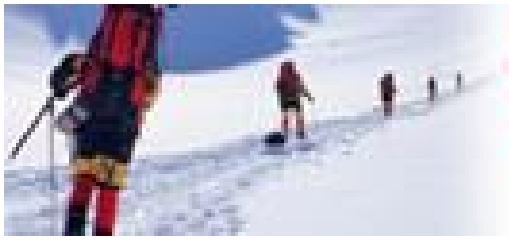


Data Communications and Networking Fourth Edition

Forouzan

第23章

UDP、TCP和SCTP

第五部分 传输层

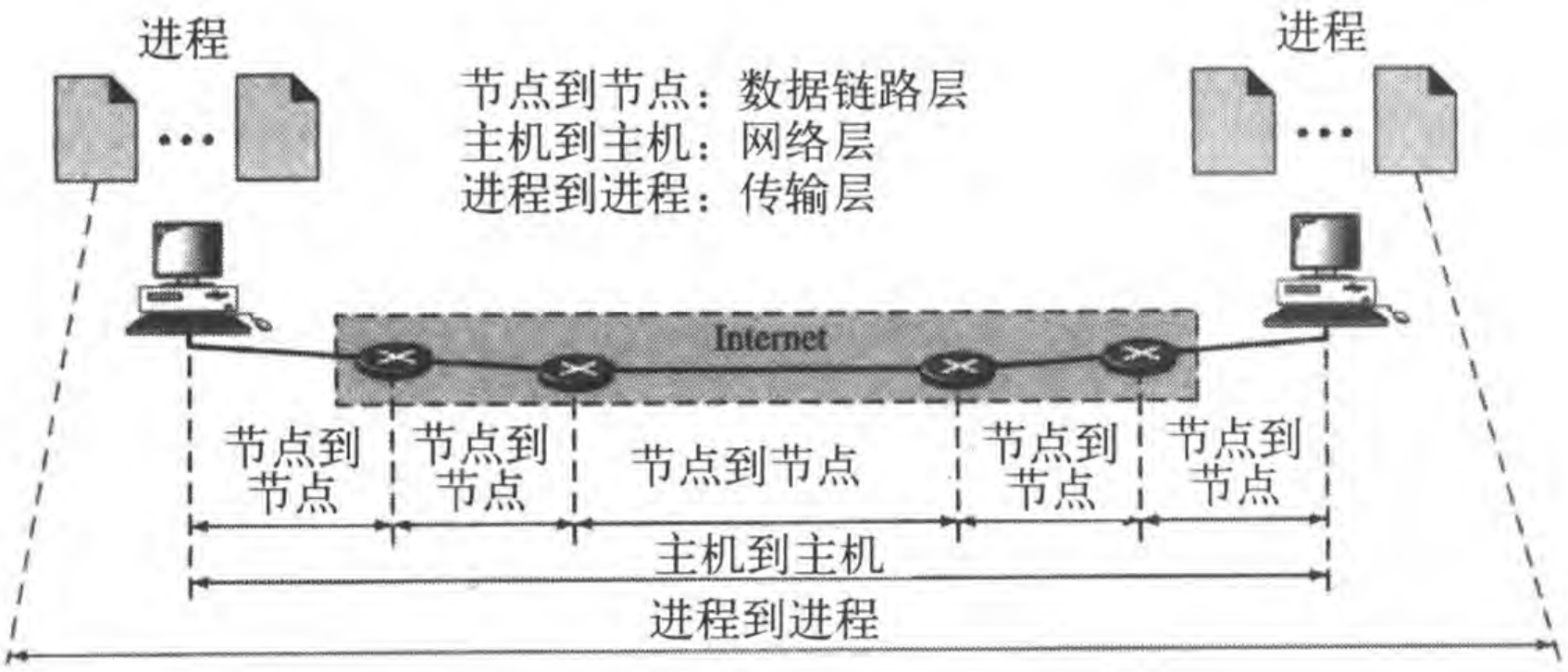
- 传输层负责整个报文的进程到进程的传递；
- 虽然网络层监管独立分组从源端到目的端的传递，但它不辨认这些分组之间的关系，它独立对待每个分组；另一方面，传输层在从源端到目的端的层次中，监管差错控制和流量控制以确保全部报文完整地、按序地到达；
- 从源端到目的端的传递不仅是一个计算机到下一个计算机的传递，还是计算机上的一个特定进程到另一计算机上的一个特定进程的传递；
- 在OSI模型中，传输层头部必须包含一个称为服务点地址的地址类型，或在因特网和TCP/IP协议族中的端口号或端口地址。

- 传输层协议可以是无连接的，也可以是面向连接的；
- 无连接传输层将每段作为一个独立的分组处理，将它传递到目的机器的传输层；面向连接传输层在传递分组前要与目的机器的传输层建立一个连接，当所有数据传输完成后，终止这个连接；
- 在传输层，一个报文通常被划分成可传输的段；无连接协议（**UDP**）对每一段独立处理，面向连接的协议（**TCP**和**SCTP**）还要用序号生成这些段之间的关系；
- 传输层也负责流量控制和差错控制，但它们是端到端的，而不是在单一一条链路上；
- **UDP**不包含流量控制和差错控制，**TCP**和**SCTP**使用滑动窗口进行流量控制，使用确认系统进行差错控制

23-1 进程到进程的传递

- 数据链路层负责链路上的两个相邻节点之间的帧传递，这称为节点到节点的传递；
- 网络层负责两台主机之间的数据报（分组）传递，这称为主机到主机的传递；
- 因特网中的通信实际发生在两个进程（应用程序）之间的；
- 任何时刻，源主机和目的主机上可能都运行着多个进程，为了完成传递过程，需要一种机制将源主机上运行的某个进程的数据发送到目的主机上运行的对应进程上；
- 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

图23.1 数据传送类型


进程
进程
节点到节点：数据链路层
主机到主机：网络层
进程到进程：传输层
Internet
节点到节点
节点到节点
节点到节点
节点到节点
节点到节点
主机到主机
进程到进程

客户/服务器（C/S）模式

- 最常用的一种实现进程到进程通信的方法；
- 本地主机上的进程称为客户（client），它通常需要来自远程主机上的进程提供的服务，这个远程主机上的进程称为服务器（server）；
- 请求服务的进程称为客户，提供服务的进程称为服务器；
- 目前操作系统支持多用户和多程序运行的环境，对通信来说，必须定义：
  - 1. 本地主机
  - 2. 本地进程
  - 3. 远程主机
  - 4. 远程进程

寻址

- 数据链路层需要MAC地址（如果连接不是点到点的）选择一个节点，网络层需要一个IP地址来选择一个主机；
- 传输层需要端口号从主机上运行的多个进程中选择相应的进程，目的端口号用于传送，源端口号用于接收回答；
- 端口号是0-65535间的16位整数，客户程序用端口号定义它自己，这个端口号由运行在客户主机上的传输层软件随机选择，称为临时端口号；
- 因特网给服务器使用全局端口号（事先分配好），它们称为熟知端口号（有例外，有一些客户端也被分配了熟知端口号，如DHCP）；
- 目的IP地址定义了在世界范围的不同主机中确定一个主机，端口号定义了在该特定主机上的多个进程中的一个进程（图23.3）

图23.2 端口号

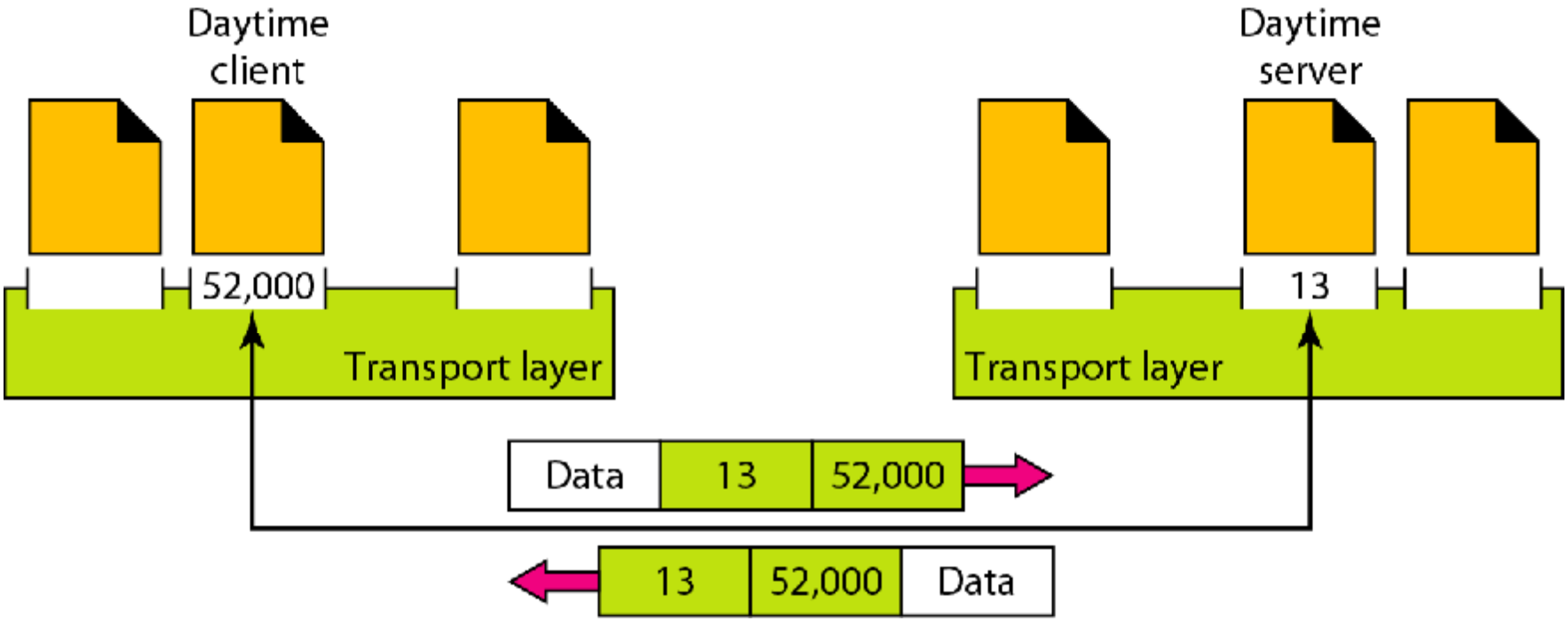

Daytime
client
Daytime
server
52,000
13
Transport layer
Transport layer
Data
13
52,000
13
52,000
Data

图23.3 IP地址与端口号

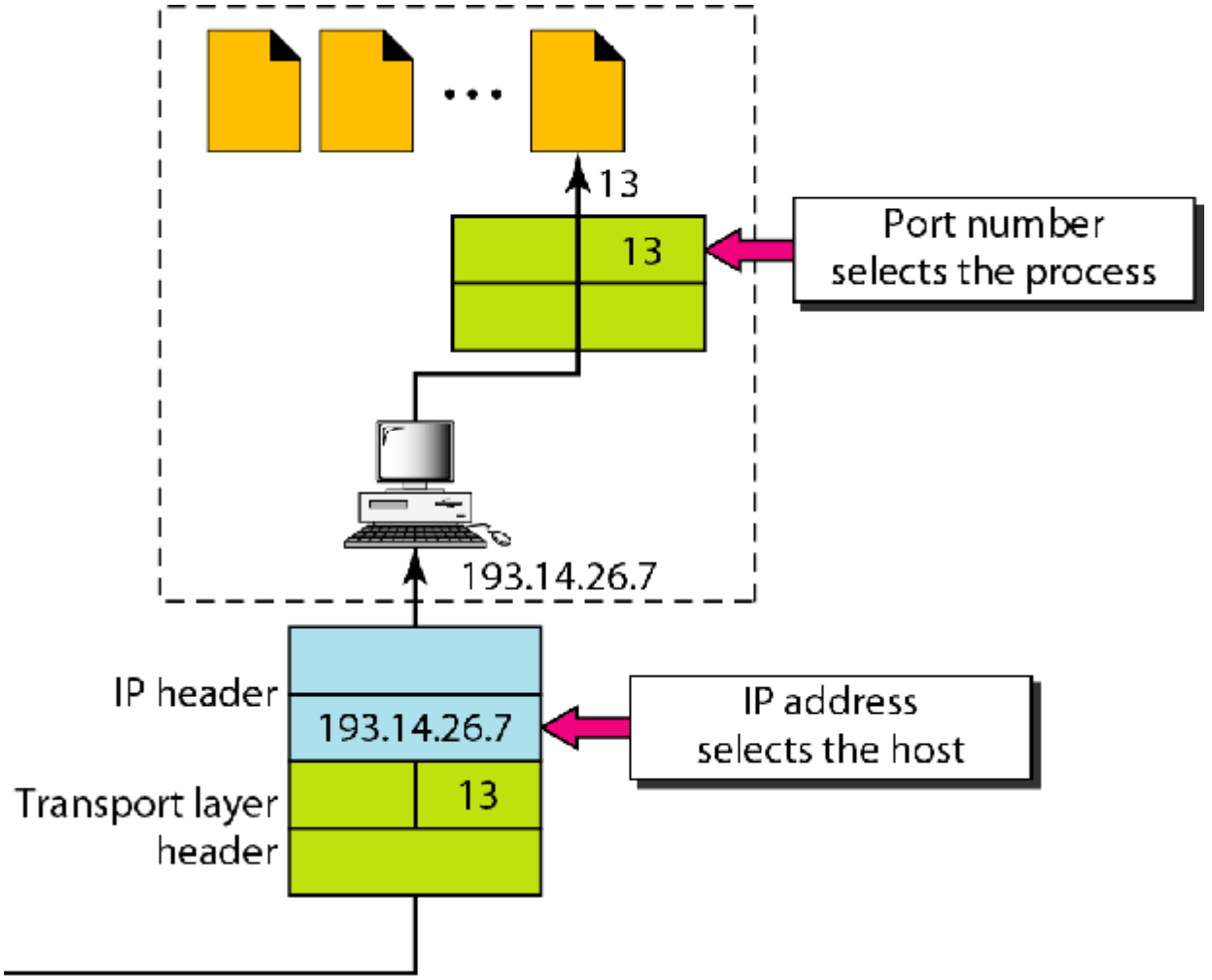


图23.4 IANA 范围

p因特网号码分配管理局（IANA）把端口编号划分为三种范围：

Ø熟知端口：范围0-1023，由IANA分配和控制，这些是熟知端口；

Ø注册端口：范围是1024-49151，IANA不分配或也不控制，可在IANA 注册以防重复；

Ø动态端口：范围是49152-65535，既不受控制也不需要注册，可以由任何进程使用，是临时端口。

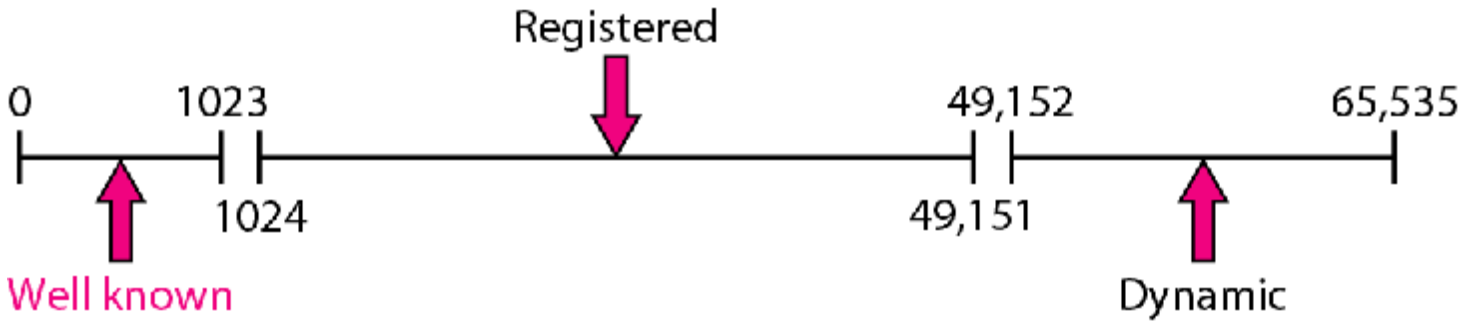


图23.5 套接字地址

- 一个IP地址和一个端口号结合起来称为套接字地址；
- 客户套接字地址唯一定义了客户机进程，而服务器套接字地址唯一地定义了服务器进程；
- 传输层协议需要一对套接字地址：客户套接字地址和服务器套接字地址；
- 这四条信息是IP头部和传输层协议头部的组成部分，IP头部包含IP地址，而UDP或TCP头部包含端口号。

