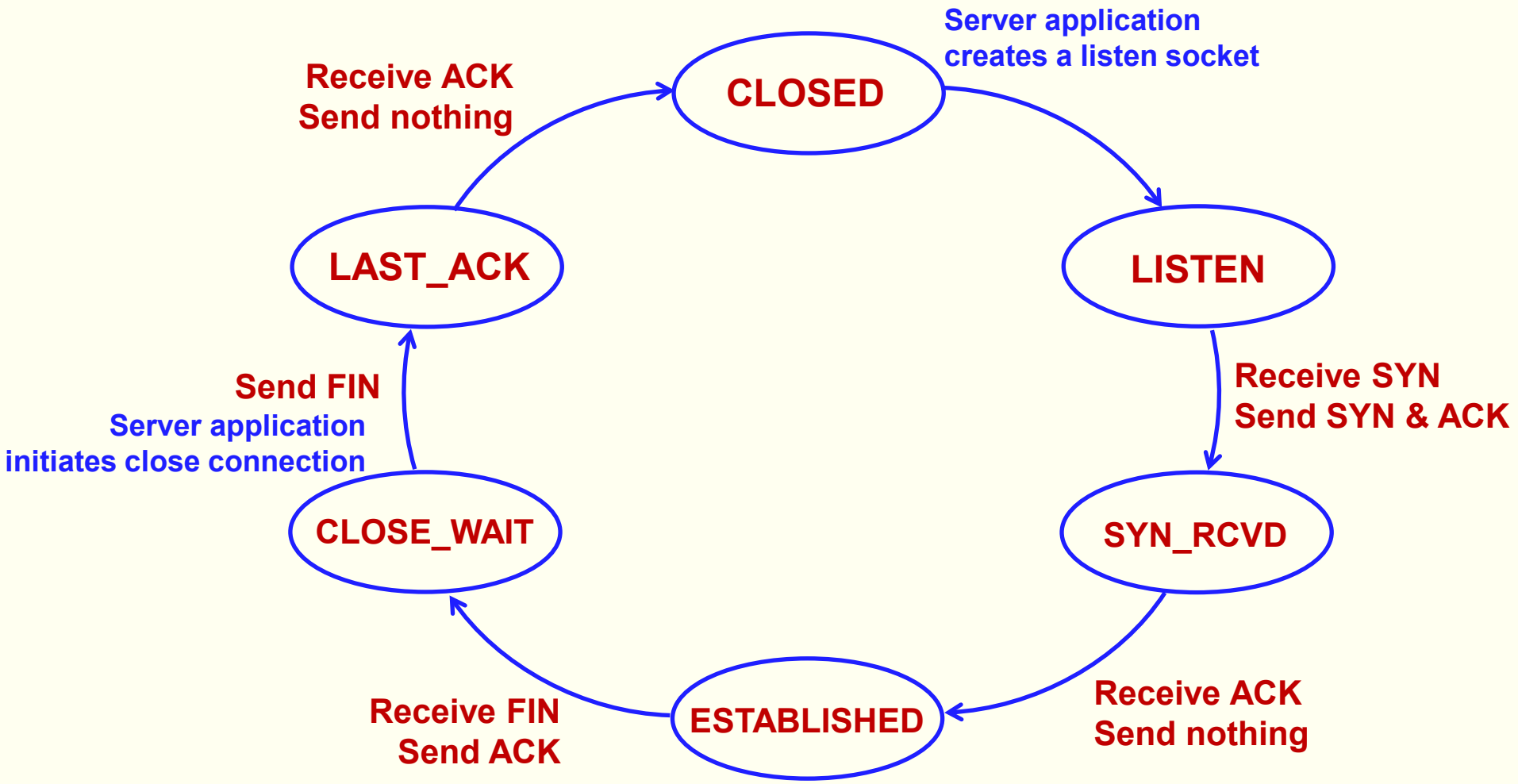
پروتکل TCP: خاتمه اتصال (closing a TCP connection)



پروتکل TCP: دیاگرام حالت (سمت سرویس‌گیرنده – client)



پروتکل TCP: دیاگرام حالت (سمت سرویس‌دهنده – server)



کنترل ازدحام

ازدحام: زمانی‌که حجم ترافیک ورودی به شبکه بیش از ظرفیت ارسال آن باشد، ازدحام رخ داده است.

- در واقع ازدحام بر روی لینک‌ها شبکه رخ می‌دهد، زمانی که حجم ترافیکی که می‌خواهد از یک لینک فیزیکی عبور کند از ظرفیت ارسال آن لینک بیشتر می‌شود.

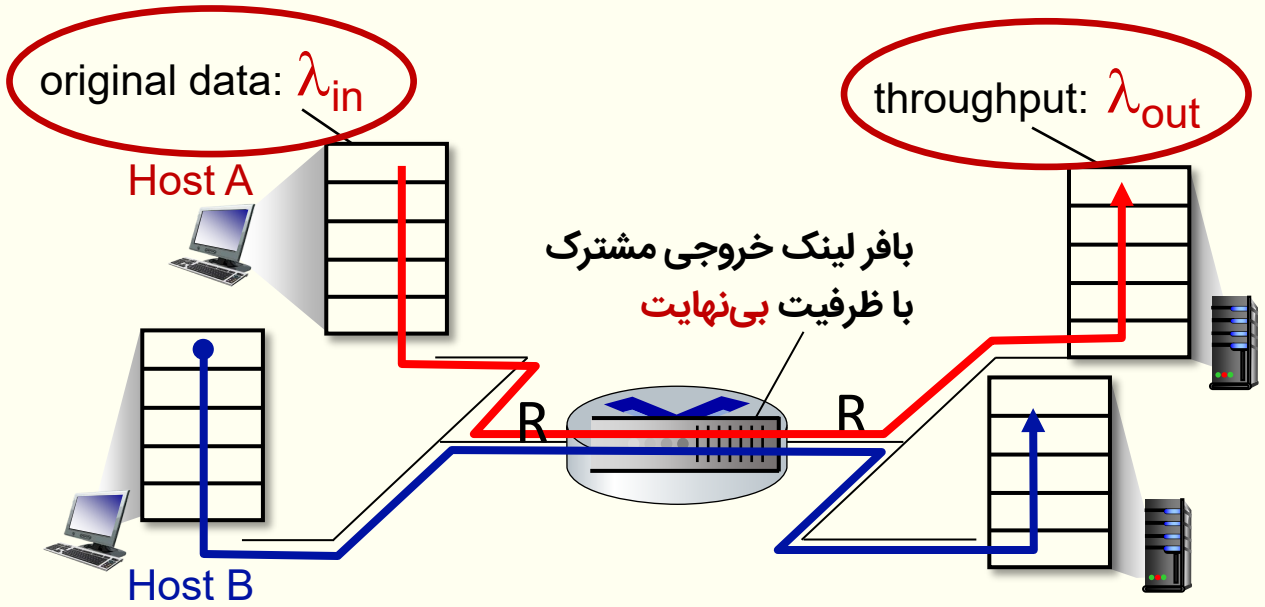
پیآمدهای ازدحام:

- افزایش تأخیر
- ازدست دادن بسته‌ها (packet loss)

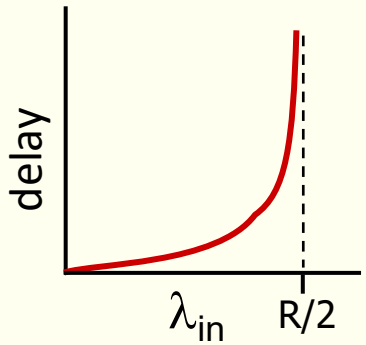
کنترل ازدحام

سناریور ۱:

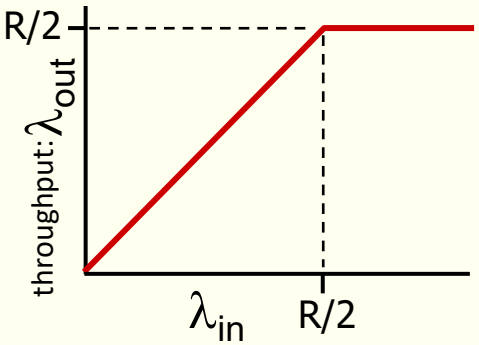
- یک مسیریاب، بافر لینک خروجی با ظرفیت نامحدود
- ظرفیت لینک ورودی و لینک خروجی: R
- دو جریان ترافیکی
- بدون ارسال مجدد



سوال: چه اتفاقی رخ می‌دهد، اگر نرخ ارسال ورودی λ_{in} به $R/2$ نزدیک شود؟



با افزایش نرخ ارسال ورودی λ_{in}، تأخیر نیز افزایش می‌یابد.

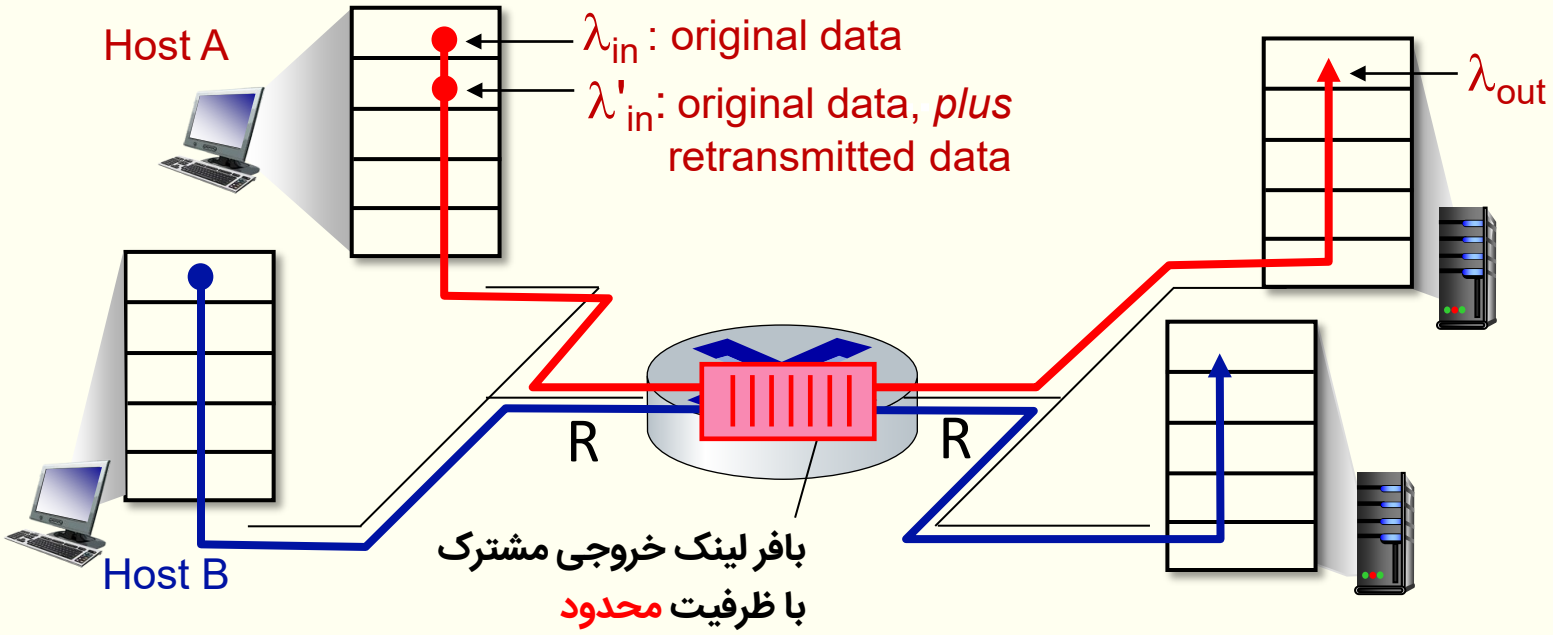


گذردهی حداکثر به ازای هر اتصال: $R/2$

کنترل ازدحام

سناریور ۲:

