

第3章 运输层

Transport Layer

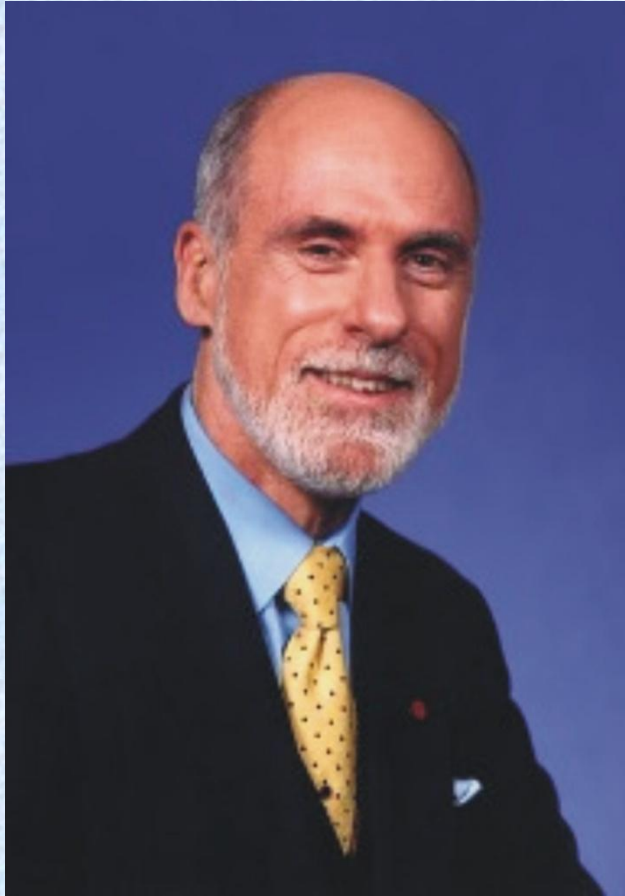
TCP

任课老师：周军海
Email:rj_zjh@hnu.edu.cn

第3章 要点

- 3.1 传输层服务
- 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

2004年图灵奖 TCP/IP协议发明者



Vinton G. Cerf
温顿·瑟夫



Robert E. Kahn
罗伯特·卡恩

TCP概述

□ 端到端:

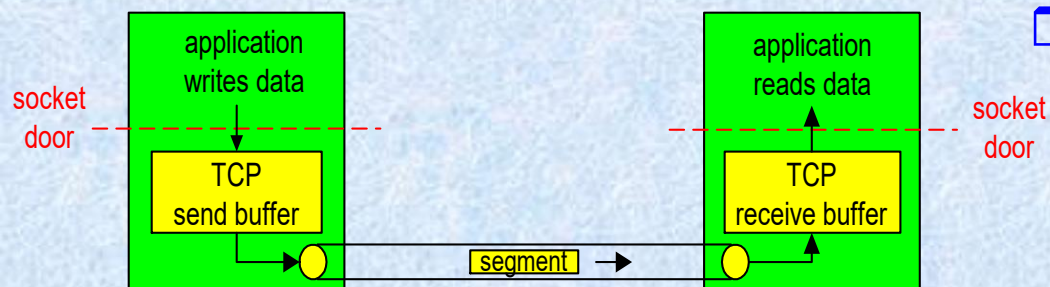
- 一个发送方, 一个接收方
- 连接状态与端系统有关, 不为路由器所知

□ 可靠、有序的字节流

□ 流水线:

- TCP拥塞和流量控制设置滑动窗口协议

□ 发送和接收缓冲区



□ 全双工数据:

- 同一连接上的双向数据流
- MSS: 最大报文段长度
- MTU: 最大传输单元

□ 面向连接:

- 在进行数据交换前, 初始化发送方与接收方状态, 进行握手(交换控制信息)

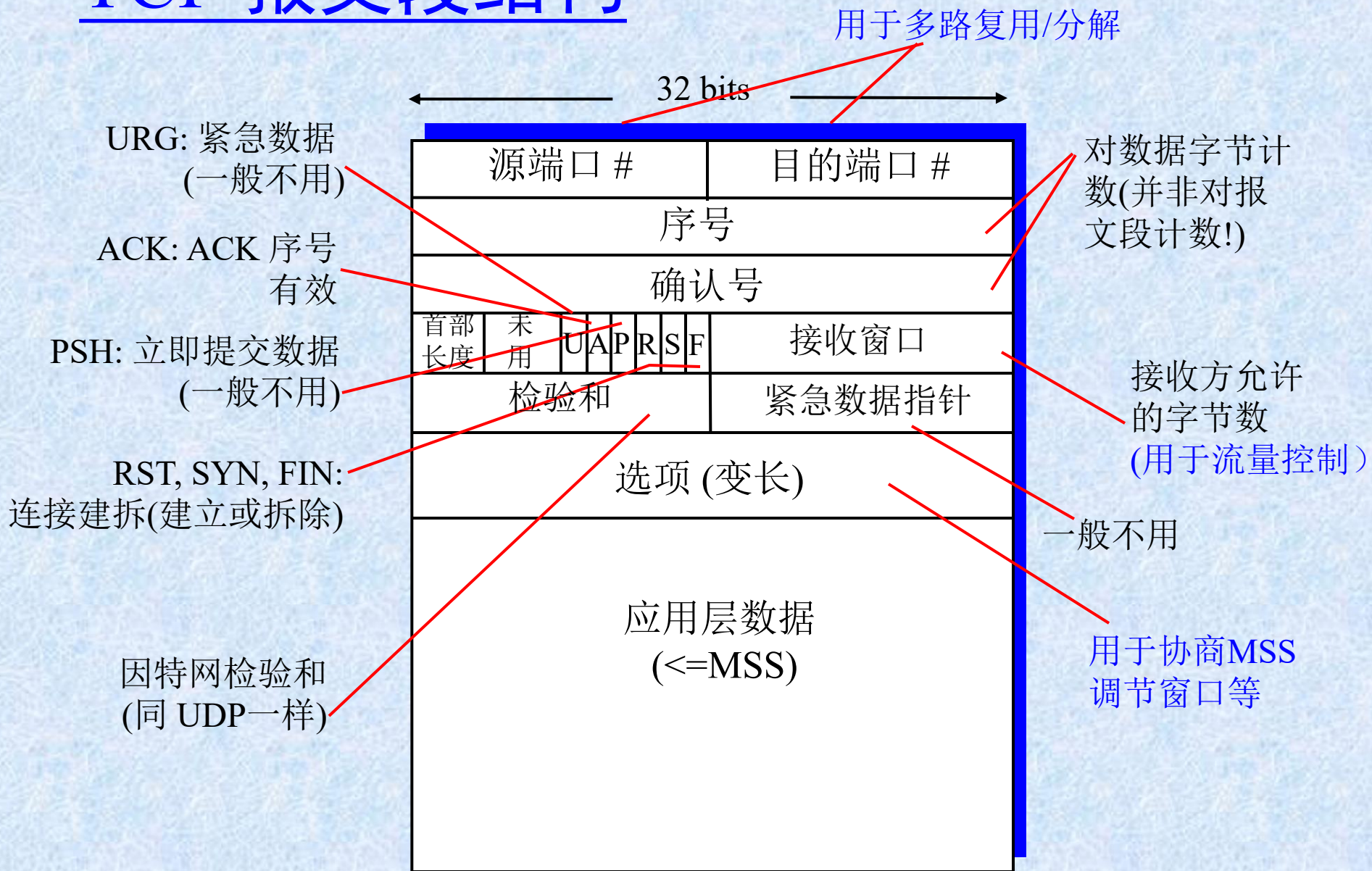
□ 流量控制:

