



第五部分 传输层

第 23 章

UDP、TCP 和 SCTP

23-1 进程到进程的传递

传输层负责进程到进程的传递，即进程之间的分组传递以及部分消息传递。后面将会看到两个进程以客户/服务器的方式通信。

本节要点:

客户端/服务器模式

复用和分离

无连接服务与面向连接的服务

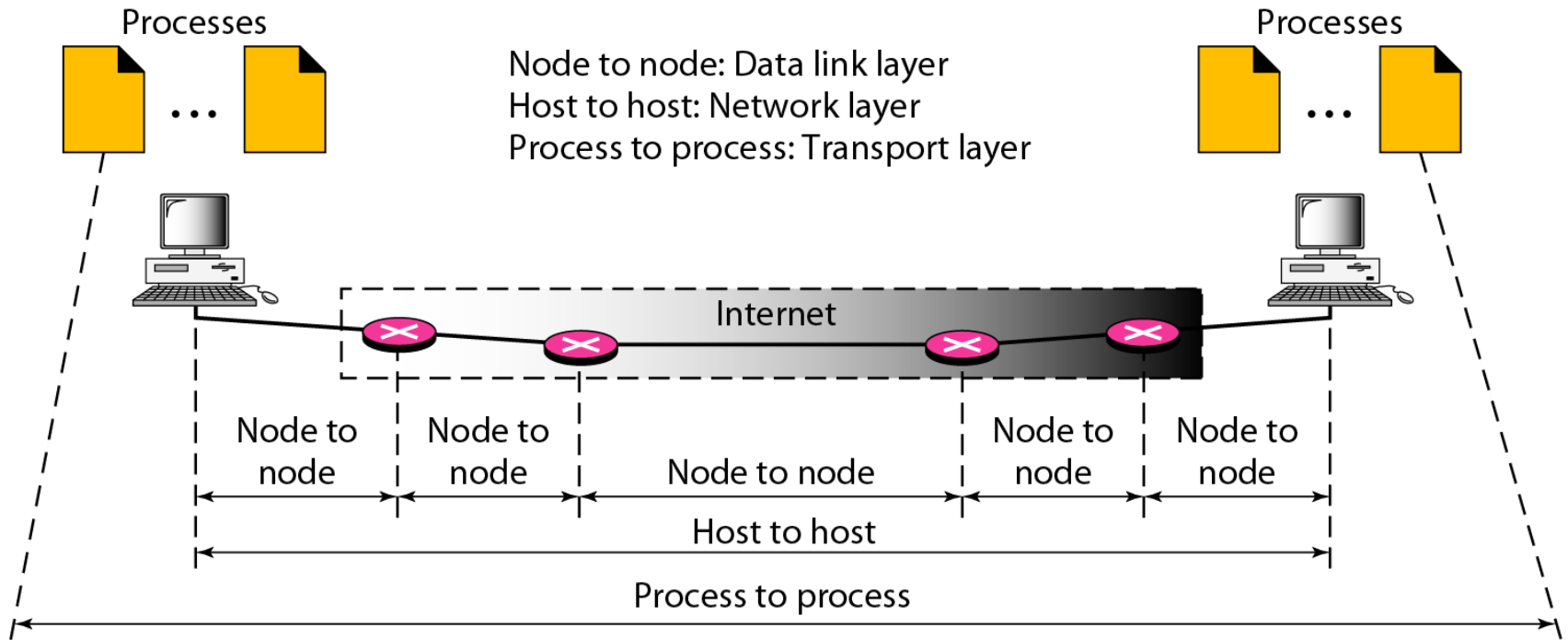
可靠服务与不可靠服务

三种协议

传输层特性

- ◆ 网络层监管独立分组从源端到目的端的传递，但不辨认分组之间的关系；传输层负责进程到进程的传递，监管差错控制和流量控制，确保全部报文完整地按序到达。
- ◆ 传输层的流量控制和差错控制是端到端的。
- ◆ 传输层协议可以是无连接的，也可以是面向连接的。
- ◆ 在传输层，一个报文通常被划分为可传输的段。无连接的协议对每段独立处理，面向连接的协议需要序号生成这些段之间的关系。

图 23.1 数据传送类型



寻址

- 数据链路层：节点到节点 —— MAC地址
- 网络层：主机到主机 —— IP地址
- 传输层：进程到进程 —— 端口号

16位，0 ~ 65535之间的整数

- ◆ 客户端：临时端口号
- ◆ 服务器端：熟知端口号

图 23.2 端口号

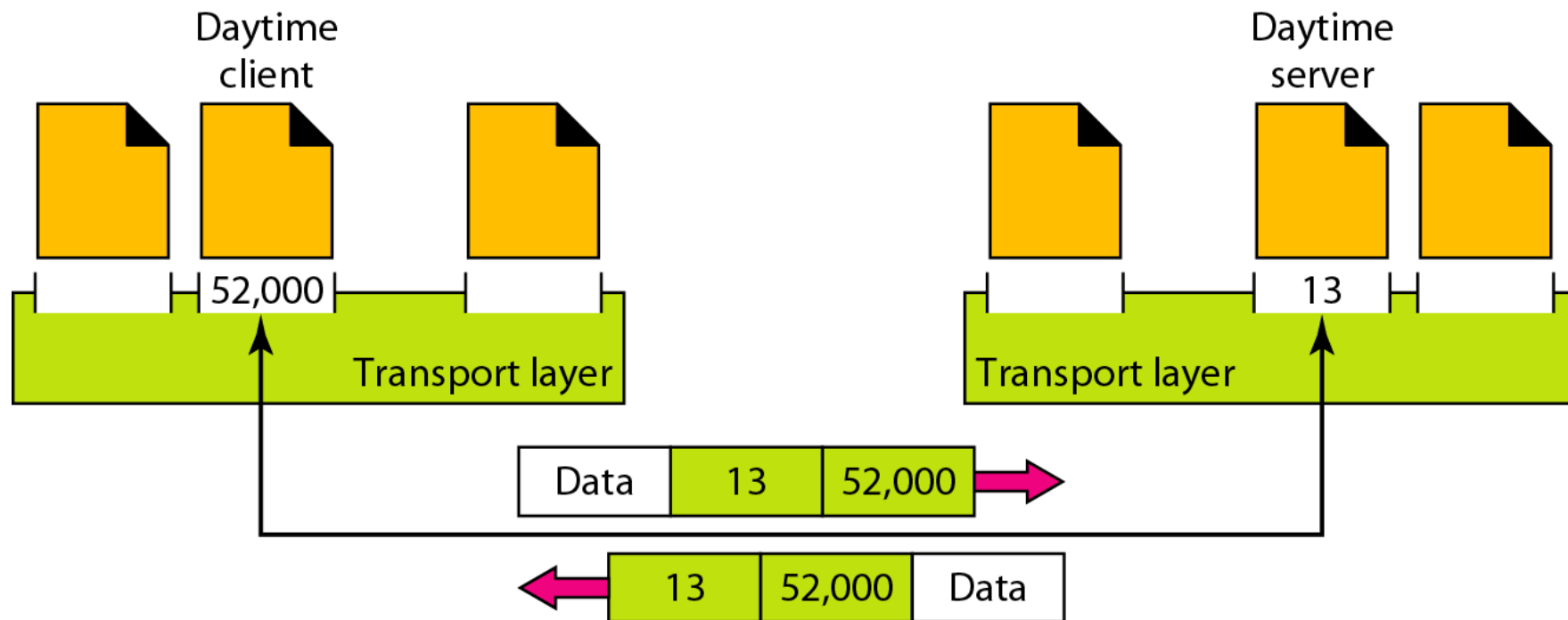
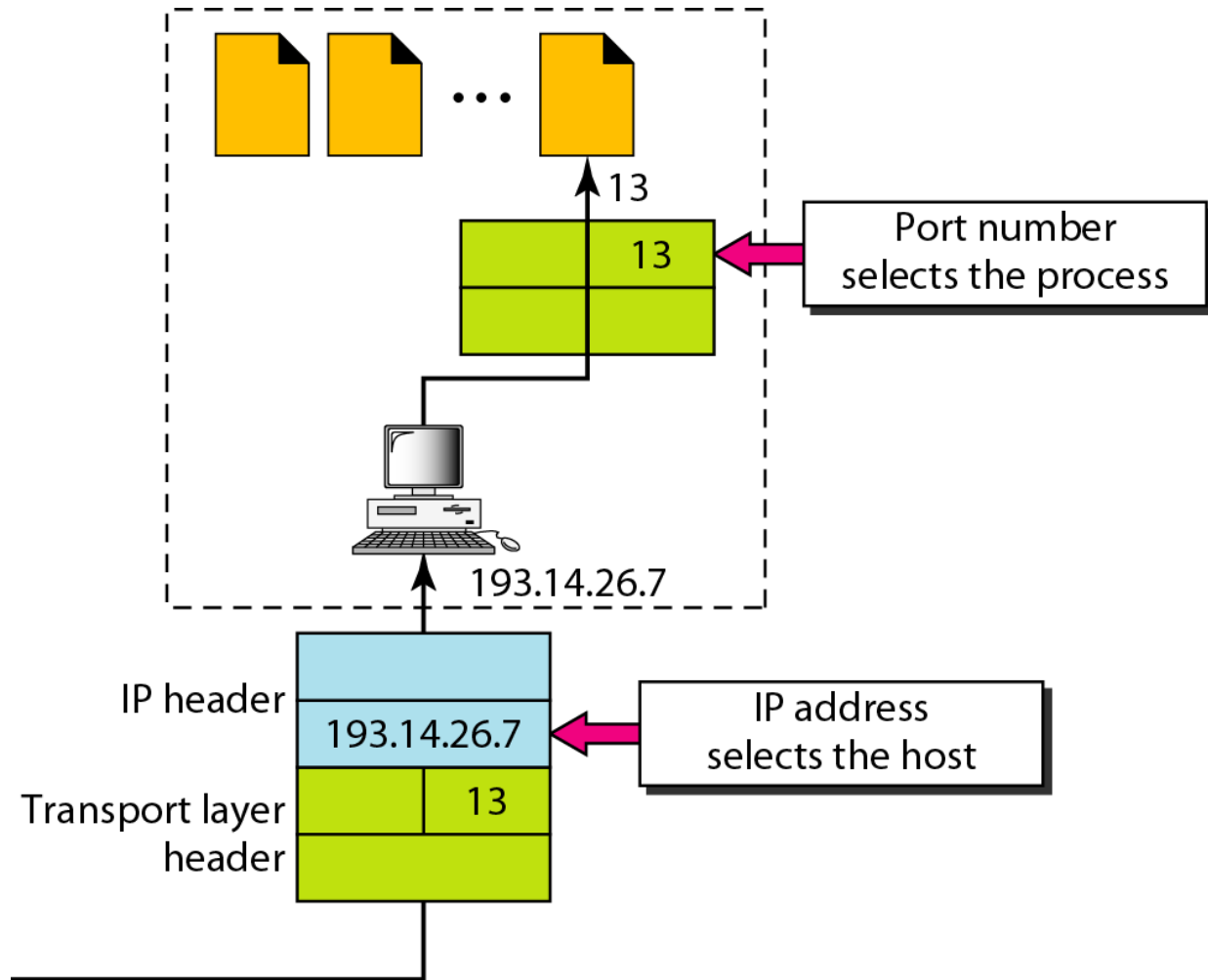


图 23.3 IP地址与端口号



端口号

- 熟知端口号：0 ~ 1023，由IANA分配和控制
- 注册端口号：1024 ~ 49151，在IANA注册
- 动态端口号：49152 ~ 65535，临时使用

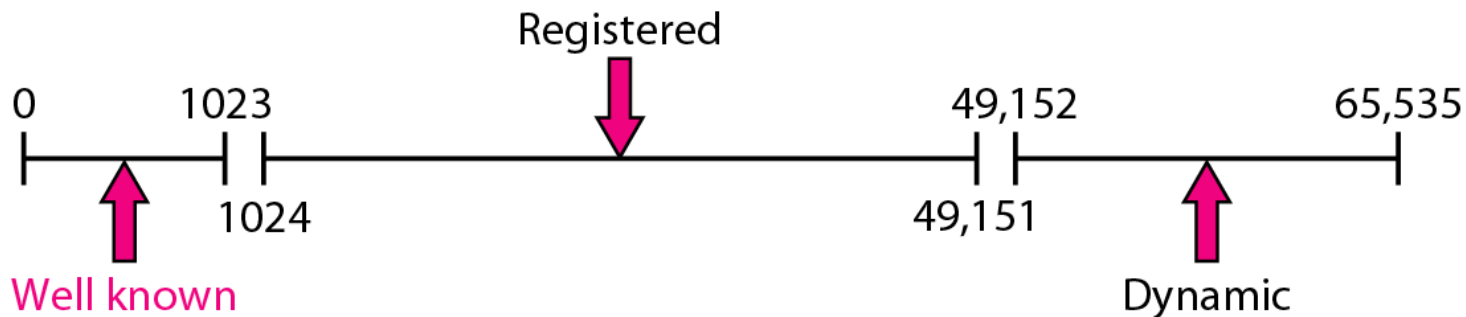
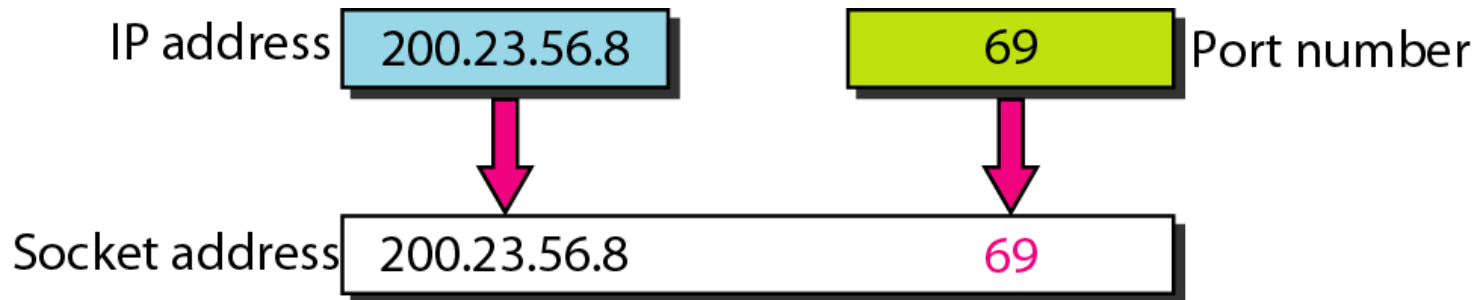


图 23.4 IANA 定义的范围

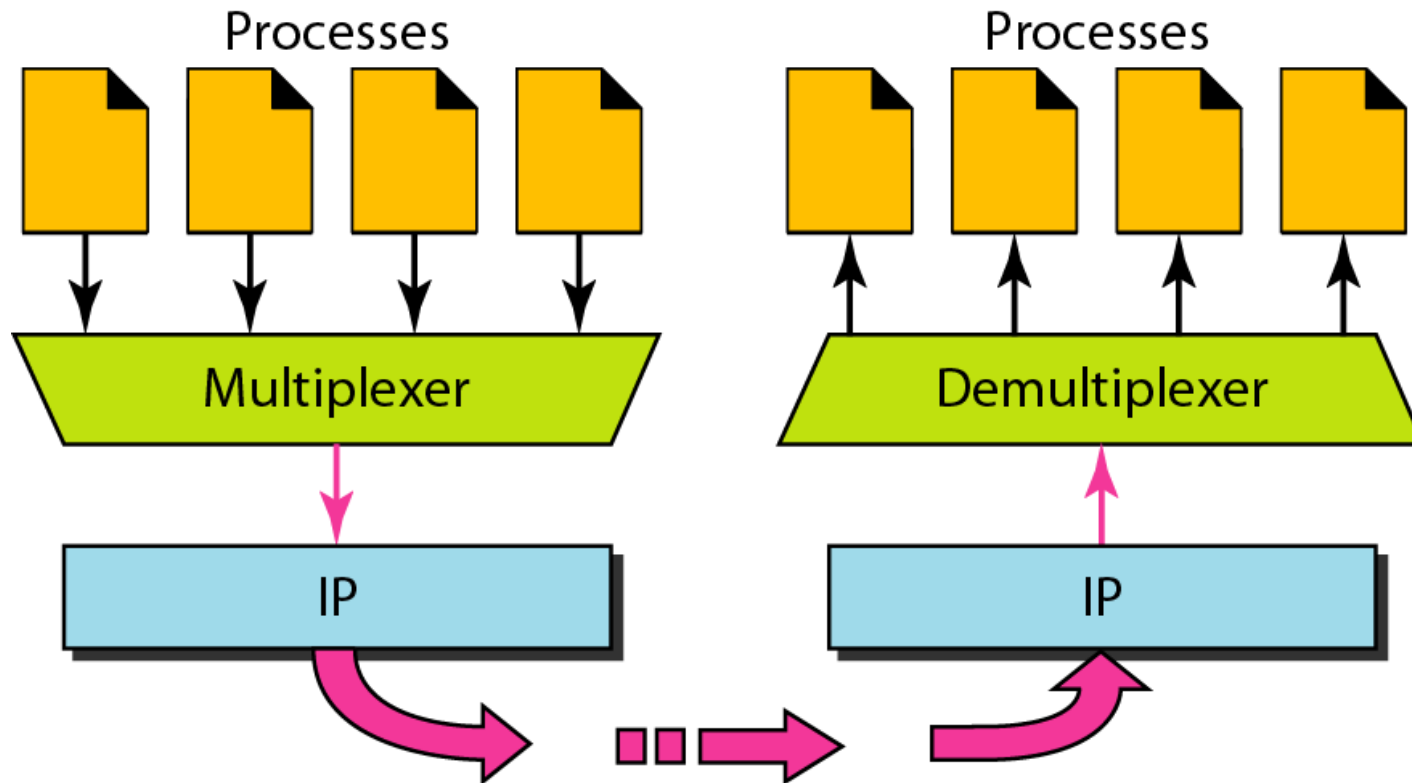
IANA (Internet Assigned Number Authority): 因特网号码分配管理局

图 23.5 套接字地址



- IP地址+端口 = 套接字
- 套接字唯一标识了一个进程
- 一个TCP连接被一对套接字地址确定

图 23.6 复用和分离

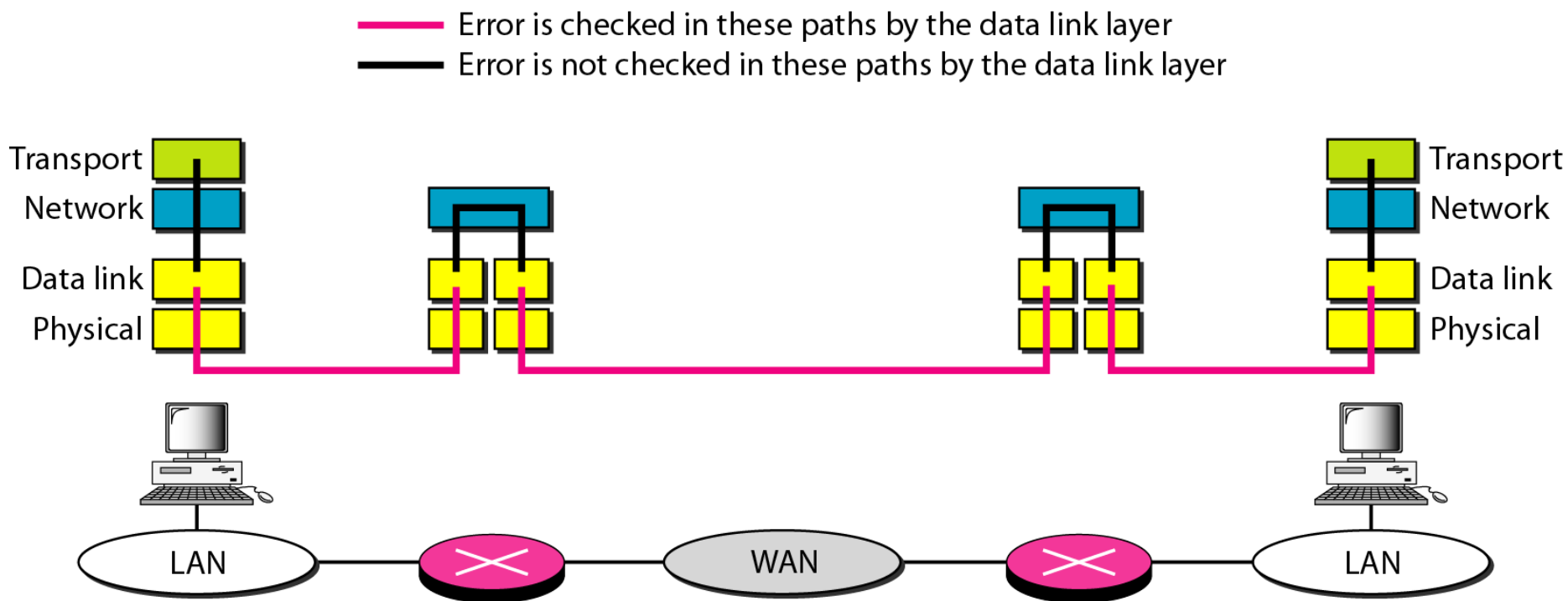


服务

- 无连接服务(connectionless service)
 - 不可靠服务，无连接，无确认，无编号
 - 延迟、丢失、重复、失序
 - UDP

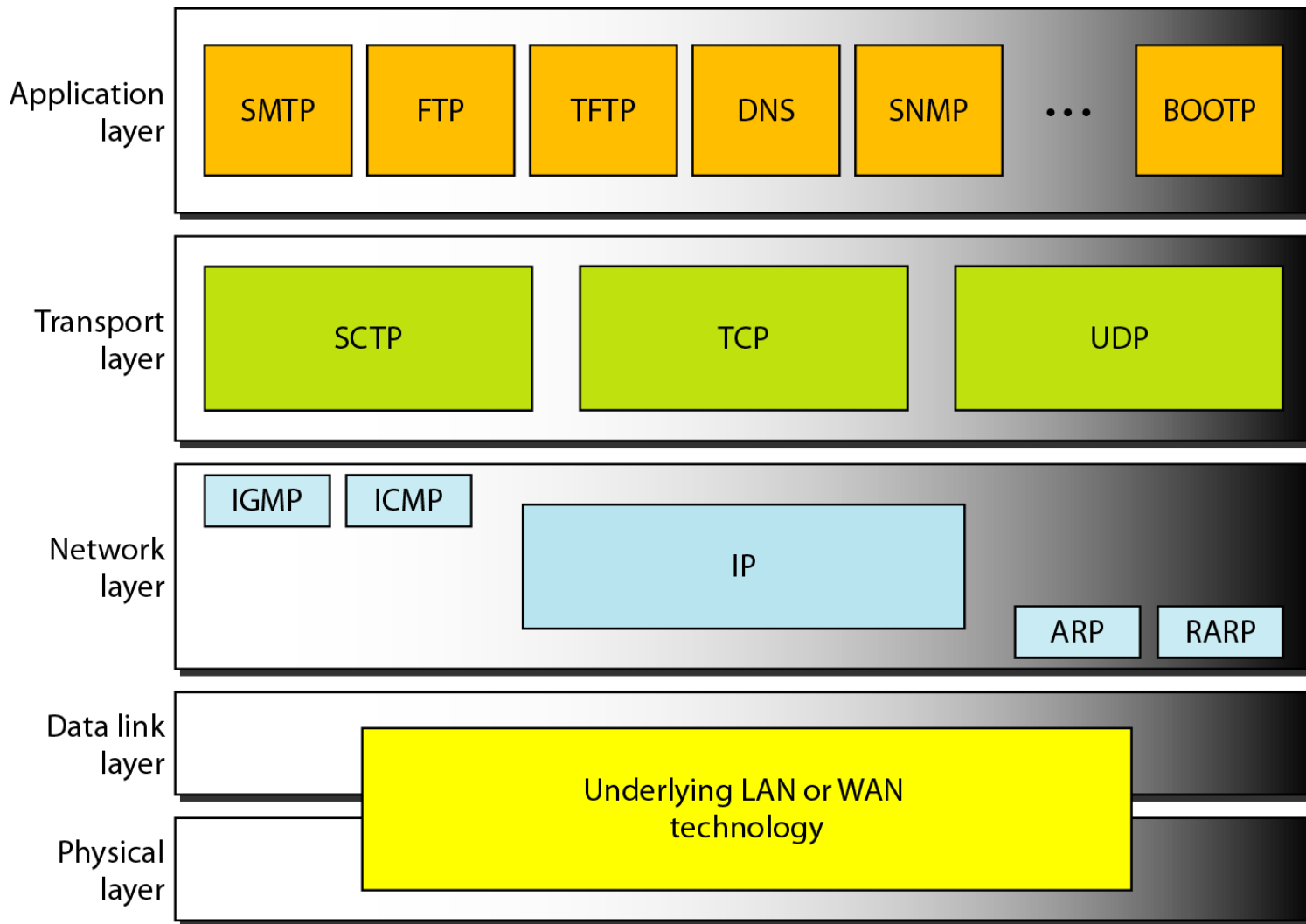
- 面向连接的服务(connection-oriented service)
 - 可靠服务，建立连接
 - TCP、SCTP