- یک مسیریاب، بافر لینک خروجی با ظرفیت محدود
- ظرفیت لینک ورودی و لینک خروجی: R
- دو جریان ترافیکی
- با ارسال مجدد داده‌های اردست رفته

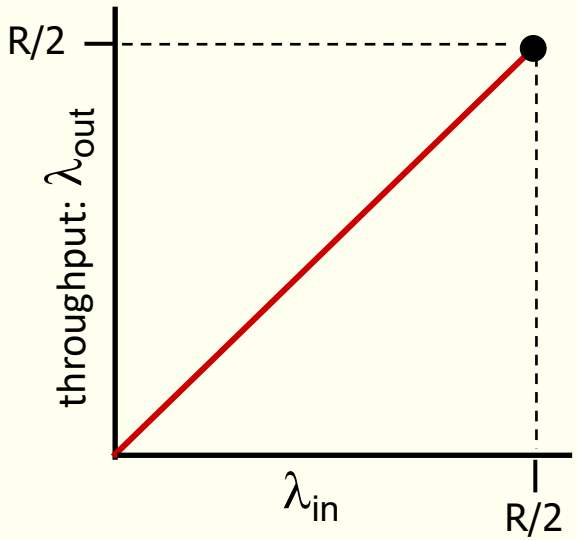


- فرستنده داده‌های از دست داده شده را مجددا ارسال می‌کند:
  - نرخ ارسال ورودی از لایه کاربرد فرستنده با نرخ ارسال خروجی به لایه کاربرد مقصد برابر است ($\lambda_{in} = \lambda_{out}$).
  - نرخ ارسال ورودی به شبکه از لایه انتقال برابر است با نرخ ارسال لایه ماربرد به علاوه نرخ ارسال‌های مجدد ($\lambda'_{in} \geq \lambda_{in}$).

کنترل ازدحام

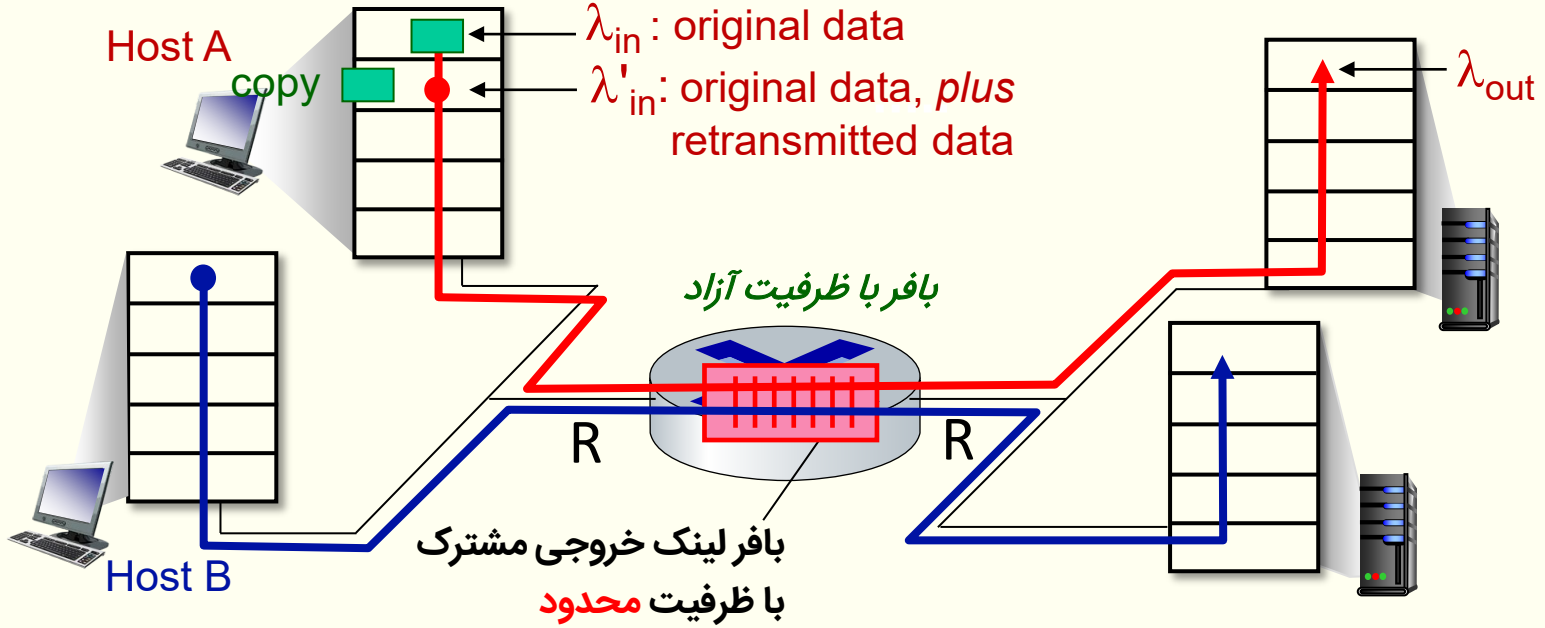
سناریور ۲:

- یک مسیریاب، بافر لینک خروجی با ظرفیت محدود
- ظرفیت لینک ورودی و لینک خروجی: R
- دو جریان ترافیکی
- با ارسال مجدد داده‌های اردست رفته



اقدام ایده‌آل: با دانش کامل

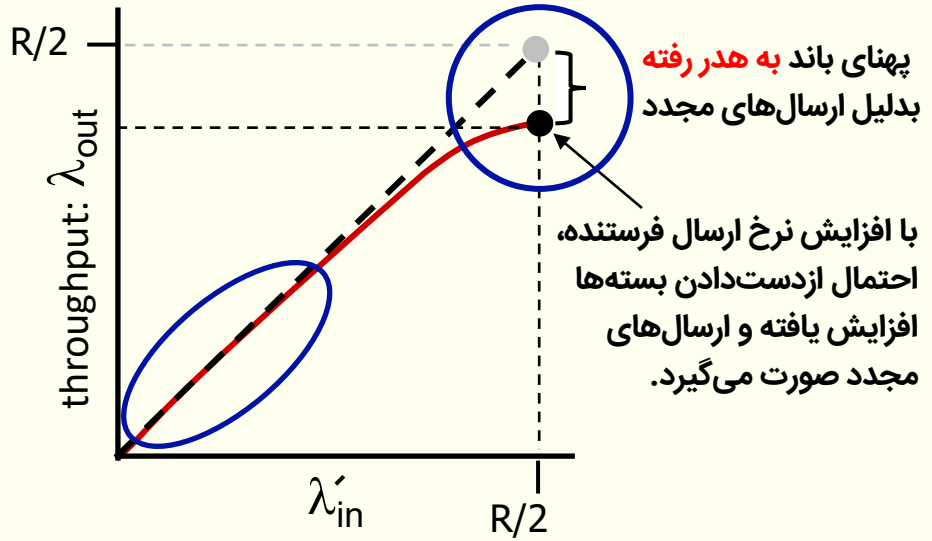
- فرستنده فقط زمانی اقدام به ارسال کند که بافر لینک خروجی مسیریاب فضای خالی داشته باشد.



کنترل ازدحام

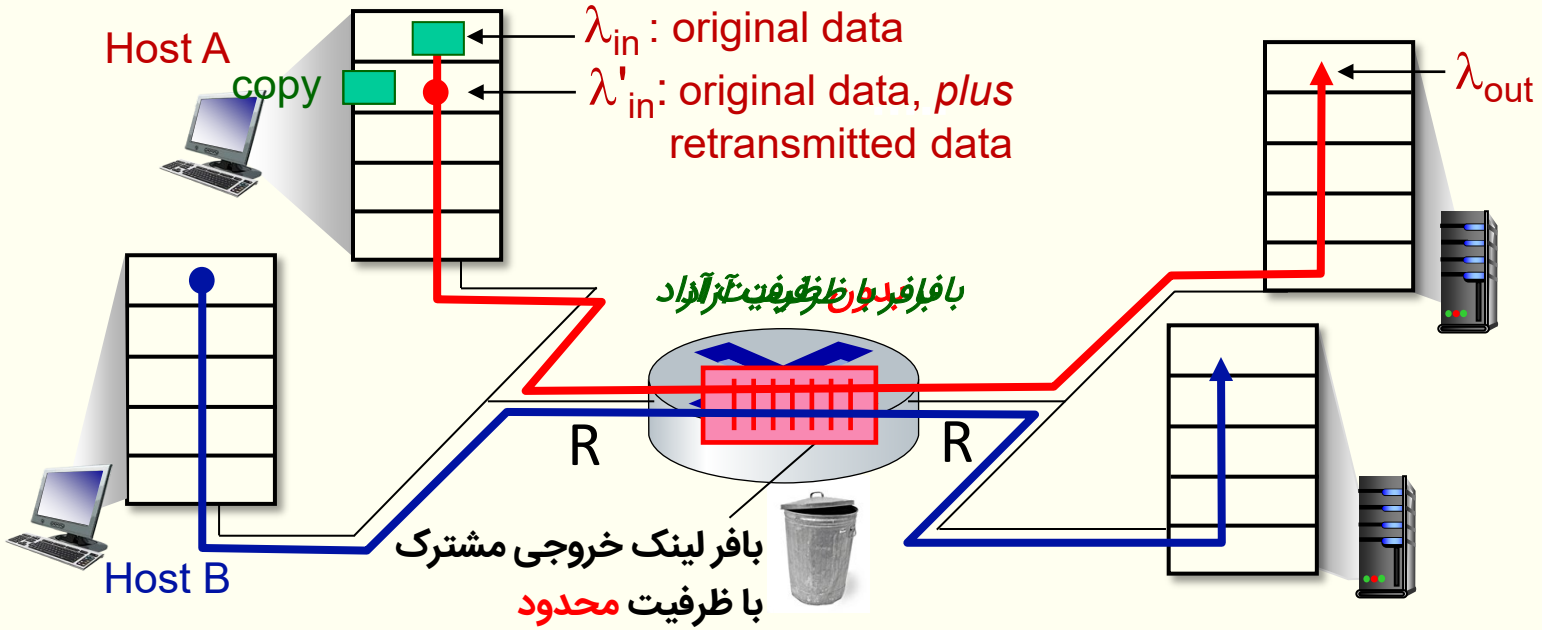
سناریور ۲:

- یک مسیریاب، بافر لینک خروجی با ظرفیت محدود
- ظرفیت لینک ورودی و لینک خروجی: R
- دو جریان ترافیکی
- با ارسال مجدد داده‌های اردست رفته



اقدام ایده‌آل: با دانش نه چندان کامل

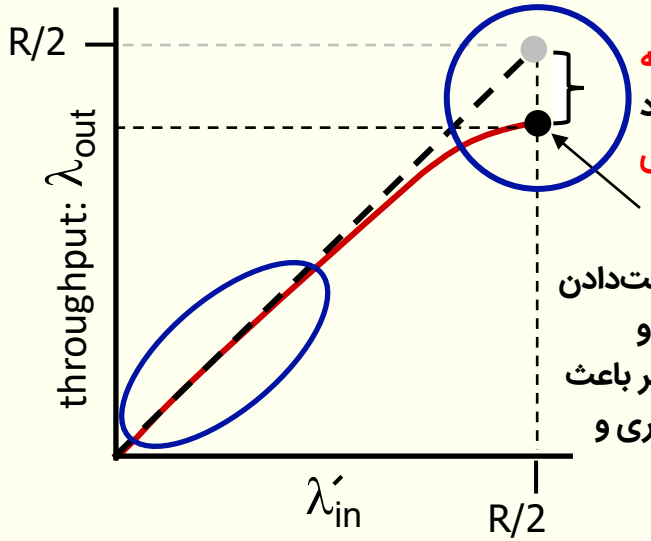
- بدلیل پر بودن بافر لینک خروجی، ممکن است بسته‌هایی از دست داده شوند.
- فرستنده با تشخیص از دست دادن بسته‌ها، فقط بسته‌های از دست داده شده را مجدداً ارسال می‌کند.



کنترل ازدحام

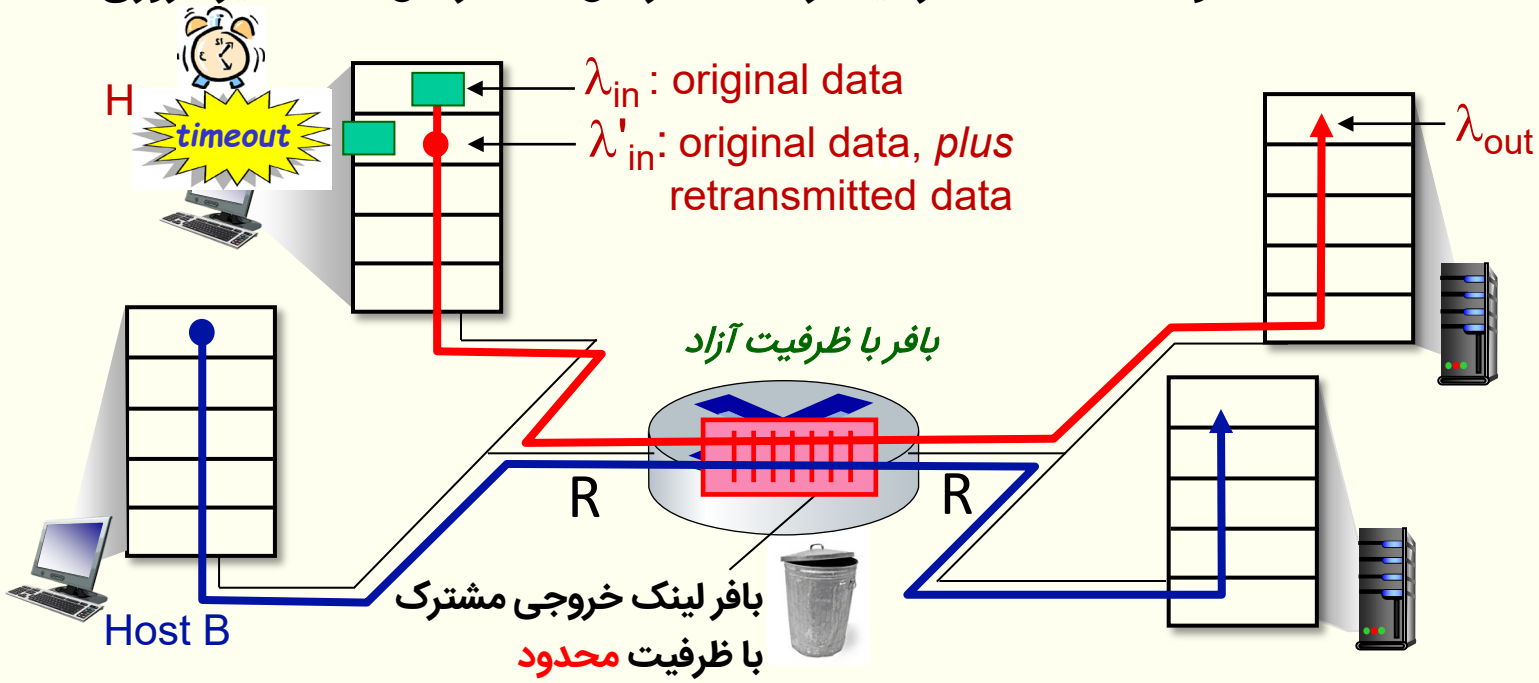
سناریور ۲:

- یک مسیریاب، بافر لینک خروجی با ظرفیت محدود
- ظرفیت لینک ورودی و لینک خروجی: R
- دو جریان ترافیکی
- با ارسال مجدد داده‌های اردست رفته



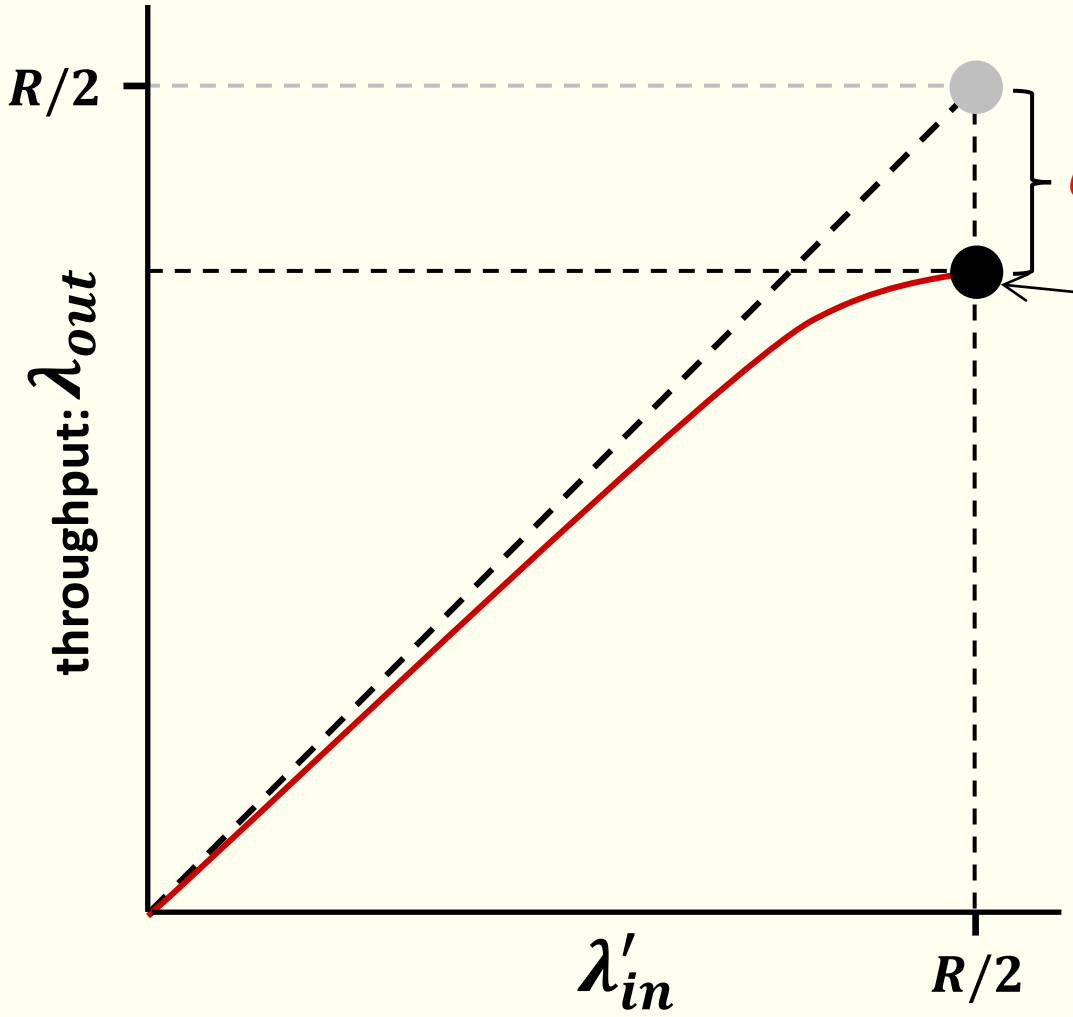
اقدام ایده‌آل: ارسال مجددهای غیرضروری

- بدلیل پر بودن بافر لینک خروجی، ممکن است بسته‌هایی از دست داده شوند.
- بدلیل افزایش تأخیر صف‌بندی در زمان ازدحام، ممکن است زمانبند فرستنده منقضی شده (timeout) و بسته به مقصد رسیده را مجدداً ارسال کند (ارسال مجدد غیرضروری).