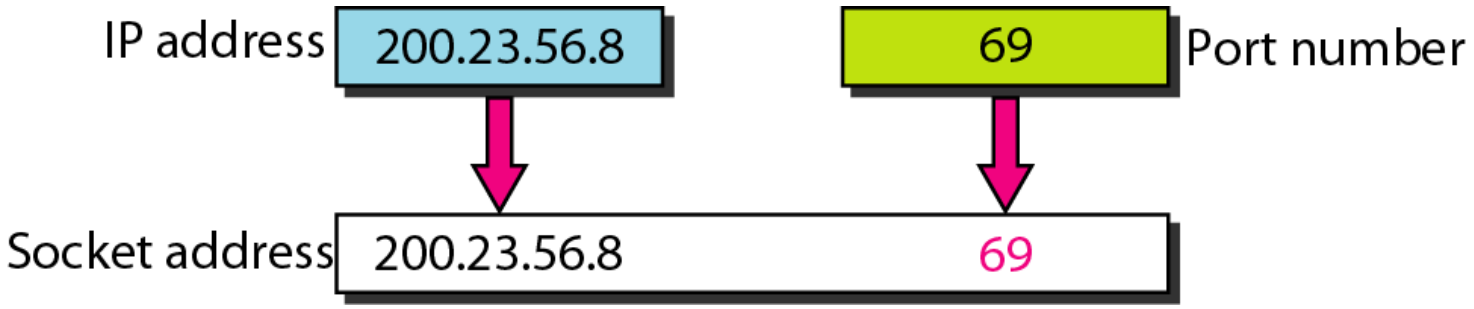
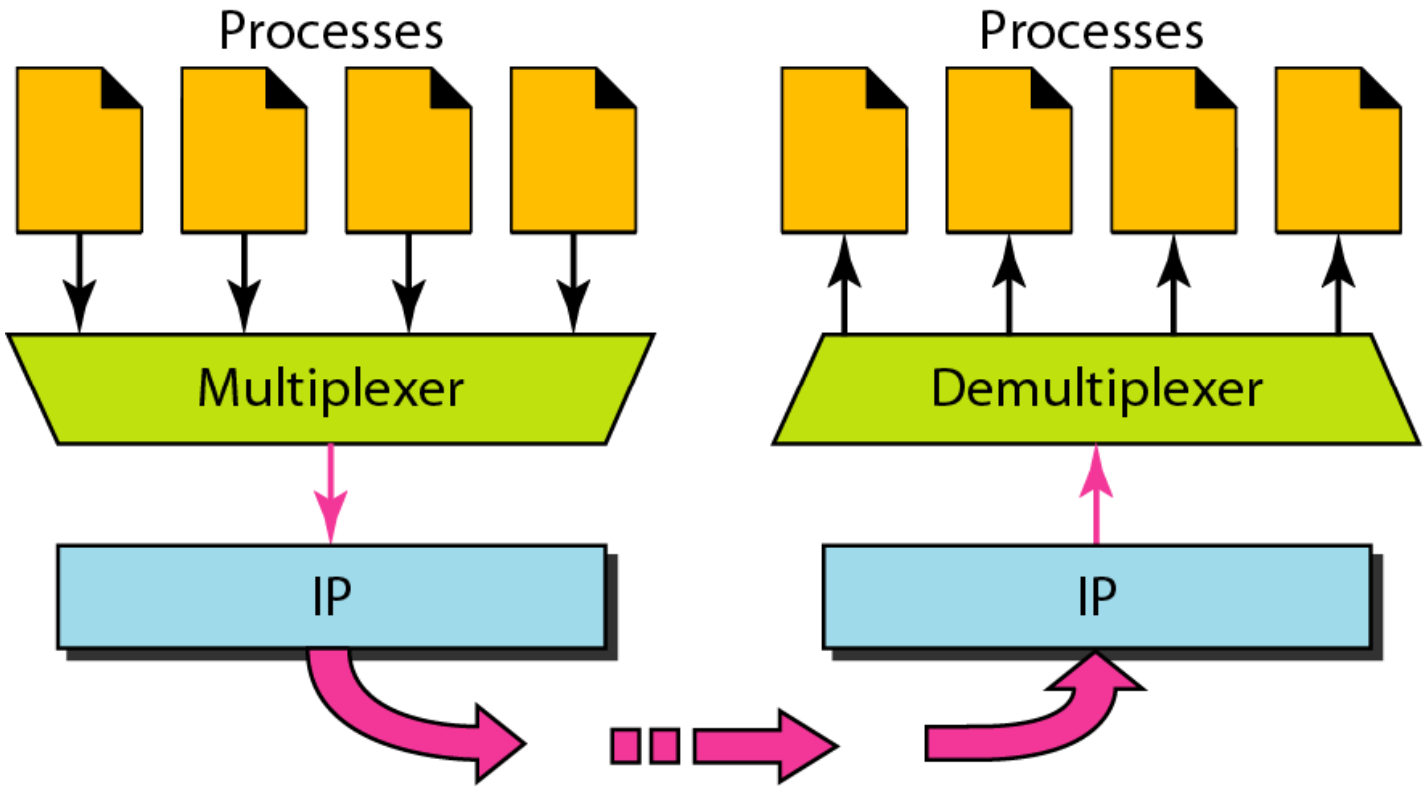


图23.6 复用和分离（传输层借助于端口）



无连接服务与面向连接的服务

- 无连接服务：分组从一方发送给另一方，不需要建立连接和释放连接；分组没有编号，它们可能被延迟、丢失或无序到达，也没有确认过程；UDP就是无连接的；
- 面向连接的服务：首先在发送方和接收方之间建立一个连接，然后传送数据，最后释放连接；TCP和SCTP都是一种面向连接的协议。

可靠服务与不可靠服务

- 传输层服务可以是可靠的或不可靠的；
- 传输层通过实现流量控制和差错控制来获得可靠性，这意味着一种较慢和更复杂的服务；
- 数据链路层可靠并不代表传输层不需要流量控制和差错控制，因为在数据链路层的可靠性存在于两个节点之间，而不是端到端的可靠性；并且网络层是不可靠的，必须在传输层实现可靠性；
- 数据链路层的差错控制并不能保证传输层的差错控制

图23.7 差错控制

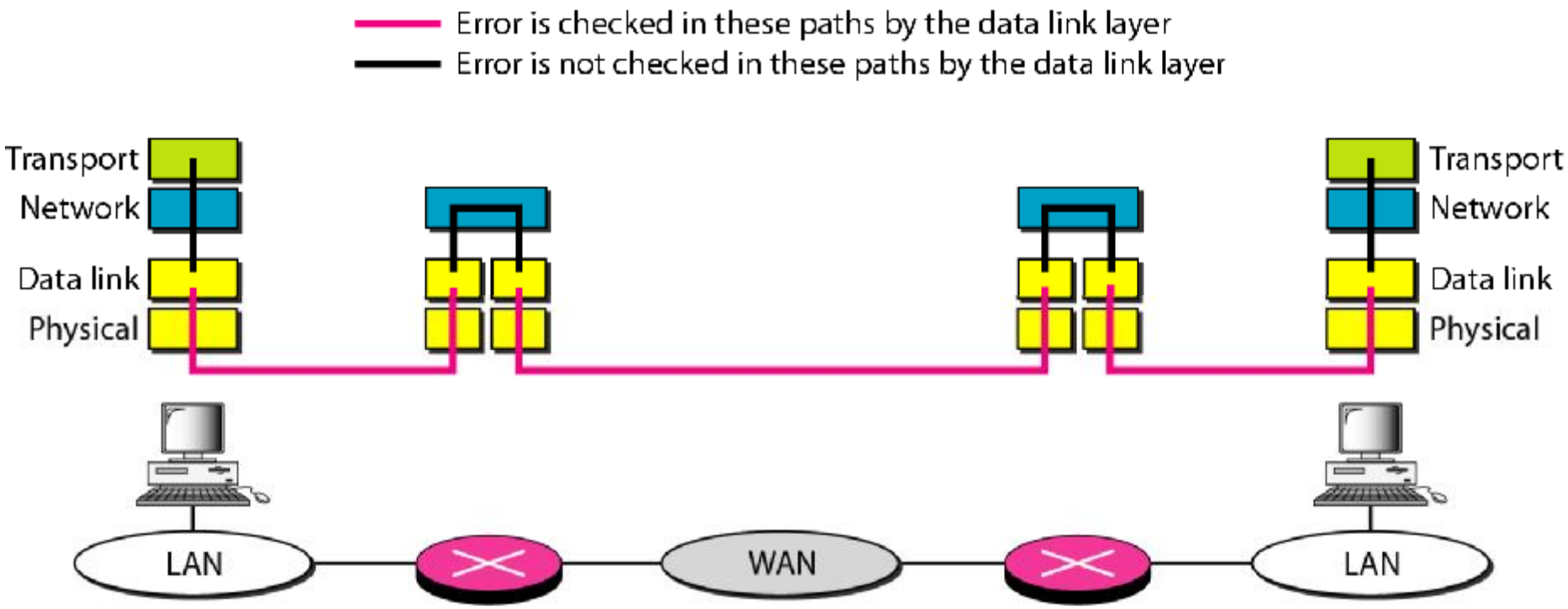
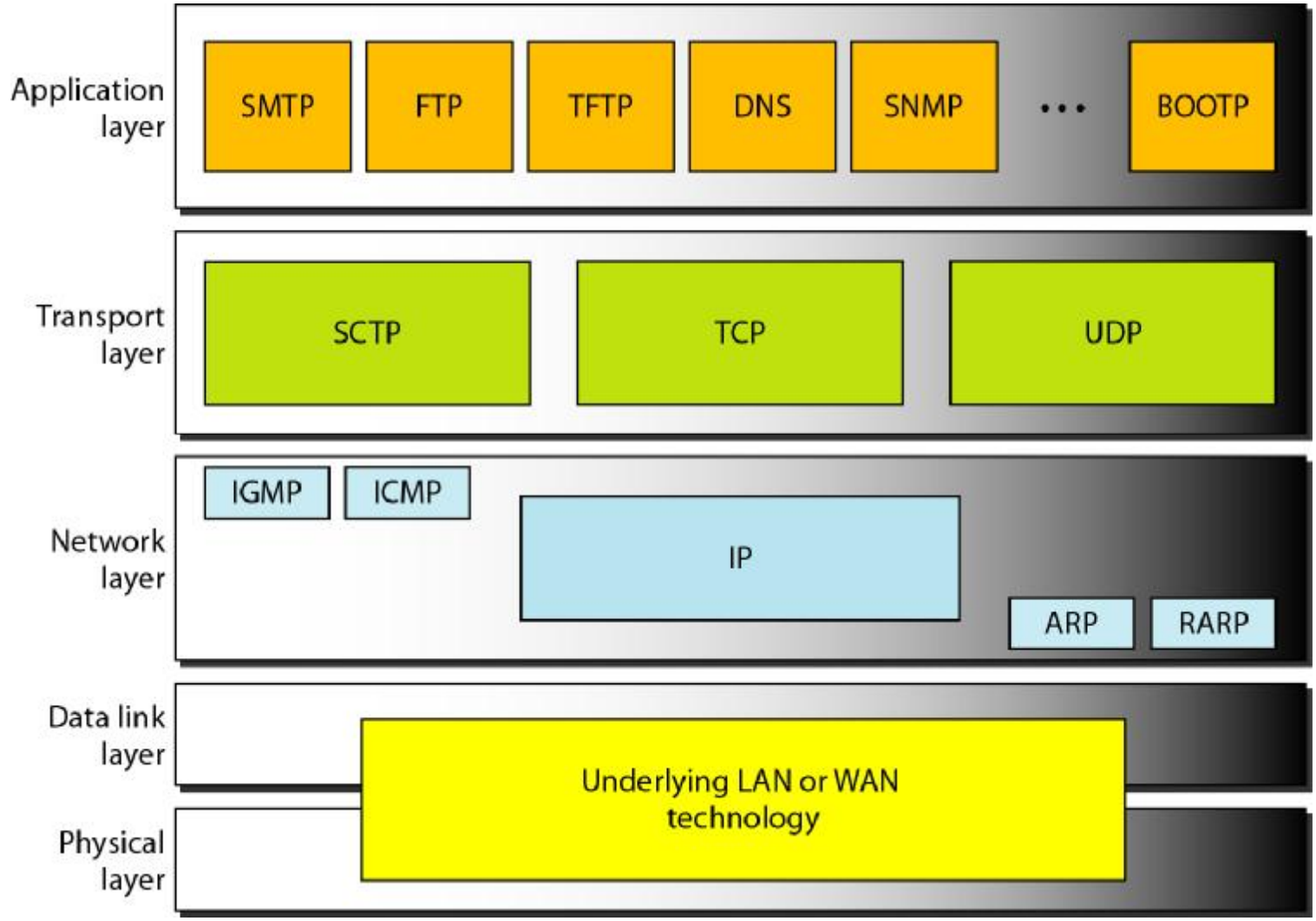

Error is checked in these paths by the data link layer
Error is not checked in these paths by the data link layer
Transport
Network
Data link
Physical
Transport
Network
Data link
Physical
LAN
WAN
LAN

图23.8 UDP、TCP和SCTP在TCP/IP协议族中的位置



23-2 用户数据报协议（UDP）

- 用户数据报协议（User Datagram Protocol，UDP）称为无连接不可靠的传输层协议；
- 它除了提供进程到进程通信而不是主机到主机的通信外，没有给IP服务增加任何东西，此外它还完成非常有限的差错检验；
- UDP是一个非常简单的协议，开销最小；如果一个进程想发送很短的报文，而且不在意可靠性，就可以使用UDP；
- 使用UDP发送一个很短的报文，在发送方和接收方之间的交互要比使用TCP或SCTP时少得多。