图 23.7 差错控制



数据链路层的可靠性存在于两个节点之间，网络层提供不可靠的服务，因此端到端的可靠性需要在传输层实现。

图 23.8 UDP、TCP和SCTP在TCP/IP协议簇中的位置



TCP和UDP协议的应用场景

TCP的应用场景：

- ◆ 客户端程序和服务器端程序需要多次交互才能实现特定功能应用。如接收电子邮件POP3，发送电子邮件SMTP，传输文件FTP等；
- ◆ 应用程序传输的文件需要分段传输。如浏览器访问网页时网页中的图片和HTML文件需要分段后发送给浏览器，QQ传文件时也需要进行分段，此时使用TCP协议。

UDP的应用场景：

- ◆ 客户端程序和服务器端程序通信，应用程序发送的数据包不需要分段。如域名解析时的请求报文和返回的解析结果；
- ◆ 实时通信如QQ或微信语音聊天。发送方和接收方需要实时交互，不允许较长时延；
- ◆ 多播或广播通信。

23-2 用户数据报协议UDP (User Datagram Protocol)

用户数据报协议(UDP) 称为无连接的不可靠的传输层协议。它除了提供进程到进程而不是主机到主机的通信外，没有给IP服务增加任何东西。协议简单，开销小。

本节要点:

UDP的熟知端口

用户数据报

校验和

UDP的操作

UDP的使用

表 23.1 UDP使用的熟知端口

<i>Port</i>	<i>Protocol</i>	<i>Description</i>
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Nameserver	Domain Name Service
67	BOOTPs	Server port to download bootstrap information
68	BOOTPc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

例 23.1

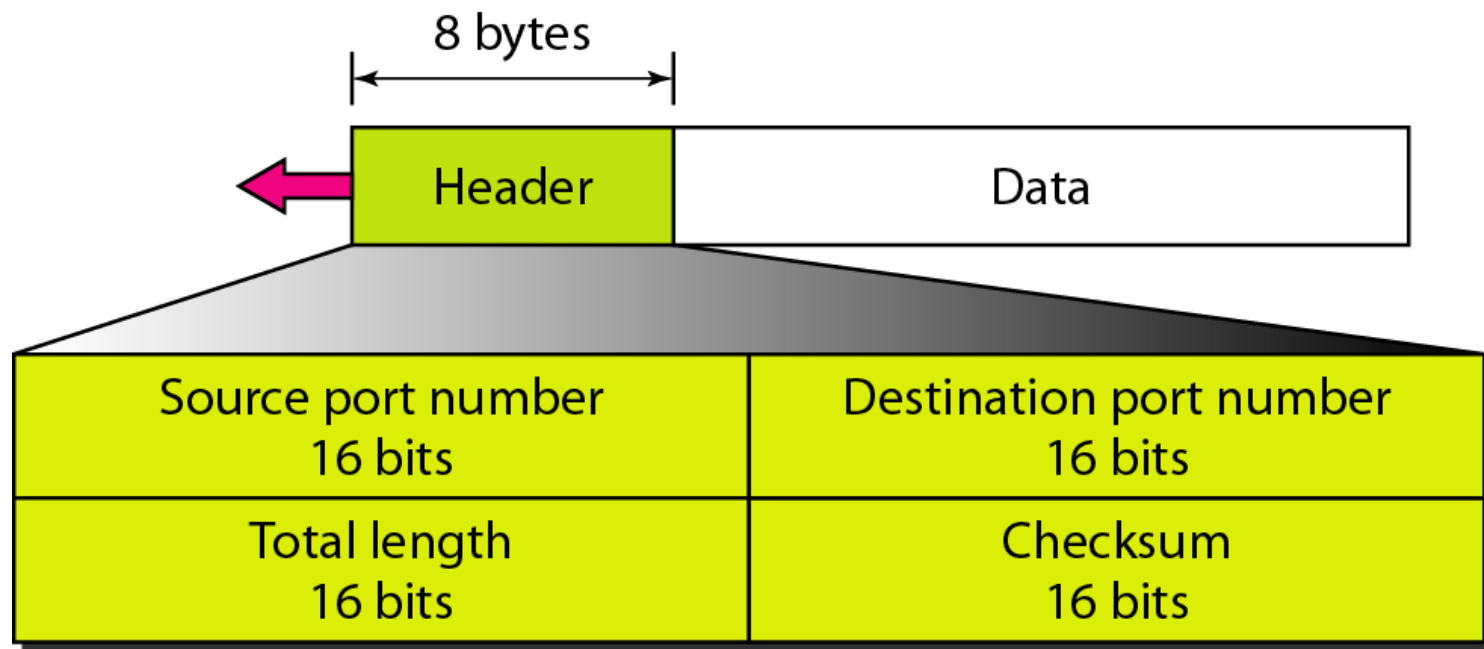
在UNIX中，熟知端口存储在/etc/services文件中。这个文件中的每行给出服务器名和端口号。我们可以用grep命令提取该行所对应的应用。下面表示了FTP的端口。 注：FTP可使用UDP或TCP的端口号是21。

```
$ grep ftp /etc/services
ftp      21/tcp
ftp      21/udp
```

SNMP 使用两个端口号(161 and 162), 在28章将看到，它们每个都用于不同目的。

```
$ grep snmp /etc/services
snmp      161/tcp      #Simple Net Mgmt Proto
snmp      161/udp      #Simple Net Mgmt Proto
snmptrap  162/udp      #Traps for SNMP
```

图 23.9 用户数据报格式



$$\text{UDP长度} = \text{IP长度} - \text{IP首部长度}$$

图 23.10 用于校验和计算的伪头部

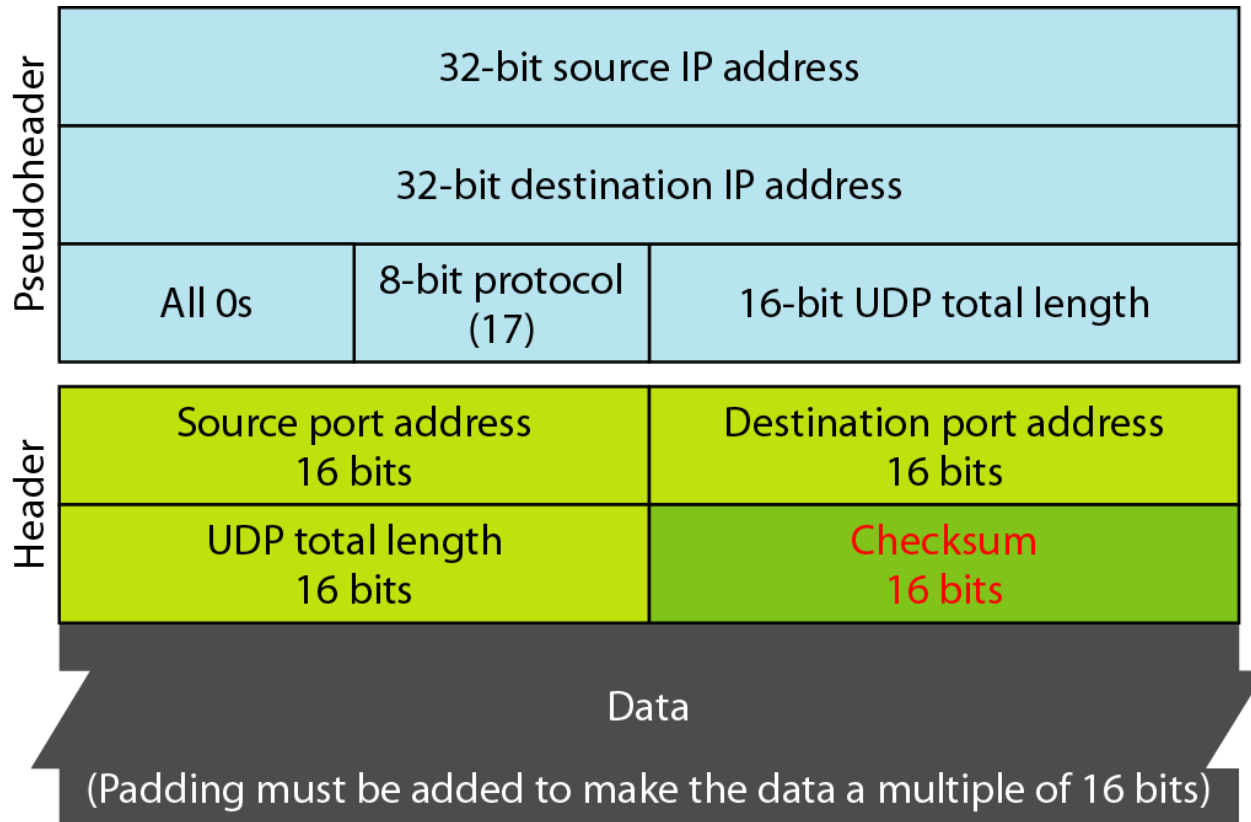


图23.11给出了只有7个字节数据的很小的用户数据报的校验和计算。因为数据的字节数是奇数，因此为了计算校验和需要填充。当用户数据报传递给IP时，就将伪头部和填充部分丢弃。

图 23.11 简单UDP用户数据报的校验和计算

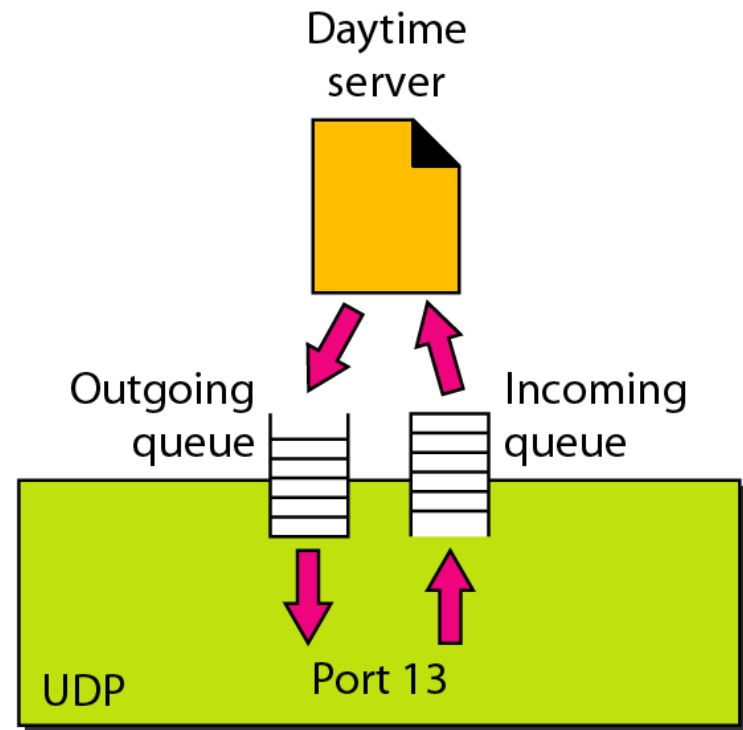
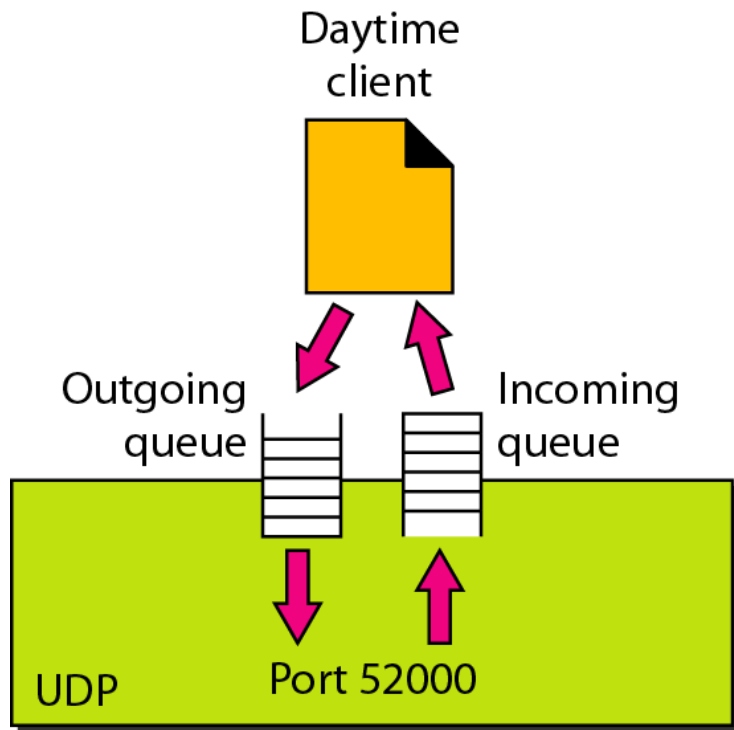
153.18.8.105			
171.2.14.10			
All 0s	17	15	
1087		13	
15		All 0s	
T	E	S	T
I	N	G	All 0s

10011001	00010010	→	153.18
00001000	01101001	→	8.105
10101011	00000010	→	171.2
00001110	00001010	→	14.10
00000000	00010001	→	0 and 17
00000000	00001111	→	15
00000100	00111111	→	1087
00000000	00001101	→	13
00000000	00001111	→	15
00000000	00000000	→	0 (checksum)
01010100	01000101	→	T and E
01010011	01010100	→	S and T
01001001	01001110	→	I and N
01000111	00000000	→	G and 0 (padding)
<hr/>			
10010110	11101011	→	Sum
01101001	00010100	→	Checksum

UDP的特点

- 无连接的服务。独立数据报，无编号，无连接，沿不同路径传递，减小了建立连接的开销和时延。
- 尽最大努力交付。不保证可靠传输，无需维持复杂的连接状态，节省系统资源。
- 面向报文。对应用层交付的报文，既不合并也不拆分，一次交付一个完整的报文。应用程序需选择合适大小的报文，报文太长时由网络层分片。
- 无流量控制和差错控制。没有流量控制，也没有窗口机制；除校验和之外没有差错控制，使用UDP的进程自己必须要有流控和差错控制。接收方可能溢出，报文可能丢失或重复，出错时悄悄丢弃。
- 无拥塞控制。网络出现拥塞时不会使源主机的发送速率降低，适用于允许网络拥塞时丢失一些数据但不允许太大时延的实时应用。
- 支持一对一、一对多、多对一和多对多的通信。
- 首部开销小。

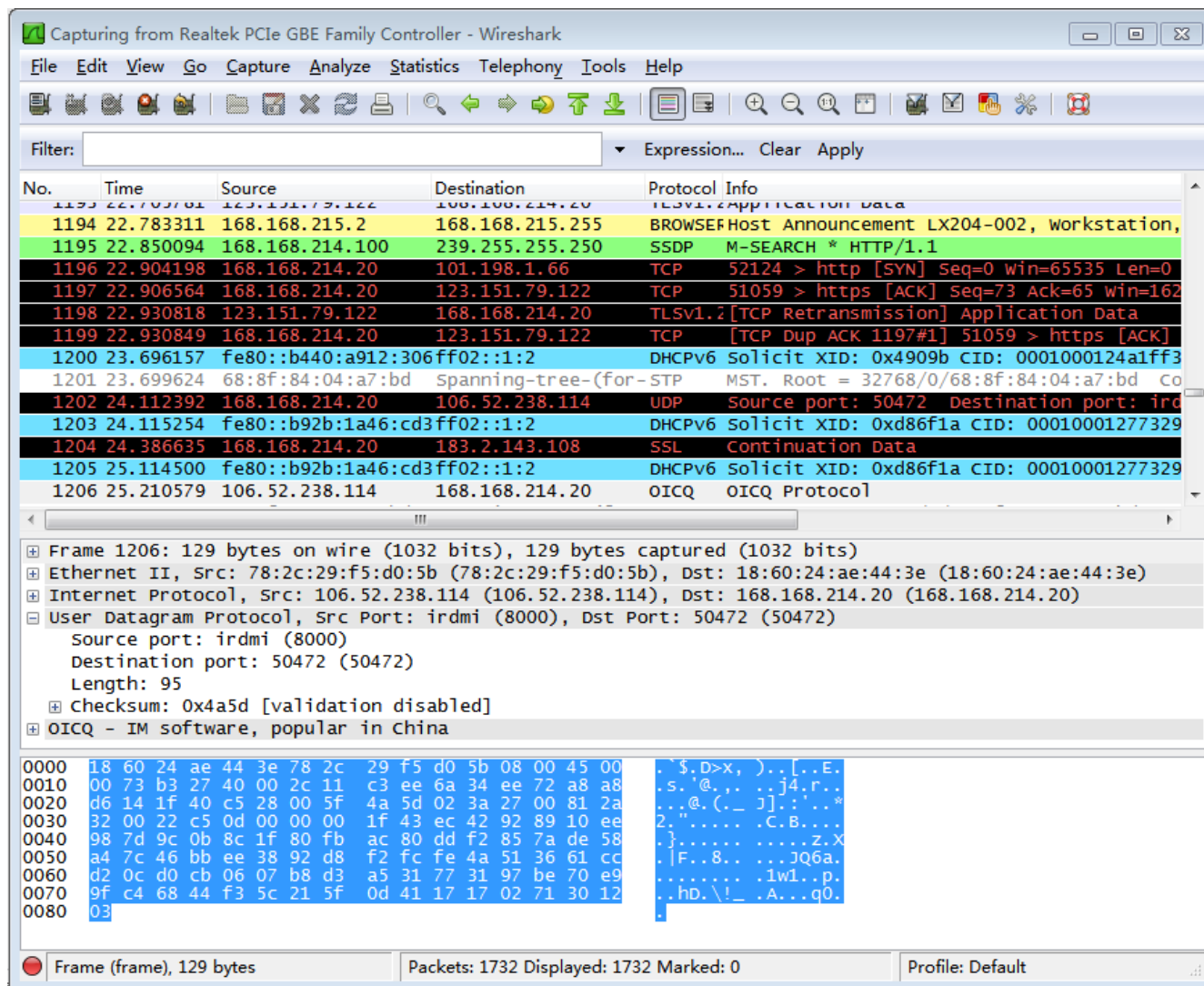
图 23.12 UDP中的队列



UDP的使用

- 适用于需要有简单请求 - 响应通信的进程，较少考虑流量控制和差错控制：P2P
- 适用于具有内部流量控制和差错控制的进程：TFTP
- 适用于多播
- 适用于网络管理：SNMP
- 适用于某些路由协议：RIP

通过抓包查看UDP首部



Capturing from Realtek PCIe GBE Family Controller - Wireshark

File Edit View Go Capture Analyze Statistics Telephony Tools Help

Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Info
1193	22.707881	123.151.79.122	168.168.214.20	TLSv1.2	Application Data
1194	22.783311	168.168.215.2	168.168.215.255	BROWSERHost	Announcement LX204-002, workstation,
1195	22.850094	168.168.214.100	239.255.255.250	SSDP	M-SEARCH * HTTP/1.1
1196	22.904198	168.168.214.20	101.198.1.66	TCP	52124 > http [SYN] Seq=0 win=65535 Len=0
1197	22.906564	168.168.214.20	123.151.79.122	TCP	51059 > https [ACK] Seq=73 Ack=65 win=162
1198	22.930818	123.151.79.122	168.168.214.20	TLSv1.2	[TCP Retransmission] Application Data
1199	22.930849	168.168.214.20	123.151.79.122	TCP	[TCP Dup ACK 1197#1] 51059 > https [ACK]
1200	23.696157	fe80::b440:a912:306ff02::1:2		DHCPv6	Solicit XID: 0x4909b CID: 0001000124a1ff3
1201	23.699624	68:8f:84:04:a7:bd	Spanning-tree-(for-STP	MST.	Root = 32768/0/68:8f:84:04:a7:bd Co
1202	24.112392	168.168.214.20	106.52.238.114	UDP	Source port: 50472 Destination port: ird
1203	24.115254	fe80::b92b:1a46:cd3ff02::1:2		DHCPv6	Solicit XID: 0xd86f1a CID: 00010001277329
1204	24.386635	168.168.214.20	183.2.143.108	SSL	Continuation Data
1205	25.114500	fe80::b92b:1a46:cd3ff02::1:2		DHCPv6	Solicit XID: 0xd86f1a CID: 00010001277329
1206	25.210579	106.52.238.114	168.168.214.20	OICQ	OICQ Protocol

Frame 1206: 129 bytes on wire (1032 bits), 129 bytes captured (1032 bits)

Ethernet II, Src: 78:2c:29:f5:d0:5b (78:2c:29:f5:d0:5b), Dst: 18:60:24:ae:44:3e (18:60:24:ae:44:3e)

Internet Protocol, Src: 106.52.238.114 (106.52.238.114), Dst: 168.168.214.20 (168.168.214.20)

User Datagram Protocol, Src Port: irdmi (8000), Dst Port: 50472 (50472)

Source port: irdmi (8000)

Destination port: 50472 (50472)

Length: 95

Checksum: 0x4a5d [validation disabled]

OICQ - IM software, popular in China

0000 18 60 24 ae 44 3e 78 2c 29 f5 d0 5b 08 00 45 00 .\$.D>x,)..[.E.
0010 00 73 b3 27 40 00 2c 11 c3 ee 6a 34 ee 72 a8 a8 .s.'@.,. .j4.r..
0020 d6 14 1f 40 c5 28 00 5f 4a 5d 02 3a 27 00 81 2a ..@.(.]].:.*
0030 32 00 22 c5 0d 00 00 00 1f 43 ec 42 92 89 10 ee Z.".....C.B....
0040 98 7d 9c 0b 8c 1f 80 fb ac 80 dd f2 85 7a de 58 .}.....Z.X
0050 a4 7c 46 bb ee 38 92 d8 f2 fc fe 4a 51 36 61 cc |F.8.JQ6a.
0060 d2 0c d0 cb 06 07 b8 d3 a5 31 77 31 97 be 70 e91w1..p.
0070 9f c4 68 44 f3 5c 21 5f 0d 41 17 17 02 71 30 12 ..hd.\!_ .A...q0.
0080 03