کنترل ازدحام

سناریور ۲: حالت واقع‌بینانه



پهنای باند **به هدر رفته** بدلیل ارسال‌های مجدد **ضروری** و **غیرضروری**

با افزایش نرخ ارسال فرستنده، احتمال ازدست‌دادن بسته‌ها افزایش یافته و همچنین افزایش تأخیر باعث ارسال‌های مجدد ضروری و غیرضروری می‌شود.

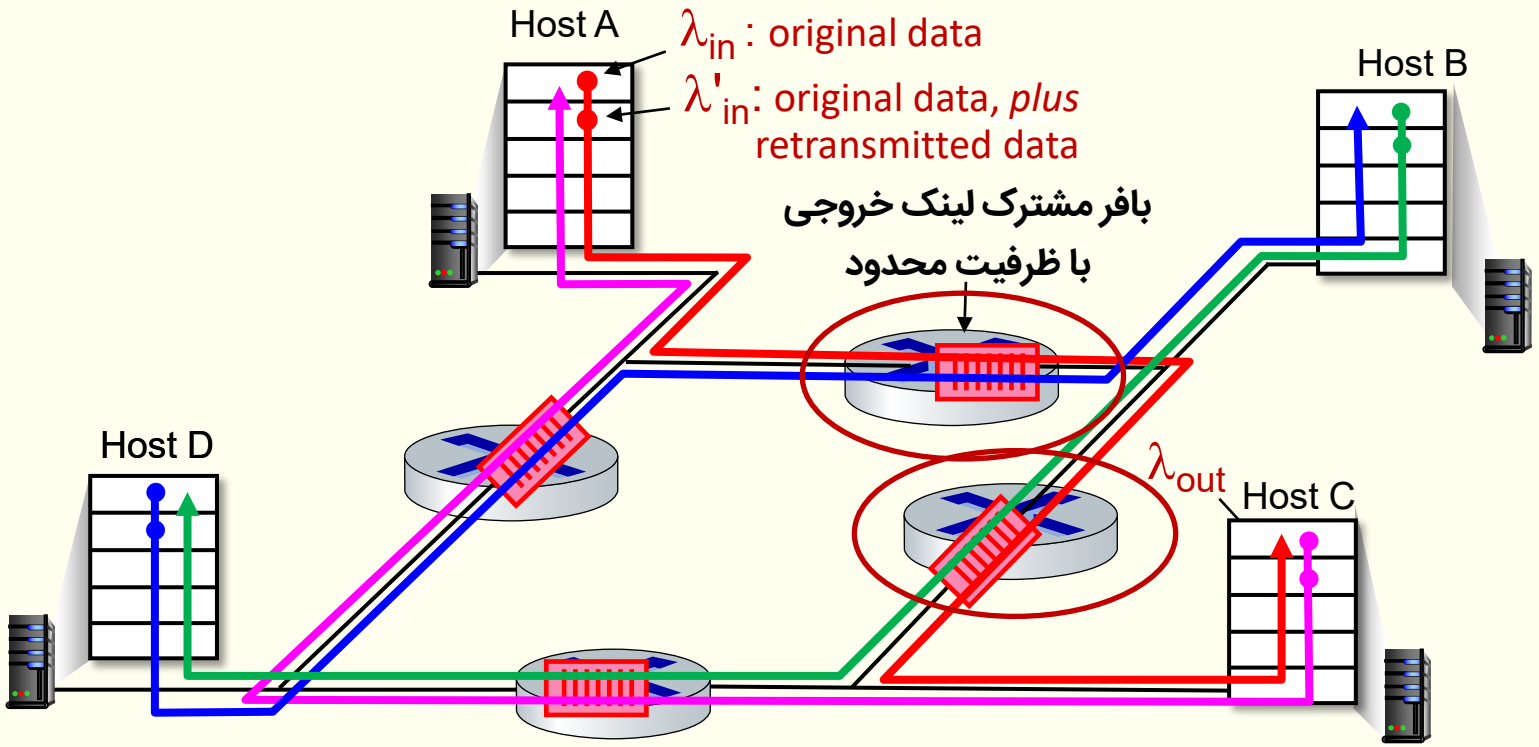
هزینه‌های ازدحام:

- کاهش نرخ‌گذردهی بدلیل افزایش ارسال‌های مجدد (ضروری و غیرضروری)
- ایجاد صف طولانی در بافر لینک خروجی مسیریاب و افزایش تأخیر بدلیل پرشدن بافر لینک خروجی

کنترل ازدحام

سناریو ۳:

- چهار فرستنده
- مسیرهای چندگانه
- منقضی‌شدن زمانبند و ارسال‌های مجدد



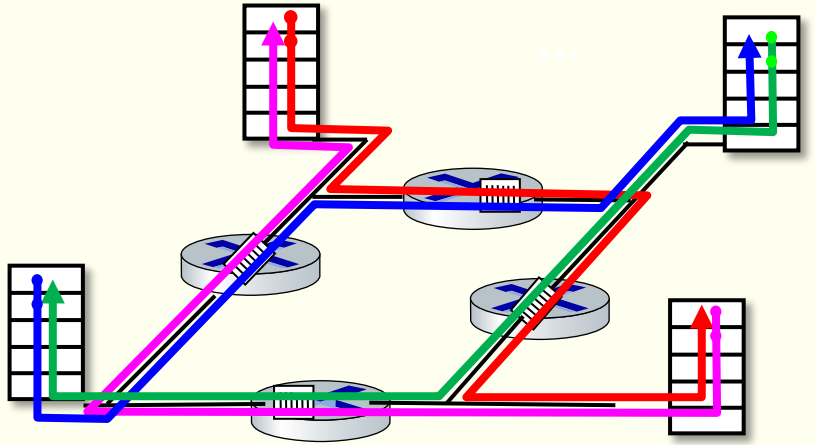
سوال: چه اتفاقی می‌افتد اگر λ_{in} و λ'_{in} افزایش یابد.

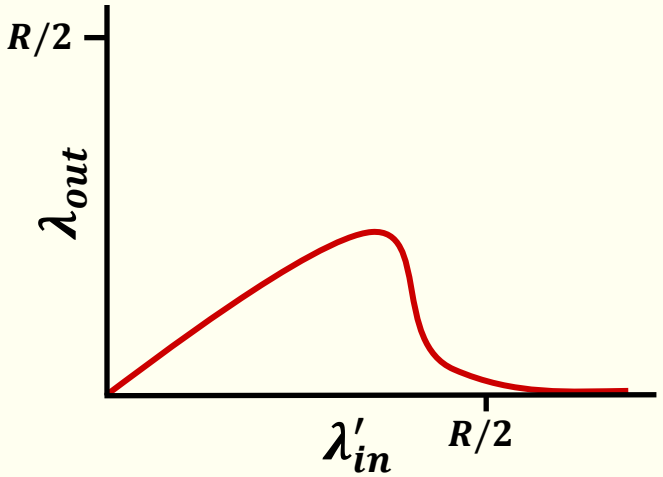
جواب: با افزایش λ'_{in}، تمام بسته‌های از جریان دیگر (رنگ آبی) در انتهای صف حذف می‌شوند و گذردهی جریان رنگ آبی به سمت صفر میل می‌کند.

کنترل ازدحام

سناریور ۳:

- چهار فرستنده
- مسیرهای چندگانه
- منقضی‌شدن زمانبند و ارسال‌های مجدد



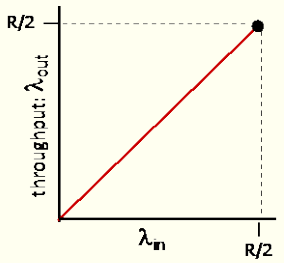


هزینه دیگر ازدحام:

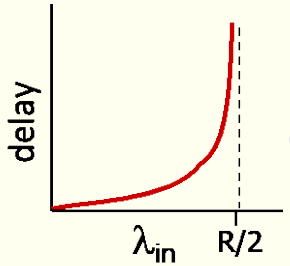
زمانی که یک بسته حذف می‌شود، ظرفیت ارسال و همچنین ظرفیت بافرهای استفاده شده تا مسیریابی که بسته در آن حذف شده است، به هدر رفته است.

لایه انتقال

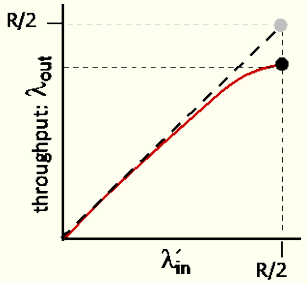
کنترل ازدحام



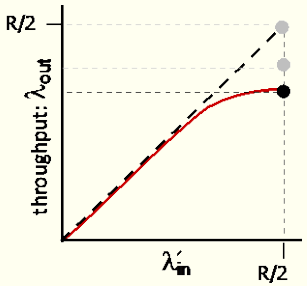
گذردهی نمی‌تواند از ظرفیت ارسال بیشتر شود.



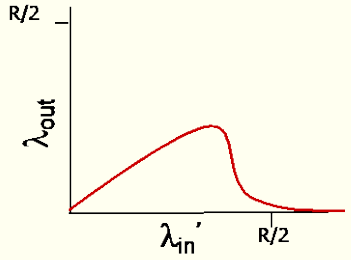
افزایش تأخیر با نزدیک شدن نرخ ارسال به ظرفیت ارسال



ازدست‌دادن بسته‌ها و ارسال‌های مجدد باعث کاهش گذردهی مؤثر می‌شود.



تکرارهای غیرضروری نیز باعث کاهش گذردهی مؤثر می‌شود.

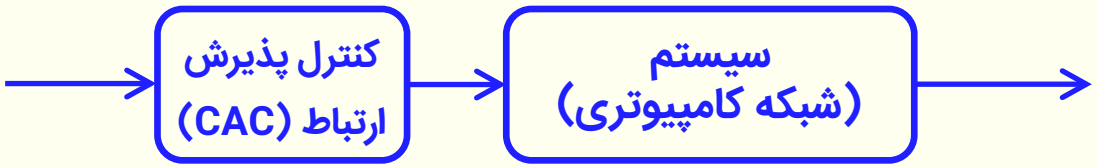


هدر رفت ظرفیت ارسال و بافر لینک‌های مسیر تا مسیریابی که بسته را حذف کرده است.

