第3章:运输层

目的:

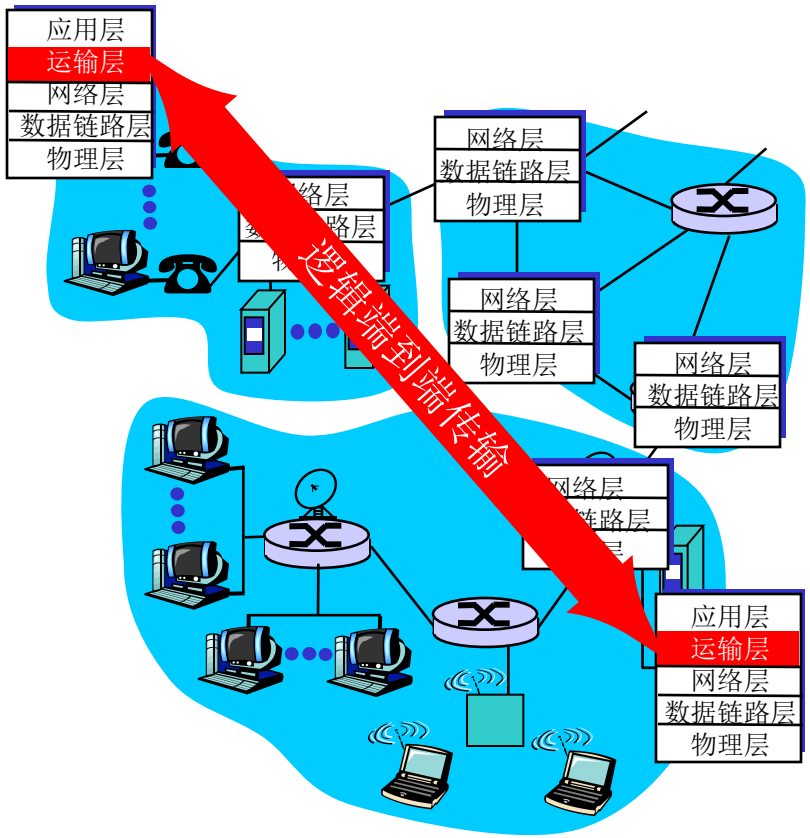
- 理解运输层服务依据的原理:
  - 复用/分解
  - 可靠数据传输
  - 流量控制
  - 拥塞控制
- 学习因特网中的运输层协议:
  - UDP: 无连接传输
  - TCP: 面向连接传输
  - TCP 拥塞控制

第3章 主要内容

- 3.1 运输层服务
- 3.2 多路复用与多路分解
- 3.3 无连接传输: UDP
- 3.4 可靠数据传输原理
  - rdt1
  - rdt2
  - rdt3
  - 流水线协议
- 3.5 面向连接的传输: TCP
  - 报文段结构
  - 可靠数据传输
  - 流量控制
  - 连接管理
- 3.6 拥塞控制原理
- 3.7 TCP拥塞控制
  - 机制
  - TCP吞吐量
  - TCP公平性

3.1 概述和运输服务

- 在运行不同主机上应用进程之间提供*逻辑通信*
- 运输协议运行在端系统中
  - 发送方：将应用报文划分为*段*，传向网络层
  - 接收方：将段重新装配为报文，传向应用层
- 应用可供使用的运输协议不止一个
  - 因特网：TCP和UDP



