

华中科技大学电信学院 2016

拥塞控制与资源分配

计算机网络
计 算 机 网 络

学习目标



- 掌握拥塞控制的概念，不同拥塞控制机制的分类；
- 掌握基于窗口的TCP拥塞控制机制，理解其与TCP流量控制的区别与联系；
- 理解加性增加、乘性减少、慢启动、快速重传等TCP拥塞控制机制；
- 掌握流量控制的概念，TCP流量控制的基本机制。

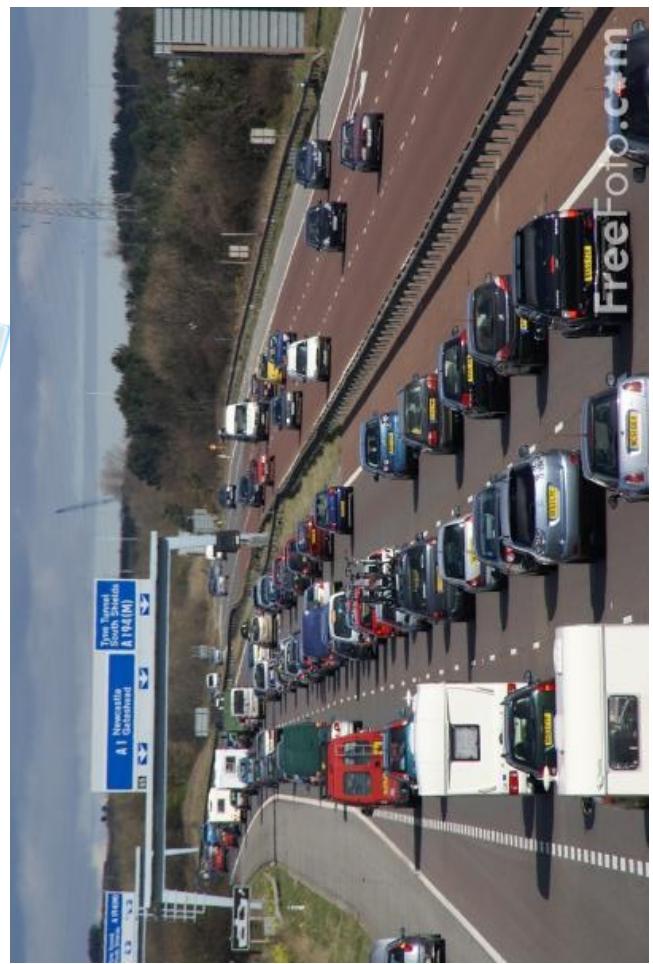
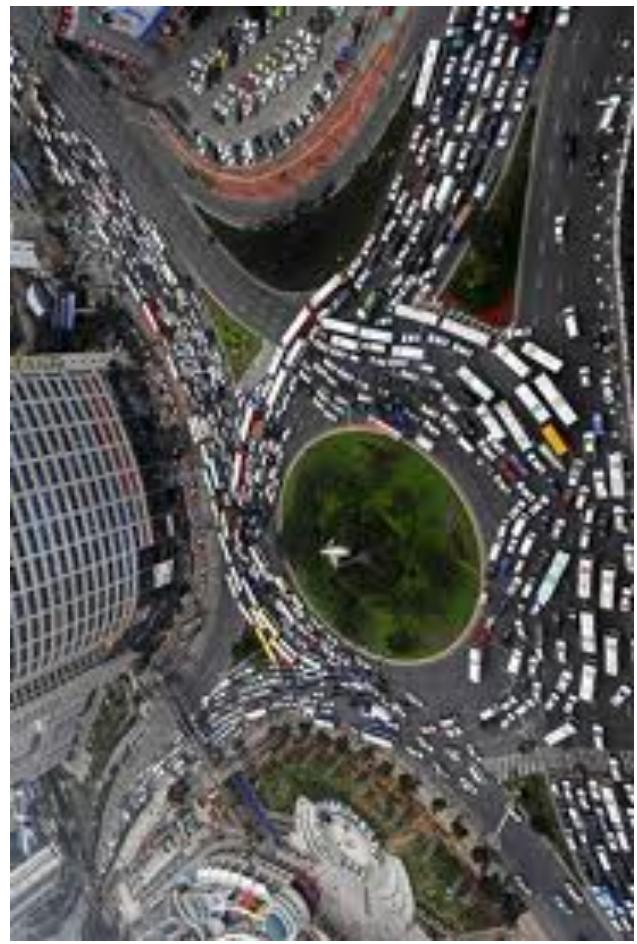
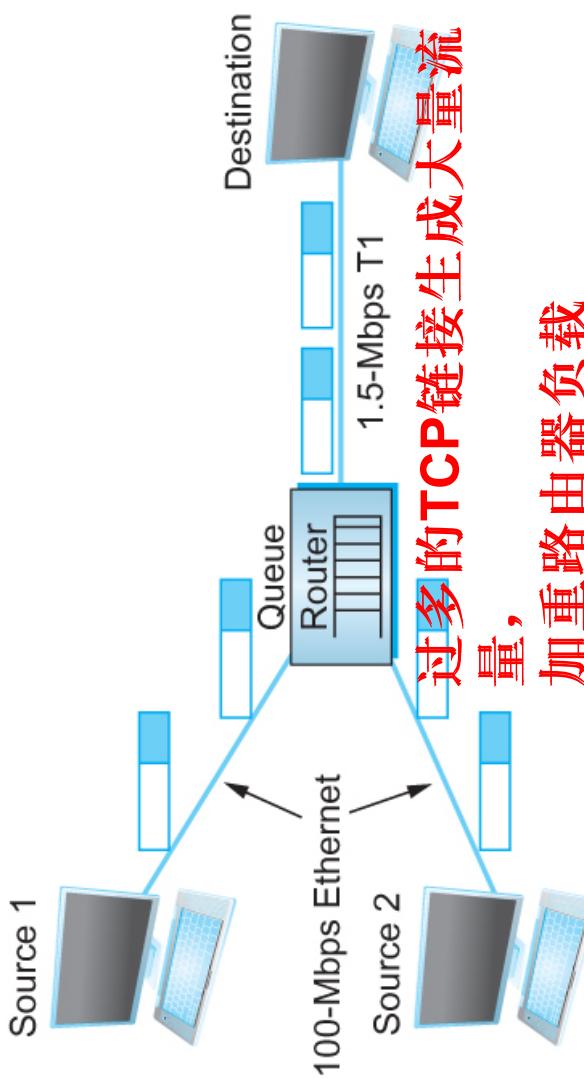
引言

- 核心问题: 分配资源
- 资源分配问题
- 排队规则
- TCP 拥塞控制
- 拥塞避免机制
- 服务质量(QoS)
- 总结



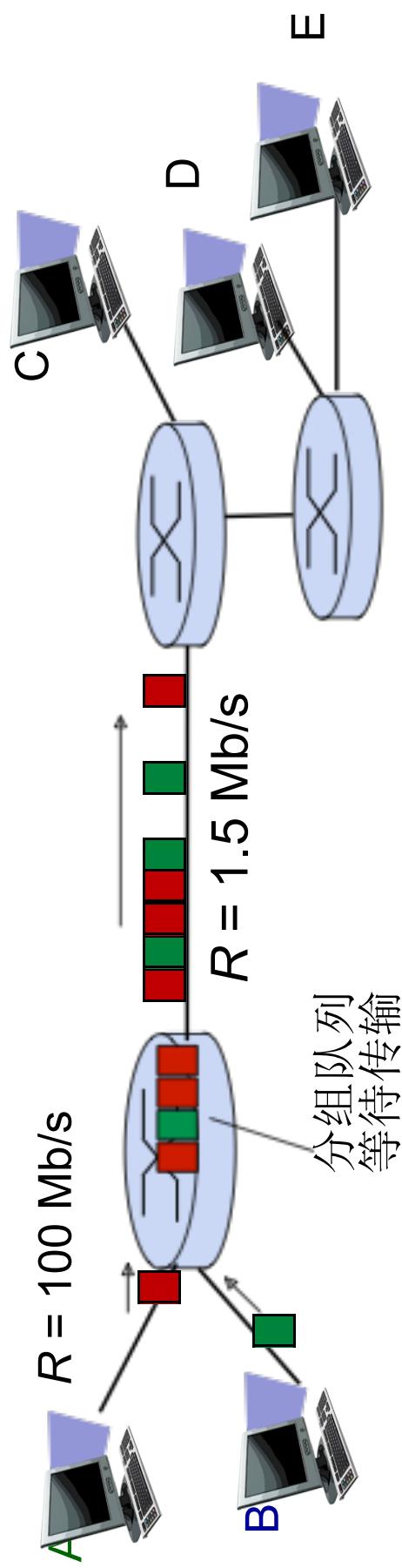


TCP可靠传输面临的新挑战



拥塞

- 当网络负载持续大于其承载能力则发生拥塞
 - 拥塞现象的体现
 - 持续的分组丢失
 - 分组时延不断增加



congestion control

- a top-10 problem!

IEEE Xplore®
Digital Library

Access provided by:
Huazhong University of Science
and Technology
[» Sign Out](#)



BROWSE ▾

MY SETTINGS ▾

GET HELP ▾

WHAT CAN I ACCESS? ▾

congestion control

Search

Basic Search

Author Search

Publication Search

Advanced Search

Other Search Options ▾

Displaying results 1-25 of 20,088 for **congestion control** ✖

Searched for **congestion control** [new search] [edit/save query]

Searched The ACM Full-Text Collection: 452,591 records [Expand your search to The ACM Guide to Computing Literature: 2,596,948 records] ?

53,964 results found



Huazhong University of Science and
Technology

[SIGN IN](#) [SIGN UP](#)

[SEARCH](#)

[advanced search]



?

Export Results: bibtex | endnote | acmref | csv

Result 1 - 20 of **53,964**

Refine by People

Names ▾
Institutions ▾
Authors ▾
Editors ▾
Reviewers ▾

Result page: **1** 2 3 4 5 6 7 8 9 10 >=

Sort by: relevance

0

Rethinking congestion control architecture: performance-oriented congestion control

Mo Dong, Qingxi Li, Doron Zarchy, Brighten Godfrey, Michael Schapira



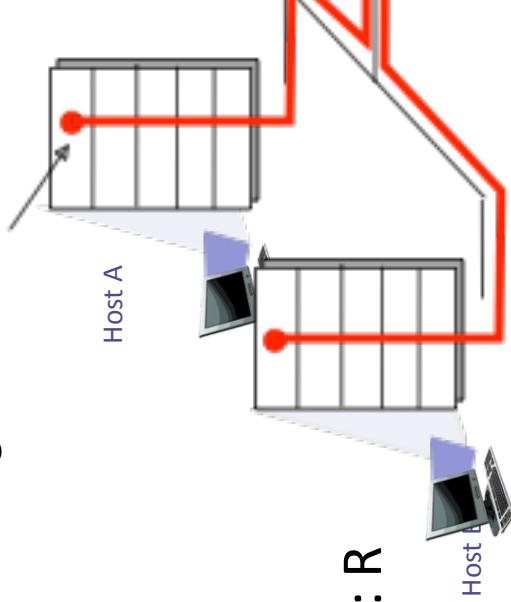
August 2014 SIGCOMM '14: Proceedings of the 2014 ACM conference on SIGCOMM

Causes/costs of congestion: scenario 1

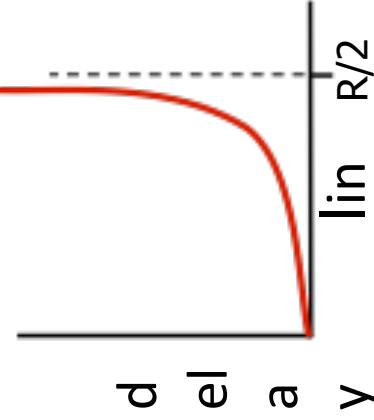
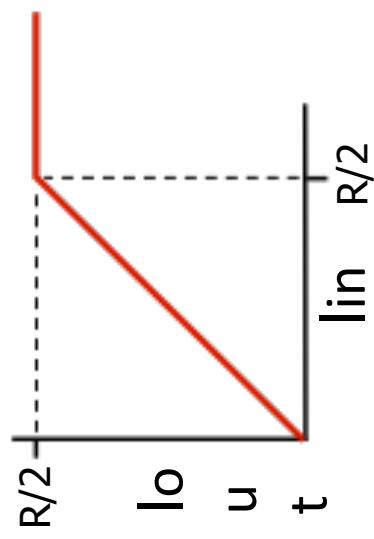
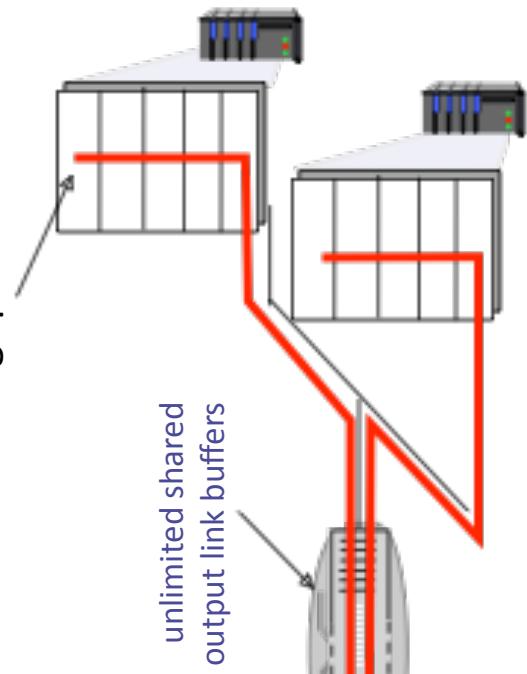


- two senders, two receivers
- one router, **infinite buffers**
- output link capacity: R
- no retransmission

original data: $|in|$



throughput: $|out|$

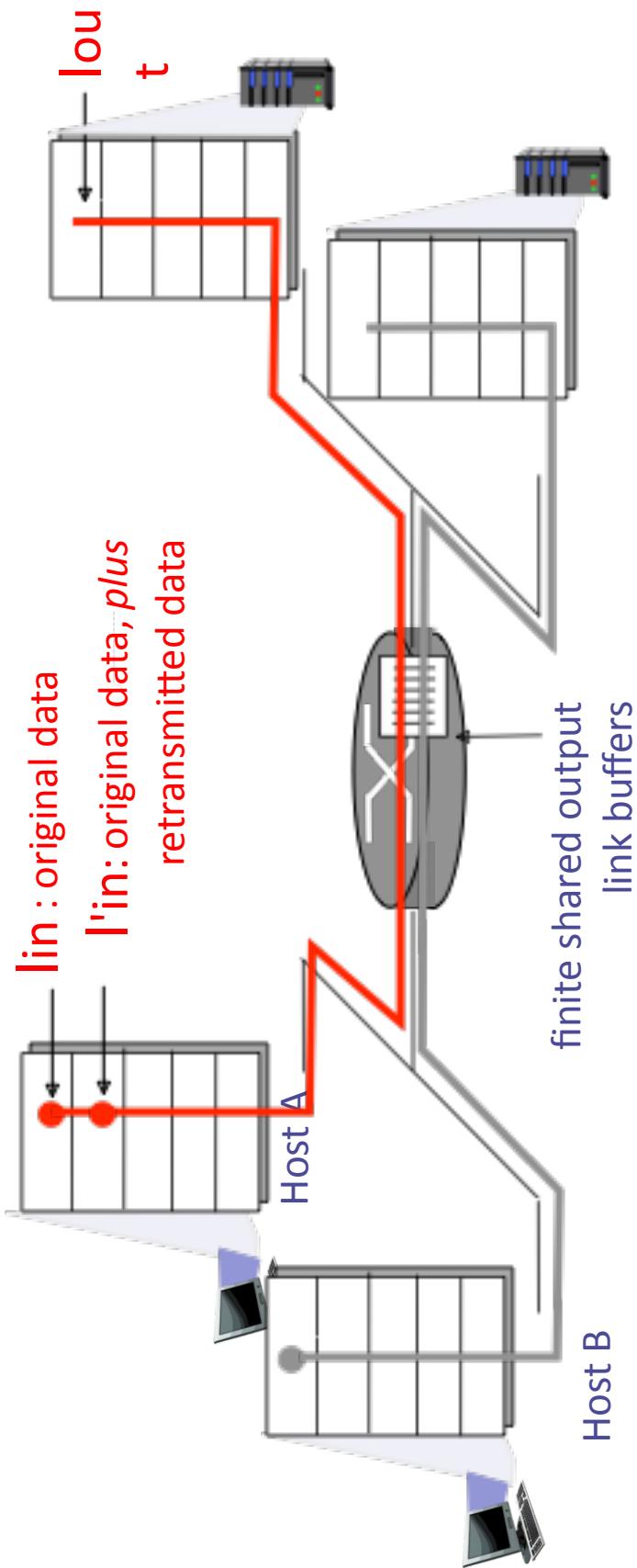


- maximum per-connection throughput: $R/2$
- large delays as arrival rate, $|in|$, approaches capacity



Causes/costs of congestion: scenario 2

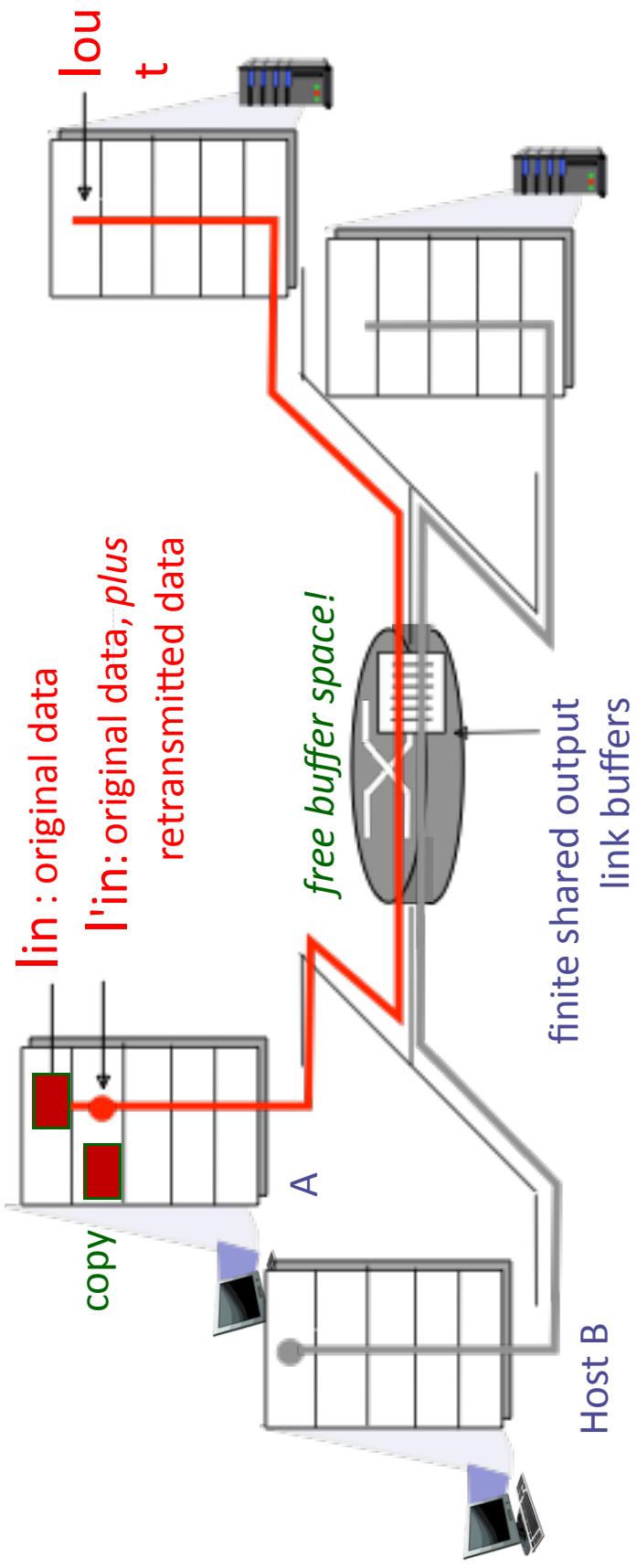
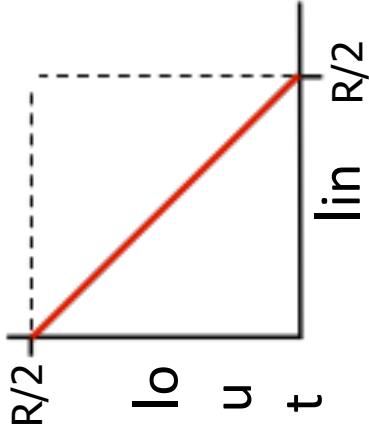
- one router, *finite* buffers
- sender retransmission of timed-out packet
 - application-layer input = application-layer output: $l_{in} = l_{out}$
 - transport-layer input includes *retransmissions*: $l_{in} \geq l_{in}$