- 发送方不能淹没接收方

□ 拥塞控制:

- 抑制发送方速率来防止过分占用网络资源

TCP 报文段结构



4bit首部长度字段，指示了以32bit的字为单位的TCP首部长度。

TCP序号和确认号

序号:

- 报文段中第1个数据字节在字节流中的位置编号 (P155图3-30)

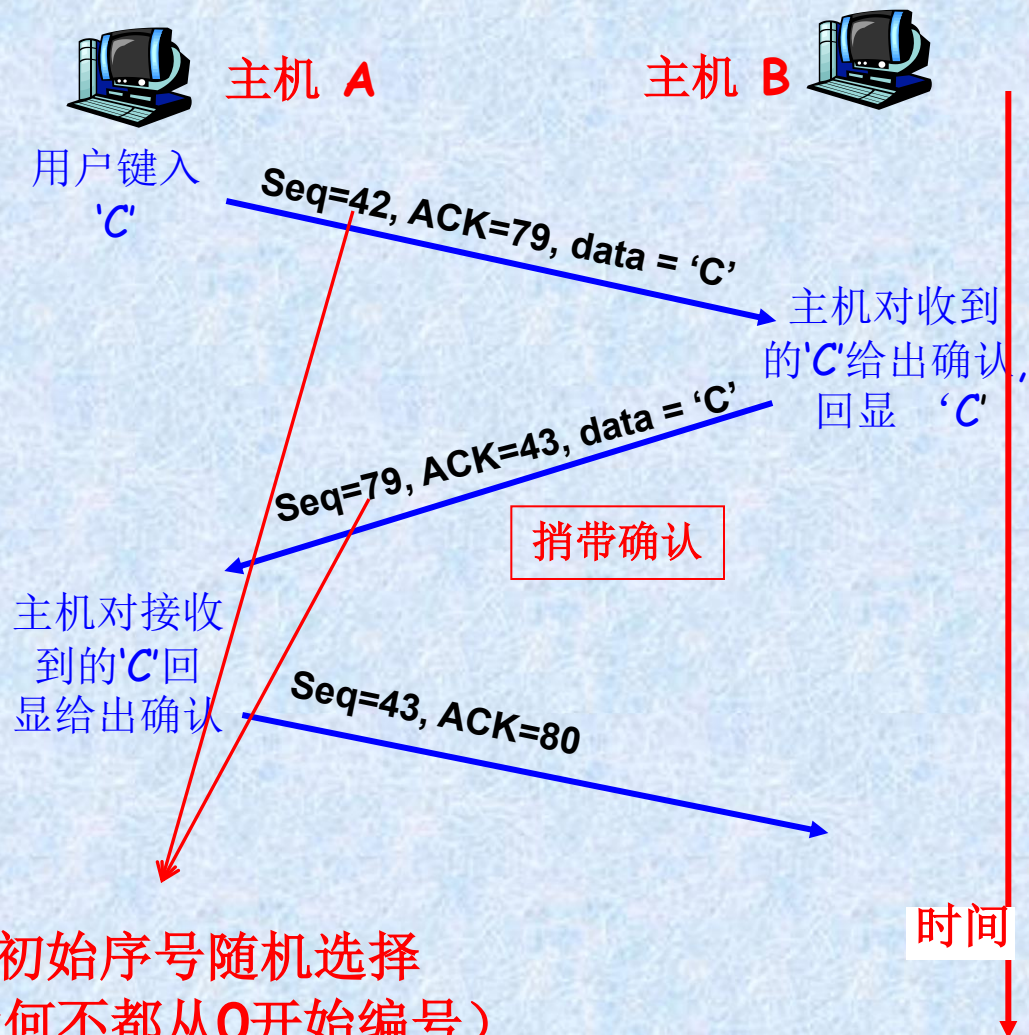
确认号:

- 期望从对方收到的下一个字节的序号
- 累计确认(P156举例)

问题: 接收方如何处理失序报文段?

回答: TCP规范没有说明, 由实现者自行选择实现: 抛弃/缓存

简单的telnet情况



初始序号随机选择
(为何不都从0开始编号)

TCP往返时延(RTT)的估计与超时

问题: 如何设置TCP
超时值?

- ❑ 应大于RTT
 - 但RTT是变化的
- ❑ 太短: 过早超时
 - 不必要的重传
- ❑ 太长: 对报文段的丢失响应太慢

问题: 如何估计RTT?

- ❑ **SampleRTT**: 从发送报文段到接收到ACK的测量时间
 - 忽略重传
- ❑ **SampleRTT**会变化, 希望估计的RTT“较平滑”
 - 平均最近的测量值, 并不仅仅是当前**SampleRTT**

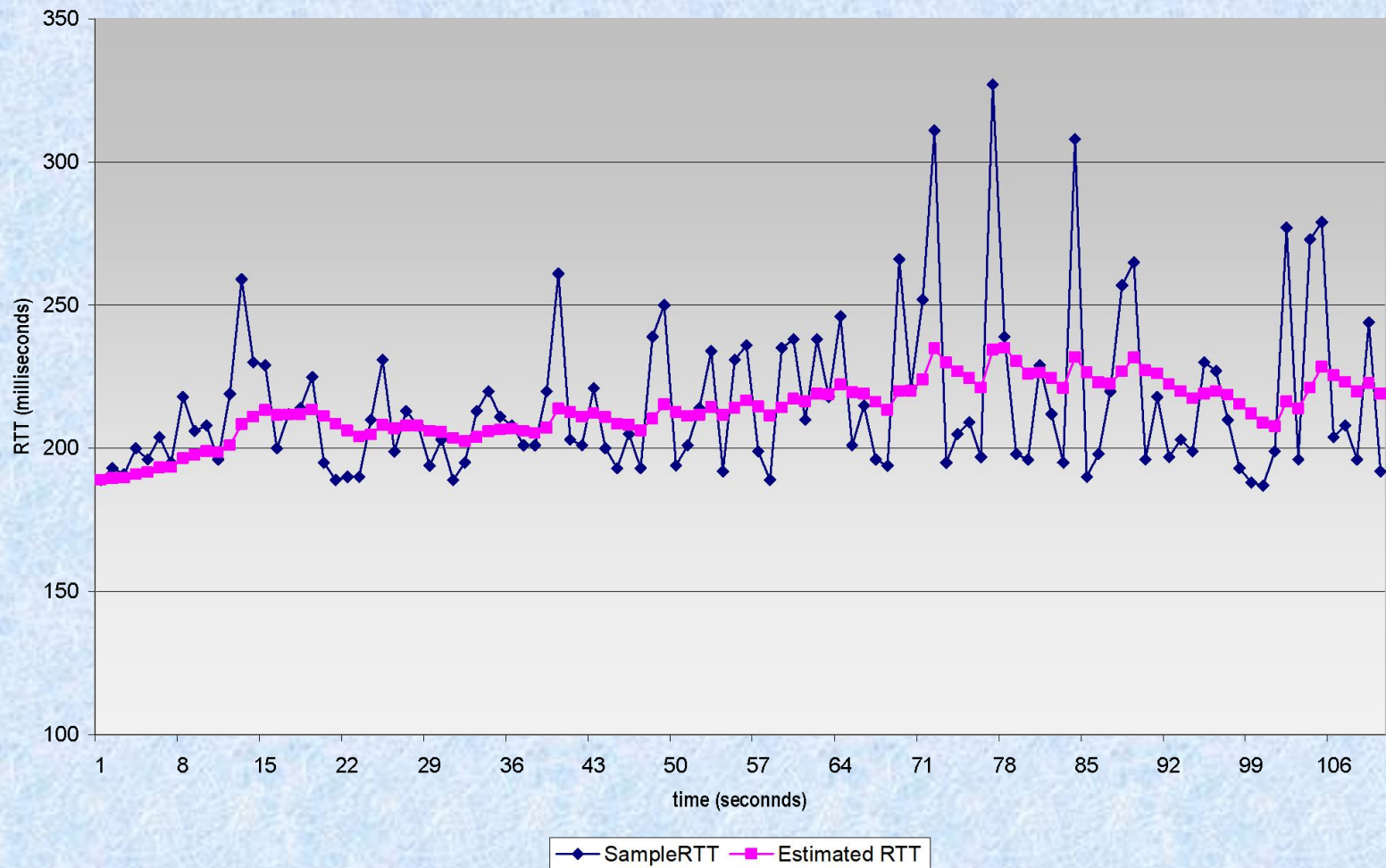
TCP往返时延估计与超时 (续)