运输层 vs. 网络层

- *网络层:* 主机间的逻辑通信
- *运输层:* 进程间的逻辑通信
  - 依赖、强化网络层服务

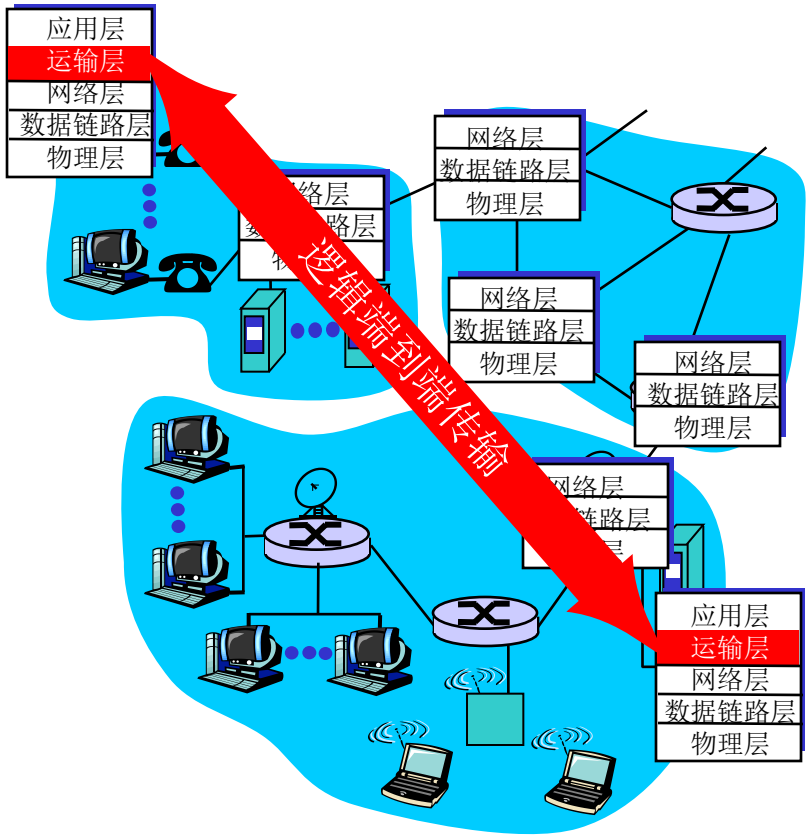
家庭类比:

12个孩子向12个孩子发信

- 进程 = 孩子
- 应用报文= 信封中的信
- 主机 = 家庭
- 运输协议 = Ann和Bill
- 网络层协议= 邮政服务

因特网运输层协议

- 可靠的、按序的交付(TCP)
  - 拥塞控制
  - 流量控制
  - 连接建立
- 不可靠、不按序交付: UDP
  - “尽力而为”IP的不提供不必要服务的扩展
- 不可用的服务:
  - 时延保证
  - 带宽保证