表 23.1 UDP使用的熟知端口

Port	*Protocol*	*Description*
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。注意：FTP可使用UDP或者TCP的端口号是21。

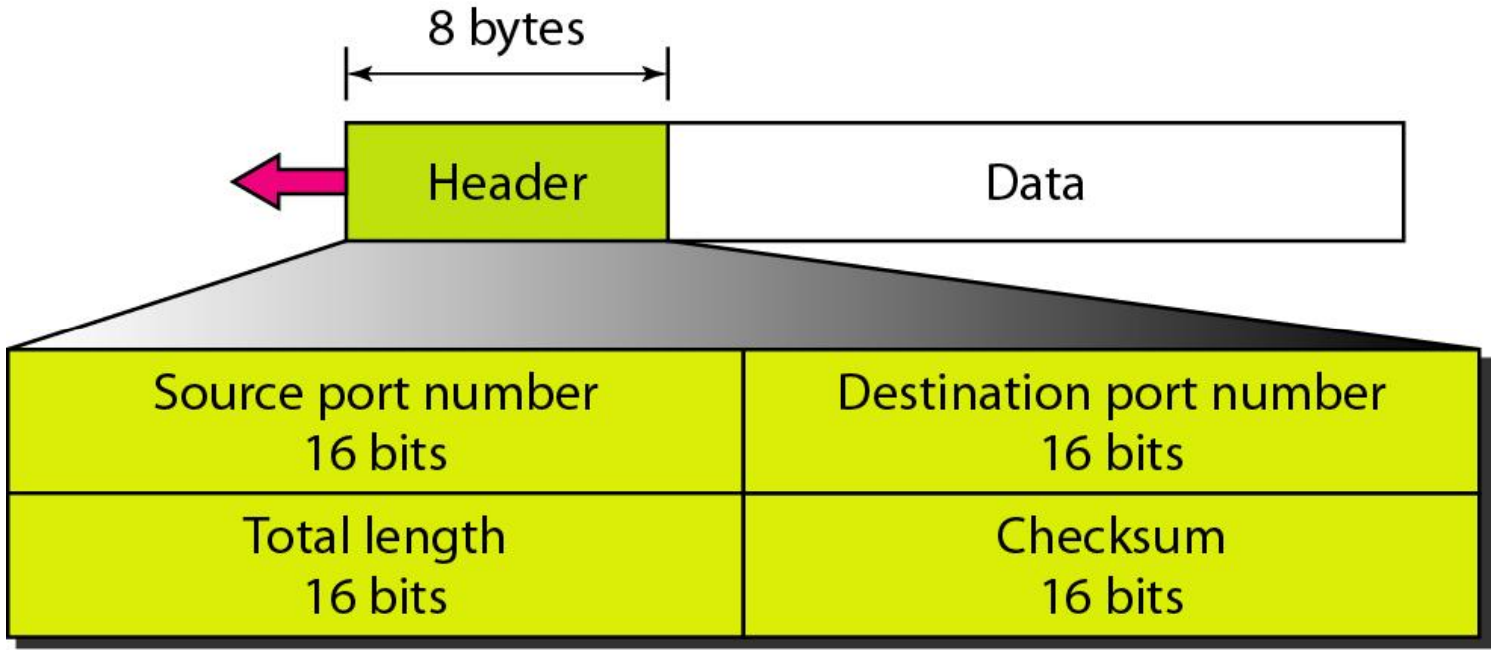
```
$ grep    ftp   /etc/services
ftp          21/tcp
ftp          21/udp
```

例23.1（cont.）

SNMP使用两个端口号（161和162），我们将会在28章看到，它们每一个用于不同目的。

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

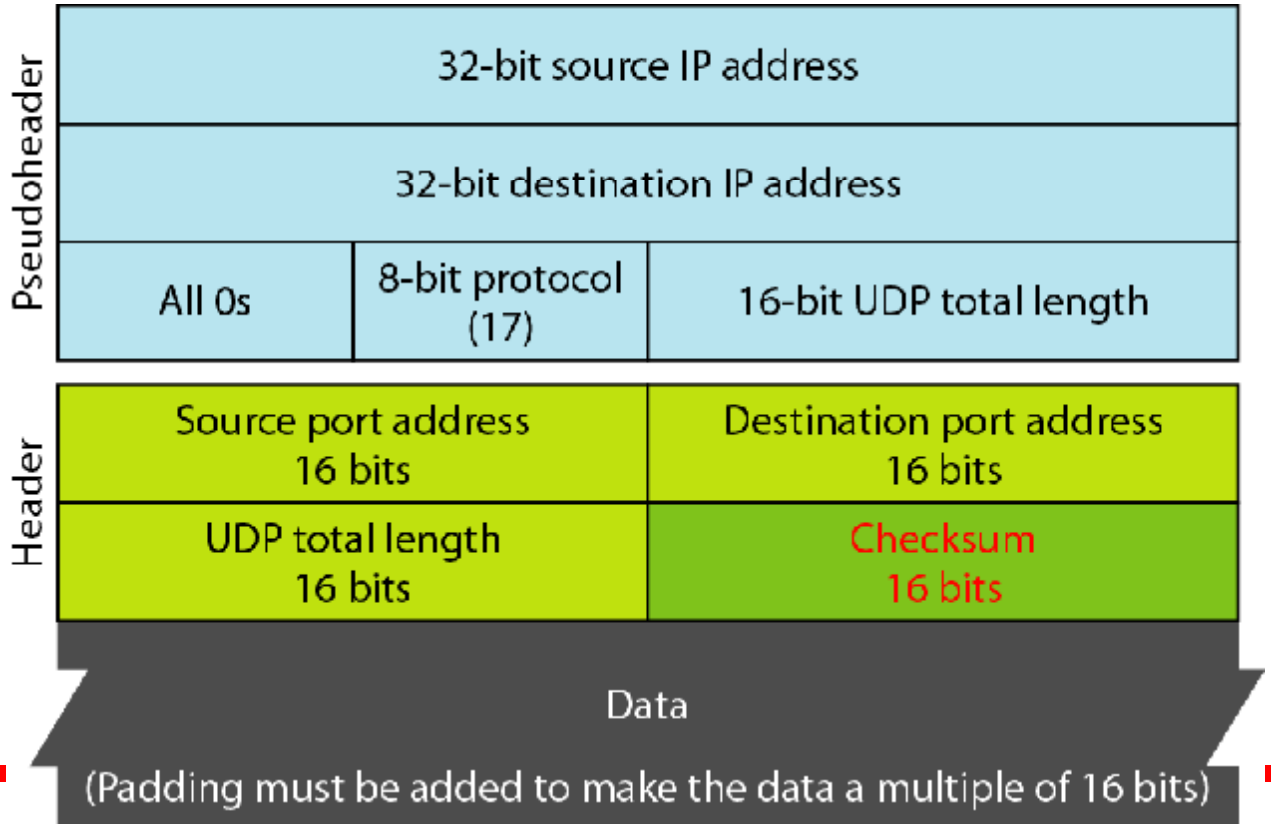
图 23.9 用户数据报格式（8字节固定头部）



UDP长度 = IP长度 - IP头部长度

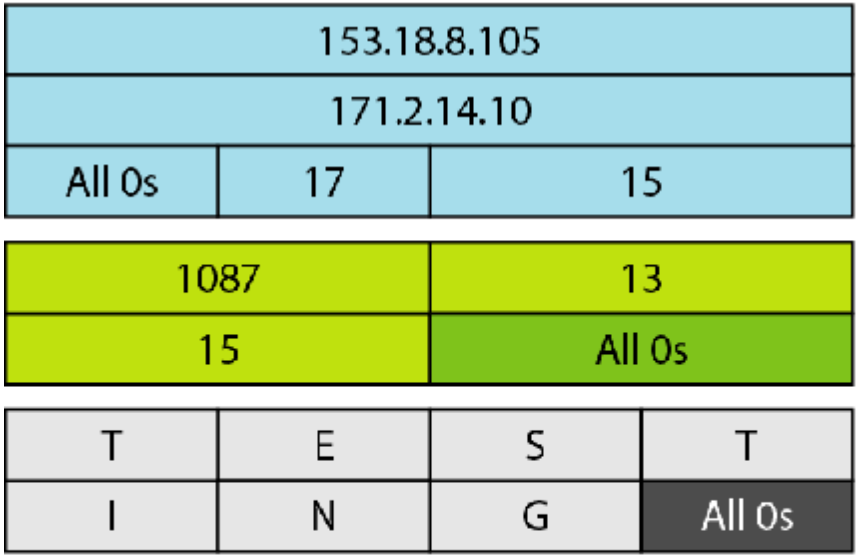
图 23.10 用于校验和计算的伪头部

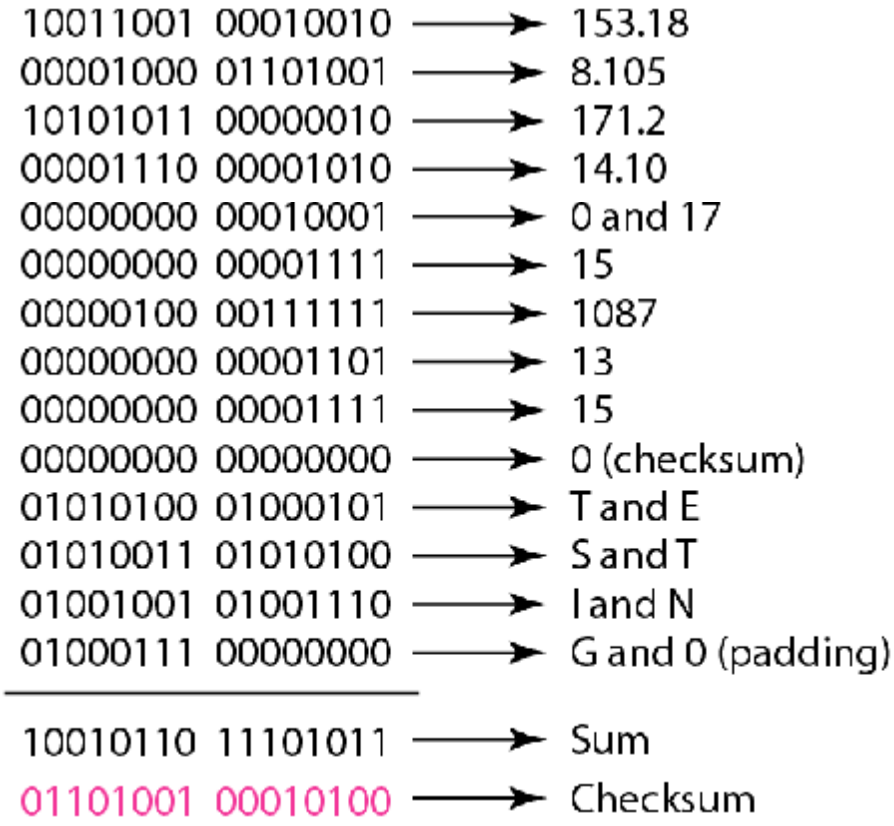
- UDP校验和计算需要加入伪头部，伪头部是IP分组头部的一部分；
- 伪头部保证数据报被递交给正确的主机、正确的传输层协议；
- 校验和计算是可选的，若不计算，填入全1。



例23.2

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。





UDP的操作

- 无连接服务：用户数据报独立处理，不编号，不建立连接和终止连接，无连接的结果是使用UDP的进程不能向UDP发送数据流（基于数据流的用户数据报一般前后相关联）；
- 没有流量控制，除校验和外没有差错控制机制，如果需要流量和差错控制则只能由进程提供；
- UDP报文封装在IP数据报中传输

UDP的操作-排队（客户端）

p在客户机端，当进程启动时，它从操作系统请求一个端口号；有些实现创建一个入队列和一个出队列与每一个进程相关联，而有些实现只创建与每一个进程相关的入队列；

p多数情况下，客户打开的队列由暂时端口号来标识；

p客户进程使用源端口将报文发送到出队列，UDP逐个将报文取出，加上头部递交给IP，如果发生溢出，操作系统就要求客户进程等待；

p报文到达客户端时，UDP检查以确认对应于目的端口号是否创建了入队列；如果已创建，UDP就将收到的用户数据报放在队列末尾，否则UDP就丢弃该用户数据报，并请求ICMP协议向服务器发送揣口不可达报义；

p所有发送给一个特定客户程序的入报文，不管是来自哪个服务器，都被放在同一个队列中；如果入队列发生溢出，UDP就丢弃此用户数据报，并请求向服务器发送端口不可达报文。

UDP的操作-排队（服务器端）

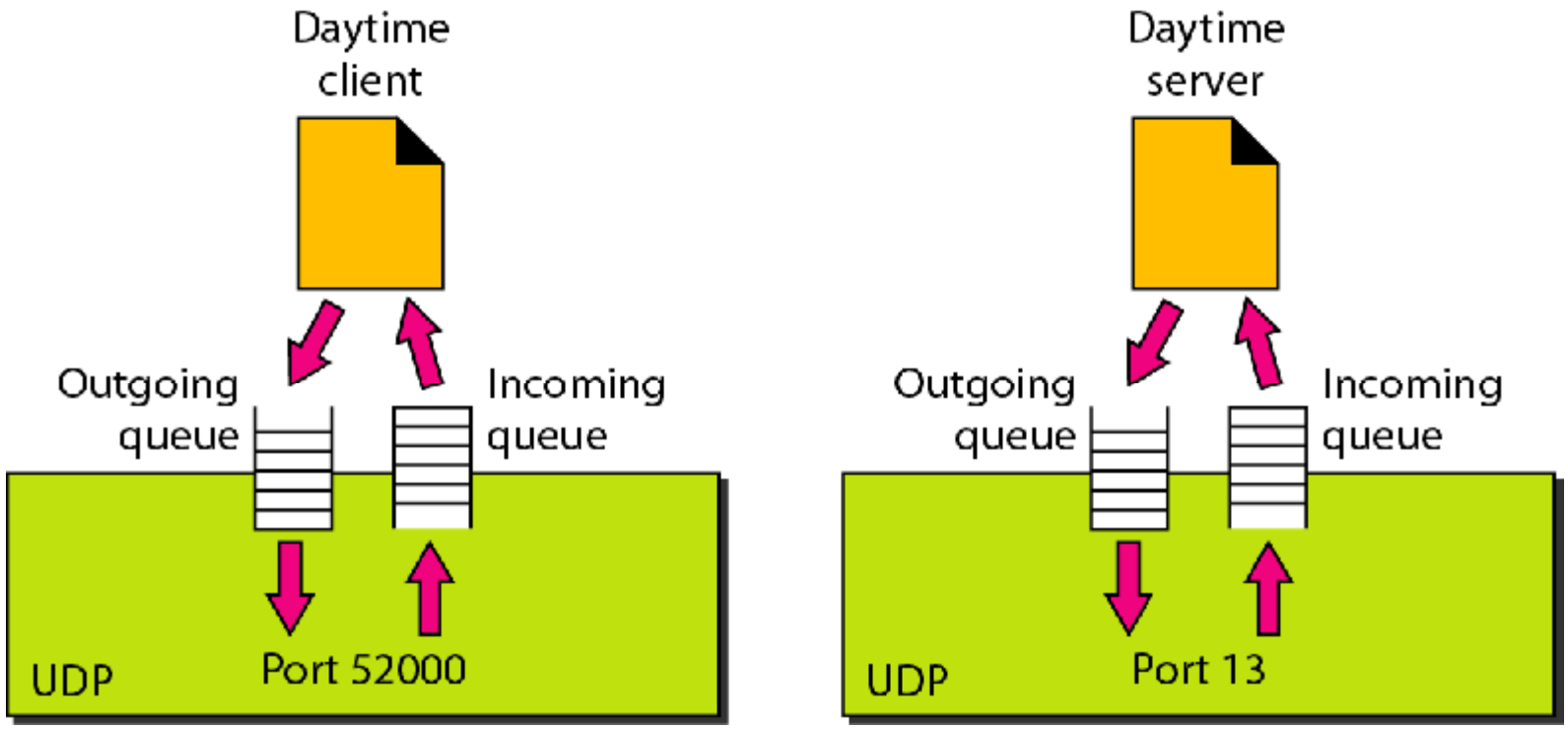
p在服务器端，服务器在它开始运行时就用它的熟知端口请求入队列和出队列，只要服务器运行，队列就一直是打开的；

p当报文到达服务器进程时，UDP检查以确认对应于目的端口号是否创建了入队列；如果已创建，UDP就将接收到的用户数据报放在队列末尾，否则UDP就丢弃该用户数据报，并请求ICMP协议向客户端发送一个端口不可达报文；

p一个特定服务器程序的所有入报文，不管是来自哪个客户机，都放入同一队列；如果入队列发生溢出，UDP就丢弃这个用户数据报，并请求向客户发送端口不可达报文；

p当服务器想要回答客户时，它就使用在请求报文中指明的源端口号将报文发送到出队列；UDP逐个将报文取出，加上头部递交给IP；如果出队列发生溢出，操作系统就要求服务器进程在继续发送报文之前要等待。