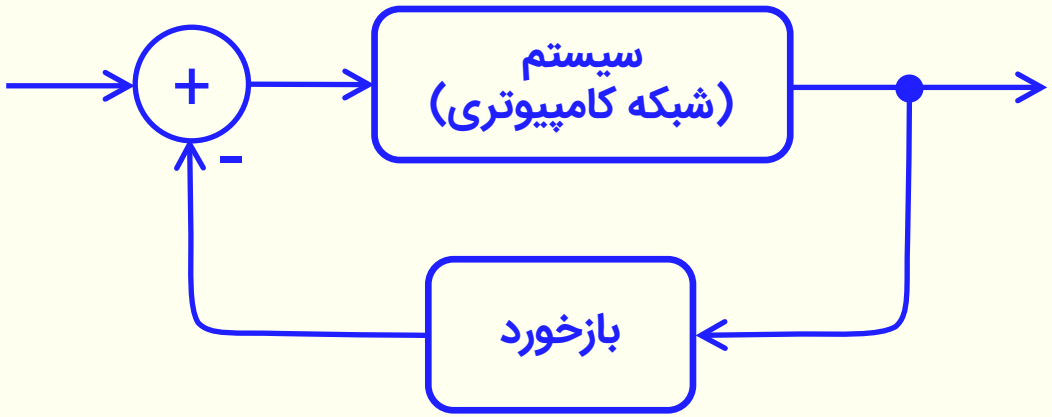
کنترل ازدحام: روش‌های کنترل ازدحام

روش‌های کنترل ازدحام:

- **روش‌های حلقه باز (پیشگیرانه) – open loop (preventive)**
  - فقط جریان‌های ترافیکی پذیرفته شده وارد شبکه می‌شوند.
  - جریان‌های ترافیکی که در داخل شبکه ازدحام ایجاد می‌کنند، پذیرش نمی‌شوند.
  - کیفیت سرویس جریان‌های پذیرش شده توسط شبکه تضمین می‌شود.
  - ظرفیت ارسال شبکه به صورت بهینه استفاده نمی‌شود (بدلیل احتیاط در پذیرش جریان‌های ترافیکی به منظور تضمین کیفیت سرویس



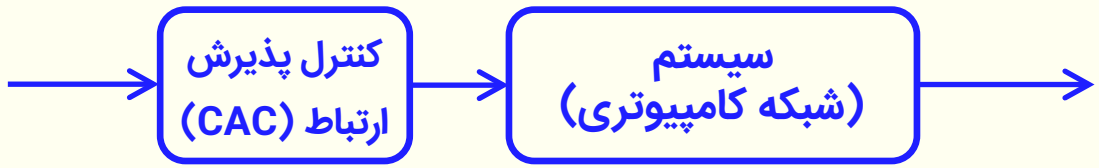
- **روش‌های حلقه بسته (واکنشی) – closed loop (reactive)**
  - جریان‌های ترافیکی آزادانه وارد شبکه می‌شوند (نیازی به پذیرش ندارند).
  - بعد از وقوع ازدحام، با کاهش ترافیک ورودی ازدحام از بین می‌رود.
  - منبع تولید جریان ترافیکی از شبکه بازخورد می‌گیرد و در صورت تشخیص ازدحام نرخ ترافیک خود را کاهش می‌دهد.
  - ظرفیت ارسال شبکه به صورت بهینه استفاده می‌شود (بدلیل اینکه بدون محدودیت ظرفیت ارسال شبکه تا مرز ازدحام پر می‌شود).
  - تضمین کیفیت سرویس وجود ندارد.



کنترل ازدحام: روش حلقه باز (پیشگیرانه) – open loop (preventive)

روش حلقه باز (پیشگیرانه) – open loop (preventive)

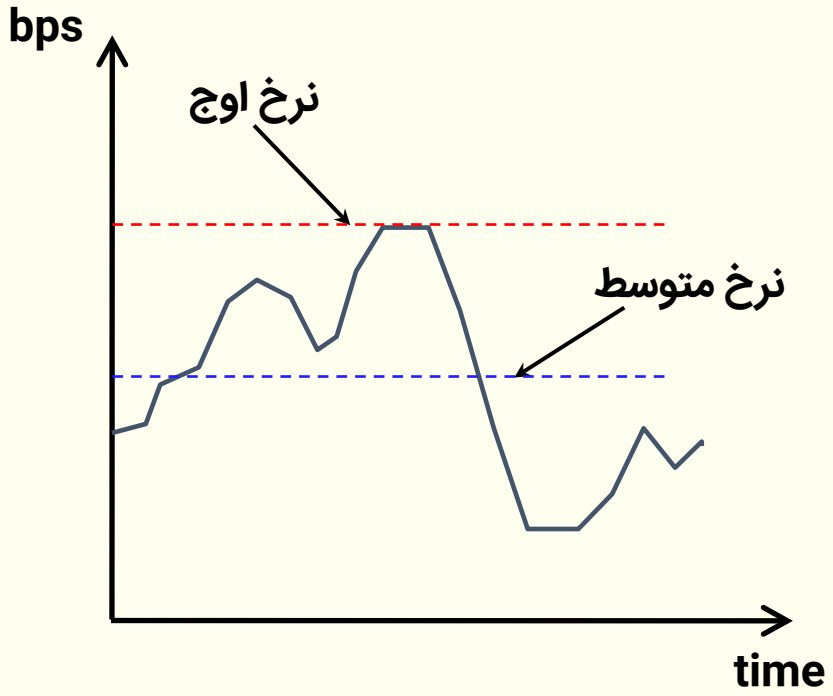
- کارآیی شبکه (کیفیت سرویس) برای همه جریان‌های ترافیکی پذیرفته شده در شبکه تضمین می‌شود.
  - گذردهی (thrughput)
  - تأخیر (delay)
  - احتمال ازدست‌دادن داده‌ها (data loss)
- سازوکارهای کلیدی روش‌های کنترل ازدحام حلقه باز
  - کنترل پذیرش ارتباط (call admission control)
  - پلیس ترافیکی (traffic policing)
  - شکل‌دهی ترافیک (traffic shaping)



کنترل ازدحام: روش حلقه باز (پیشگیرانه) – open loop (preventive)

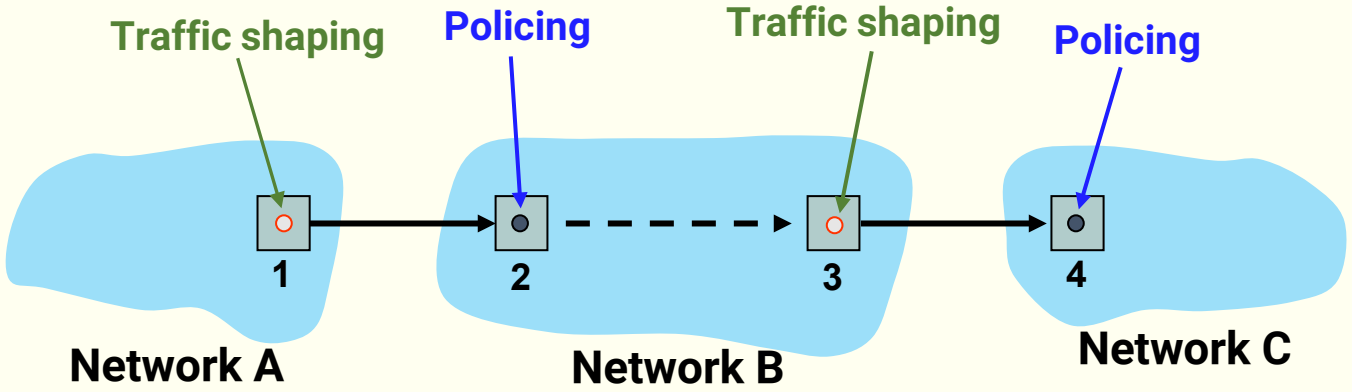
کنترل پذیرش ارتباط (call admission control)

- دریافت مشخصات ترافیکی از کاربر:
  - نرخ ترافیکی (گذردهی) حداقل، متوسط و اوج مورد تقاضا
  - حداکثر اندازه اوج
  - تأخیر و احتمال از دست دادن قابل قبول
- پیداکردن مسیری در شبکه که نیازمندی‌های ترافیکی را تأمین کند و ازدحام در شبکه ایجاد نکند (امکان مذاکره بین کاربر و شبکه برای تغییر نیازمندی‌ها).
- توافق ترافیکی بین کاربر و شبکه در صورت پذیرش جریان ترافیکی
  - تضمین کیفیت سرویس توسط شبکه
  - پایش و کنترل جریان ترافیکی برای جلوگیری از تخلفات ترافیکی (traffic policing)
  - رعایت مشخصات ترافیکی توسط کاربر (traffic shaping)

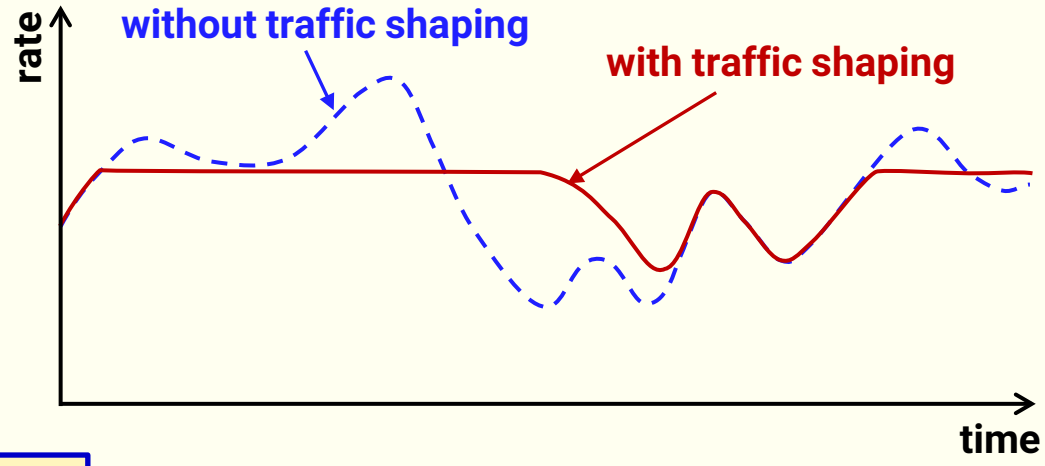


تقاضای نرخ ارسال داده‌ها برای یک جریان ترافیکی با نرخ ارسال متغیر

کنترل ازدحام: روش حلقه باز (پیشگیرانه) – open loop (preventive)