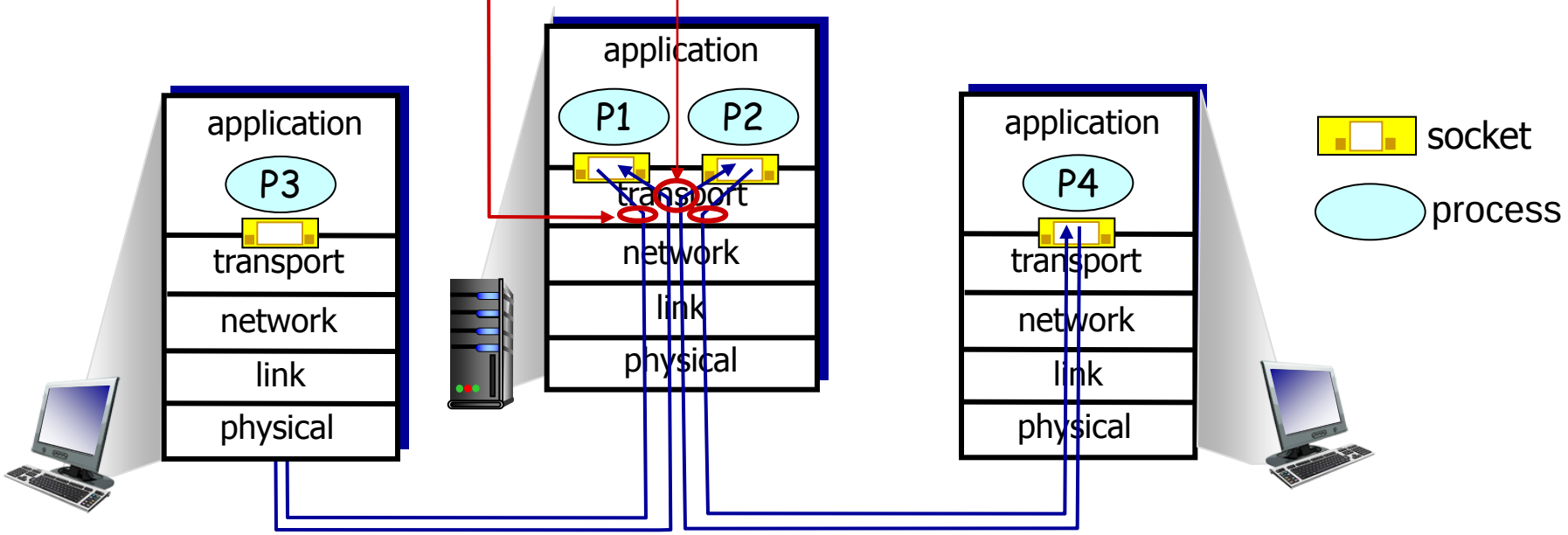
第3章 主要内容

- 3.1 运输层服务
- 3.2 复用与分解
- 3.3 无连接传输: UDP
- 3.4 可靠数据传输的原则
  - rdt1
  - rdt2
  - rdt3
  - 流水线协议
- 3.5 面向连接的传输: TCP
  - 报文段结构
  - 可靠数据传输
  - 流量控制
  - 连接管理
- 3.6 拥塞控制的原则
- 3.7 TCP拥塞控制
  - 机制
  - TCP吞吐量
  - TCP公平性
  - 时延模型

3. 2 多路复用/多路分解

multiplexing at sender:
handle data from multiple sockets, add transport header (later used for demultiplexing)

demultiplexing at receiver:
use header info to deliver received segments to correct socket



分解工作过程

- 主机接收IP数据报
  - 每个数据报承载1个运输层段
  - 每个段具有源、目的端口号
    (回想: 对特定应用程序的周知端口号)
- 主机使用IP地址 &端口号将段定向到适当的套接字

TCP/UDP 段格式

无连接分解

- 生成具有端口号的套接字:

```
DatagramSocket mySocket1 = new
   DatagramSocket(99111);
DatagramSocket mySocket2 = new
   DatagramSocket(99222);
```

- UDP套接字由二元组标识：

(目的地IP地址, 目的地端口号)