$$\text{EstimatedRTT} = (1 - \alpha) * \text{EstimatedRTT} + \alpha * \text{SampleRTT}$$

- 指数加权移动平均(Exponential weighted moving average)
- 过去的样本指数级衰减来产生影响
- 典型值: $\alpha = 0.125$

RTT估计的例子

RTT: gaia.cs.umass.edu to fantasia.eurecom.fr



TCP往返时延估计与超时 (续)

设置超时间隔

□ EstimatedRTT 加 “安全余量”

○ SampleRTT大变化-> 更大的安全余量

□ 首先估算EstimatedRTT与SampleRTT之间差值有多大：

$$\text{DevRTT} = (1-\beta) * \text{DevRTT} + \beta * |\text{SampleRTT} - \text{EstimatedRTT}|$$

(典型地, $\beta = 0.25$)

然后估算超时值:

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 * \text{DevRTT}$$

初始TimeoutInterval为1s, 当出现超时后, TimeoutInterval加倍

第3章 要点

- 3.1 传输层服务
- 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

TCP 可靠数据传输

- ❑ TCP在IP不可靠服务的基础上创建可靠数据传输服务
- ❑ 流水线发送报文段
- ❑ 累计确认
- ❑ TCP使用单个重传计时器
- ❑ 重传被下列事件触发:
 - 超时事件
 - 重复ACK
- ❑ 先考虑简化的TCP发送方:
 - 忽略重复ACK
 - 忽略流量控制，拥塞控制

TCP 发送方事件

1.从应用层接收数据:

□ 创建报文段

- 序号是报文段中第一个数据字节的数据流编号

□ 如果计时器未启动, 启动之(考虑计时器用于最早的没有确认的报文段)

- 超时时间间隔:

$\text{TimeOutInterval} =$
 $\text{EstimatedRTT} +$
 $4 * \text{DevRTT}$

2.超时:

□ 重传导致超时的报文段

□ 重新启动计时器

3.收到确认(累积确认):

