7. TCP拥塞

7.1 机制

端到端的拥塞控制机制：
- 路由器不向主机有关拥塞的反馈信息
  - 路由器的负担较轻
  - 符合网络核心简单的TCP/IP架构原则
- 端系统根据自身得到的信息，判断是否发生拥塞，从而采取动作

拥塞控制的几个问题：
- 如何检测拥塞
  - 轻微拥塞
  - 拥塞
- 控制策略
  - 在拥塞发送时如何动作，降低速率
    - 轻微拥塞
    - 拥塞
  - 在拥塞缓解时如何动作，增加速率

7.2 如何检测拥塞（拥塞感知）

发送端如何探测到拥塞？
- 某个段超时了（丢失事件）：拥塞
  - 超时时间到，某个段的确认没有来
  - 原因1：网络拥塞（某个路由器缓冲区没空间了，被丢弃）概率大
  - 原因2：出错被丢弃了（各级错误，没有通过检验，被丢弃）概率小
  - 一旦超时：就认为拥塞了，有一定误判，但是总体控制方向是对的
- 有关某个段的3个冗余ACK：轻微拥塞　即一共4个ACK
  - 段的第一个ACK:正常，表示确认接收
  - 段的第二个ACK：说明乱序
  - 段的第三四个ACK：ACKx表示x后三四个段到了，x段可能丢失了且可能性很大
  - 此时x计时器还没超时，但是直接重发x---->快速重发
  - 网络这时还能够进行一定程度的传输，拥塞，但是情况比第一种好

7.3 如何控制发送方注入速率

维持一个拥塞窗口的值：CongWin
- 以字节为单位，在发送方未确认的情况能注入多少字节

发送端限制已发送但是未确认的数据量（的上限）
- CongWin>=LastByteSent-LastByteAcked

从而粗略的控制发送方往网络中注入的速率
- Rate=CongWin / RTT
- 为什么粗略：因为RTT不确定，导致速率是不精确的

CongWin是动态的，感知到的网络拥塞程度的函数
- 超时或者3个冗余ACK，CongWin：
  - 超时：CongWin降为1MSS(最长报文字段)，进入SS阶段然后再倍增到CongWin/2，从而进入CA阶段
  - 三个冗余ACK：CongWin降为CongWin/2，CA阶段
- 否则（正常收到ACK，没有发生以上情况）：CongWin跃跃欲试
  - SS阶段 慢启动：加倍增加 （每间隔1个RTT加倍）
  - CA阶段 拥塞避免：线性增加（每间隔1个RTT就增加）

7.4 TCP拥塞控制和流量控制的联合动作

联合控制的方法：
- 发送端控制发送但是未确认的量，同时也不能超过接收窗口，满足流量控制要求
  - SendWin=min（CongWin，RecvWin）

RecvWin：通过捎带技术返回过来的字段，代表缓冲区余量
同时满足 拥塞控制和流量控制的要求

7.5 策略描述

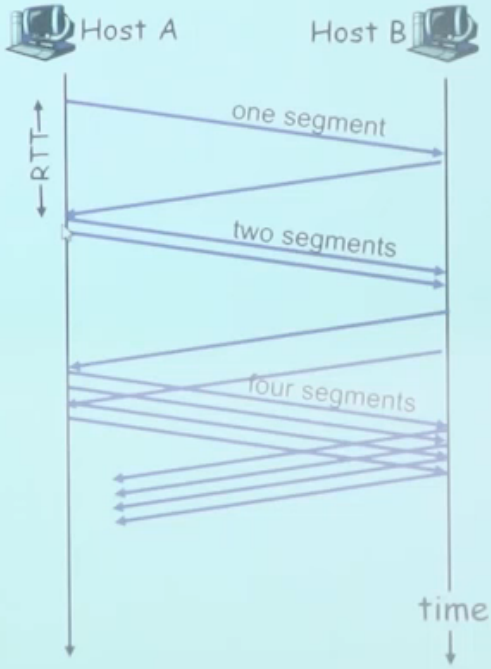
拥塞控制策略：
慢启动：SS Slow-Start
连接刚建立时，CongWin=1MSS
如MSS=1460Bytes&RTT=200msec
初始速率=58.4kbps
可用带宽可能>=MSS/RTT
应该加快速率，到达希望的速率，每个RTT加倍
当连接开始时，指数线性增加发送速率，直到发生丢失时间
启动初值很低
但是速度很快
AIMD：
线性增加，乘性减少
乘性减少：速度减半
线性增加：一个RTT加1个MSS
超时时间后的保守策略

重复收到三个冗余ACK后：
CongWin减半
窗口（缓冲区大小）线性增长
当超时时间发生时：
CongWin被设置成1MSS，进入SS阶段
窗口指数增长
增长到一个阈值（上次拥塞窗口的1/2）时候再线性增加
改进：
Q：什么时候指数增长变成线性增长？
A：超时之前，当CongWin编程上次发生超时的窗口的一半
实现：
变量：Threshold
出现丢失：Threshold设置成CongWin的一半