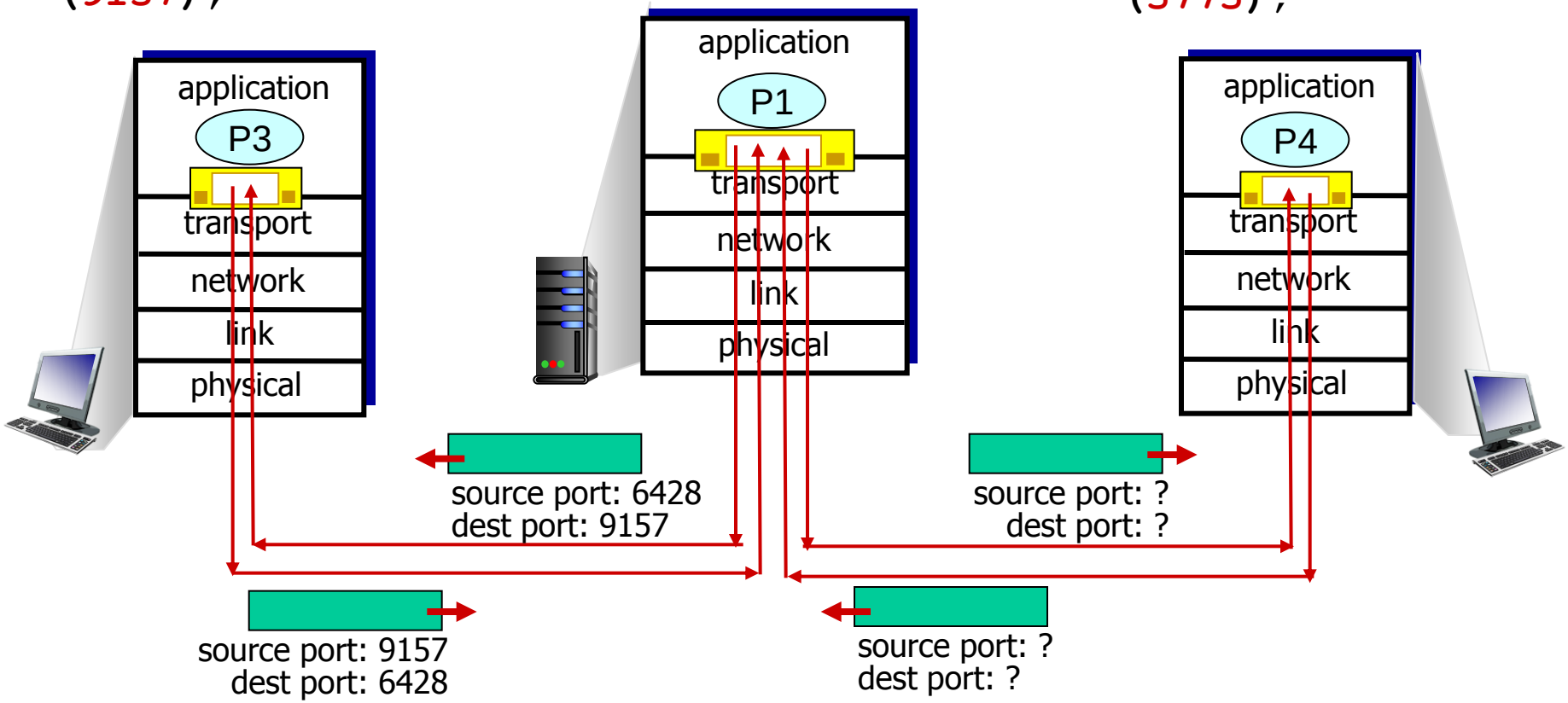
- 当主机接收UDP段时:
  - 在段中检查目的地端口号
  - 将UDP段定向到具有该端口号的套接字
- 具有**不同源IP地址和/或源端口号**，但是**具有相同目的IP地址和相同目的端口号**的IP数据报将定向到相同的套接字

