



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场



# 计算机网络之探赜索隐

主讲人：聂兰顺

# 本讲主题

## TCP概述



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

## ❖ 点对点

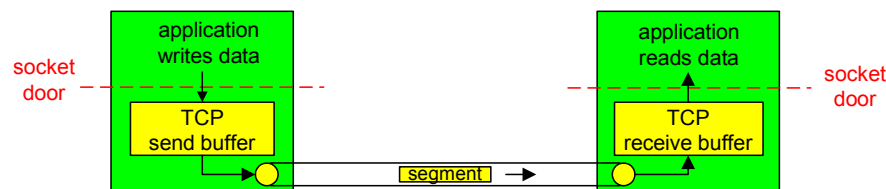
- 一个发送方，一个接收方

## ❖ 可靠的、按序的字节流

## ❖ 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

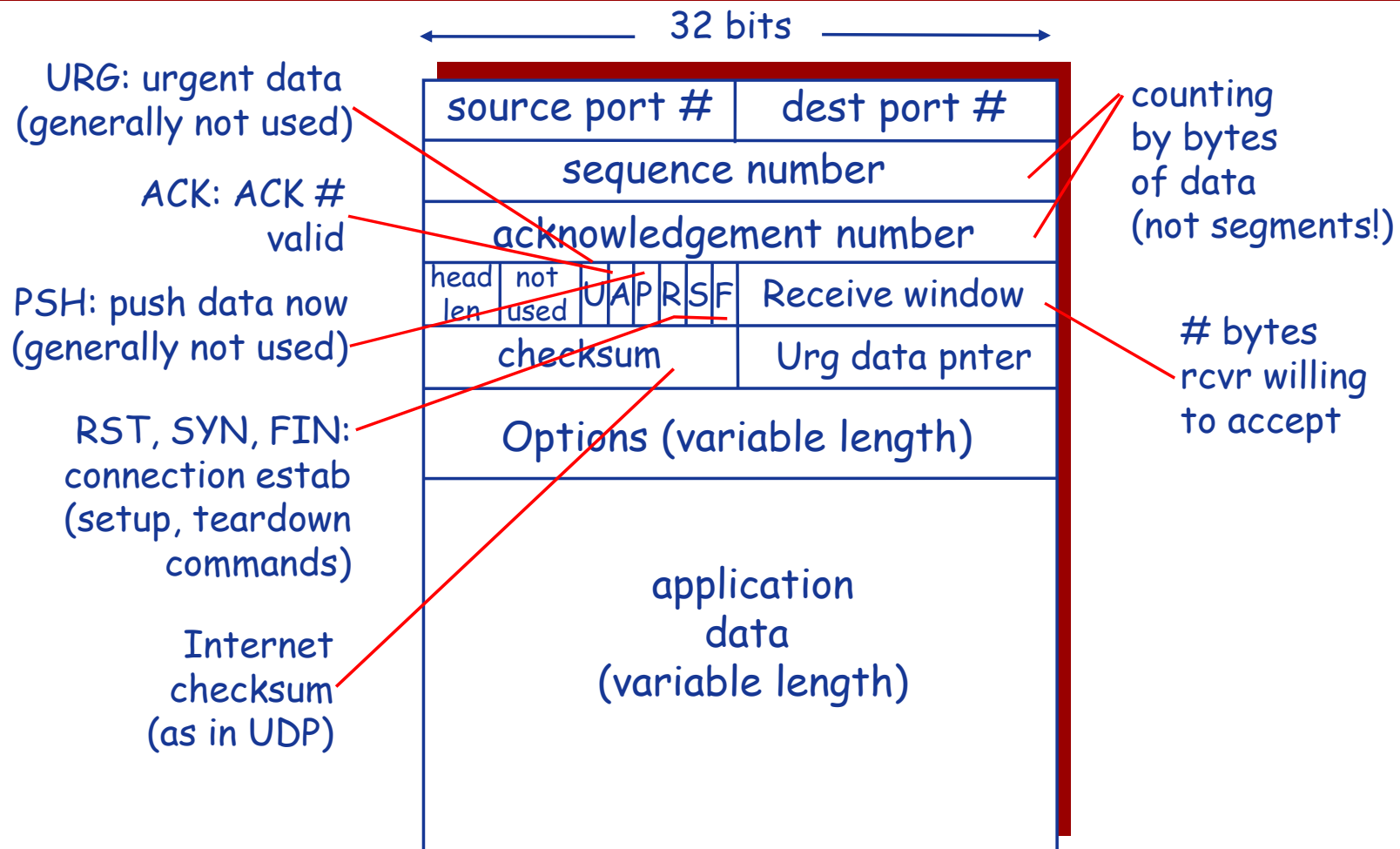
## ❖ 面向连接

- 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



# TCP段结构



# TCP: 序列号和ACK

## 序列号:

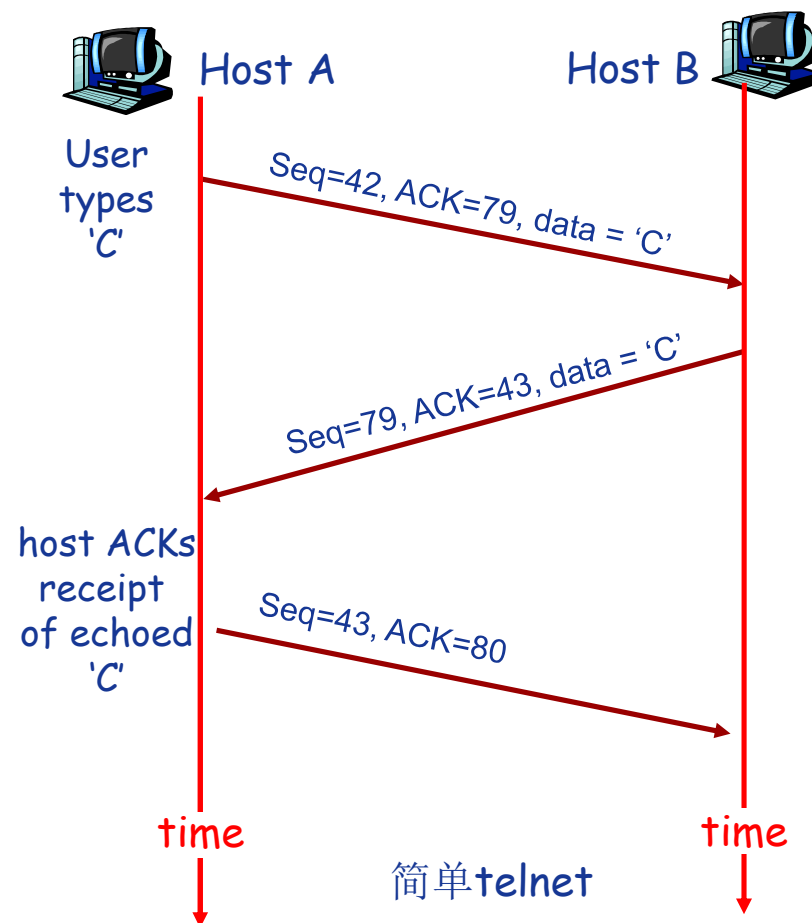
- 序列号指的是**segment**中第一个字节的编号，而不是**segment**的编号
- 建立**TCP**连接时，双方随机选择序列号

## ACKs:

- 希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**Segment**?

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





哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场



谢谢!





哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场



# 计算机网络之探赜索隐

主讲人：聂兰顺

# 本讲主题

## TCP可靠数据传输





# TCP可靠数据传输概述

- ❖ TCP在IP层提供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
  - 暂不考虑重复ACK
  - 暂不考虑流量控制
  - 暂不考虑拥塞控制



# 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

典型值：0.125



# TCP RTT和超时

## 定时器超时时间的设置:

- **EstimatedRTT + “安全边界”**
- **EstimatedRTT变化大→较大的边界**

## 测量RTT的变化值: SampleRTT与EstimatedRTT的差值

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

(typically,  $\beta = 0.25$ )

## 定时器超时时间的设置:

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



# TCP发送方事件

## ❖ 从应用层收到数据

- 创建Segment
- 序列号是Segment第一个字节的编号
- 开启计时器
- 设置超时时间:  
TimeoutInterval

## ❖ 超时

- 重传引起超时的Segment
- 重启定时器

## ❖ 收到ACK

- 如果确认此前未确认的Segment
  - 更新SendBase
  - 如果窗口中还有未被确认的分组, 重新启动定时器



# TCP发送端程序

```
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  
        NextSeqNum = NextSeqNum + length(data)
```

```
    event: timer timeout  
        retransmit not-yet-acknowledged segment with  
            smallest sequence number  
        start timer
```

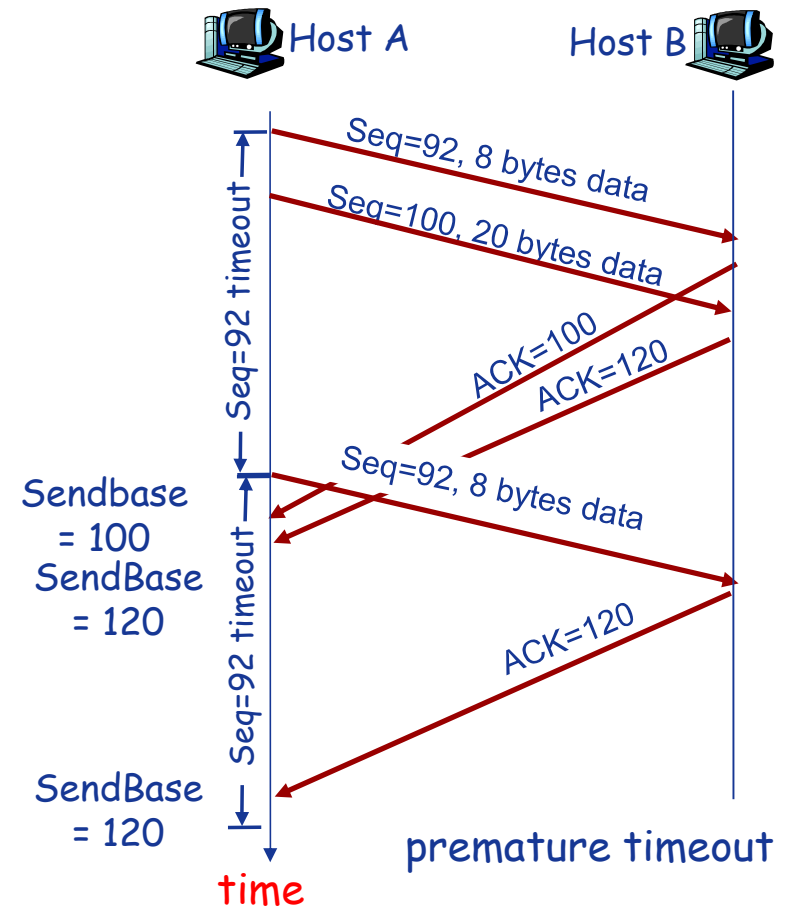
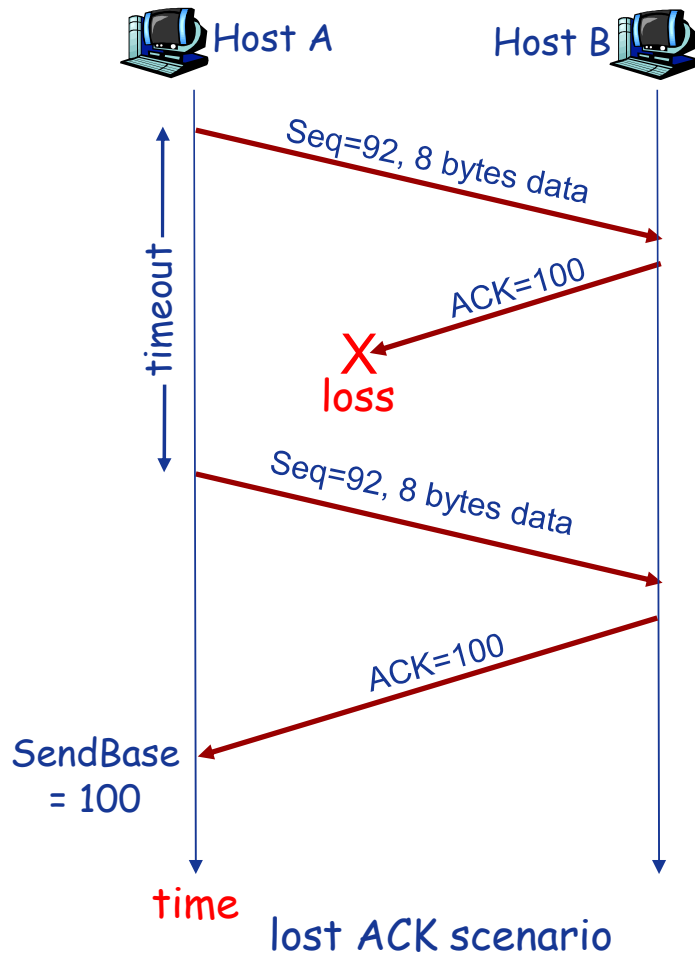
```
    event: ACK received, with ACK field value of y  
        if (y > SendBase) {  
            SendBase = y  
            if (there are currently not-yet-acknowledged segments)  
                start timer  
        }
```

```
} /* end of loop forever */
```

人：聂兰顺

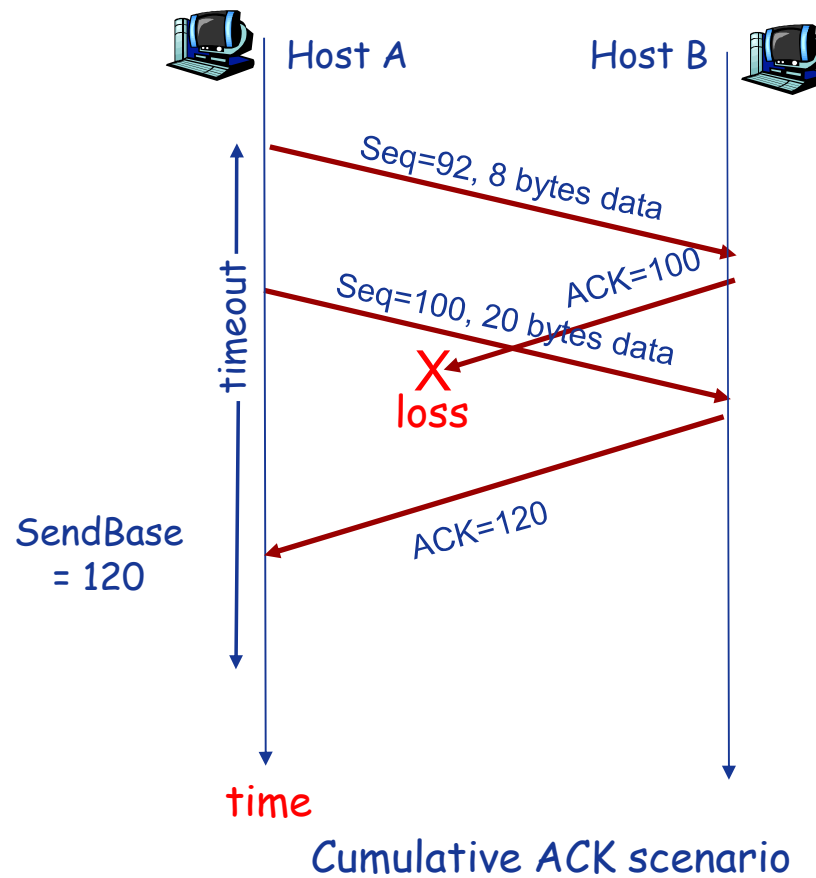


# TCP重传示例





# TCP重传示例



# TCP ACK生成: RFC 1122, RFC 2581

Event at Receiver	TCP Receiver action
Arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	Delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
Arrival of in-order segment with expected seq #. One other segment has ACK pending	Immediately send single cumulative ACK, ACKing both in-order segments
Arrival of out-of-order segment higher-than-expect seq. # . Gap detected	Immediately send <i>duplicate ACK</i> , indicating seq. # of next expected byte
Arrival of segment that partially or completely fills gap	Immediate send ACK, provided that segment starts at lower end of gap



# 快速重传机制

❖ TCP的实现中，如果发生超时，超时时间间隔将重新设置，即将超时时间间隔加倍，导致其**很大**

- 重发丢失的分组之前要等待很长时间

❖ 通过重复ACK检测分组丢失

- Sender会背靠背地发送多个分组
- 如果某个分组丢失，可能会引发多个重复的ACK

❖ 如果sender收到对同一数据的3个ACK，则假定该数据之后的段已经丢失

- **快速重传**：在定时器超时之前即进行重传



# 快速重传算法

```
event: ACK received, with ACK field value of y
  if (y > SendBase) {
    SendBase = y
    if (there are currently not-yet-acknowledged segments)
      start timer
  }
  else {
    increment count of dup ACKs received for y
    if (count of dup ACKs received for y = 3) {
      resend segment with sequence number y
    }
  }
```

a duplicate ACK for  
already ACKed segment

fast retransmit





哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场



谢谢!