- **پلیس ترافیکی (traffic policing)**
  - پایش و کنترل مداوم جریان ترافیکی به منظور اطمینان از رعایت توافق ترافیکی توسط شبکه
  - زمانی که بسته‌ای از توافق ترافیکی تخطی می‌کند:
    - بسته متخلف حذف می‌شود یا
    - بسته متخلف علامت‌گذاری شده و حذف آن در صورت مواجهه با ازدحام

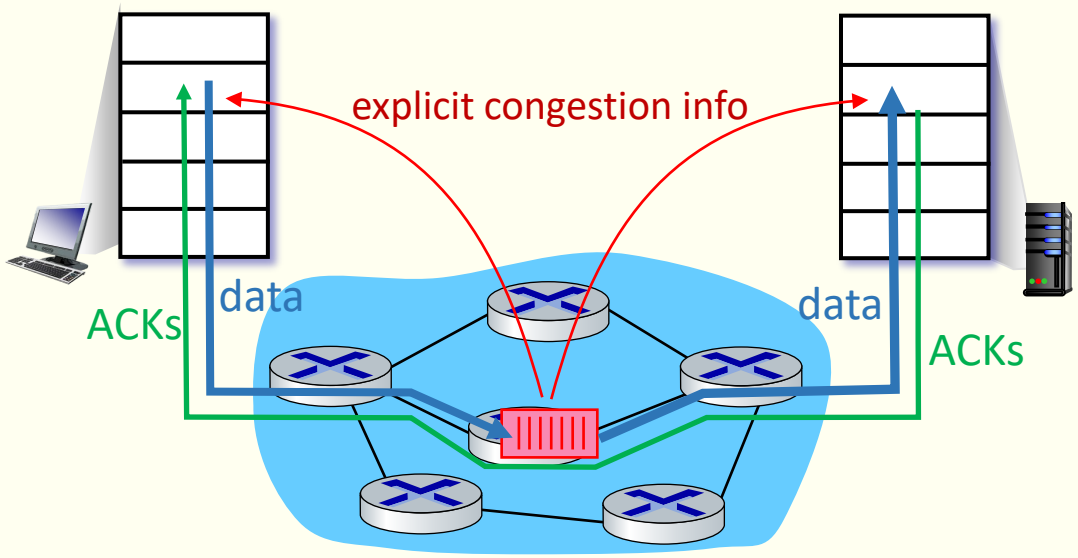


- **شکل‌دهی ترافیکی (traffic shaping)**
  - تغییر شکل ترافیکی به منظور رعایت توافق ترافیکی توسط کاربر
  - بافر کردن بسته‌های متخلف و ارسال آن در زمان مناسب
  - شکل‌دهی ترافیک موجب افزایش تأخیر می‌شود.

کنترل ازدحام: روش حلقه بسته (واکنشی) – closed loop (reactive)

روش حلقه بسته (واکنشی) – closed loop (reactive)

- ورود آزادانه جریان‌های ترافیکی به شبکه
- دریافت بازخورد از شبکه و تنظیم نرخ ارسال توسط گره مبدأ
- استفاده بهینه از ظرفیت ارسال شبکه
- عدم تضمین کیفیت سرویس
- سازوکارهای دریافت بازخورد:
  - به کمک شبکه (بازخورد صریح (Explicit))
    - تشخیص توسط مسیریاب‌ها (از روی وضعیت بافر یا لینک‌ها یا ...)
    - علامت‌گذاری بسته‌های عبوری و اعلام گیرنده به مبدأ
    - اعلام مستقیم مسیریاب به مبدأ
  - بر اساس مشاهدات رفتار شبکه (بازخورد ضمنی (Implicit))
    - از دست دادن بسته‌ها یا افزایش تأخیر



کنترل ازدحام پروتکل TCP: AIMD

رویکرد: مادامی که بسته‌ای از دست نرفته (ازدحام رخ نداده)، فرستنده نرخ ارسال را افزایش می‌دهد،

با وقوع ازدحام (از دست رفتن بسته) فرستنده نرخ ارسال را کاهش می‌دهد:

Additive Increase

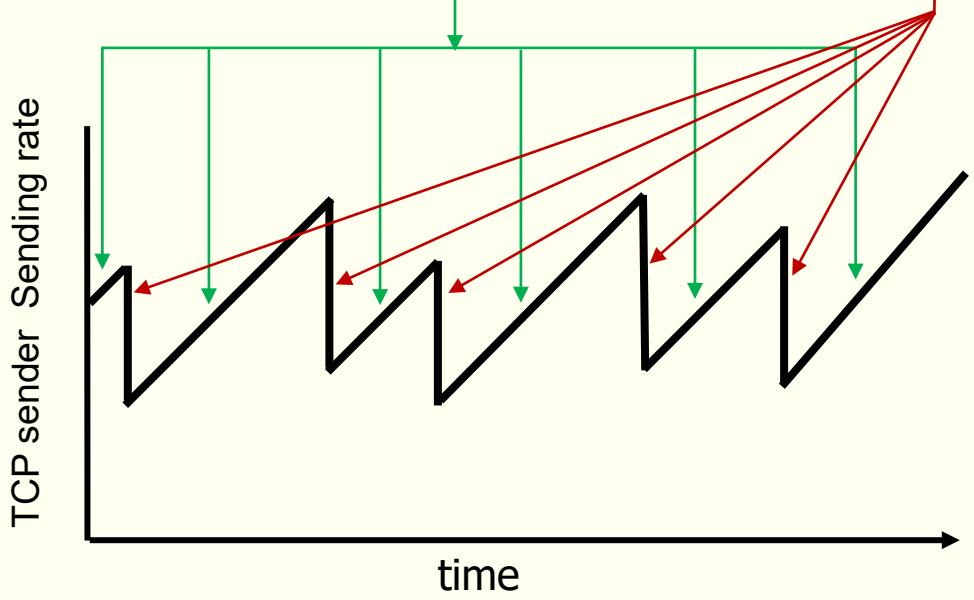
تا قبل از تشخیص ازدست دادن بسته:

- افزایش نرخ ارسال به اندازه 1 MSS در هر RTT

Multiplicative Decrease

در هر رخداد از دست دادن بسته:

- کاهش نرخ ارسال به نصف

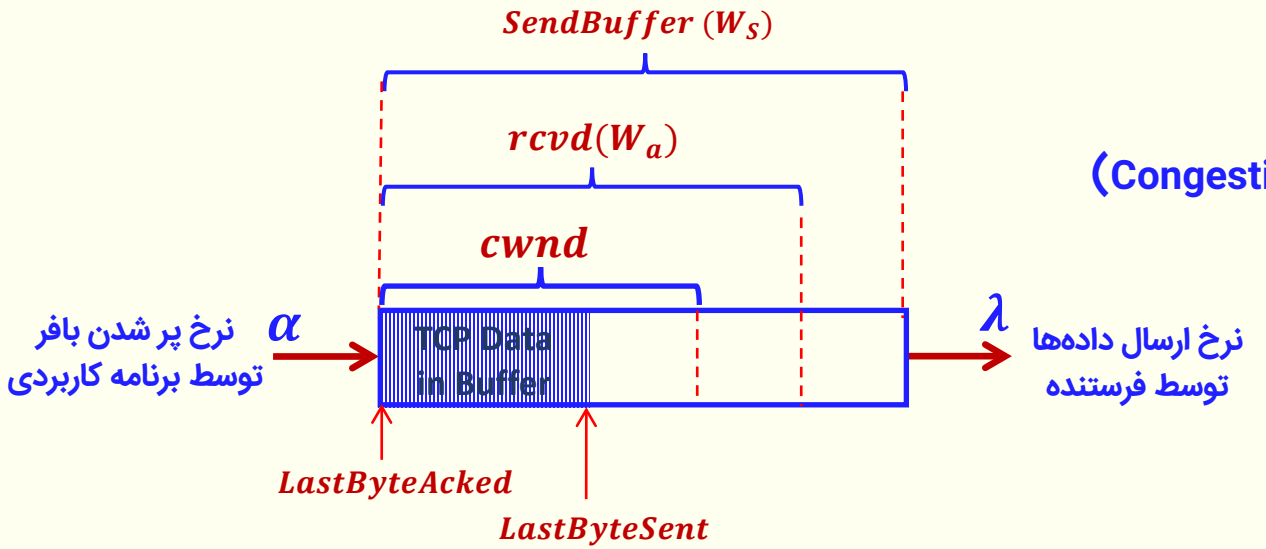


AIMD sawtooth behavior:
probing for bandwidth

کنترل ازدحام پروتکل TCP: AIMD

روش‌های حلقه بسته یا واکنشی – (closed loop or reactive)

- هدف: رسیدن به حداکثر گذردهی و کنترل ازدحام
- تنظیم نرخ ارسال بر اساس تنظیم پنجره ازدحام (Congestion Window - $cwnd$)
- بازخورد از شبکه:
  - تشخیص وجود ازدحام:
    - بر اساس مشاهدات به صورت ضمنی:
      - منقضی شدن زمانبند
      - دریافت سه تاییده تکراری
    - در صورت پشتیبانی مسیریاب‌های شبکه به صورت صریح:
      - استفاده از بیت‌های ECN (Explicit Congestion Notification)
    - کاهش نرخ ارسال با کاهش اندازه پنجره ازدحام
  - تشخیص وضعیت بدون ازدحام:
    - دریافت تاییده ارسال سگمنت
    - افزایش نرخ ارسال با افزایش اندازه پنجره ازدحام



$$\left.\begin{aligned} LastByteSent - LastByteAcked &\le W_S \\ LastByteSent - LastByteAcked &\le W_a \\ LastByteSent - LastByteAcked &\le cwnd \end{aligned}\right\}$$

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

کنترل ازدحام پروتکل TCP: تنظیم اندازه پنجره ازدحام

تعریف حد آستانه ازدحام (ssthresh) و تنظیم پویای آن بر اساس پنجره ازدحام در زمان وقوع ازدحام ($ssthresh = \frac{cwnd}{2}$)

تشخیص وضعیت بدون ازدحام:

- دریافت تاییده ارسال سگمنت
- افزایش اندازه پنجره ارسال:
  - احتمال پایین وقوع ازدحام (شروع آهسته کنترل ازدحام) slow start