Causes/costs of congestion: scenario 2

idealization: perfect knowledge

- sender sends only when router buffers available

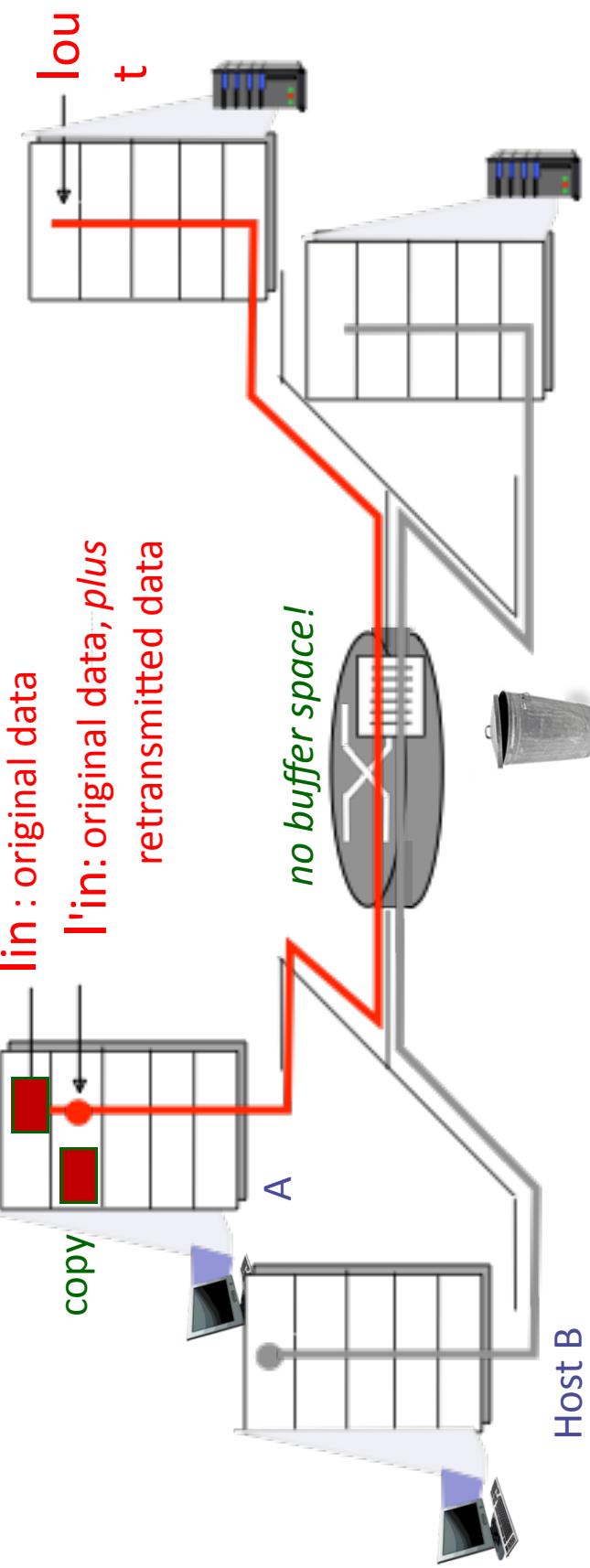


Causes/costs of congestion: scenario 2

Idealization: known

loss packets can be lost, dropped at router due to full buffers

- sender only resends if packet known to be lost

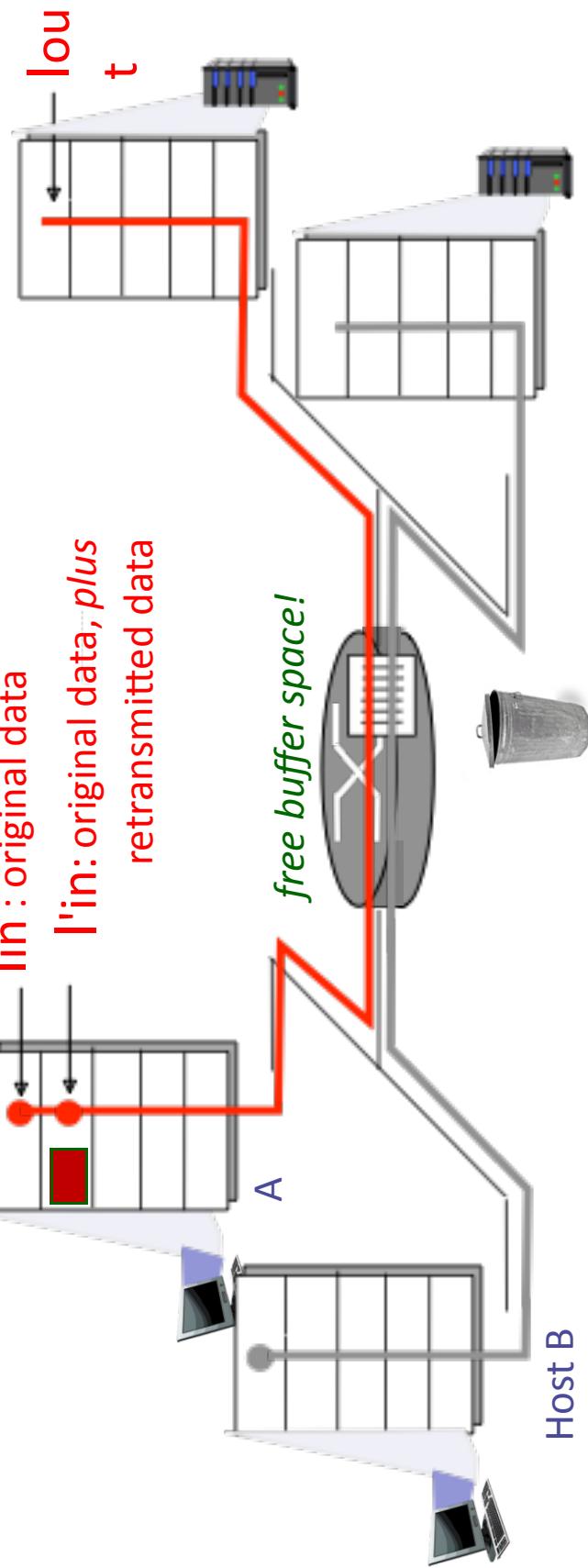
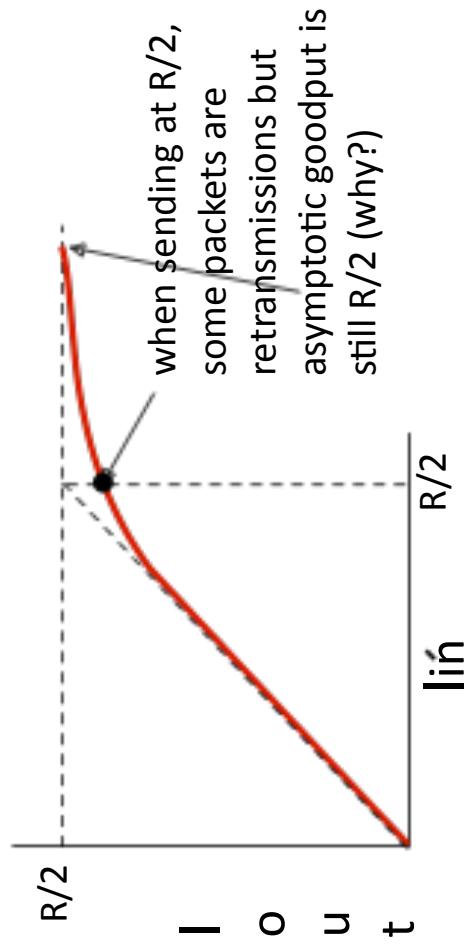




Causes/costs of congestion: scenario 2

Idealization: known

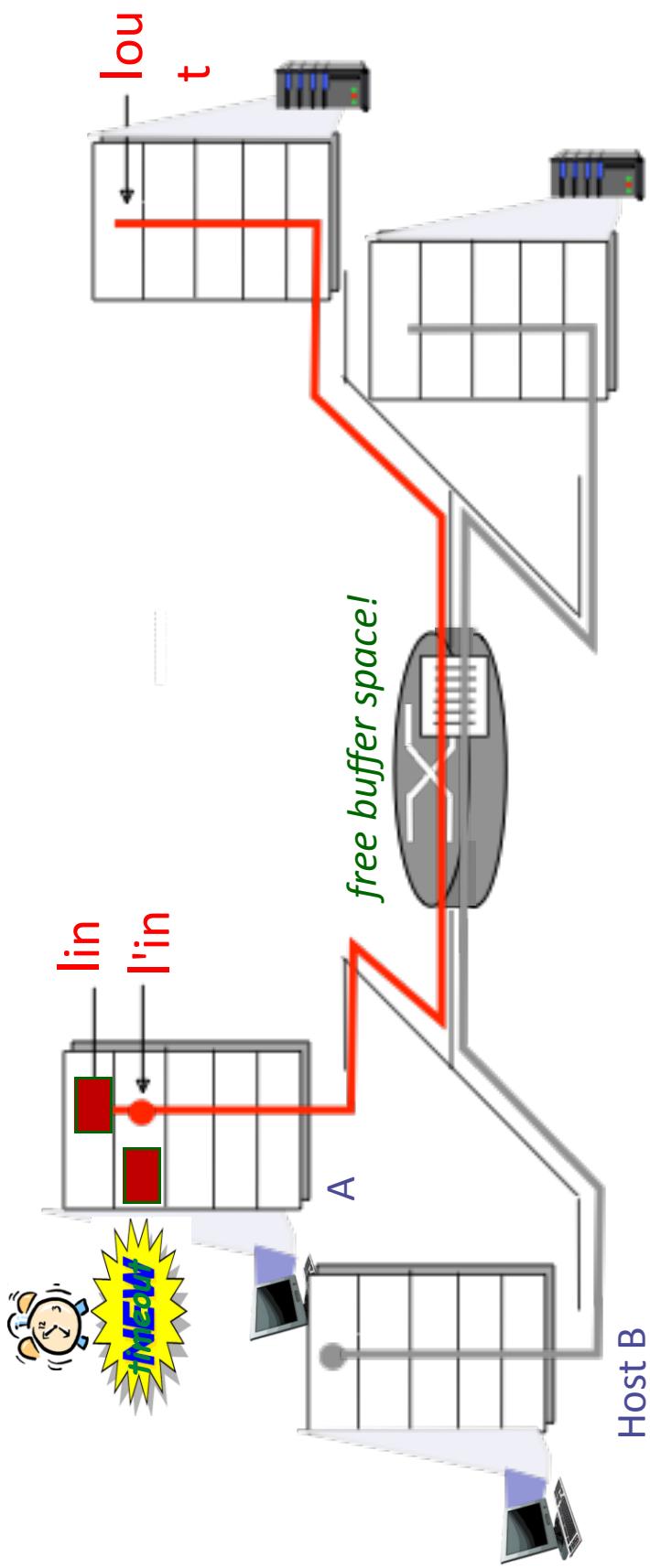
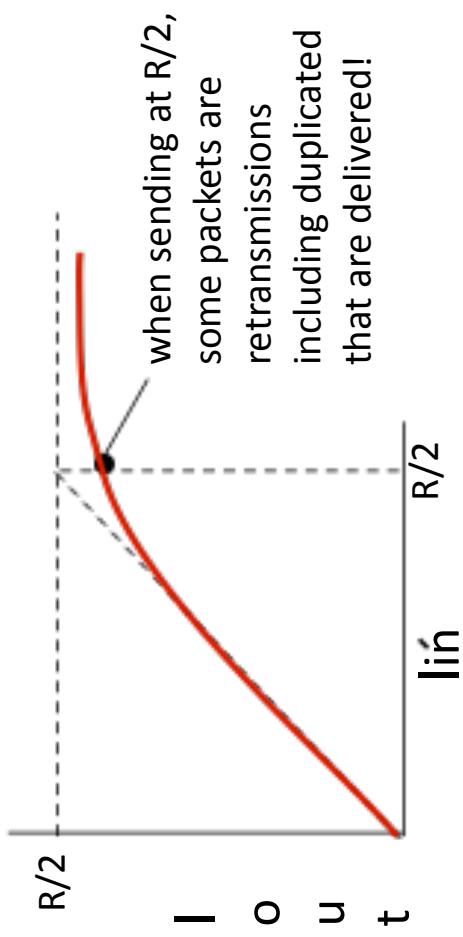
- loss** packets can be lost, dropped at router due to full buffers
- sender only resends if packet known to be lost



Causes/costs of congestion: scenario 2

Realistic: duplicates

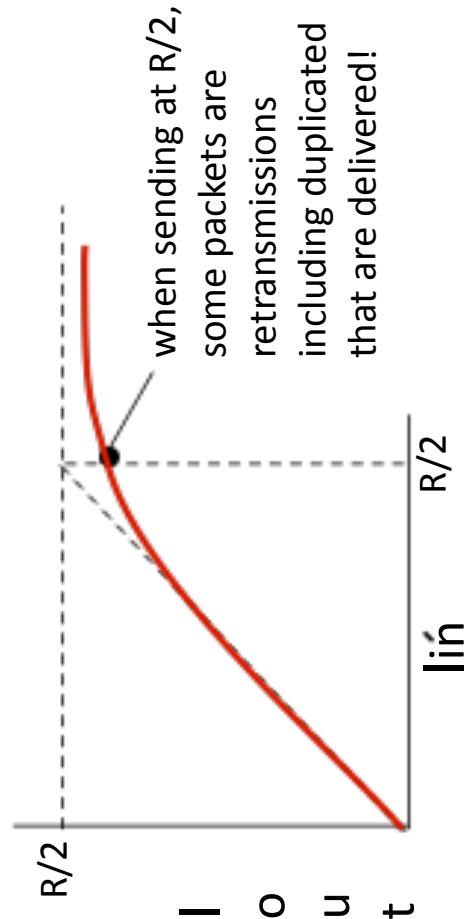
- packets can be lost, dropped at router due to full buffers
- sender times out prematurely, sending **two** copies, both of which are delivered



Causes/costs of congestion: scenario 2

Realistic: duplicates

- packets can be lost, dropped at router due to full buffers
- sender times out prematurely, sending **two** copies, both of which are delivered

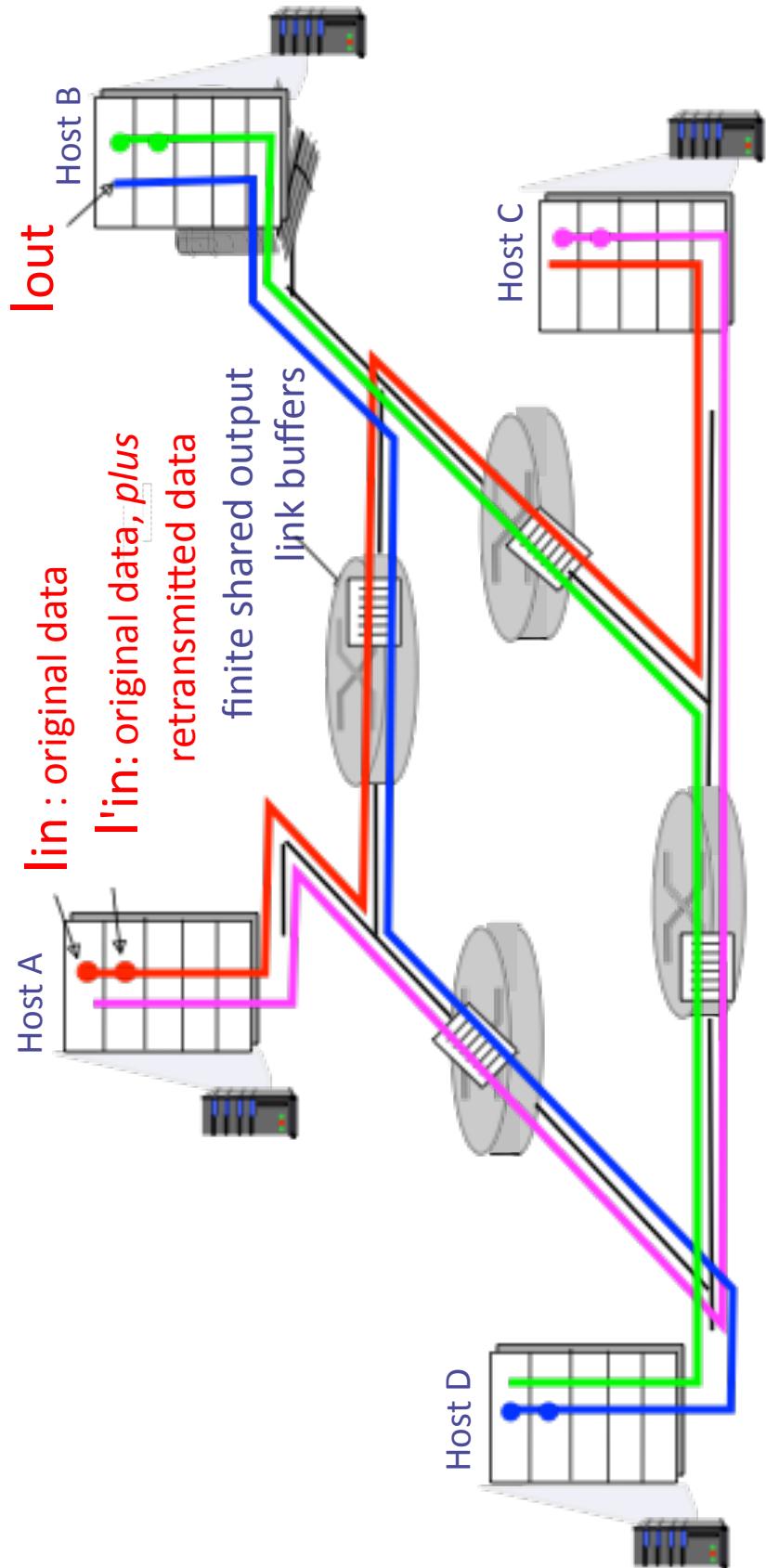


“costs” of congestion:

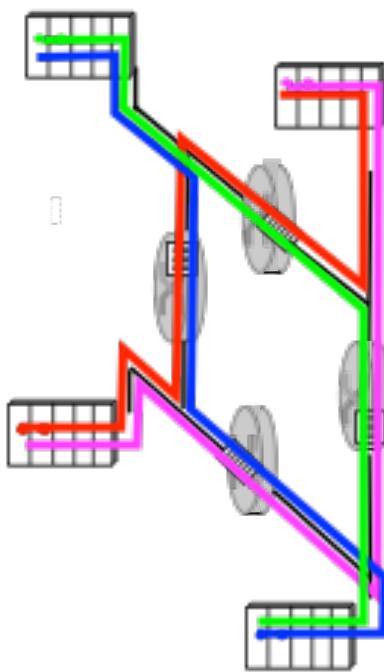
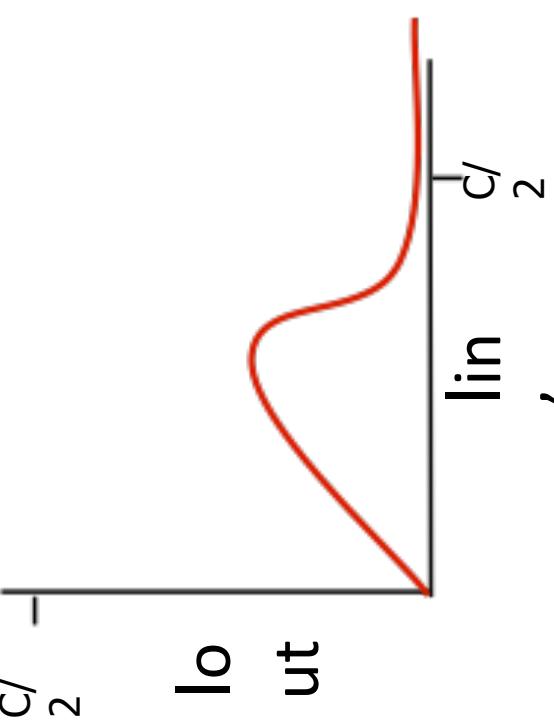
- more work (retrans) for given “goodput”
- unneeded retransmissions: link carries multiple copies of pkt
 - decreasing goodput

Causes/costs of congestion: scenario 3

- four senders
 - multihop paths
 - timeout/retransmit
- Q:** what happens as l_{in} and $l_{in'}$ increase ?
- A:** as red $l_{in'}$ increases, all arriving blue pkts at upper queue are dropped, blue throughput $\rightarrow 0$



Causes/costs of congestion: scenario 3



- another “cost” of congestion:**
- when packet dropped, any “upstream transmission capacity used for that packet was wasted!



