第3章 运输层
Transport Layer

任课老师：周军海
联系方式：13107423988
Email:rj_zjh@hnu.edu.cn

第3章:运输层

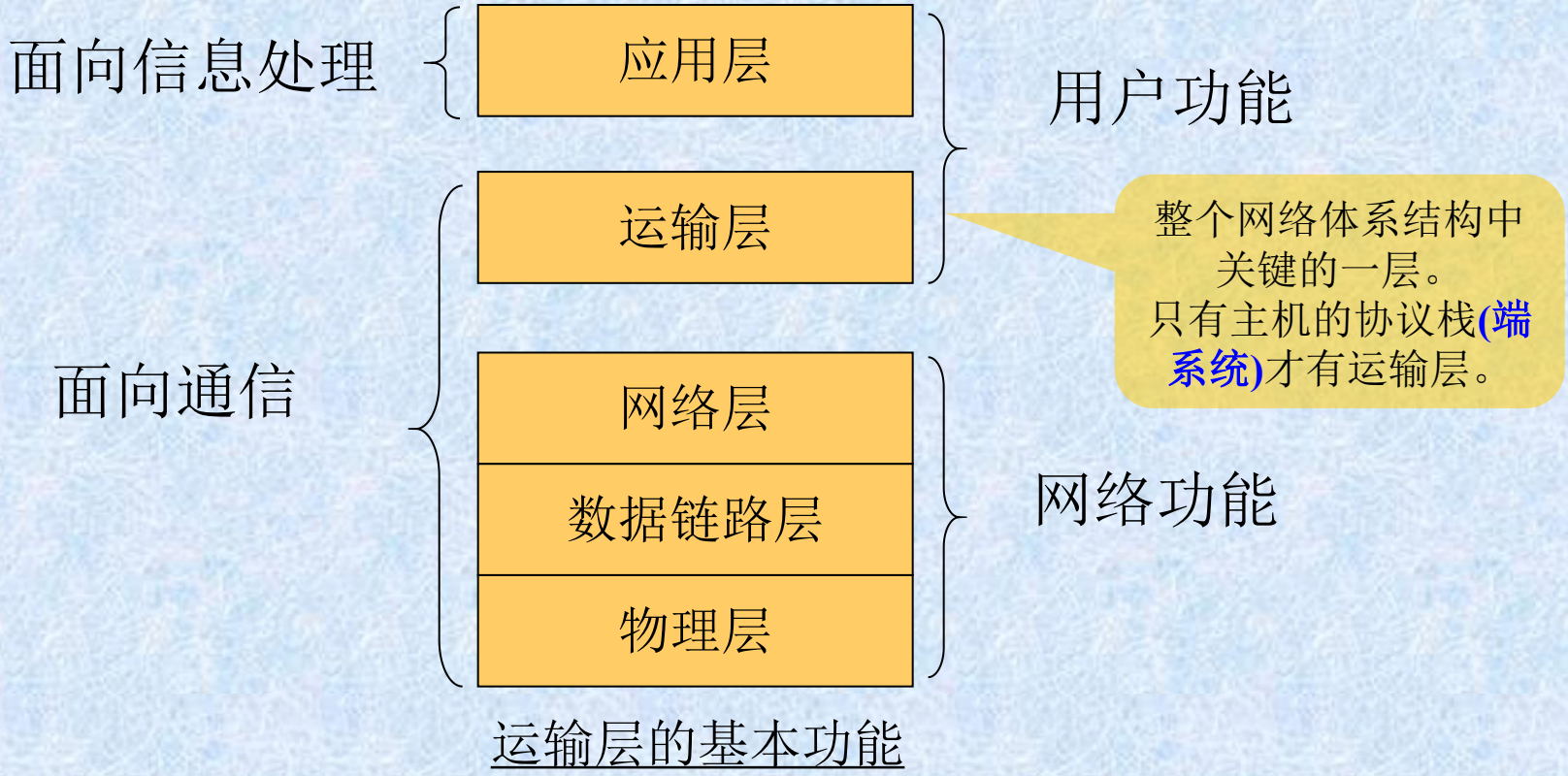
我们的目的:

- 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 要点

- 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 拥塞控制的原则
- 3.7 TCP拥塞控制
  - 机制
  - TCP吞吐量
  - TCP公平性
  - 时延模型