无连接的多路分解的示例

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

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

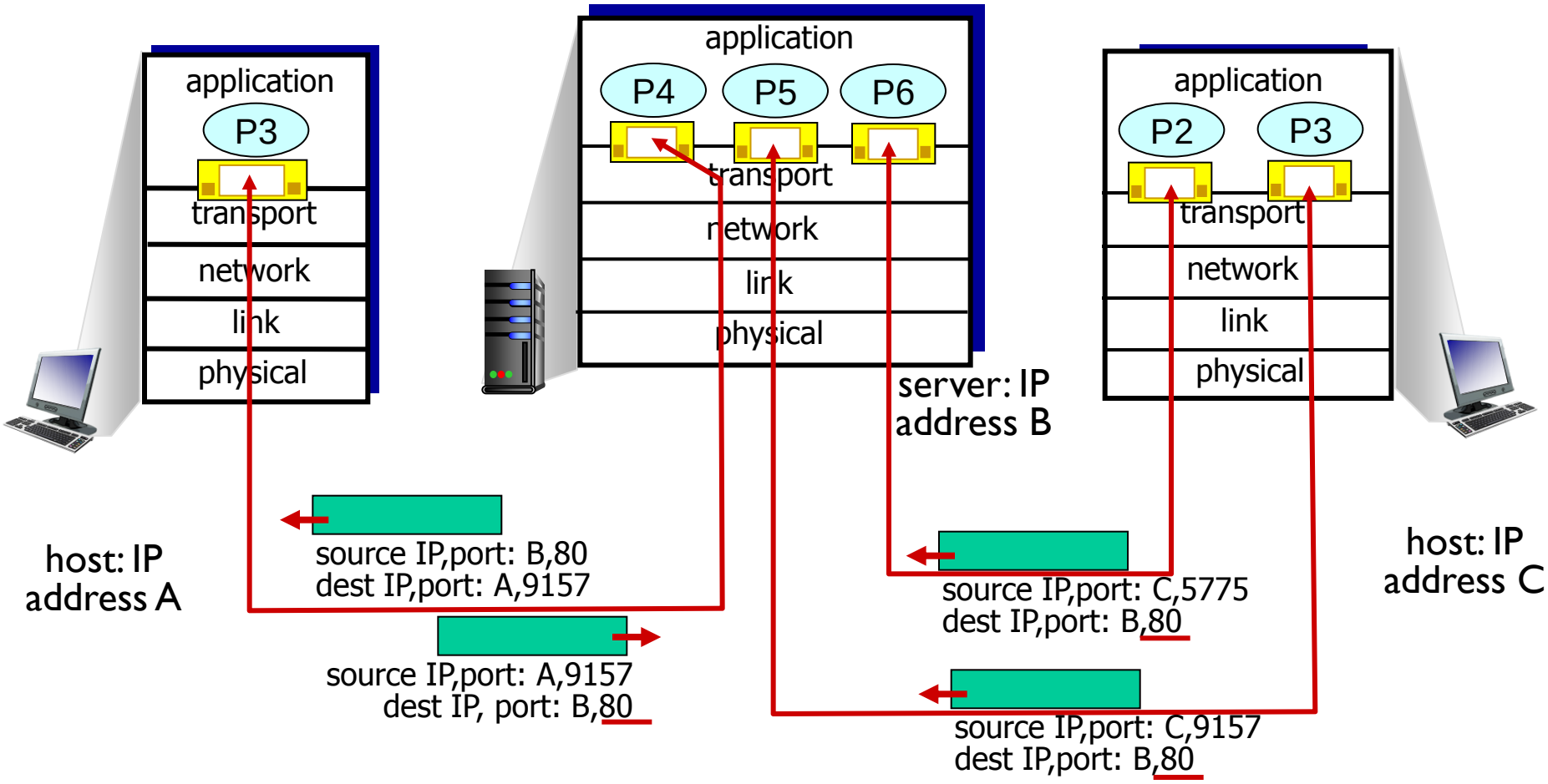
```
DatagramSocket
 mySocket1 = new
 DatagramSocket
 (5775);
```



