



# 传输层 vs. 网络层

## 3.1 传输层服务

- ❖ **网络层**：提供**主机**之间的逻辑通信机制
- ❖ **传输层**：提供**应用进程**之间的逻辑通信机制
  - 位于网络层之上
  - 依赖于网络层服务
  - 对网络层服务进行（可能的）增强

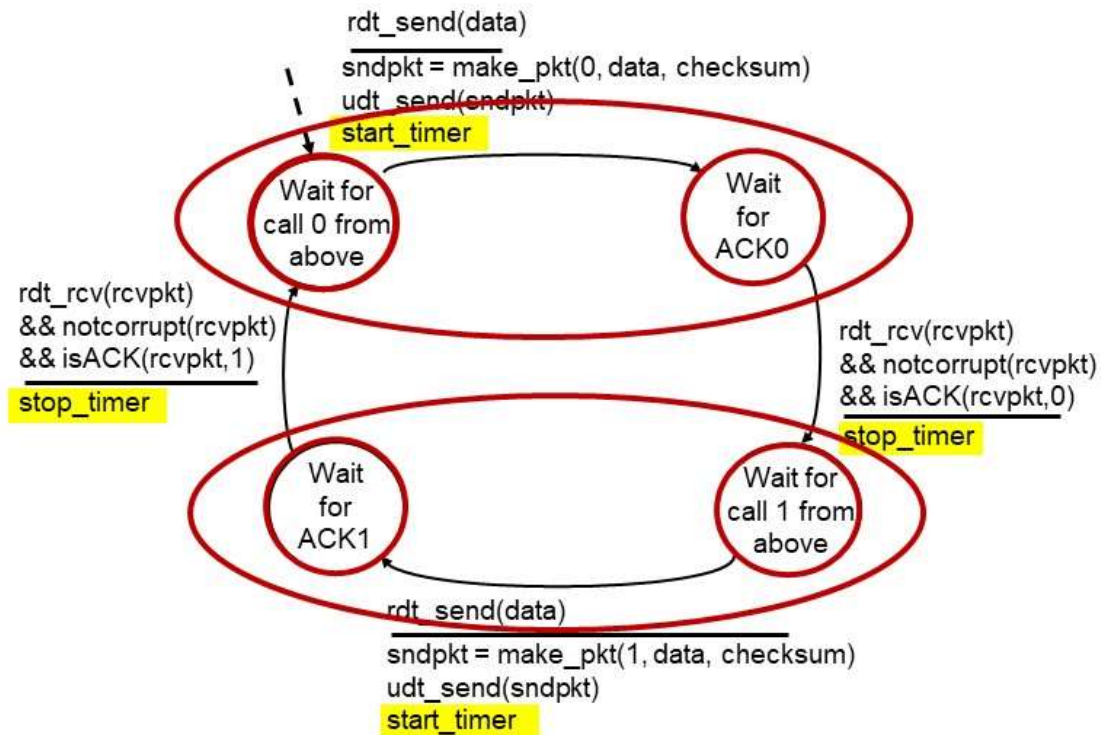
### 家庭类比：

12个孩子给12个孩子发信

- ❖ 应用进程 = 孩子
- ❖ 应用消息 = 信封里的信
- ❖ 主机 = 房子
- ❖ 传输层协议 = 李雷和韩梅梅
- ❖ 网络层协议 = 邮政服务

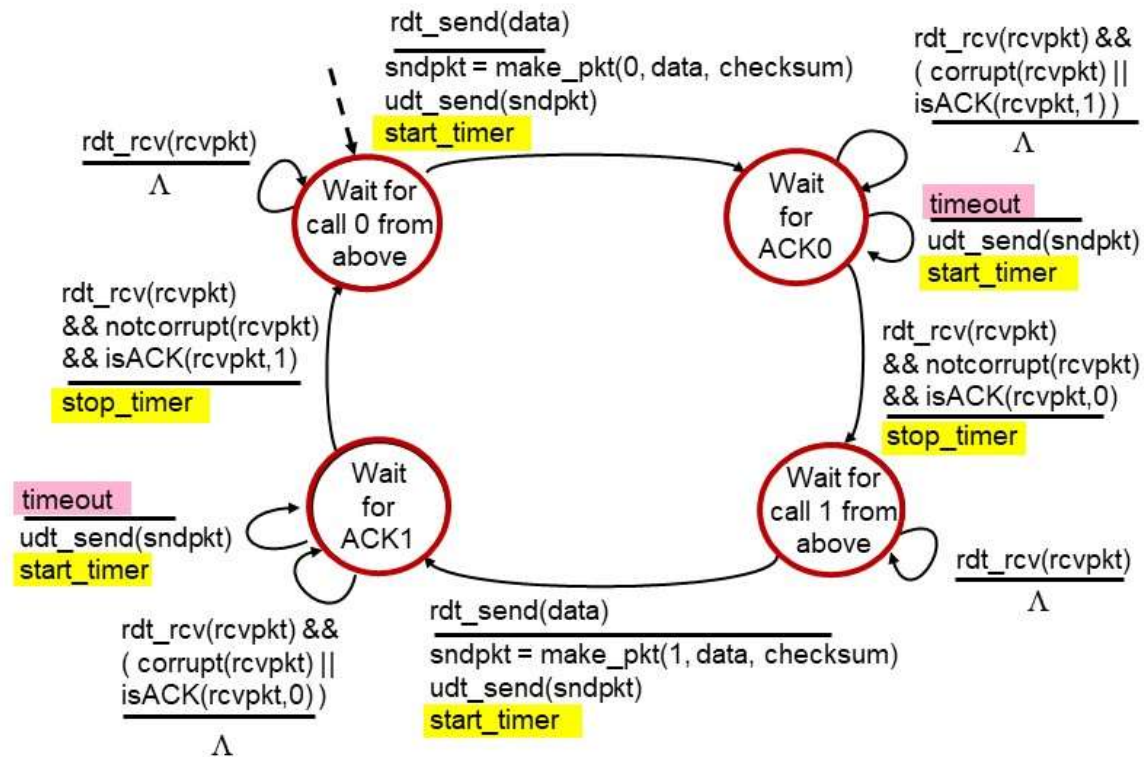


# rdt3.0 发送方



Transport Layer: 3-70

# rdt3.0 发送方



Transport Layer: 3-71



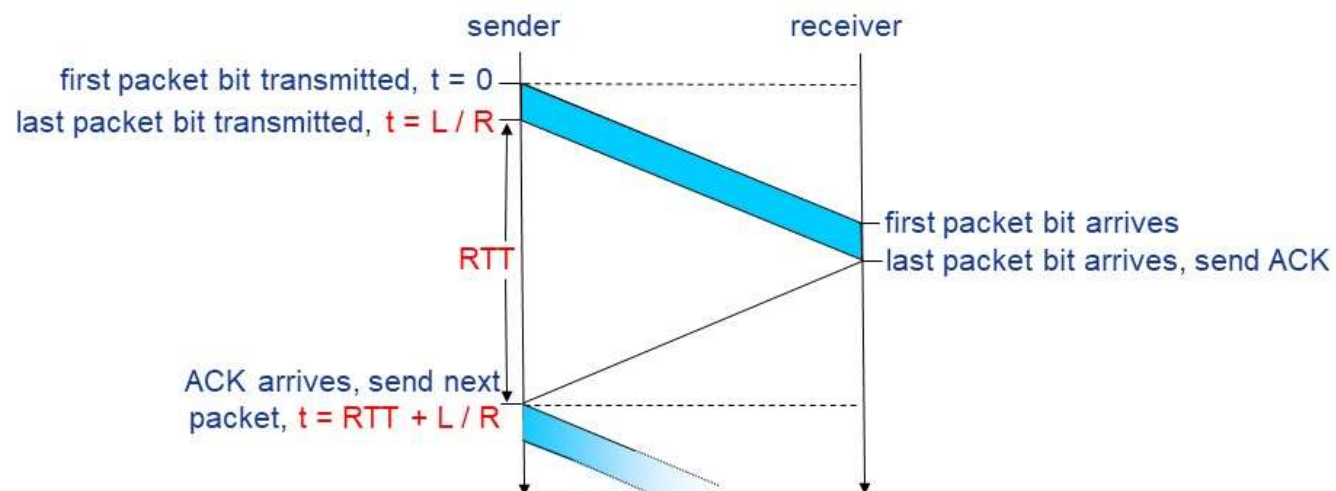
3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