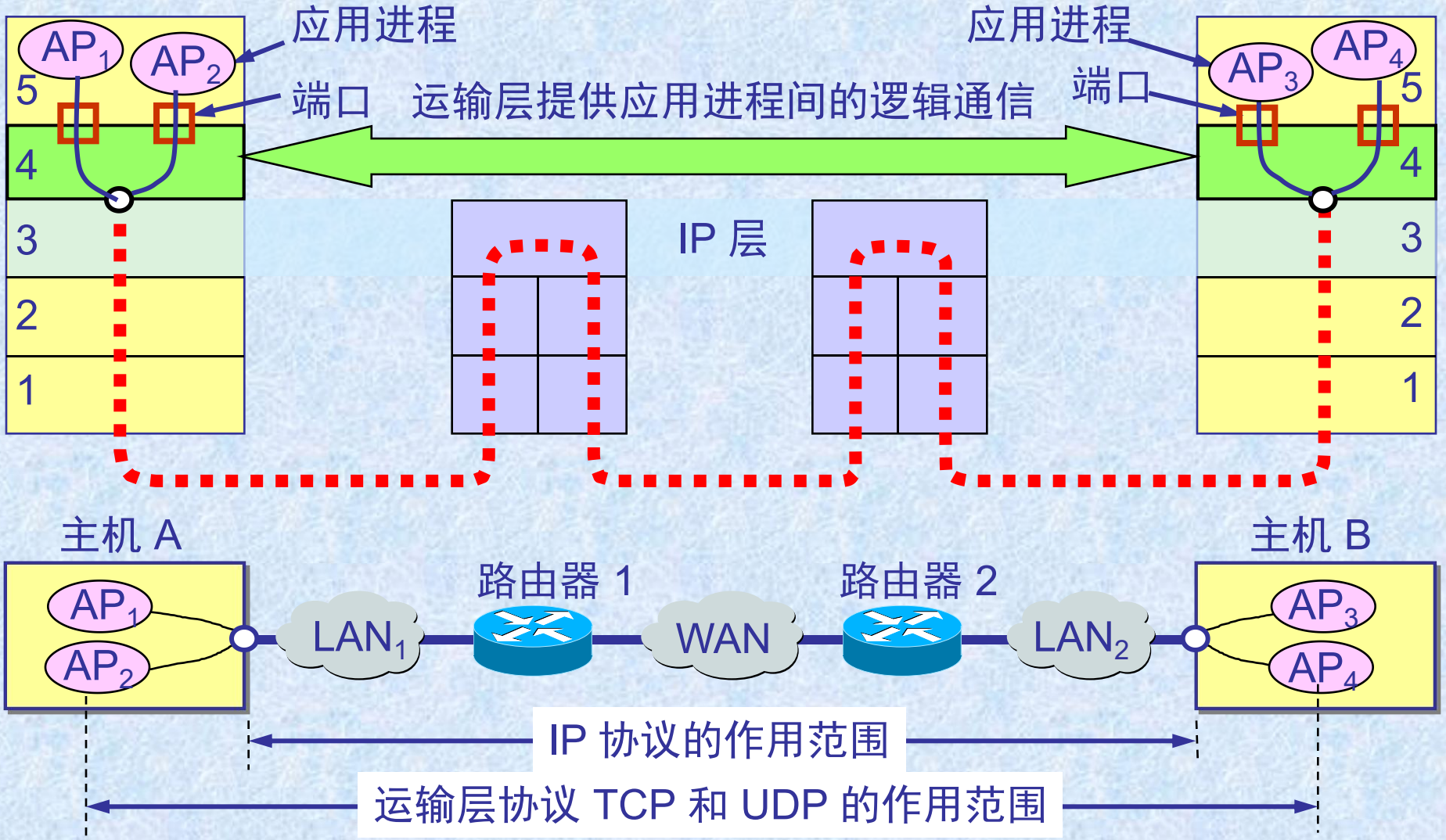
运输层服务



运输层的基本功能

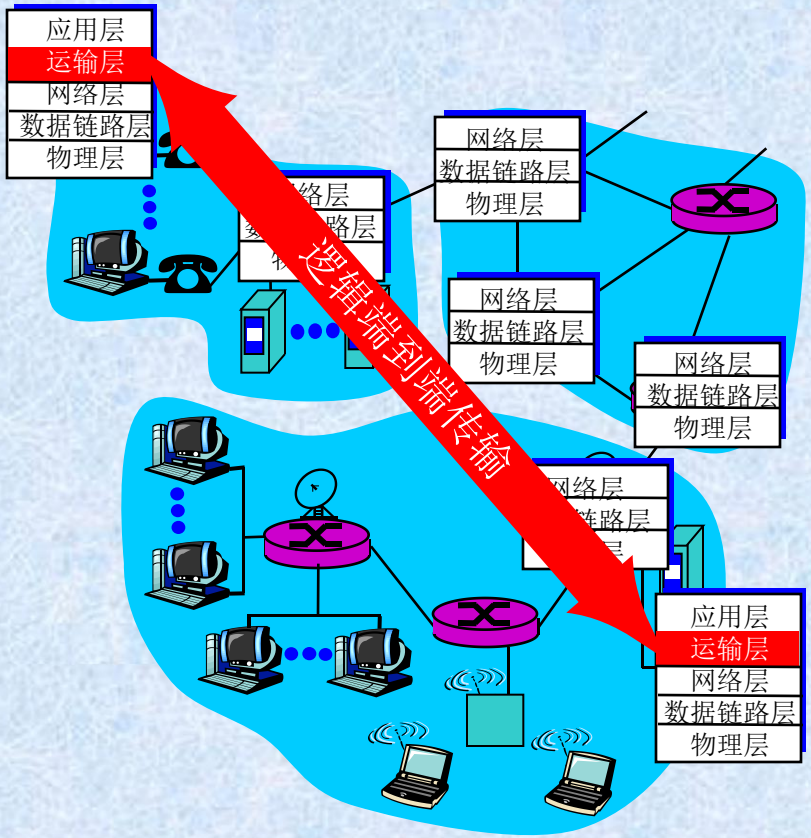
- 计算机网络本质是实现分布在不同地理位置的联网主机之间的进程通信，以实现各种网络服务功能；
- 运输层的主要作用向应用层提供通信服务，实现端到端的数据传输。