面向连接的多路分解

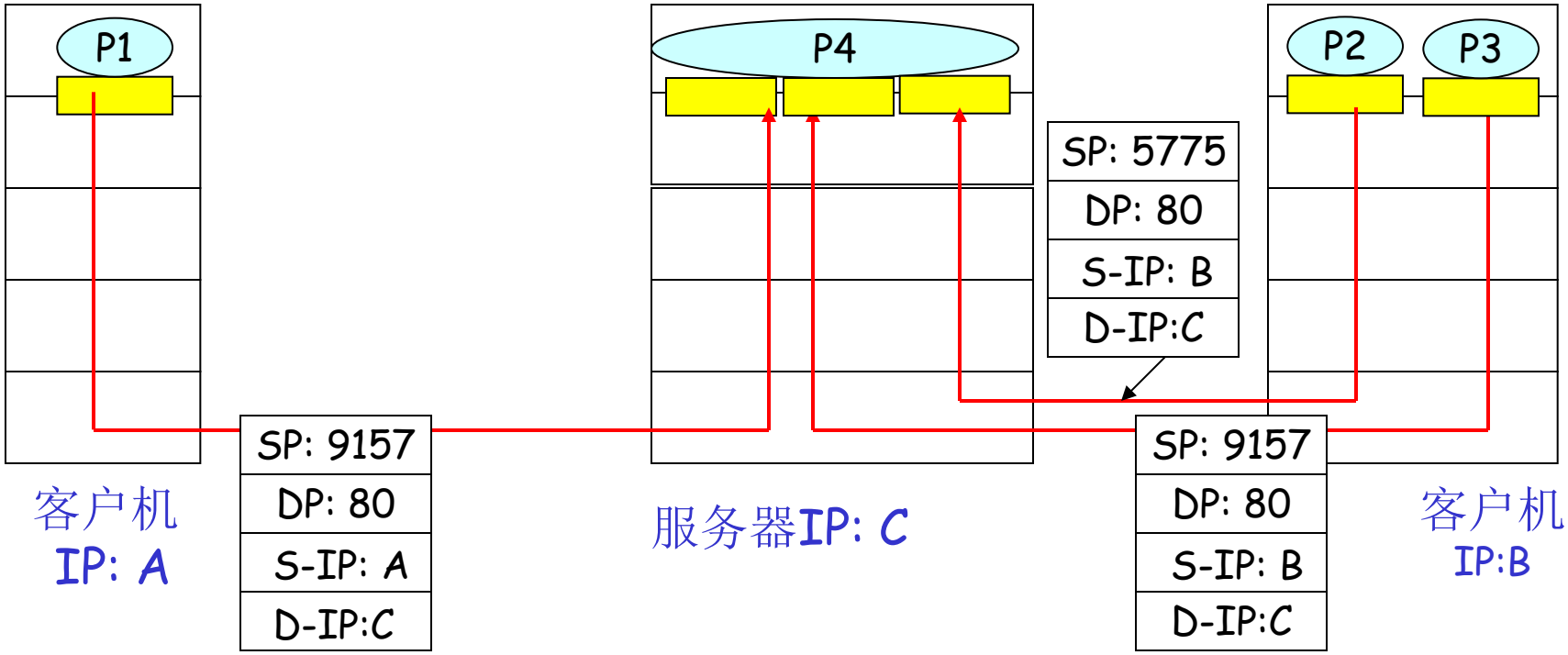
- TCP socket标识是由4-tuple:
  - source IP address
  - source port number
  - dest IP address
  - dest port number
- 分解:接收方使用4个值来将收到的报文段提交到正确的socket
- server host may support many simultaneous TCP sockets:
  - each socket identified by its own 4-tuple
- web servers have different sockets for each connecting client
  - non-persistent HTTP will have different socket for each request

面向连接的多路分解示例



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

面向连接分解: 多线程Web服务器



第3章 主要内容

- 3.1 运输层服务
- 3.2 复用与分解
- 3.3 无连接传输: UDP
- 3.4 可靠数据传输的原则
  - rdt1
  - rdt2
  - rdt3
  - 流水线协议
- 3.5 面向连接的传输: TCP
  - 报文段结构
  - 可靠数据传输
  - 流量控制
  - 连接管理
- 3.6 拥塞控制的原则
- 3.7 TCP拥塞控制
  - 机制
  - TCP吞吐量
  - TCP公平性
  - 时延模型