**3.4 可靠数据传输原理**

# Rdt 3.0: 停等操作



$$U_{\text{sender}} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$



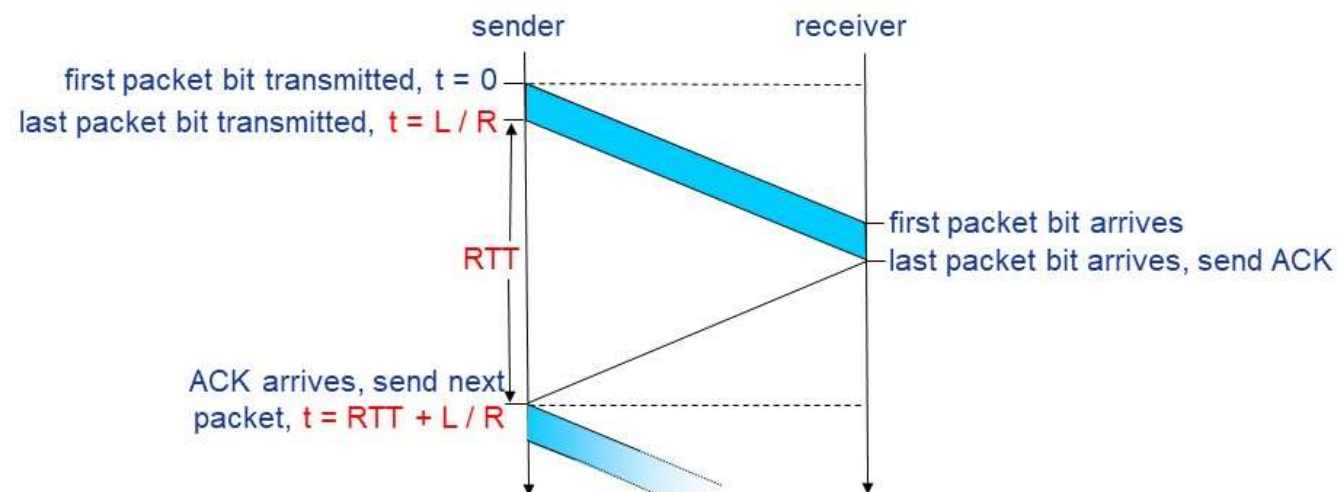
3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

**3.4 可靠数据传输原理**

# Rdt 3.0: 停等操作



$$U_{\text{sender}} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$





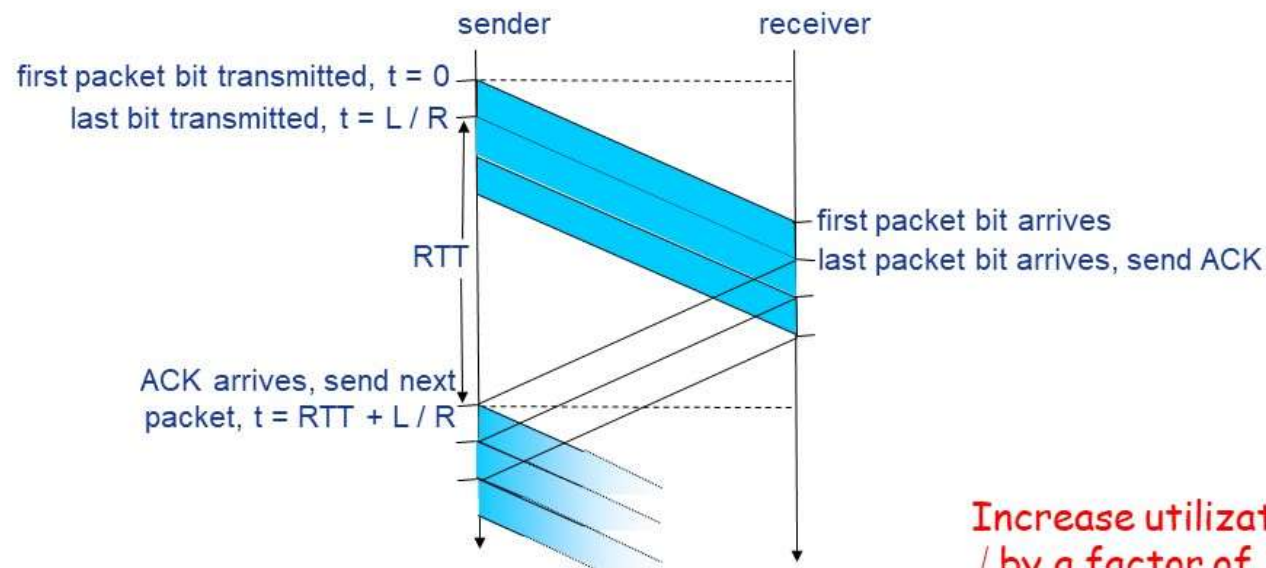
3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

**3.4 可靠数据传输原理**

# 流水线机制：提高资源利用率



$$U_{\text{sender}} = \frac{3 * L / R}{RTT + L / R} = \frac{.024}{30.008} = 0.0008$$

Increase utilization  
by a factor of 3!



3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

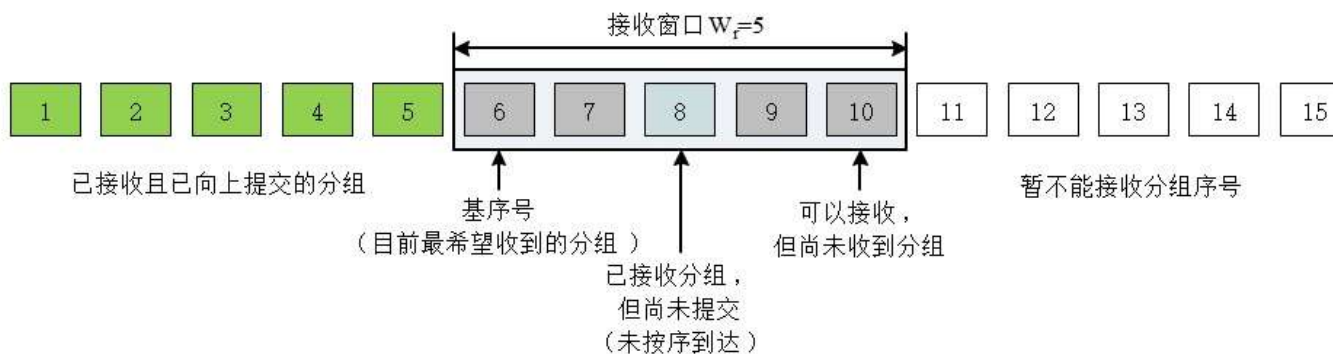
**3.4 可靠数据传输原理**

# 滑动窗口协议

发送方:



接收方:





## 3.1 传输层服务

## 3.2 传输层多路复用/分解

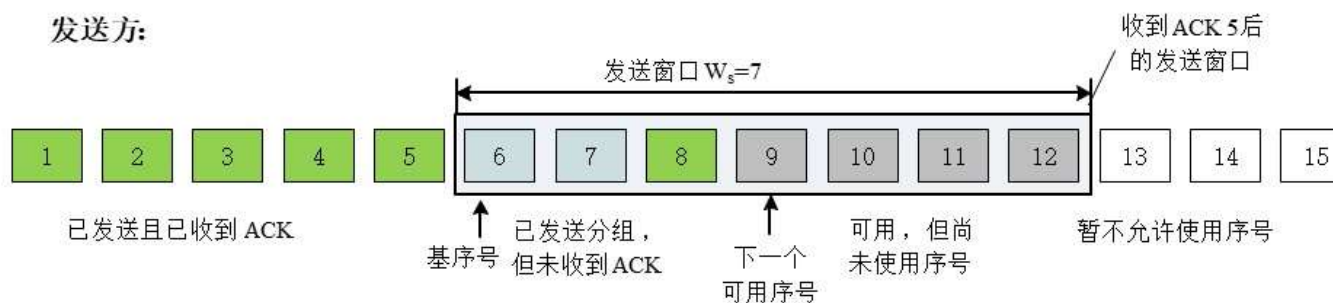
## 3.3 UDP协议

## 3.4 可靠数据传输原理

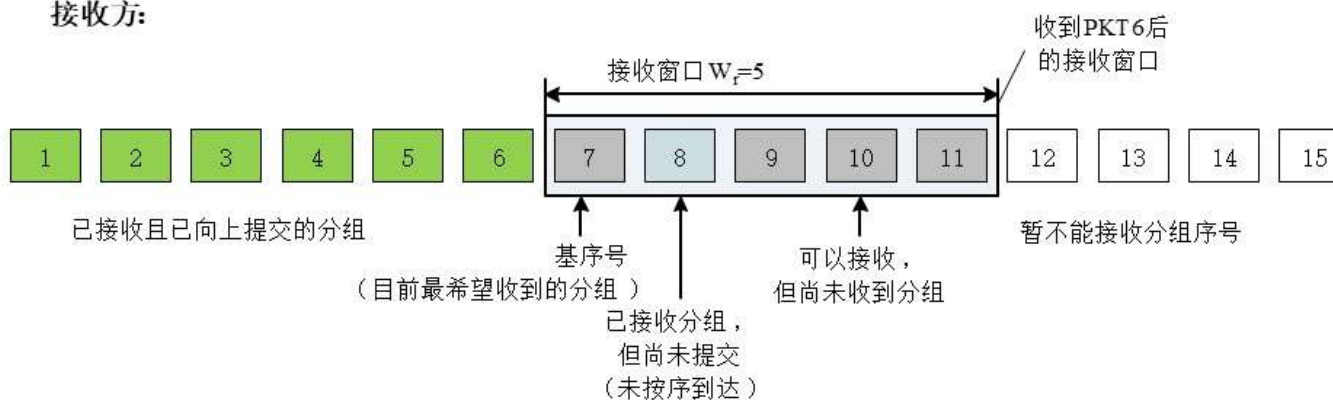


# 滑动窗口协议

发送方:



接收方:







3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

3.4 可靠数据传输原理



# 滑动窗口协议的信道利用率

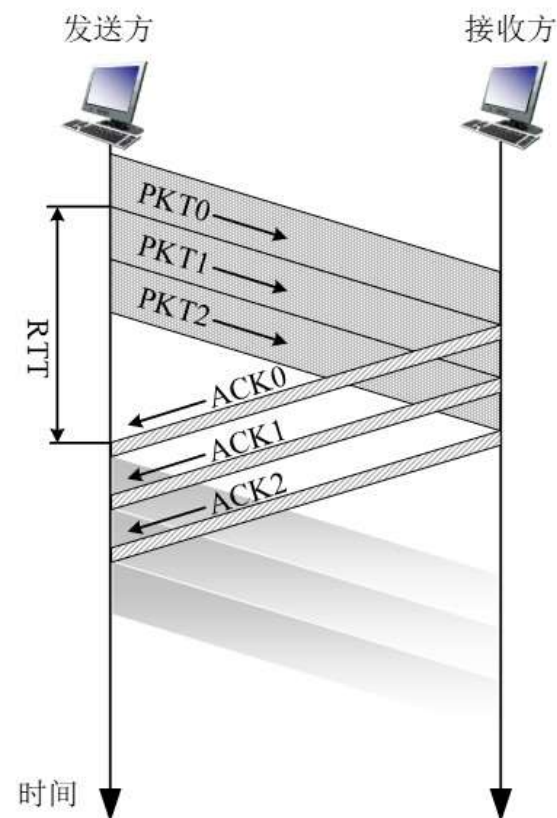
$$U = \frac{W_s \times t_{seg}}{t_{seg} + RTT + t_{ACK}}$$



$$U = \frac{W_s \times L/R}{L/R + 2dp + L'/R}$$



$$U = \frac{W_s \times L}{L + 2dpR + L'}$$





3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

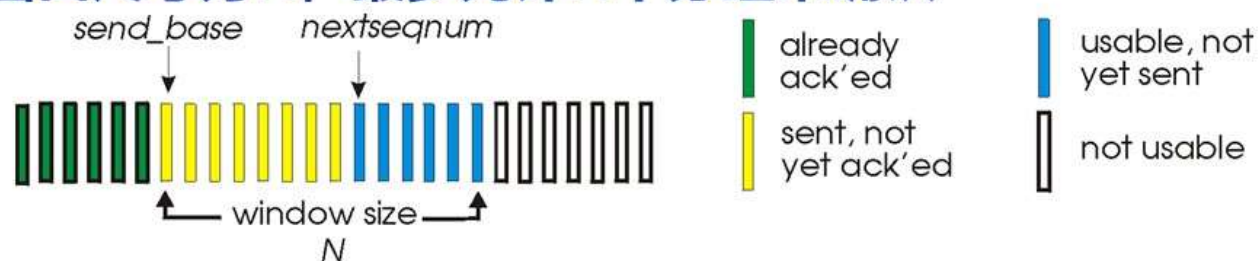
3.4 可靠数据传输原理



# Go-Back-N(GBN)协议: 发送方

❖ 分组头部包含k-bit序列号

❖ 窗口尺寸为N, 最多允许N个分组未确认



❖ ACK(n): 确认到序列号n(包含n)的分组均已被正确接收

- 累积确认
- 可能收到重复ACK

❖ 为“空中”的分组设置计时器(timer)

❖ 超时Timeout(n)事件: 重传序列号大于等于n, 还未收到ACK的所有分组



3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

3.4 可靠数据传输原理



# Go-Back-N(GBN)协议: 接收方

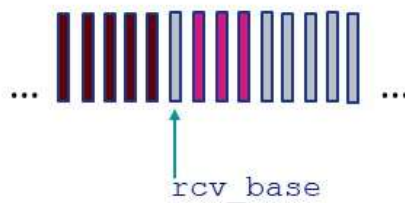
❖ ACK-only: 始终为**目前为止**正确接收的分组发送 ACK, 序号为最高的**in-order seq#**

- 可能会生成重复的 ACK
- 只需要记住 **rcv\_base**

❖ 收到out-of order分组时:

- 可以丢弃 (不缓冲) 或缓冲: 由实现决策
- 重新ACK分组, 使用最高的in-order seq #

Receiver view of sequence number space:



received and ACKed

Out-of-order: received but not ACKed

Not received



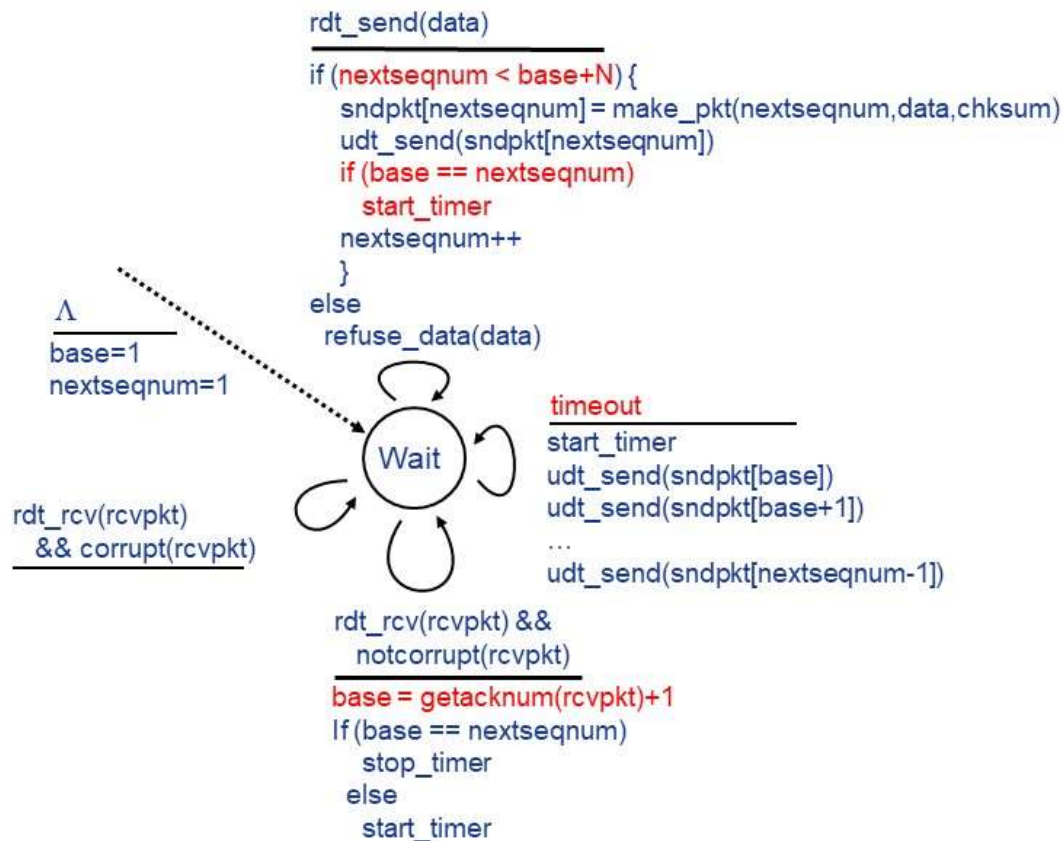
# GBN: 发送方扩展FSM

3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

3.4 可靠数据传输原理



93



## 3.1 传输层服务

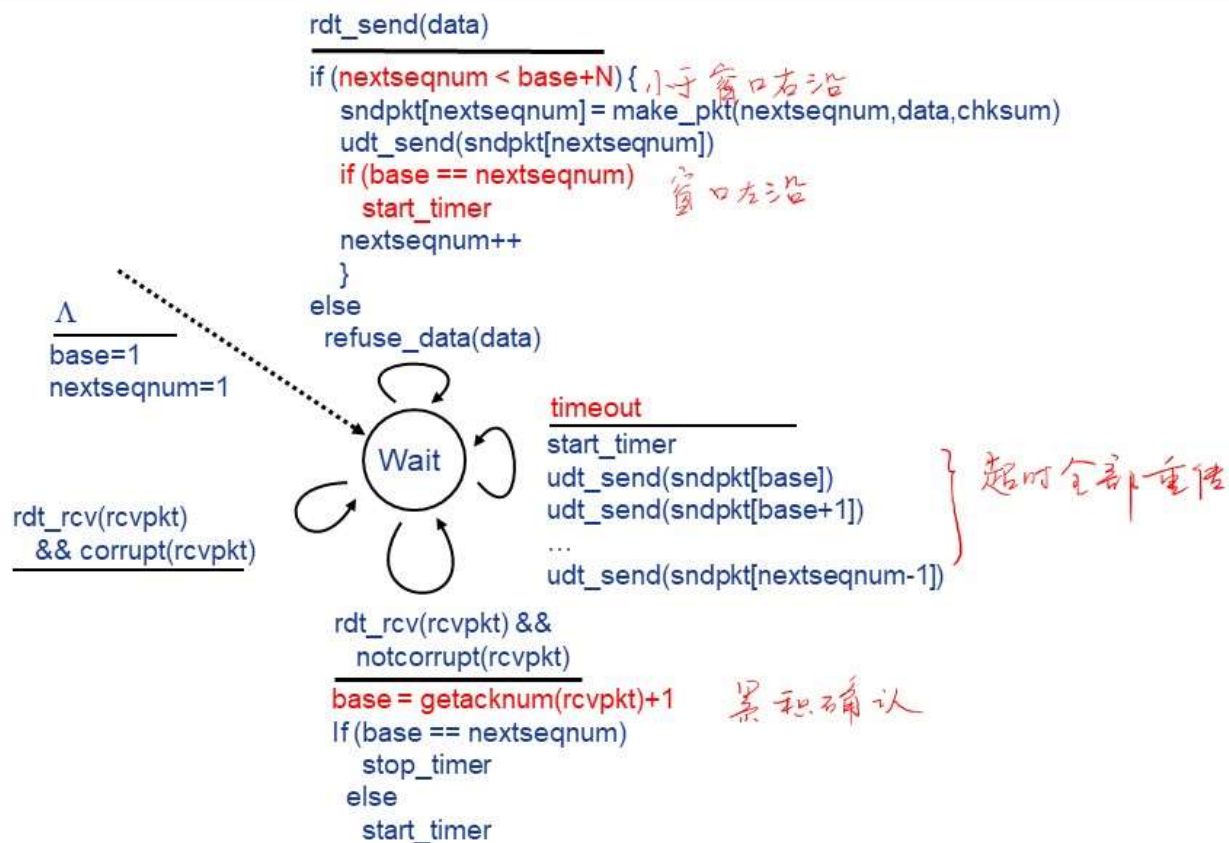
## 3.2 传输层多路复用/分解

## 3.3 UDP协议

## 3.4 可靠数据传输原理



# GBN: 发送方扩展FSM







3.1 传输层服务

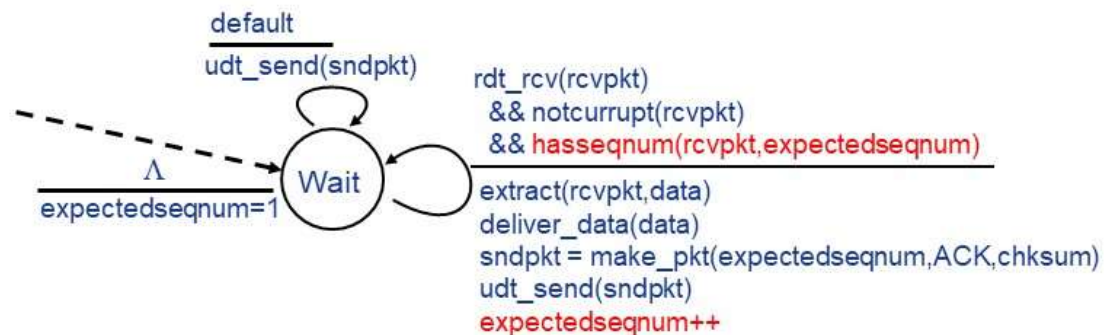
3.2 传输层多路复用/分解

3.3 UDP协议

3.4 可靠数据传输原理



# GBN: 接收方扩展FSM



❖ ACK机制: 发送拥有最高序列号的、已被正确接收的分组的ACK

- 可能产生重复ACK
- 只需要记住唯一的`expectedseqnum`, 即 $W_R=1$

❖ 乱序到达的分组:

- 直接丢弃→接收方没有缓存
- 重新确认序列号最大的、按序到达的分组



3.1 传输层服务

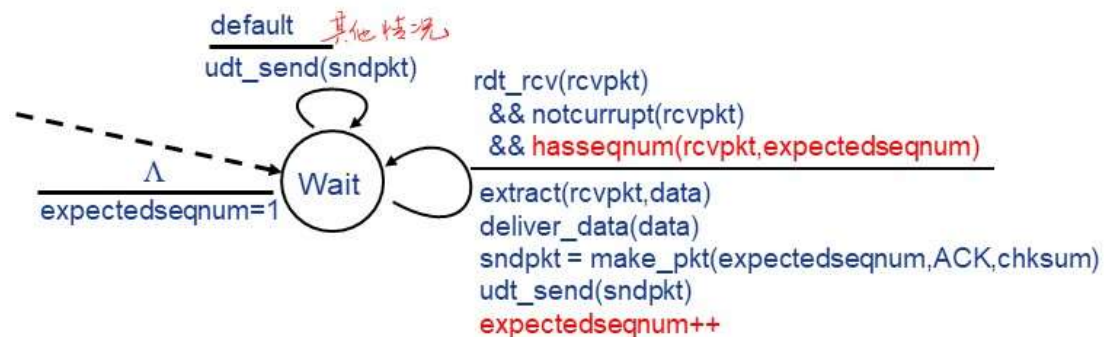
3.2 传输层多路复用/分解

3.3 UDP协议

3.4 可靠数据传输原理



# GBN: 接收方扩展FSM



❖ ACK机制: 发送拥有最高序列号的、已被正确接收的分组的ACK

- 可能产生重复ACK
- 只需要记住唯一的**expectedseqnum**, 即 **$W_R=1$**

❖ 乱序到达的分组:

- 直接丢弃→接收方没有缓存
- 重新确认序列号最大的、按序到达的分组



## 3.1 传输层服务

## 3.2 传输层多路复用/分解

## 3.3 UDP协议

## 3.4 可靠数据传输原理



## GBN示例

sender window (N=4)

0 1 2 3 4 5 6 7 8  
0 1 2 3 4 5 6 7 8  
0 1 2 3 4 5 6 7 8  
0 1 2 3 4 5 6 7 8

0 1 2 3 4 5 6 7 8  
0 1 2 3 4 5 6 7 8

0 1 2 3 4 5 6 7 8  
0 1 2 3 4 5 6 7 8  
0 1 2 3 4 5 6 7 8  
0 1 2 3 4 5 6 7 8

sender

send pkt0  
send pkt1  
send pkt2  
send pkt3  
(wait)

rcv ack0, send pkt4  
rcv ack1, send pkt5

ignore duplicate ACK



pkt 2 timeout

send pkt2  
send pkt3  
send pkt4  
send pkt5

receiver

receive pkt0, send ack0  
receive pkt1, send ack1

receive pkt3, discard,  
(re)send ack1

receive pkt4, discard,  
(re)send ack1

receive pkt5, discard,  
(re)send ack1

rcv pkt2, deliver, send ack2  
rcv pkt3, deliver, send ack3  
rcv pkt4, deliver, send ack4  
rcv pkt5, deliver, send ack5



## 填空题 2分



数据链路层采用后退N帧（GBN）协议，发送方已经发送了编号为0~7的帧。当计时器超时时，若发送方只收到0、2、3号帧的确认，则发送方需要重发的帧数是 [填空1] 个。分别是 [填空2] 。





3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

3.4 可靠数据传输原理



# Selective Repeat协议

## ❖ GBN有什么缺陷？

## ❖ SR协议：

### ❖ 接收方对每个分组单独进行确认

- 设置缓存机制，缓存乱序到达的分组

### ❖ 发送方只重传那些没收到ACK的分组

- 为每个分组设置定时器

### ❖ 发送方窗口

- N个连续的序列号
- 限制已发送且未确认的分组数

### ❖ 接收窗口

- 可以接收的无差错到达的分组序号



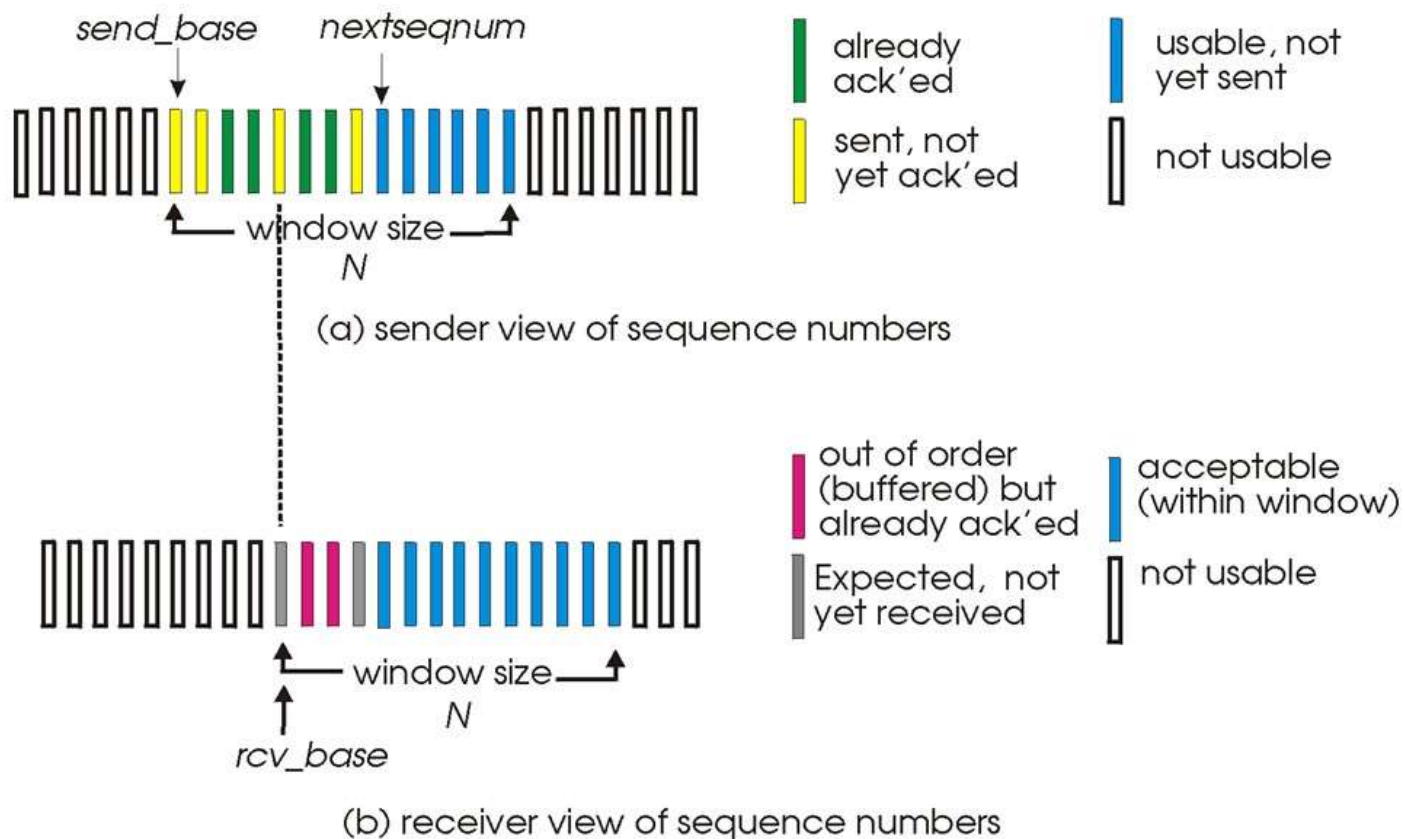
# SR协议：发送方/接收方窗口

3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

3.4 可靠数据传输原理



101



3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

3.4 可靠数据传输原理



# SR协议

**sender****data from above :**

- ❖ if next available seq # in window, send pkt

**timeout(n):**

- ❖ resend pkt n, restart timer

**ACK(n)** in [sendbase, sendbase+N]:

- ❖ mark pkt n as received
- ❖ if n is smallest unACKed pkt, advance window base to next unACKed seq #