运输层为相互通信的应用进程提供了逻辑通信



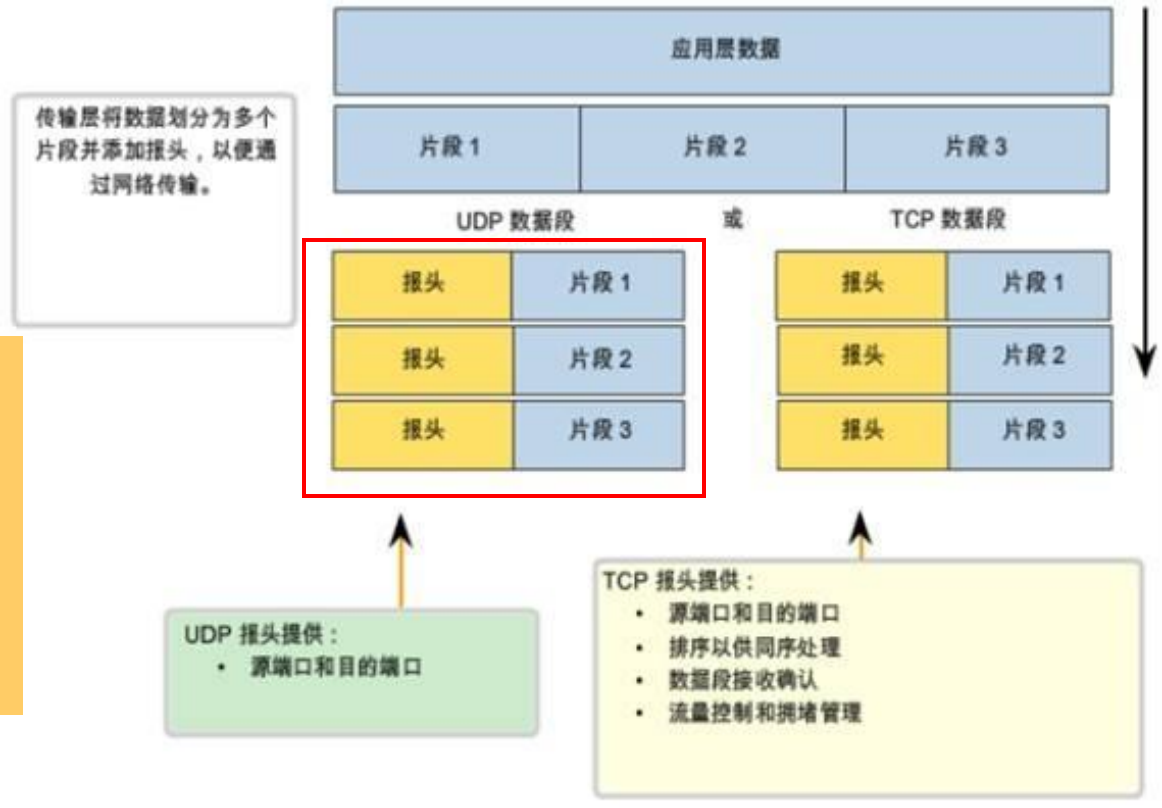
运输层服务

- 为运行在不同主机上的应用进程之间提供*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



运输层协议数据封装

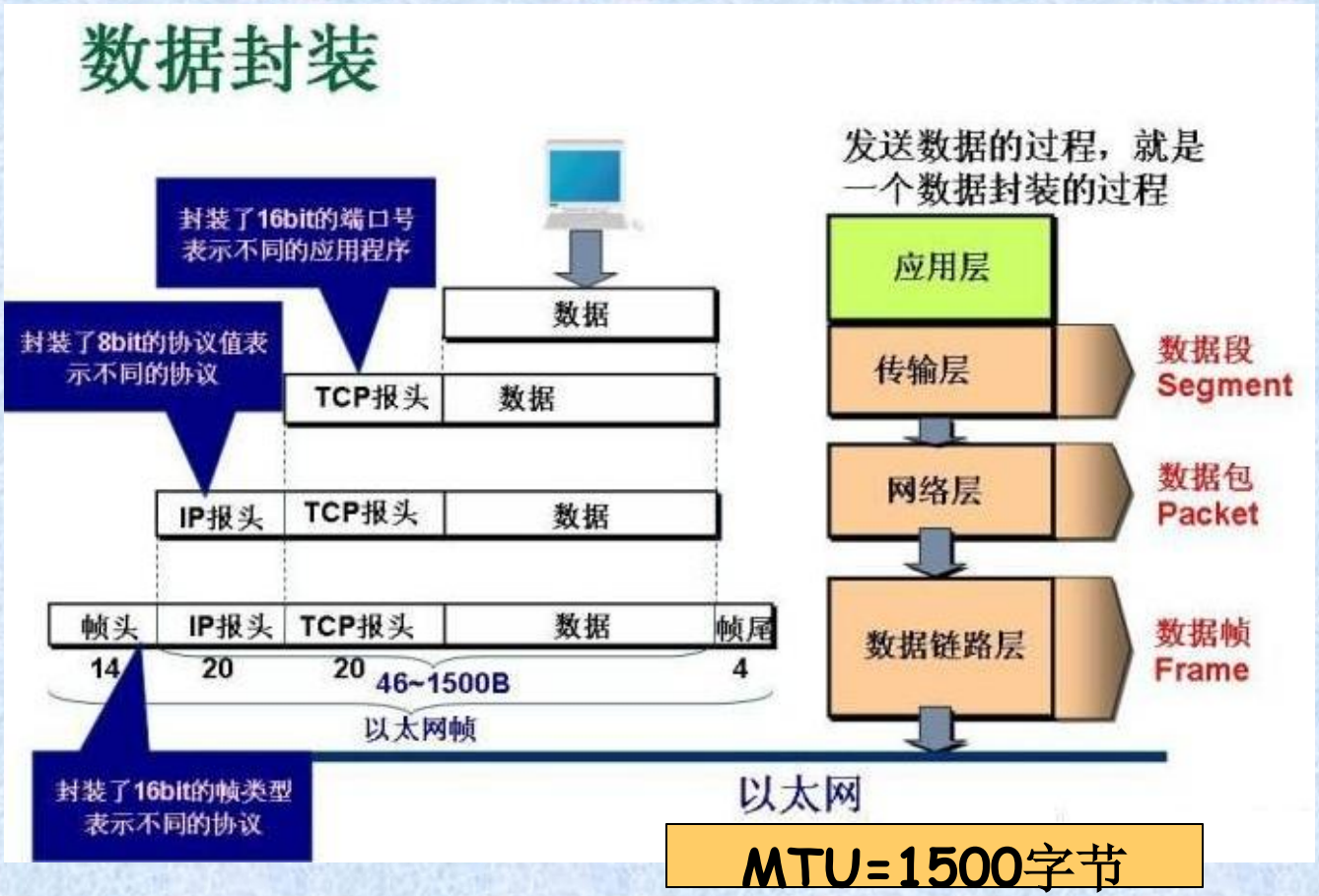
- 为何需要数据分段？
  - TCP具有最大报文长度（Maximum Segment Size, MSS）