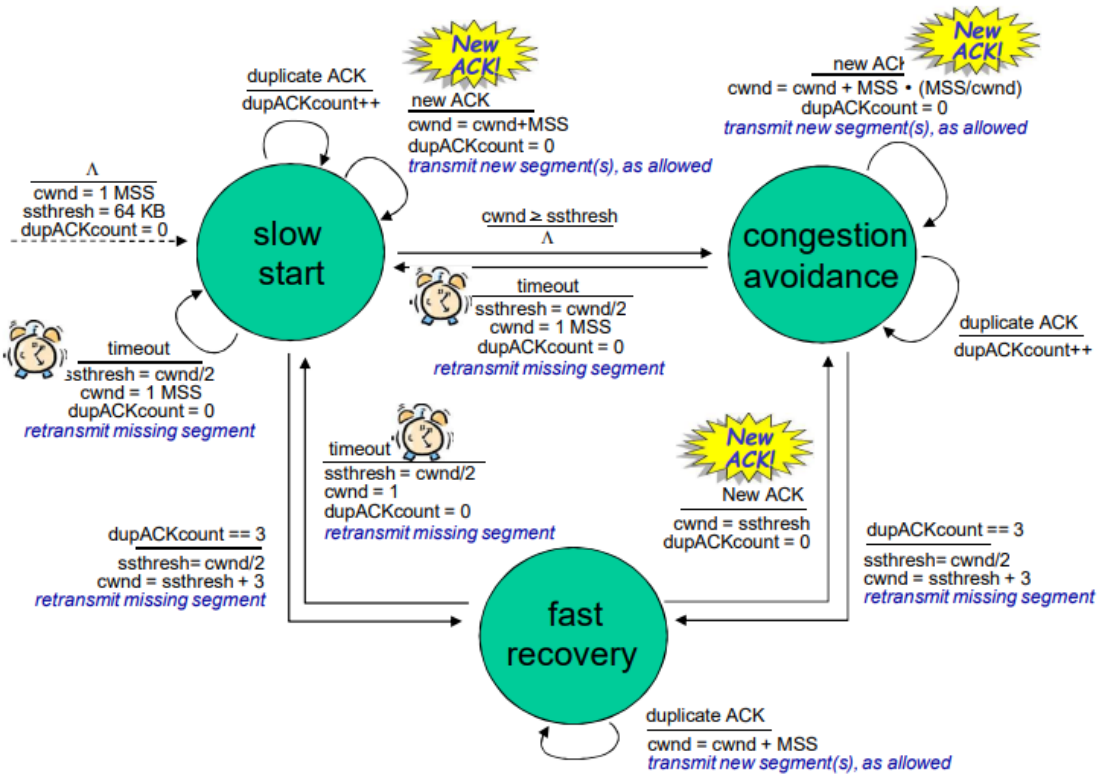
总结：
当CongWin < Threshold,发送端处于慢启动阶段SS，窗口指数增长
当CongWin>Threshold，发送端处于拥塞避免状态CA（congestion-avoidance），窗口线性增长
当收到三个重复ACKs，Threshold设置成CongWin的一般
当超时时间发生时Timeout，Threshold=CongWin/2，CongWin=1MSS，进入SS阶段

TCP 慢启动（续）

- 当连接开始时，指数性增加（每个RTT）发送速率直到发生丢失事件
  - 每一个RTT，CongWin加倍
  - 每收到一个ACK时，CongWin加1（why）
  - 慢启动阶段：只要不超时或3个重复ack，一个RTT，CongWin加倍
- 总结：初始速率很慢，但是加速却是指数性的
  - 指数增加，SS时间很短，长期来看可以忽略