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场



# 计算机网络之探赜索隐

主讲人：聂兰顺



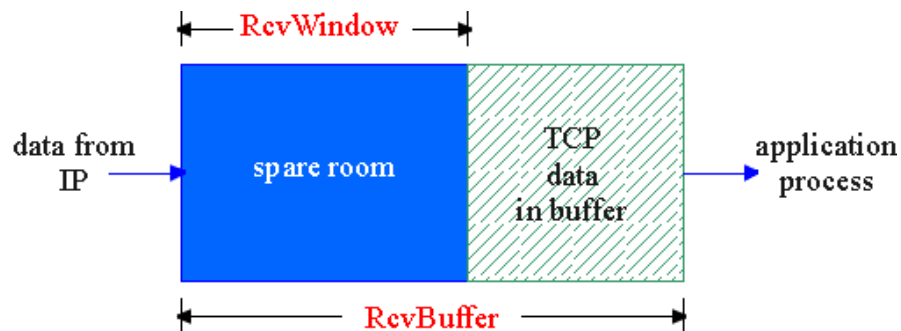
# 本讲主题

## TCP流量控制



# TCP流量控制

## ❖ 接收方为TCP连接分配buffer



## ❑ 上层应用可能处理buffer中数据的速度较慢

## flow control

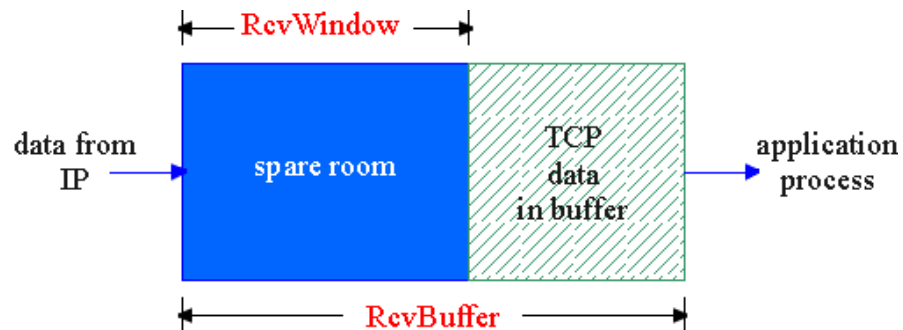
发送方不会传输的太多、太快以至于淹没接收方

(buffer溢出)

## ❖ 速度匹配机制



# TCP流量控制



(假定TCP receiver丢弃乱序的 segments)

❖ Buffer中的可用空间(spare room)

= RcvWindow

=  $\text{RcvBuffer} - [\text{LastByteRcvd} - \text{LastByteRead}]$

❖ Receiver通过在Segment的头部字段将 **RcvWindow** 告诉Sender

❖ Sender限制自己已经发送的但还未收到ACK的数据不超过接收方的空闲 **RcvWindow** 尺寸

❖ Receiver告知Sender **RcvWindow=0**, 会出现什么情况?





哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场



谢谢!



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场



# 计算机网络之探赜索隐

主讲人：聂兰顺

# 本讲主题

## TCP连接管理





# TCP连接管理

❖ TCP sender和receiver在传输数据前需要建立连接

❖ 初始化TCP变量

- Seq. #
- Buffer和流量控制信息

❖ Client: 连接发起者

```
Socket clientSocket = new  
Socket("hostname", "port number");
```

❖ Server: 等待客户连接请求

```
Socket connectionSocket =  
welcomeSocket.accept();
```

## Three way handshake:

Step 1: client host sends TCP SYN segment to server

- specifies initial seq #
- no data

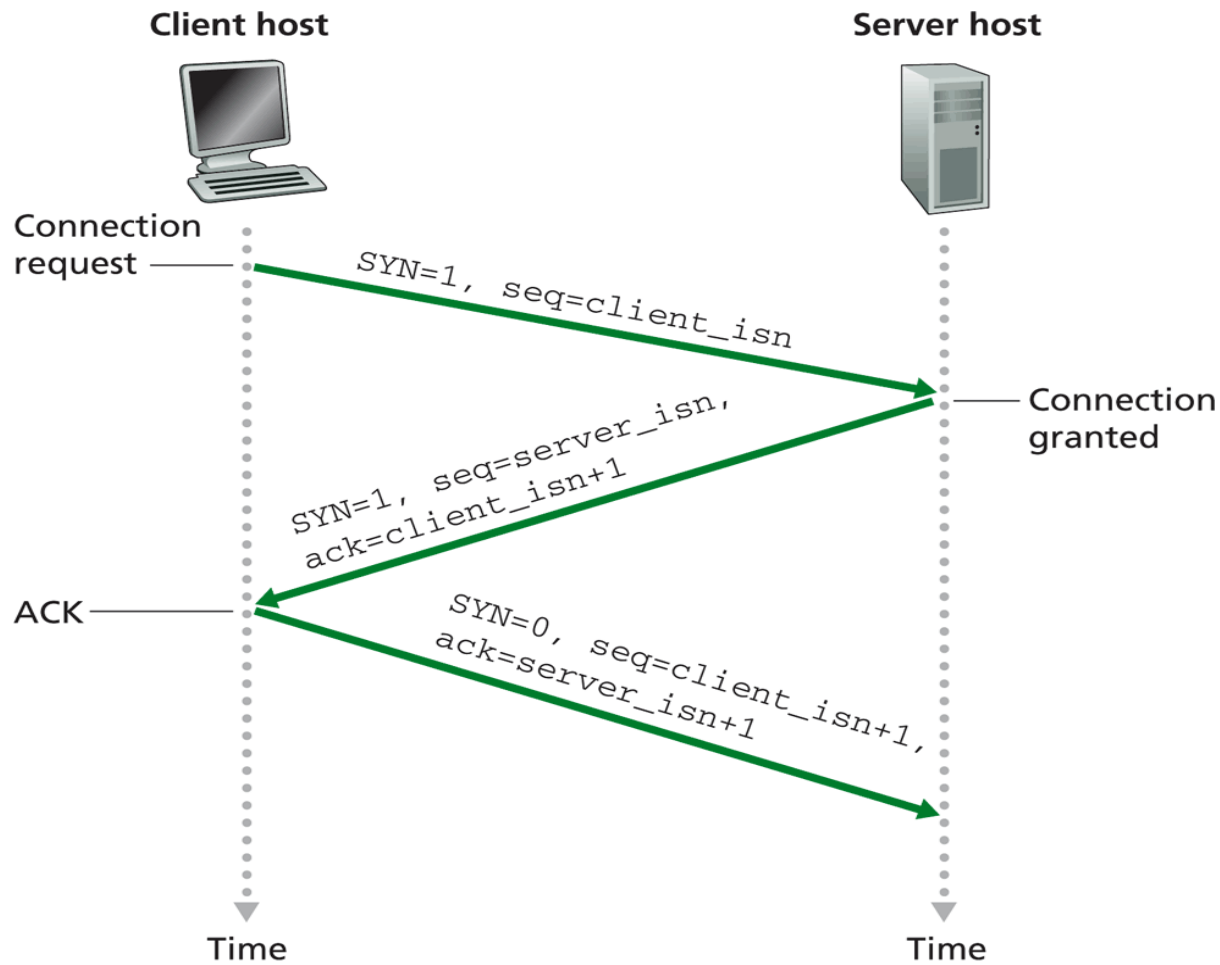
Step 2: server host receives SYN, replies with SYNACK segment