拥塞控制和资源分配



- 拥塞控制和资源分配是同一事物的两个方面。

- 以任意精度分配资源非常困难
资源是分布式部署在网络中

- 用户的观点
主机到主机，端到端协议

- 网络的观点
 - 多个“流”竞争资源
资源：交换机或路由器的缓存，链路的带宽
 - 拥塞：过多的分组竞争同一条链路，队列溢出导致分组丢弃
 - 如何在竞争的用户之间合理有效的分配资源？

拥塞控制和资源分配



- 拥塞控制和资源分配是同一事物的两个方面。

- 如果网络承担积极主动角色
 资源分配
- 预先进行网络资源分配,例如,调度虚电路占用的物理链路
- 如果网络承担消极被动角色
 拥塞控制
- 允许发送方想发多少数据就发多少数据,当拥塞发生后再进行恢复



核心問題

分配資源

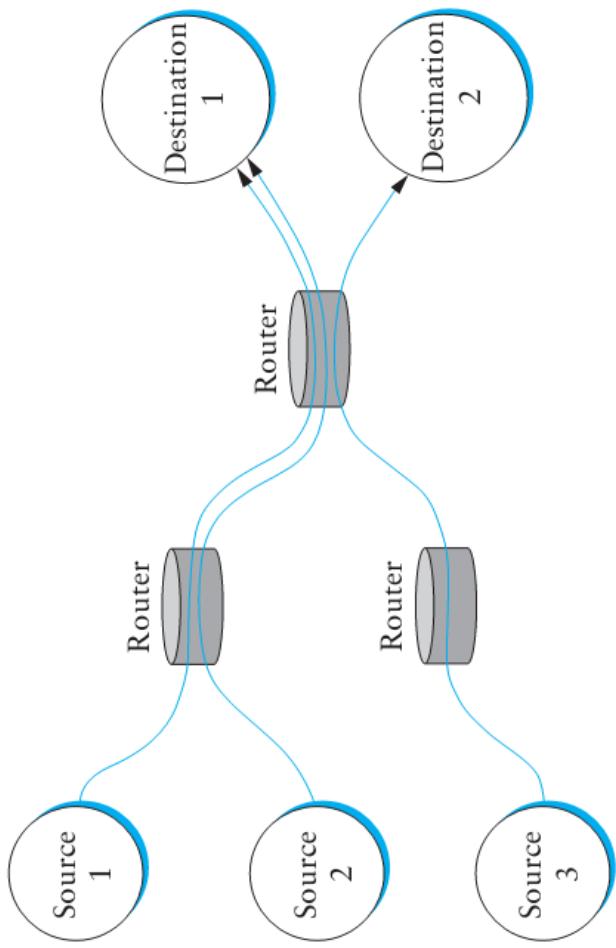


- 引言
- 核心问题: 分配资源
- 资源分配问题
- 排队规则
- TCP 拥塞控制
- 拥塞避免机制
- 服务质量
- 总结

网络模型



- 分组交换网
- 无连接流
- 会话开始时无资源预留
- 流可以有不同粒度的定义(进程到进程, 源主机到目的主机, 等.)



- 服务模型
- 最大努力交付: 所有分组采用相同的方式处理



资源分配机制的分类

- 以路由器为中心vs.以主机为中心
 - 以路由器为中心: 路由器决定什么时候转发分组, 丢弃哪些分组, 通知主机允许的发送速率
 - 以主机为中心: 端主机观测网络状态
 - 以路由器为中心和以主机为中心并不完全相互排斥
- 基于预留方式 vs. 基于反馈方式
 - 基于预留方式: 事先预留资源, 如果资源无法获得则拒绝流
 - 基于反馈方式: 端主机在未预留任何容量的情况下发送数据
 - 显式或隐式反馈
- 基于窗口方式 vs. 基于速率方式
 - 基于窗口方式: 采用与可靠传输相同的窗口机制
 - 基于速率方式: 接收方或网络控制速率

资源分配性能评估指标

评估指标用来定量分析资源分配的有效性

常用指标

吞吐量

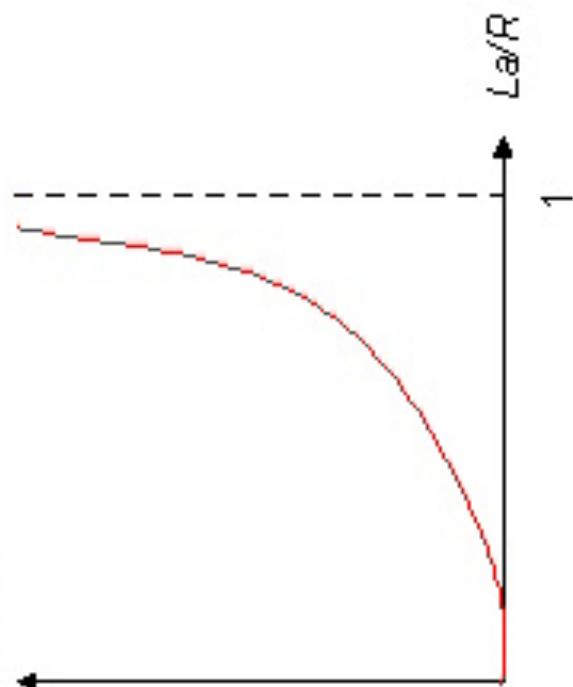
时延

平衡性能

链路利用率越高(接近100%)，
吞吐量越大

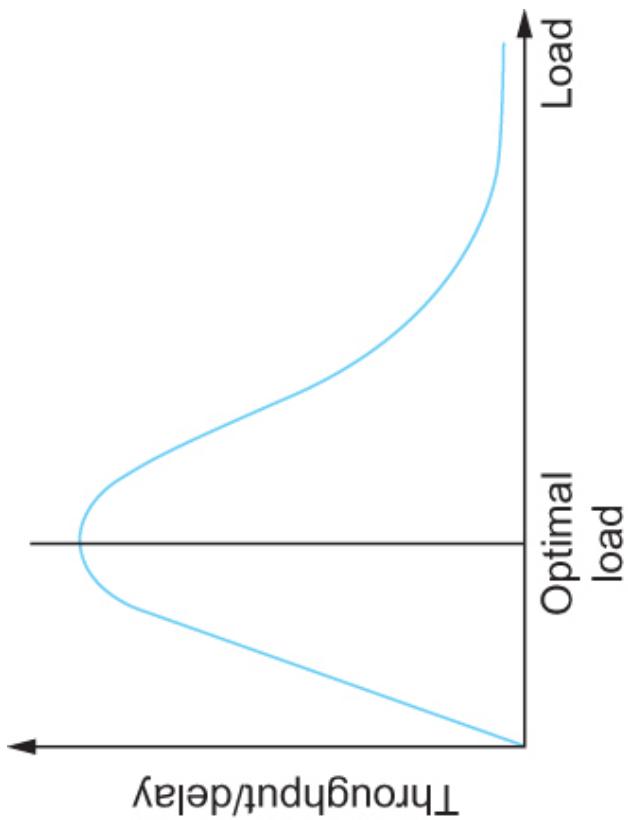
如果链路空闲，吞吐量会减小
链路利用率越高，在路由器缓存中等待传输的分组数量越长，增加了队列长度，延长了时延

average
queueing delay



资源分配性能评估指标

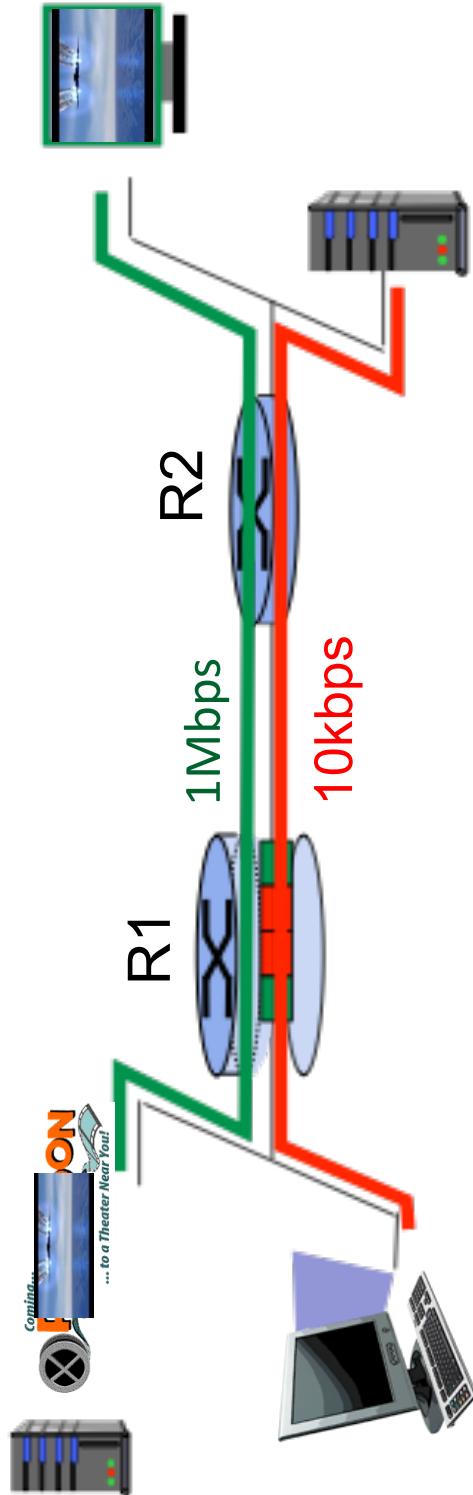
- **有效资源分配**
- 指标：
 - 网络能力(Power)
 - 吞吐量(Throughput)
 - 时延(Delay)
- 定义：
 - 能力 = 吞吐量/时延



资源分配性能评估指标

公平资源分配

- 缺乏共识：如何定义公平的资源分配？
- 基于预约的资源分配提供了受控的不公平方式
- 案例：某条链路上，我们可以预约1Mbps带宽用于传输视频流量，而相同链路上一个文件传输流仅获得10kbps

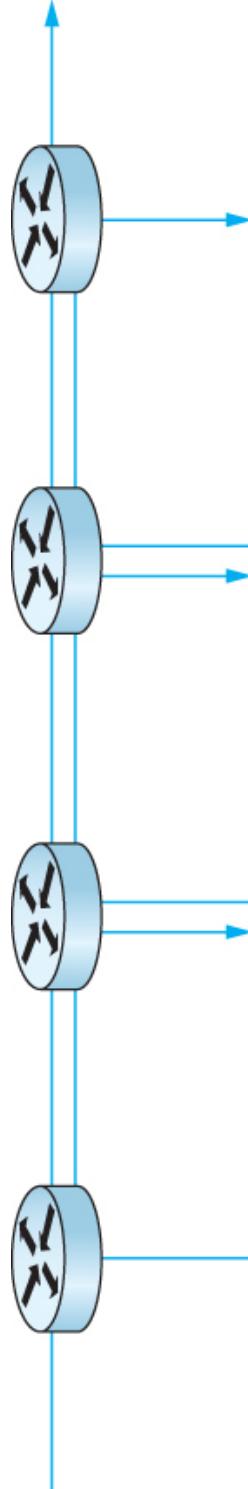




资源分配性能评估指标

公平资源分配

- 缺乏共识：如何定义公平的资源分配？
- 当多个数据流共享一段瓶颈链路，各个流应享有平等带宽份额
- 以上定义假设公平的带宽分配，意味着相同的带宽份额
- 但是，即使不考虑资源预留，相同的带宽可能并不等效于公平份额
- 假如我们考虑端到端的路径长度



1个4跳数据流与三个一跳数据流竞争带宽资源



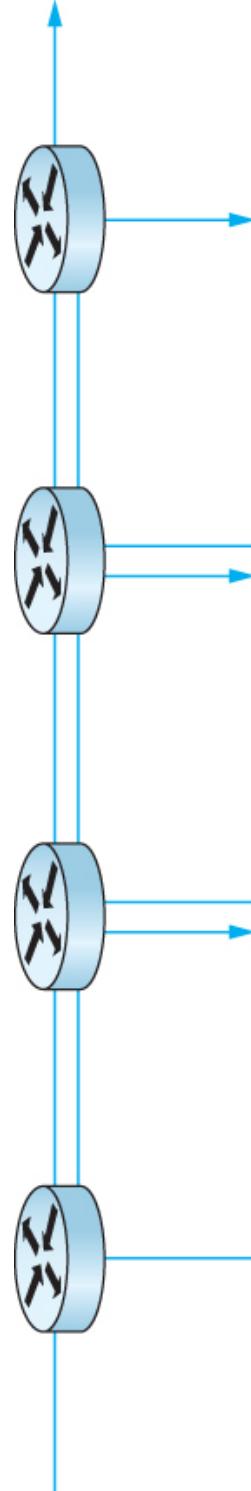
资源分配性能评估指标

公平资源分配

- 假设公平意味着相等，且所有的路径长度相等
- Rai Jain提出公平指教，来评价拥塞控制算法的公平性
- 定义：假设n个数据流的吞吐量(x_1, x_2, \dots, x_n)

$$f(x_1, x_2, \dots, x_n) = \frac{(\sum_{i=1}^n x_i)^2}{n \sum_{i=1}^n x_i^2}$$

- 公平指教是[0,1]的一个实数，1表示最公平



1个4跳数据流与三个1跳数据流竞争带宽资源

拥塞控制的实现



基于主机的拥塞控
制

基于主机的拥塞控
制



基于路由器的拥塞控
制

制



基于主机的拥塞控
制

基于主机的拥塞控
制



拥塞控制方法分类

主要分为两大类：

端到端拥塞控制：

- 网络不提供明确反馈信息
- 终端主机通过观察丢包和时延推测拥塞情况
- TCP所采用

网络协助拥塞控制：

- 路由器向终端系统提供反馈信息

- 单比特表明拥塞 (SNA, DECbit, TCP/IP ECN, ATM)
- 发送端明确控制发送速率

第6章 拥塞控制及资源分配



策略	路由器	主机	章节
基于主机的拥塞控制	FIFO排队	TCP 拥塞控制 6.2, 6.3	
基于路由器的拥塞控制		主动队列管理(AQM) TCP 拥塞控制	
资源分配和拥塞避免		资源预留 提供QoS	6.5

提纲

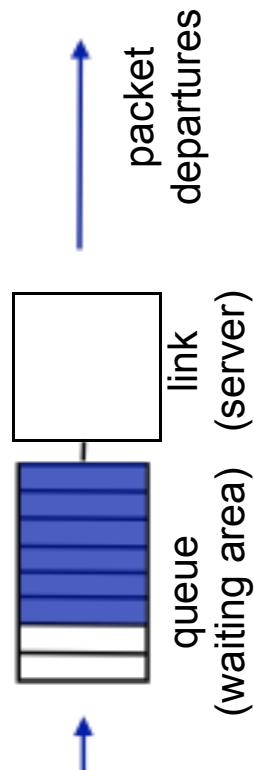
- 引言
- 核心问题：分配资源
- 资源分配问题
- 排队规则
 - FIFO
 - 公平排队
- TCP 拥塞控制
- 拥塞避免机制
- 服务质量
- 总结



排队规则: FIFO



- 调度算法 vs 缓存管理
 - 调度算法:** 选择排队的分组在链路上传输的方式
 - 缓存管理:** 当链路缓冲区已满, 对于新到达的分组是选择丢弃还是移除其他分组来容纳该分组?
- FIFO: First-In-First-Out / FCFS: First-Come-First-Served**
- 按照分组到达次序来提供服务
 - 缓存管理: 分组丢弃策略**
 - 丢弃尾: 丢弃刚到达的数据包
 - 优先级: 按照优先级丢弃 / 移除分组
 - 分组随机: 随机的丢弃 / 移除分组

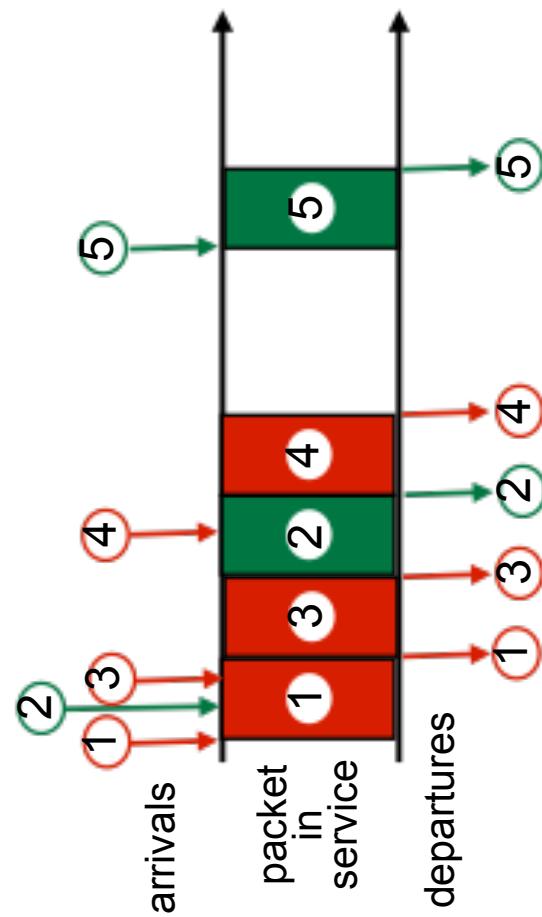
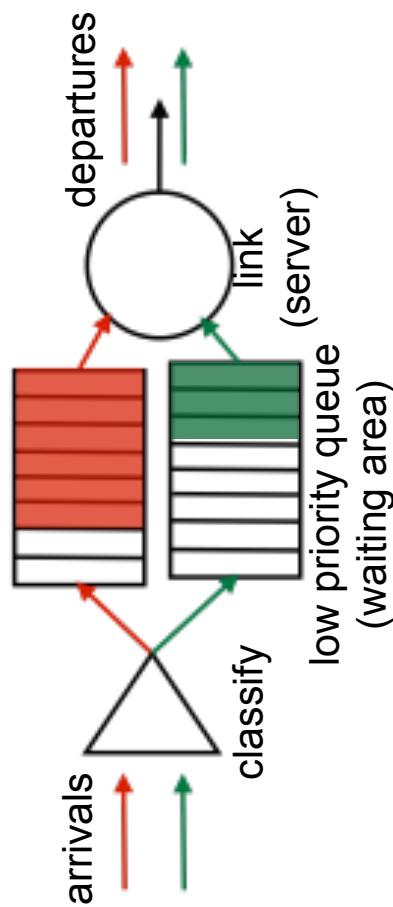