□ 如果确认了先前未被确认的报文段

□ 更新SendBase

□ 如果还有未被确认的报文段, 重新启动计时器


```
NextSeqNum = InitialSeqNum
```

```
SendBase = InitialSeqNum
```

```
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 layer

NextSeqNum = NextSeqNum + length(data)

break;

event: timer timeout

retransmit not-yet-acknowledged segment with

smallest sequence number

start timer

break;

event: ACK received, with ACK field value of y

if (y > SendBase) { /* 累计确认到y */

SendBase = y

if (there are currently not-yet-acknowledged segments)

start timer

}

break;

} /* end of loop forever */ P160图3-33

TCP 发送方 (简化的)

注释:

- **SendBase-1:** 上次累计的已确认字节

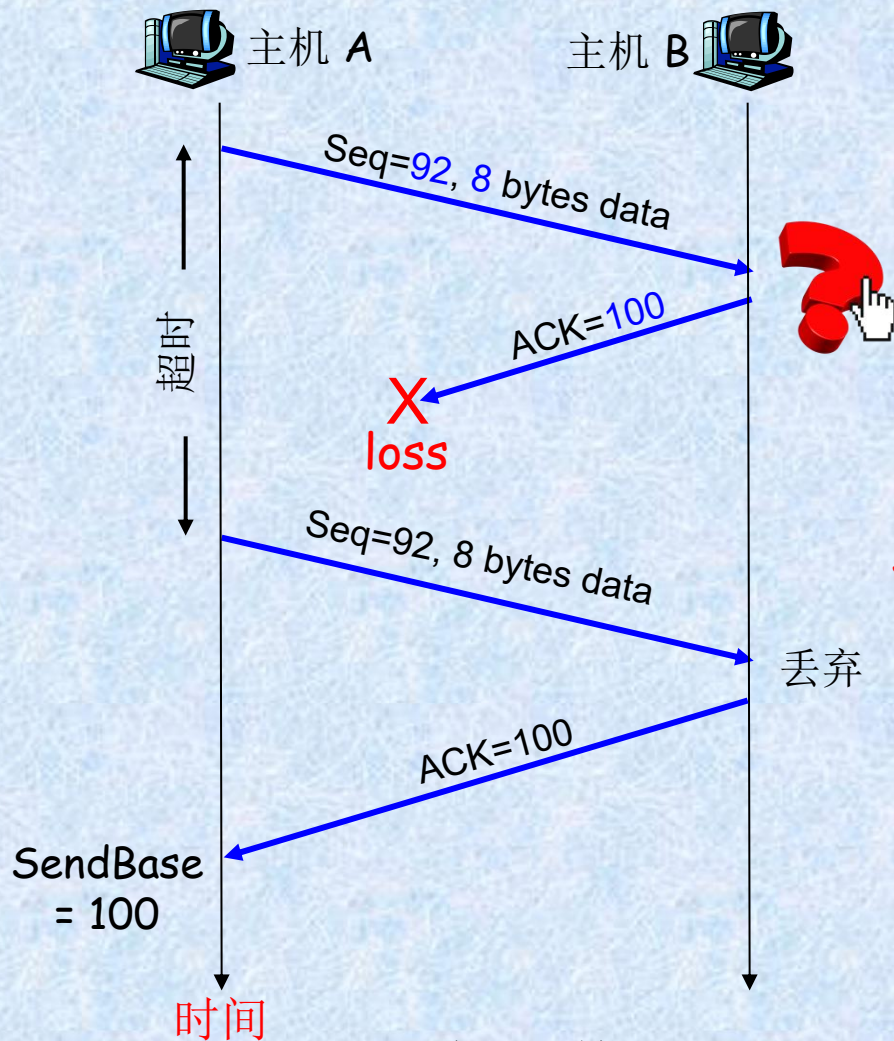
例如:

- **SendBase-1 = 71;**

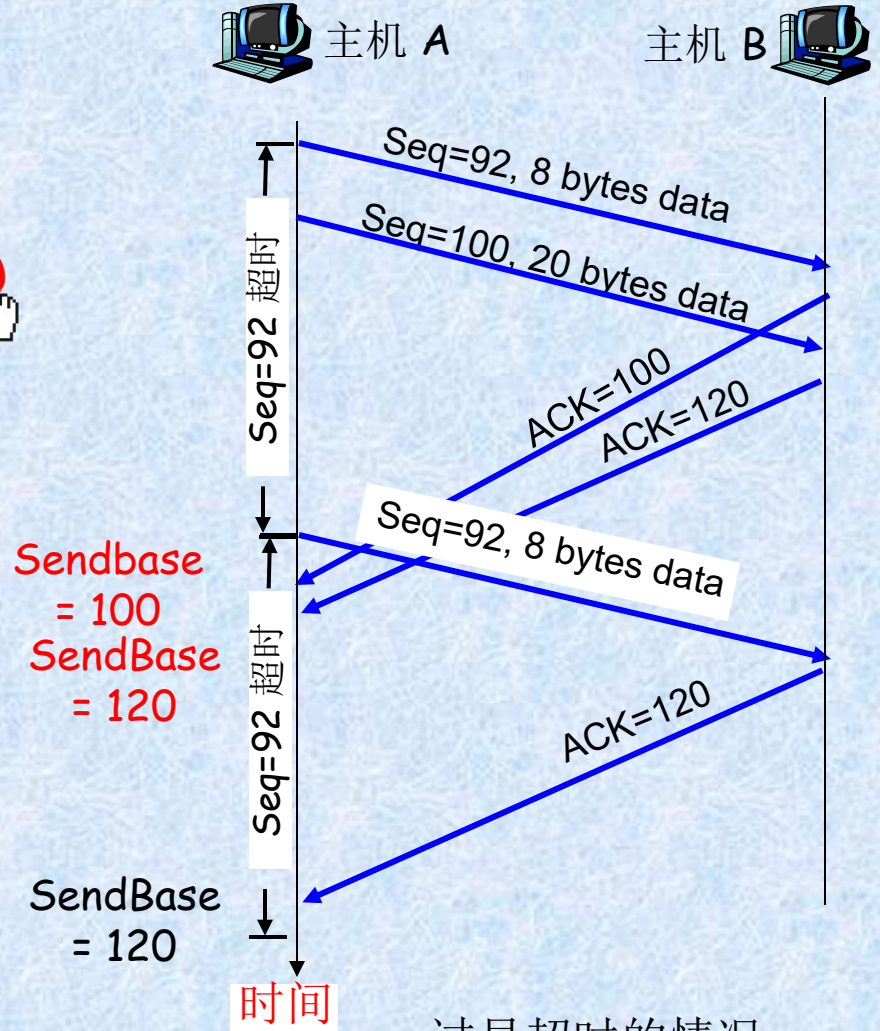
y = 73, 因此接收方期待**73**;

y > SendBase, 因此新数据被确认

TCP: 重传的情况

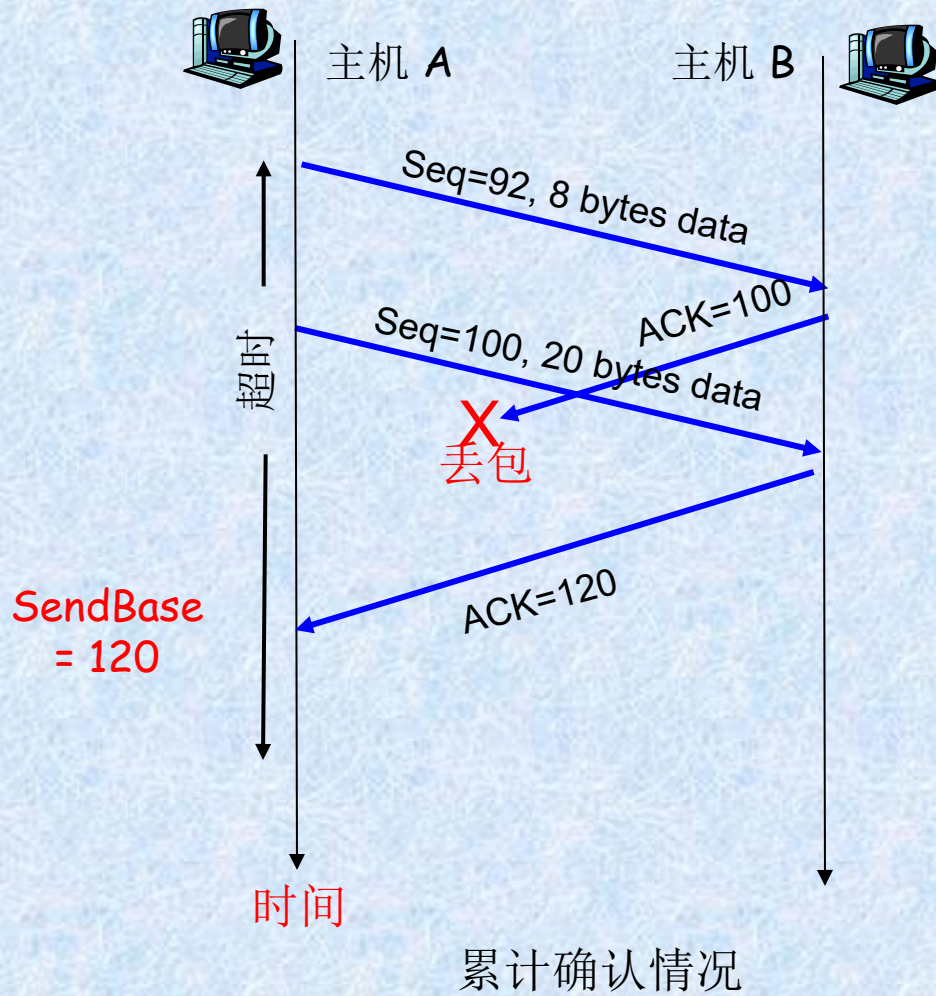


丢失确认的情况



过早超时的情况

TCP 重传情况(续)

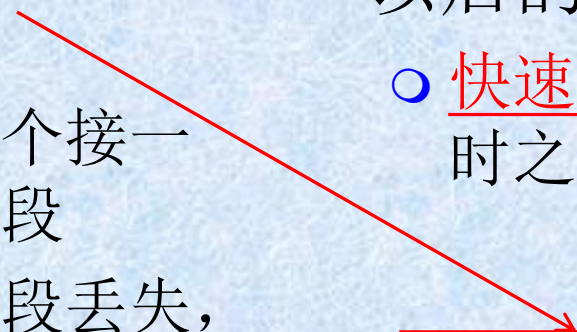


思考题: P188 R15,
P192 P26,P27

快速重传

- ❑ 超时间隔常常相对较长:
 - 重传丢失报文段以前有较长时延
- ❑ 通过冗余ACK, 检测丢失的报文段
 - 发送方经常一个接一个的发送报文段
 - 如果中间报文段丢失, 将会收到很多重复ACK

- ❑ 如果对相同数据, 发送方收到3个冗余ACK, 假定被确认的报文段以后的报文段丢失了:
 - 快速重传: 在定时器超时之前重传



何时会产生冗余的
ACK?

TCP ACK 产生 [RFC 1122, RFC 2581]

接收方事件	TCP 接收方行为
所期望序号的报文段按序到达。 所有在期望序号及以前的数据都已经被确认	延迟的ACK。对另一个按序报文段的到达最多等待500 ms。如果下一个按序报文段在这个时间间隔内没有到达，则发送一个ACK
有期望序号的报文段按序到达。 前一个按序报文段等待发送ACK	立即发送单个累积ACK，以确认两个按序报文段
比期望序号大的失序报文段到达，检测出数据流中的间隔。	立即发送冗余ACK，指明下一个期待字节的序号（也就是间隔的低端字节序号）
部分或者完全填充已接收到数据间隔的报文段到达	倘若该报文段起始于间隔的低端，则立即发送ACK

快速重传算法:

事件: 收到ACK, ACK 域的值为 y