Frame (frame), 129 bytes Packets: 1732 Displayed: 1732 Marked: 0 Profile: Default

23-3 传输控制协议TCP

(Transmission Control Protocol)

TCP是一个面向连接的协议，它在两个TCP之间建立一个虚拟连接来发送数据。另外，TCP在传输层使用流量控制和差错控制机制。

本节要点:

TCP 服务

TCP 特点

段

TCP连接

流量控制

差错控制

表 23.2 TCP使用的熟知端口

<i>Port</i>	<i>Protocol</i>	<i>Description</i>
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

图 23.13 字节流传递

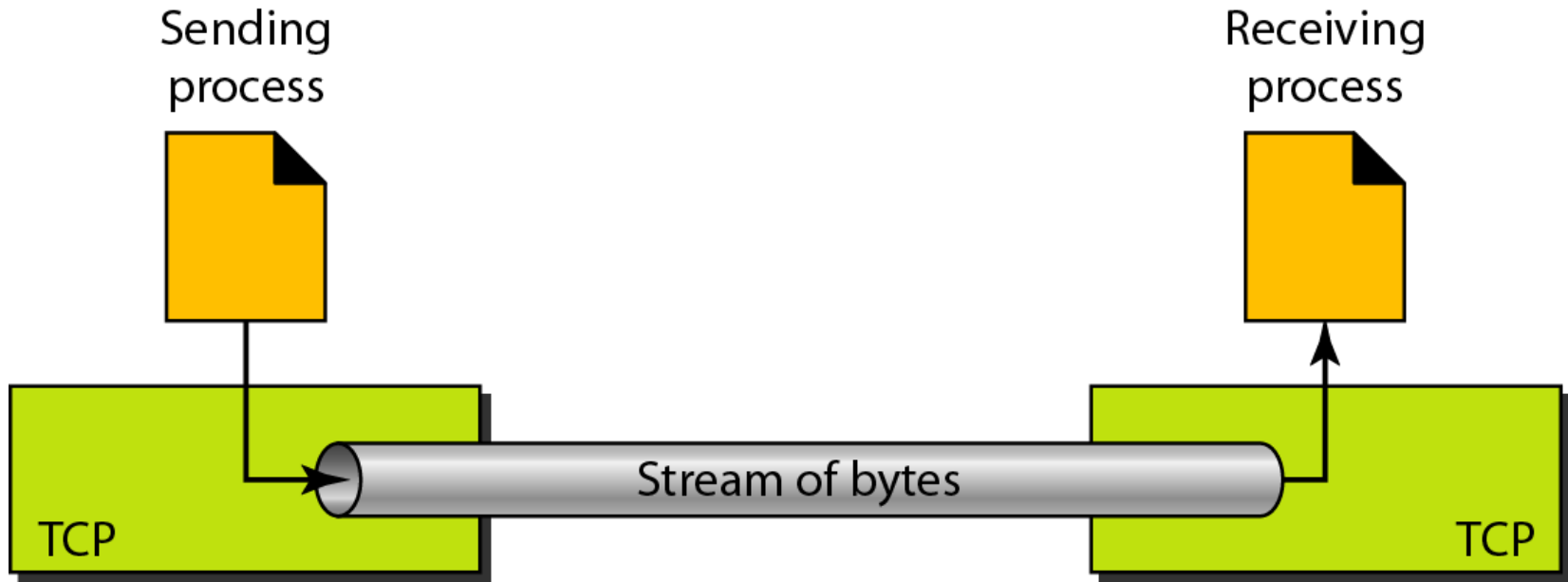


图 23.14 发送和接收缓冲区

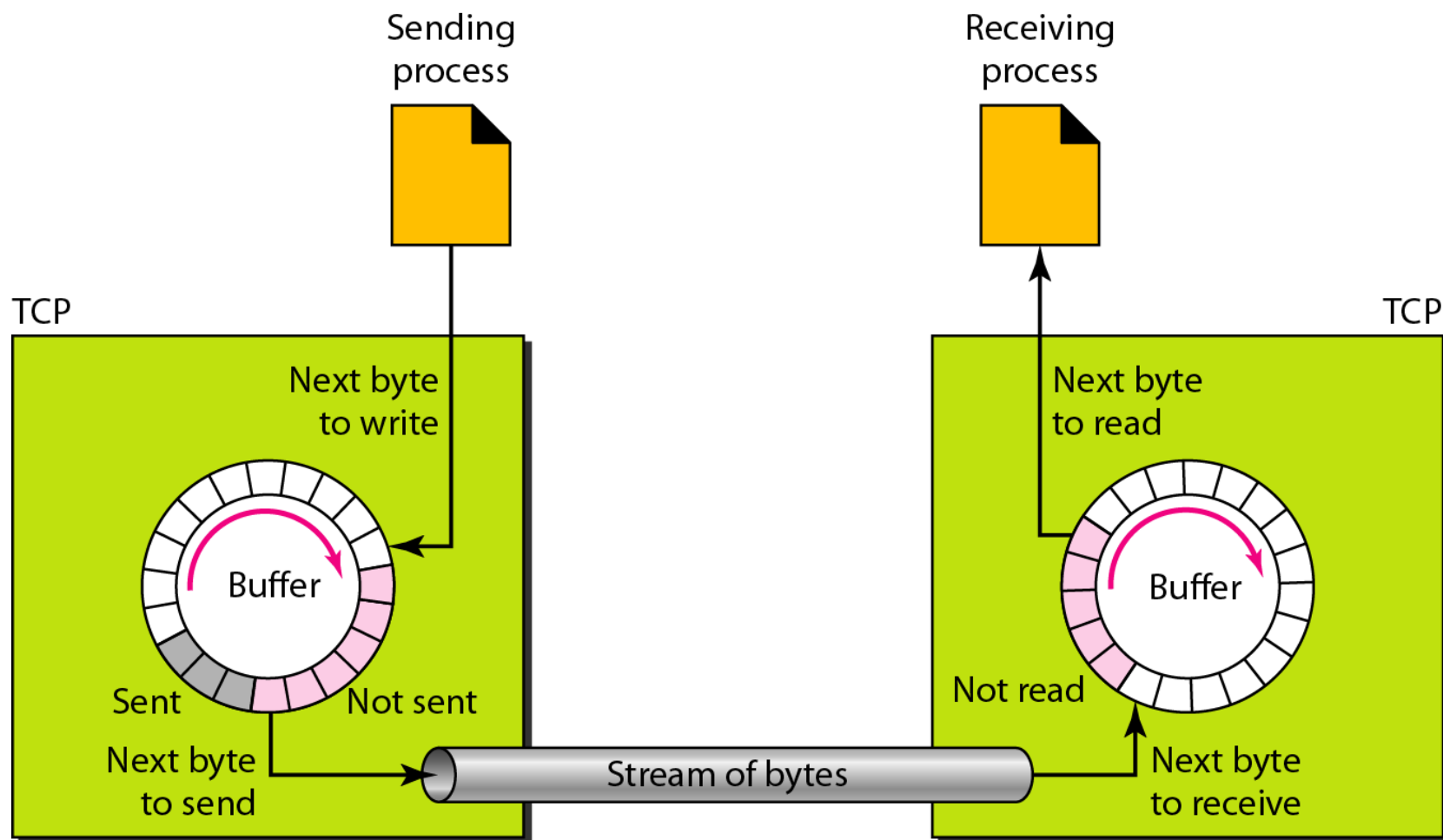
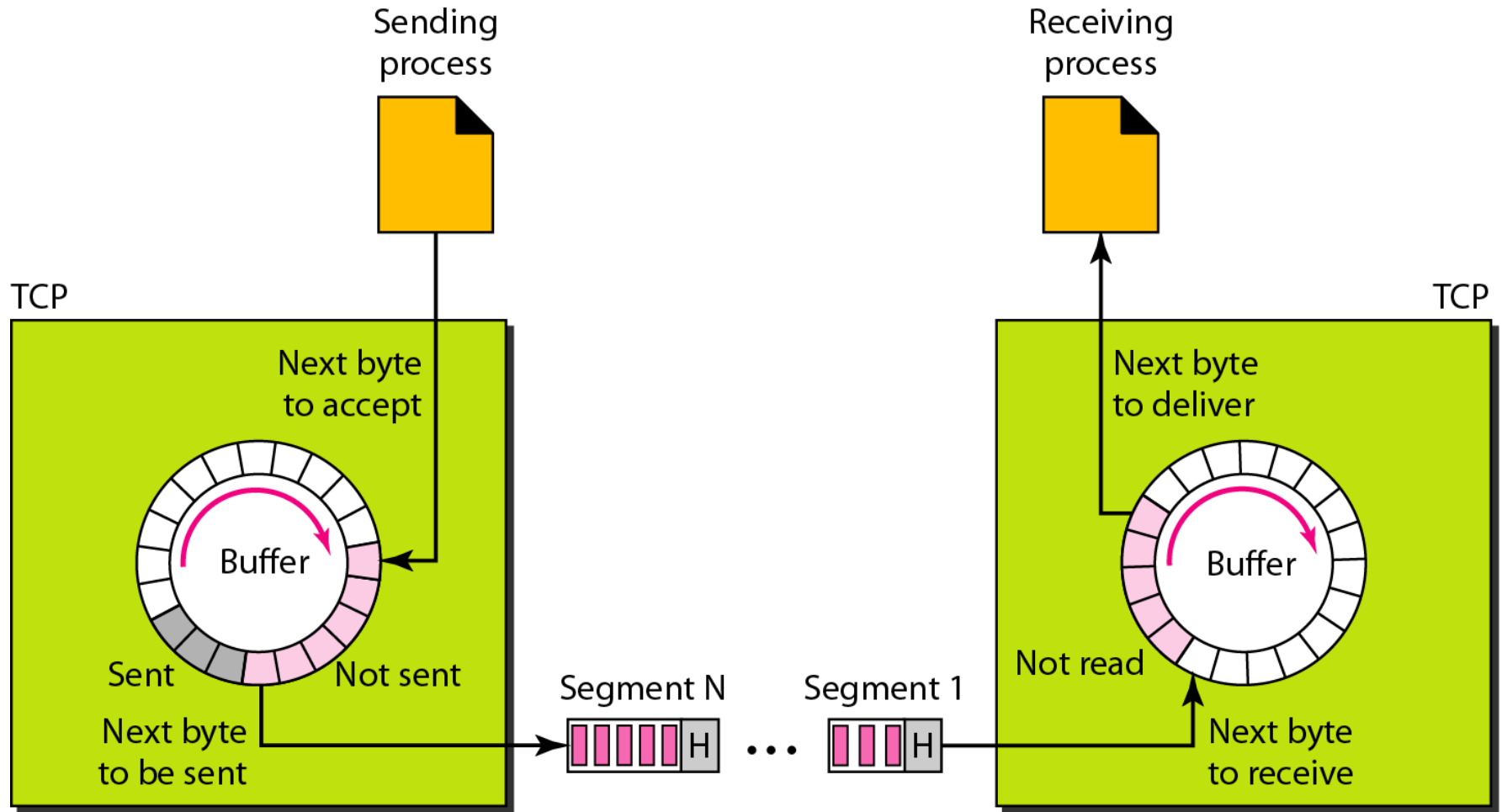
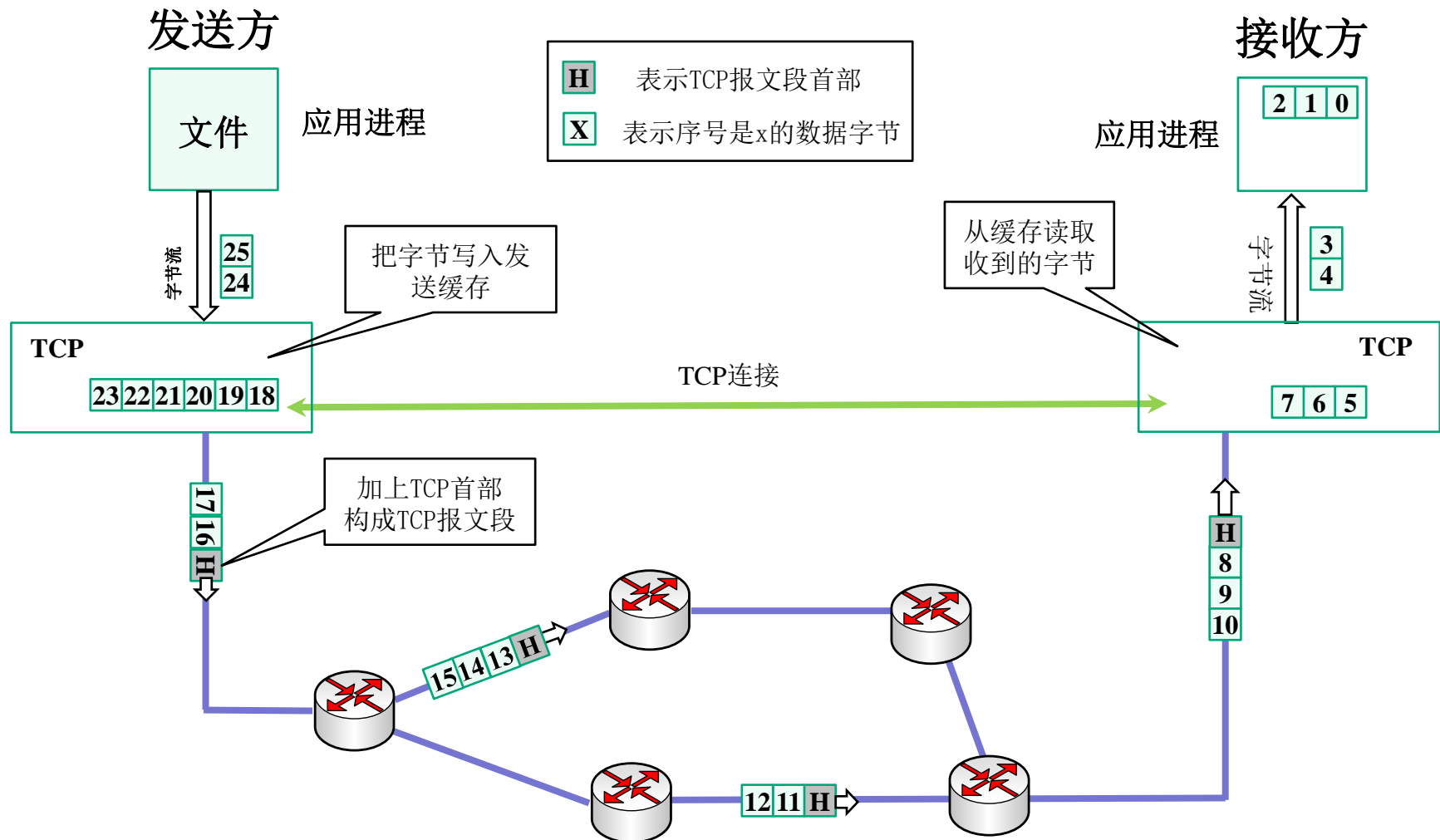


图 23.15 TCP段 (Segment)



TCP面向流的概念



序号系统

- TCP在段的首部使用序号和确认号，它们都是字节序号，而不是段序号。
- 在每个连接中传送的字节都由TCP编号，第一个字节的序号开始于0到 $2^{32}-1$ 之间的一个随机数。
- 字节被编号后，TCP对发送的每一个段分配一个序号，用这个段的第一个字节的序号表示。
- 确认号定义了通信一方预期接收的下一个字节的编号。确认号是累加的，表示接收方已正确地接收到最后一个字节的序号，加1并通告这个结果作为确认号。

例 23.3

如下展示了每一个段的序号，其中每个数据段携带1000字节：

Segment 1	➡	Sequence Number: 10,001 (range: 10,001 to 11,000)
Segment 2	➡	Sequence Number: 11,001 (range: 11,001 to 12,000)
Segment 3	➡	Sequence Number: 12,001 (range: 12,001 to 13,000)
Segment 4	➡	Sequence Number: 13,001 (range: 13,001 to 14,000)
Segment 5	➡	Sequence Number: 14,001 (range: 14,001 to 15,000)

序号和确认号

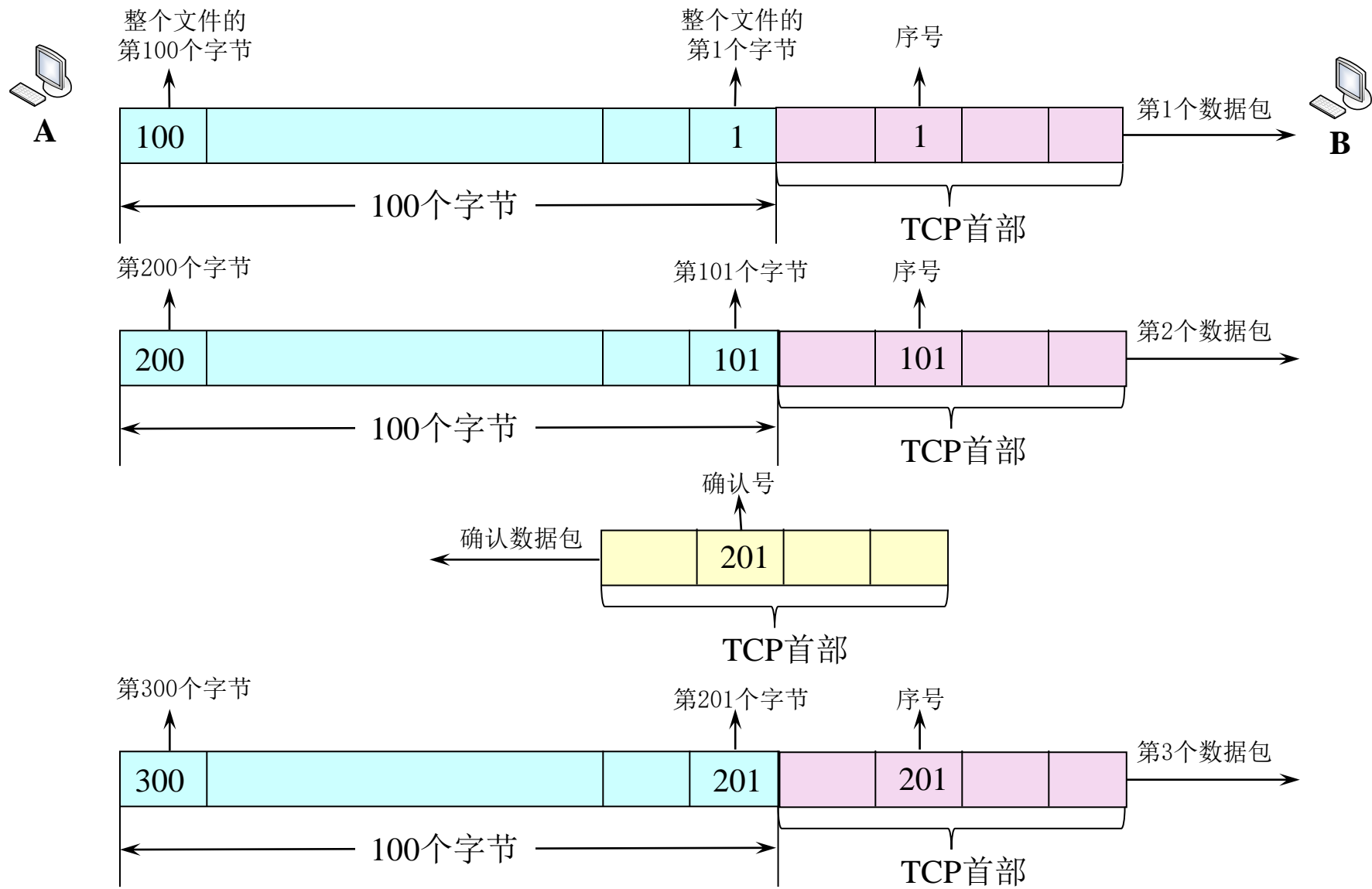
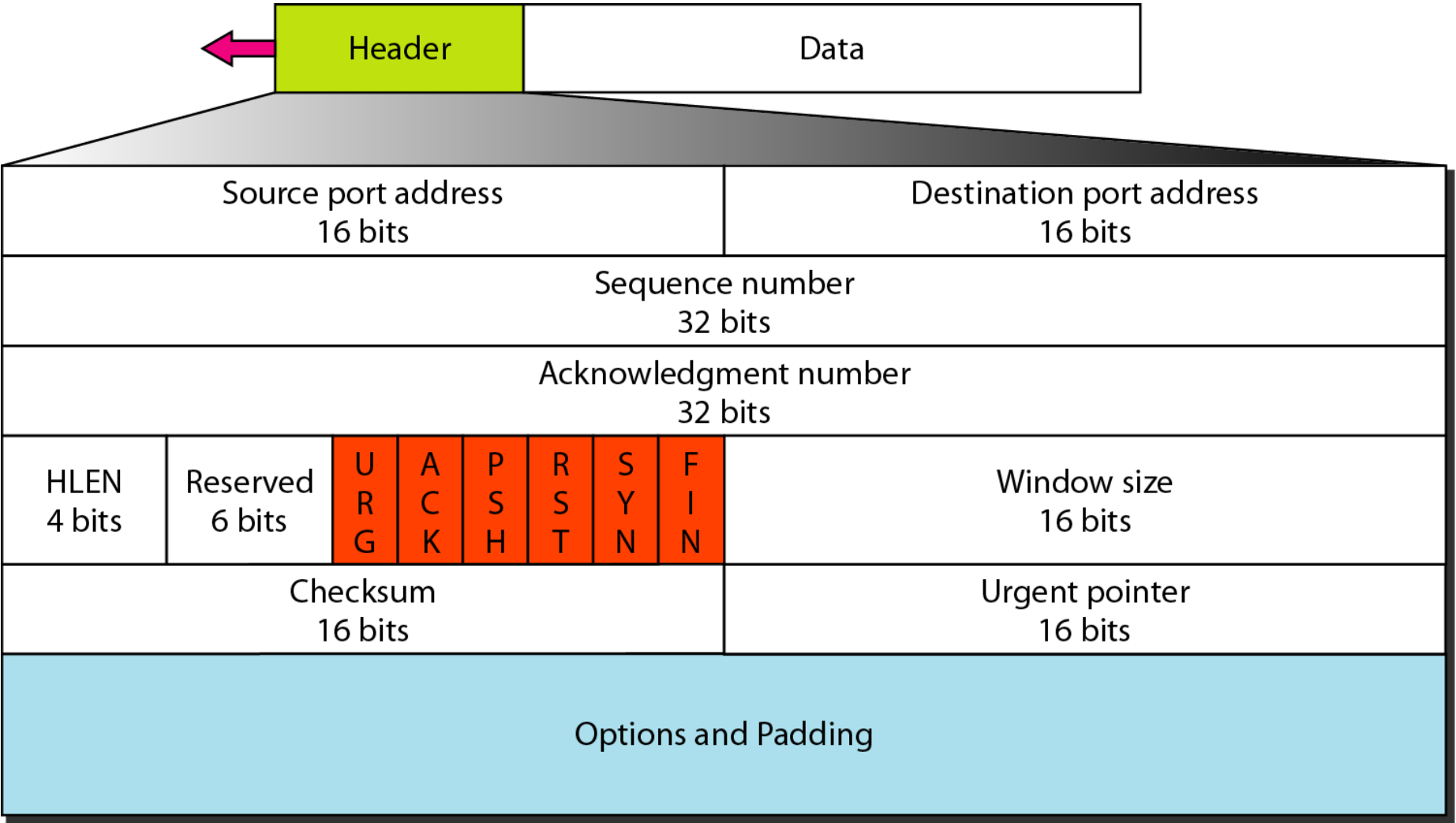