- server allocates buffers
- specifies server initial seq. #

Step 3: client receives SYNACK, replies with ACK segment, which may contain data



# TCP连接管理：建立



# TCP连接管理：关闭

## Closing a connection:

client closes socket: `clientSocket.close()`;

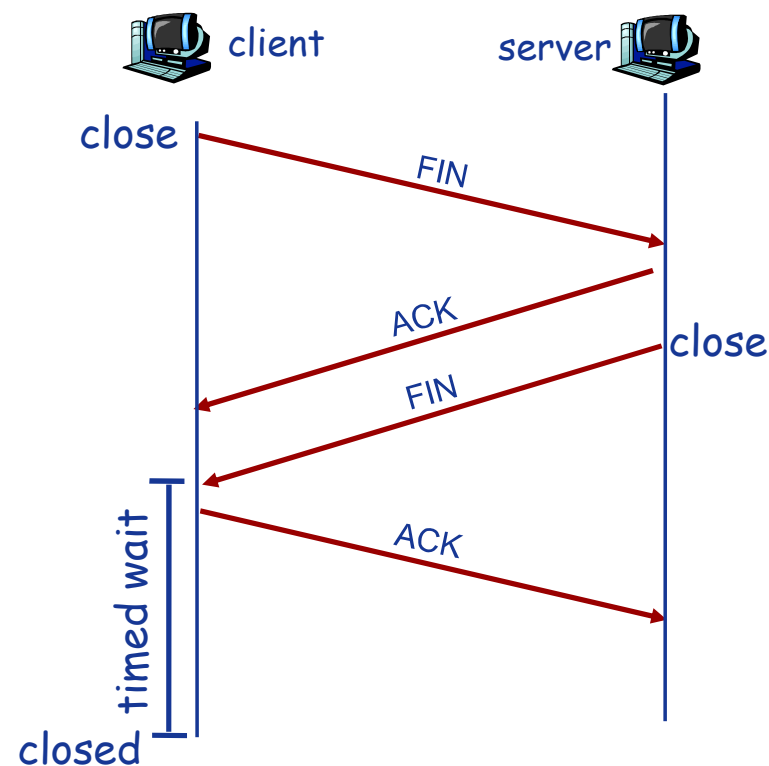
Step 1: client向server发送TCP FIN 控制segment

Step 2: server 收到FIN, 回复ACK. 关闭连接, 发送FIN.

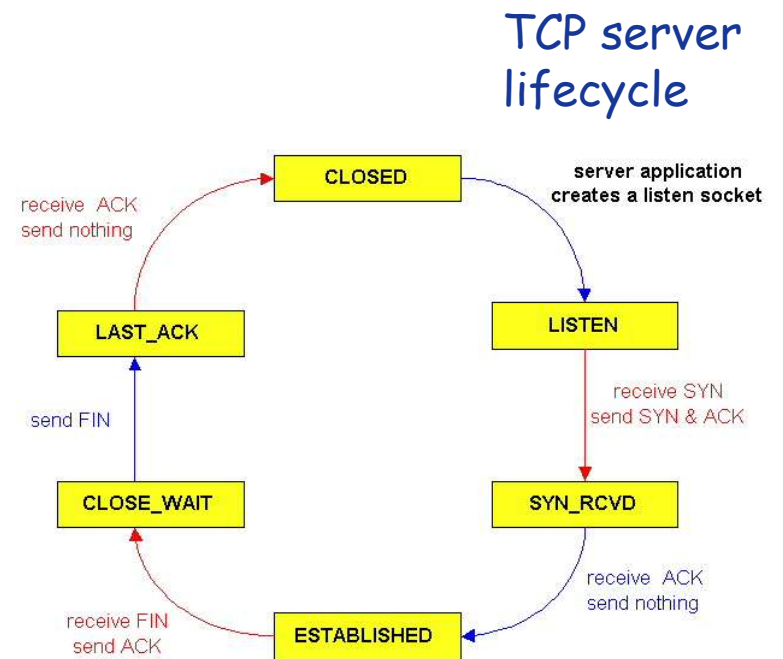
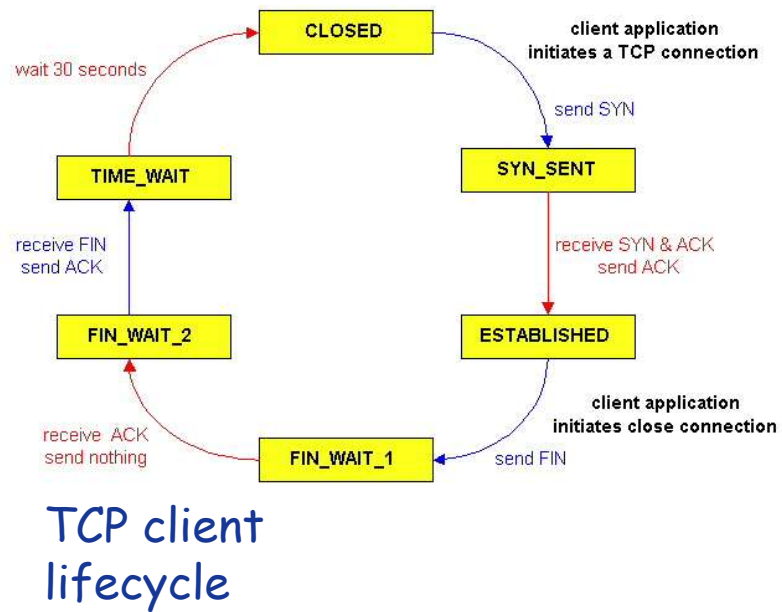
Step 3: client 收到FIN, 回复ACK.

- 进入“等待”-如果收到FIN, 会重新发送ACK

Step 4: server收到ACK. 连接关闭.



# TCP连接管理





哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场



谢谢!



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场



# 计算机网络之探赜索隐

主讲人：聂兰顺



# 本讲主题

## 拥塞控制原理(1)



# 拥塞控制

## 拥塞(Congestion)

❖ 非正式定义：“太多发送主机发送了太多数据或者发送速度太快，以至于网络无法处理”

❖ 表现：

- 分组丢失（路由器缓存溢出）
- 分组延迟过大（在路由器缓存中排队）

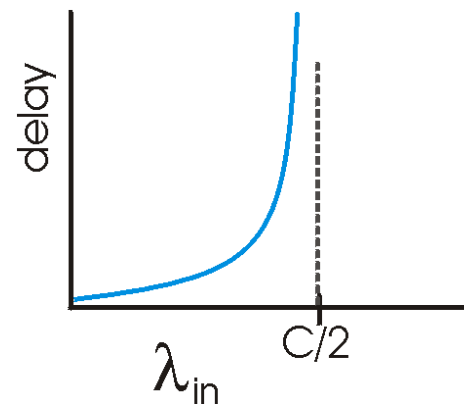
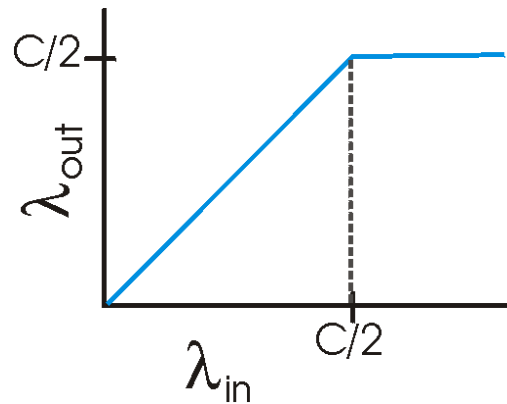
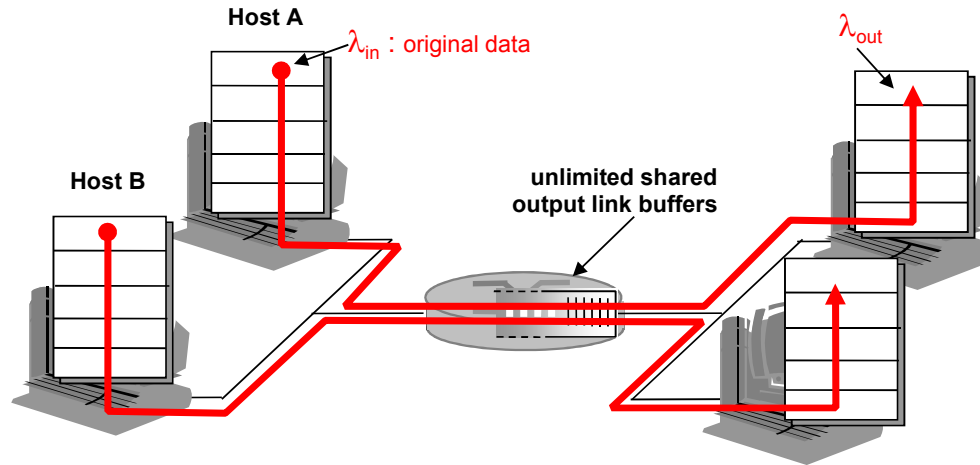
❖ 拥塞控制 vs. 流量控制

❖ A top-10 problem.