```
if ( $y > \text{SendBase}$ ) {
```

```
     $\text{SendBase} = y$ 
```

```
    if (当前还有没有确认的报文段)
```

```
        启动定时器
```

```
}
```

```
else {
```

```
    值为  $y$  的重复确认的次数加1
```

```
    if (值为  $y$  的重复确认的计数 = 3) {
```

```
        重传序号为  $y$  的报文段
```

```
    }
```

```
break;
```

对已经确认的报文段
收到一个重复ACK

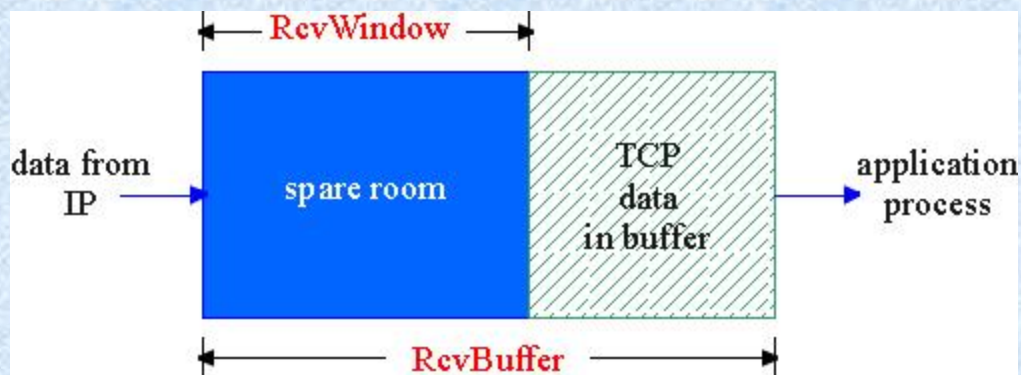
快速重传

第3章 要点

- 3.1 传输层服务
- 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

TCP 流量控制

- ❑ TCP连接的接收方有1个接收缓冲区:



- ❑ 应用进程可能从接收缓冲区读数据缓慢

流量控制

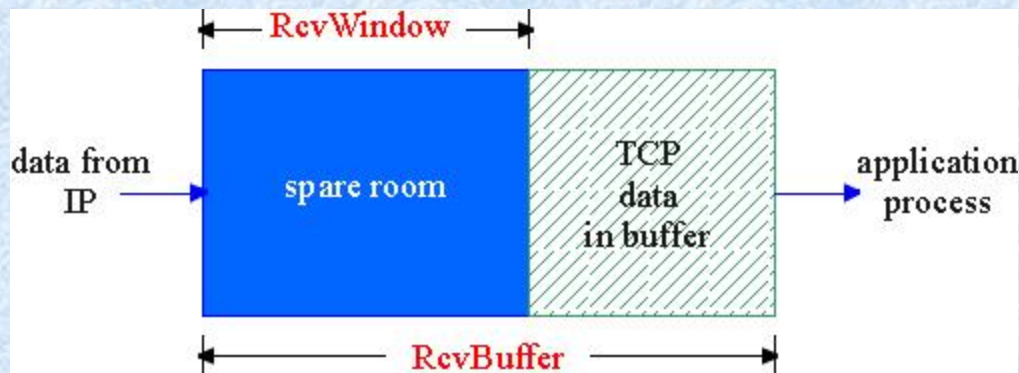
发送方不能发送太多、太快的数据让接收方缓冲区溢出

- ❑ 匹配速度服务: 发送速率需要匹配接收方应用程序的提取速率



TCP流控: 工作原理

思考题: R14



(假设 TCP 接收方丢弃失序的报文段)

□ 发送方维护接受窗口 → 接收方缓冲区的剩余空间

RcvWindow

= RcvBuffer - [LastByteRcvd - LastByteRead]

- 接收方在报文段的接收窗口字段通告其接收缓冲区的剩余空间
- 发送方要限制未确认的数据不超过 RcvWindow

LastByteSent - LastByteAcked ≤ RcvWindow

- 保证接收缓冲区不溢出

最后一个已发送的字节编号 - 最后一个被确认的字节编号
= 已经发送还未被确认的数据量

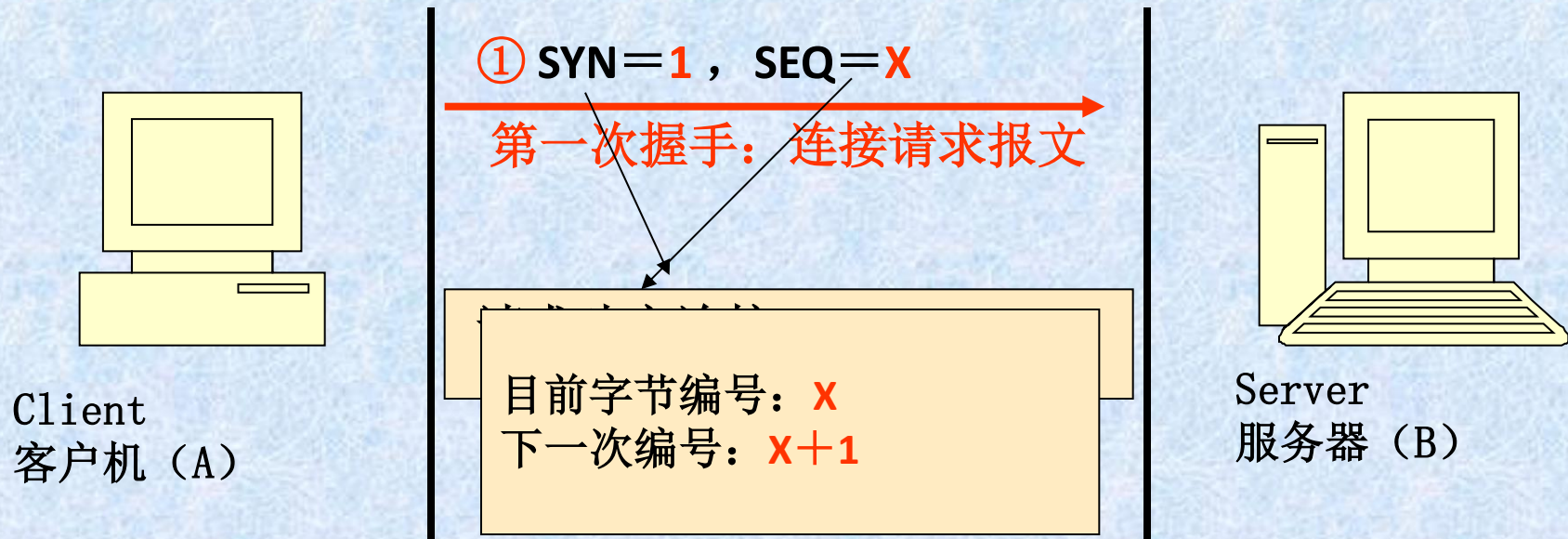
第3章 要点

- 3.1 传输层服务
- 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

TCP连接管理

- TCP是面向连接的协议，TCP连接的建立和释放是每次TCP传输中必不可少的过程。