UDP数据报不会自己进行分段，因此当长度超过了MTU时，会在网络层进行IP分片。

数据封装的过程

- 最大传输单元（Maximum Transmission Unit,MTU）----最大链路层帧长度



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

注意1：运输层协议在端系统中运行，将报文传送到网络边缘

注意2：不同运输层协议

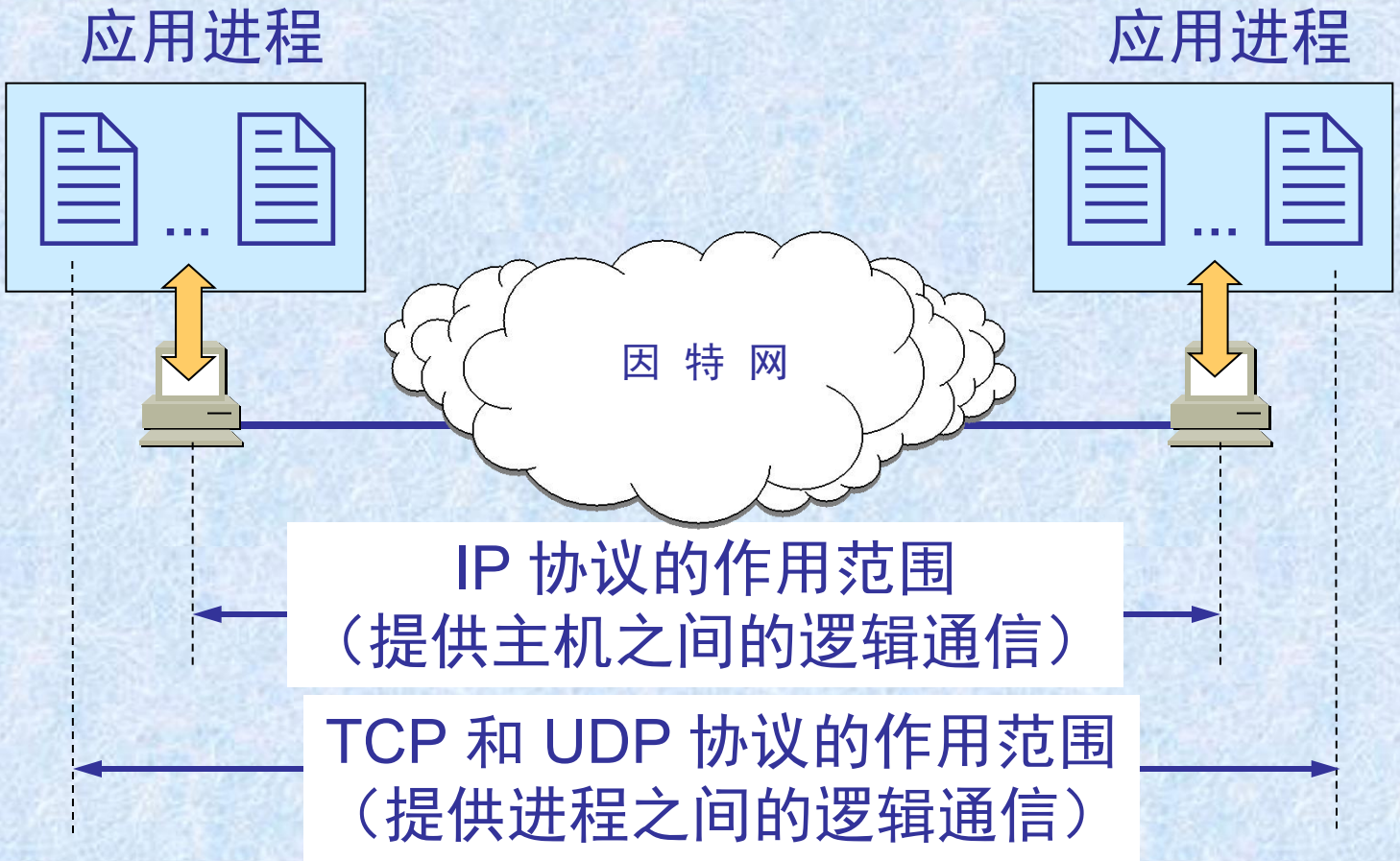
注意3：运输层提供的服务受制于网络层协议的服务模型

家庭类比:

12个孩子向12个孩子发信

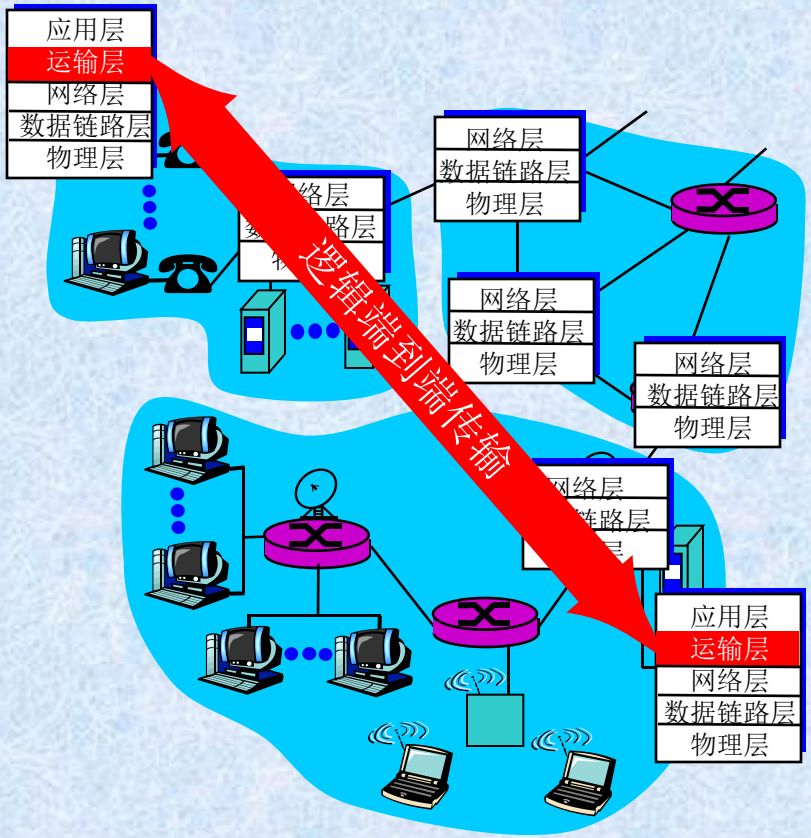
- 进程 = 孩子
- 应用报文= 信封中的信
- 主机 = 家庭
- 运输协议 = Ann和Bill
- 网络层协议= 邮政服务