排队规则：优先调度



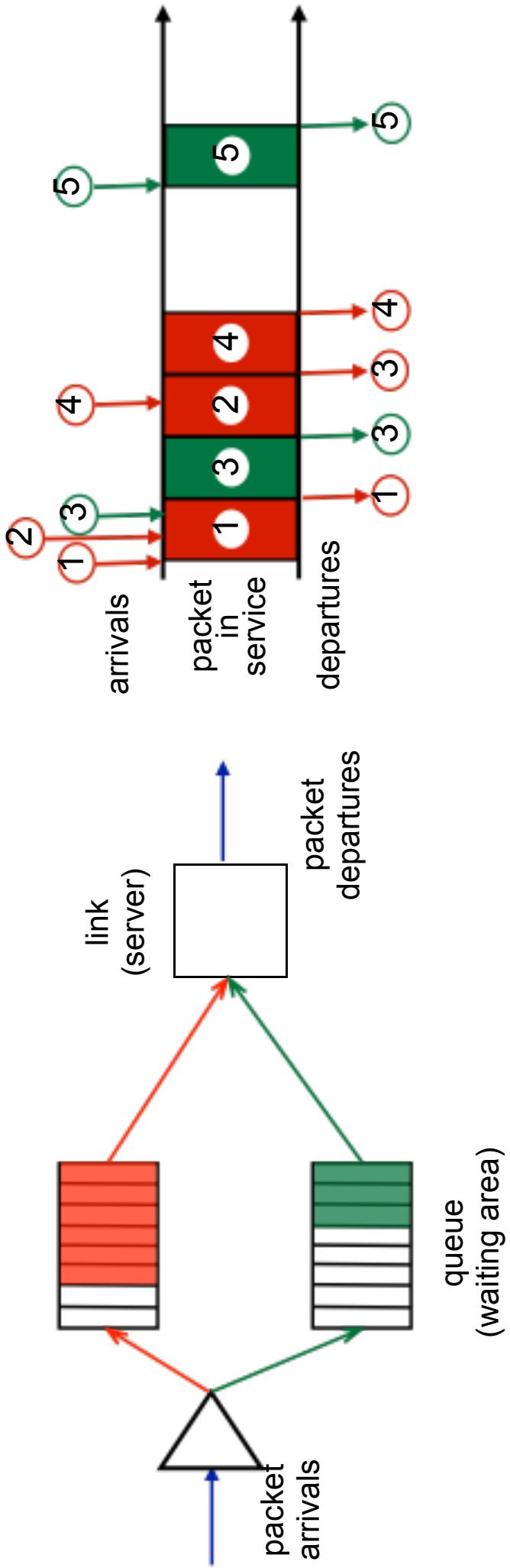
- 优先传输高优先级队列中的分组
- 不同类型的流量具有不同的优先级
- 取决于分组首部的标记(如Tos字段的值), 源或目的地址, 目的端口号等

high priority queue
(waiting area)



排队规则: 循环调度

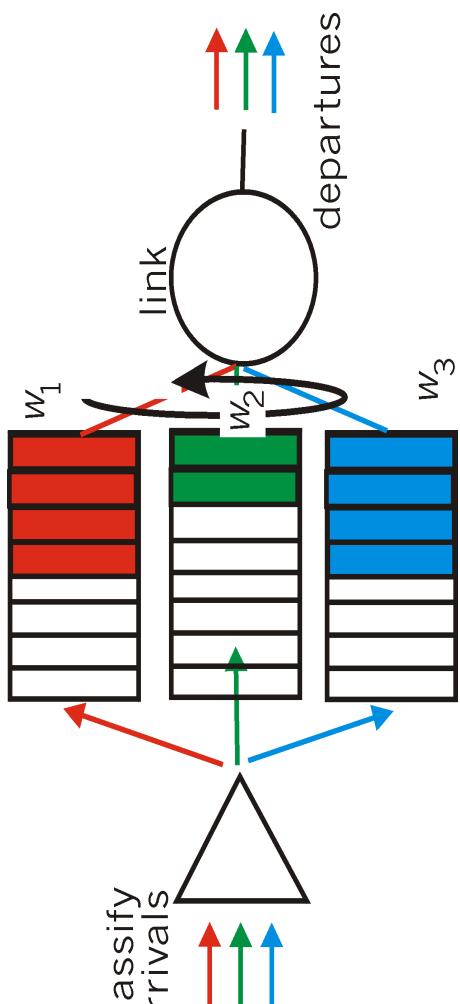
- 分组被分成不同类别(与优先级排队一样)
- 在类之间不存在严格的服务优先级，循环调度器在这些类之间轮流提供服务。



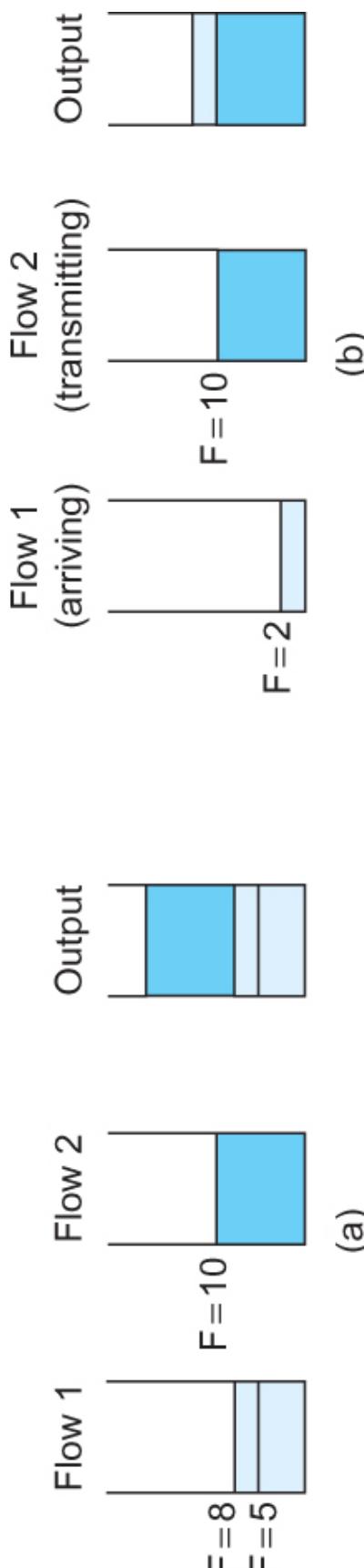
排队规则: 加权公平排队调度



分组分类，在每个类的等待区域排队，调度器循环为各分类分组提供服务。因此每一种类被分配一个加权，因此每一种类在任何时间间隔内可能得到不同的服务。



模拟比特级的循环调度，计算每个分组的发送完毕时间Fi，并按照Fi的先后时间发送分组。





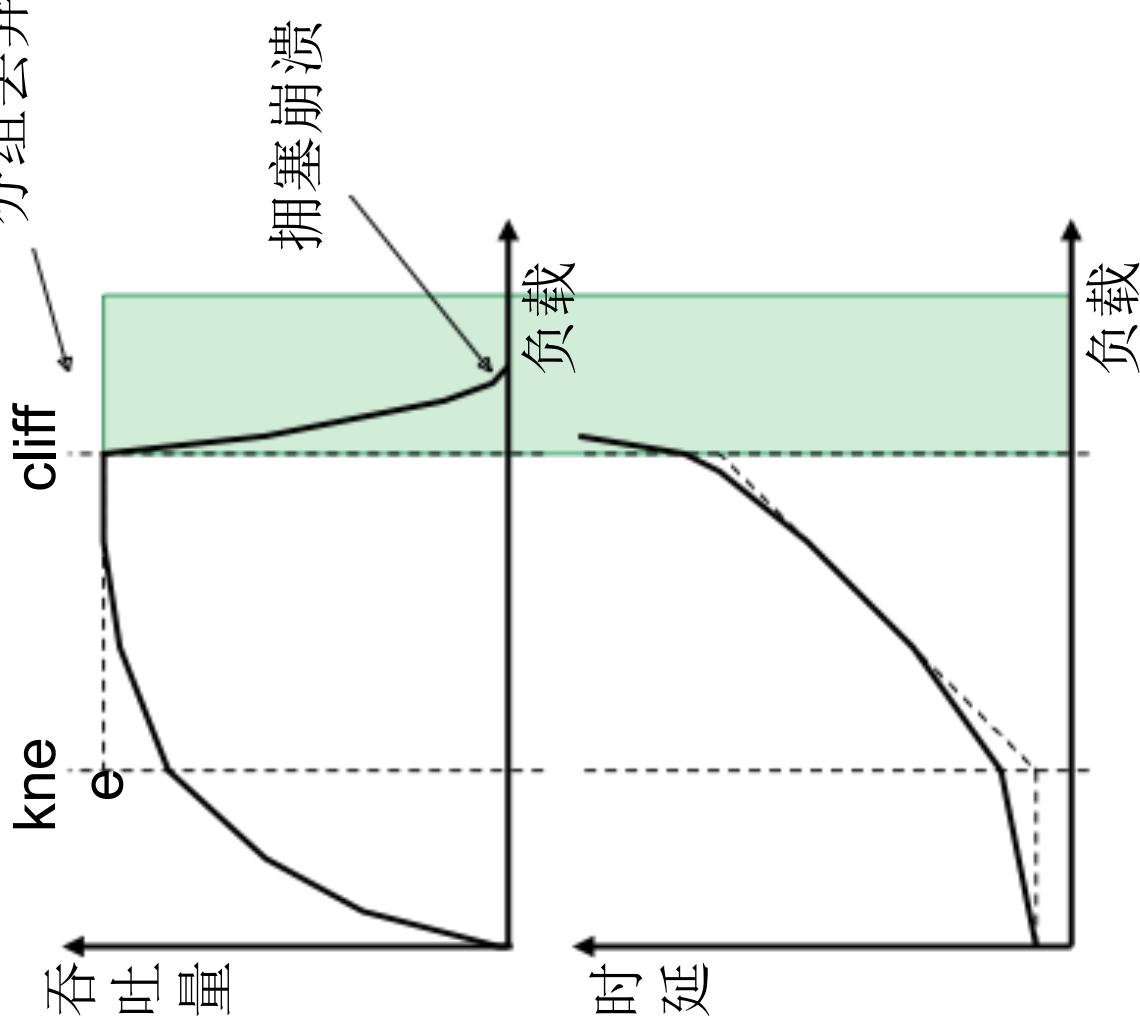
- 引言
- 核心问题: 分配资源
- 资源分配问题
- 排队规则
- TCP 拥塞控制
- 拥塞避免机制
- 流量控制
- 服务质量
- 总结

网络负载和拥塞



- 膝盖
- 吞吐量缓慢增加
- 时延开始快速增长

分组丢弃

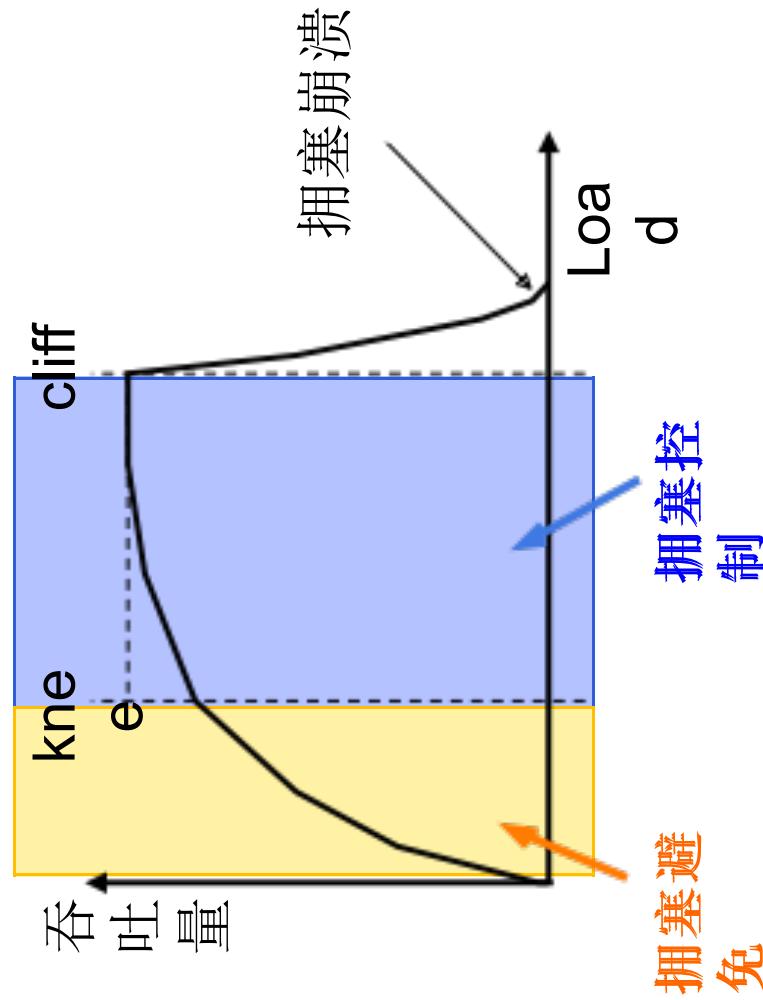


- 悬崖
- 吞吐量开始快速减少直至为0 (拥塞崩溃)
- 时延趋近于无穷大

拥塞控制和避免



- 增加路由器的缓存无法避免拥塞
- 避免拥塞的方式
 - 源端减少负载 - 短期流量工程 - 中长期
 - 不断增加链路的容量 - 长期



TCP 拥塞控制



- TCP 源端调整发送速率以避免网络过载
- 考虑网络的可用容量
- 两个基本问题
 - 如何察觉网络拥塞?
 - 以超时作为网络拥塞的标志
 - 如何调整发送速率?
- TCP 维护一个 CongestionWindow 变量

TCP拥塞窗口



- 每一个TCP的发送方维护一个拥塞窗口
给定时间内允许传送的最大数据量(未确认的)
- 自适应拥塞窗口
 - 分组丢失时减少: 退让
 - 成功传送时增加: 乐观的探索
 - 总是努力找到合适的传送速率
- Tradeoff
 - 优点: 不需要专门的网络反馈机制
 - 缺点: 持续在“合适”的速率上下波动

接收窗口 vs. 拥塞窗口

- 流量控制
 - 避免快速发送方使慢速接收方过载
- 拥塞控制
 - 避免多个发送方使网络过载
- 不同的概念，但是采用类似的机制
 - TCP 流量控制：接收窗口
 - TCP 拥塞控制：拥塞窗口
发送方TCP窗口 = $\min \{ \text{拥塞窗口}, \text{接收窗口} \}$

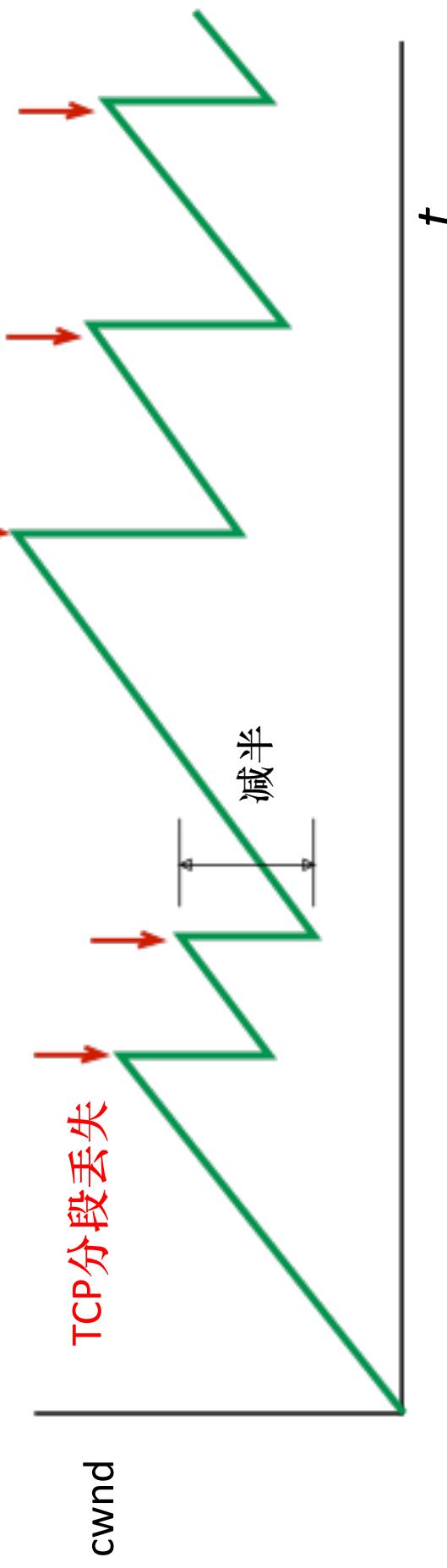


累次增加/成倍减少(AIMD)

- 如何自适应变化?