图 23.16 TCP段格式



说明

源端口和目的端口字段——各占 2 字节。端口是传输层与应用层的服务接口。传输层的复用和分用功能都要通过端口才能实现。

序号字段——占 4 字节。**TCP** 连接中传送的数据流中的每一个字节都编上一个序号。序号字段的值则指的是本报文段所发送的数据的第一个字节的序号。初始序号随机生成。

确认号字段——占 4 字节，是期望收到对方下一个报文段数据的第一个字节的序号。

首部长度的单位是 4 bit，其单位不是字节而是 4 字节，这个字段的值在 5（20 字节）到 15（60 字节）之间。

保留字段——占 6 bit，保留为今后使用，但目前应置为 0。

控制字段——占 6 bit，定义了 6 中不同的控制位或标记。

窗口字段——占 2 字节。窗口字段用来控制对方发送的数据量，单位为字节。

TCP 连接的一端根据设置的缓存空间大小确定自己的接收窗口大小，然后通知对方以确定对方的发送窗口的上限。

检验和——占 2 字节。检验和字段检验的范围包括首部和数据这两部分。在计算检验和时，要在 **TCP** 报文段的前面加上 12 字节的伪首部。

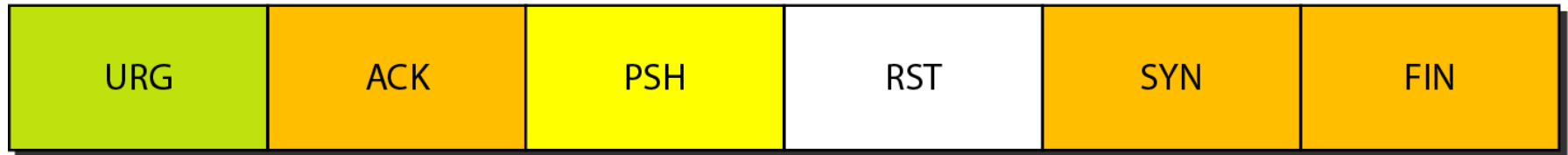
紧急指针字段——占 16 bit。紧急指针指出：在本报文段中紧急数据共有多少个字节（紧急数据放在本报文段数据的最前面）。

选项字段——长度可变。**TCP** 只规定了一种选项，即最大报文段长度 **MSS** (Maximum Segment Size)。MSS 是 **TCP** 报文段中的数据字段的最大长度。数据字段加上 **TCP** 首部才等于整个的 **TCP** 报文段。

图 23.17 控制字段

URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push

RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection



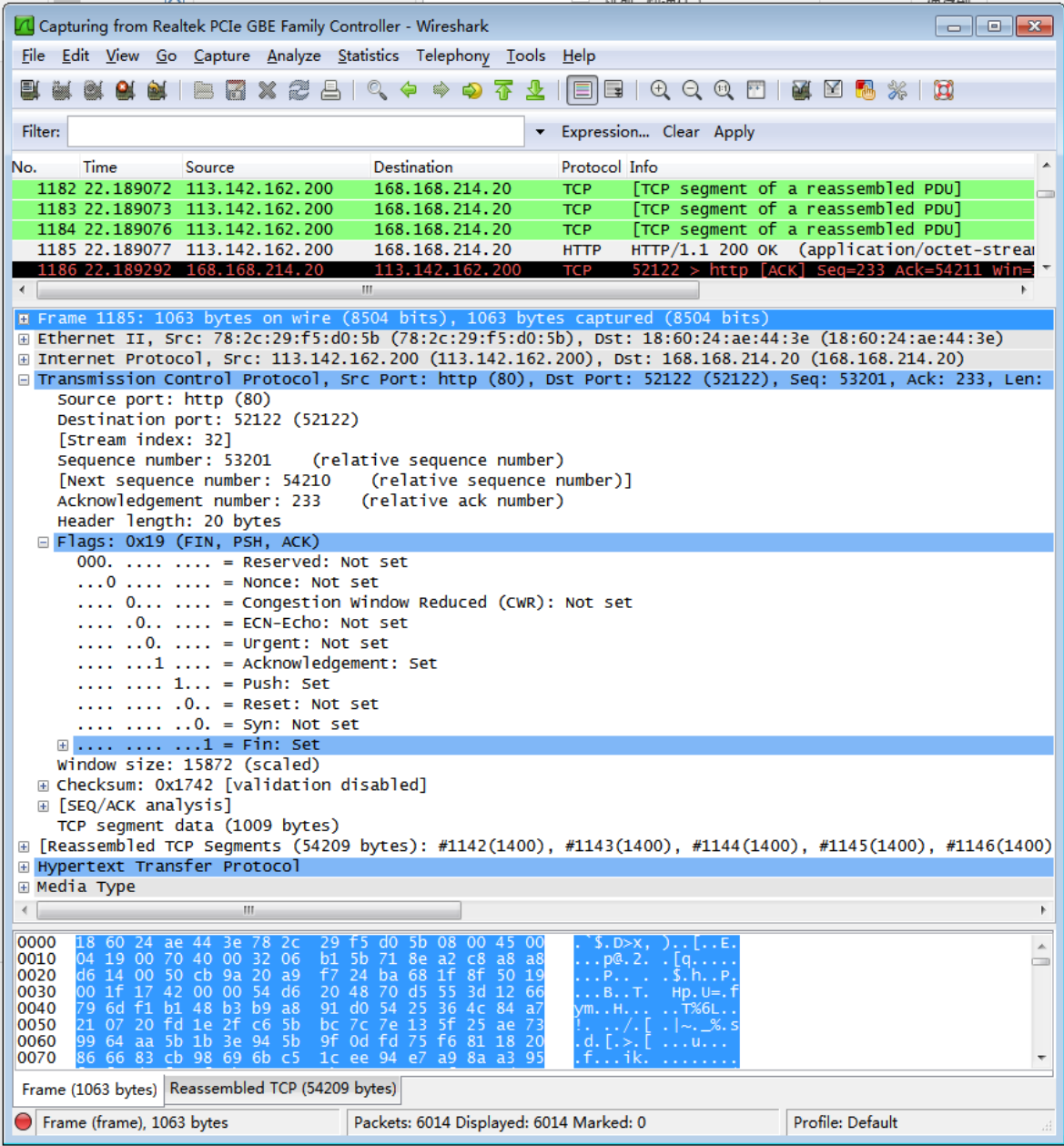
用于TCP的流量控制、连接建立和终止、连接失败和数据传送方式等方面。

表 23.3 控制字段的标志描述

<i>Flag</i>	<i>Description</i>
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

- **URG:** 紧急指针。置1时表示发送端应用程序告诉发送端TCP这块数据是紧急的，需要将紧急数据放在段的开始，接收端TCP利用紧急指针的值从段中取出紧急数据，并不按序地将其传递给接收应用。
- **PSH:** 请求急迫。发送端TCP不必等待窗口被填满，这个段包含的数据必须尽快地传递给接收应用程序，而不要等待更多数据的到来。

通过抓包查看TCP首部



UDP和TCP的区别

1、连接

- TCP是面向连接的协议，传输数据前先要建立连接。
- UDP 不需要连接，即刻传输数据。

2、服务对象

- TCP是一对一的两点服务，即一条连接只有两个端点。
- UDP支持一对一，一对多，多对多的交互通信。

3、可靠性

- TCP可靠交付数据，数据无差错、不丢失、不重复、按需到达。
- UDP 是尽最大努力交付，不保证数据交付的可靠性。

4、拥塞控制和流量控制

- TCP 有拥塞控制和流量控制机制，保证数据传输的安全性。
- UDP 没有这两种控制机制，即使网络非常拥堵，也不会影响 UDP 的发送速率。

UDP和TCP的区别

5、首部开销

- TCP 首部长长度较长（20-60字节），存在一定开销。
- UDP 首部只有8个字节，并且是固定不变的，开销较小。

6、传输方式

- TCP 是字节流传输，没有边界，但保证顺序和可靠。
- UDP是一个包一个包发送，有边界，但可能会丢包和乱序。

7、分片不同

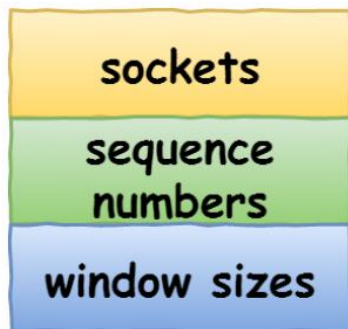
- TCP的数据大小如果大于 MSS 大小，则会在传输层进行分片，目标主机收到后，也同样在传输层组装TCP数据包，如果中途丢失了一个分片，只需要传输丢失的这个分片。
- UDP的数据大小如果大于 MTU 大小，则会在IP层进行分片，目标主机收到后，在 IP 层组装完数据，接着再传给传输层。但是如果中途丢了一个分片，在实现可靠传输的 UDP 时则就需要重传所有的数据包，这样传输效率非常差，所以通常 UDP 的报文应该小于 MTU。

网络层提供不可靠的服务

- 给传输层带来了很大的困难
 - 段可能会丢失
 - 段可能会失序
- 传输层需要解决的问题
 - 连接建立 & 终止
 - 有序交付
 - 副本检测
 - 重传策略
 - 故障恢复
 - 流量控制
 - 拥塞控制

TCP连接

- RFC 793 定义“连接”：The reliability and flow control mechanisms described above require that TCPs initialize and maintain certain status information for each data stream. The combination of this information, including sockets, sequence numbers, and window sizes, is called a connection.
- TCP是一种面向连接的协议，在源端和目的端之间建立一条虚路径。一个报文的所有段都沿着这条虚路径发送，整个报文使用单一的虚路径有利于确认处理及对损坏或丢失帧的重发。
- TCP使用IP服务向接收方传递独立的段，它对连接进行控制，如果段丢失或损失，则重新发送。如果一个段失序到达，TCP保存它直到缺少的段到达。IP不知道这个重发过程，也不知道重新排序的过程。



Socket: 由IP地址和端口号组成

序列号: 用来解决乱序问题

窗口大小: 用于流量控制

连接的三个阶段

- 面向连接的传输有三个阶段：**连接建立**、**数据传输**和**连接终止**。连接的管理就是使连接的建立和释放都能正常地进行。
- 连接建立过程中要解决三个问题：
 - 要使每一方能够确知对方的存在；
 - 要允许双方协商一些参数（如最大报文段长度，最大窗口大小，服务质量等）；
 - 能够对传输实体资源（如缓存大小，连接表中的项目等）进行分配。

著名的协议举例

- 占据两个山顶的蓝军1和蓝军2与驻扎在山谷的白军作战。单独的蓝军打不过白军，只有协同作战才可战胜白军。
- 蓝军1拟于次日正午向白军发起进攻，于是用计算机发送电文给蓝军2。但通信线路很不好，电文出错或丢失的可能性很大。因此要求收到电文的友军必须送回一个确认电文，但此确认电文也可能出错或丢失。
- 能否设计出一种协议使蓝军1和蓝军2能够实现协同作战因而一定（100%）取得胜利？

明日正午进攻？

同意

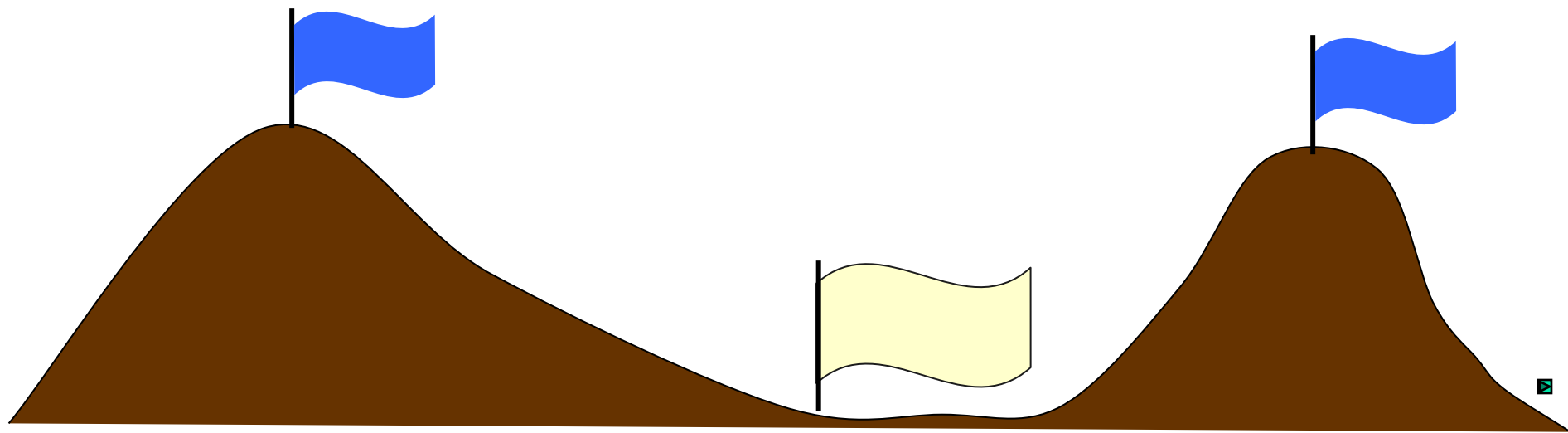
收到“同意”

蓝军联合进攻白军——必胜
蓝军单独进攻白军——必败

收到：收到“同意”

协议无法保证必胜！

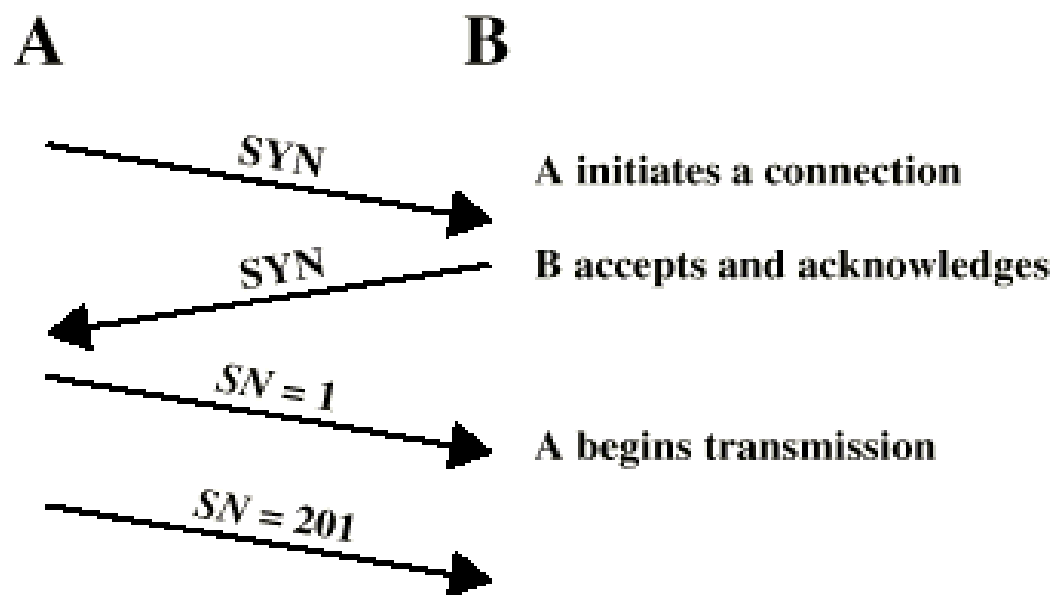
...



结论

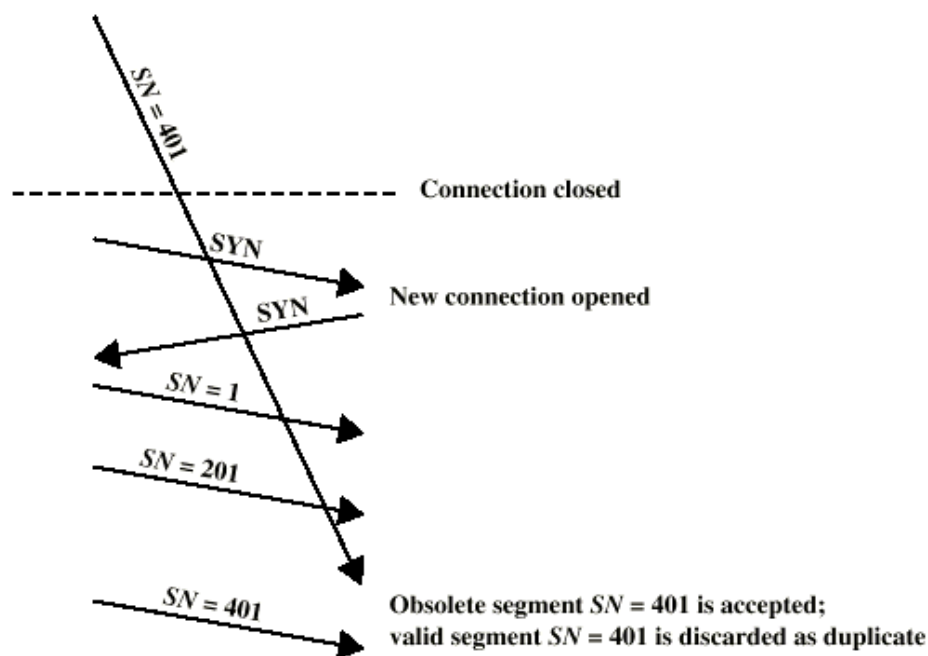
- 这样无限循环下去，两边的蓝军都始终无法确认自己最后发出的电文对方是否已经收到。
- 没有一种协议能够确定蓝军100%获胜。
- 协议必须事先把所有不利的条件都估计到。

二次握手

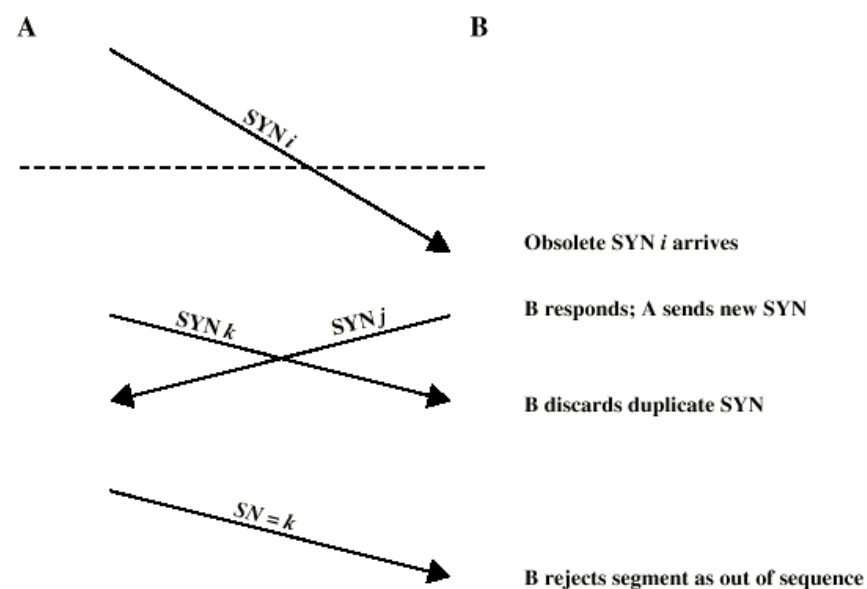


二次握手存在的问题

迟来的报文段



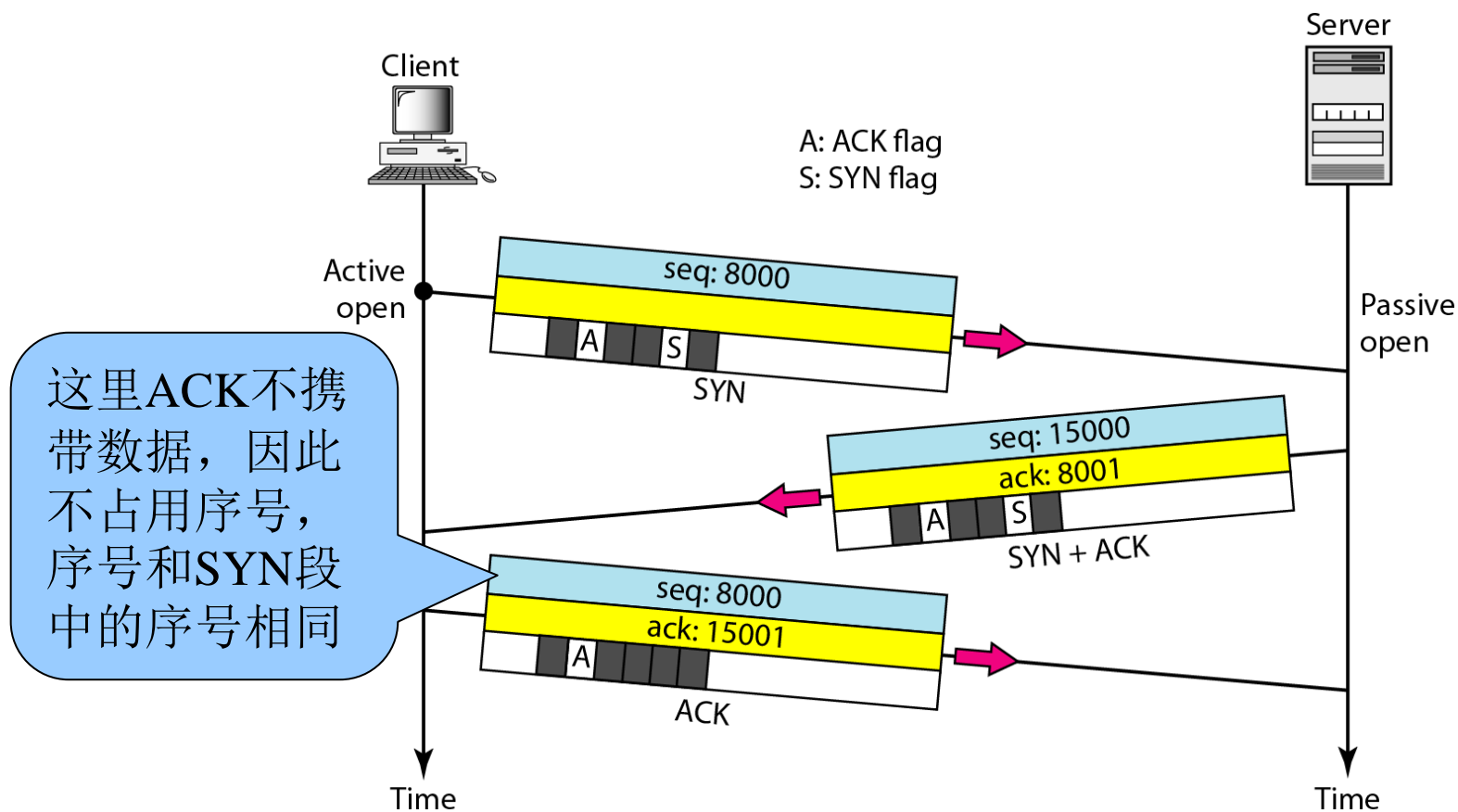
迟来的SYN



三次握手

- 被动打开：服务器程序告诉TCP，已准备好接受一个连接。它自己不能完成这个连接。
- 主动打开：客户程序发出请求，告诉TCP它需要连接到特定服务器。

图 23.18 使用三次握手建立连接



谢希仁第五版P216页上说：TCP标准规定，ACK报文可以携带数据，因此第三次Seq=8000+1

请求建立TCP连接的数据包

请求连接

确认连接

确认的确认

序号 seq = 0

ACK = 0

SYN = 1

MSS = 1460

选项部分

这是客户端发给服务器的第一个数据包所以seq = 0

请求连接的特征

Request connection features

Request connection

Acknowledge connection

Acknowledge the acknowledgment

Sequence number = 0

ACK = 0

SYN = 1

MSS = 1460

Options part

This is the first data packet sent by the client to the server, so seq = 0

Request connection features

Request connection

Acknowledge connection

Acknowledge the acknowledgment

Sequence number = 0

ACK = 0

SYN = 1

MSS = 1460

Options part

This is the first data packet sent by the client to the server, so seq = 0

Request connection features

Request connection

Acknowledge connection

Acknowledge the acknowledgment

Sequence number = 0

ACK = 0

SYN = 1

MSS = 1460

Options part

This is the first data packet sent by the client to the server, so seq = 0

请求建立TCP连接的数据包

请求连接

确认连接

确认的确认

序号 seq = 0

确认号 ack = 1

ACK = 1

SYN = 1

服务器支持的
MSS = 1460

这是服务器发给客户端的第一个数据包所以seq = 0

服务器收到连接请求seq = 0
后发送确认号为0+1

选项部分

请求建立TCP连接的数据包

Wireshark packet capture analysis showing a TCP connection establishment sequence.