# 拥塞的成因和代价：场景1

- ❖ 两个senders,两个receivers
- ❖ 一个路由器, 无限缓存
- ❖ 没有重传

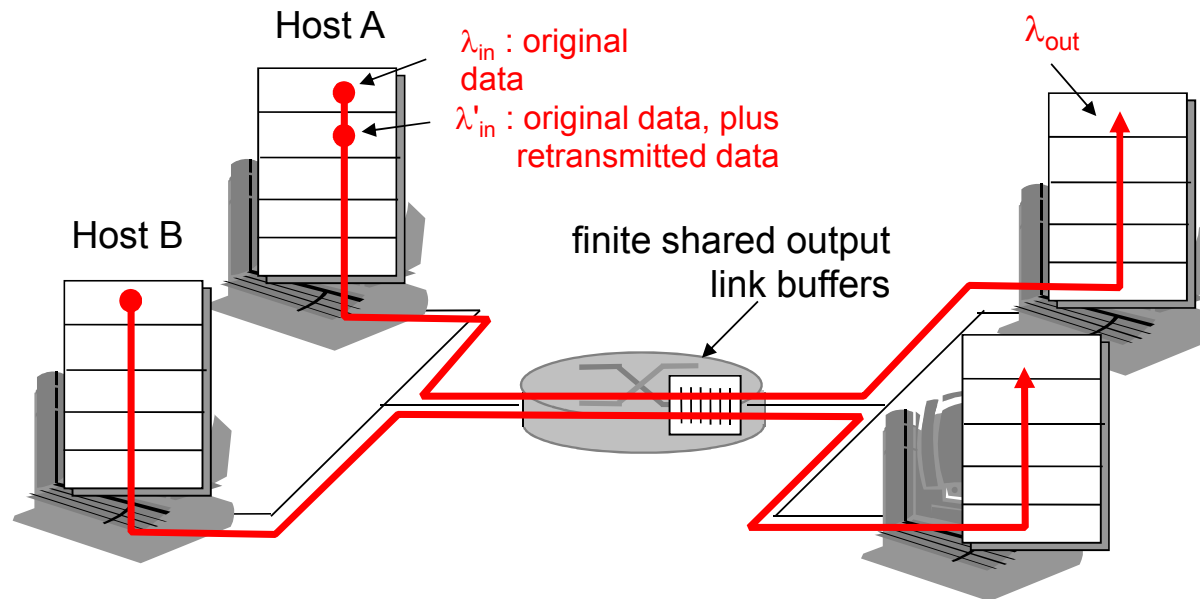


- ❖ 拥塞时分组延迟太大
- ❖ 达到最大 throughput



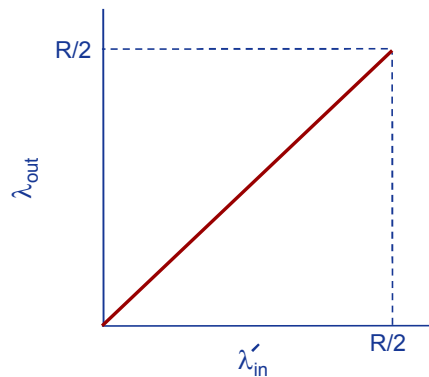
## 拥塞的成因和代价：场景2

- ❖ 一个路由器, 有限buffers
- ❖ Sender重传分组

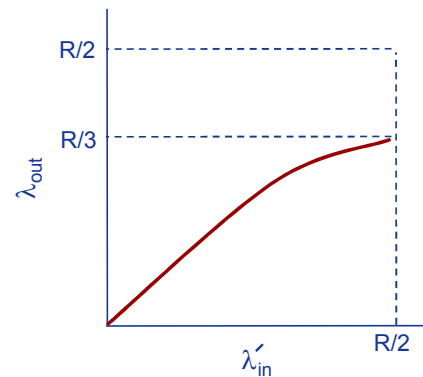


## 拥塞的成因和代价：场景2

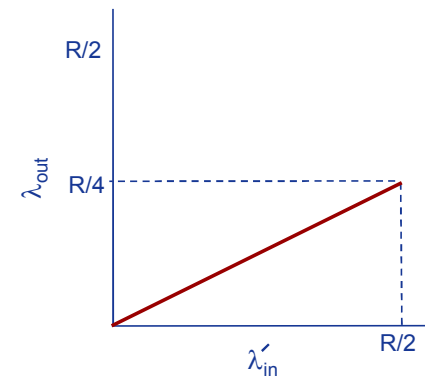
- ❖ 情况a: Sender能够通过某种机制获知路由器buffer信息，有空闲才发  $\lambda_{in} = \lambda_{out}$  (goodput)
- ❖ 情况b: 丢失后才重发:  $\lambda'_{in} > \lambda_{out}$
- ❖ 情况c: 分组丢失和定时器超时而都重发,  $\lambda'_{in}$  变得更大



a.



b.



c.

拥塞的代价:

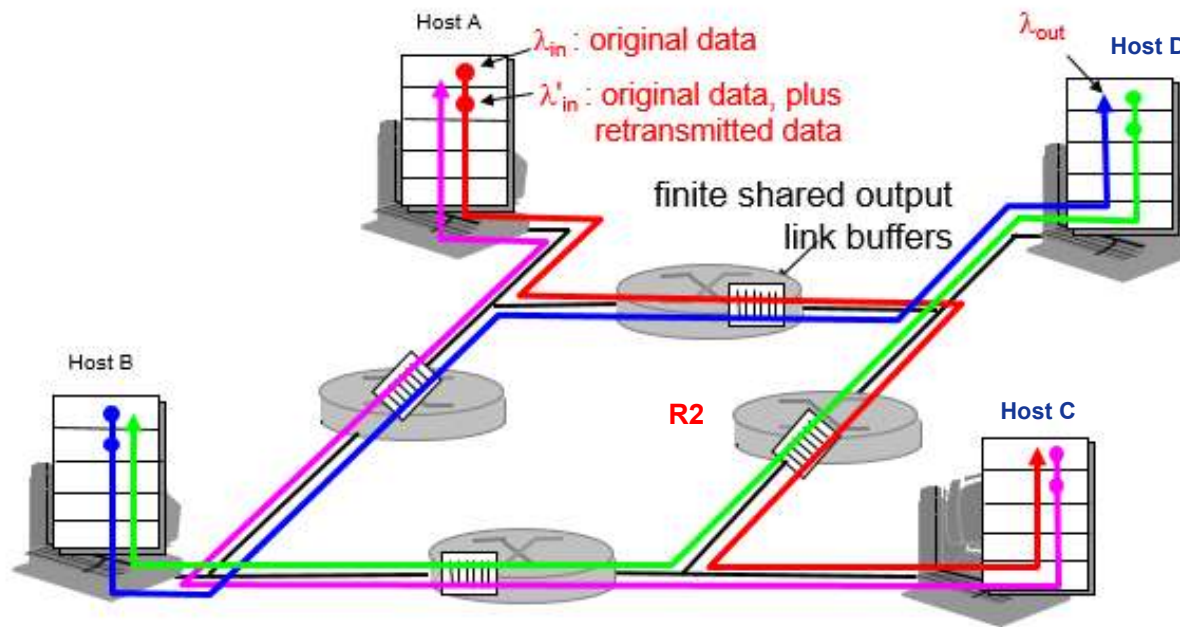
- ❑ 对给定的“goodput”，要做更多的工作（重传）
- ❑ 造成资源的浪费