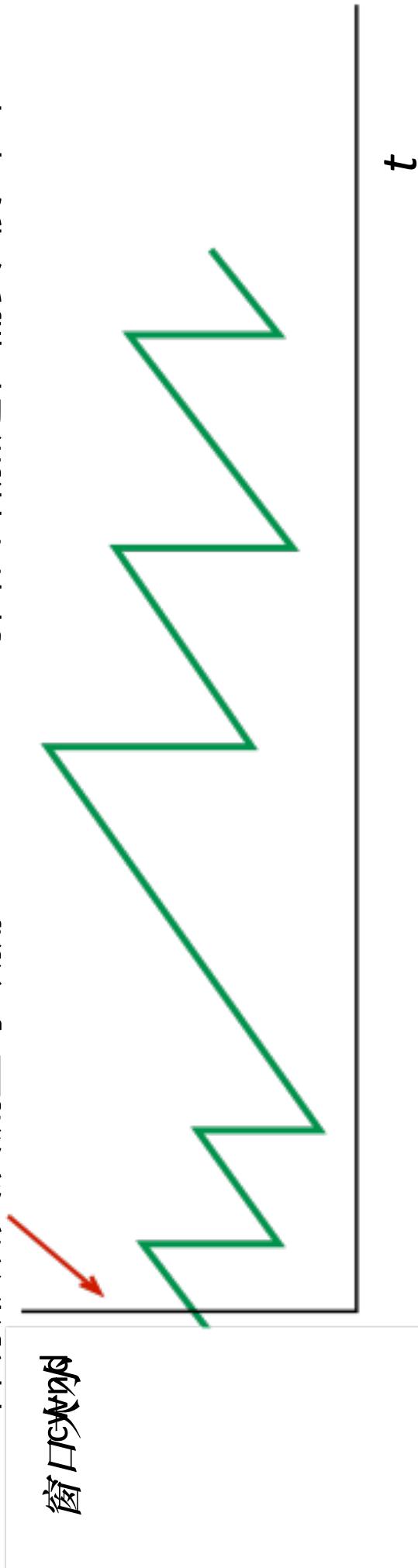
- 分组成功传送,将窗口大小加1,即多传输一个最大数据段大小(MSS)的数据
一旦分组丢失,则将拥塞窗口减半

- 减小拥塞窗口的速度要比加大窗口快得多,称为AIMD
- 加性增加(Additive Increase)
- 乘性减少(Multiplicative Decrease)



新的数据流如何启动？

- 问题
 - 最初的TCP发送窗口直接按照接收方的通知窗口设置，窗口值偏大，容易形成突发流量导致拥塞 \rightarrow cwnd的窗口增加过程需要“慢”下来

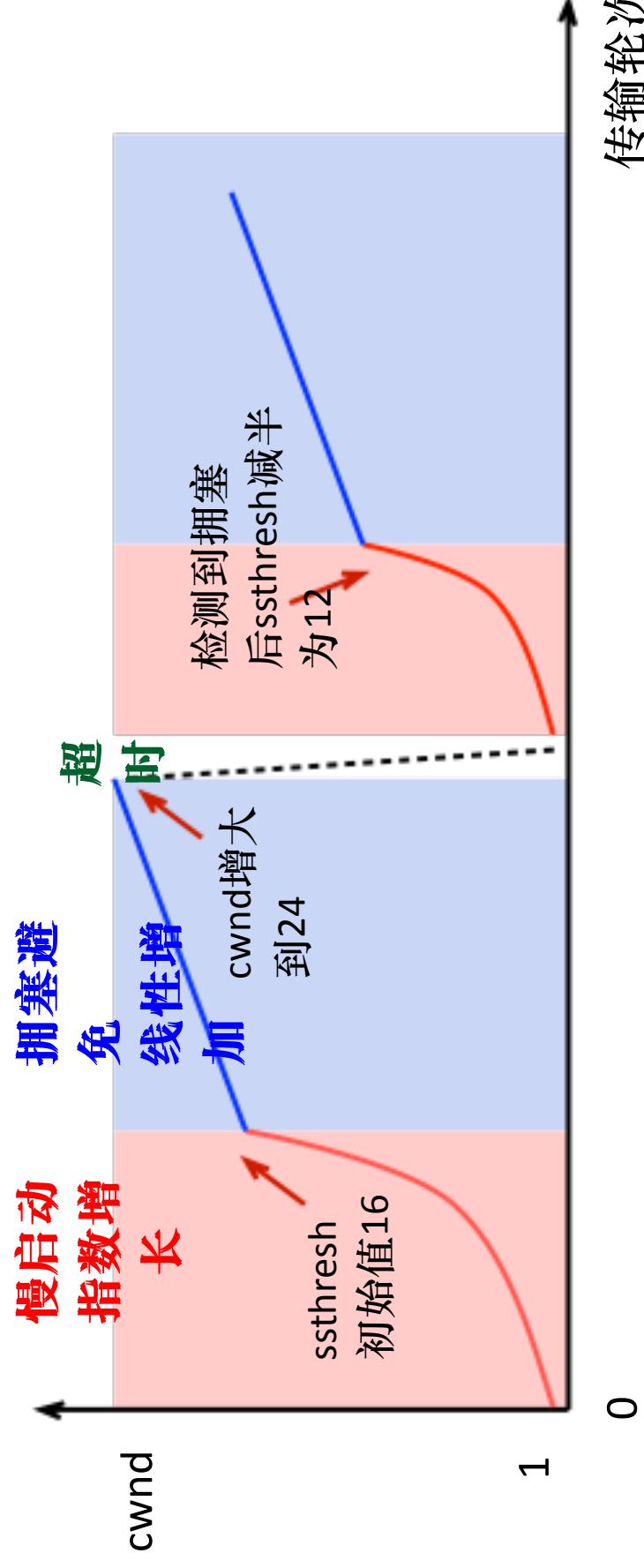


- 需要设计一种机制控制窗口启动阶段的cwnd
 - 初始值不能过大，需要从1开始，需要一个很长的时间
 - cwnd的增加速度要快，加性增加的速度慢了

慢启动



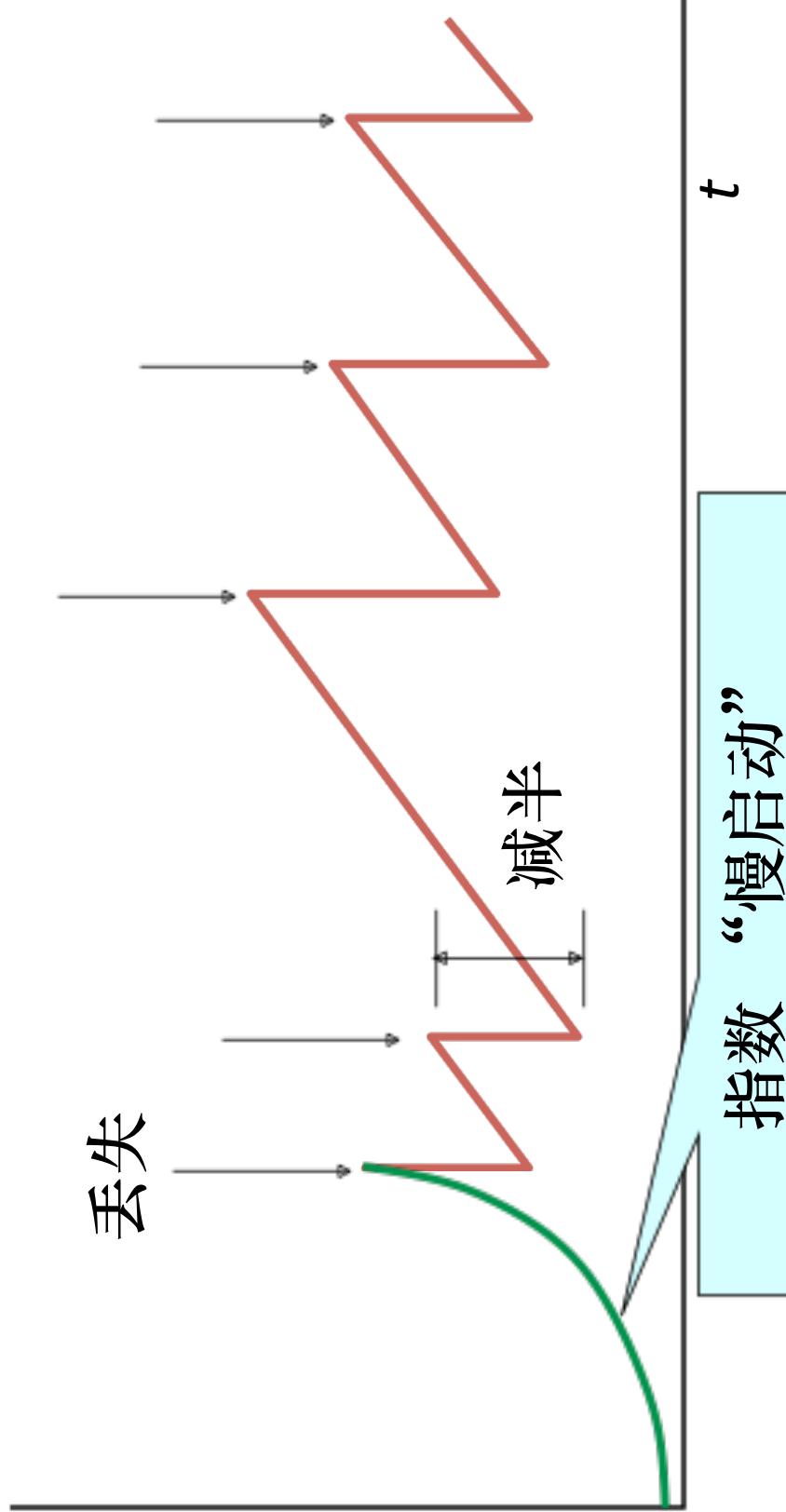
- 慢启动的解决方案
- TCP初始启动时，设置拥塞窗口 cwnd 为1
- 慢启动阶段：以指数方式增加cwnd直到特定门限值ssthresh
 - ssthresh: slow start threshold
- 拥塞避免阶段：超过ssthresh后，cwnd加性增加
- 再次拥塞时，ssthresh值减半，重新进入慢启动阶段





慢启动与TCP锯齿

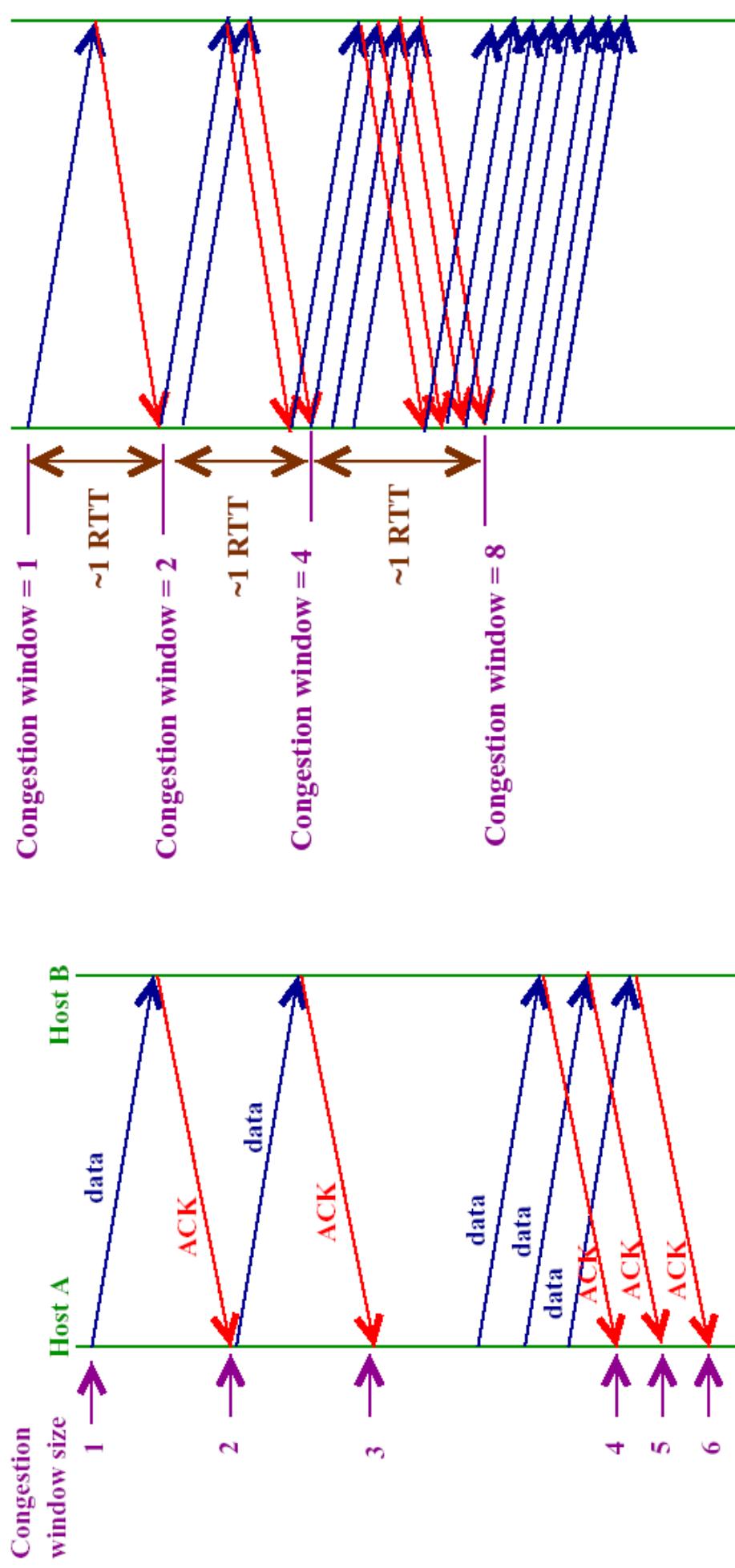
窗口大小



- 这种称呼（“慢”）是因为最初TCP无拥塞控制机制

- 源端启动时发送整个AdvertisedWindow
- 导致拥塞崩溃！

累次增加 vs. 慢启动





TCP的两种丢包

超时

- 分组 n 丢失，超时定时器超时
何时？ n 是窗口的最后一个分组，或者所有发送分组均丢失

超时后，大的CWND会导致更严重的分组丢失

以较小的CWND重新开始

三个重复的ACK

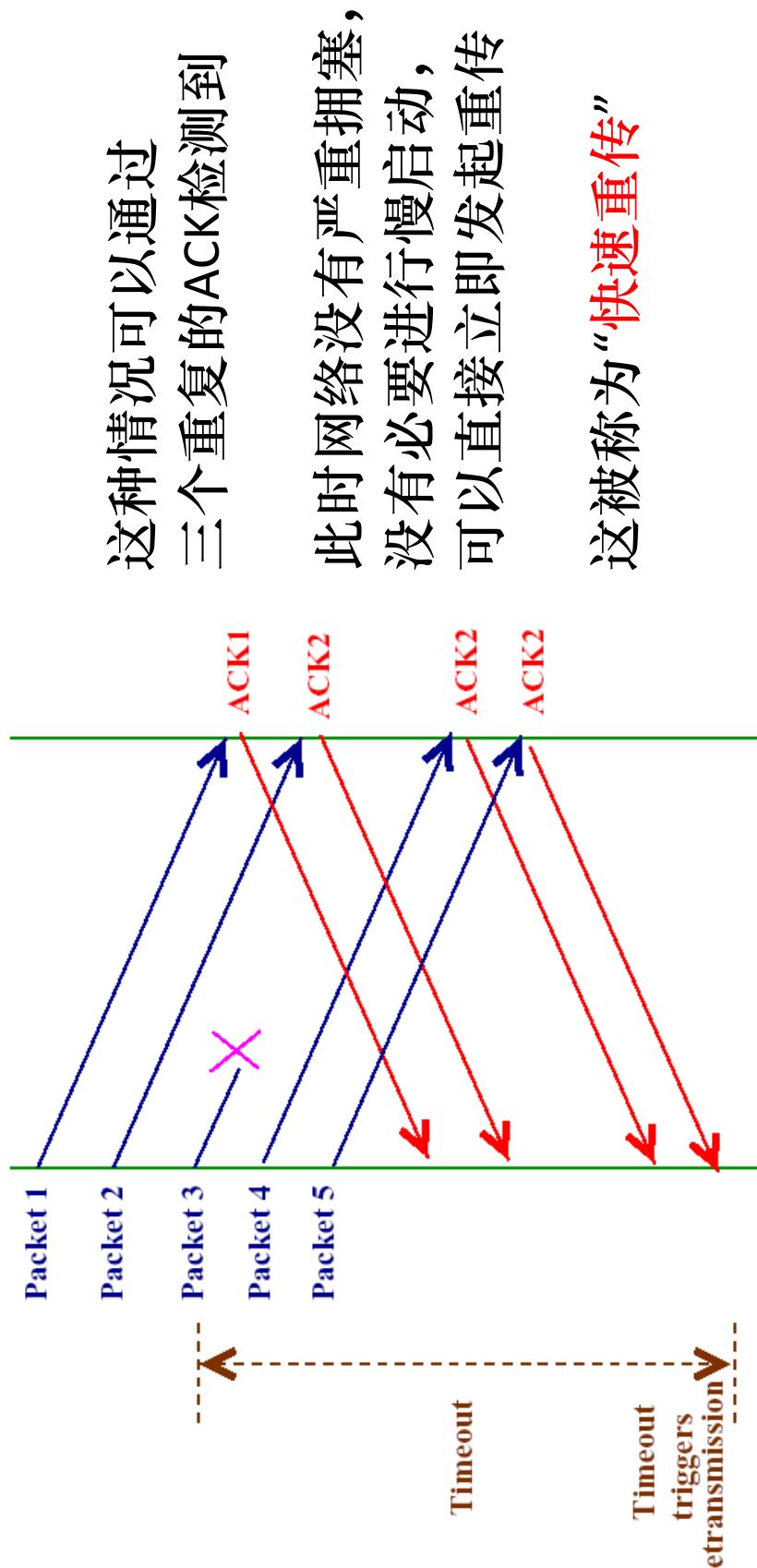
- 分组 n 丢失，但是分组 $n+1, n+2, \dots$ 等到达
如何检测？接收方多次通过ACK表示期望接收分组
何时？收到分组 n 后续分组
收到三个重复的ACK后，发送方快速重传分组
CWND减半，再线性增加

快速重传与快速恢复



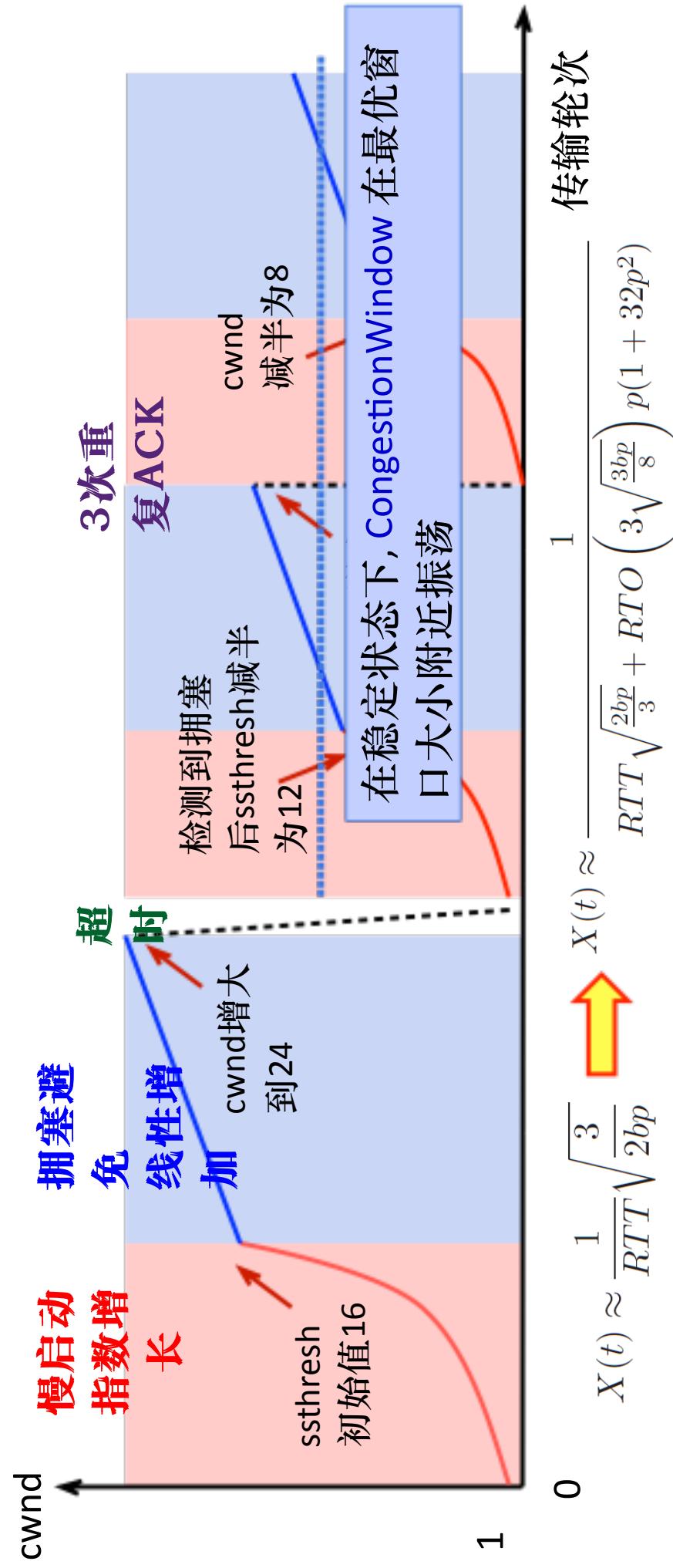
- 如何检测拥塞的发生？传输超时 timeout
- 分组n丢失，超时定时器超时
- 超时后，大的cwnd会导致更严重的分组丢失，因此需要采用慢启动