**receiver****pkt n** in [rcvbase, rcvbase+N-1]

- ❑ send ACK(n)
- ❑ out-of-order: buffer
- ❑ in-order: deliver (also deliver buffered, in-order pkts), advance window to next not-yet-received pkt

**pkt n** in [rcvbase-N, rcvbase-1]

- ❑ ACK(n)

**otherwise:**

- ❑ ignore

思考：SR协议还可以有其他设计吗？



3.1 传输层服务

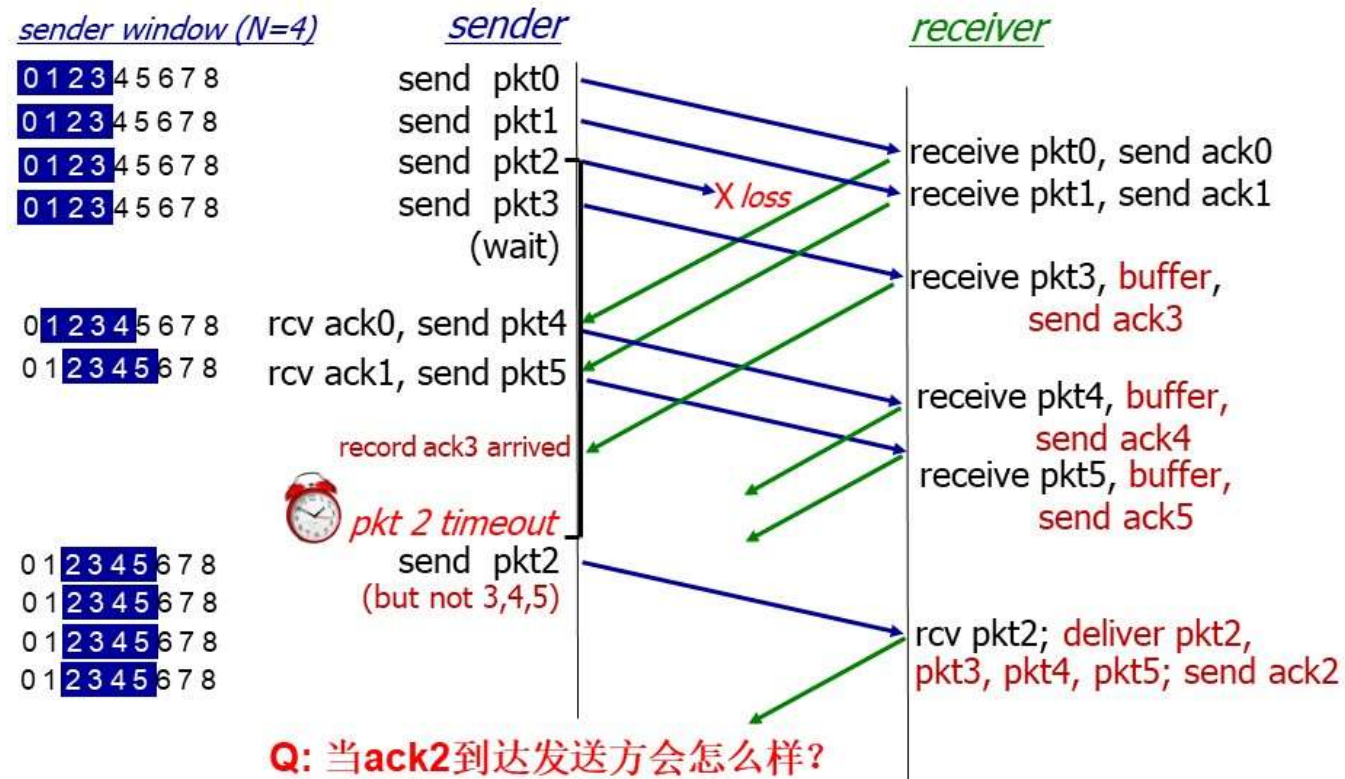
3.2 传输层多路复用/分解

3.3 UDP协议

## 3.4 可靠数据传输原理



## SR协议示例







3.1 传输层服务

3.2 传输层多路复用/分解

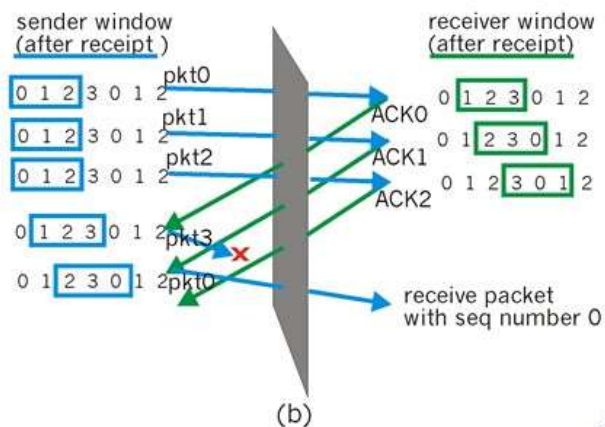
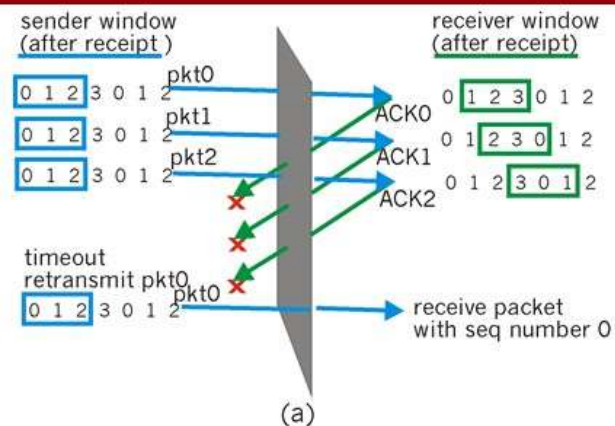
3.3 UDP协议

3.4 可靠数据传输原理



# SR协议：困境

- ❖ 序列号: 0, 1, 2, 3
- ❖ 窗口尺寸: 3
- ❖ 接收方能区分开右侧两种不同的场景吗?
- ❖ (a)中, 发送方重发0号分组, 接收方收到后会如何处理?



.05



# 窗口大小与序号空间的约束条件?

❖ 问题：序列号空间大小与窗口尺寸需满足什么关系?

$$W_s + W_r \leq 2^k$$

▪  $W_s$  发送窗口,  $W_r$  接收窗口,  $k$  序号位数

❖ 对于GBN协议:  $W_r=1$

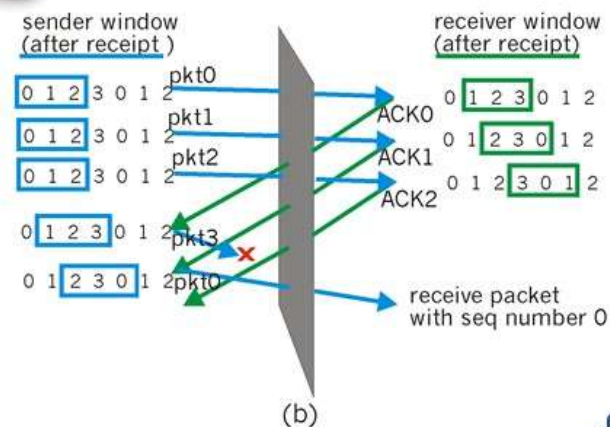
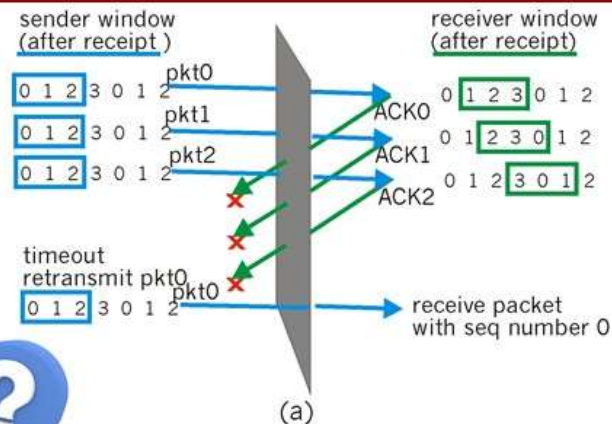
$$W_s \leq 2^k - 1$$

❖ 对于典型的  $W_s=W_r=W$  的SR协议

$$W_s \leq 2^{(k-1)}$$

❖ 对于停-等协议, 即  $W_s=W_r=1$

$$k \geq 1$$







3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

3.4 可靠数据传输原理



# 滑动窗口协议的窗口大小

## ❖ 讨论:

- 1. 滑动窗口协议的窗口大小影响协议哪些性能?
- 2. 哪些因素会影响滑动窗口大小的确定?