图 23.12 UDP中的队列



UDP的使用

- UDP适用于这样的进程，它需要简单的请求-响应通信，而较少考虑流量控制和差错控制；对于需要传送成块数据的进程（如FTP）则通常不使用它；
- UDP适用于具有内部流量控制和差错控制机制的进程，例如简单文件传输协议TFTP；
- 对多播来说，UDP是一个合适的传输协议；多播能力已被嵌入在UDP软件中，但没有嵌入在TCP软件中；
- UDP可用于管理进程，如SNMP；
- UDP可用于某些路由选择更新协议，如RIP。

23-3 TCP

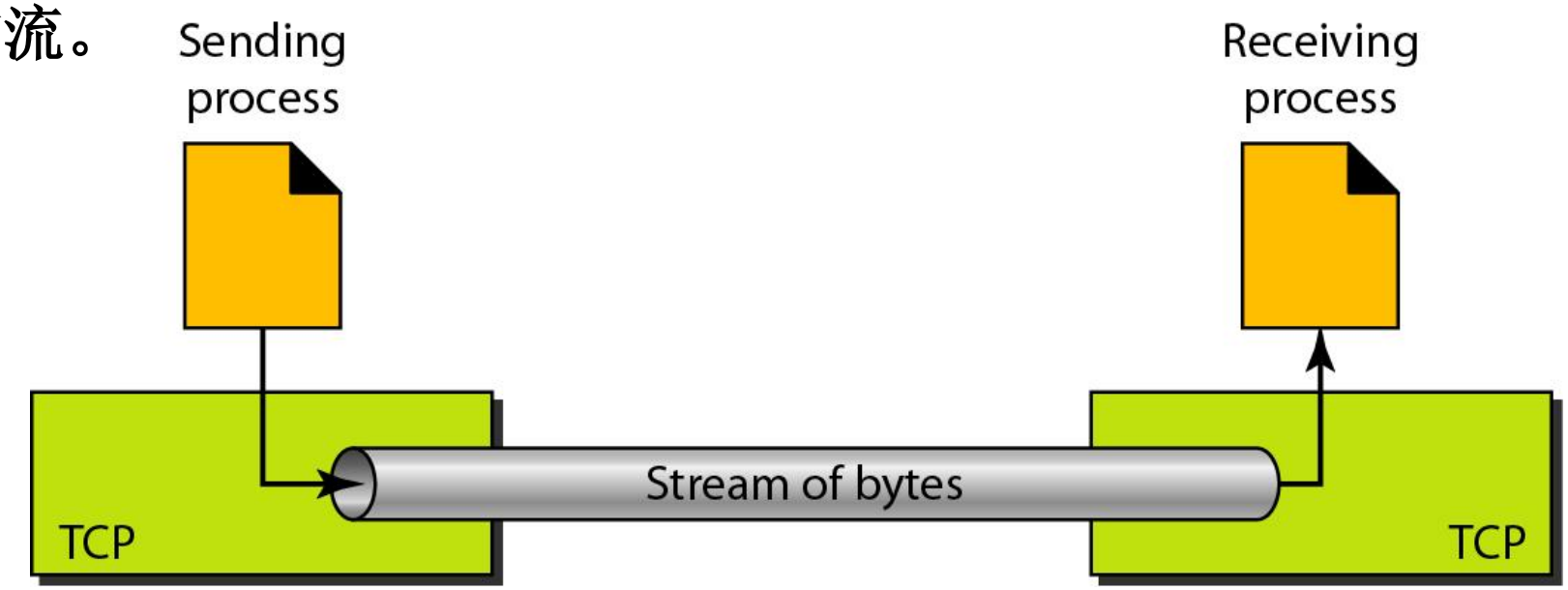
- TCP：Transmission Control Protocol；
- TCP是一个面向连接的协议，它在两个TCP端之间建立一个虚拟连接来发送数据；
- TCP是一个可靠的协议，在传输层使用流量控制和差错控制机制；
- TCP被称为面向连接的、可靠的传输协议，它为IP服务增加了面向连接和可靠性的特性；
- TCP提供的服务：
  - 进程到进程的通信；
  - 流传递服务；
  - 全双工通信；
  - 面向连接的服务；
  - 可靠的服务（使用确认机制）

表23.2 TCP使用的熟知端口

Port	*Protocol*	*Description*
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

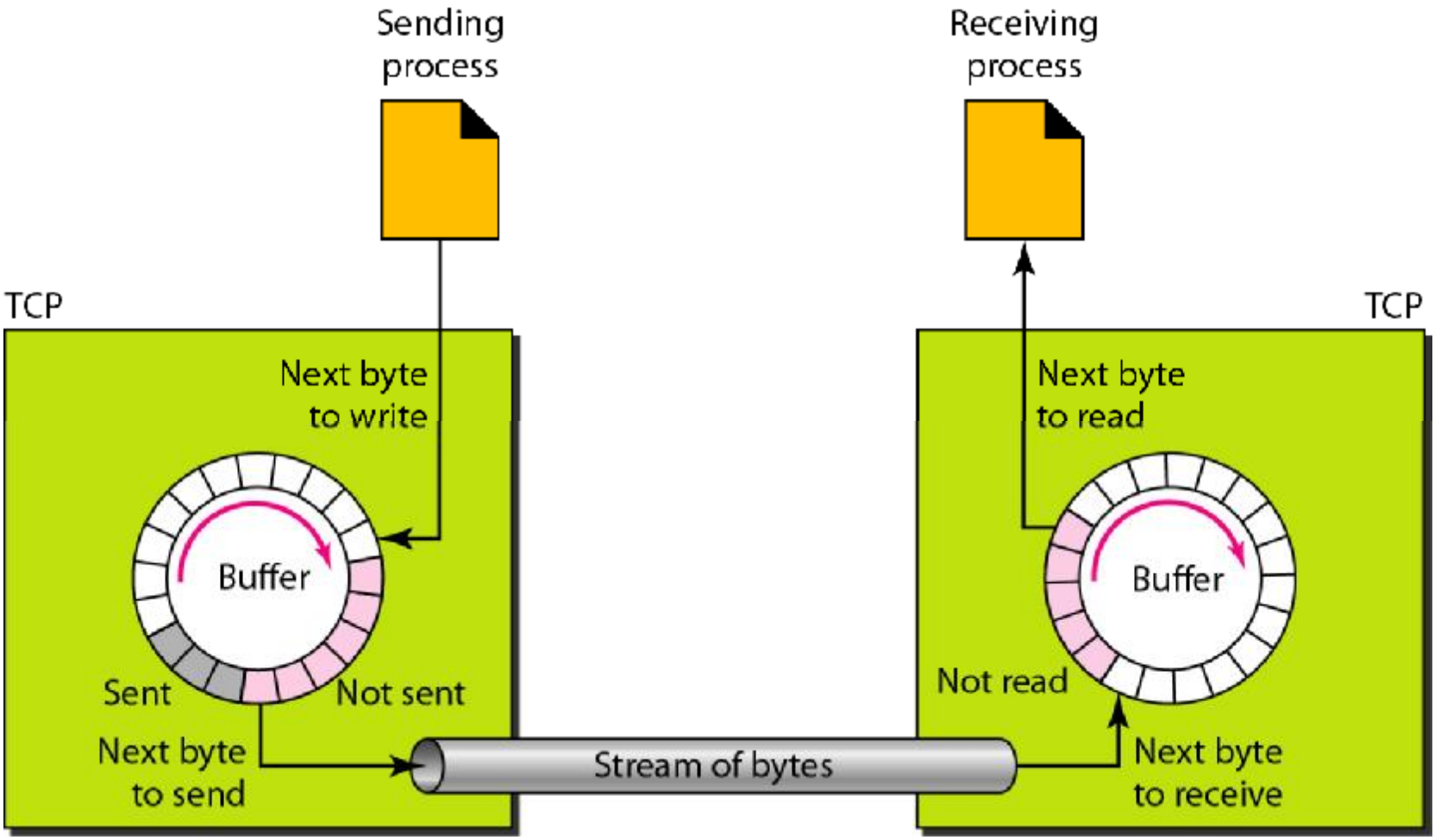
- 与UDP不同，TCP是一个面向流的协议；
- TCP允许发送进程以字节流形式传递数据，接收进程也以字节流形式接收数据；
- TCP建立一种环境，两个进程好像由一个假想的“管道”连接，这个管道通过因特网传送这些进程的数据；
- 发送进程产生（写入）字节流，接收进程消费（读出）这些字节流。



发送和接收缓冲区

- 在发送端，缓冲区有三种类型的存储单元：白色的部分是空存储单元，可以由发送进程（生产者）填充；灰色的部分用于保存已经发送但还没有得到确认的字节，TCP在缓冲区中保留这些字节，直到收到确认为止；灰色缓冲区是将要由TCP发送的字节；
- 接收端的缓冲区分成两个区域（表示为白色和灰色）；白色区域包含空存储单元，可以由从网络上接收的字节进行填充；灰色区域表示接收到的字节，可以由接收进程读出，当某个字节被接收进程读出以后，这个存储单元可被回收，并加入到空存储单元池中。

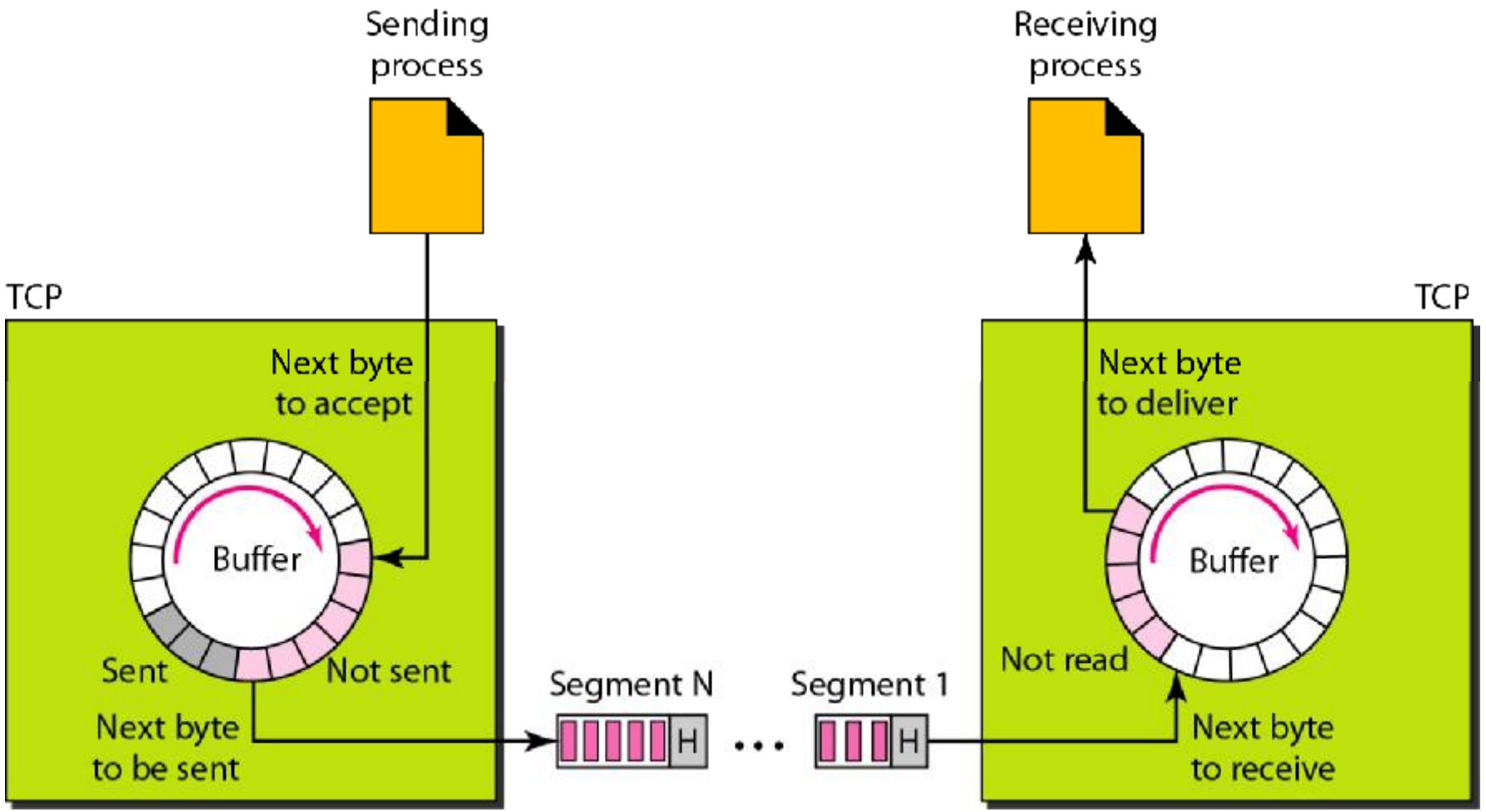
图23.14 发送和接收缓冲区



段

- 尽管缓冲能够处理生产进程速度和消费进程速度之间的不相称问题，但在发送数据之前，还需要多个步骤；
- IP层需要以分组而不是字节流的方式发送数据；
- 在传输层，TCP将多个字节组合在一起成为一个分组，这个分组称为段（segment）；
- TCP给每个段添加头部并传递给IP层，封装到IP数据报中，然后再进行传输；
- 整个操作对接收进程是透明的；
- 这些段可能被无序接收、丢失、或者损坏和重发，所有这些均由TCP处理，接收进程不会觉察到任何操作