总结：TCP拥塞控制



7.6 TCP吞吐量

使用窗口window尺寸为W，和RTT描述

忽略慢启动，假设发送端总有数据传输
W：发生丢失事件时窗口尺寸
平均窗口尺寸=（w+0.5w）/2=0.75w
平均吞吐量=平均窗口尺寸/RTT

7.7 TCP公平性

公平性目标：如果K个TCP会话分享一个链路带宽为R的瓶颈，每一个会话的有效带宽为R/K

每次降为一半的时候双方速率就会靠近，一直趋向相等

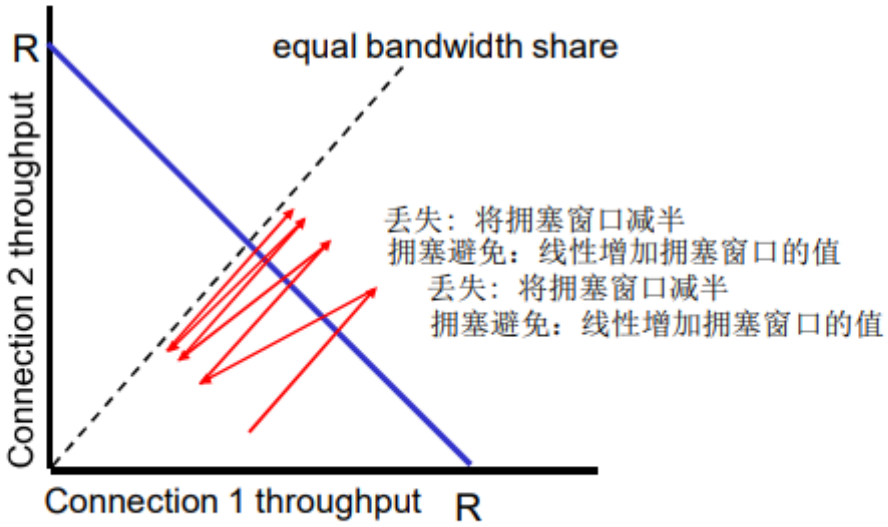
公平性和UDP：
多媒体应用通常不是用TCP
应用发送的数据速率希望不受到拥塞控制的节制
使用UDP
音视频应用泵出数据的速率是恒定的，忽略数据的丢失
研究领域：TCP友好性

公平性和并行TCP连接
两个主机间可以打开多个并行的TCP连接
Web浏览器
例如：带宽为R的链路支持了9个连接
如果新的应用要求建1个TCP连接，获得带宽的R/10
如果新的应用要求建11个TCP连接，获得带宽的R/2

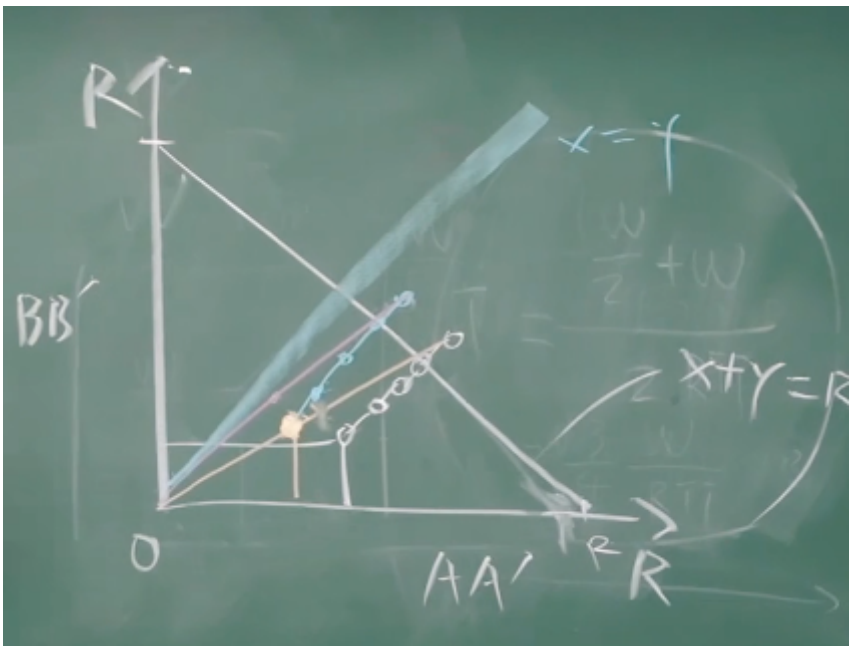
为什么TCP是公平的?

2个竞争的TCP会话:

- 加性增加，斜率为1, 吞吐量增加
- 乘性减，吞吐量比例减少



Transport Layer 3-139



7.8 网络辅助拥塞控制：

TOS字段中2个bit被网络路由器标记，用于指示是否发生拥塞
拥塞指示被传送到接收主机
在接收方-发送方的ACK中，接收方（在IP数据报中看到了拥塞指示）,设置ECE bit，指示发送方发生了拥塞

7.9 总结

传输层提供的服务
    应用进程间的逻辑通信
        VS网络层提供的是主机到主机的通信服务
    互联网上传输层协议：UDP/TCP

多路复用解复用
    端口：传输层SAP
    无连接的多路复用解复用
    面向连接的多路复用和解复用

实例1：无连接传输层协议UDP
    多路复用解复用
    UDP报文格式
    检错机制：校验和

可靠数据传输原理
    问题描述
    停止等待协议
        Rdt1.0 2.0 2.1 2.2 3.0
    流水线协议
        GBN
        SR

实例2：面向连接的传输层协议-TCP
    概述：TCP特性
    报文段格式
        序号，超时机制及时间
    TCP可靠传输机制
    重传，快速重传
    流量控制
    连接管理
        三次握手
        四次挥手：对称连接解放
    拥塞控制原理
        网络辅助的拥塞控制
        端到端的拥塞控制
    TCP拥塞控制
        AIMD
        SS 慢启动
        超时之后的保守策略