## ❖ 性能:

- 信道利用率
- 吞吐率
- .....

## ❖ 因素:

- 序号空间
- 缓存大小
- 流量控制
- 拥塞控制
- .....



107

## 单选题 1分

3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

3.4 可靠数据传输原理



主机甲通过128 kbps卫星链路，采用滑动窗口协议向主机乙发送数据，链路单向传播延迟为250 ms，帧长为1000字节。不考虑确认帧的开销，为使链路利用率不小于80%，帧序号的比特数至少是\_\_\_\_\_。

A 2

B 3

C 4

D 5



## 3.5 TCP协议

刘亚维

108



# TCP概述: RFCs-793, 1122, 1323, 2018, 2581

3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

3.4 可靠数据传输原理

## 3.5 TCP协议



### ❖ 点对点

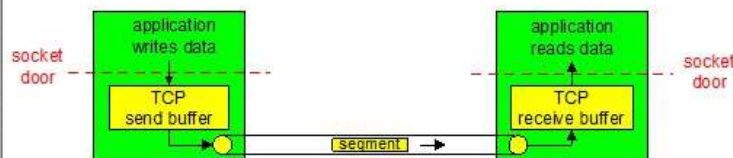
- 一个发送方, 一个接收方

### ❖ 可靠的、按序字节流

### ❖ 流水线机制

- TCP拥塞控制和流量控制  
机制设置窗口尺寸

### ❖ 发送方/接收方缓存



### ❖ 全双工(full-duplex)

- 同一连接中能够传输双向数据流

### ❖ 面向连接

- 通信双方在发送数据之前必须建立连接。
- 连接状态只在连接的两端中维护, 在沿途节点中并不维护状态。
- TCP连接包括: 两台主机上的缓存、连接状态变量、socket等