图23.15 TCP段


Sending process
Receiving process
TCP
TCP
Next byte to accept
Next byte to deliver
Buffer
Buffer
Sent
Not sent
Not read
Segment N
Segment 1
H
H
Next byte to be sent
Next byte to receive

面向连接的服务

- 三个步骤：建立连接，在两个方向交换数据，连接终止；
- 注意：这是一个虚连接，而不是一个物理连接；
- TCP段封装成IP数据报，可能被无序地发送、或丢失或被破坏，然后重发；
- 每个段都可以通过不同的路径到达目的端；
- 虽然不存在物理连接，但TCP会建立一种面向字节流的环境，在这种环境中，TCP能承担按顺序传递这些字节到其他站点的任务；
- 这就好像建立了横跨多个岛屿的一座桥，以单一的连接从一个岛屿到另一个岛屿传送所有字节。

TCP特点

- 序号系统：虽然TCP软件能记录发送或接收的段，但在段头部没有段序号字段，TCP在段的头部采用称为序号（sequence number）和确认号（ack number）的两个字段，这两个字段指的是字节序号而不是段序号；
- 流量控制：数据的接收方控制发送方发送数据的数量，防止接收方数据溢出，序号系统允许使用面向字节的流量控制；
- 差错控制：虽然差错控制以段作为差错检测（丢失或损坏段）的数据单元，但它是面向字节的差错控制；
- 拥塞控制：TCP考虑网络中的拥塞，发送方发送的数据量不仅由接收方控制（流量控制），而且还要由网络中的拥塞程度决定。

TCP字节序号和序号

- TCP为在一个连接中传输的所有数据字节编号，在每个方向上序号都是独立的；
- 当TCP接收来自进程的一些数据字节时，将它们存储在发送缓冲区中时并给它们编号；
- 编号不必从0开始，TCP在0到$2^{32}-1$之间生成一个随机数作为第一个字节的序号；
- 字节被编号后，TCP对发送的每一个段分配一个序号，每个段的序号是这个段中的第一个字节的序号。

在每个连接中传送的字节都由TCP编号，序号开始于一个随机产生的数。

例23.3

假设一个TCP连接正在传送一个5000字节的文件，第一个字节序号是10001。如果数据被分成5个段，每一个数据段携带1000字节，试问每个段的序列号是什么?

解：

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)

一个段的序号字段的值定义为该段包含的第一个字节的序号。

p 当一个段携带数据和控制信息（捎带）时，它使用一个序号；

p 如果一个段没有携带用户数据，那么它逻辑上不定义序号，虽然字段存在，但值是无效的；

p 然而，有些段当仅携带控制信息时也需要有一个序号用于接收方的确认，这些段用做连接建立、连接终止或连接废弃；

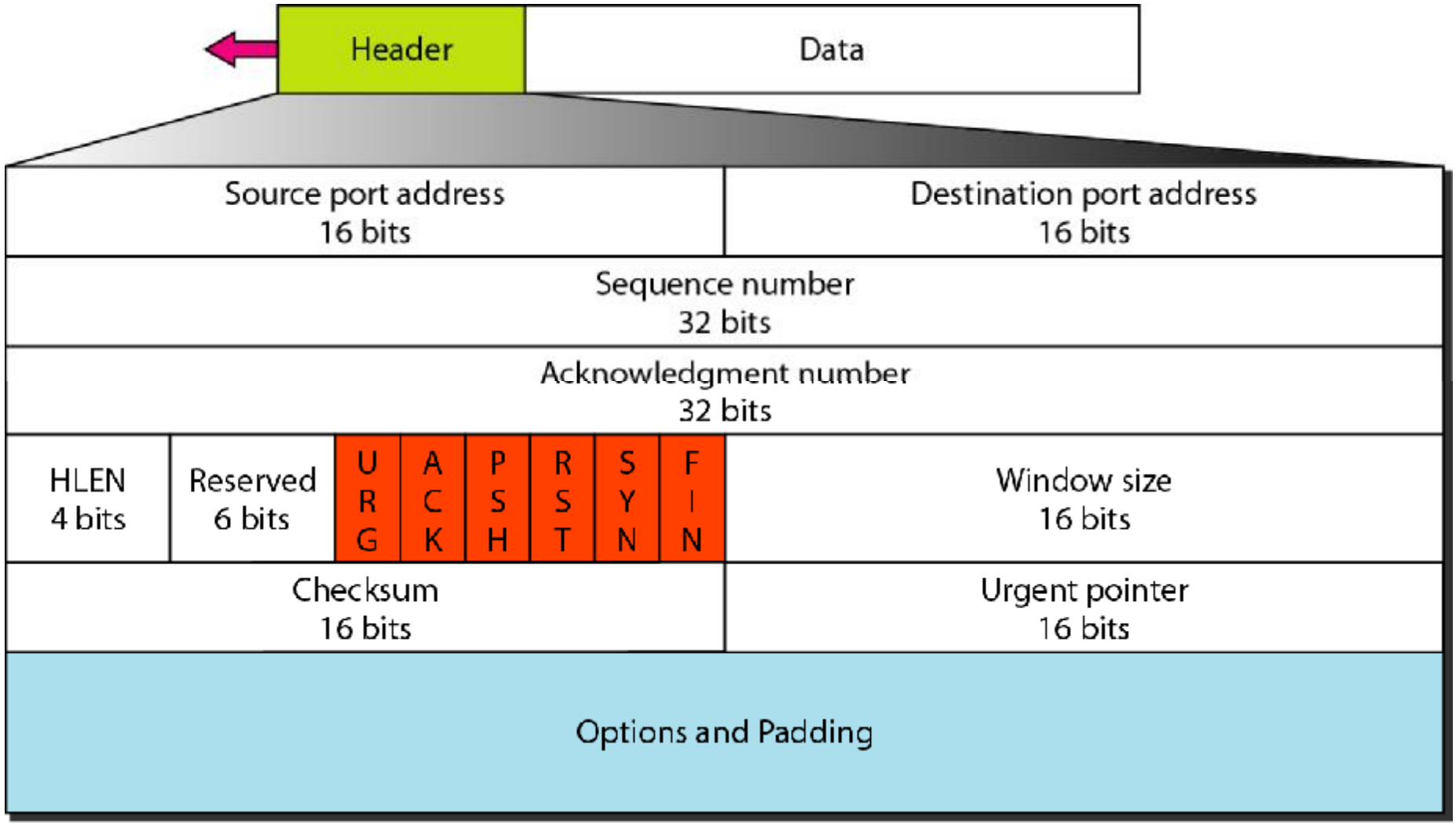
p 这些段中的每一个好像携带一个字节那样使用一个序号，但都没有实际的数据；如果随机产生的序号是x，则第一个数据字节的编号是x+1，字节x被认为是一个控制段打开一个连接的一个伪字节

确认号

- TCP中的通信是全双工的，当建立一个连接时，双方同时都能发送和接收数据；
- 每一方为字节编号，每一方向的序号表明了该段所携带的第一个字节的序号，每方也使用确认号来确认它已收到的字节，确认号定义了该方预期接收的下一个字节的序号；
- 确认号是累积的，这意味接收方记下它已安全而且正确地接收到的最后一个字节的序号，然后将它加1，并通告这个结果作为确认号：
- 术语“累积”指的是：如果一方使用5643作为确认号，则表示它已经接收了所有从开始到序号为5642的字节；但要注意，这并不是指接收方已经接收了5642个字节，因为第一个字节的编号通常并不是从0开始的。

段中确认字段的值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。

图23.16 TCP分组（称为段，segment）格式



TCP段头部字段

- 头部：固定20个字节+选项，最多60个字节；
- 源端口地址：16位，指发送该段的应用程序的端口号；
- 目的端口地址：16位，指接收该段的应用程序的端口号；
- 序列号：32位，分配给段中数据的第一个字节；对要发送的每一个字节都进行编号，序列号告诉目的端，在这个序列中哪一个字节是第一个字节；在连接建立时，每一方都使用随机数生成器产生一个初始序列号ISN，通常每一个方向的ISN都不同；
- 确认号：32位，定义了段的接收方期望从对方接收的字节号，确认和数据可捎带一起发送；
- 头部长度：4位，以4字节为单位进行计数；
- 保留：6位，保留为将来使用；
- 控制：定义了6种不同的控制位或标记，同一时间可设置一位或多位，这些控制位用在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

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

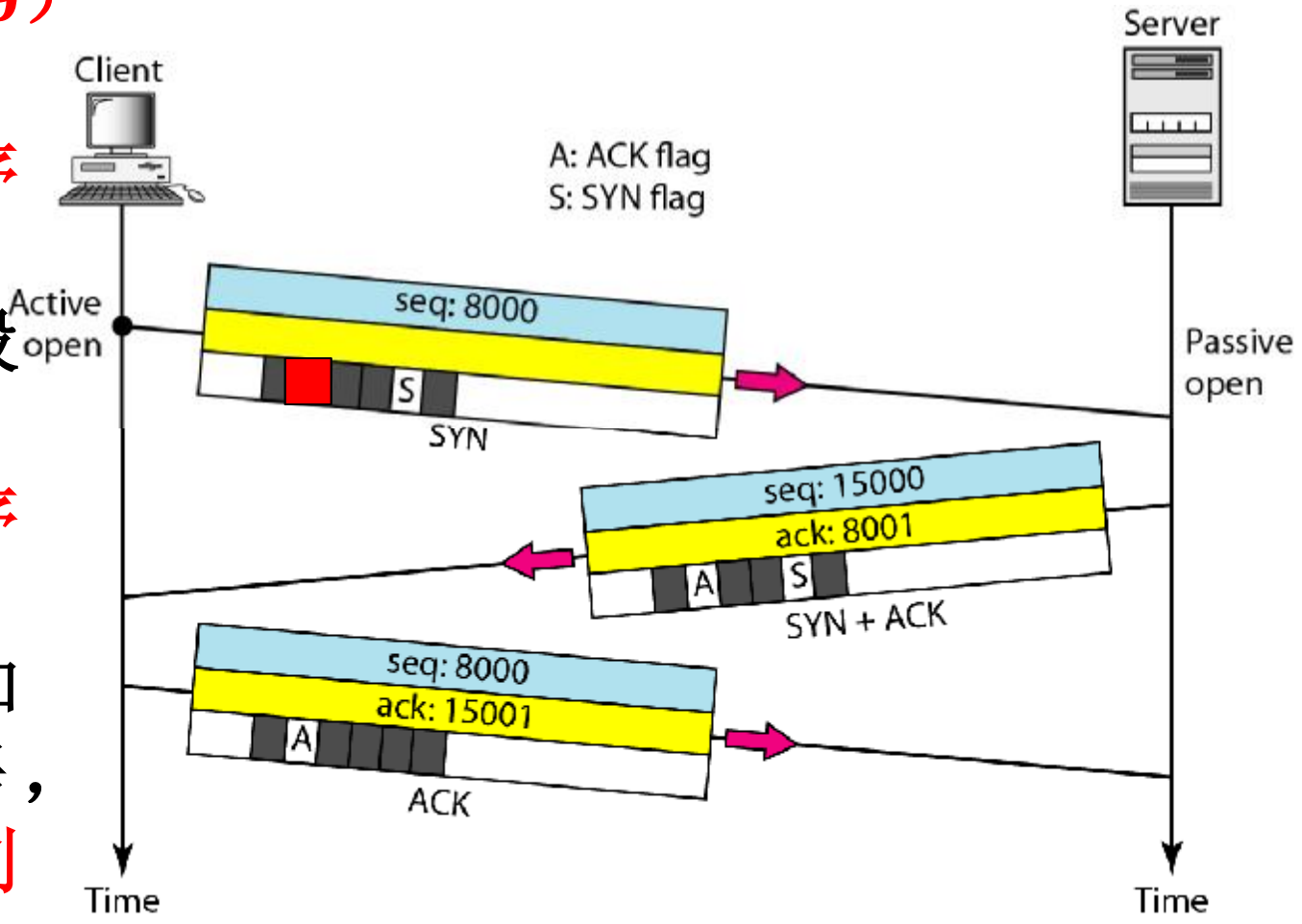
Flag	*Description*
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

TCP段头部字段（cont.）

- 窗口大小：定义对方必须维持的窗口的大小（以字节为单位）；16位，意味着窗口最大长度是65535字节，这个值通常称为接收窗口（rwnd），由接收方确定，发送方必须服从接收端的支配；
- 校验和：16位，计算时需要伪头部（协议字段的值是6），对于TCP，校验和是强制的；
- 紧急指示符：16位，只有当紧急标志置位时才有效，说明这个段包含了紧急数据；定义一个数，将此数加到序列号上就得出此段数据部分中最后一个紧急字节；
- 选项：可以有多达40字节的可选信息；TCP最常见的选项为最大报文段长度MSS，告诉对方TCP：我的缓存所能接收报文段的数据部分的最大长度是MSS个字节