Packet List:

No.	Time	Source	Destination	Protocol	Info
1	0.000000	168.168.214.20	202.117.112.26	TCP	64810 > https [SYN] Seq=0 Win=8192 Len=0 MSS=1460
2	0.000588	168.168.214.20	202.117.112.26	TCP	64811 > https [SYN] Seq=0 Win=8192 Len=0 MSS=1460
3	0.000671	202.117.112.26	168.168.214.20	TCP	https > 64810 [SYN, ACK] Seq=0 Ack=1 Win=2105 Len=0
4	0.000762	168.168.214.20	202.117.112.26	TCP	64810 > https [ACK] Seq=1 Ack=1 Win=66608 Len=0
5	0.001163	168.168.214.20	202.117.112.26	TLSv1.2	Client Hello
6	0.001407	202.117.112.26	168.168.214.20	TCP	https > 64811 [SYN, ACK] Seq=0 Ack=1 Win=2105 Len=0
7	0.001483	168.168.214.20	202.117.112.26	TCP	64811 > https [ACK] Seq=1 Ack=1 Win=66608 Len=0
8	0.001795	168.168.214.20	202.117.112.26	TLSv1.2	Client Hello
9	0.004855	202.117.112.26	168.168.214.20	TLSv1.2	Server Hello
10	0.005556	202.117.112.26	168.168.214.20	TCP	[TCP segment of a reassembled PDU]
11	0.005559	202.117.112.26	168.168.214.20	TCP	[TCP segment of a reassembled PDU]
12	0.005561	202.117.112.26	168.168.214.20	TCP	[TCP segment of a reassembled PDU]
13	0.005562	202.117.112.26	168.168.214.20	TLSv1.2	Certificate, Server Key Exchange, Server Hello Done
14	0.005563	202.117.112.26	168.168.214.20	TLSv1.2	Server Hello
15	0.005565	202.117.112.26	168.168.214.20	TCP	[TCP segment of a reassembled PDU]
16	0.005582	168.168.214.20	202.117.112.26	TCP	64810 > https [ACK] Seq=1 Ack=5002 Win=66608 Len=0

Packet 4 Details:

- Frame 4: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
- Ethernet II, Src: 18:60:24:ae:44:3e (18:60:24:ae:44:3e), Dst: 78:2c:29:f5:d0:5b (78:2c:29:f5:d0:5b)
- Internet Protocol, Src: 168.168.214.20 (168.168.214.20), Dst: 202.117.112.26 (202.117.112.26)
- Transmission Control Protocol, Src Port: 64810 (64810), Dst Port: https (443), Seq: 1, Ack: 1, Len: 0
 - Source port: 64810 (64810)
 - Destination port: https (443)
 - [Stream index: 0]
 - Sequence number: 1 (relative sequence number)
 - Acknowledgement number: 1 (relative ack number)
 - Header length: 20 bytes
 - Flags: 0x10 (ACK)
 - 000. = Reserved: Not set
 - ...0. = Nonce: Not set
 - 0... = Congestion Window Reduced (CWR): Not set
 -0.. = ECN-Echo: Not set
 -0. = Urgent: Not set
 -1 = Acknowledgement: Set
 - 0... = Push: Not set
 -0. = Reset: Not set
 -0. = Syn: Not set
 - 0 = Fin: Not set
 - window size: 66608 (scaled)
 - Checksum: 0xb967 [validation disabled]
 - [SEQ/ACK analysis]
 - [This is an ACK to the segment in frame: 3]
 - [The RTT to ACK the segment was: 0.000091000 seconds]

Annotations:

- 请求连接 (Request connection) points to packet 1.
- 确认连接 (Acknowledge connection) points to packet 3.
- 确认的确认 (Acknowledge the acknowledgment) points to packet 4.
- 序号 seq = 1 (Sequence number seq = 1) points to the Seq: 1 field in packet 4 details.
- 确认号 ack = 1 (Acknowledgment number ack = 1) points to the Ack: 1 field in packet 4 details.
- ACK = 1 (Circled text) points to the Acknowledgement: Set flag in packet 4 details.
- 这是客户机给确认的一个确认seq = 1 (This is an acknowledgment from the client with seq = 1) points to the Seq: 1 field in packet 4 details.
- 客户机收到确认连接seq = 0后发送确认号为0+1 (The client sends an acknowledgment number of 0+1 after receiving the acknowledgment connection seq = 0) points to the Acknowledgement number: 1 field in packet 4 details.

Packet 4 Hex Data:

```
0000 78 2c 29 f5 d0 5b 18 60 24 ae 44 3e 08 00 45 00 x.)...$.D>..E.
0010 00 28 1a 50 40 00 40 06 00 00 a8 a8 d6 14 ca 75 .(.P@.@.....u
0020 70 1a fd 2a 01 bb b6 e7 54 c3 49 78 ba 65 50 10 p..*....T.Ix.eP.
0030 41 0c b9 67 00 00 A..q..
```



注意

- SYN段不携带数据，但它占用一个序列号。
- SYN + ACK 段不携带数据，但它占用一个序列号。
- ACK段如果不携带数据，则它不占用序列号。

SYN洪泛攻击

- TCP连接建立过程容易遭受SYN洪泛攻击，属于拒绝服务攻击（Denial of Service Attack）。恶意攻击者将大量SYN段发送到一个服务器，导致服务器资源耗尽。
- 应对策略：特定时间内限制连接请求；过滤来自不需要源地址的数据报；使用cookie推迟资源分配直到建立一个完整的连接。

图 23.19 数据传输

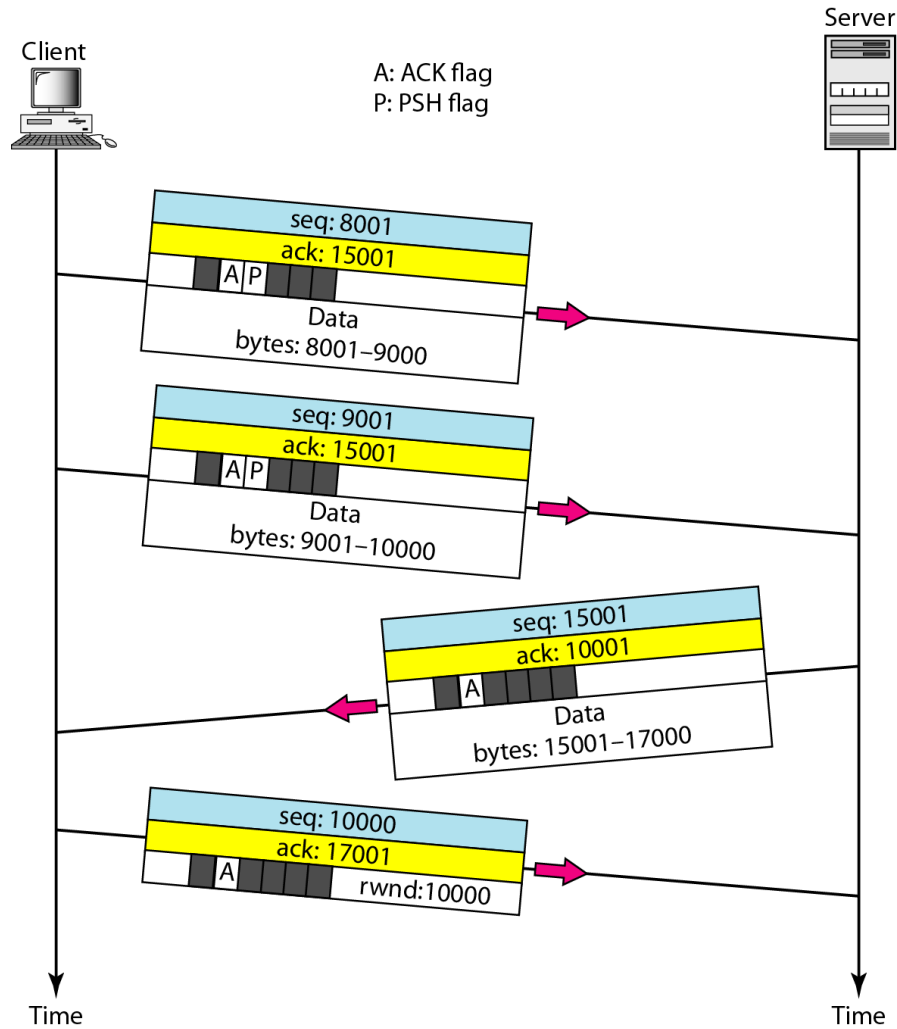
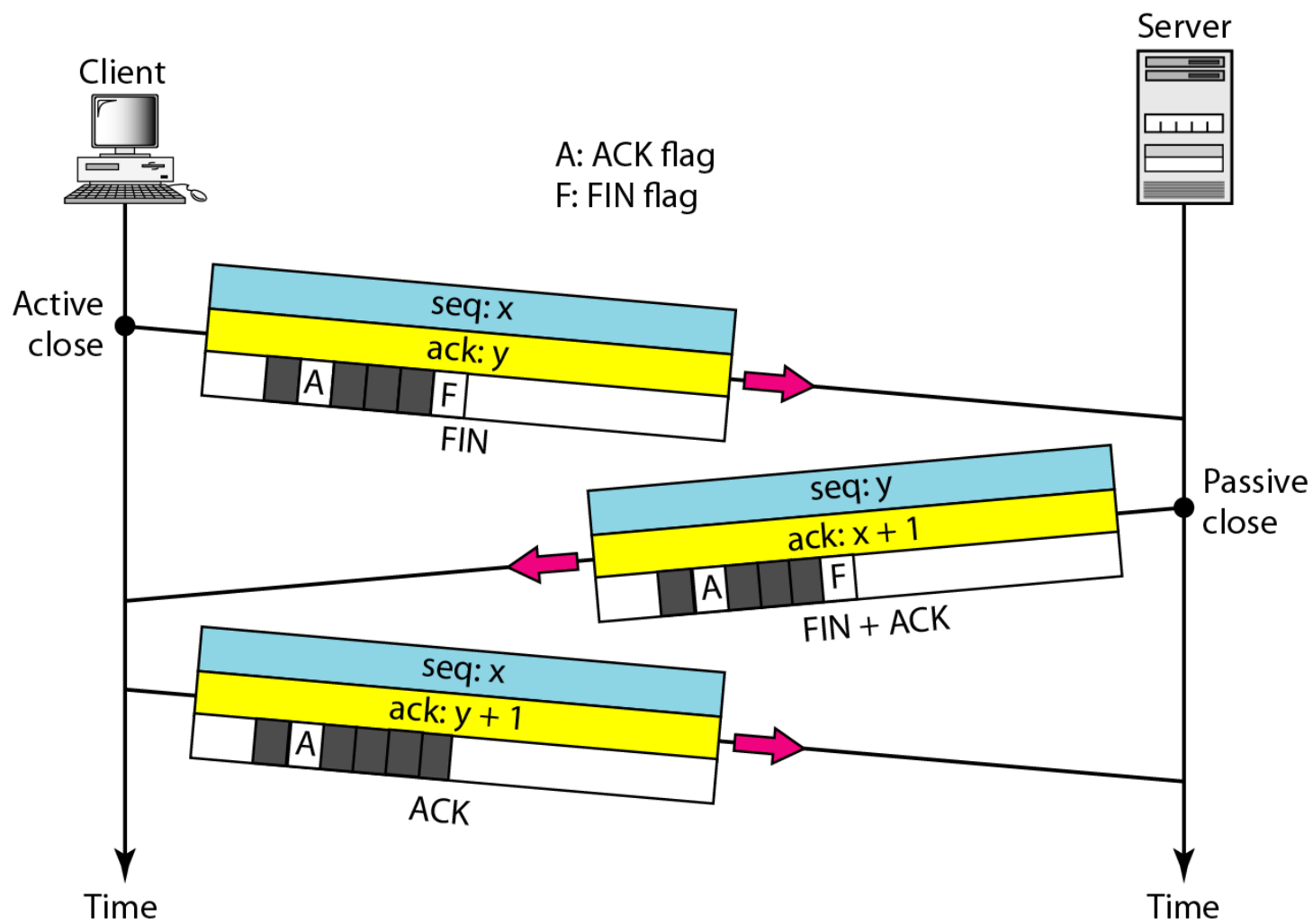


图 23.20 使用三次握手终止连接

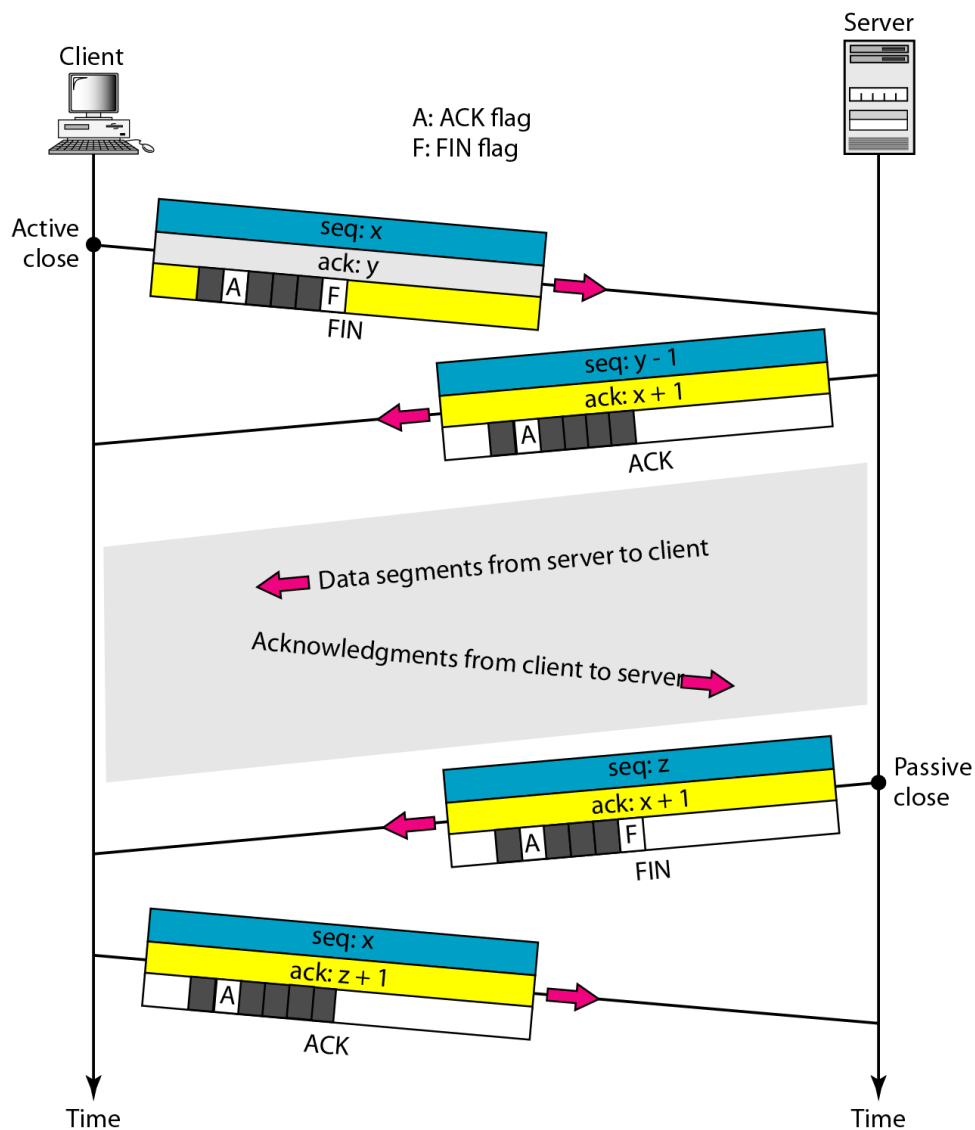




注意

- 如果FIN段不携带数据，则该段占用一个序列号。
- 如果 $\text{FIN} + \text{ACK}$ 段没有携带数据，则该段仅占用一个序列号。

图 23.21 带有半关闭的四次握手



查看TCP连接的状态

```
管理员: 命令提示符

C:\Users\Administrator>netstat -n

活动连接

协议 本地地址          外部地址          状态
TCP    127.0.0.1:16804    127.0.0.1:62740    ESTABLISHED
TCP    127.0.0.1:62740    127.0.0.1:16804    ESTABLISHED
TCP    168.168.214.20:49167 117.34.40.244:80    CLOSE_WAIT
TCP    168.168.214.20:49168 117.34.40.244:80    CLOSE_WAIT
TCP    168.168.214.20:49198 101.199.128.185:80   ESTABLISHED
TCP    168.168.214.20:49212 101.199.128.196:80   ESTABLISHED
TCP    168.168.214.20:49271 180.163.238.163:443  ESTABLISHED
TCP    168.168.214.20:62635 180.163.249.3:80     ESTABLISHED
TCP    168.168.214.20:65124 140.210.88.44:443    ESTABLISHED
TCP    168.168.214.20:65125 27.19.253.1:443      ESTABLISHED
TCP    168.168.214.20:65126 27.19.253.1:443      ESTABLISHED
TCP    168.168.214.20:65127 27.19.253.1:443      ESTABLISHED
TCP    168.168.214.20:65128 27.19.253.1:443      ESTABLISHED
TCP    168.168.214.20:65129 27.19.253.1:443      ESTABLISHED
TCP    168.168.214.20:65130 27.19.253.1:443      ESTABLISHED
TCP    168.168.214.20:65131 140.210.88.44:443    ESTABLISHED
TCP    168.168.214.20:65132 172.217.160.106:443  SYN_SENT
TCP    168.168.214.20:65133 172.217.160.106:443  SYN_SENT
TCP    168.168.214.20:65134 202.117.112.26:443   ESTABLISHED
TCP    168.168.214.20:65135 202.117.112.26:443   ESTABLISHED
TCP    168.168.214.20:65136 202.117.112.26:443   ESTABLISHED
TCP    168.168.214.20:65137 202.117.112.26:443   ESTABLISHED
TCP    168.168.214.20:65138 202.117.112.26:443   ESTABLISHED
TCP    168.168.214.20:65139 202.117.112.26:443   ESTABLISHED
```