- 有的情况，拥塞并未发生，而是偶尔丢包



快速重传

- 快速重传的解决方案
- 检测到3次重复ACK后，不必等待超时计时器，立即重传未被确认的报文段

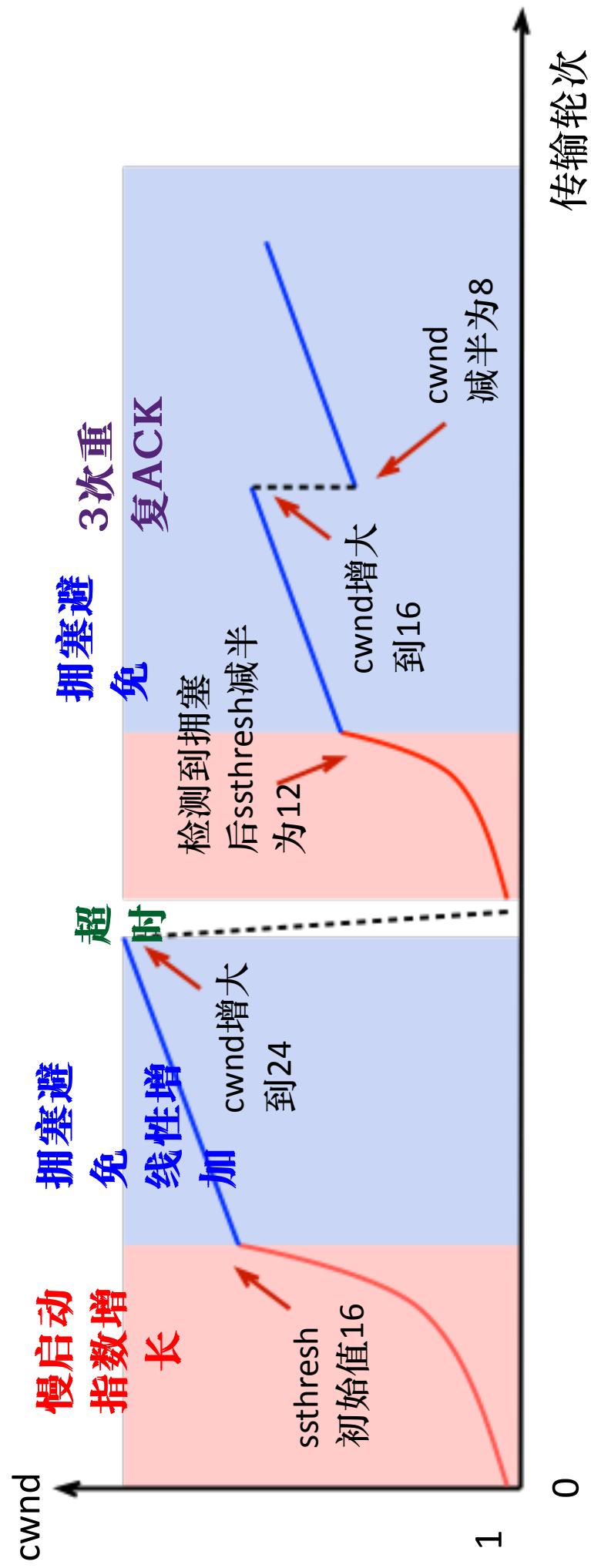


Padhye et al., "Modeling TCP throughput: a simple model and its empirical validation," ACM SIGCOMM 98

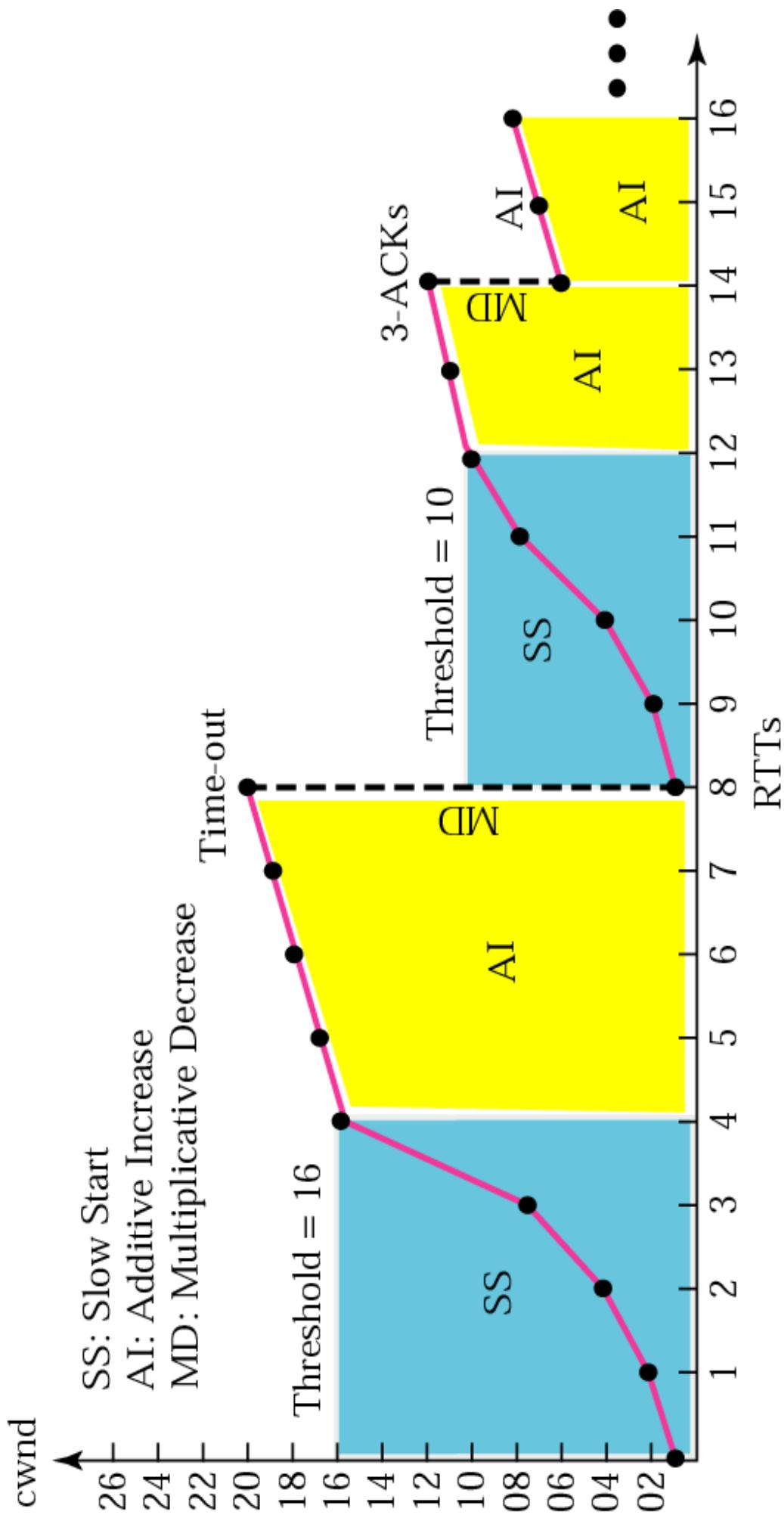
快速恢复



- 基本想法：3次重复ACK时网络不是严重拥塞，不必使用慢启动过程， $ssthresh$ 仍然减半，但直接进入线性增加过程



TCP拥塞控制示例

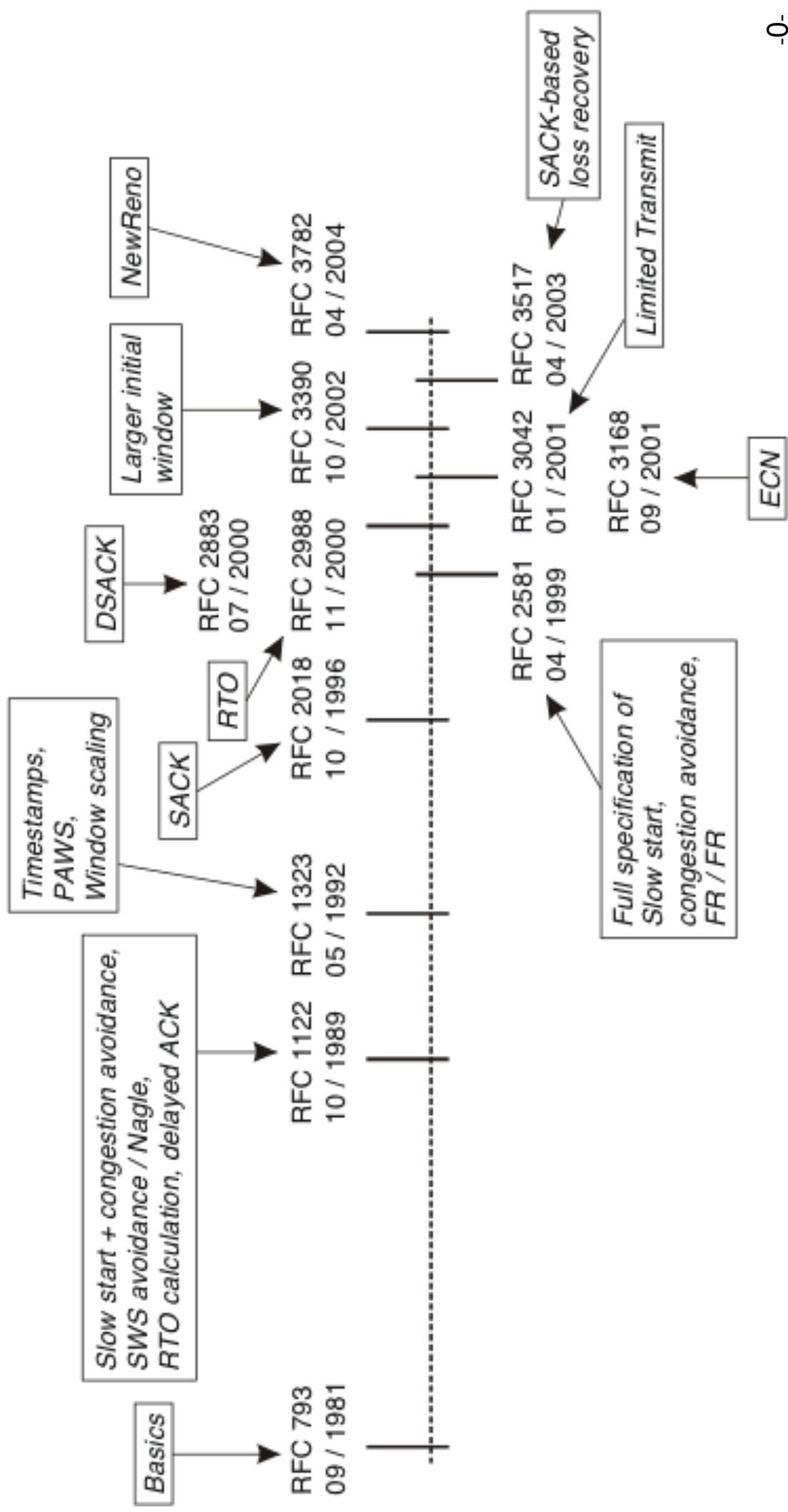


TCP的发展

- RFC 793 (1981)
 - 基于简单滑动窗口的流量控制机制.
- Tahoe (1988)
 - 慢启动, 拥塞避免, 快速重传.
- Reno (1990)
 - 快速恢复.
- 新版 Reno (1995)
 - 对快速重传进行了修订.



TCP 的发展



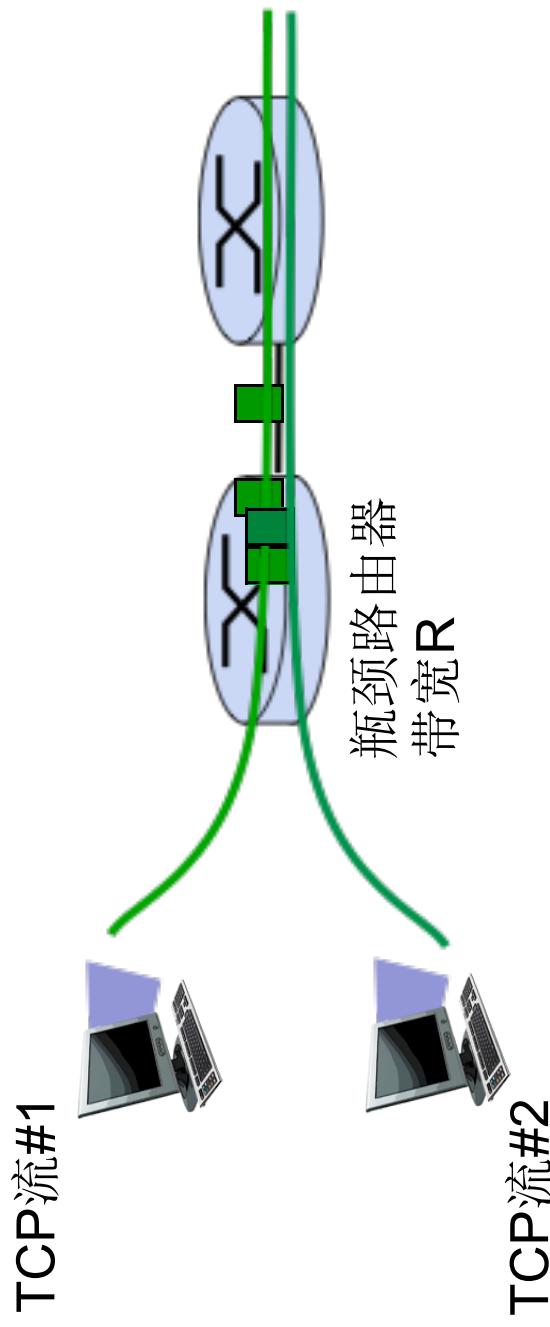
TCP Reno 及扩展



- TCP Reno
- 当今最流行的TCP实现
集成AIMD, 慢启动, 快速重传和快速恢复
- 其他扩展
 - 选择性ACK机制: TCP SACK
 - 显式的拥塞通知: ECN(Explicit Congestion Notification)
 - 基于时延的拥塞避免: TCP Vegas

TCP公平性

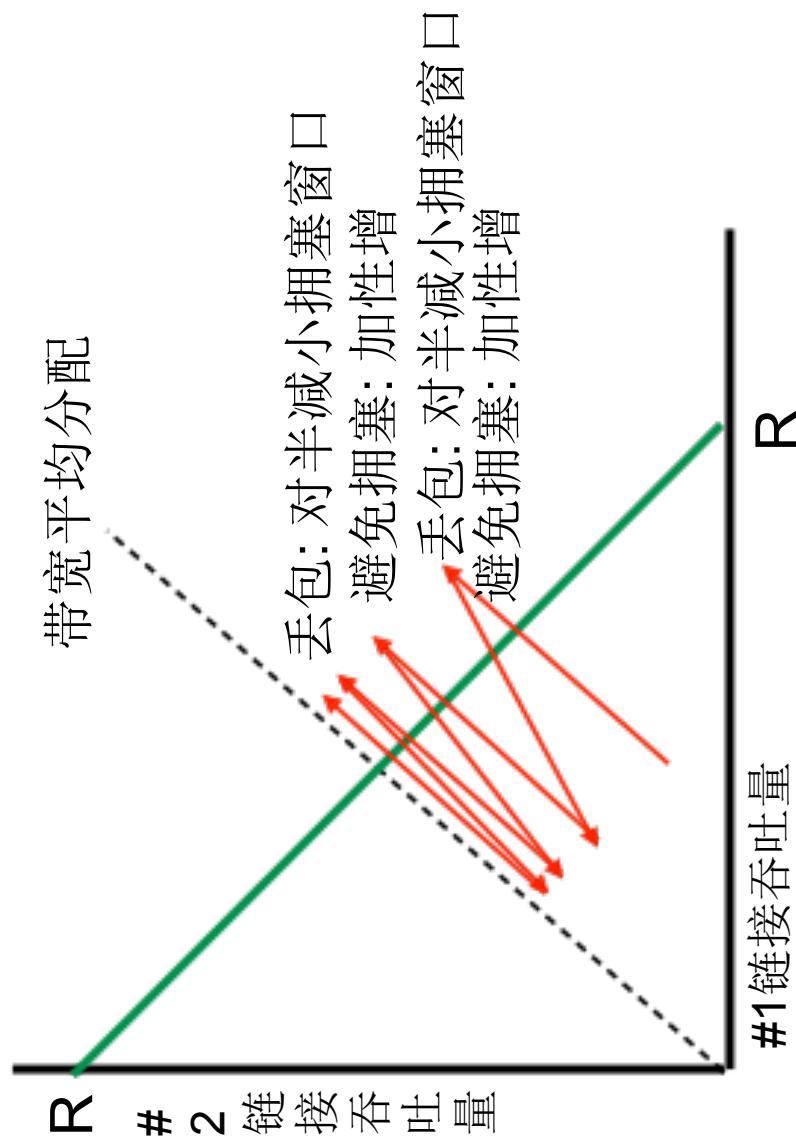
公平目标: 如果K个TCP会话共享瓶颈链路带宽R, 每个TCP流应获得平均吞吐量 R/K



为什么TCP公平?

两个竞争TCP会话：

- 当吞吐量增加，加性增加拥塞控制窗口，带来吞吐量曲线斜率45度乘性减小拥塞控制窗口带来成比例减少吞吐量



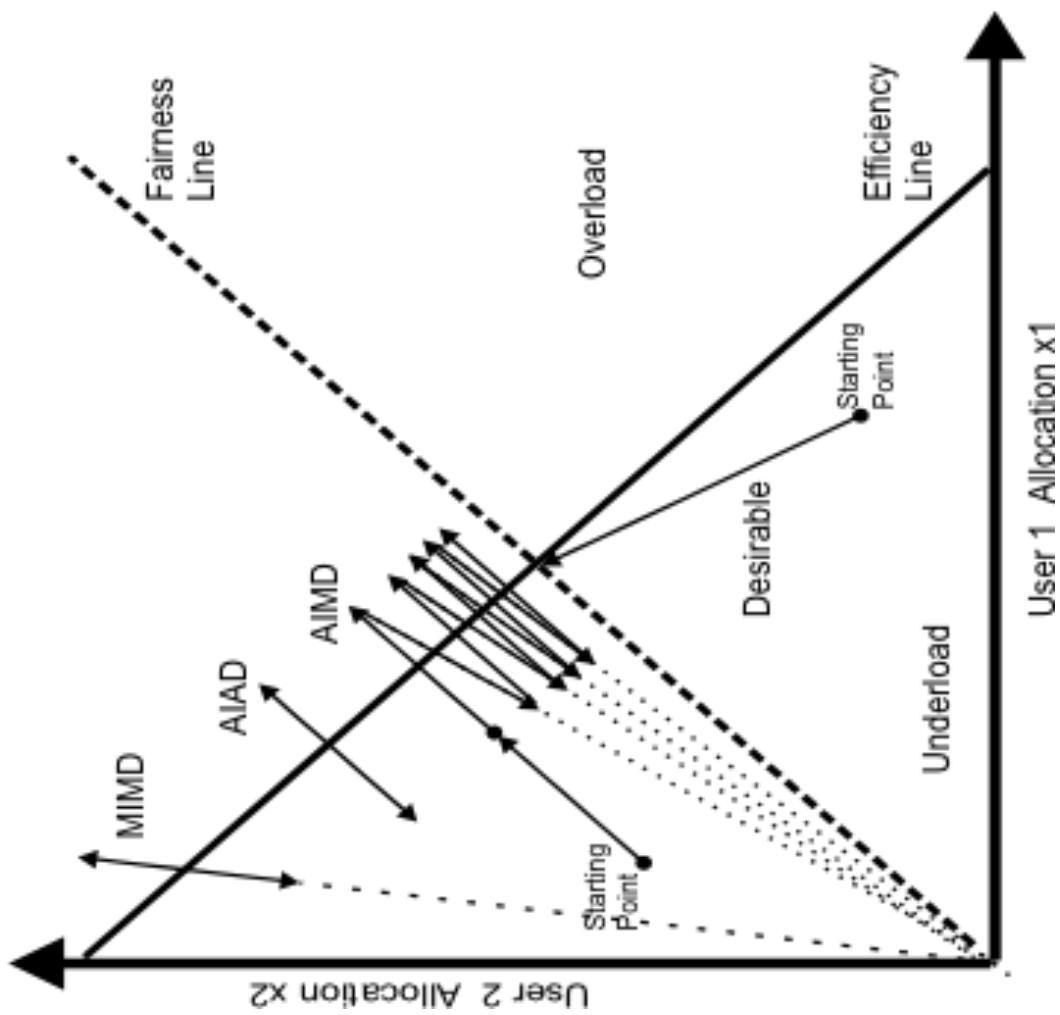
Fairness of AIMD



DM Chiu and R Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks", Computer Networks and ISDN Systems Vol 17, pp 1-14, 1989.