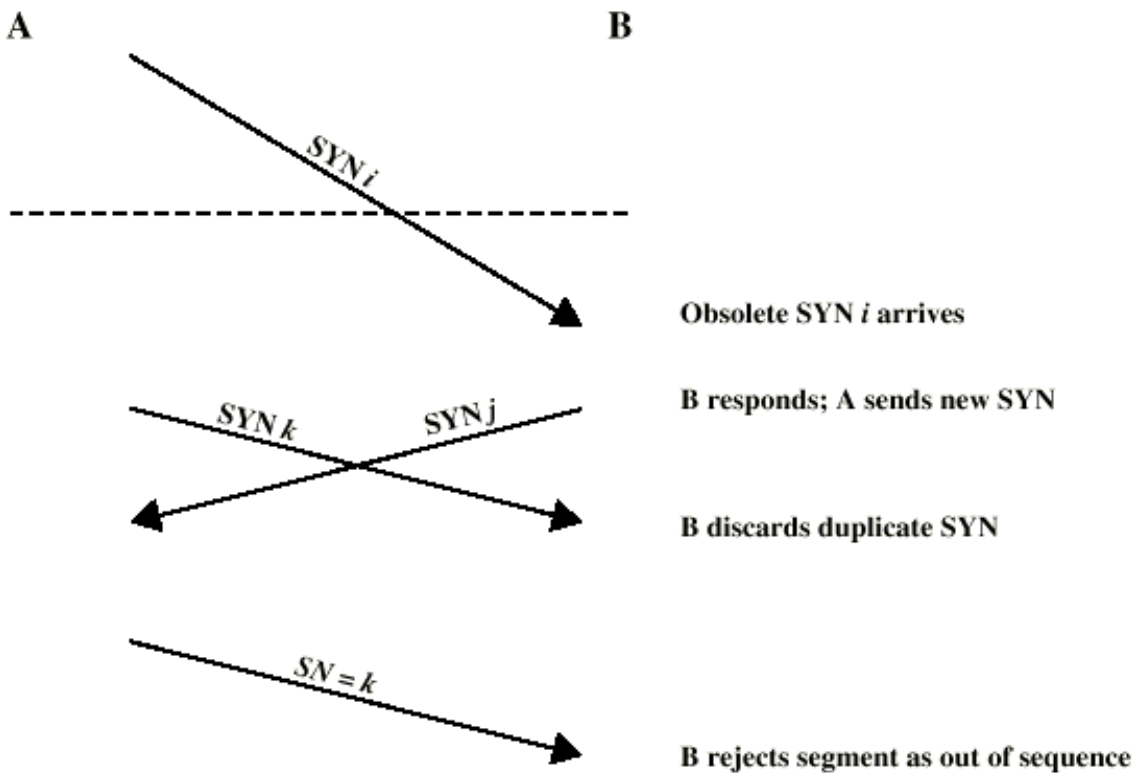
图23.18 使用三次握手建立连接

- SYN段（P479）不携带数据，但占用一个序列号；
- SYN+ACK 段不携带数据，但占用一个序列号；
- ACK 段，如果不携带数据，则不占用序列号。



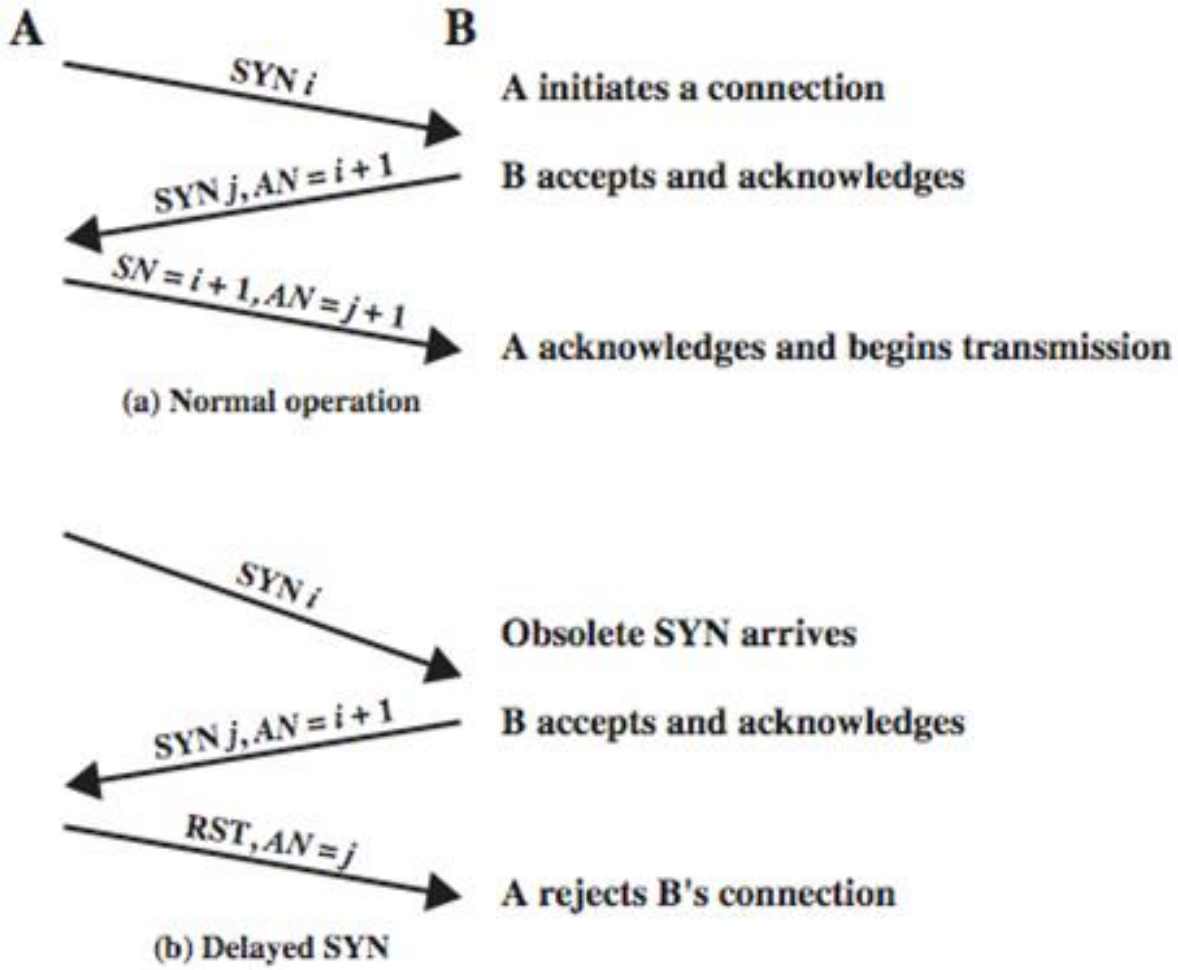
为什么不能使用两次握手（A发送SYN i，B以SYN j响应）建立连接？有什么问题？

- 连接终止后，旧的SYN i到达，B认为是一个新请求以SYN j响应；
- 同时，A打开一条与B的新连接，并发送SYN k，它被B作为副本丢弃；
- 双方都已传输了SYN，并在之后都收到了对方的SYN，认为连接有效；
- 然而，当A以编号k传输报文段时，B却因为序号超出范围而拒绝接收



三次握手不存在延迟SYN i带来的问题

p图(b)中，如果延迟的SYN i到达，A会拒绝B发来的SYN+ACK（reset连接），因为与它期待的ACK序号不符（假设A刚发送了SYN k，它期待B发来的ACK是k+1，现在得到的却是i+1）

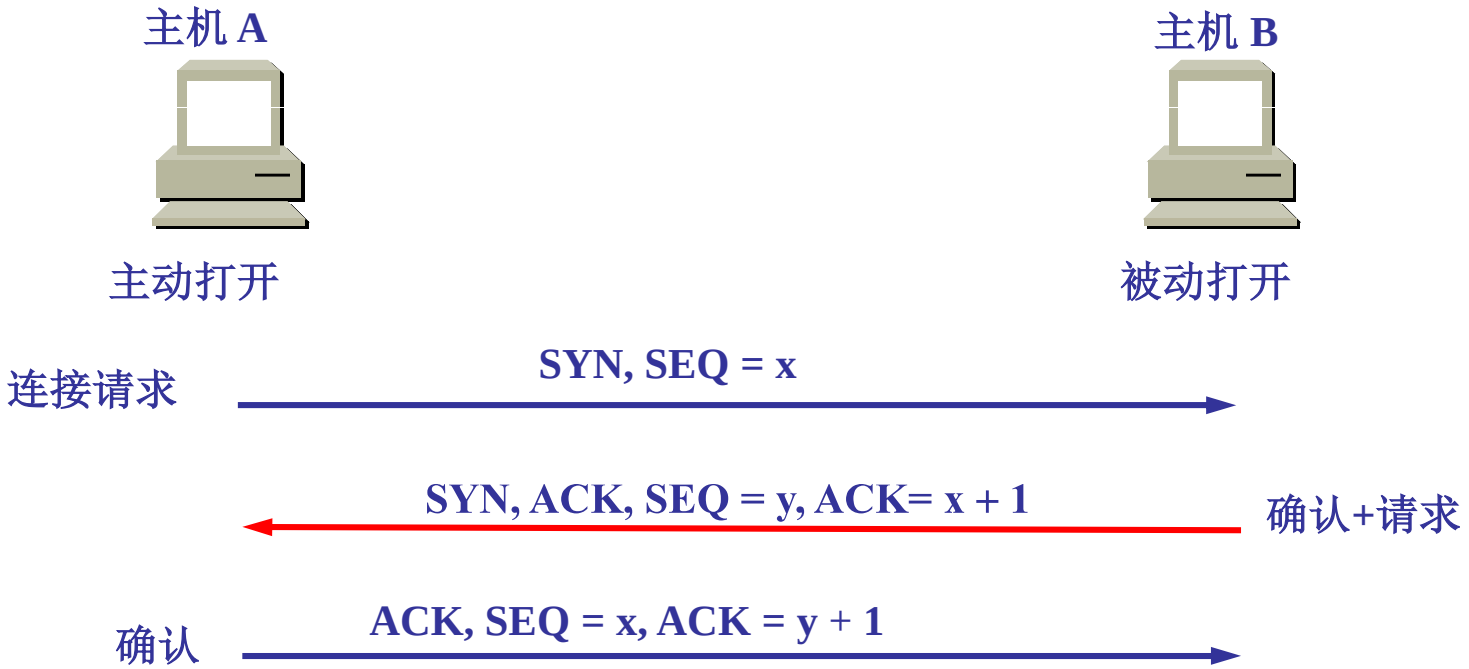


练习题

p简述TCP协议采用三次握手建立连接的原因，并画出正常的操作过程图示。

解：

三次握手解决了两次握手中延迟SYN所造成的错误。



SYN洪泛（或泛滥）攻击

- 攻击者将大量SYN段发送到一个服务器，在数据报中通过伪装源IP地址假装这些段来自不同的客户端；
- 客户机发出主动打开，服务器分配必要的资源，如生成通信表和设置计时器等；然后服务器送SYN+ACK段给这些假装客户，但这些段都丢失了；
- 而这段期间许多资源被占用但没有被使用，如果短时间内SYN段数量很大，服务器最终耗尽它的资源而崩溃；
- SYN洪泛攻击属于拒绝服务攻击（denial-of-service attack）类型，此时一个攻击者独占系统如此多的服务请求使得系统崩溃，拒绝对每个请求提供服务；
- 减轻SYN攻击影响的策略：（1）特定时间周期内对连接请求进行限制；（2）过滤掉来自不需要的源地址的数据报；（3）使用cookie（P492）推迟资源分配直到一个完整的连接建立。

图23.19 数据传输

p急迫数据：通常接收方的TCP在数据到达时将数据缓存，但有时希望接收到立即响应；TCP用PSH位告诉接收端，这个段所包括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；

p紧急数据：用URG标志+紧急指针实现，当接收端的TCP接收到URG位置1的段时，它就利用紧急指针的值从段中提取出紧急数据，并不按序地将其传递给接收应用程序，比如Ctrl+C异常终止命令。

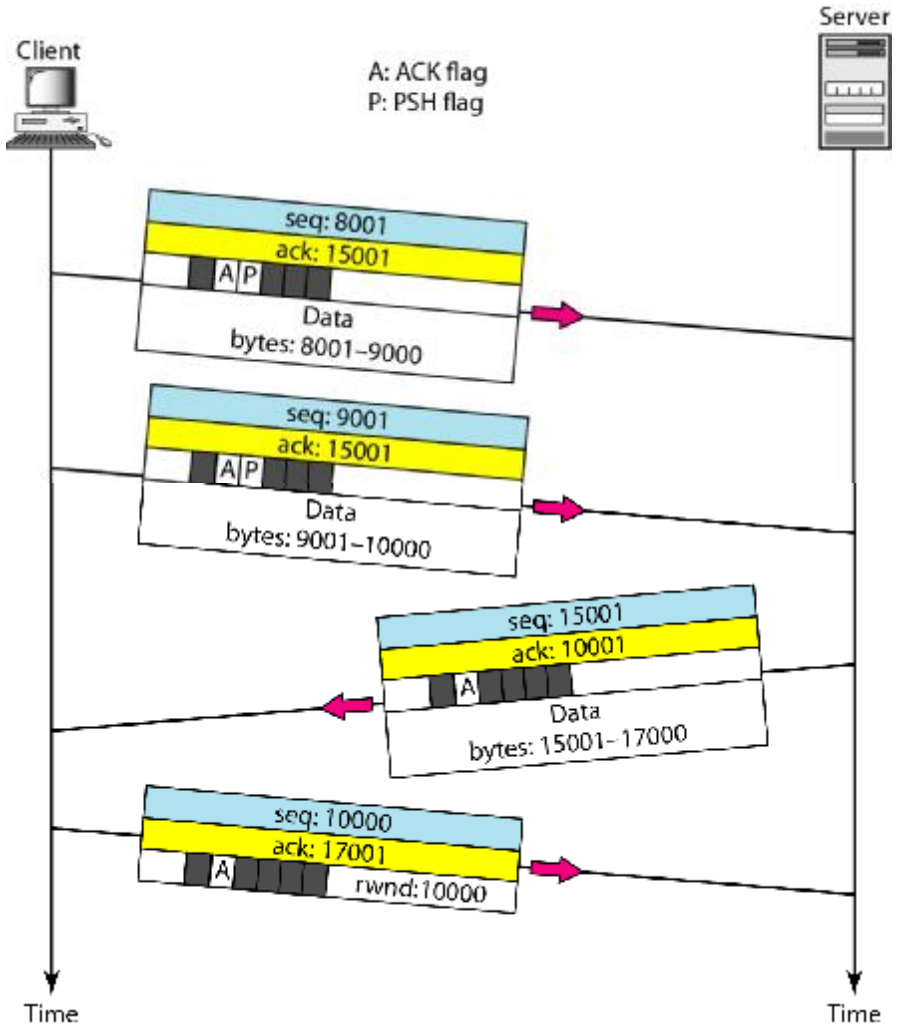


图23.20 TCP终止连接方法一：三次握手

- 如果FIN段不携带数据（它可包含客户机要发送的最后数据块），则该段占用一个序列号；
- 如果FIN+ACK段没有携带数据（它可包含来自服务器的最后数据块），则该段仅占用一个序列号；
- ACK段包含确认号，不携带数据也不占用序列号