运输层协议和网络层协议的主要区别



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

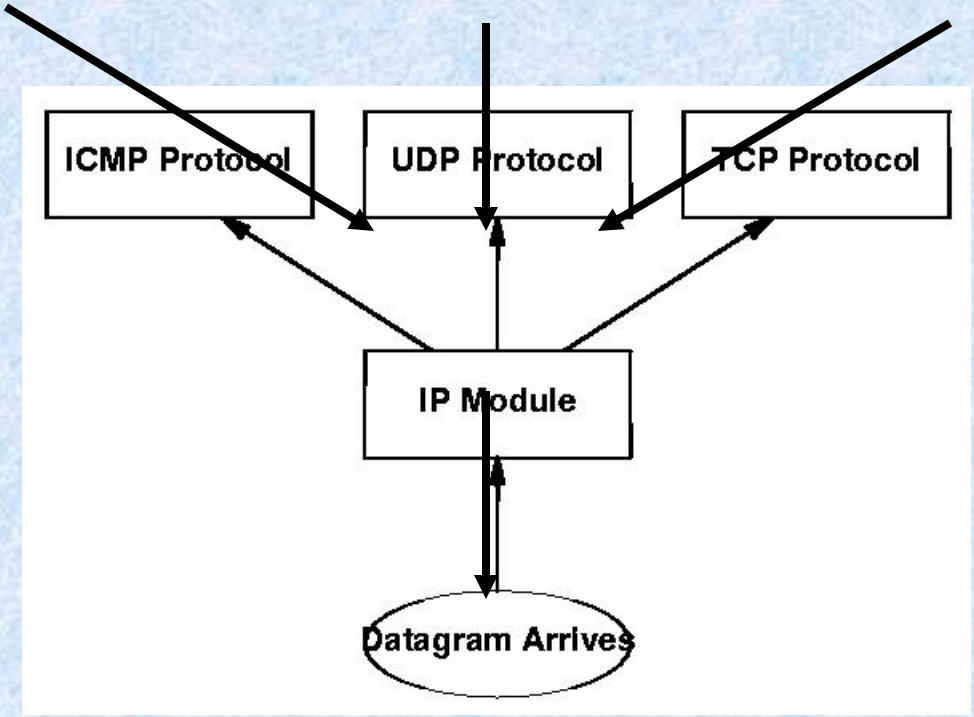


第3章 要点

- 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 拥塞控制的原则
- 3.7 TCP拥塞控制
  - 机制
  - TCP吞吐量
  - TCP公平性
  - 时延模型

Internet 层的复用与分解

- 多路复用与多路分解将由网络层提供的主机到主机的交付服务延伸到进程到进程的交付服务。



复用/分解

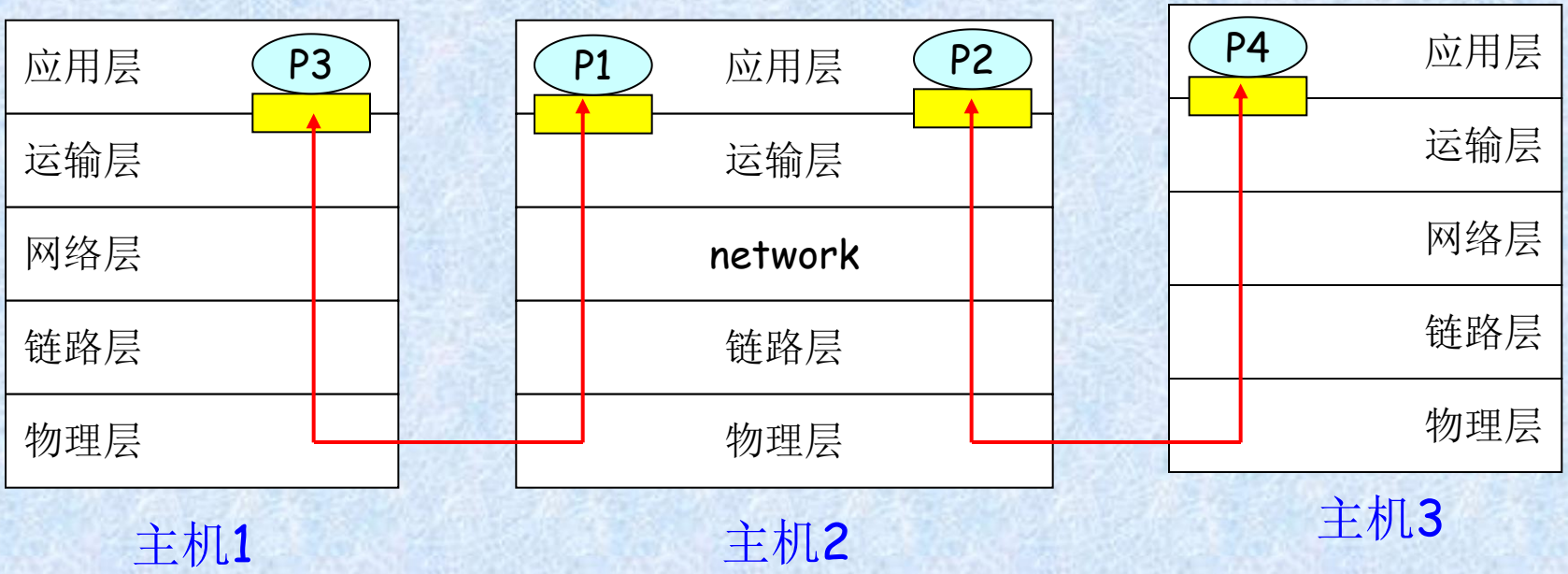
在接收主机分解：

将运输层接收到的报文段交付给正确的套接字（一路到多路，向上）

在发送主机复用:

从多个套接字收集数据，用首部封装数据（多路到一路，向下）

= 套接字　　= 进程

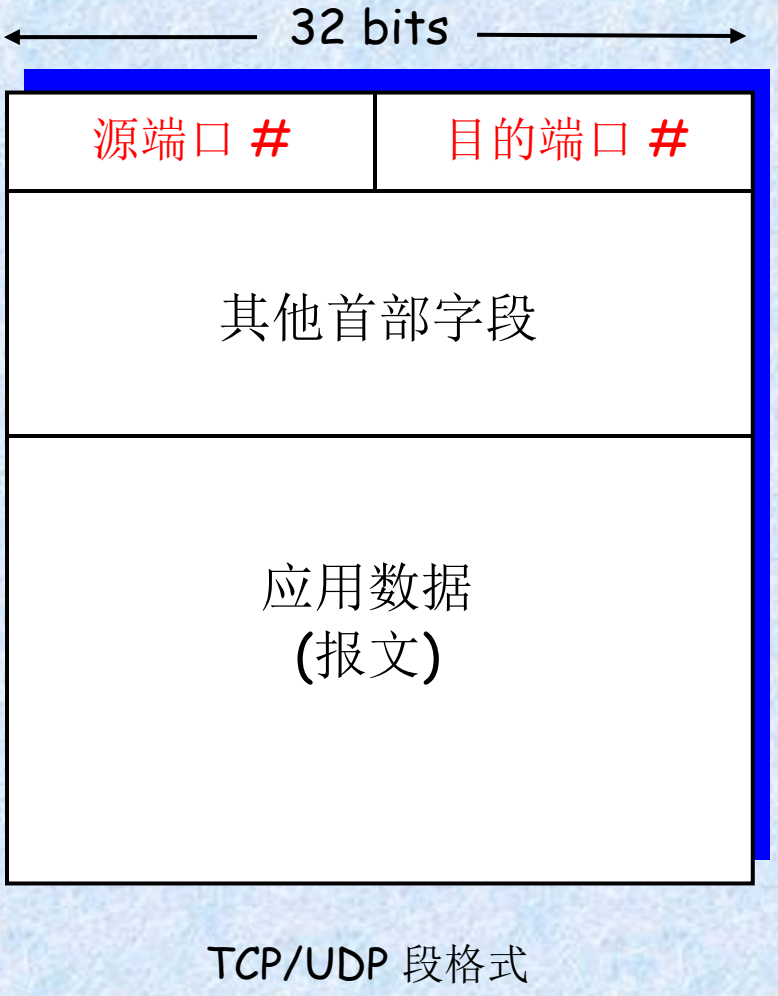


分解工作过程

- 主机接收IP数据

  - 每个段具有源、目的端口号。
    - 端口号16bit，0~1023周知端口
  - 每个数据报承载1个运输层段

- 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

无连接的复用和分解

- 生成具有端口号的套接字:

```
DatagramSocket mySocket1 = new
    DatagramSocket(9911);
DatagramSocket mySocket2 = new
    DatagramSocket(9922);
```

- 新建数据报文:
- `DatagramPacket dp=new DatagramPacket(buf, buf.length, destIA, destPort);`
- 发送数据:

```
mySocket1.send(dp);
```

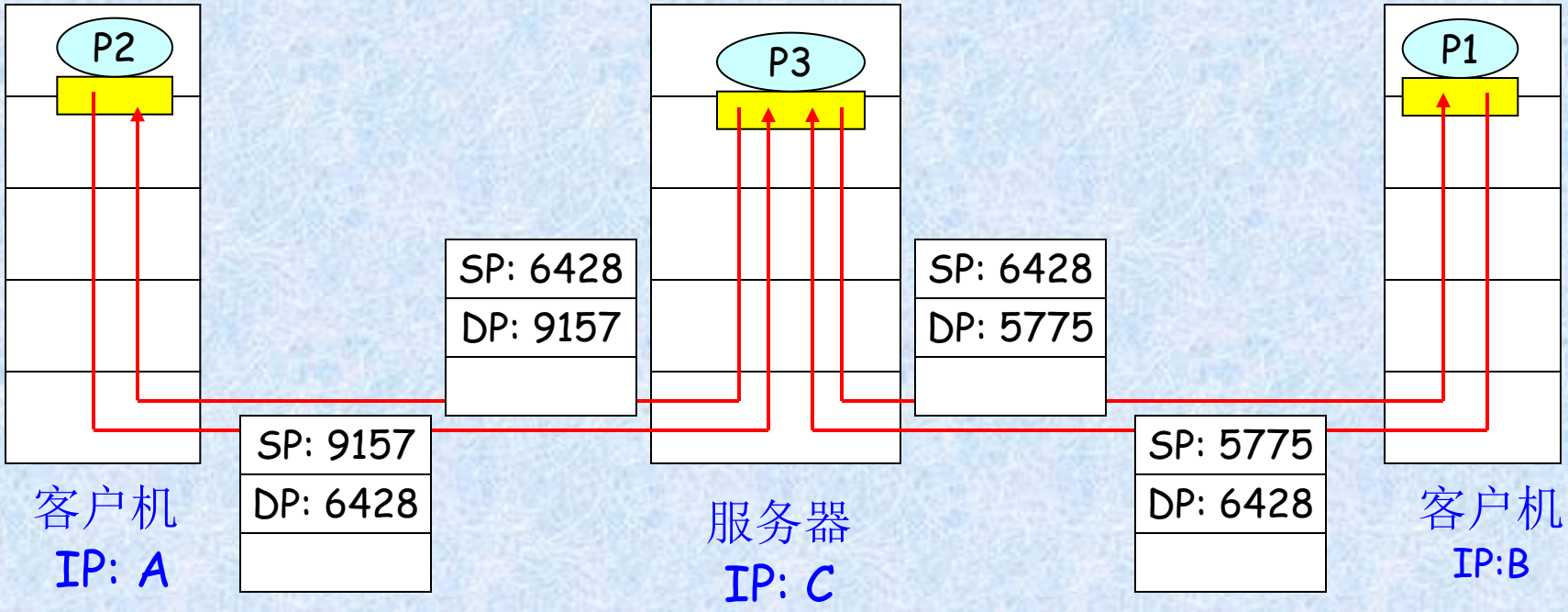
- 接收数据

```
mySocket2.receive(dp);
```

- 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
- 思考：具有不同源IP地址和/或源端口号的IP数据报可能定向到相同的套接字吗？P188 R7

无连接分解(续)

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



SP提供了“返回地址”