اگر $cwnd < ssthresh$ آنگاه بعد از هر RTT

$$cwnd = cwnd \times 2$$

  - احتمال بالای وقوع ازدحام (اجتناب از ازدحام) congestion avoidance

اگر $cwnd \geq ssthresh$ آنگاه بعد از هر RTT

$$cwnd = cwnd + 1\ \text{MSS}$$

تشخیص ازدحام (congestion occur):

- تنظیم حد آستانه ازدحام ($ssthresh = \frac{cwnd}{2}$)
- تشخیص ازدحام شدید: منقضی شدن زمانبند (timeout)
  - کاهش شدید اندازه پنجره ازدحام

$$cwnd = 1\ \text{MSS}$$

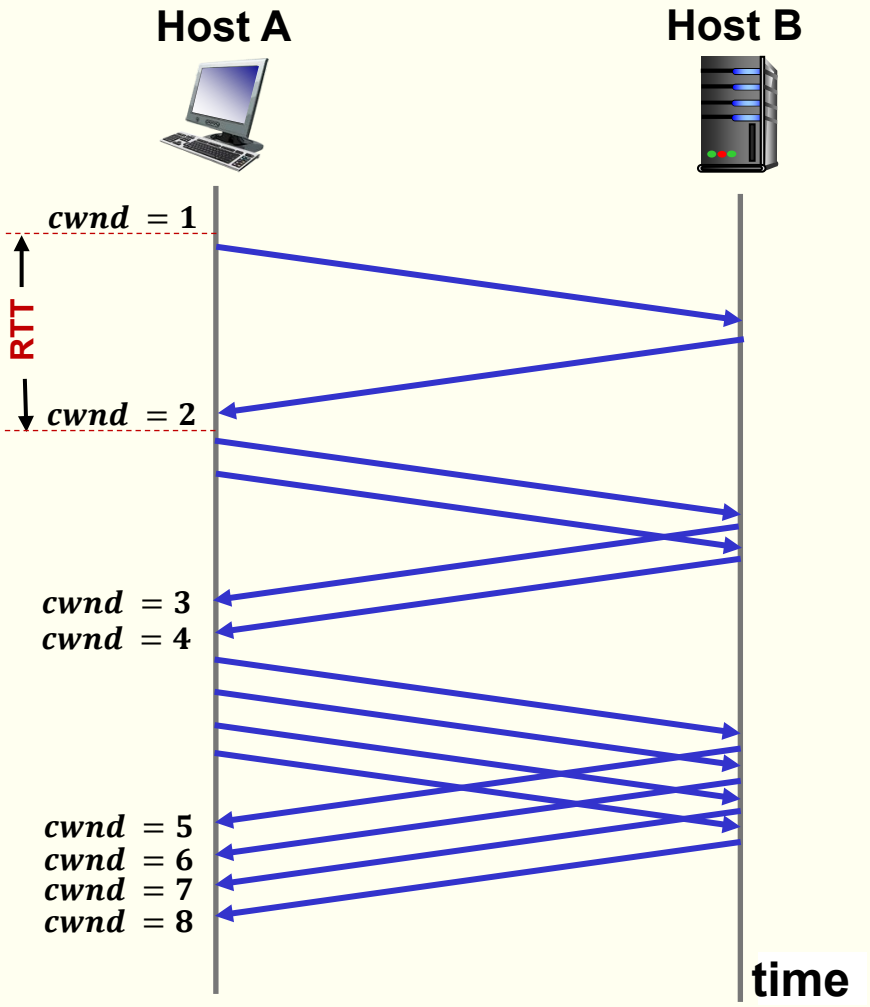
- تشخیص ازدحام خفیف: دریافت سه تاییده تکراری
  - TCP Tahoe: کاهش شدید اندازه پنجره ازدحام

$$cwnd = 1\ \text{MSS}$$

  - TCP Reno: کاهش کم اندازه پنجره ازدحام

$$cwnd = ssthresh$$

کنترل ازدحام پروتکل TCP: تنظیم اندازه پنجره ازدحام



شروع آهسته (slow start) کنترل ازدحام:

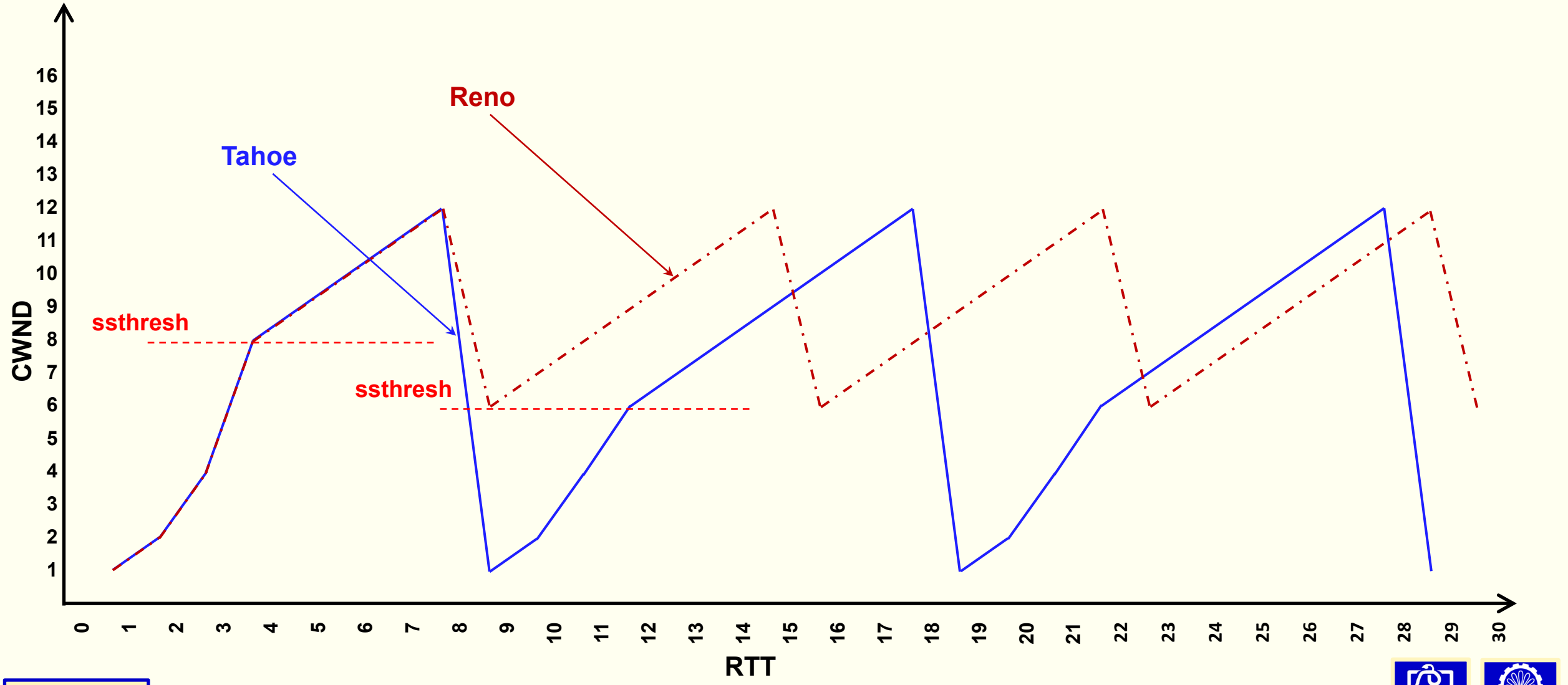
- با دریافت هر تاییده پنجره ازدحام یک MSS اضافه می‌شود (بعد از یک RTT پنجره ازدحام ۲ برابر می‌شود.)

اجتناب از ازدحام (congestion avoidance):

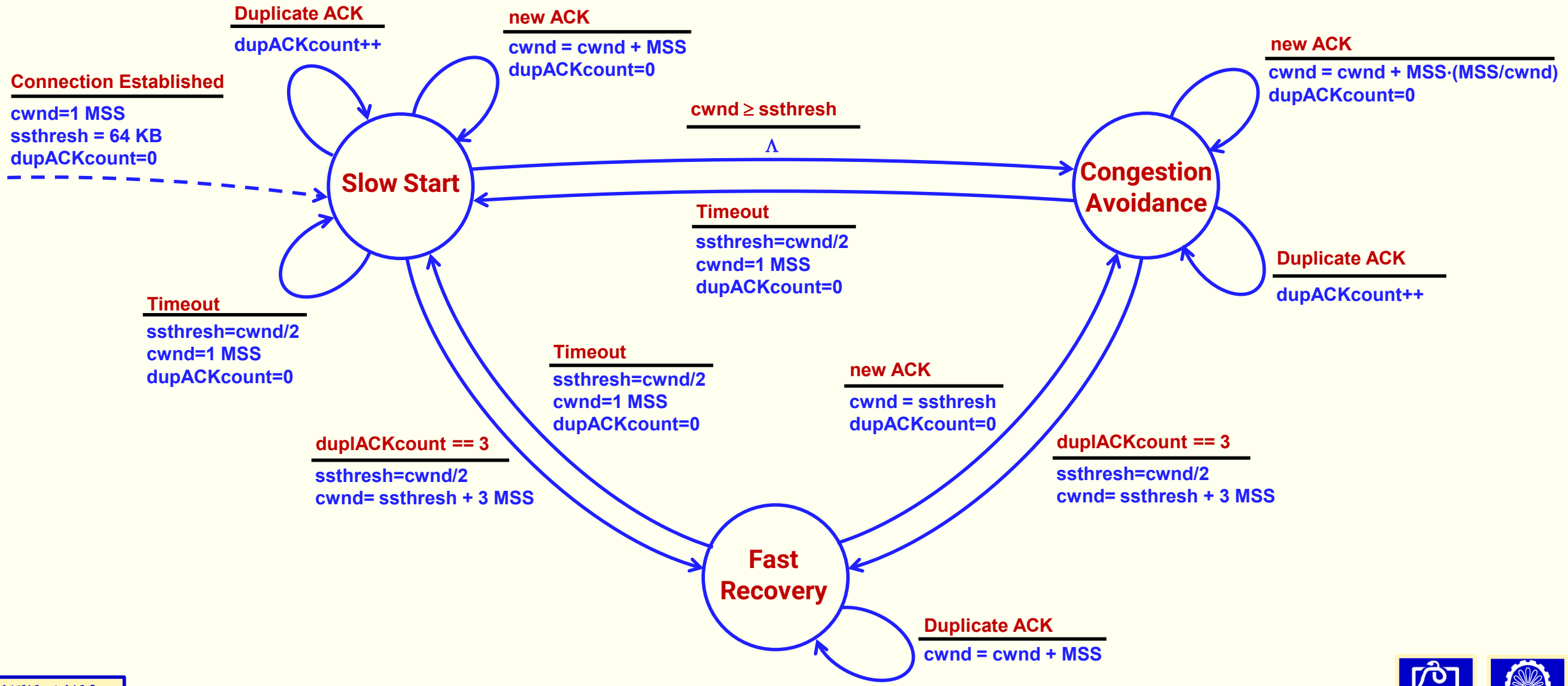
- با دریافت هر تاییده پنجره ازدحام به اندازه $MSS \cdot (MSS/cwnd)$ اضافه می‌شود (بعد از یک RTT پنجره یک MSS اضافه می‌شود).