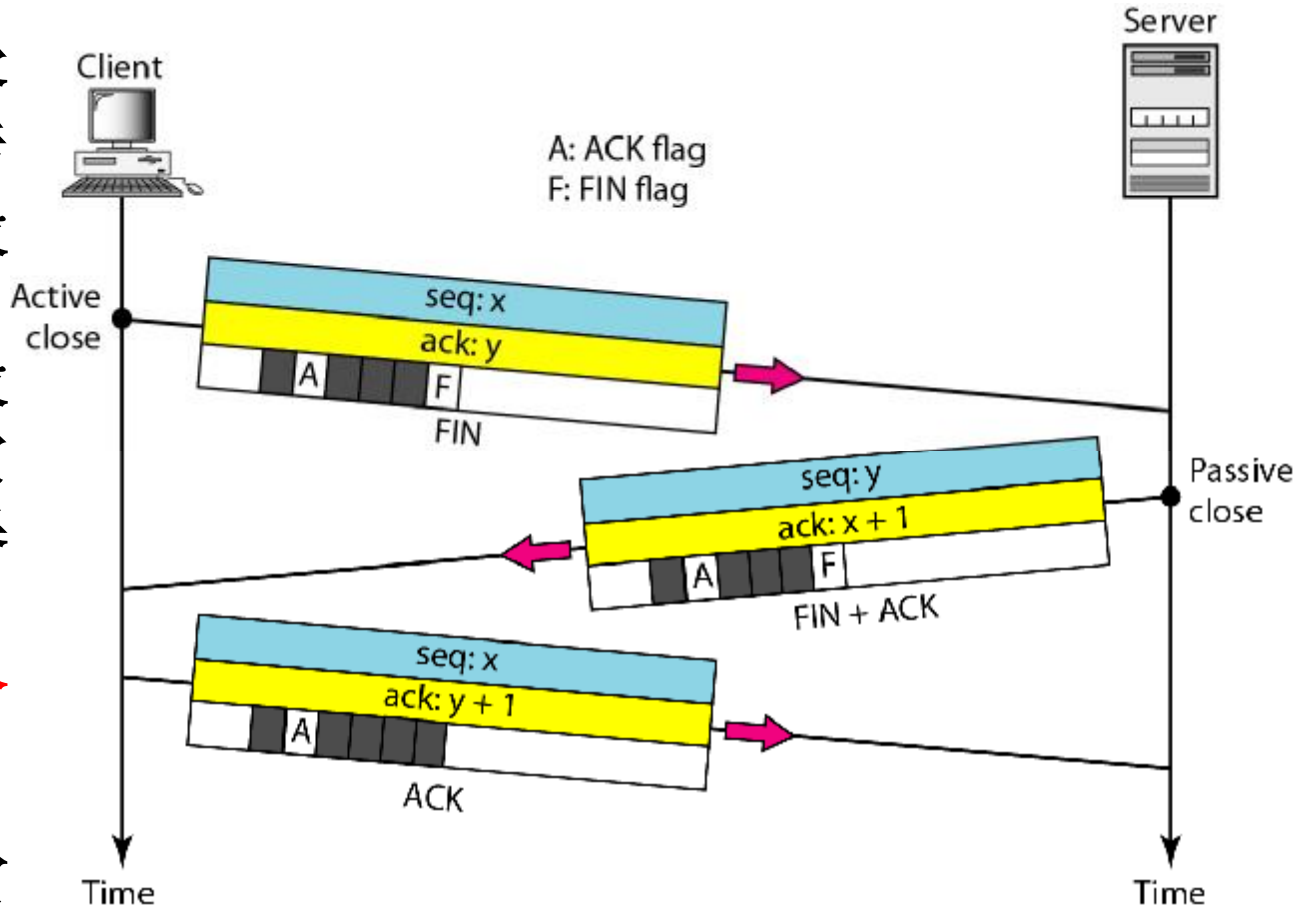
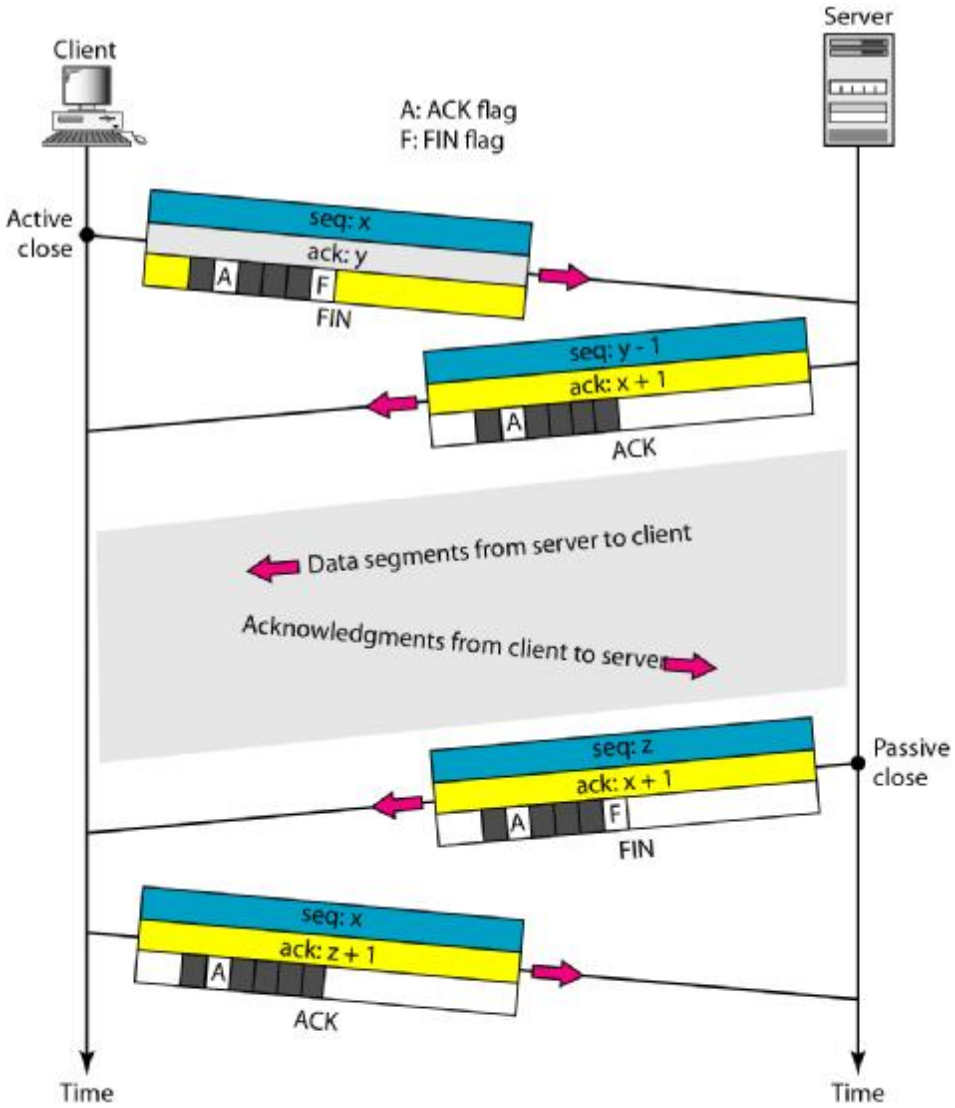


图23.21 TCP终止连接方法二：带有半关闭选项的四次握手（大多数常用方法）

- 一端停止发送数据后，还可以继续接收数据（半关闭）；
- 通常由客户端发起，服务器开始处理之前需要所有数据时，通常使用半关闭，例如排序；
- 开始排序之前，服务器需要接收到全部数据，所以客户端发送完全部数据之后，它在向外方向可连接可关闭；
- 但在向内方向，为了接收排序结果必须保持打开，服务器接收到数据后还需要时间进行排序，它的向外方向保持打开
- 注意：理解seq y-1和seq x



TCP流量控制

- TCP使用滑动窗口处理流量控制，它使用的滑动窗口协议界于回退N帧与选择重发之间；
- TCP不使用NAK，像回退N帧协议；TCP接收方保存失序的段直到丢失的段到达，又像选择重发；
- TCP滑动窗口与数据链路滑动窗口有两大点不同：
  - 第一，TCP的滑动窗口是面向字节的，而数据链路层滑动窗口是面向帧的；
  - 第二，TCP的滑动窗口是可变大小，而数据链路层的滑动窗口是固定大小
- TCP滑动窗口机制又被称为信用量机制或信贷滑窗协议（数据链路层的滑动窗口称为固定的滑动窗口协议）

传输层流量控制可选择的四种方法

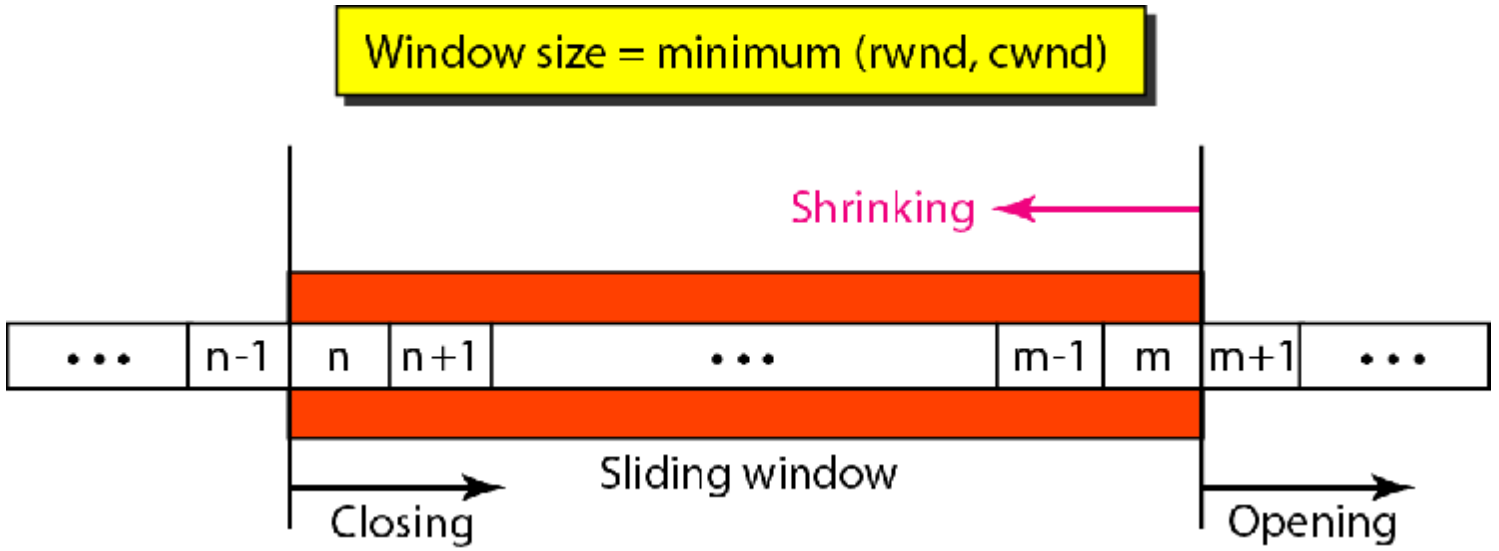
- 接收方什么也不做（溢出缓冲区的报文将被丢弃）；
- 接收方拒绝后续的来自网络服务的报文段（是一种反压机制，抑制发送端的网络服务，笨拙且粗糙）；
- 使用固定的滑动窗口机制（就像数据链路层那样）；
- 使用信用量机制
  - Ø传输数据的每个字节都有一个序号（调控的粒度更细）；
  - Ø每个传输的报文段在它的首部中包括三个与流量控制相关的字段：序号、确认号和窗口；
  - Ø窗口是动态变化的，与信用量紧密相关（更灵活，可以与拥塞控制关联）

TCP滑动窗口

- 窗口覆盖了从进程接收到的字节的部分缓冲区，窗口内字节都是传输中的字节，可以发送它们而不必考虑来自另一端的确认；
- 窗口的活动（张开、合拢或收缩）由接收方控制（同时依赖于网络中的拥塞）而不是发送方，发送方必须服从接收方的命令；
- 窗口张开是指右边沿向右移动，它将允许缓存区中存储更新的字节；窗口合拢是指左边沿向右移动，表示有些字节被确认；
- 窗口不能收缩（即右边沿向左移动）；
- （发送）一端的TCP窗口大小取决于两个值中较小的一个：接收方通告窗口（rwnd）值和拥塞窗口（cwnd）；
- 接收方通告窗口值是在另一端的确认段中宣布的值，它就是另一端在它的缓存溢出和数据被丢弃之前能够接收的字节个数；
- 拥塞窗口值是网络防止拥塞所确定的值（后面讨论）

图23.22 TCP滑动窗口

- 使用滑动窗口可使传输更加有效，同时也可以控制数据流，使得目的端不致因数据来的过多而瘫痪；
- TCP的滑动窗口是面向字节的（给每个字节编号）



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

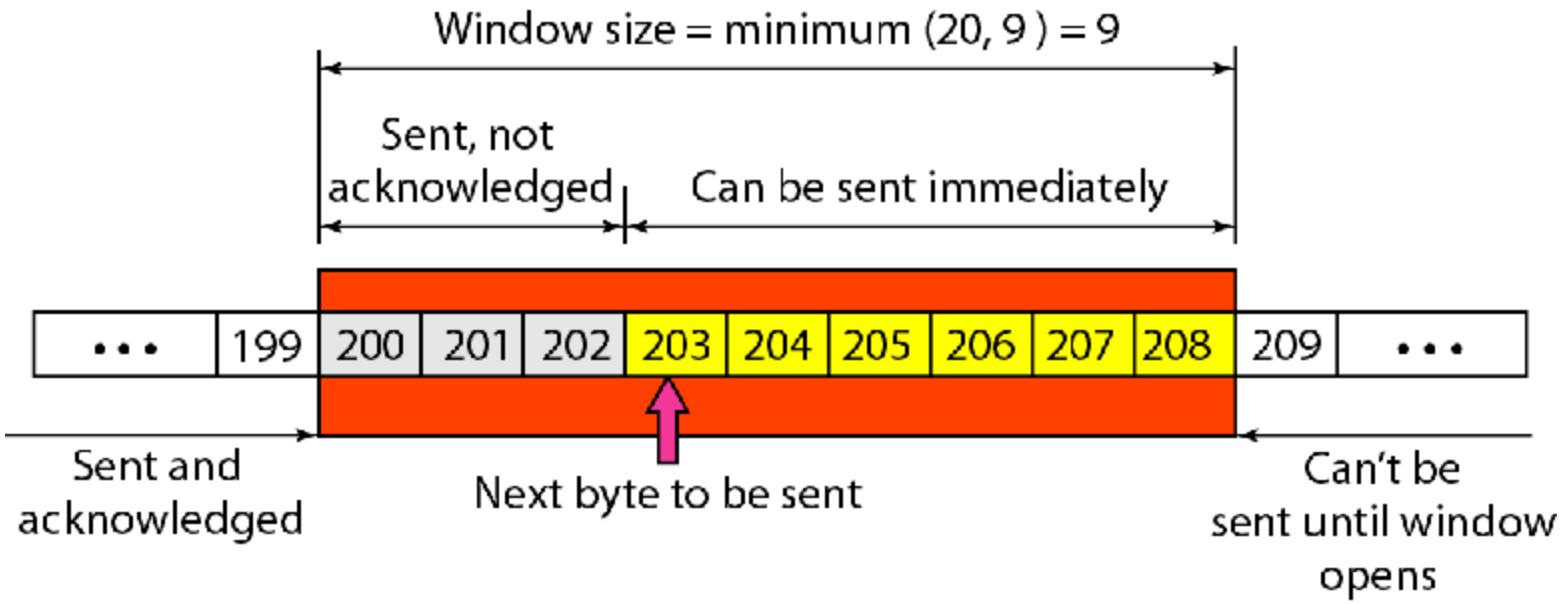
如果rwnd的值是3000个字节，cwnd的值是3500个字节，试问主机A的窗口大小是多少？

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表示了一个滑动窗口，发送方已发送了202个字节。我们假定cwnd是20，接收方已经发送一个rwnd为9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。



TCP滑动窗口要点：

- 窗口大小是rwnd和cwnd中的最小值。
- 发送方不必发送一个全窗口大小的数据（可分片）。
- 接收方可张开或合拢窗口，但不能收缩窗口。
- 只要不引起窗口收缩，目的方可随时发送一个确认。
- 接收方可暂时关闭窗口，但在窗口关闭后发送方总是发送一个1字节的段（发送方用来探测窗口大小，以防止接收方窗口通知报文丢失）。