- TCP的传输连接包括三个状态
 - ◆ 连接建立
 - ◆ 数据传输
 - ◆ 连接释放

建立连接（三次握手）



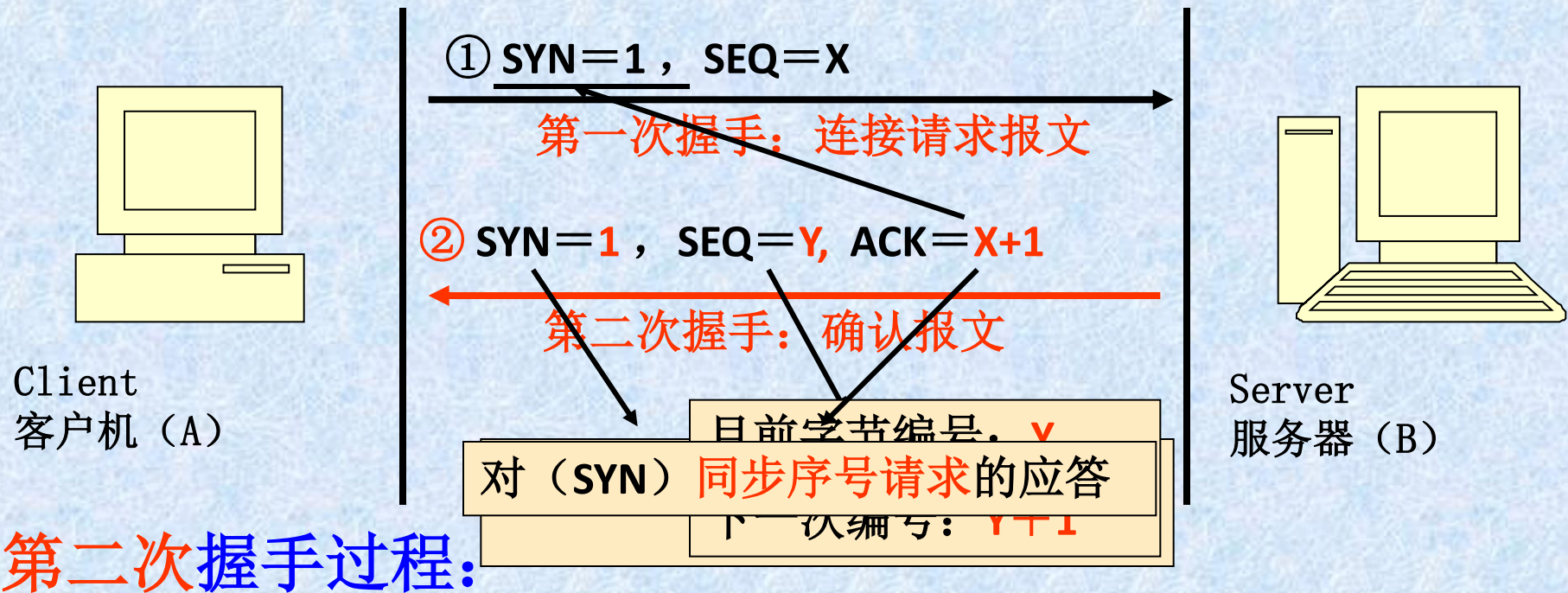
第一次握手过程：

注：

SYN：同步序列编号(Synchronize Sequence Number)

SEQ：序列号(Sequence Number)，表示当前数据传输字节的编号。

建立连接（三次握手）



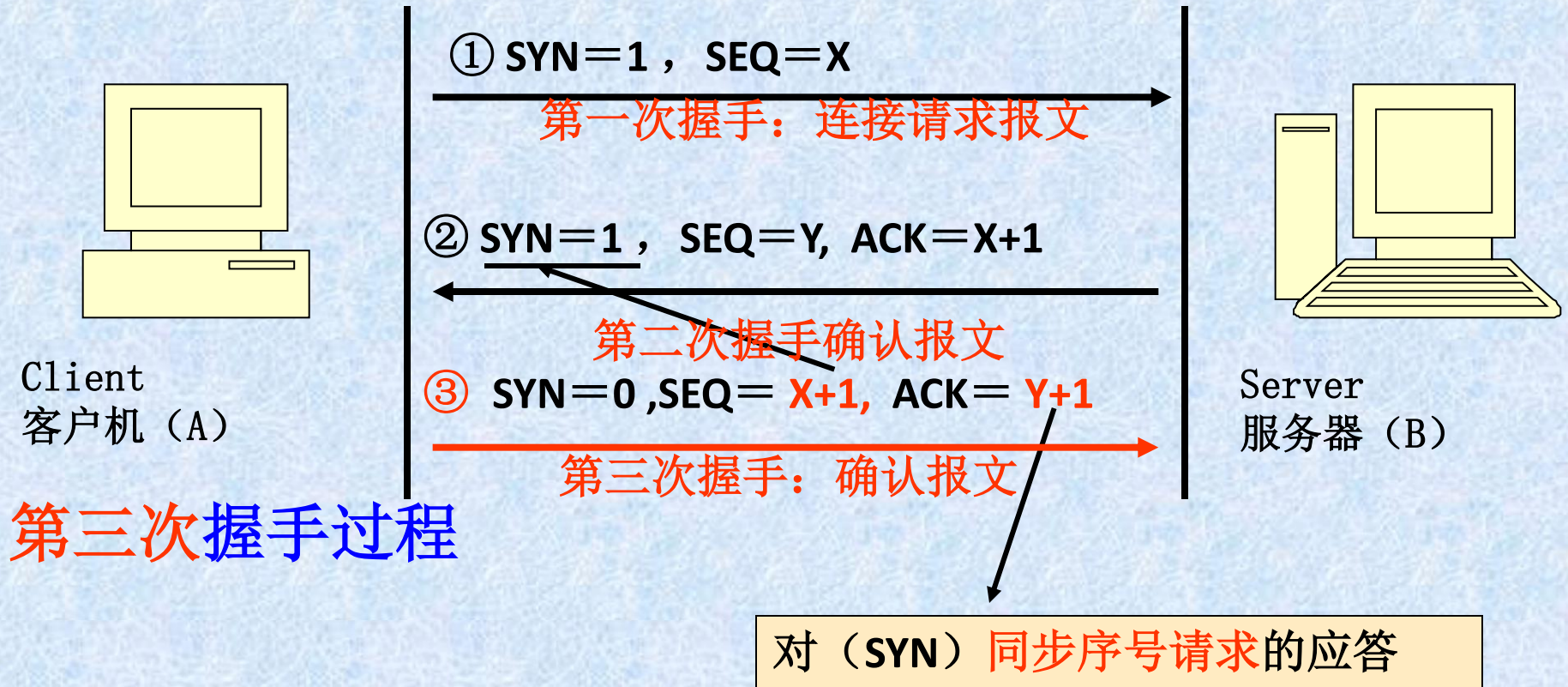
注：

SYN : 同步序列编号 (Synchronize Sequence Numbers)

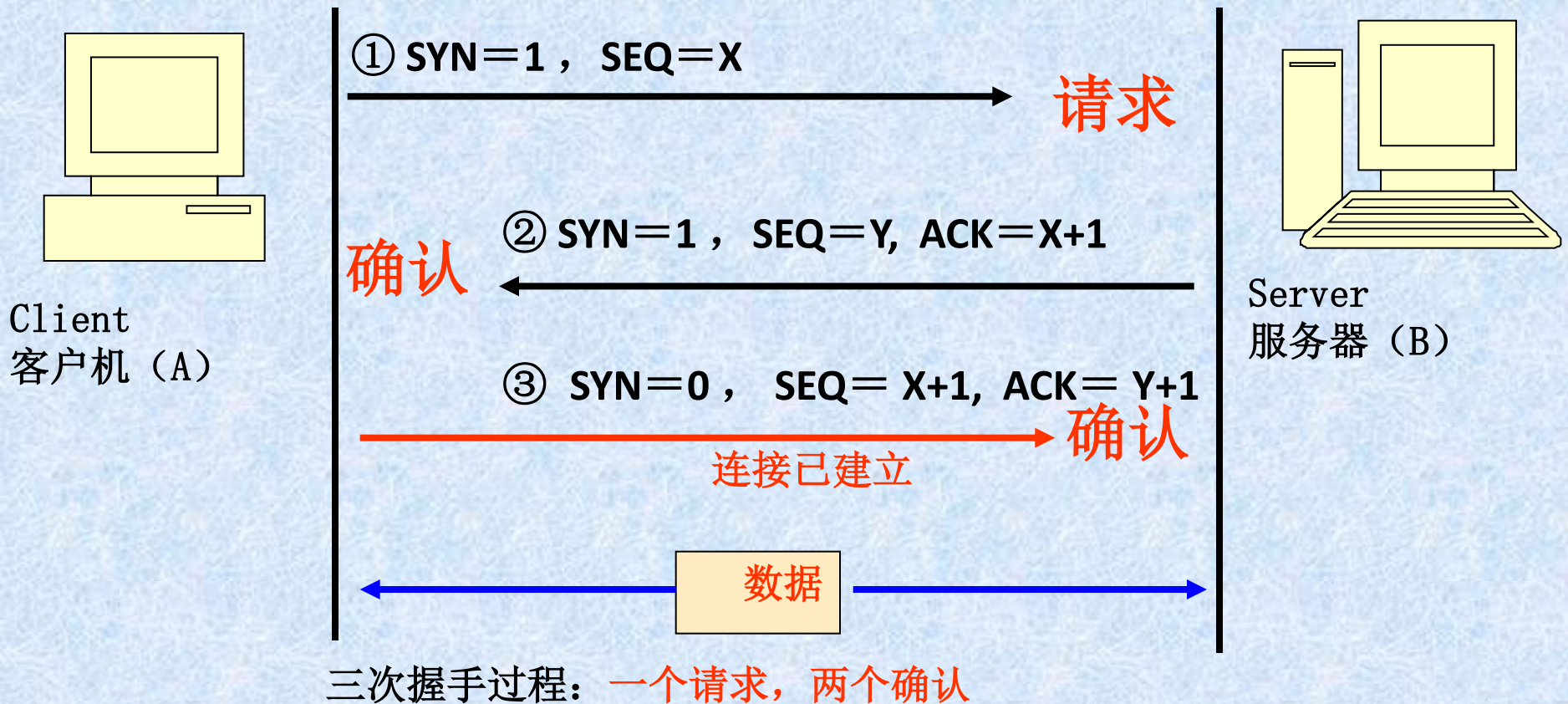
SEQ : 序列号 (Sequence Number)

ACK : 确认编号 (Acknowledgement Number)

建立连接（三次握手）



建立连接（三次握手）



建立连接

三次握手:

步骤 1: 客户机向服务器发送 TCP SYN报文段

- 指定初始序号
- 没有数据

步骤 2: 服务器收到SYN报文段, 用SYN ACK报文段回复

- 服务器为该连接分配缓冲区和变量
- 指定服务器初始序号

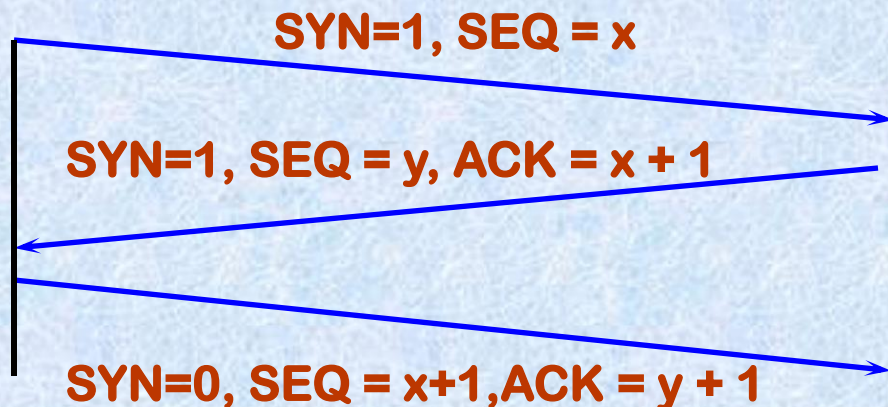
步骤 3: 客户机接收到 SYN ACK, 用ACK报文段回复,可能包含数据



客户机



服务器



课后阅读:
P168-SYN
泛洪攻击和
预防方法

释放连接

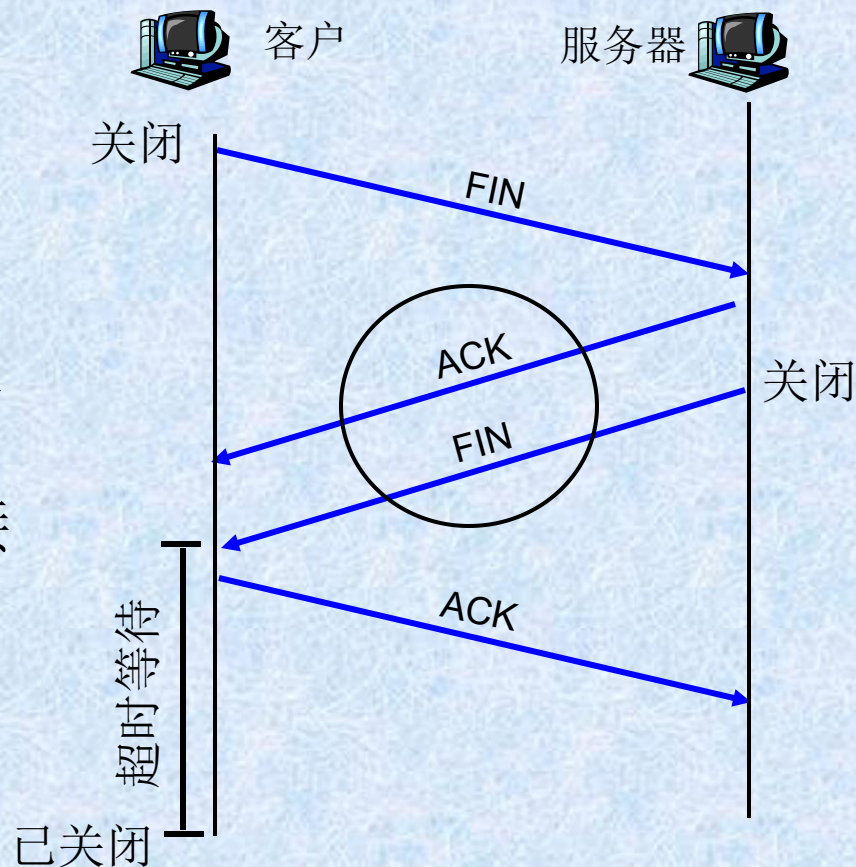
步骤 1: 客户机向服务器发送TCP
FIN控制报文段

步骤 2: 服务器收到FIN，用ACK
回答。关闭连接，发送FIN

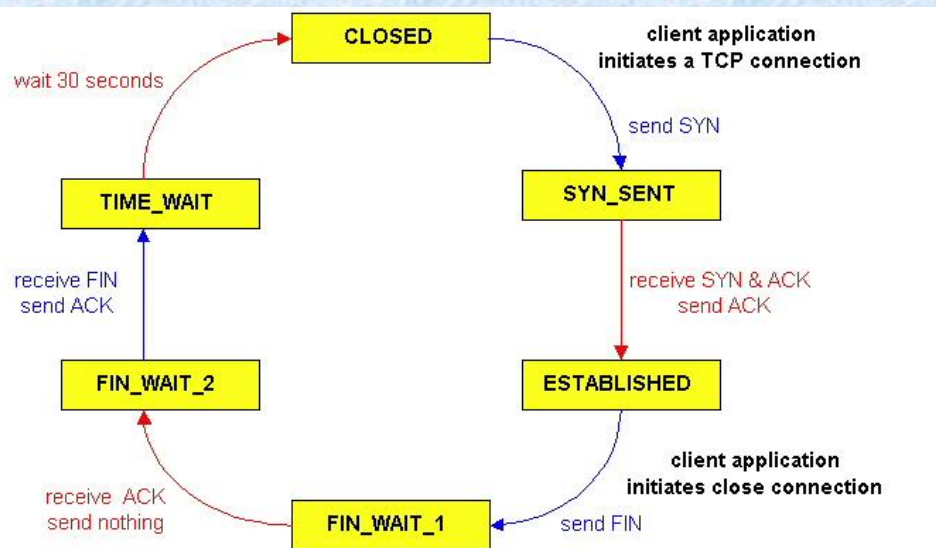
步骤 3: 客户机收到FIN, 用ACK答

- 进入“超时等待” - 将对接收到的FIN进行确认

步骤 4: 服务器接收ACK，连接关
闭

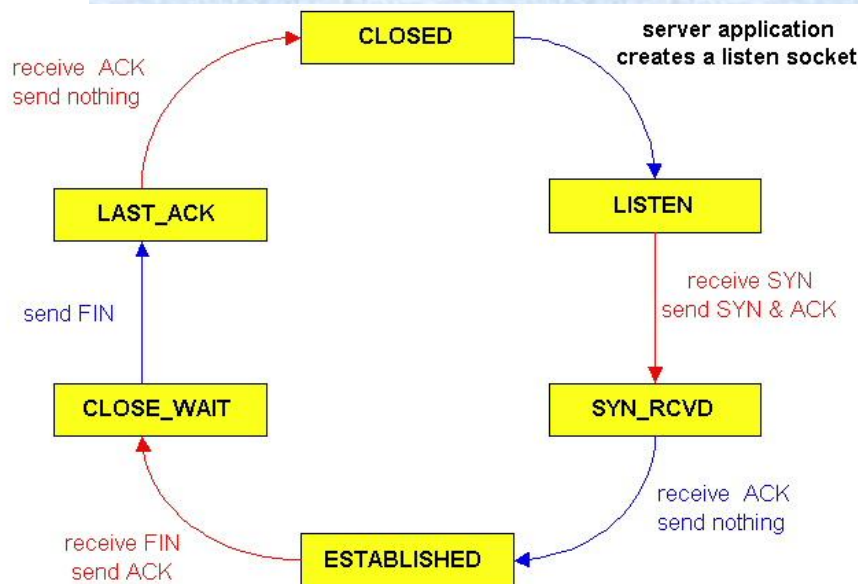


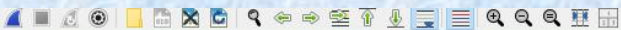
TCP 连接管理 (续)



P168图3-41
TCP 客户生命周期