Dah Ming Chiu, Professor
Department of Information Engineering,
cuhk.edu.hk
Associate Editor, IEEE/ACM Transactions on
Networking
IEEE Fellow



讨论：公平性

公平与UDP

- 多媒体应用经常不使用TCP
 - 不希望数据率受限于拥塞窗口
- 使用UDP:
 - 以定常速率传输音频
 - 频流量，容忍丢包

公平与并行TCP链接

- 应用可以建立多条并行TCP链接连接两台主机
- Web浏览器采取此策略
 - 例如，已有10条链接共享链路数据率R：
- 新应用建立1条TCP链接，获得数据率R/11
- 新应用建立10条TCP链接，获得数据率R/2



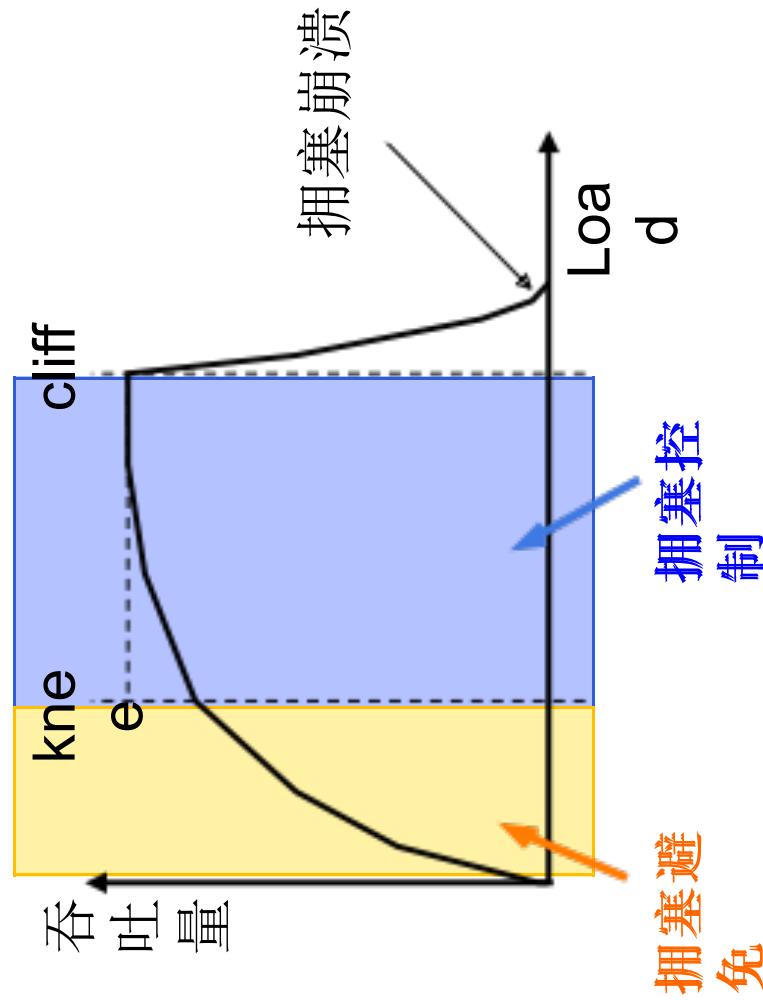
- 引言
- 核心问题: 分配资源
- 资源分配问题
- 排队规则
- TCP 拥塞控制
- 拥塞避免机制
- DECbit
- 随机早期检测(RED)
- 基于源的拥塞避免
- 流量控制
- 服务质量(QoS)
- 总结



拥塞控制和避免



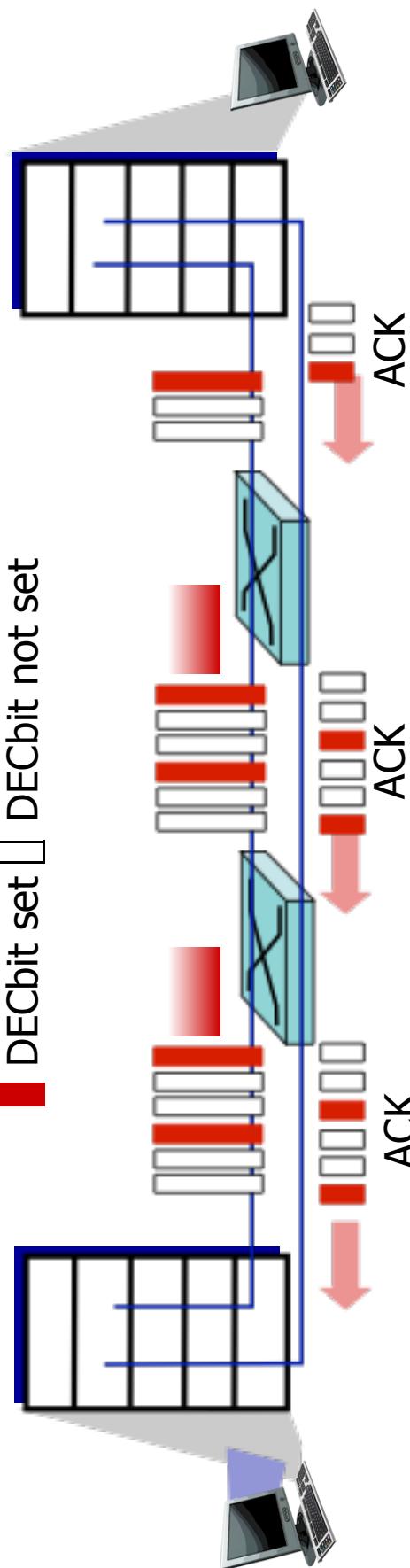
- 增加路由器的缓存无法避免拥塞
- 避免拥塞的方式
 - 源端减少负载 - 短期流量工程 - 中长期
 - 不断增加链路的容量 - 长期



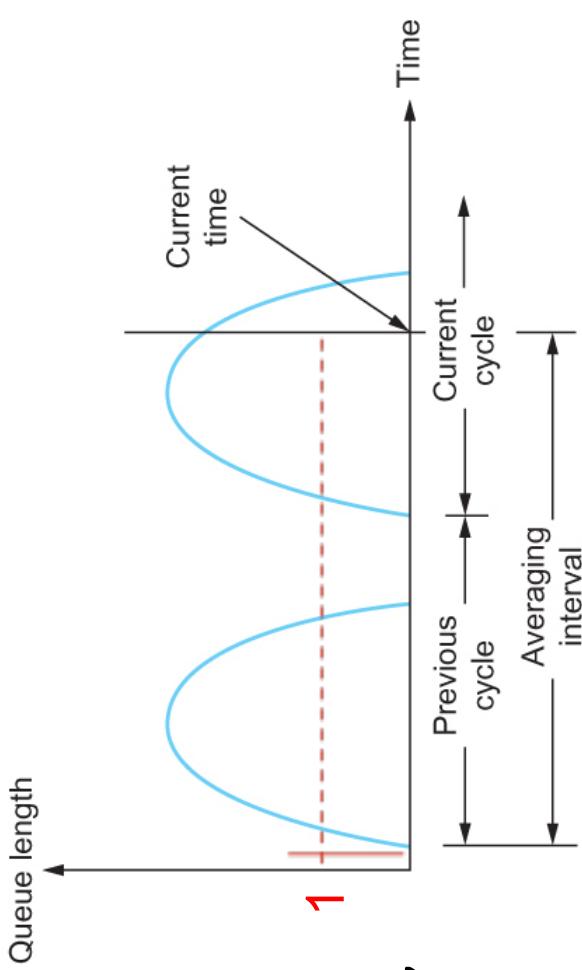
拥塞避免: DECbit



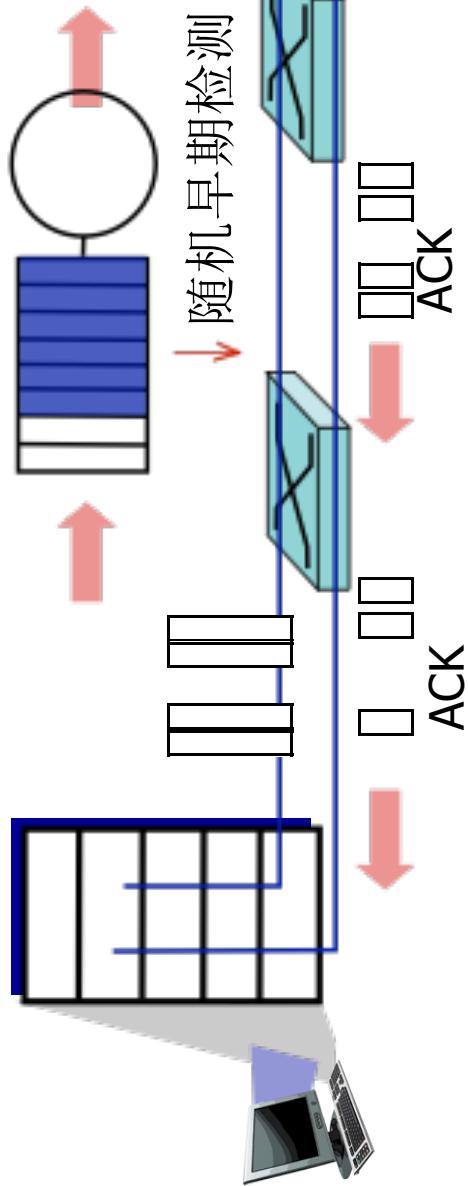
DECbit set DECbit not set



DECbit 用于数字网络体系结构(DNA),
用面向连接的传输层协议的一个无连接
网络。路由器监视当前负载, 在拥塞将发生时
通过设置流经分组的DECbit通知节点
目标主机将DECbit复制到传回源端的ACK
源主机根据收到的设置了DECbit位的ACK,
控制发送窗口大小。

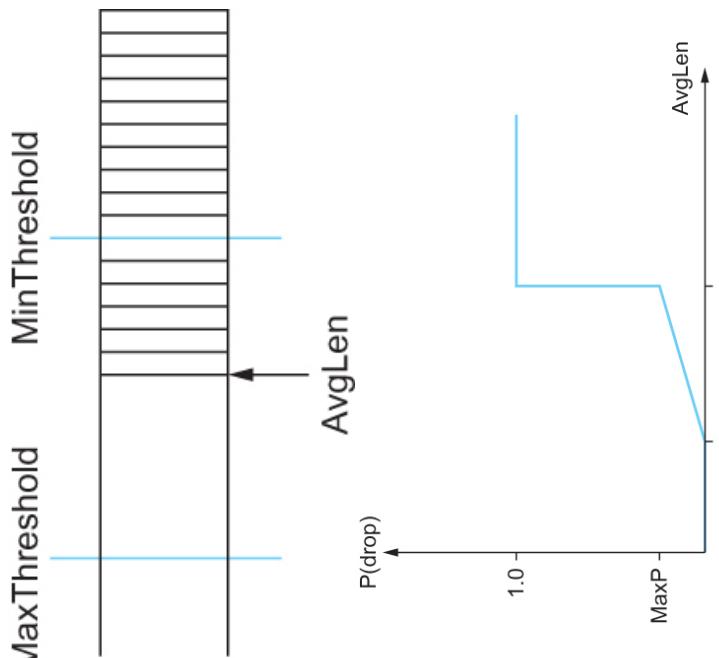


拥塞避免: RED



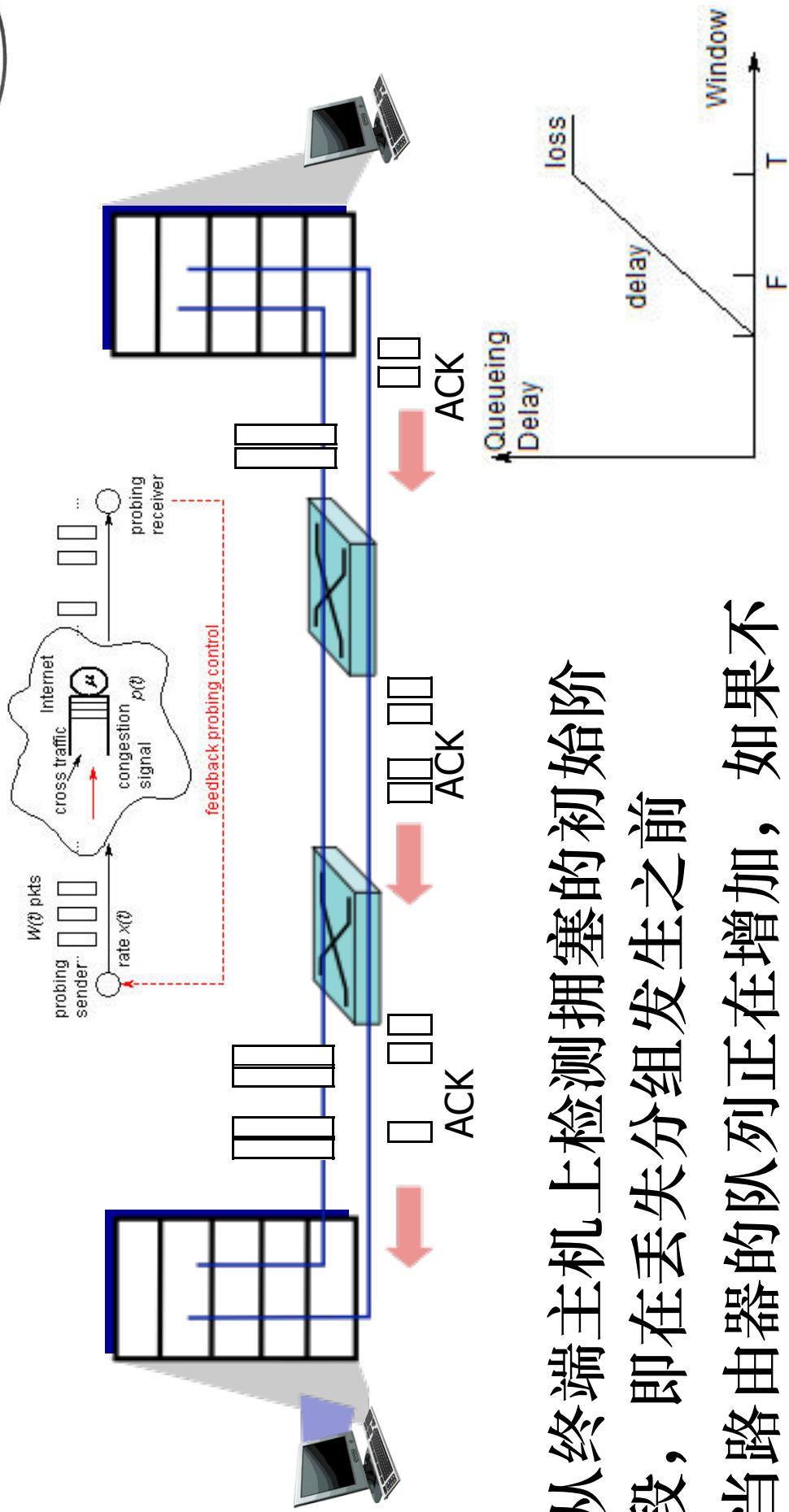
随机早期检测
路由器监视平均队列长度，当检测到拥塞即将发生时，将发送通知消息，即拥塞窗口塞

RED通过丢包隐式的通知源发生拥塞



RED应用早期随机丢弃分组
RED的参数设置影响系统性能

拥塞避免：基于源



- 从终端主机上检测拥塞的初始阶段，即在丢失分组发生之前
- 当路由器的队列正在增加，如果不采取措施很快就会发生拥塞
 - e.g. $(\text{CurrentWindow} - \text{OldWindow}) \times (\text{CurrentRTT} - \text{OldRTT})$

高速TCP协议类型

- 基于丢包的拥塞避免 (LCA):
 - HS-TCP [Flo03], S-TCP [Kel03], BIC-TCP [XHR04], CUBIC-TCP [RX05], H-TCP [SL04], and E-TCP (for small buffer) [GTHZ07]
- 基于时延的拥塞避免 (DCA):
 - FAST TCP [JWL04]
 - 混合式拥塞避免 (HCA):
 - TCP-Africa [KBR05], Compound-TCP [TSZS06]
- DCA 和 HCA 协议的基本思想是基于 RTT 的变化来检测早期拥塞，但可能存在下面的不利因素
 - 反向路径的排队时延影响
 - 由于路由变化带来的传播时延变化
 - 由于数据链路层带来的时延变化: Automatic Repeat reQuest (ARQ)

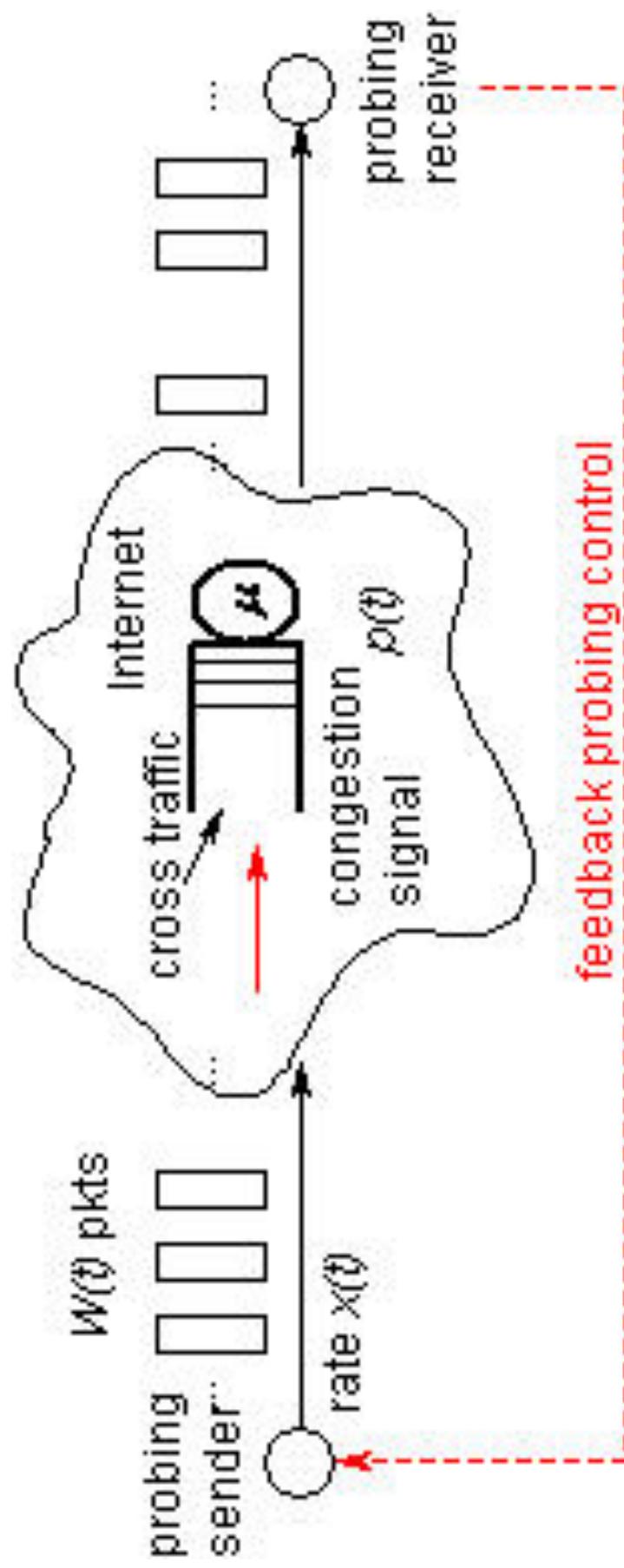




针对无线随机丢包的TCP协议变型

- TCP Westwood [CGM+02, GM02]
 - 基于带宽估计改进TCP拥塞控制
 - 带宽估计方法：带宽估计(BE), 速率估计(RE), 带宽/速率联合估计(CRB)
- TCP Veno [FL03]: Vegas [BOP94] + Reno [Jac90]