### ❖ 流量控制机制

### ❖ 拥塞控制





3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

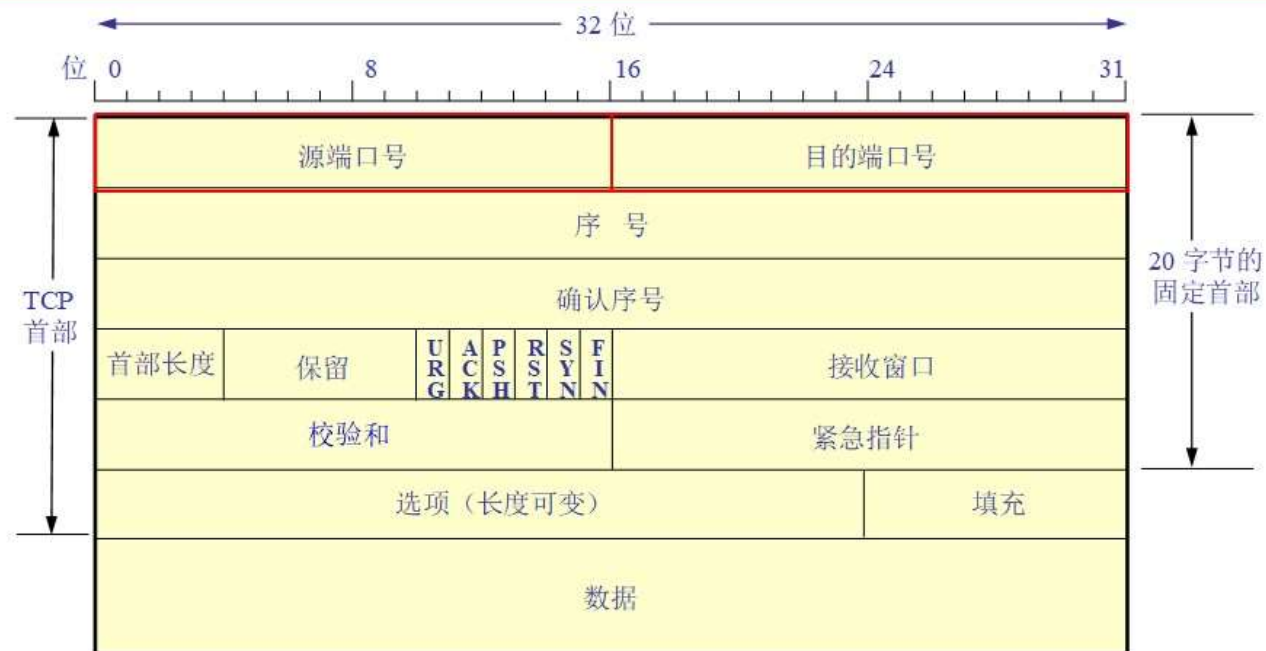
3.4 可靠数据传输原理

**3.5 TCP协议**

## TCP段结构



# TCP段结构



❖ 源端口号与目的端口号字段分别占16位

▪ 多路复用/分解





3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

3.4 可靠数据传输原理

**3.5 TCP协议****TCP段结构**

# TCP段结构



## ❖ 序号字段与确认序号字段分别占32位

- 对每个应用层数据的每个**字节**进行编号
- 确认序号是**期望**从对方接收数据的字节序号，累计确认



3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

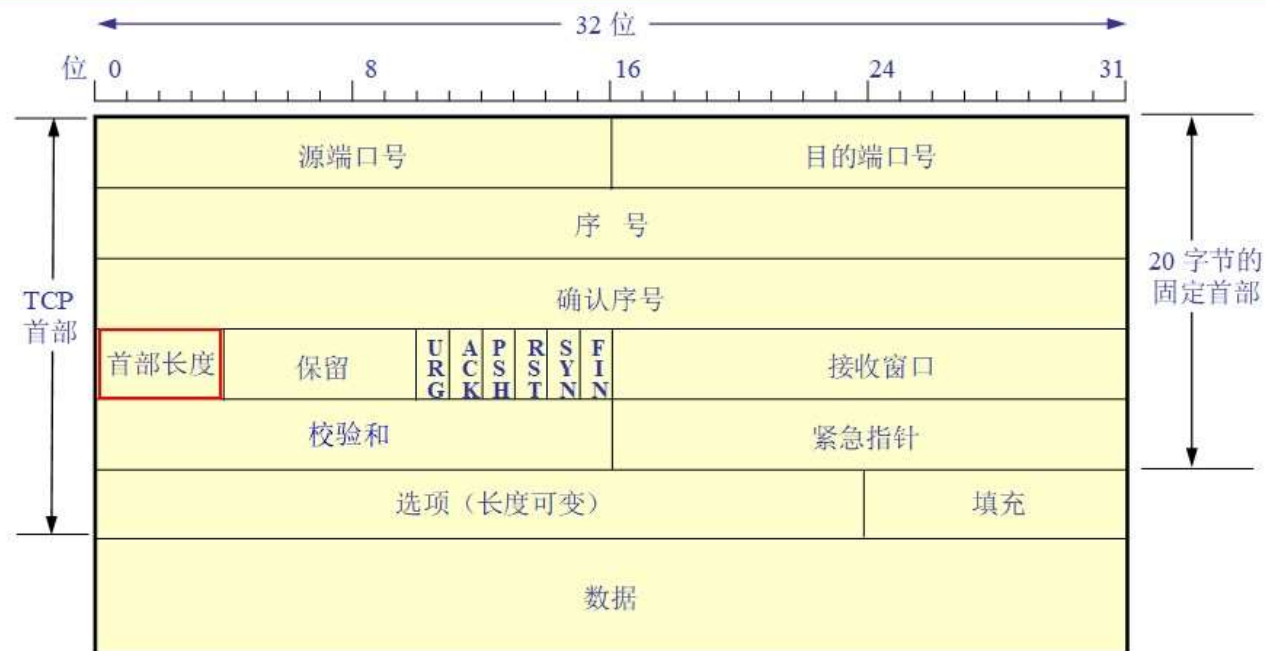
3.4 可靠数据传输原理

**3.5 TCP协议**

## TCP段结构



# TCP段结构



## ❖ 首部长度的字段占4位

- 4字节为计算单位



3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

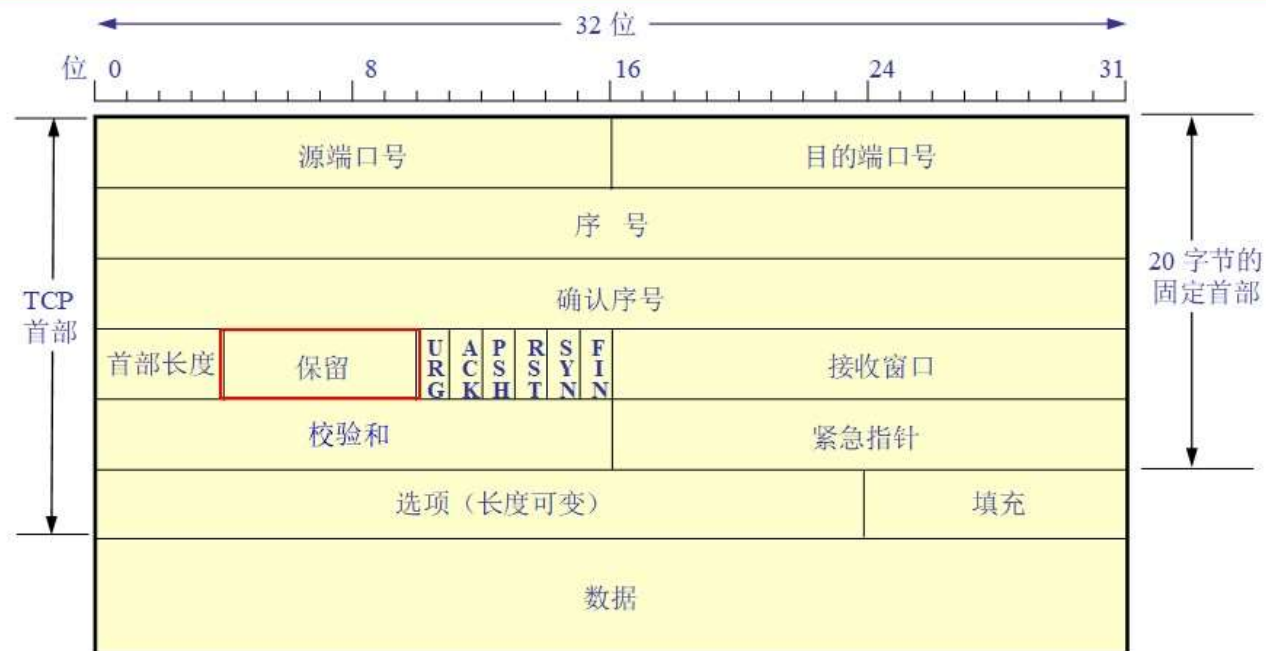
3.4 可靠数据传输原理

**3.5 TCP协议**

## TCP段结构



# TCP段结构



❖ **保留字段占6位**

■ **目前值为0**



3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

3.4 可靠数据传输原理

3.5 TCP协议

## TCP段结构



# TCP段结构



## ❖ 6位标志位 (字段)

- URG=1时, 表明紧急指针字段有效
- ACK=1时, 标识确认序号字段有效
- PSH=1时, 尽快将段中数据交付接收应用进程
- RST=1时, 重新建立TCP连接
- SYN=1时, 表示该TCP段是一个建立新连接请求控制段
- FIN=1时, 表明请求释放TCP连接