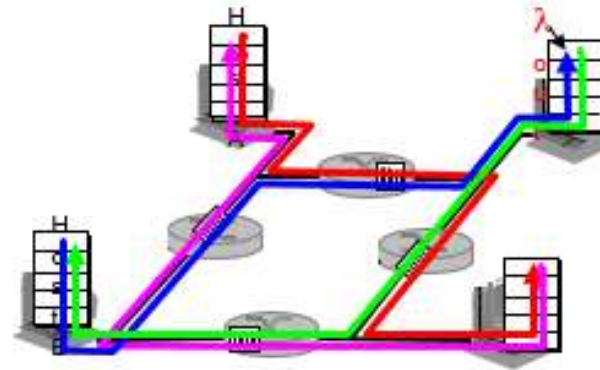
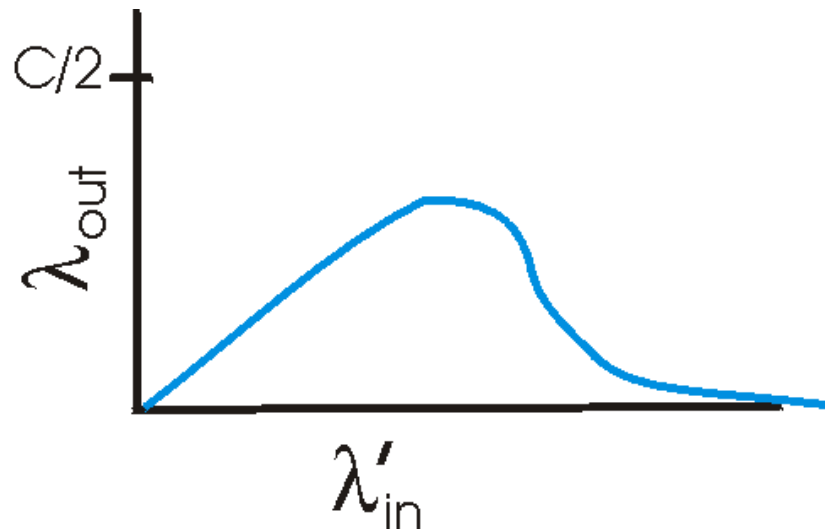
## 拥挤的成因和代价：场景3

- ❖ 四个发送方
- ❖ 多跳
- ❖ 超时/重传

Q: 随着 $\lambda_{in}$ 和 $\lambda'_{in}$ 不断增加, 会怎么样?



## 拥塞的成因和代价：场景3



拥塞的另一个代价:

- ❑ 当分组被**drop**时，任何用于该分组的“上游”传输能力全都被浪费掉







哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场



谢谢!



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场



# 计算机网络之探赜索隐

主讲人：聂兰顺

# 本讲主题

## 拥塞控制原理(2)



# 拥塞控制的方法

## ❖ 端到端拥塞控制:

- 网络层不需要显式的提供支持
- 端系统通过观察loss, delay等网络行为判断是否发生拥塞
- TCP采取这种方法

## ❖ 网络辅助的拥塞控制:

- 路由器向发送方显式地反馈网络拥塞信息
- 简单的拥塞指示(1bit): SNA, DECbit, TCP/IP ECN, ATM)
- 指示发送方应该采取何种速率



# 案例：ATM ABR拥塞控制

## ❖ ABR: available bit rate

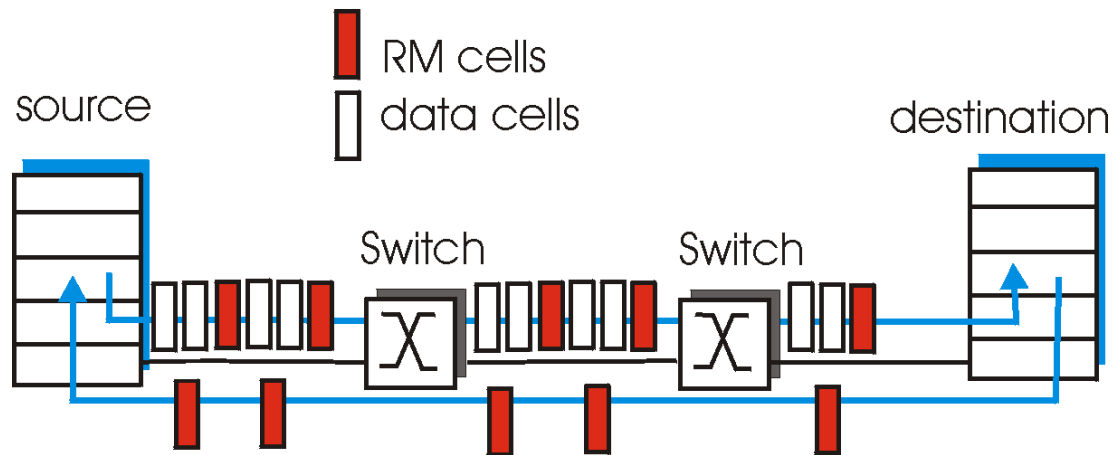
- “弹性服务”
- 如果发送方路径“underloaded”
  - ⑩使用可用带宽
- 如果发送方路径拥塞
  - ⑩将发送速率降到最低保障速率

## ❖ RM(resource management) cells

- 发送方发送
- 交换机设置RM cell位(网络辅助)
  - NI bit: rate不许增长
  - CI bit: 拥塞指示
- RM cell由接收方返回给发送方



# 案例： ATM ABR拥塞控制



- ❖ 在RM cell中有显式的速率(ER)字段：两个字节
  - 拥塞的交换机可以将ER置为更低的值
  - 发送方获知路径所能支持的最小速率
- ❖ 数据cell中的EFCI位：拥塞的交换机将其设为1
  - 如果RM cell前面的data cell的EFCI位被设为1，那么发送方在返回的RM cell中置CI位







哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场



谢谢!





哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场



# 计算机网络之探赜索隐

主讲人：聂兰顺

# 本讲主题

## TCP拥塞控制



# TCP拥塞控制的基本原理

## ❖ Sender限制发送速率

$\text{LastByteSent} - \text{LastByteAcked} \leq \text{CongWin}$

$$\text{rate} \approx \frac{\text{CongWin}}{\text{RTT}} \text{ Bytes/sec}$$

## ❖ CongWin:

- 动态调整以改变发送速率
- 反映所感知到的网络拥塞

问题：如何感知网络拥塞？

## ❖ Loss事件=timeout或3个重复ACK

## ❖ 发生loss事件后，发送方降低速率

如何合理地调整发送速率？

## ❖ 加性增—乘性减: AIMD

## ❖ 慢启动: SS



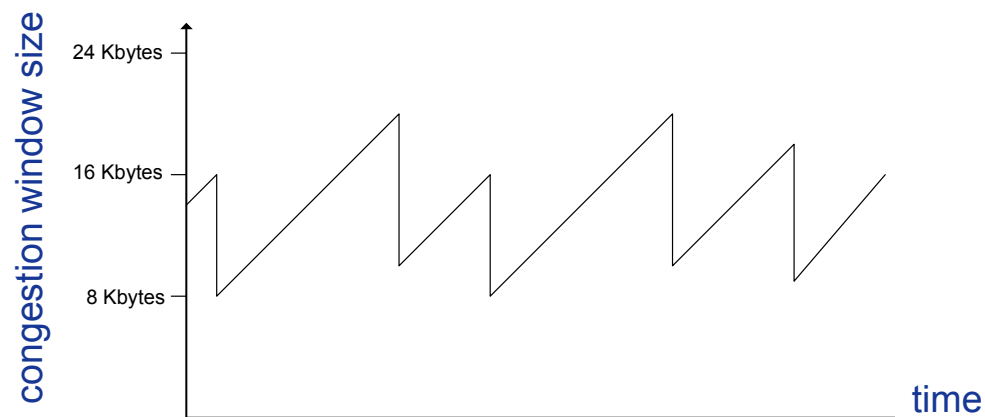
# 加性增—乘性减: AIMD

❖ **原理:** 逐渐增加发送速率, 谨慎探测可用带宽, 直到发生loss

❖ **方法:** AIMD

- Additive Increase: 每个RTT将CongWin增大一个MSS——拥塞避免
- Multiplicative Decrease: 发生loss后将CongWin减半