关闭等待

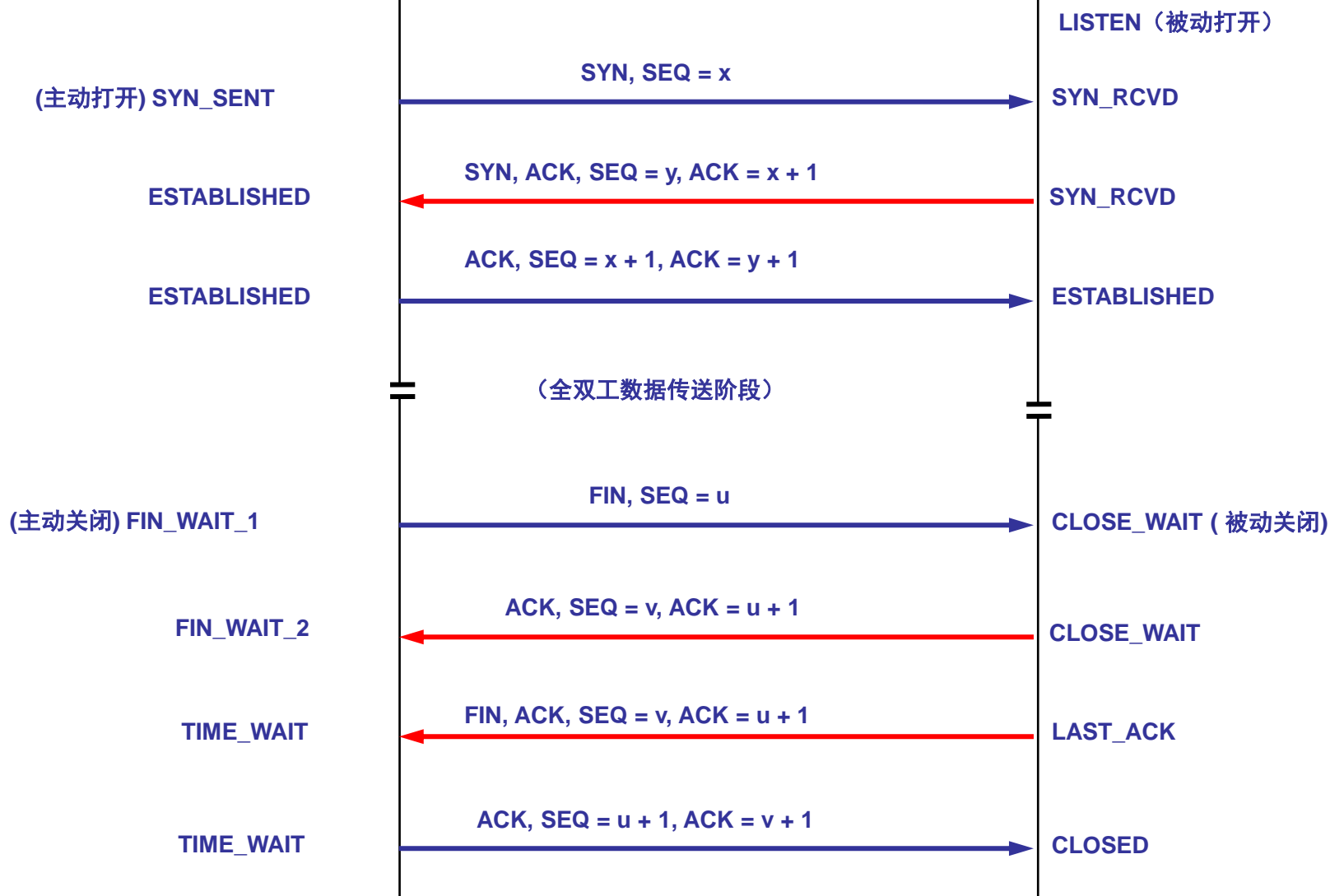
已建立连接

发送连接请求

TCP 的正常的连接建立和关闭

客户进程

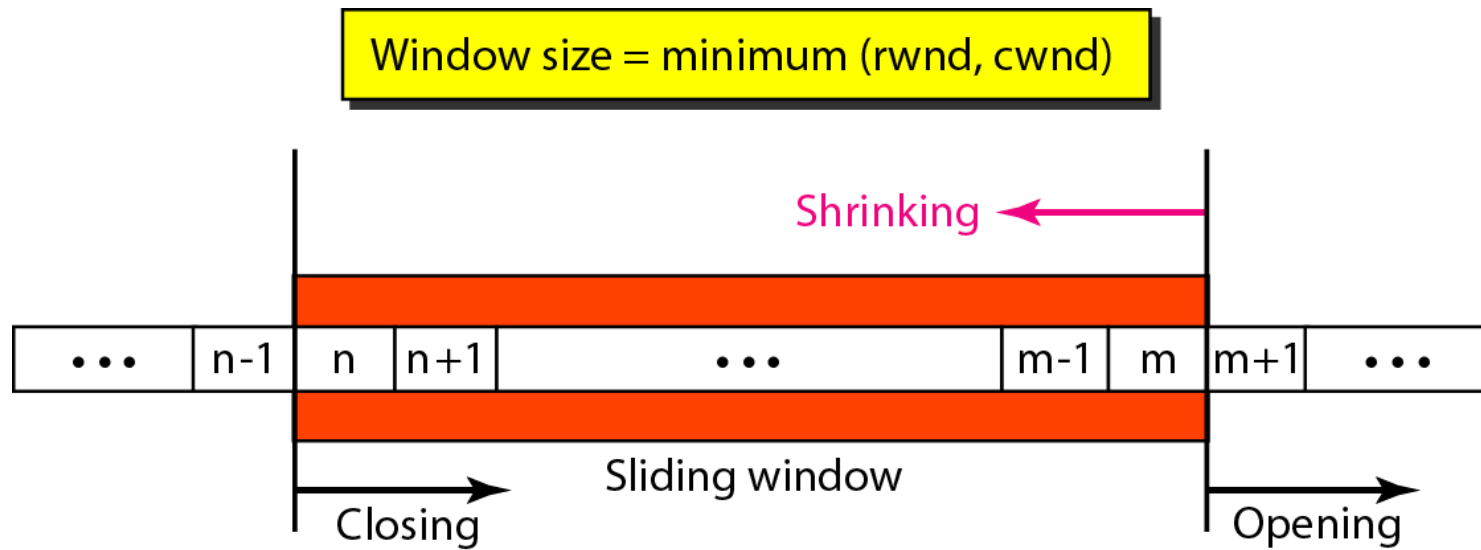
服务器进程



流量控制

- 人们总是希望数据传输得更快一些。但如果发送方把数据发送得过快，接收方就可能来不及接收，这就会造成数据的丢失。
- 流量控制就是让发送方的发送速率不要太快，既要让接收方来得及接收，也不要使网络发生拥塞。
- 利用TCP的滑动窗口机制可以很方便地在TCP连接上实现流量控制。

图 23.22 滑动窗口



传输层的流量控制

数据链路层的流量控制是一种比较简单的机制，到了传输层则变得复杂。

- 与实际的传输时间相比，传输实体之间的传输时延都比较长，即流量控制信息的通信存在着相当可观的时延。
- 传输层是在网络或互联网之上操作的，传输时延是高度可变的，这使得为重传丢失数据而使用的超时机制难以高效。

TCP的滑动窗口

- TCP的滑动窗口是面向字节的，数据链路层的滑动窗口是面向帧的；
- TCP的滑动窗口是可变大小的，而数据链路层的滑动窗口是固定大小；
- 窗口的滑动是由接收方而不是发送方控制的，发送方必须服从接收方的命令；
- TCP窗口受接收方窗口（`rwnd`）和拥塞窗口（`cwnd`）大小的影响，取两者中的最小值；
- 发送方不必发送一个全窗口大小的数据；
- 接收方可暂时关闭窗口，之后发送方可发送一个1字节的探测报文。

例 23.4

如果接收方主机B有一个5000字节的缓冲区，已接收但并未处理1000个字节数据。试问主机A的接收方窗口（**rwnd**）的值是多少？

解：

rwnd的值 = $5000 - 1000 = 4000$ ，主机B在它的缓冲溢出之前仅能接收4000个字节数据。主机B在下一段向主机A宣布这个值。



例 23.5

如果rwnd的值是3000个字节， cwnd的值是3500个字节， 试问主机A的窗口大小是多少？

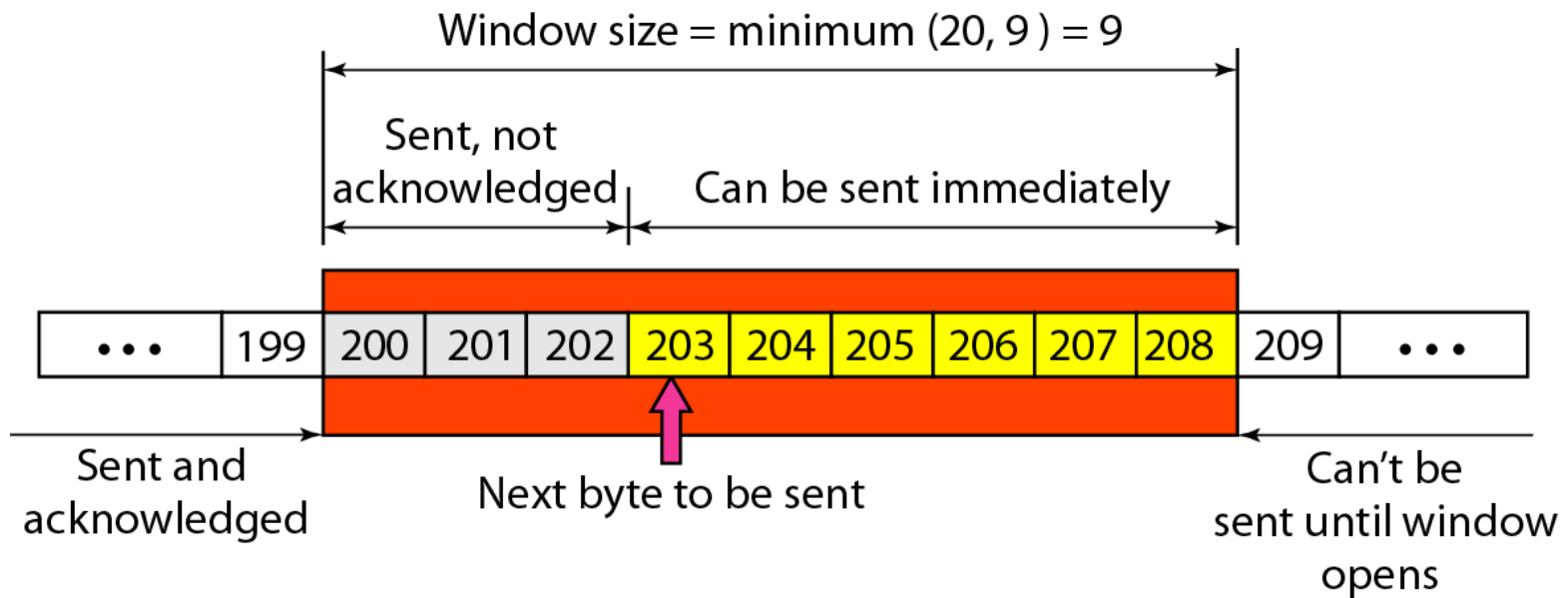
解：

窗口大小是rwnd和cwnd中较小的一个， 它应该是3000个字节。

例 23.6

图 23.23 表示了一个不切实际的滑动窗口，发送方已发送了202个字节。假定cwnd是20（实际上这个值是上千个字节），接收方已经发送一个rwnd为 9（实际上这个值也是上千个字节）而确认号为200的ACK段。发送窗口的大小是rwnd和cwnd的最小值，即9个字节。字节号200到202都已发送但并未被确认。字节号203到208可以发送而不必考虑从另一端来的确认。字节号209和它以上各字节不可能被发送。

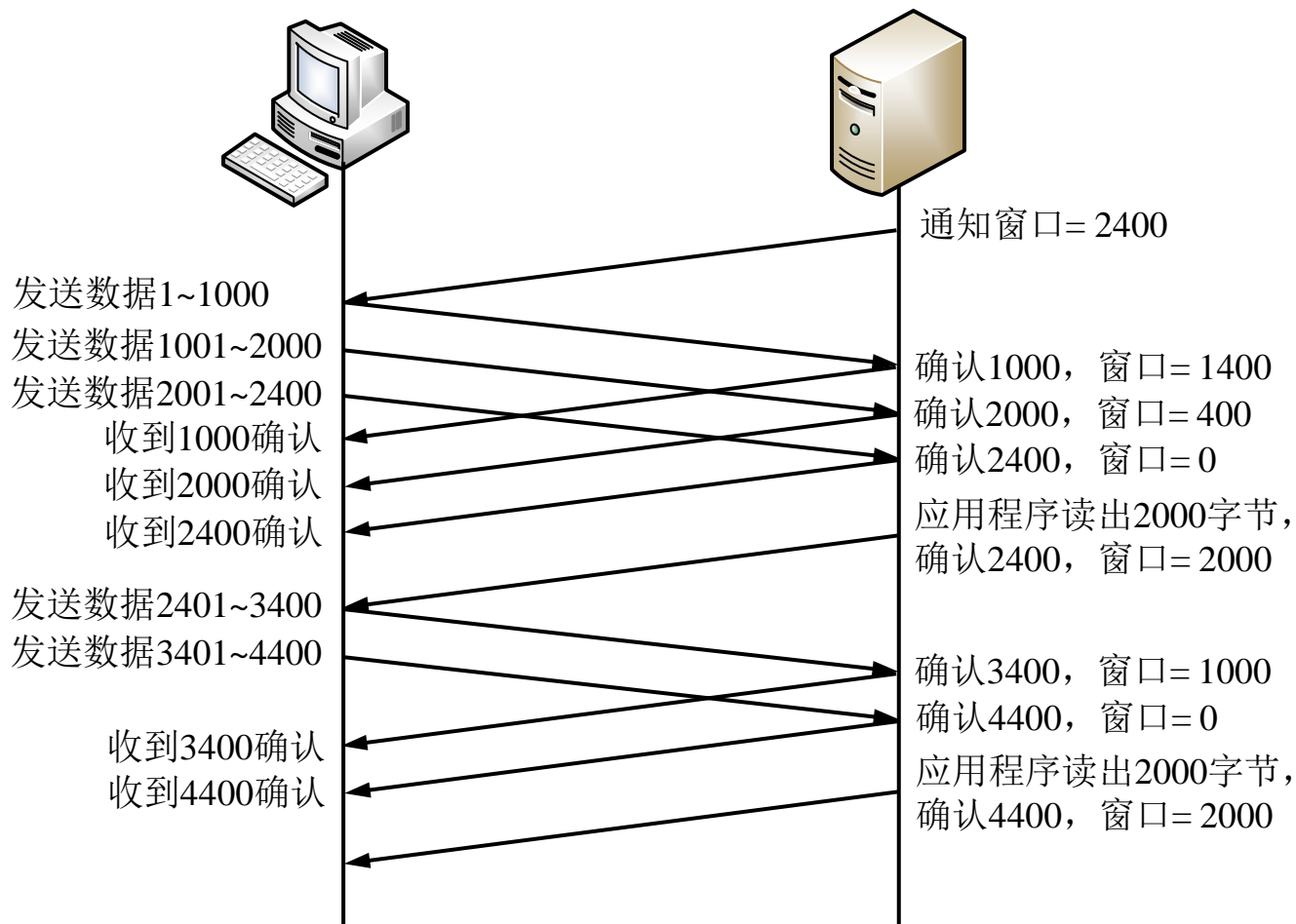
图 23.23 例 23.6



TCP通知窗口

- 通知窗口是接收方根据接收能力确定的窗口值。接收方将通知窗口值放在报文首部中发送给发送方。根据接收方工作状态来改变窗口大小是TCP流量控制的主要方法。
- 如果接收方读取数据的速度与数据到达的速度一样，接收方在每个确认中发送一个非零的窗口通告。如果发送方比接收方速度快，造成缓冲区被全部占用，接收方发送一个“零窗口”通告，发送方停止发送。

TCP利用窗口进行流量控制



坚持计时器

- 如果接收方通告窗口为零，发送方需要停止发送，直至接收方确认并通告一个非零窗口。但是这个报文可能会丢失，造成双方都在等待对方，出现死锁。
- TCP为每个连接设置一个坚持计时器，当发送方收到零窗口的确认时，就启动坚持计时器。计时器到期后，发送方会发送一个1字节的探测报文，提醒接收方确认报文已丢失。

糊涂窗口综合征（Silly Windows Syndrome）

- 如果TCP缓存已满，而应用进程每次只从缓存读取1字节，则接收缓存空出1字节，接收方向发送方发送窗口为1的确认报文。
- 此时发送方会以41字节的代价发送1字节的数据（假设TCP和IP首部各20字节），这样继续下去，传输效率极低。
- 解决办法是禁止接收方发送只有1字节的更新报文，接收方会等待一段时间，使得接收缓存具有足够空间接收一个较长的报文段后再发送窗口更新报文。

差错控制

- TCP是可靠的传输层协议，使用差错控制来提供可靠性；
- 差错控制包括检测出报文的差错、丢失、失序和重复并纠正；
- 差错检测和纠正有三种方式：校验和、确认和超时。

差错检测和纠正的三种方式

- 校验和：16位，用来检查受到损坏的段。如果损坏则会被丢弃。
- 确认：数据段需确认，不携带数据但占用序号的控制段也要确认，ACK段不需要确认。
- 重传：差错控制的核心。当一个段损坏、丢失或延迟时需要重传。不占用序号的段不重传，尤其是ACK不重传。

重传的两种情况

- RTO后重传：段或ACK延迟或丢失。根据段的往返时间RTT（Round-Trip Time）设置重传超时RTO（Retransmission Time Out）计时器的时间。
- 收到三个重复ACK：快速重传。在重传计时器超时之前进行重传。

失序控制

- 当一个段被延迟、丢失或废弃时，后面一些段的到达就失序了。
- 数据可以失序到达，TCP暂时存储失序的段，并标记它们为失序的段直到缺少的段到达。
- 失序的段不传递给进程，TCP确保传递给进程的段是无失序的。

TCP操作 I

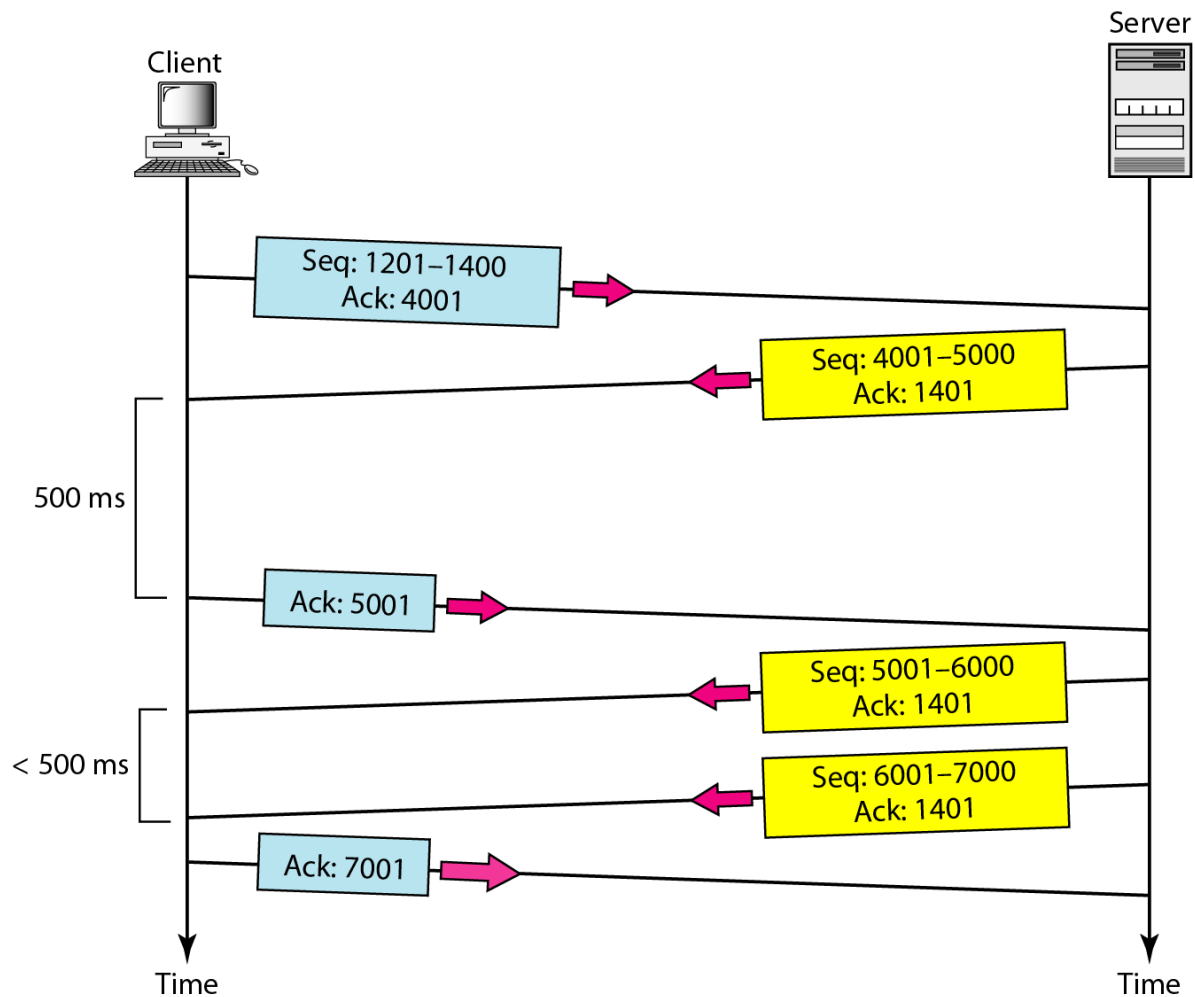


图 23.24 正常操作

TCP操作 II

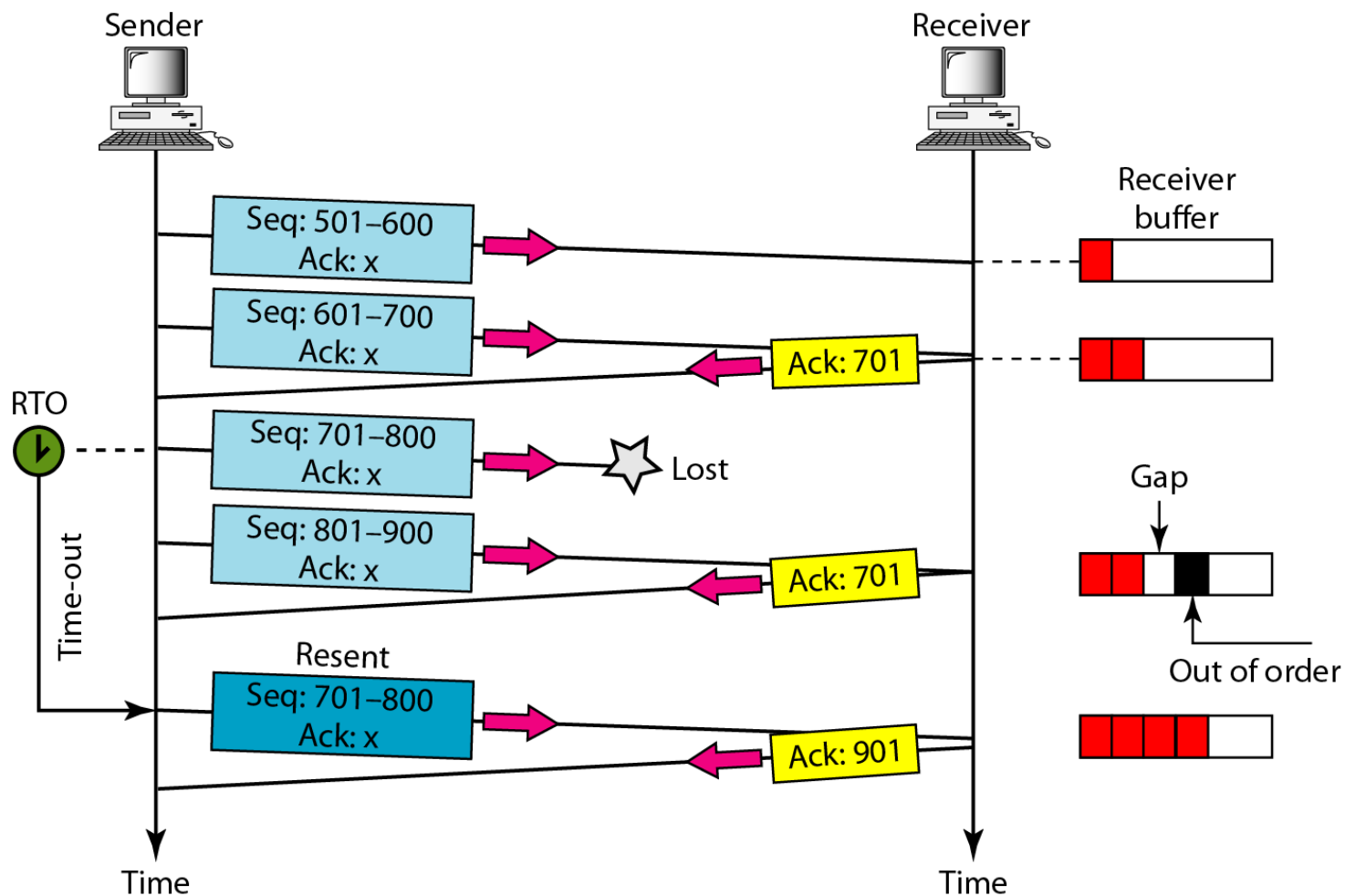


图 23.25 丢失段

TCP操作 III

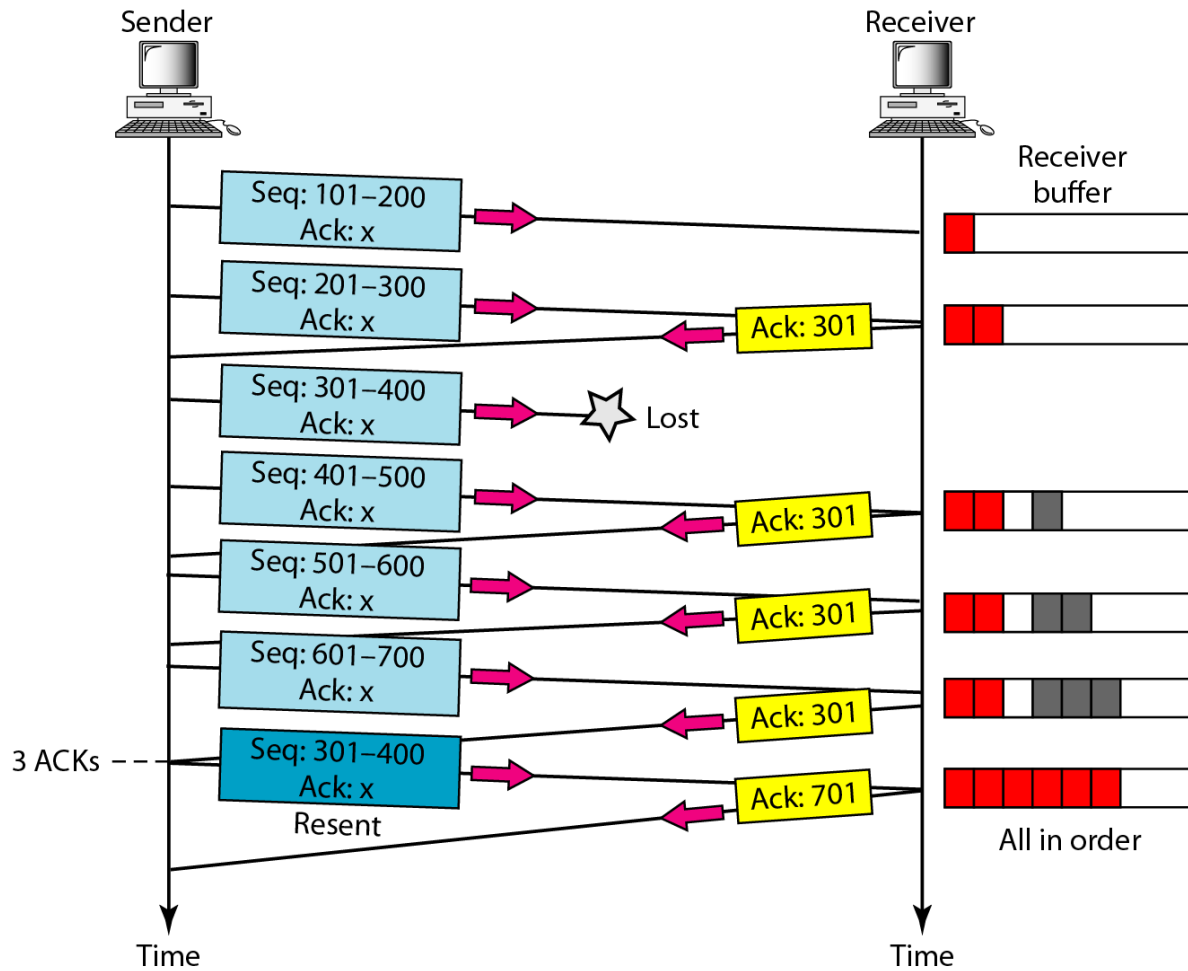


图 23.26 快速重传

拥塞控制（Congestion Control）

拥塞是分组交换网络中的一个重要问题。如果网络中的载荷即发送到网络中的分组数量，超过了网络的容量即网络中能处理的分组数量，在网络中就可能发生拥塞。拥塞控制指的是控制拥塞和使载荷低于网络容量的机制和技术。拥塞会引起数据丢失、时延增加、资源浪费和网络应用性能下降。