P169图3-42
TCP 服务器生命周期

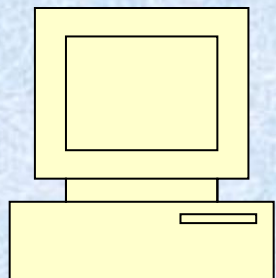




ip.addr == 128.119.245.12 and tcp.port == 10891

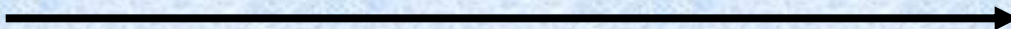
No.	Time	Source	Destination	Prot.	Len	Info
20	2.088338	192.168.31.67	128.119.245.12	TCP	66	10891 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
24	2.316071	128.119.245.12	192.168.31.67	TCP	66	80 → 10891 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1200 SACK_PERM WS=128
25	2.316148	192.168.31.67	128.119.245.12	TCP	54	10891 → 80 [ACK] Seq=1 Ack=1 Win=65792 Len=0
464	7.317071	192.168.31.67	128.119.245.12	TCP	54	10891 → 80 [FIN, ACK] Seq=1 Ack=1 Win=65792 Len=0
610	7.545640	128.119.245.12	192.168.31.67	TCP	60	80 → 10891 [ACK] Seq=1 Ack=2 Win=29312 Len=0
611	7.545640	128.119.245.12	192.168.31.67	TCP	60	80 → 10891 [FIN, ACK] Seq=1 Ack=2 Win=29312 Len=0
612	7.545708	192.168.31.67	128.119.245.12	TCP	54	10891 → 80 [ACK] Seq=2 Ack=2 Win=65792 Len=0

练习

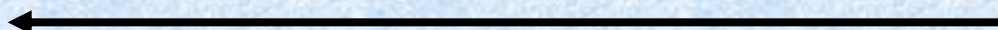


Client
客户机 (A)

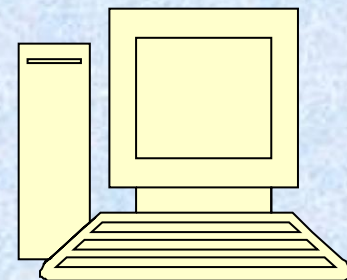
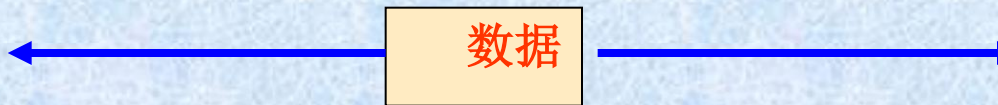