3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

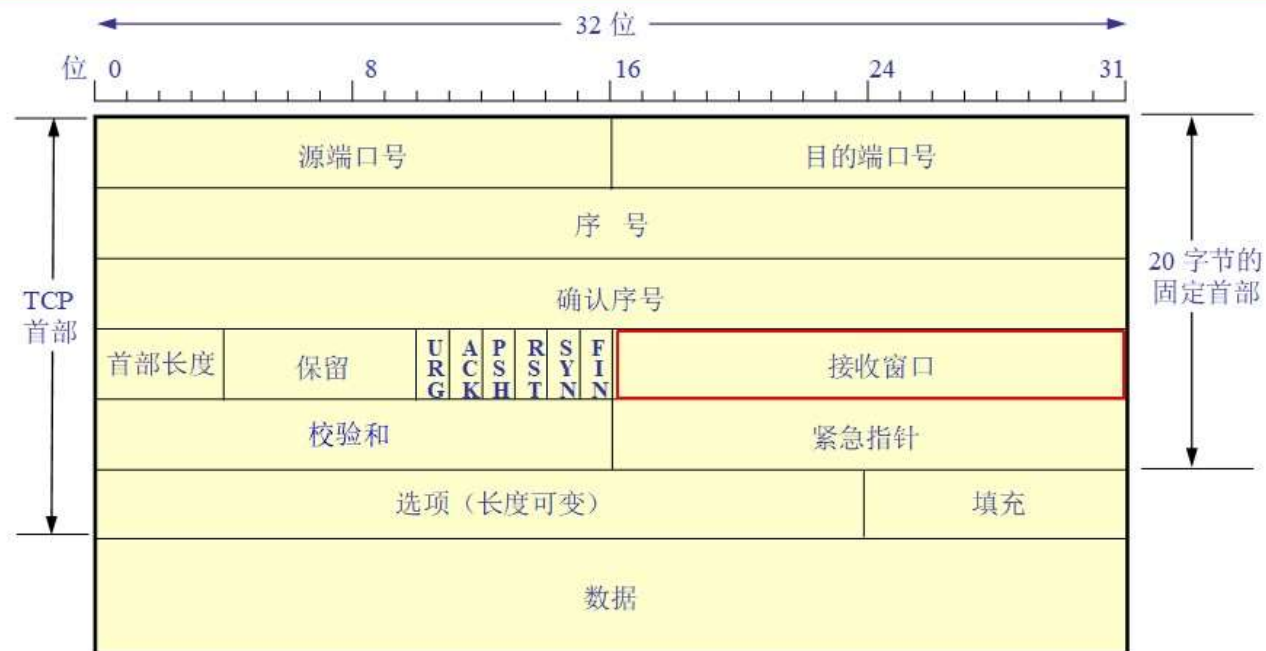
3.4 可靠数据传输原理

**3.5 TCP协议**

## TCP段结构



# TCP段结构



## ❖ 接收窗口字段占16位

### ▪ 流量控制



3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

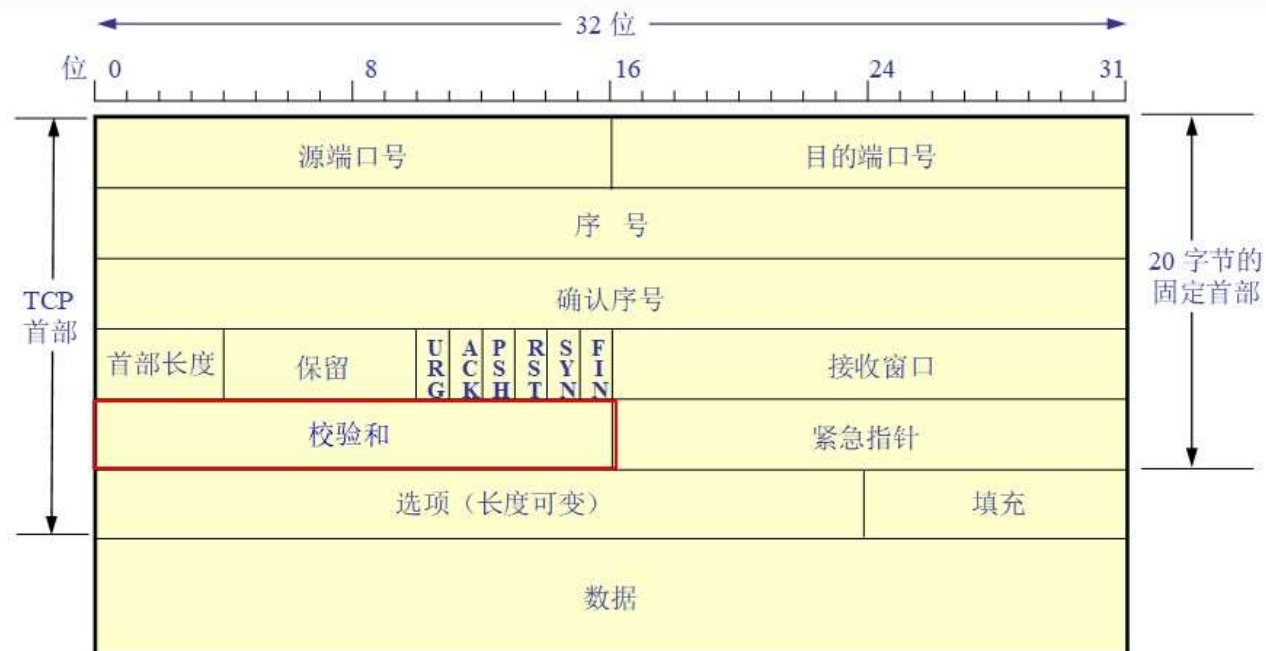
3.4 可靠数据传输原理

3.5 TCP协议

## TCP段结构



## TCP段结构



## ❖ 校验和字段占16位

- 包括TCP伪首部、TCP首部和应用层数据三部分



3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

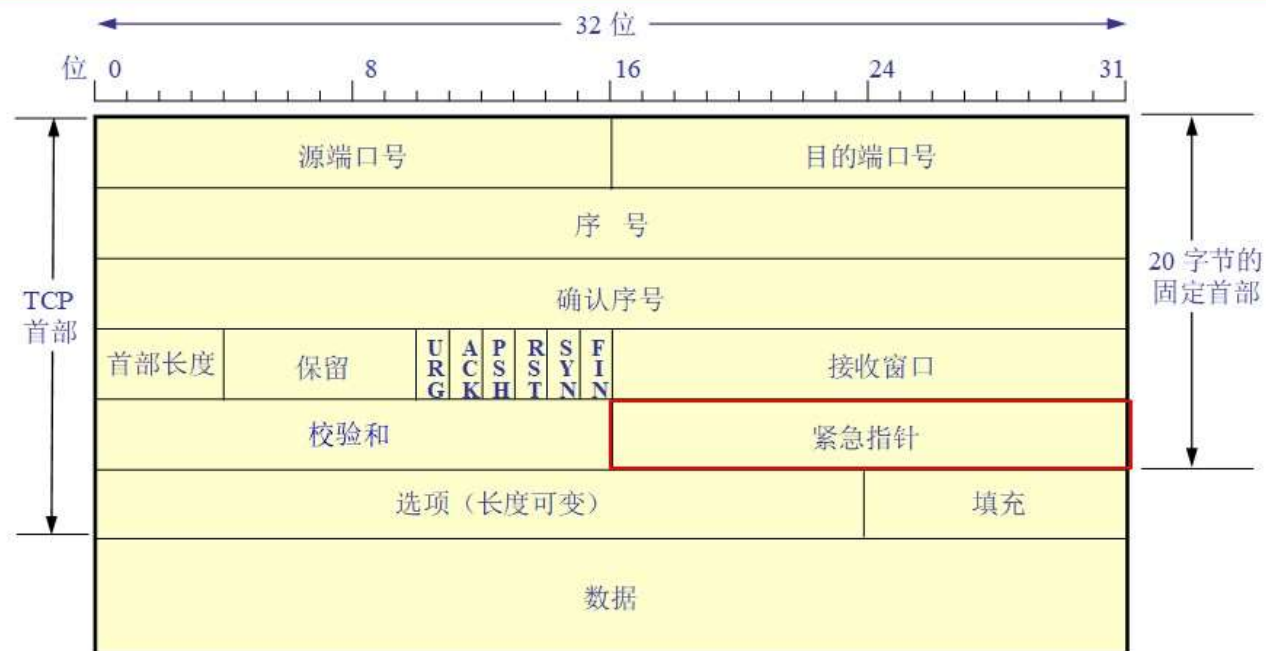
3.4 可靠数据传输原理

**3.5 TCP协议**

## TCP段结构



# TCP段结构



## ❖ 紧急指针字段占16位

- URG=1时才有效
- 指出紧急数据最后一个字节在数据中的位置



3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

3.4 可靠数据传输原理

**3.5 TCP协议****TCP段结构**

# TCP段结构



## ❖ 选项字段的长度可变

- 最大段长度MSS
- 时间戳
- SACK





3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

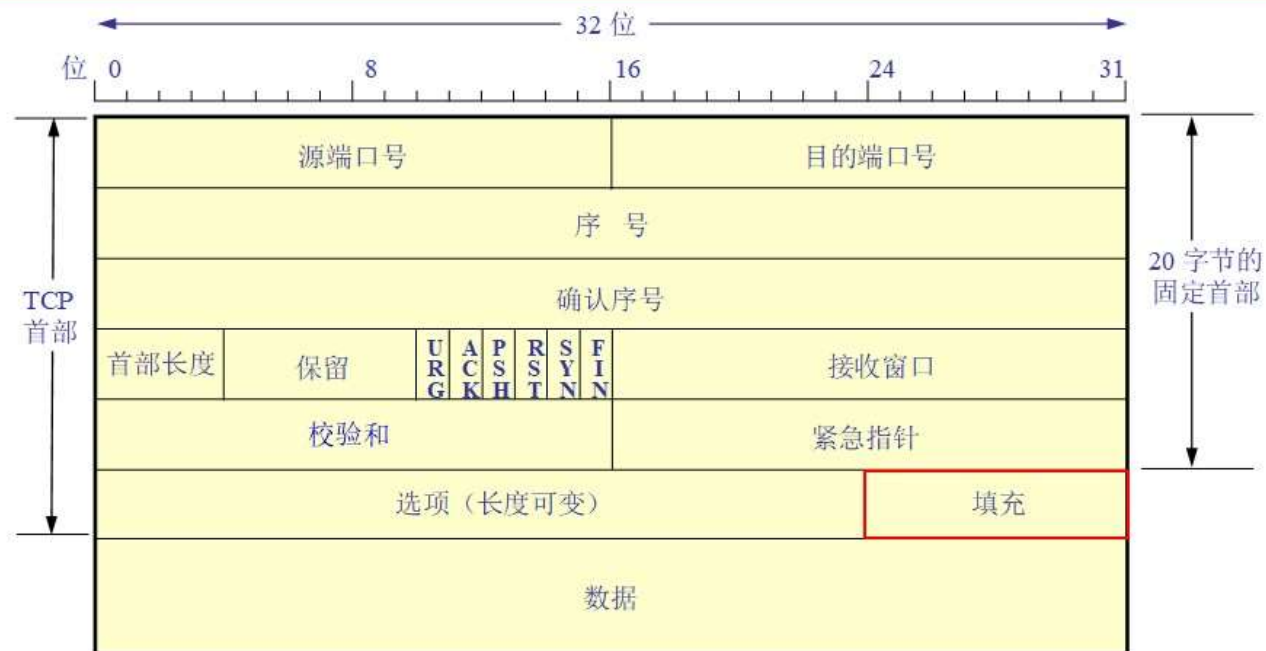
3.4 可靠数据传输原理

3.5 TCP协议

## TCP段结构



## TCP段结构



❖ 填充字段，长度为0~3个字节

■ 取值全0



3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

3.4 可靠数据传输原理

## 3.5 TCP协议

## TCP段结构



# TCP: 序列号和ACK

## 序列号:

- 序号是段（**Segment**）中**第1个字节**的编号，而不是段的“连续”编号
- 建立**TCP**连接时，双方随机选择序列号

## ACKs:

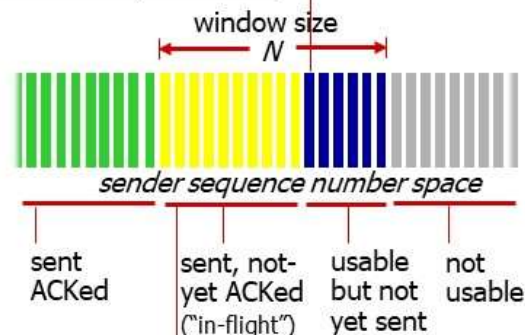
- **期望**接收到的下一个字节的序列号
- **累计确认**：该序列号之前的所有字节均已被正确接收到

**Q:** 接收方如何处理乱序到达的段？

- **A:** **TCP**规范中没有规定，由**TCP**的实现者做出决策

outgoing segment from sender

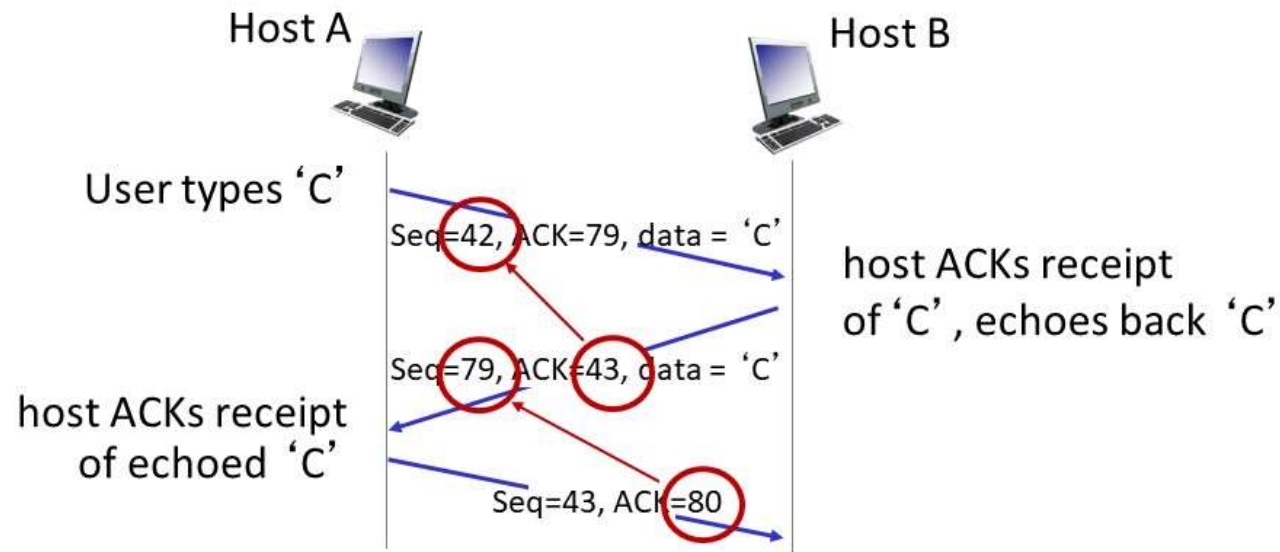
source port #	dest port #
sequence number	
acknowledgement number	
	rwnd
checksum	urg pointer



outgoing segment from receiver

source port #	dest port #
sequence number	
acknowledgement number	
	A
checksum	urg pointer

# TCP sequence numbers, ACKs



simple telnet scenario

Transport Layer: 3-123



3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

3.4 可靠数据传输原理

**3.5 TCP协议**

TCP可靠数据传输



# TCP可靠数据传输概述

- ❖ TCP在IP层的不可靠服务基础上实现可靠数据传输服务
- ❖ 流水线机制
- ❖ 累积确认
- ❖ TCP使用单一重传定时器
- ❖ 触发重传的事件
  - 超时
  - 收到重复ACK
- ❖ 渐进式





3.1 传输层服务

3.2 传输层多路复用/分解

3.3 UDP协议

3.4 可靠数据传输原理

3.5 TCP协议

TCP可靠数据传输



# TCP RTT和超时

❖ **问题：如何设置定时器的超时时间？**

❖ **大于RTT**

- 但是RTT是变化的

❖ **过短：**

- 不必要的重传

❖ **过长：**

- 对段丢失时间反应慢

❖ **问题：如何估计RTT？**

❖ **SampleRTT：测量从段发出去到收到ACK的时间**

- 忽略重传

❖ **SampleRTT变化**

- 测量多个SampleRTT，求平均值，形成RTT的估计值  
EstimatedRTT

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

指数加权移动平均

$\alpha$ 典型值：0.125