锯齿行为: 探测  
可用带宽



# TCP慢启动: SS

❖ TCP连接建立时,  
CongWin=1

- 例: MSS=500 byte,  
RTT=200msec
- 初始速率=20k bps

❖ 可用带宽可能远远高于初始  
速率:

- 希望快速增长

❖ 原理:

- 当连接开始时, 指数性增长

## Slowstart algorithm

```
initialize: Congwin = 1
for (each segment ACKed)
    Congwin++
until (loss event OR
      CongWin > threshold)
```

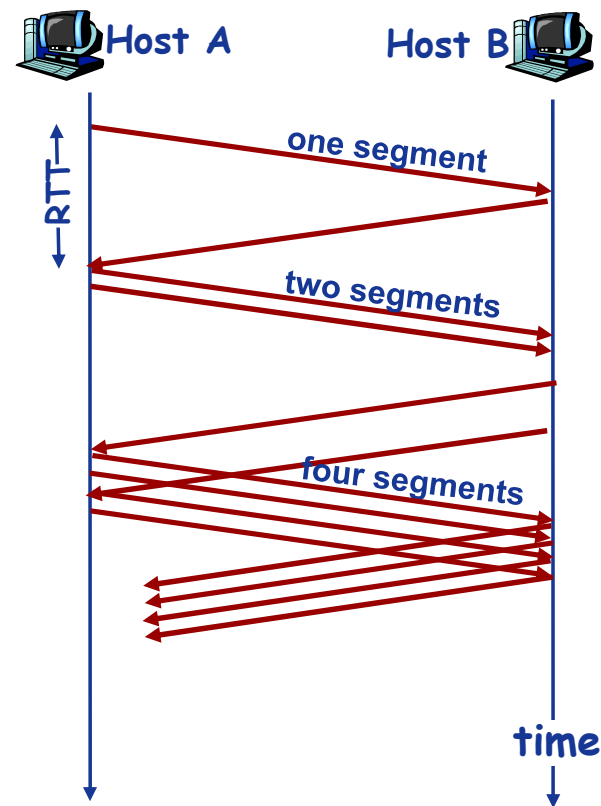


# TCP慢启动: SS

## ❖ 指数性增长

- 每个RTT将CongWin翻倍
- 收到每个ACK进行操作

## ❖ 初始速率很慢，但是快速攀升



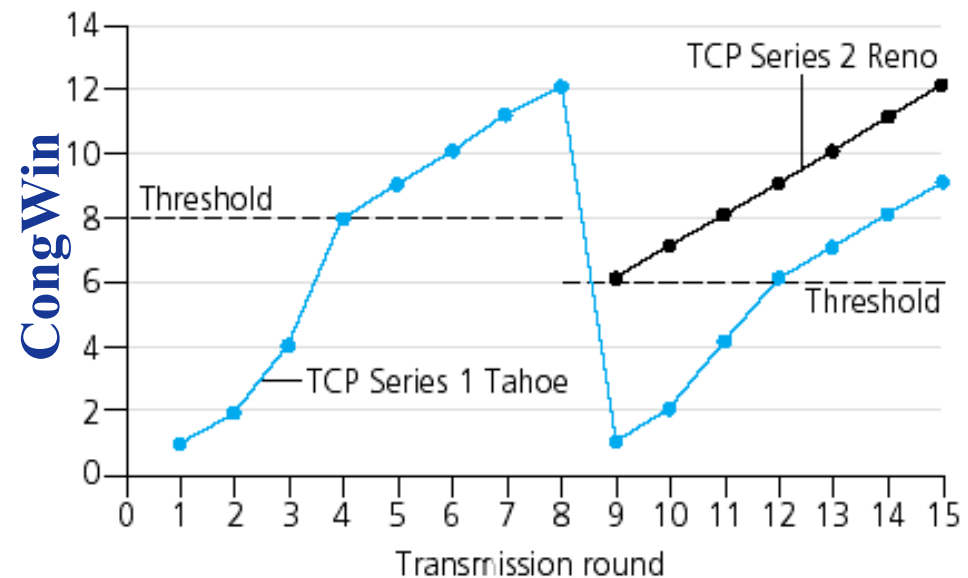
# Threshold变量

Q: 何时应该指数性增长切换为线性增长(拥塞避免)?

A: 当CongWin达到Loss事件前值的1/2时.

实现方法:

- ❖ 变量 **Threshold**
- ❖ Loss事件发生时, **Threshold** 被设为Loss事件前**CongWin**值的1/2。





# Loss事件的处理

## ❖ 3个重复ACKs:

- CongWin切到一半
- 然后线性增长

## ❖ Timeout事件:

- CongWin直接设为1个MSS
- 然后指数增长
- 达到threshold后, 再线性增长

## Philosophy:

- ❑ 3个重复ACKs表示网络还能够传输一些 segments
- ❑ timeout事件表明拥塞更为严重



# TCP拥塞控制: 总结

- ❖ When CongWin is below Threshold, sender in **slow-start** phase, window grows exponentially.
- ❖ When CongWin is above Threshold, sender is in **congestion-avoidance** phase, window grows linearly.
- ❖ When a **triple duplicate ACK** occurs, Threshold set to CongWin/2 and CongWin set to Threshold.
- ❖ When **timeout** occurs, Threshold set to CongWin/2 and CongWin is set to 1 MSS.



# TCP拥塞控制

<i>State</i>	<i>Event</i>	<i>TCP Sender Action</i>	<i>Commentary</i>
<i>Slow Start (SS)</i>	<i>ACK receipt for previously unacked data</i>	<i>CongWin = CongWin + MSS, If (CongWin &gt; Threshold) set state to "Congestion Avoidance"</i>	<i>Resulting in a doubling of CongWin every RTT</i>
<i>Congestion Avoidance (CA)</i>	<i>ACK receipt for previously unacked data</i>	<i>CongWin = CongWin + MSS * (MSS / CongWin)</i>	<i>Additive increase, resulting in increase of CongWin by 1 MSS every RTT</i>
<i>SS or CA</i>	<i>Loss event detected by triple duplicate ACK</i>	<i>Threshold = CongWin / 2, CongWin = Threshold, Set state to "Congestion Avoidance"</i>	<i>Fast recovery, implementing multiplicative decrease. CongWin will not drop below 1 MSS.</i>
<i>SS or CA</i>	<i>Timeout</i>	<i>Threshold = CongWin / 2, CongWin = 1 MSS, Set state to "Slow Start"</i>	<i>Enter slow start</i>
<i>SS or CA</i>	<i>Duplicate ACK</i>	<i>Increment duplicate ACK count for segment being acked</i>	<i>CongWin and Threshold not changed</i>

# TCP拥塞控制算法

Th = ?

CongWin = 1 MSS

/\* slow start or exponential increase \*/

While (No Packet Loss and CongWin < Th) {

    send CongWin TCP segments

    for each ACK increase CongWin by 1

}

/\* congestion avoidance or linear increase \*/

While (No Packet Loss) {

    send CongWin TCP segments

    for CongWin ACKs, increase CongWin by 1

}

Th = CongWin/2

If (3 Dup ACKs) CongWin = Th;

If (timeout) CongWin=1;



# 例题

- ❖ 一个TCP连接总是以1 KB的最大段长发送TCP段，发送方有足够多的数据要发送。当拥塞窗口为16 KB时发生了超时，如果接下来的4个RTT（往返时间）时间内的TCP段的传输都是成功的，那么当第4个RTT时间内发送的所有TCP段都得到肯定应答时，拥塞窗口大小是多少？
- ❖ 解：threshold=16/2=8 KB, CongWin=1 KB, 1个RTT后， CongWin=2 KB， 2个RTT后， CongWin=4 KB， 3个RTT后， CongWin=8 KB， Slowstart is over; 4个RTT后， CongWin=9 KB





哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场



谢谢!