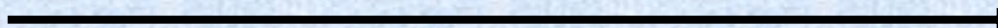
① $\text{SYN}=?$, $\text{SEQ}=1000$



② $\text{SYN}=?$, $\text{SEQ}=?$, $\text{ACK}=?$



③ $\text{SYN}=?$, $\text{SEQ}=?$, $\text{ACK}=2002$



Server
服务器 (B)

三次握手过程：一个请求，两个确认

Answer:

$\text{SYN}=1$, $\text{SEQ}=1000$

$\text{SYN}=1$, $\text{SEQ}=2001$, $\text{ACK}=1001$

$\text{SYN}=0$, $\text{SEQ}=1001$, $\text{ACK}=2002$

1. 对于滑动窗口协议，如果分组序号采用3比特编号，发送窗口大小为5，则接收窗口最大是()

☐ A 2

☒ B 3

☐ C 4

☐ D 5

提交

2、若大小为12B的应用层数据分别通过1个UDP数据报和1个TCP段传输，则该UDP数据报和TCP段实现的有效载荷（应用层数据）最大传输效率分别是

- ☐ A 37.5%,16.7%
- ☐ B 37.5%,37.5%
- ☐ C 60.0%,16.7%
- ☒ D 60.0%,37.5%

提交

3、使用http协议访问，发送的http文件大小为1MSS，包含一个大小为3MSS的图片，与对应服务器的RTT=10ms，此时已完成域名解析，求从tcp连接开始到完整收到内容所需时间为：

- ☐ A 30ms
- ☐ B 40ms
- ☒ C 50ms
- ☐ D 60ms

提交

4、假设主机甲采用停-等协议向主机乙发送数据帧，数据帧长与确认帧长均为**1000B**，数据传输速率是**10kbps**，单项传播延时是**200ms**。则甲的最大信道利用率为

- ☐ A 80%
- ☐ B 66.7%
- ☐ C 44.4%
- ☒ D 40%

提交

答案解析

发送数据帧和确认帧均需

$$1000 \times 8 / 10 \times 1000 = 0.8s$$

一个往返需要 $2 \times 0.2 = 0.4s$

所以甲的信道利用率为

$$0.8 / (0.8 + 0.8 + 0.4) = 0.8 / 2 = 40\%$$

谢谢大家！