TCP差错控制

- TCP使用差错控制提供可靠性，差错控制包括：检测受到损坏的段、丢失的段、失序的段和重复的段，还包括检测出差错后纠正差错的机制；
- TCP中的差错检测和纠正通过三种手段完成：校验和、确认和超时；
- 校验和用来检查受到损坏的段；
- TCP使用确认方法来证实收到了数据段，不携带数据但占用序列号的一些控制段也要确认，但ACK段不需要确认；
- 差错控制机制的核心是段的重传，当段损坏、丢失或延迟时就重传这个段（ACK段不重传，不设置重传计时器）；
- 在当前实现中有两种情况要重传段：重传计时器到时或当发送方收到三个重复的ACK时

RTO（retransmission time-out）后重传

- TCP对所有未完成的段（outstanding-P483，已发送但还未被确认）的最新实现使用了一个重传超时（RTO）计时器；
- 当计时器到时，即使可能是由于段被延迟、ACK被延迟或丢失确认等未接收到ACK时，重发一个最早的outstanding段；
- RTO的值是动态的，根据段的往返时间RTT更新（RTT是一个段到达目的端并接收到一个确认所需要的时间）

三个重复ACK 段之后的重发

- 如果RTT的值不是很大，关于段重发的先前的规则是可用的；
- 但有时某一段丢失了，而接收端收到了许多失序的段，它们不可能都被存储（有限的存储器大小）；
- 为了缓解这种情况，当今大多数实现遵循三次重复ACK规则，立即重发缺少的段，这一特点称为快速重传（fast retransmission）

数据可以失序到达，并被接收的TCP暂时存储；但是TCP确保传递给进程的段是无失序的。

图23.24 正常操作

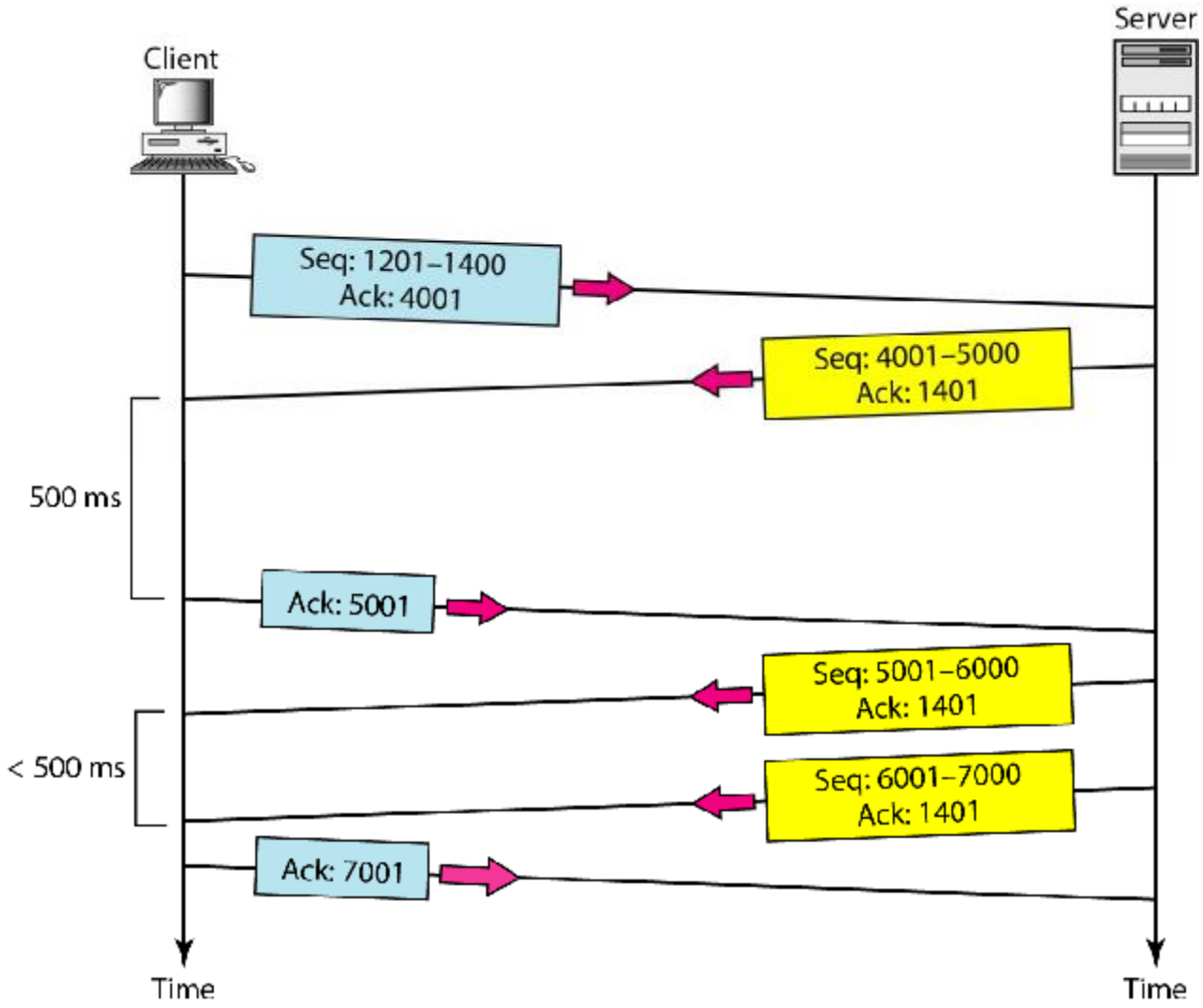
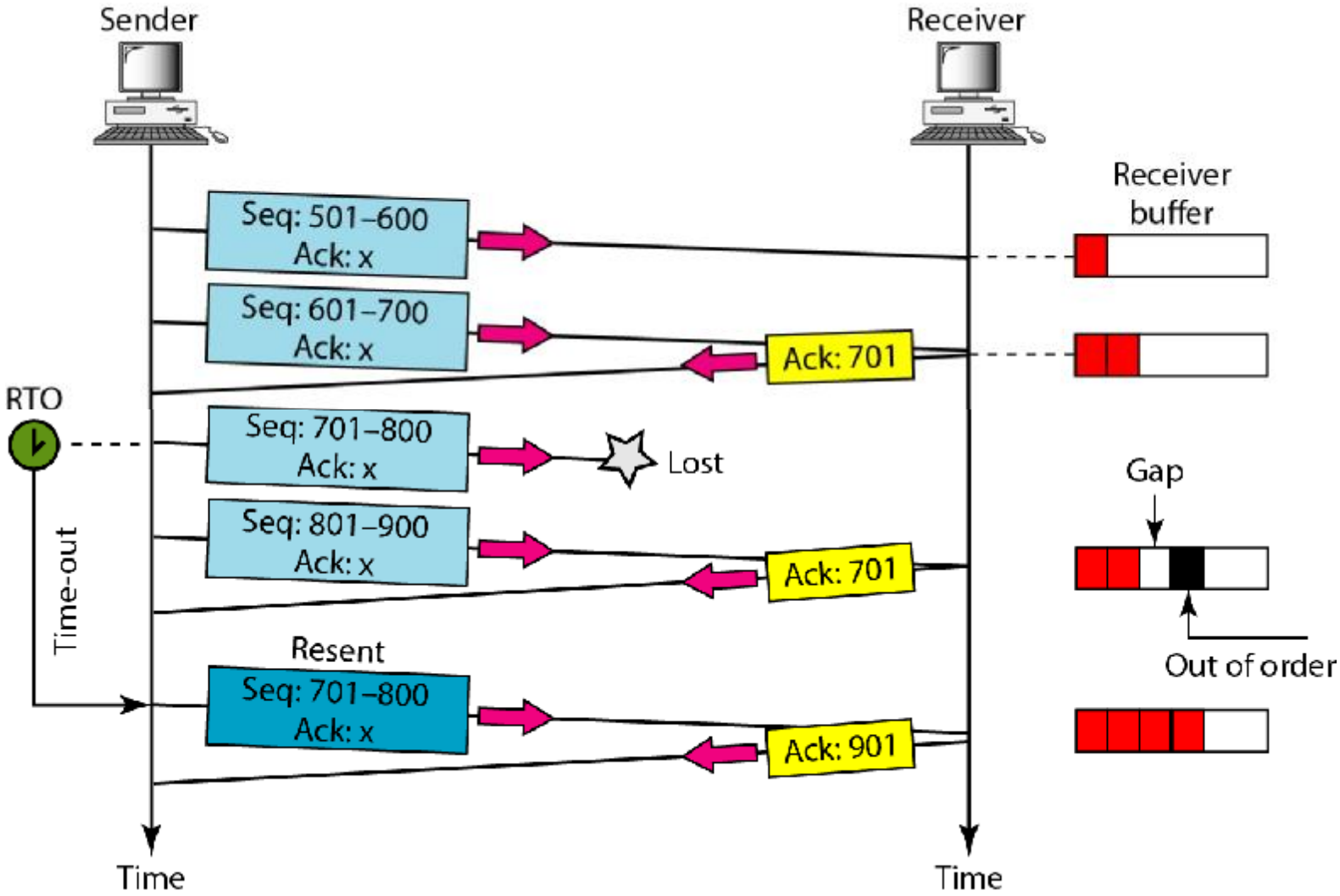
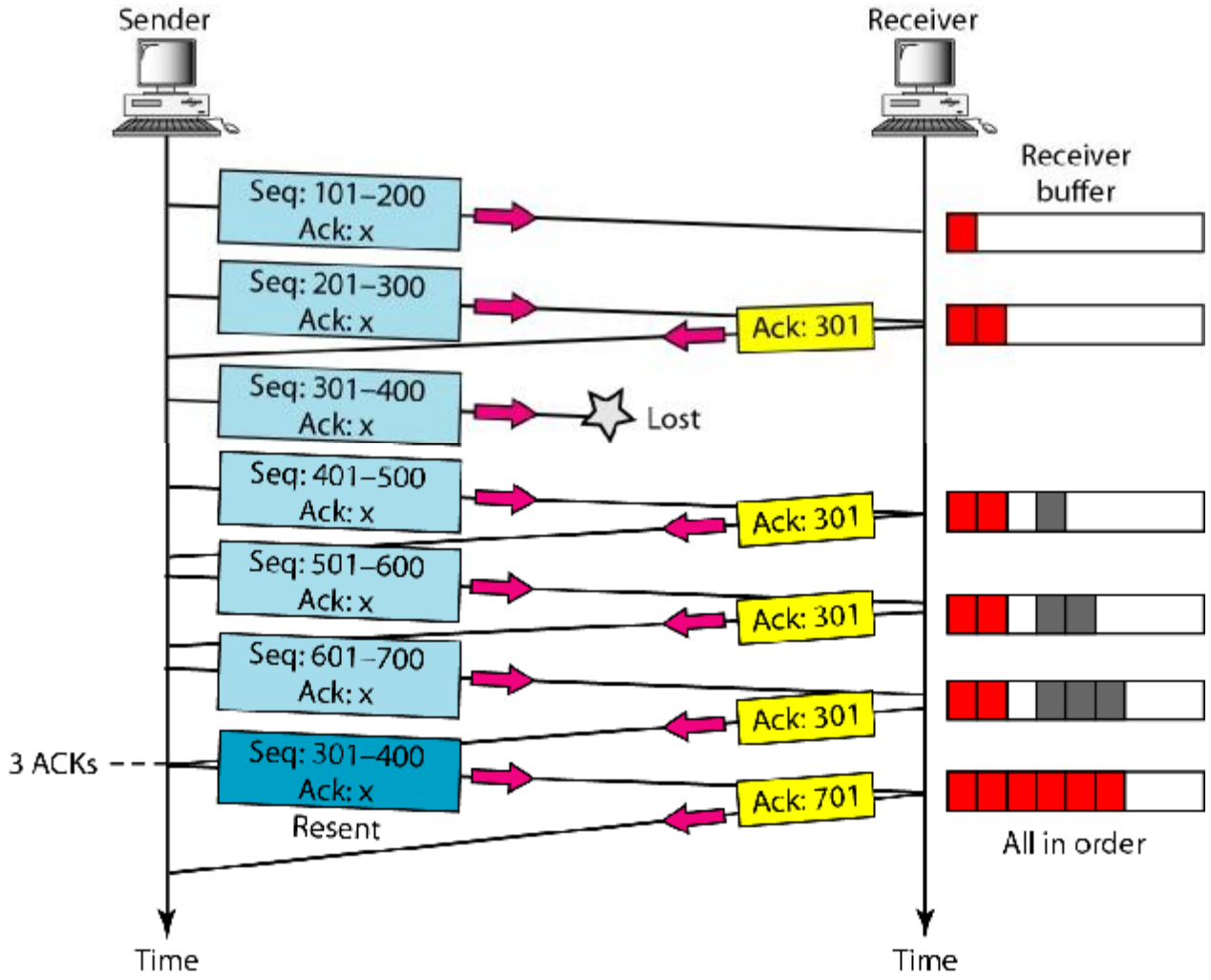

Client
Server
Seq: 1201–1400
Ack: 4001
Seq: 4001–5000
Ack: 1401
500 ms
Ack: 5001
Seq: 5001–6000
Ack: 1401
< 500 ms
Seq: 6001–7000
Ack: 1401
Ack: 7001
Time
Time

图23.25 丢失段


Sender
Receiver
Receiver buffer
Seq: 501–600
Ack: x
Seq: 601–700
Ack: x
Ack: 701
RTO
Seq: 701–800
Ack: x
Lost
Gap
Seq: 801–900
Ack: x
Ack: 701
Time-out
Resent
Seq: 701–800
Ack: x
Out of order
Ack: 901
Time
Time

接收方TCP仅将有序的数据传递给进程。

图23.26 快速重传


Sender
Receiver
Receiver buffer
Seq: 101–200
Ack: x
Seq: 201–300
Ack: x
Ack: 301
Seq: 301–400
Ack: x
Lost
Seq: 401–500
Ack: x
Ack: 301
Seq: 501–600
Ack: x
Ack: 301
Seq: 601–700
Ack: x
Ack: 301
3 ACKs
Seq: 301–400
Ack: x
Resent
Ack: 701
All in order
Time
Time

UDP、TCP和SCTP的比较

- UDP是一个面向报文的协议，进程将报文传递给UDP，将它封装在一个用户数据报中，通过网络发送；UDP保留报文边界，每个报文与其他报文无关；UDP不可靠，发送方不知道报文的命运，报文可能会丢失、重复或失序；UDP还缺乏其他特性，如缺少友好的传输层协议所需要的拥塞控制和流量控制；
- TCP是一个面向字节的协议，它从进程中接收一个报文或多个报文，并以一个字节流的方式存储它们，以段的方式发送它们；没有保存报文的边界；TCP可靠，重复段会被删除，丢失的段会重发，并且字节按序传递到端进程，TCP还有拥塞控制和流量控制机制；
- SCTP兼有UDP和TCP的特性，是一个可靠的面向报文的协议，它保存报文的边界，同时检测丢失的数据、重复的数据和失序的数据；它还有拥塞控制和流量控制机制。