链路吞吐量

理想情况

1000M/s

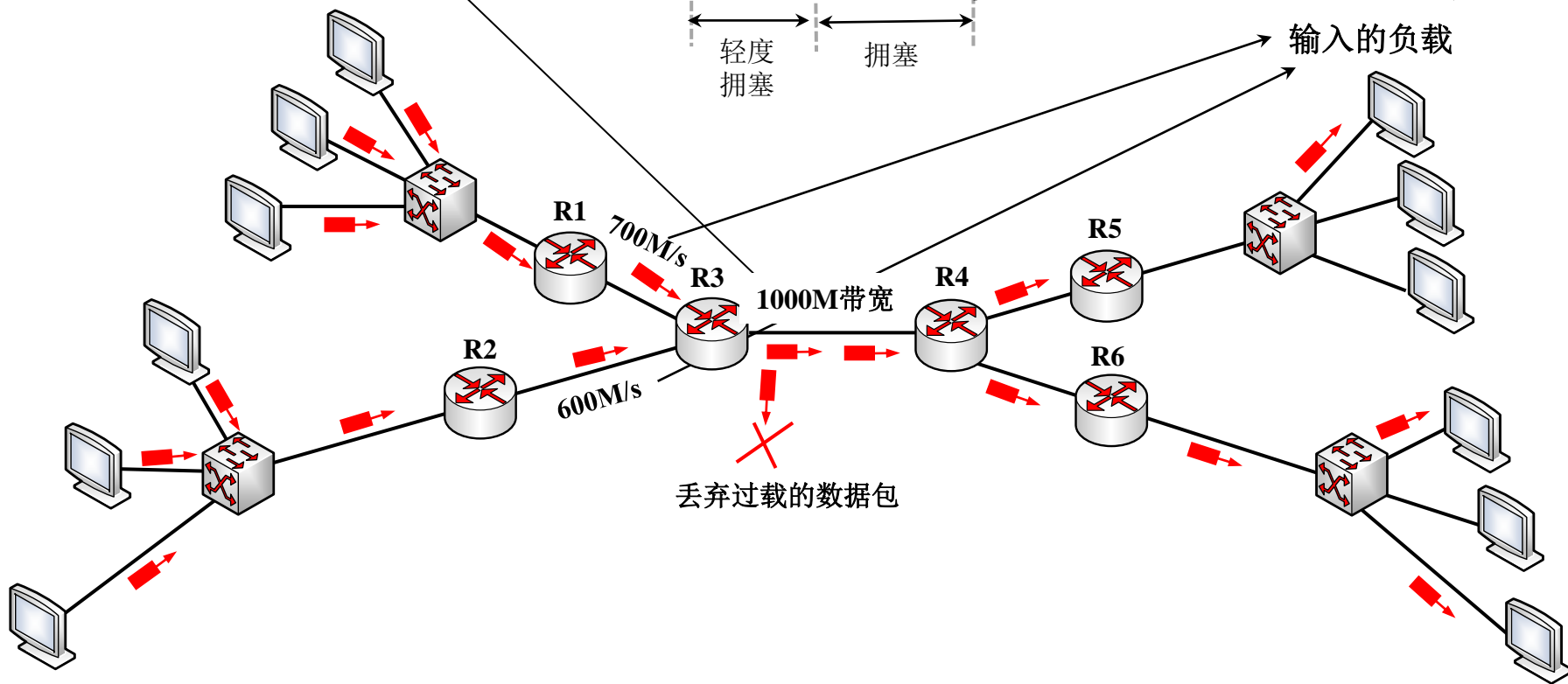
实际情况

死锁

轻度
拥塞

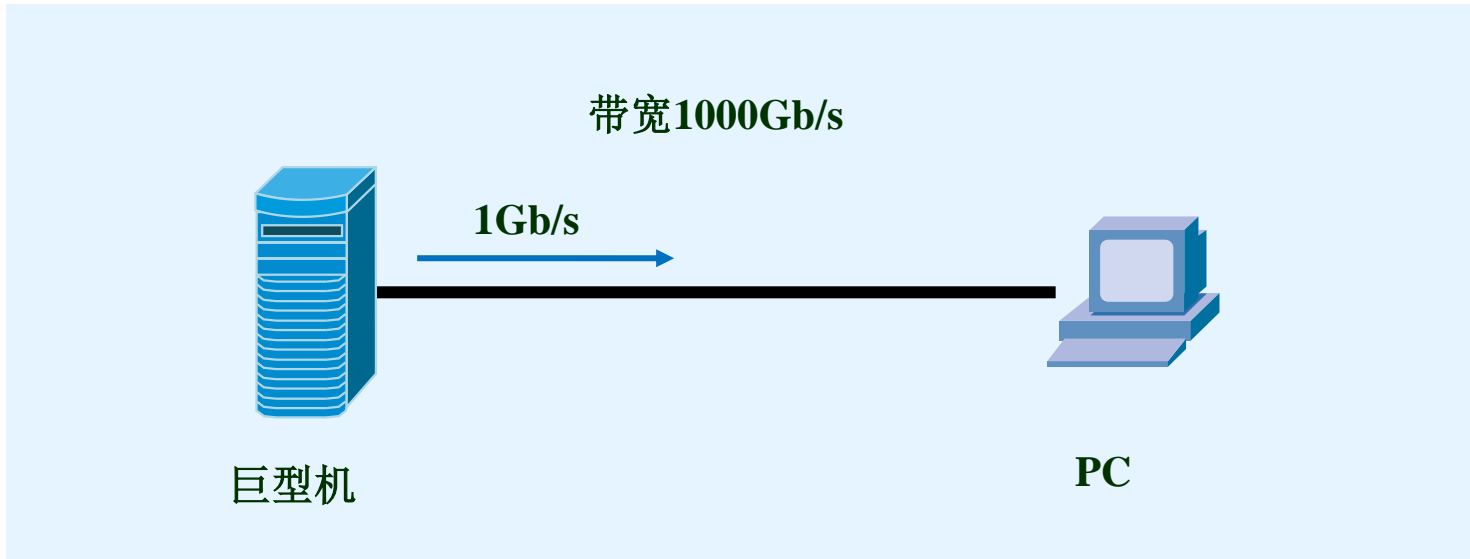
拥塞

输入的负载

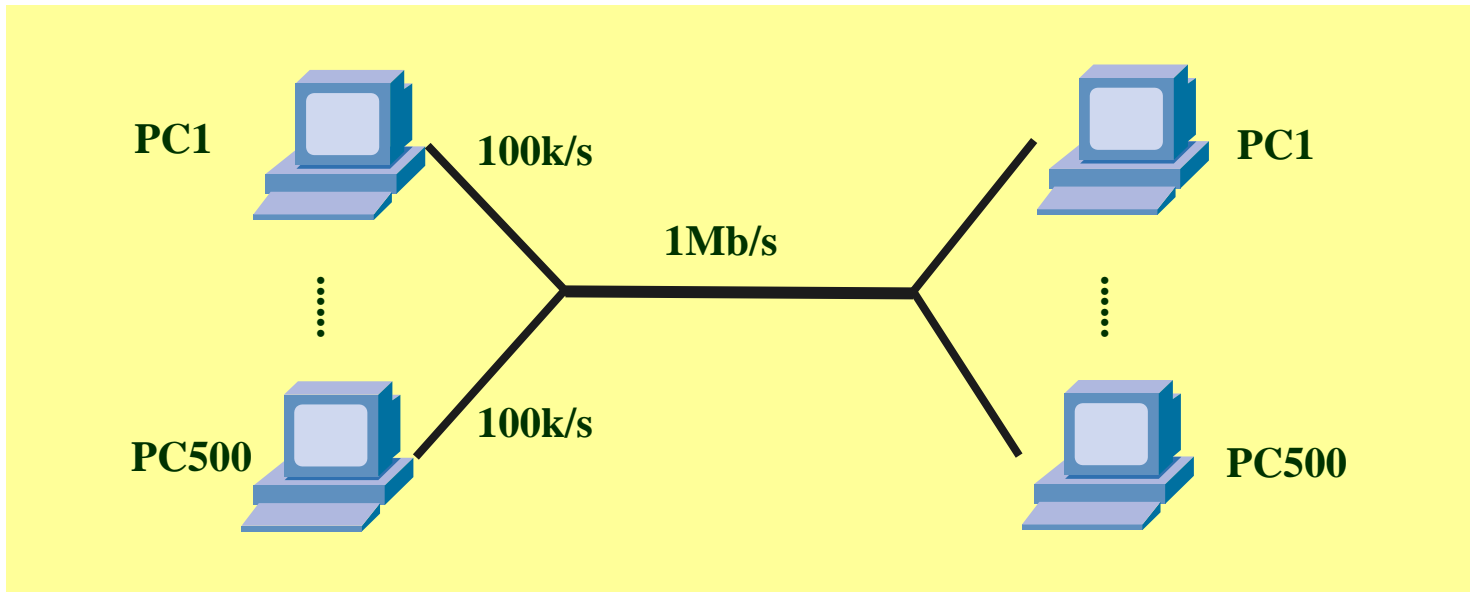


拥塞控制与流量控制的关系

- **拥塞控制**所要做的都有一个前提，就是网络能够承受现有的网络负荷。
- 拥塞控制是一个全局性的过程，涉及到所有的主机、所有的路由器，以及与降低网络传输性能有关的所有因素。拥塞窗口通过对网络拥塞的监测来调整和限定TCP发送端的发送流量。
- **流量控制**往往指在给定的发送端和接收端之间的点对点通信量的控制。
- 流量控制所要做的就是抑制发送端发送数据的速率，以便使接收端来得及接收。



无拥塞控制
需流量控制



无流量控制
需拥塞控制

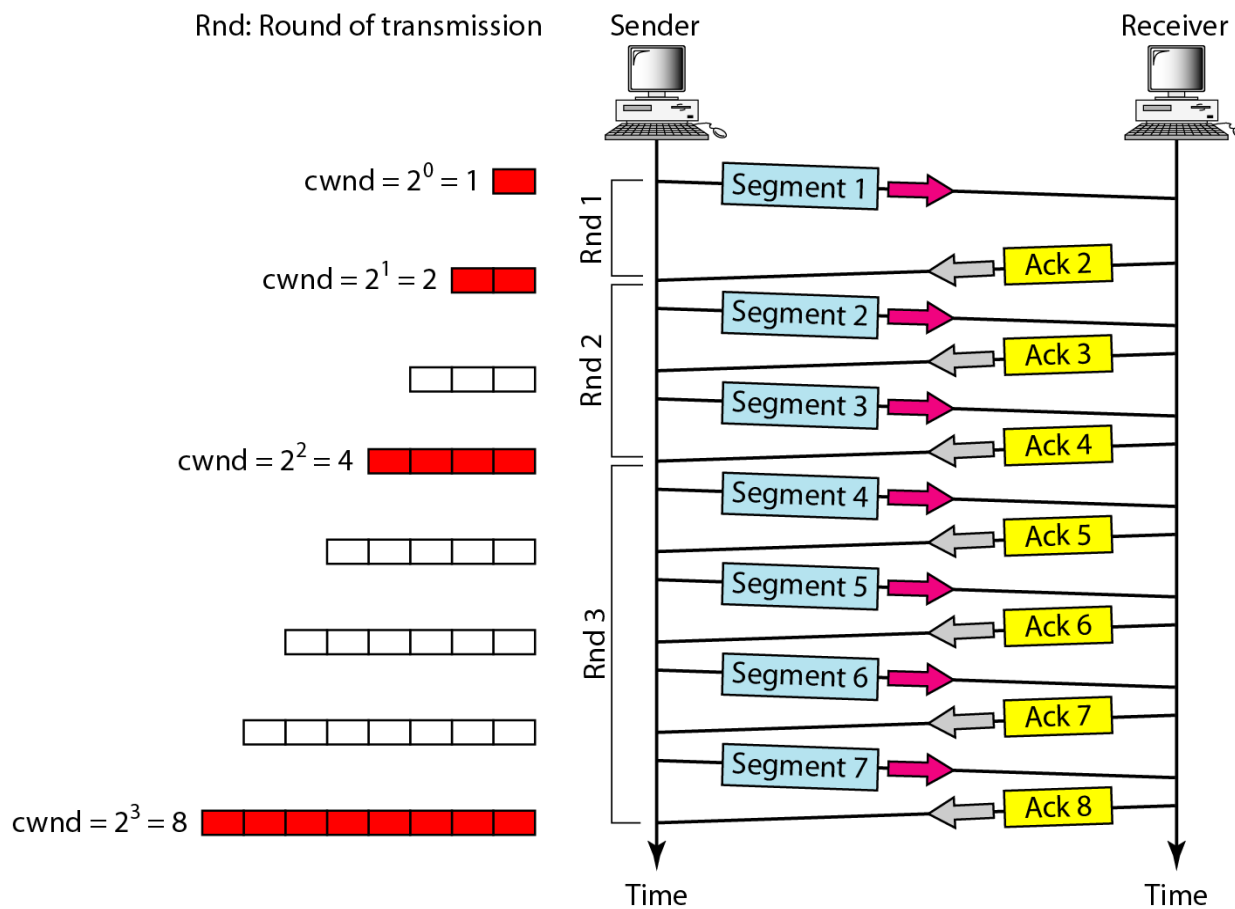
拥塞控制

- 发送方控制拥塞窗口的原则：发送方维持一个拥塞窗口cwnd，其大小取决于网络的拥塞程度，且动态变化，发送方让自己的发送窗口小于等于cwnd。只要网络没有发生拥塞，拥塞窗口就再增大一些，以发送更多的分组。如果网络发生拥塞，就将拥塞窗口减小一些，减少发送到网络中的分组数，如果发送方没有按时收到确认报文，就猜想网络可能发生了拥塞。
- 接收方窗口rwnd，拥塞窗口cwnd，实际窗口= $\min(rwnd, cwnd)$
- 拥塞策略：
 - 慢启动：很慢的传输速率启动，迅速增大到阈值
 - 拥塞避免：达到阈值后为避免拥塞降低数据速率
 - 拥塞检测：检测到拥塞返回到慢启动或拥塞避免

慢速启动：指数增长

- 慢启动：TCP发送端先以一个较低的数据率发送数据，例如先发送一个TCP报文，其长度为1个MSS（Maximum Segment Size），如果在重传定时器超时之前收到对该数据段的正确确认，则成倍增加这个拥塞窗口使其变为2个MSS。。。以此类推直至达到阈值。
- 简化前提：使用报文的个数而不是字节的个数；rwnd比cwnd大得多；每段单独确认。

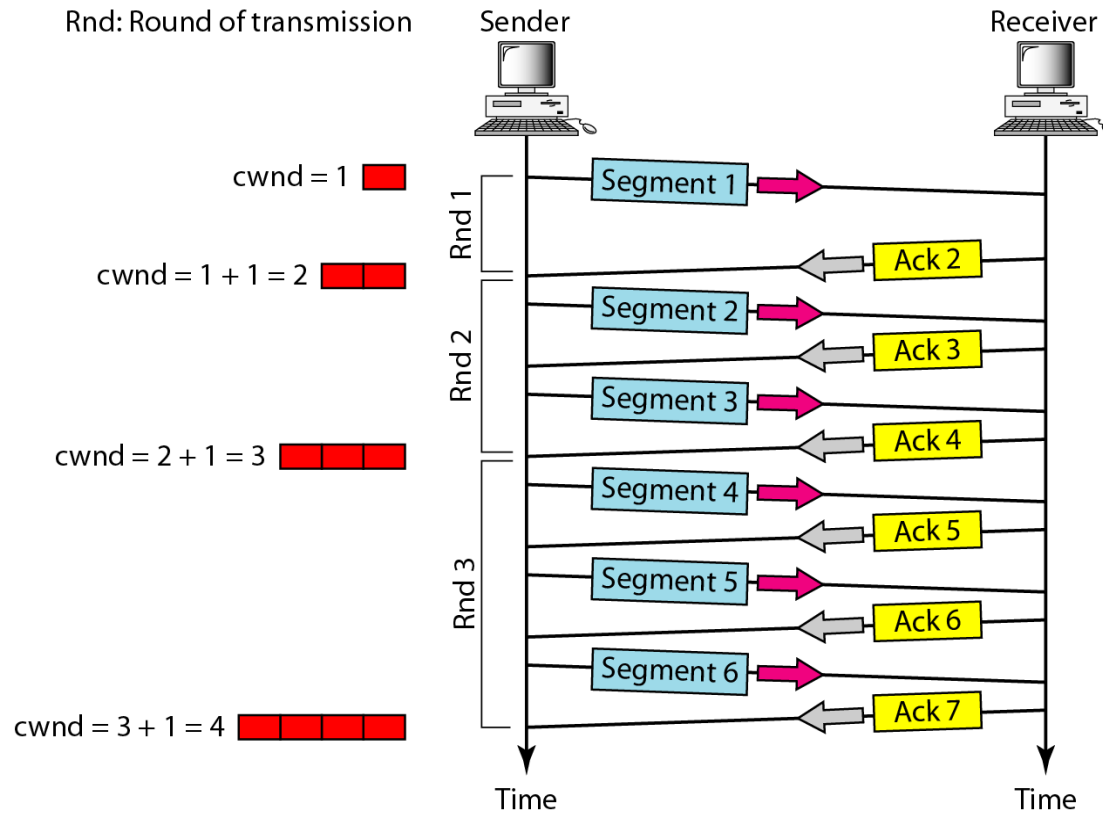
图 24.8 慢速启动: 指数增加



拥塞避免：加性增加

- 当拥塞窗口的大小达到慢启动的阈值时，慢启动阶段停止，加性增加阶段开始。
- 拥塞避免：不同于慢启动时发送数据速率的成倍增长，每次整个窗口所有段都被确认（一次传输）时，采用每次增加 1个MSS的递增方式，直到检测到数据丢失。

图 24.9 拥塞避免:加性增加

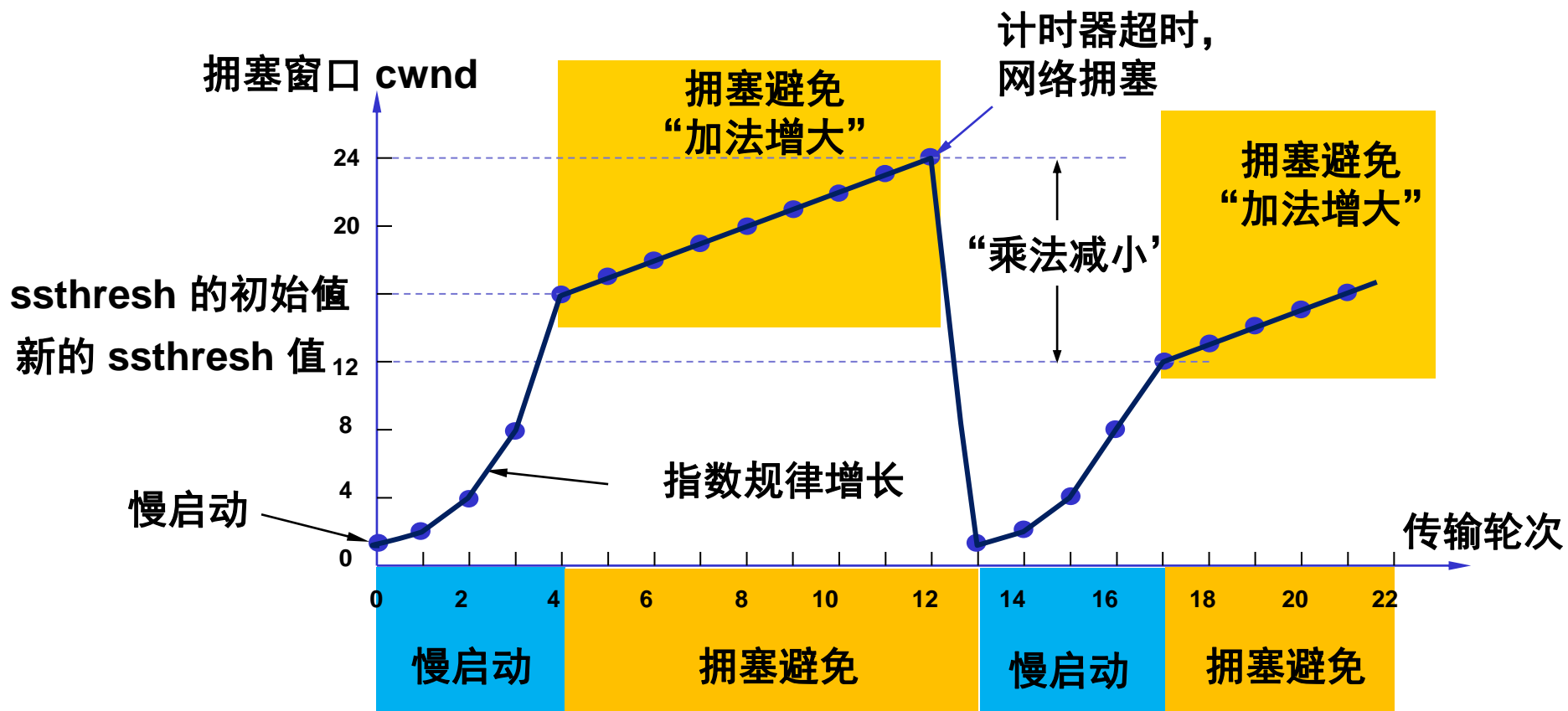


拥塞检测：乘性减少

如果发生拥塞，必须减小拥塞窗口。发送方通过重传的要求判断拥塞程度。

- 如果检测到计时器超时，说明拥塞严重，可能发生了段的丢失，TCP做出强烈反应：
 - 设置阈值为当前拥塞窗口的一半；
 - 设置cwnd为1个段的大小；
 - 进入慢启动阶段。
- 如果接收到三个ACK，说明轻度拥塞，一个段可能丢失，有些段可能到达，这个过程称为快速重传和快速恢复。TCP做出轻度反应：
 - 设置阈值为当前拥塞窗口的一半；
 - 设置cwnd为阈值的大小；
 - 进入拥塞避免阶段。