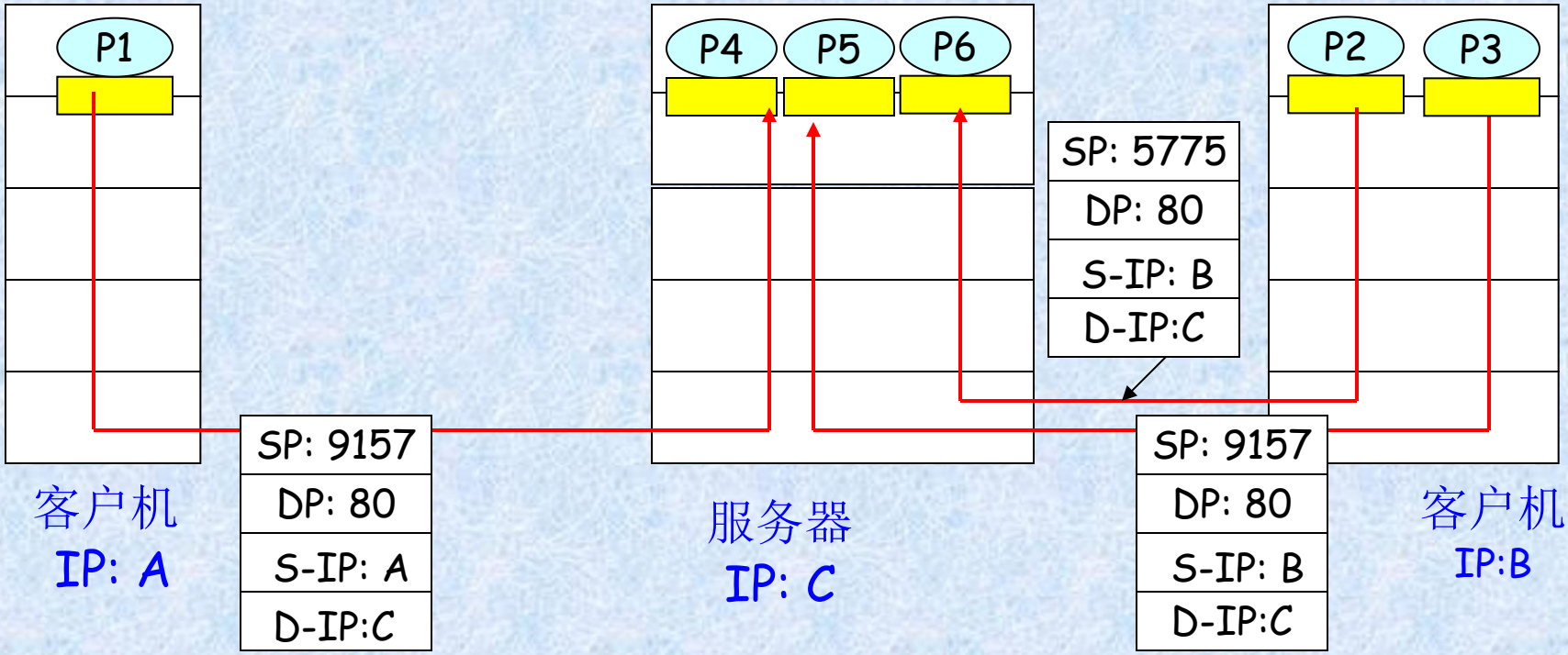
面向连接分解

- TCP套接字由四元组标识:
  - 源IP地址
  - 源端口号
  - 目的IP地址
  - 目的端口号
- 接收主机使用这四个值来将段定向到适当的套接字
- 服务器主机可能支持许多并行的TCP套接字：
  - 每个套接字由其自己的四元组标识
- Web服务器对每个连接的客户机具有不同的套接字
  - 非持久HTTP将为每个请求具有不同的套接字

```
ServerSocket welcomeSocket=new
ServerSocket(1888);
Socket socket=welcomeSocket.accept();
```

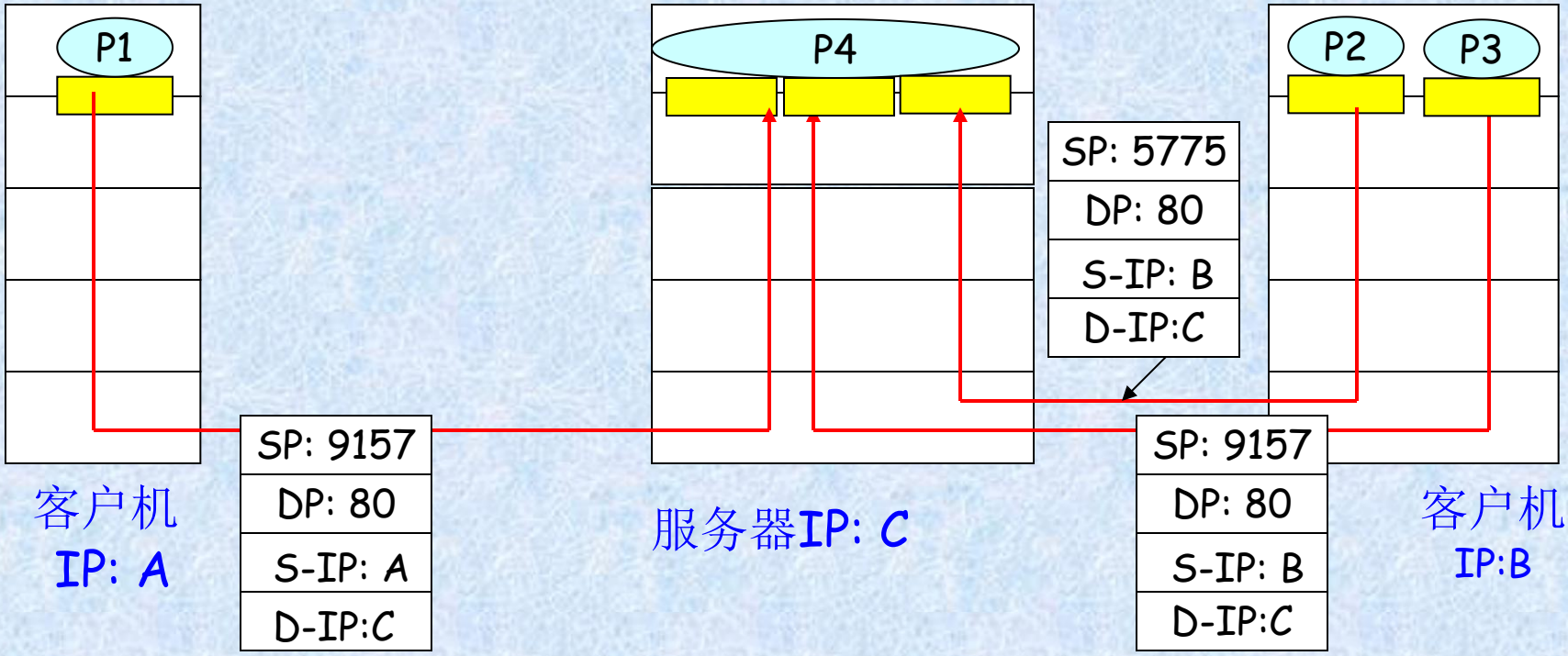
```
Socket clientSocket=new Socket();
clientSocket.connect(new InetSocketAddress("localhost", 1888));
```

面向连接分解 (续)



- 思考：P188 R8，P189 P1，P189 P2

面向连接分解: 多线程Web服务器



- 连接套接字与进程之间并非一一对应的关系。
- 当今高性能Web服务器通常只使用一个进程，但为每个新的客户连接创建一个具有新连接套接字的新线程。
  - 持续HTTP与非持续HTTP

第3章 要点

- 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

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
  - 复用/分解
  - 差错检测
- "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

UDP: 其他

- 常用于流式多媒体应用
  - 丢包容忍
  - 速率敏感
- 其他UDP应用
  - DNS
  - 路由表更新
  - SNMP（Simple Network Management Protocol,简单网络管理协议）
- 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

UDP检验和

目的: 在传输的段中检测“差错” (如比特翻转)

发送方:

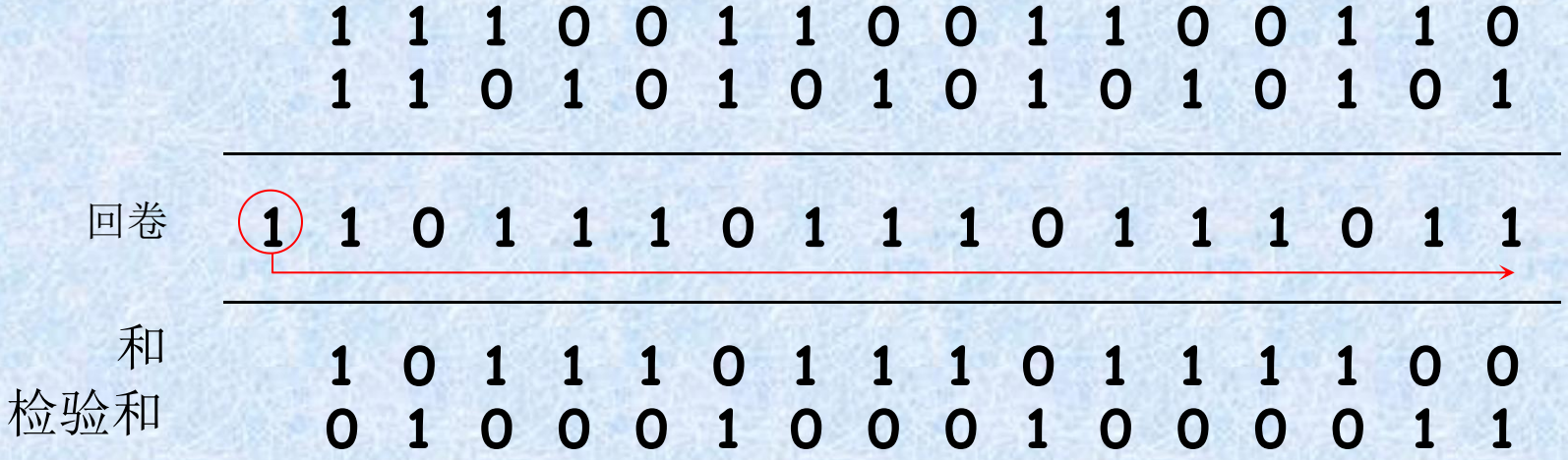
- 将段内容处理为16比特整数序列
- 检验和: 段内容的加法 (反码和，求和时溢出都被回卷)
- 发送方将检验和放入UDP检验和字段

接收方:

- 计算接收的段的检验和
- 核对全部字段之和是否为1111111111111111
  - NO - 检测到差错
  - YES - 无差错检测到。*虽然如此，还可能有差错吗？*
  - *P5*
  - *是否可以纠正差错？*

互联网检验和例子

- 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



思考：UDP提供的是端到端的差错检测，链路层对传输过程的帧进行差错检查。因此UDP提供的差错检查是否没有必要？

思考：P3，P4

思考1：

- 假定在主机C上的一个进程有一个具有端口号6789的UDP套接字。假定主机A和主机B都用目的端口号6789向主机C发送一个UDP报文段。这两台主机的这些报文段在主机C都被描述为相同的套接字吗？如果是这样的话，在主机C的该进程将怎样知道源于两台不同主机的这两个报文段？

  答：是的，两个报文段都将指向同一个套接字。对于每个接收到的报文段，操作系统都将向进程提供其源IP地址，通过所收到的IP数据报可以确定各个报文段的起源。

思考2：

- 假定在主机C端口80上运行的一个Web服务器。假定这个Web服务器使用持续连接，并且正在接收来自两台不同主机A和B的请求。被发送的所有请求都通过位于主机C的相同套接字吗？如果它们通过不同的套接字传递，这两个套接字都具有端口80吗？讨论和解释之。

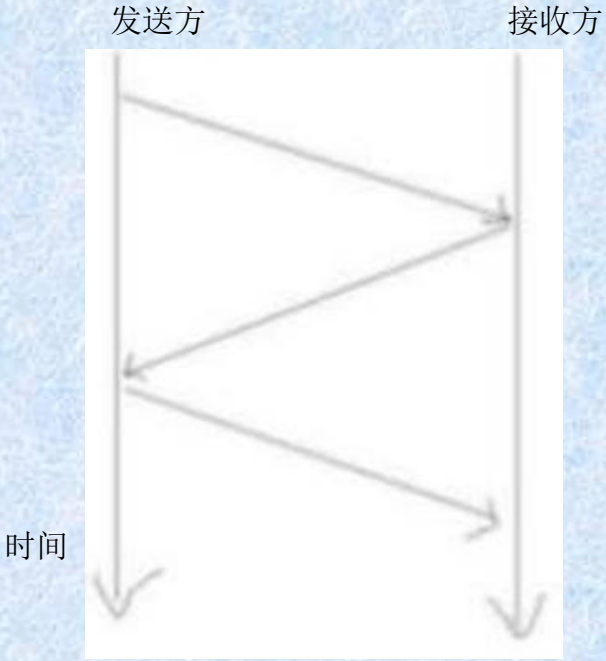
答：对于每个持久连接，Web服务器都会创建一个单独的“连接套接字”。每个连接套接字由一个四元组标识：（源IP地址、源端口号、目标IP地址、目标端口号）。当主机C接收到IP数据报时，它会检查数据报中的这四个字段，以确定它应该将TCP报文段的有效负载传递到哪个套接字。因此，来自A和B的请求通过不同的套接字。
A和B的请求会通过80端口找到Web服务器进程，就这里而言它们通过C的欢迎套接字进行三次握手连接，这个欢迎套接字具有端口80.当它们与服务器进程建立连接的时候，服务器进程会单独为它们分配套接字，通过专门的"连接套接字"响应客户端的请求. 这两个套接字的目标端口号均为80。

单选题 1分

设置

1、下图描述的协议要素是

Ⅰ、语法 Ⅱ、语义 Ⅲ、时序



A 仅Ⅰ

B 仅Ⅱ

C 仅Ⅲ

D Ⅰ、Ⅱ、Ⅲ

提交

单选题 1分

设置

2、假设OSI参考模型的应用层欲发送400B的数据（无拆分），除物理层和应用层之外，其他各层在封装PDU时均引入20B的额外开销，则应用层数据传输效率约为

A 80%

B 83%

C 87%

D 91%

提交