3.3 UDP: 用户数据报协议 [RFC 768]

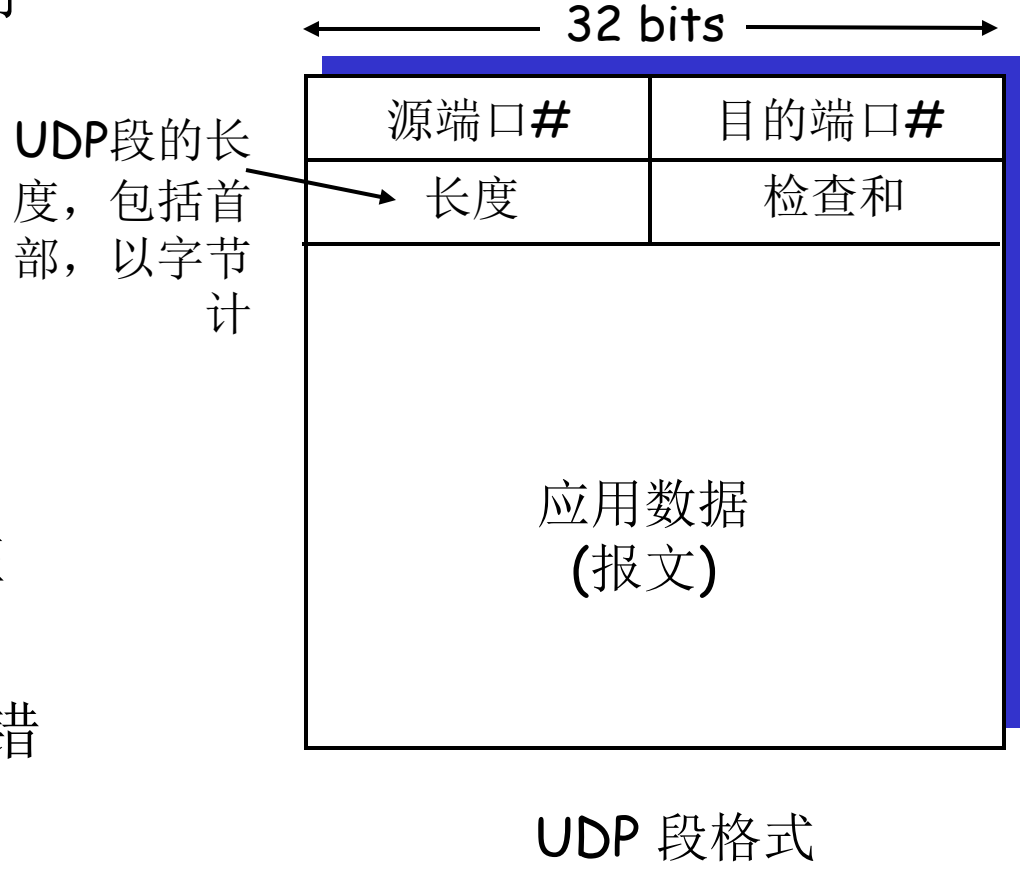
- “没有不必要的,"“基本要素” 互联网传输协议
- "尽力而为”服务，UDP段可能：
  - 丢包
  - 对应用程序交付失序
- *无连接:*
  - 在UDP发送方和接收方之间无握手
  - 每个UDP段的处理独立于其他段

为何要有 UDP协议?

- 无连接创建(它将增加时延)
- 简单：在发送方、接收方无连接状态
- 段首部小
- 无拥塞控制: UDP能够尽可能快地传输

3.3.1 UDP报文段结构

- 常用于流式多媒体应用
  - 丢包容忍
  - 速率敏感
- 其他UDP应用
  - DNS
  - SNMP
- 经UDP的可靠传输：在应用层增加可靠性
  - 应用程序特定的差错恢复！



UDP 段格式

3.3.2 UDP校验和

目的: 在传输的段中检测“差错” (如比特翻转)

发送方:

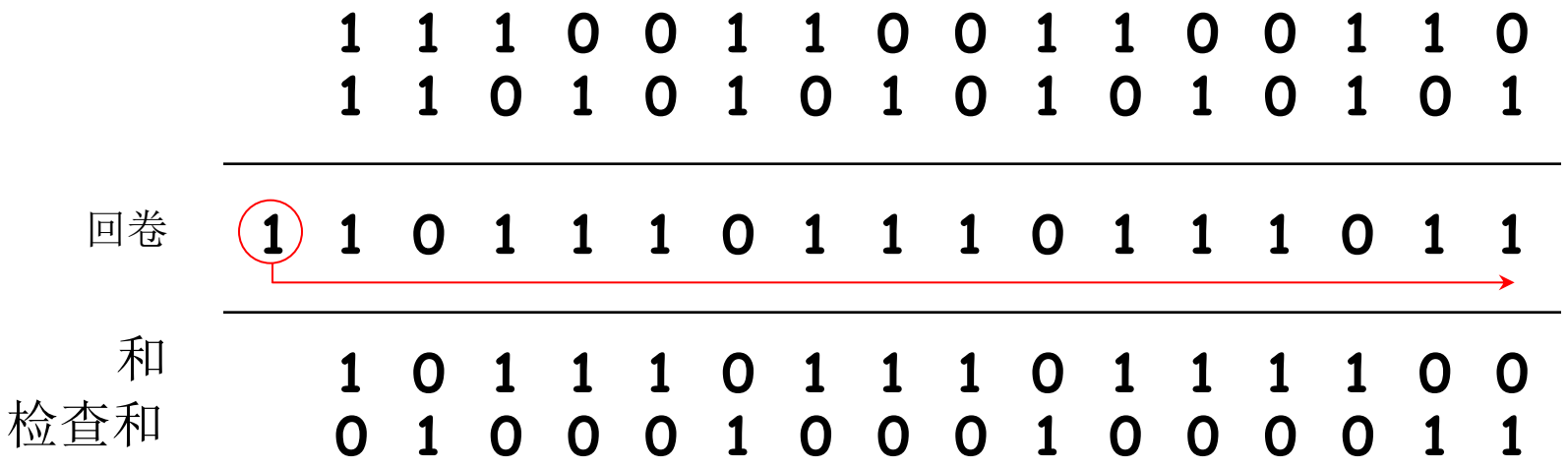
- 将段内容处理为16比特整数序列
- 检查和: 段内容的加法(反码和)
- 发送方将检查和放入UDP检查和字段

接收方:

- 计算接收的段的检查和
- 核对计算的检查和是否等于检查和字段的值:
  - NO – 检测到差错
  - YES – 无差错检测到。*虽然如此，还可能有差错吗？*详情见后……

互联网检查和例子

- 注意
  - 当数字作加法时，最高位进比特位的进位需要加到结果中
- 例子: 两个16-bit整数相加



课堂练习

P192-R3，R7，R8
P194-P3

R3. Consider a TCP connection between Host A and Host B. Suppose that the TCP segments traveling from Host A to Host B have source port number x and destination port number y. What are the source and destination port numbers for the segments traveling from Host B to Host A?

R7. Suppose a process in Host C has a UDP socket with port number 6789. Suppose both Host A and Host B each send a UDP segment to Host C with destination port number 6789. Will both of these segments be directed to the same socket at Host C? If so, how will the process at Host C know that these two segments originated from two different hosts?

R8. Suppose that a Web server runs in Host C on port 80. Suppose this Web server uses persistent connections, and is currently receiving requests from two different Hosts, A and B. Are all of the requests being sent through the same socket at Host C? If they are being passed through different sockets, do both of the sockets have port 80? Discuss and explain.

课堂练习

P192-R3，R7，R8
P194-P3

P3. UDP and TCP use 1s complement for their checksums. Suppose you have the following three 8-bit bytes: 01010011, 01100110, 01110100. What is the 1s complement of the sum of these 8-bit bytes? (Note that although UDP and TCP use 16-bit words in computing the checksum, for this problem you are being asked to consider 8-bit sums.) Show all work. Why is it that UDP takes the 1s complement of the sum; that is, why not just use the sum? With the 1s complement scheme, how does the receiver detect errors? Is it possible that a 1-bit error will go undetected? How about a 2-bit error?

第3章 主要内容

- 3.1 运输层服务
- 3.2 复用与分解
- 3.3 无连接传输: UDP
- 3.4 可靠数据传输的原则
  - rdt1
  - rdt2
  - rdt3
  - 流水线协议
- 3.5 面向连接的传输: TCP
  - 报文段结构
  - 可靠数据传输
  - 流量控制
  - 连接管理
- 3.6 拥塞控制的原则
- 3.7 TCP拥塞控制
  - 机制
  - TCP吞吐量
  - TCP公平性
  - 时延模型