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场



# 计算机网络之探赜索隐

主讲人：聂兰顺



# 本讲主题

## TCP性能分析



# TCP throughput: 吞吐率

- ❖ 给定拥塞窗口大小和RTT，TCP的平均吞吐率是多少？
  - 忽略掉Slow start
- ❖ 假定发生超时时CongWin的大小为W，吞吐率是 $W/RTT$
- ❖ 超时后， $CongWin=W/2$ ，吞吐率是 $W/2RTT$
- ❖ 平均吞吐率为： $0.75W/RTT$



# 未来的TCP

❖ 举例：每个Segment有1500个byte, RTT是100ms, 希望获得10Gbps的吞吐率

- $\text{throughput} = W * \text{MSS} * 8 / \text{RTT}$ , 则
- $W = \text{throughput} * \text{RTT} / (\text{MSS} * 8)$
- $\text{throughput} = 10\text{Gbps}$ , 则  $W = 83,333$

❖ 窗口大小为83,333



# 未来的TCP

## ❖ 吞吐率与丢包率(loss rate, L)的关系

- CongWin从W/2增加至W时出现第一个丢包，那么一共发送的分组数为

$$W/2 + (W/2 + 1) + (W/2 + 2) + \dots + W = 3W^2/8 + 3W/4$$

- W很大时， $3W^2/8 \gg 3W/4$ ，因此  $L \approx 8/(3W^2)$

$$W = \sqrt{\frac{8}{3L}} \quad \text{Throughput} = \frac{0.75 \cdot MSS \cdot \sqrt{\frac{8}{3L}}}{RTT} \approx \frac{1.22 \cdot MSS}{RTT \sqrt{L}}$$

❖  $L = 2 \cdot 10^{-10}$       **Wow!!!**

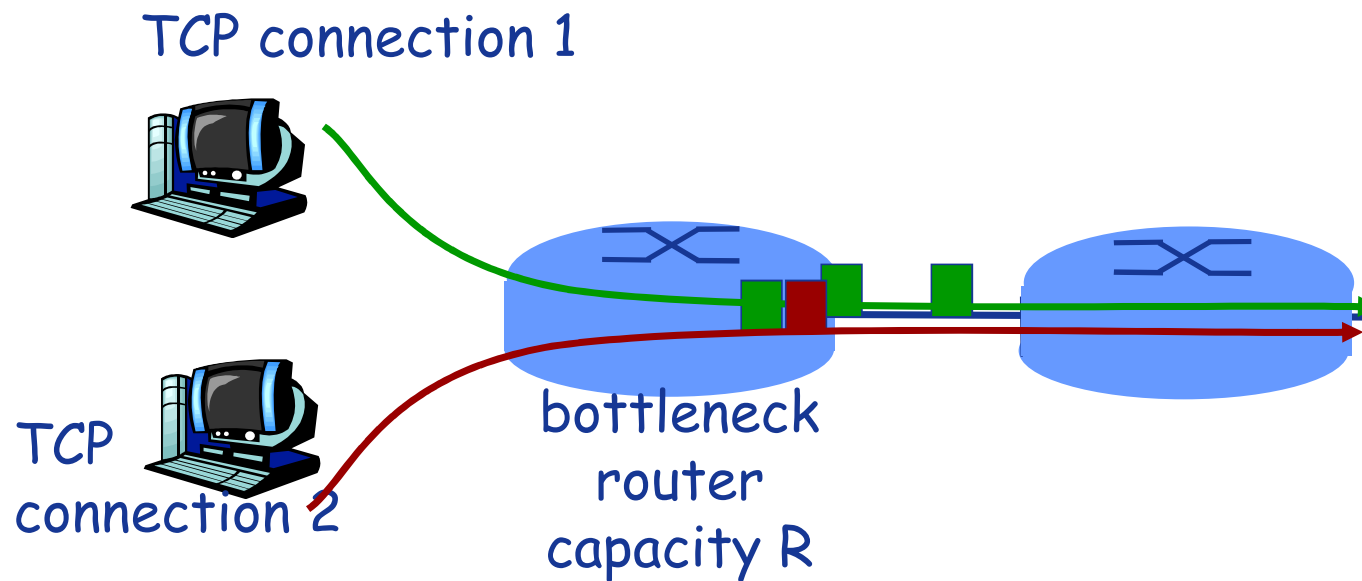
## ❖ 高速网络下需要设计新的TCP



# TCP的公平性

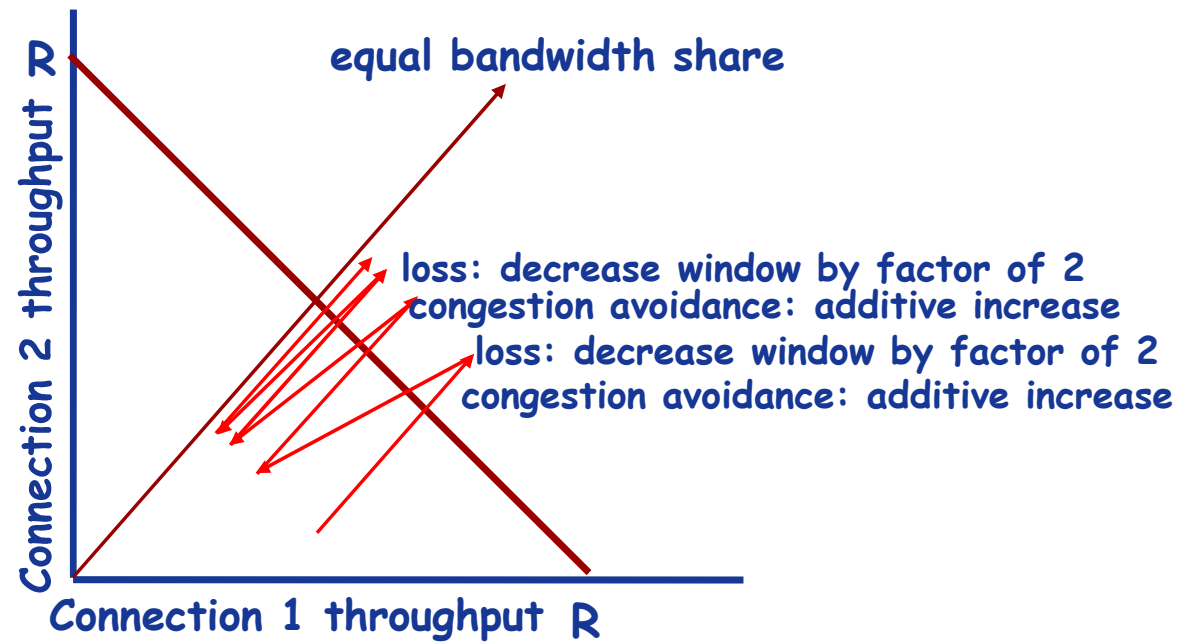
## ❖ 公平性？

- 如果K个TCP Session共享相同的瓶颈带宽R，那么每个Session的平均速率为 $R/K$



# TCP具有公平性吗？

❖ 是的



# TCP的公平性

## ❖ 公平性与UDP

- 多媒体应用通常不使用TCP，以免被拥塞控制机制限制速率
- 使用UDP：以恒定速率发送，能够容忍丢失
- 产生了不公平

## ❖ 研究：TCP friendly

## ❖ 公平性与并发TCP连接

- 某些应用会打开多个并发连接
- Web浏览器
- 产生公平性问题

## ❖ 例子：链路速率为 $R$ ，已有9个连接

- 新来的应用请求1个TCP，获得 $R/10$ 的速率
- 新来的应用请求11个TCP，获得 $R/2$ 的速率







哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场



谢谢!



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场



# 计算机网络之探赜索隐

主讲人：聂兰顺

# 本讲主题

## 传输层



# 本章知识点

## ❖ 传输层服务的基本原理

- 复用/解复用
- 可靠数据传输
- 流量控制
- 拥塞控制

## ❖ Internet的传输层

- UDP
- TCP

## ❖ 下一章

- 离开网络“边界”
- 进入网络“核心”





哈爾濱工業大學

HARBIN INSTITUTE OF TECHNOLOGY

## 立足航天，服务国防，面向国民经济主战场



谢谢!