计时器超时后的拥塞检测



收到三个ACK后的快速恢复

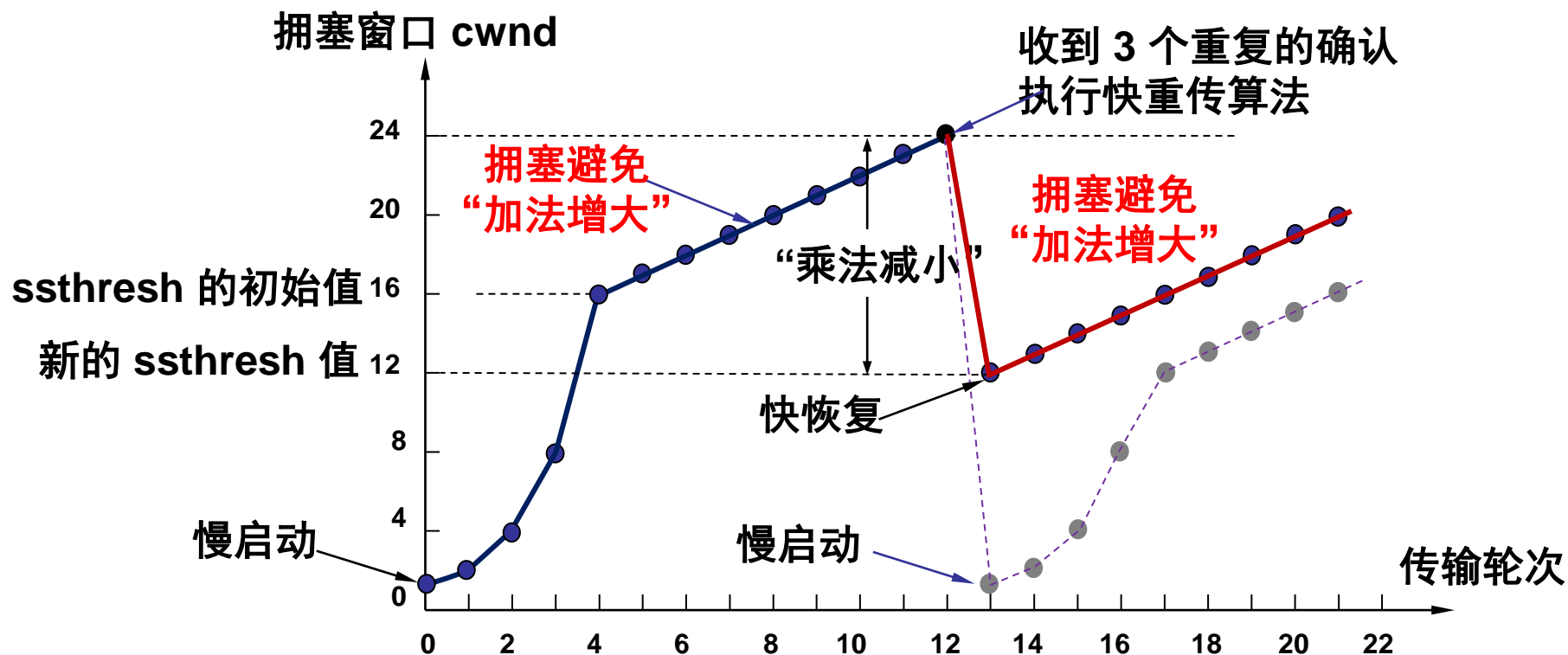


图 24.10 TCP拥塞策略总结

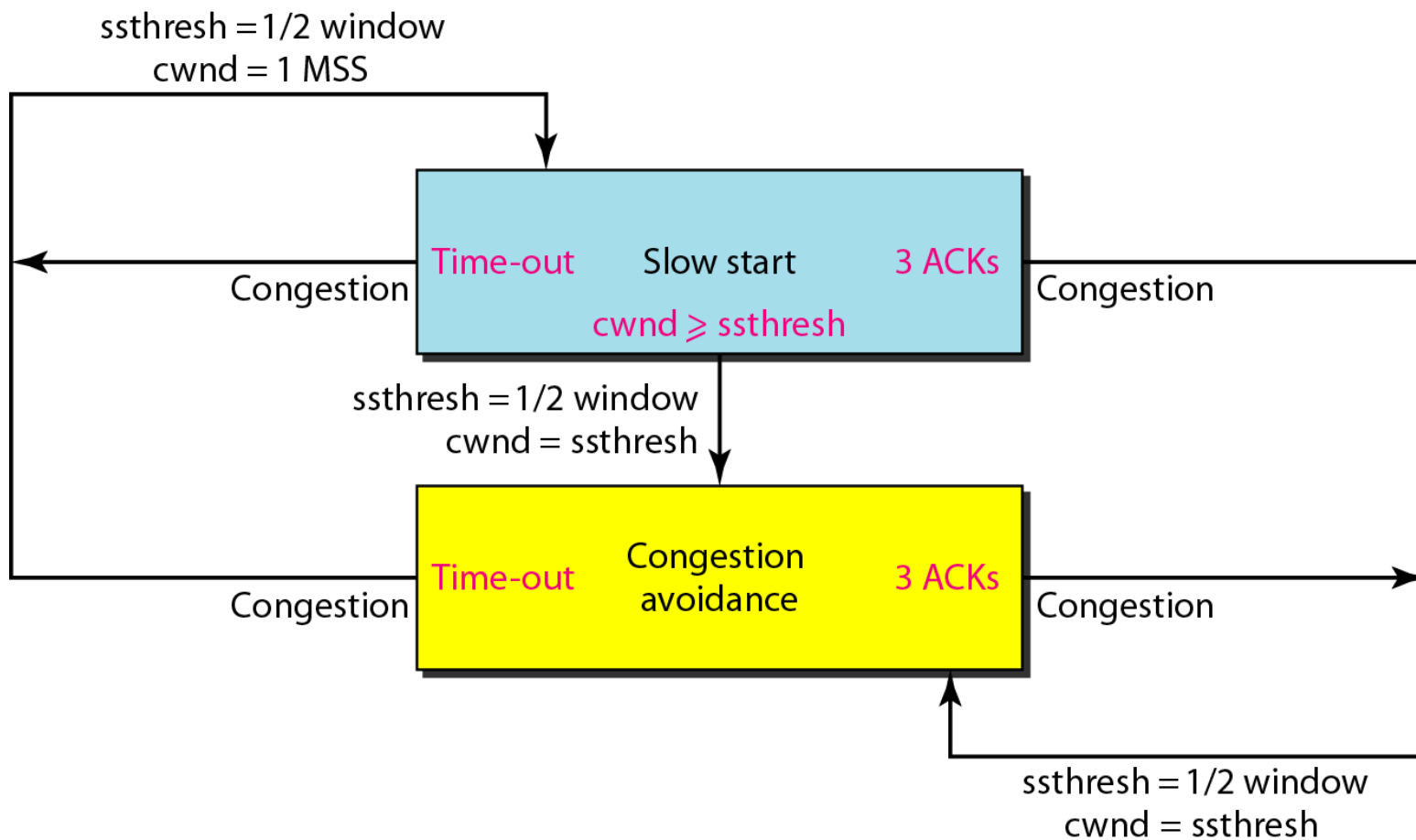
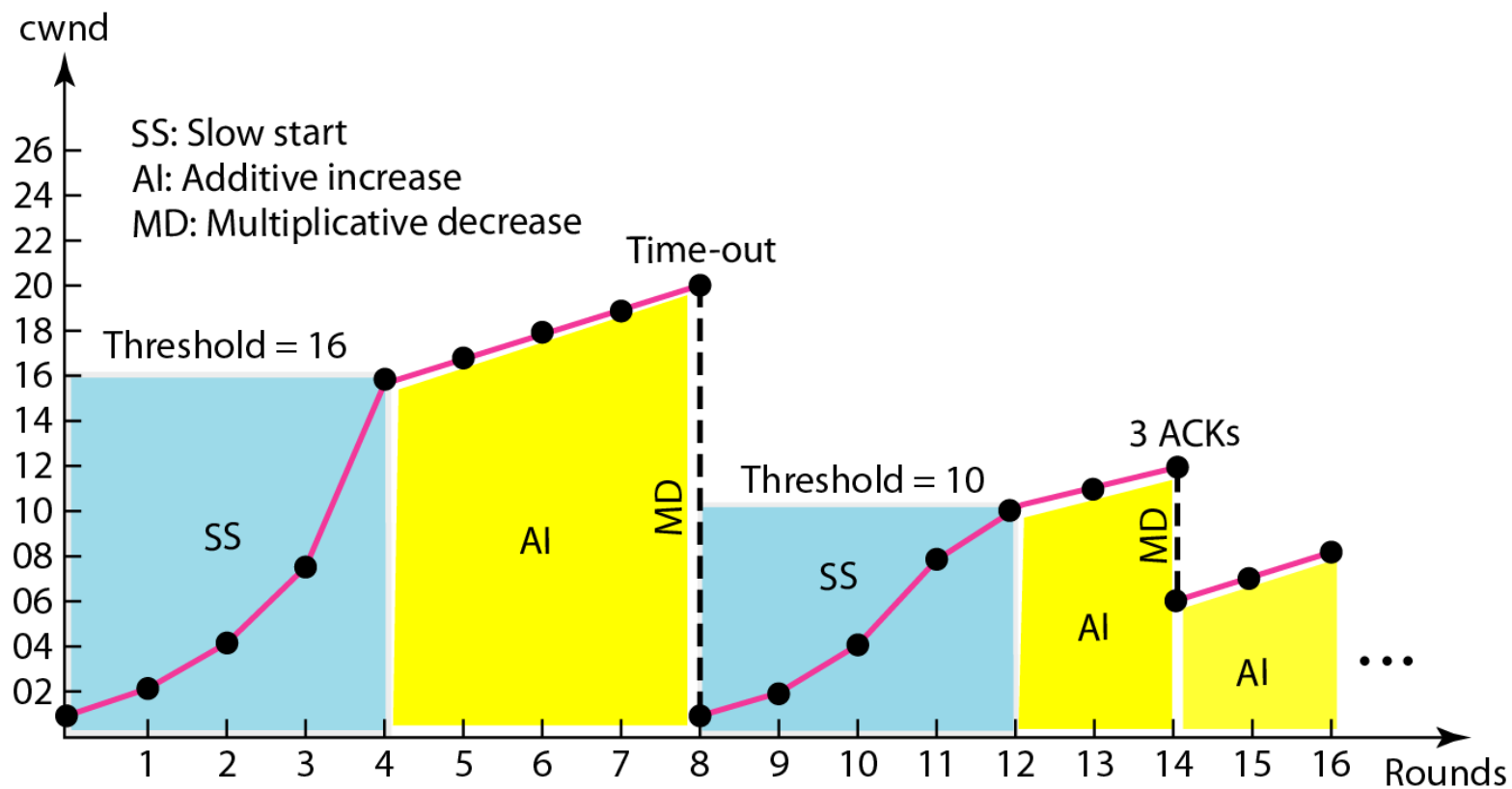


图 24.11 拥塞例子



23-4 流控传输协议SCTP (Stream Control Transmission Protocol)

流控传输协议是一种新的可靠的、面向报文的传输层协议。然而，SCTP主要是为最近引入的因特网应用而设计的。这些新应用所需要的服务都比TCP能提供的更复杂。

本节要点:

SCTP服务及其特性
分组格式
SCTP关联
流量控制和差错控制



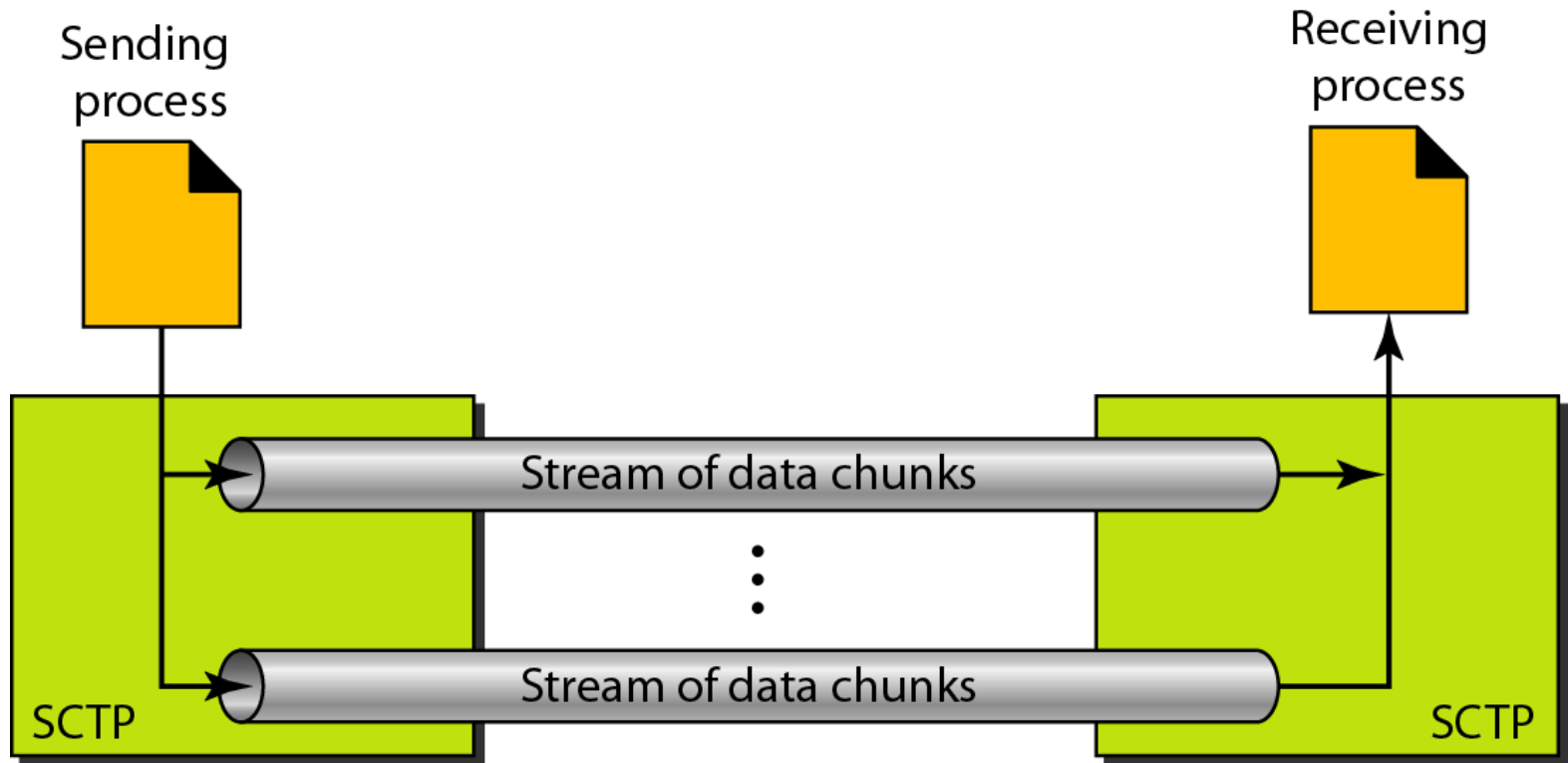
注意

SCTP是一种面向报文的可靠的协议，它兼有**UDP**和**TCP**的最佳特性。

表 23.4 某些SCTP应用

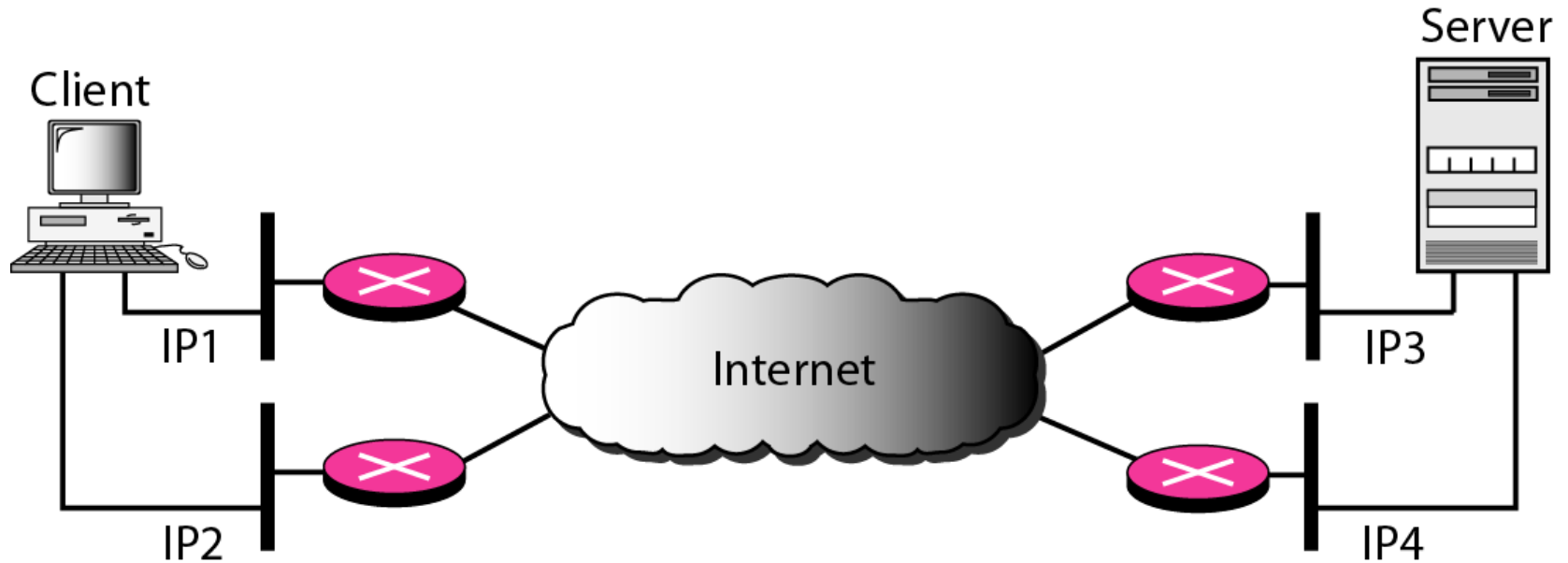
<i>Protocol</i>	<i>Port Number</i>	<i>Description</i>
IUA	9990	ISDN over IP
M2UA	2904	SS7 telephony signaling
M3UA	2905	SS7 telephony signaling
H.248	2945	Media gateway control
H.323	1718, 1719, 1720, 11720	IP telephony
SIP	5060	IP telephony

图23.27 多流概念



SCTP的一次关联可以包含多个流。

图 23.28 多接口概念

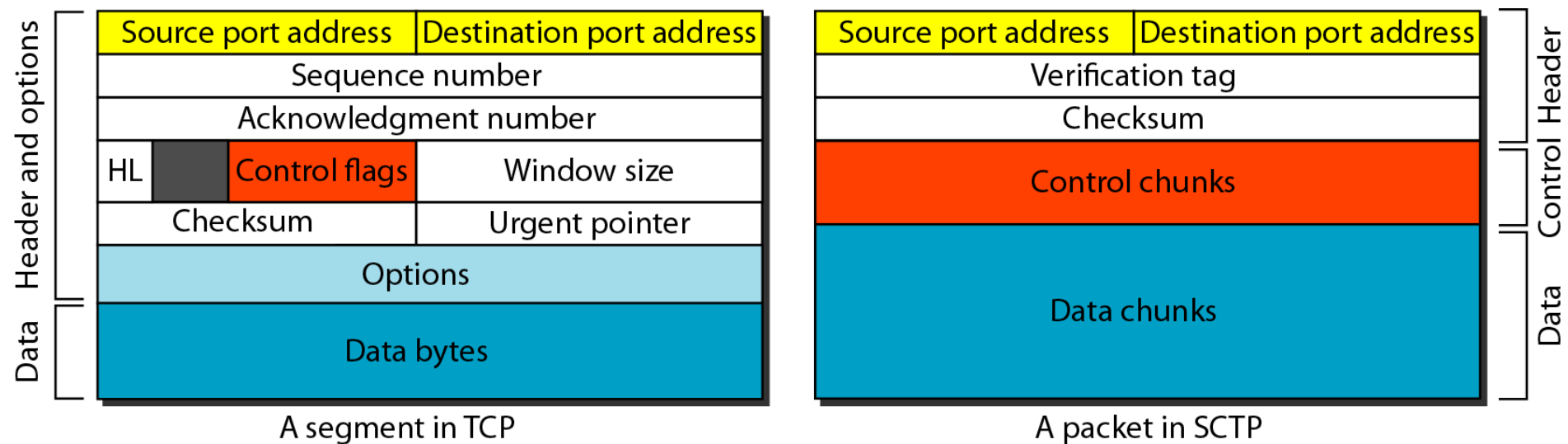



SCTP关联允许每个端有多个IP地址。



- 在SCTP中，数据大块（**Data Chunk**）按传输序列号（**TSN**）编号。
- 为了区别不同的流，SCTP使用SI。
- 为了区别属于同一个流中的不同数据大块，SCTP使用流序列号（**SSN**）。
- TCP有段，SCTP有分组。

图 23.29 TCP段与SCTP分组的比较





注意

在SCTP 中，控制信息和数据信息
在分开的大块中携带。

作 业

- P500
- 21, 26, 29