لایه انتقال

کنترل ازدحام پروتکل TCP: مثالی از نمودار زمانی تنظیم اندازه پنجره ازدحام

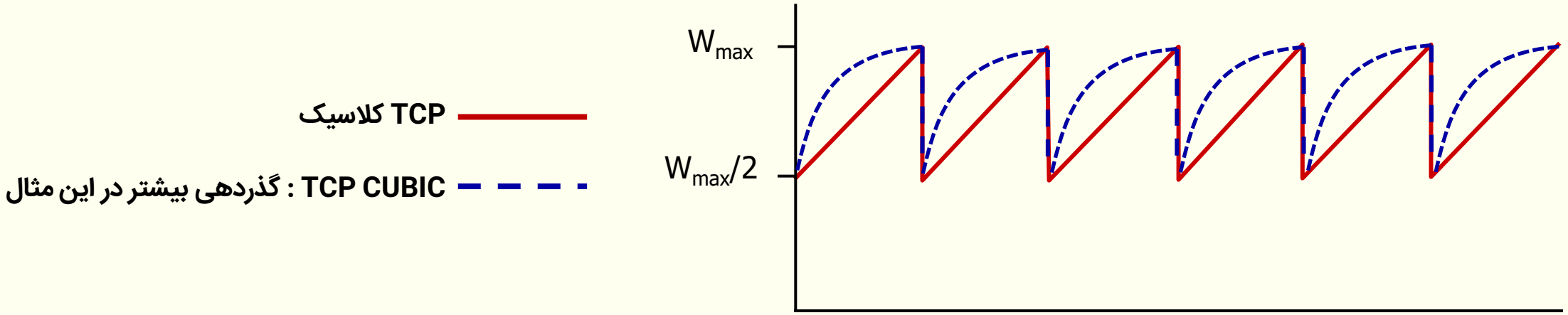


کنترل ازدحام پروتکل TCP: دیاگرام حالت کنترل ازدحام Reno



کنترل ازدحام پروتکل TCP: TCP درجه ۳ (TCP CUBIC)

- آیا راهی بهتر از AIMD برای بررسی پهنای باند قابل استفاده وجود دارد؟
- درک و فهم مشاهدات:
  - W_{max}: نرخ ارسال در زمان وقوع ازدحام (از دست دادن بسته)
  - احتمالاً (؟) وضعیت در لینک دارای ازدحام گلوگاه تغییر زیادی نکرده است.
  - با وقوع ازدحام و بعد از کاهش نرخ (پنجره) ارسال به نصف، ابتدا نرخ ارسال با شیب تند به W_{max} افزایش می‌یابد و سپس به آهستگی به W_{max} نزدیک می‌شود.

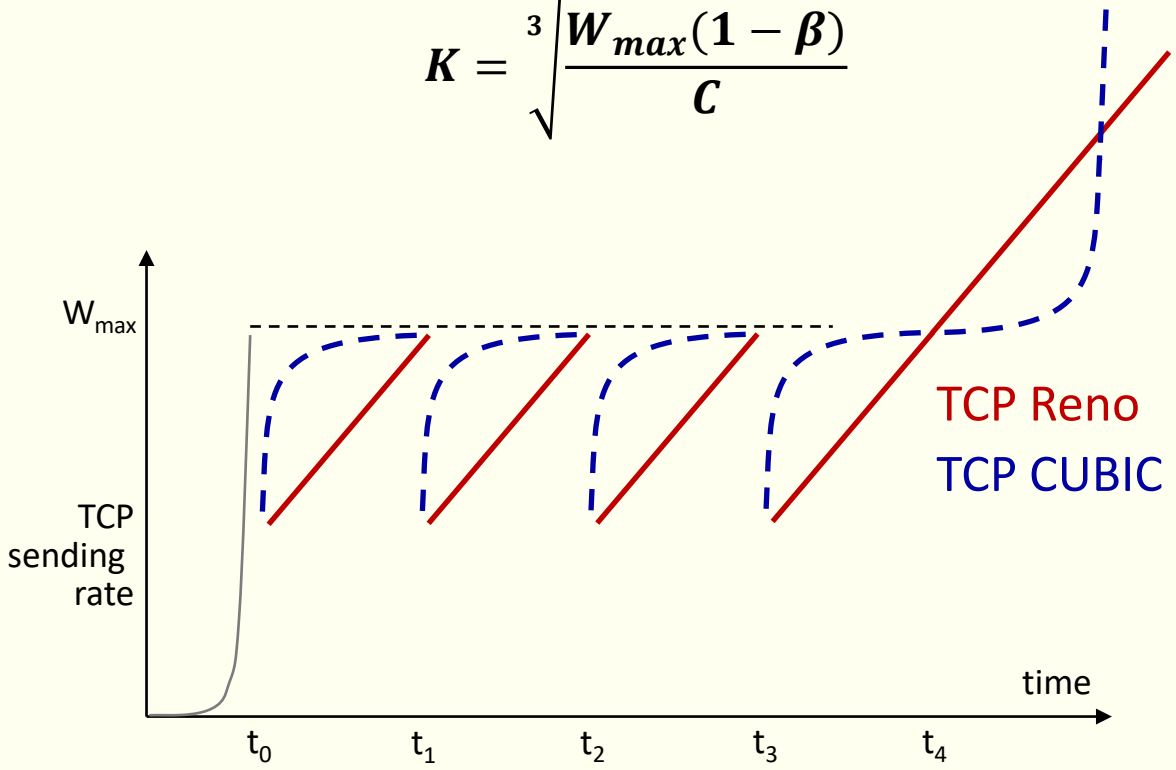


کنترل ازدحام پروتکل TCP: TCP CUBIC، RFC ۸۳۱۲

- تعریف متغیرها:
  - β: فاکتور کاهش (در RFC 8312، مقدار آن 0.7 تعیین شده است)
  - C: ضریب ثابت مقیاس (در RFC 8312، مقدار آن 0.4 تعیین شده است)
  - T: زمان سپری شده از آخرین کاهش اندازه پنجره (ازدحام)
  - W_{max}: پنجره ازدحام در زمان وقوع ازدحام
  - $cwnd$: پنجره ازدحام

$$cwnd = C(T-K)^3 + W_{max}$$

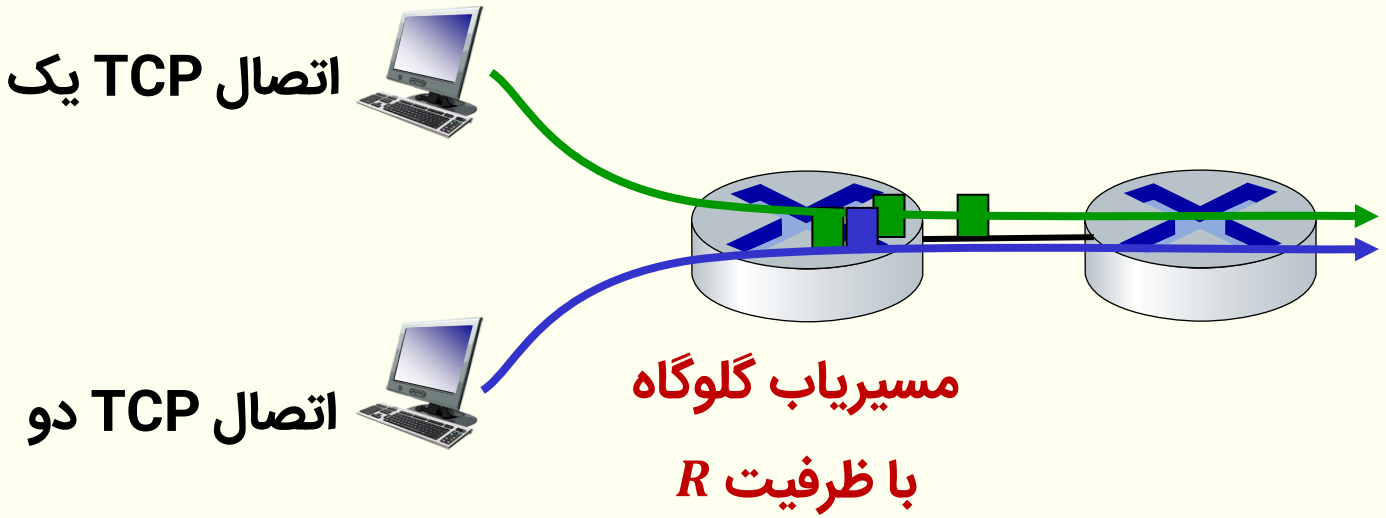
$$K = \sqrt[3]{\frac{W_{max}(1-\beta)}{C}}$$



- الگوریتم TCP CUBIC به طور پیش فرض در سیستم عامل لینوکس استفاده می‌شود.

کنترل ازدحام پروتکل TCP: رعایت عدالت بین جریان‌های ترافیکی (TCP Fairness)

هدف عدالت: منظور از عدالت این است که اگر K اتصال TCP یک لینک گلوگاه مشترک با پهنای باند R داشته باشند، این پهنای به طور مساوی بین آن تقسیم شود و هر اتصال نرخ ارسال متوسط R/K داشته باشد.



کنترل ازدحام پروتکل TCP: رعایت عدالت بین جریان‌های ترافیکی (TCP Fairness)

وقوع ازدحام:

- در زمان وقوع ازدحام یک بسته حذف می‌شود. به طور احتمالی بسته حذف شده از جریان ترافیکی است که نرخ ارسال بالاتری دارد.
- بعداز وقوع ازدحام جریانی که بسته از دست داده است، نرخ ارسالش کم می‌شود و بقیه جریان‌ها افزایش نرخ خواهند داشت.
- تکرار این حالت و تقسیم مساوی پهنای‌باند بین جریان‌های فعال $(R/n(t))$