总结：端到端拥塞控制//避免



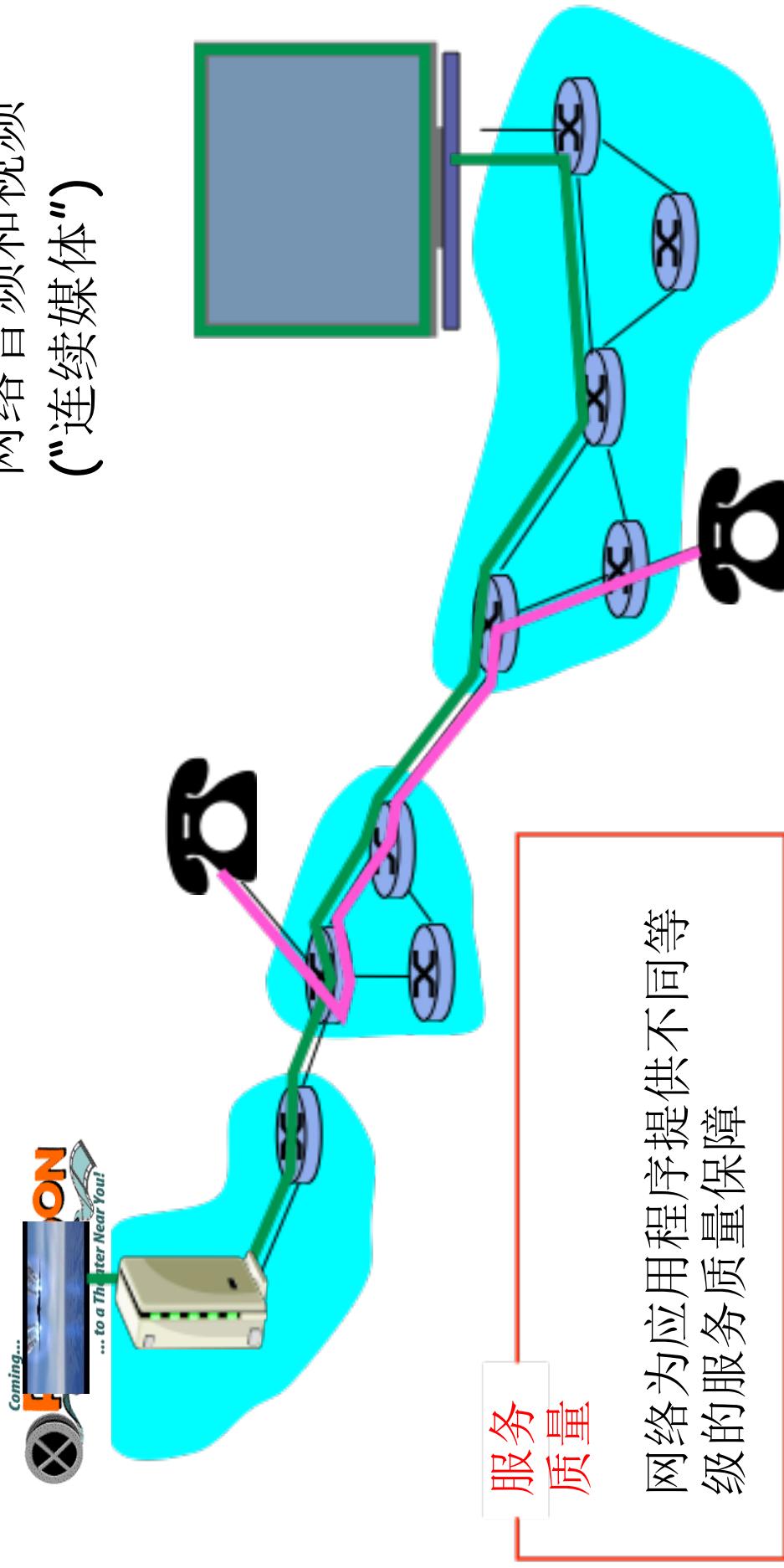
- 引言
- 核心问题: 分配资源
- 资源分配问题
- 排队规则
- TCP 拥塞控制
- 拥塞避免机制
- 服务质量(QoS)
- 服务需求
- 应用需求
- 端到端自适应
- 综合服务(RSVP)
- 区分服务(EF、AF)
- 基于等式的拥塞服务
- 总结



多媒体应用和服务质量



多媒体应用：
网络音频和视频
("连续媒体")



多媒体网络应用

多媒体应用的分类：

- 1) 流媒体存储式音频和视频
- 2) 直播流式音频和视频
- 3) 实时交互音频和视频

基本特征：

时延敏感：

- 端到端的延时
- 延时抖动

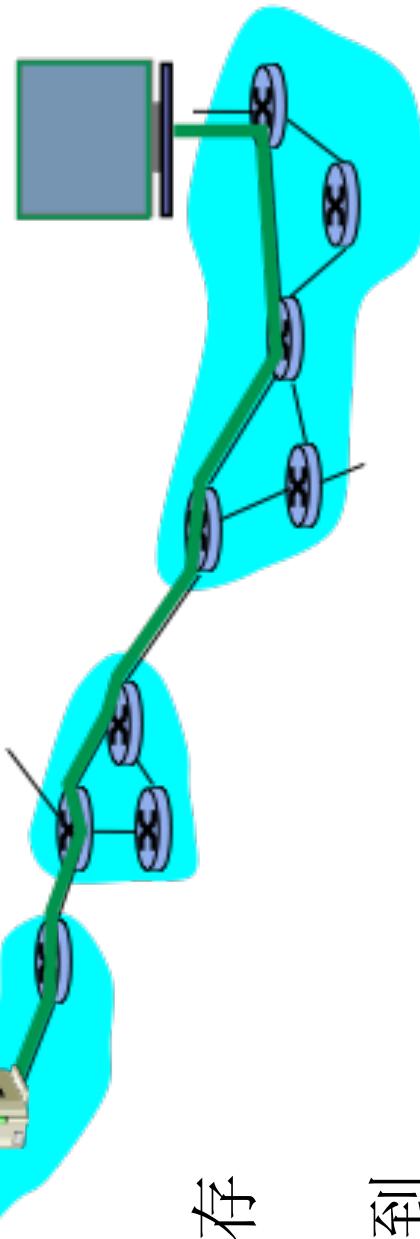
容忍丢包：
少量数据丢失
只会在音视频信号
少

数据传输应用刚好相反：
对延时不敏感，
完整性是首要需求

时延抖动是指 同一个数
据流中分组时延的变化



点播流媒体音频和视频



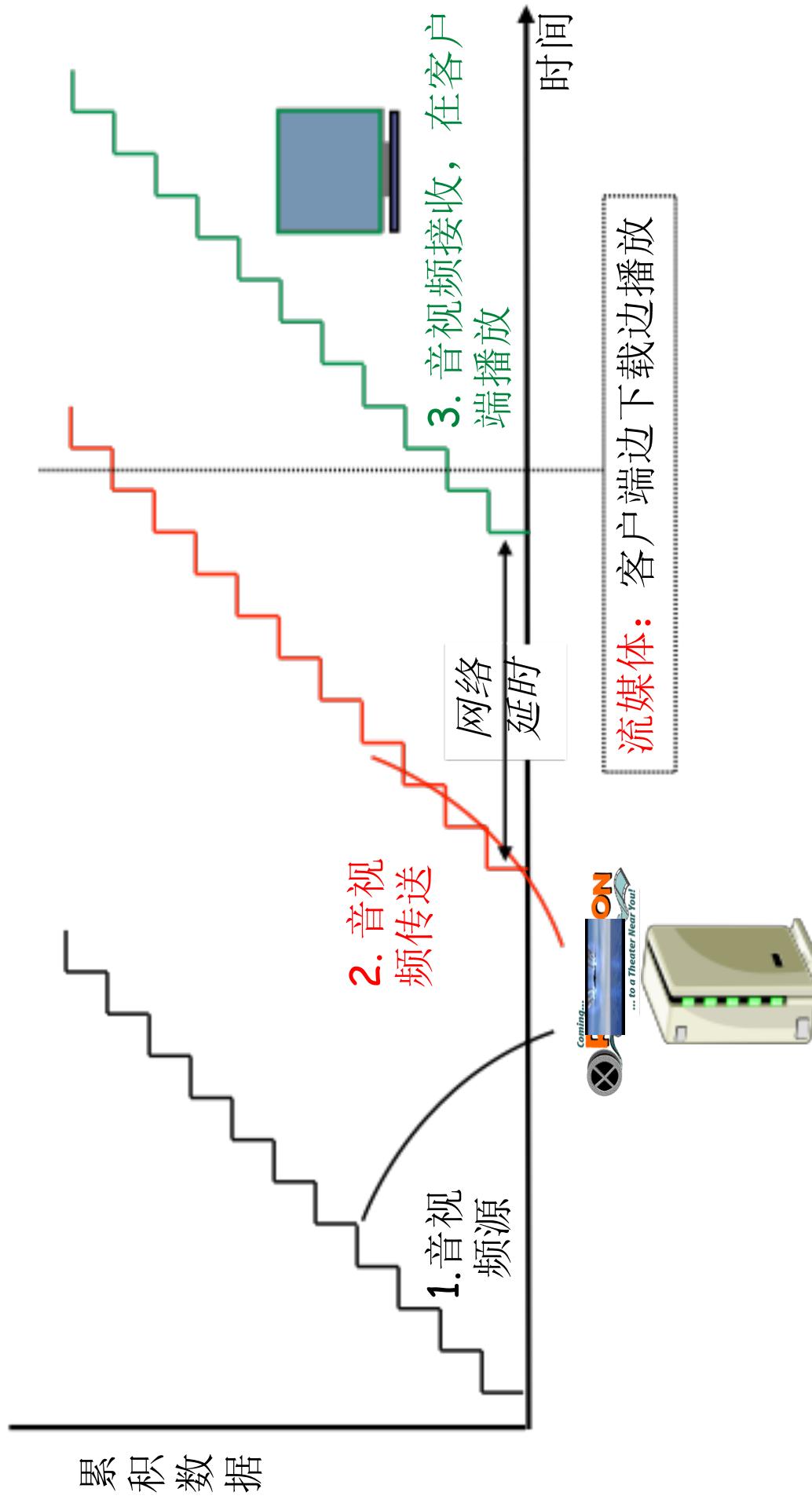
存储媒体：

- 多媒体内容预先录制存储在服务器上

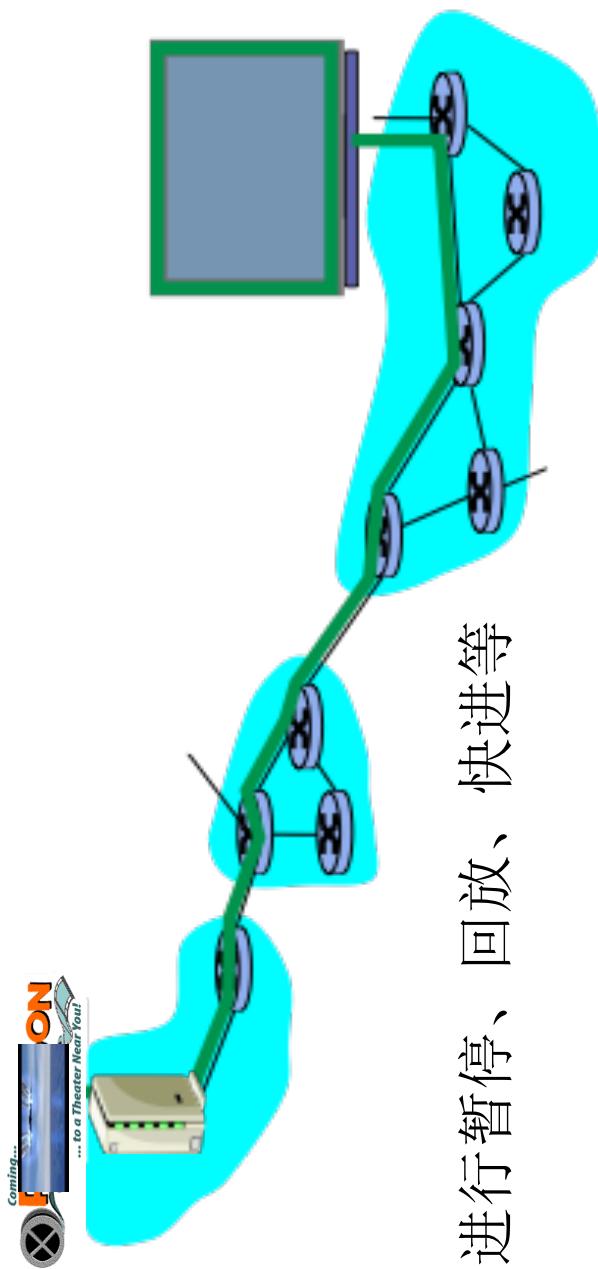
- **流媒体：**在所有数据到来之前客户即可开始播放

- **连续播放：**具有时限性，需满足及时播放的需求

存储式流媒体音频和视频



存储式流媒体音频和视频：交互属性



- **VCR功能：**客户可以进行暂停、回放、快进等操作
- 10 秒的起始时延
- 1-2 秒的操作响应时延
- **连续播放：**具有时间限制，需满足及时播放的需求

直播流媒体音频和视频



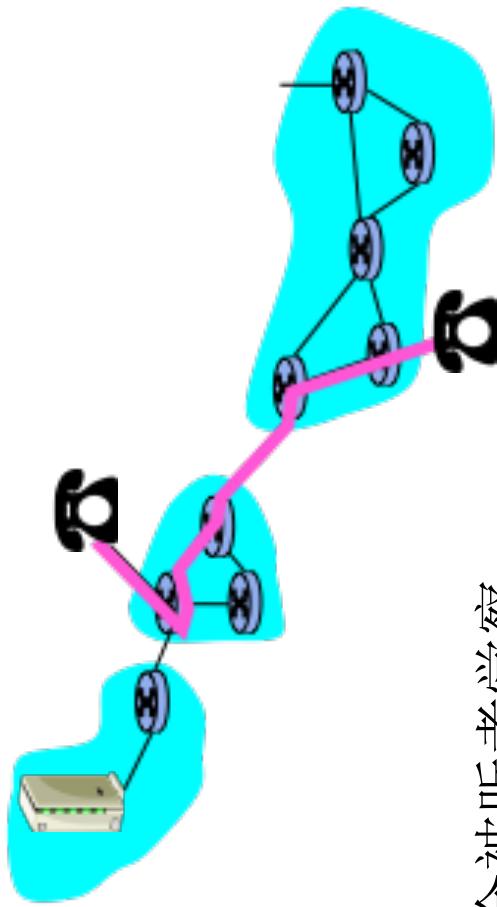
例子：

- 互联网上的访谈节目
- 赛事直播
- **直播流媒体**（与存储式流媒体中介绍的一样）
 - 播放缓存
 - 播放相对于传输可滞后数十秒
 - 仍然具有时间限制，需满足连续播放
- **交互性**
 - 不能快进
 - 有回退和暂停等操作

实时交互式音频和视频



- **应用：**IP 电话，视频会议，分布式交互世界



端到端的延时需求：

- 语音：小于 150 毫秒的时延不会被听者觉察到，150-400 毫秒的时延能够被接受，包括应用层（封装）和网络延时
- 较高的延时会影响互动性
- 会话初始化
- 被叫者如何通告其 IP 地址、端口号和编码算法？

- 引言
- 核心问题: 分配资源
- 资源分配问题
- 排队规则
- TCP 拥塞控制
- 拥塞避免机制
- 服务质量(QoS)
- 服务需求
- 应用需求
- 端到端自适应
- 综合服务(RSVP)
- 区分服务(EF、AF)
- 基于等式的拥塞服务
- 总结





目前互联网中的多媒体应用障碍

TCP/UDP/IP: “尽力交付的服务”

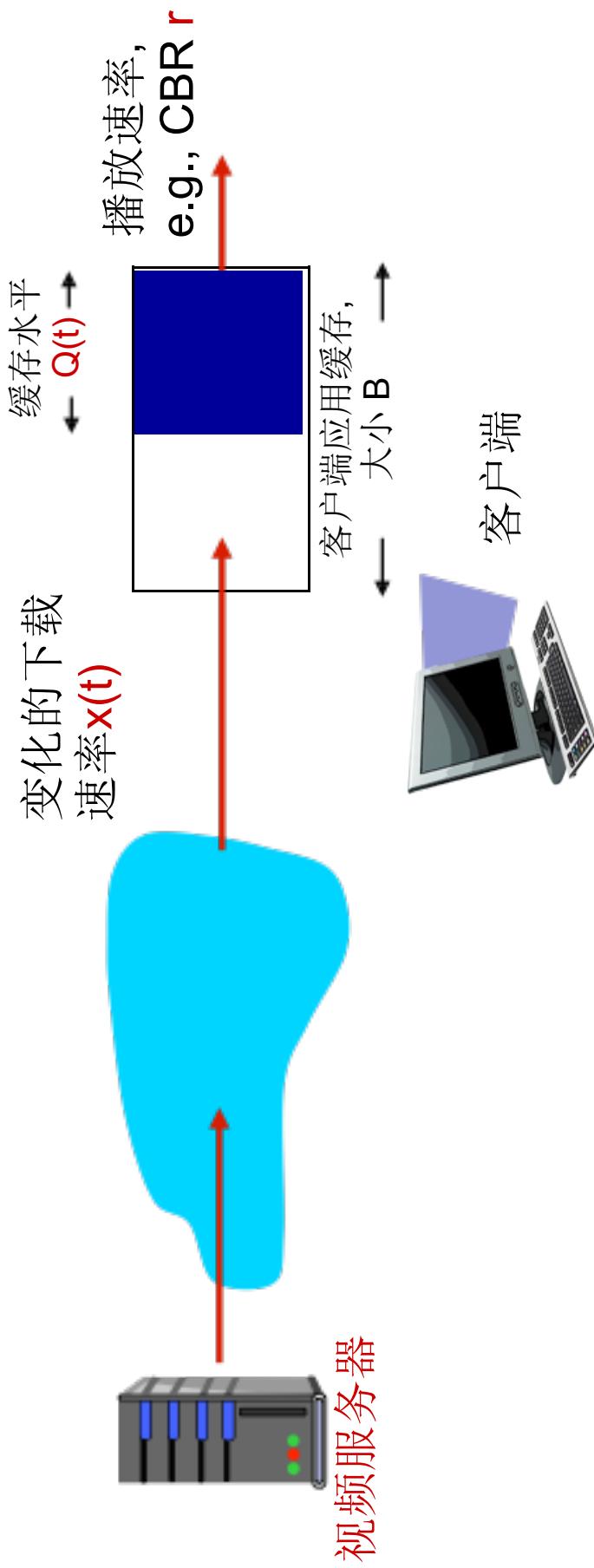
- 没有时延和丢包性能保障

？ 但是：多媒体应用需要有效的
QoS 保障和不同性能等级的服
务

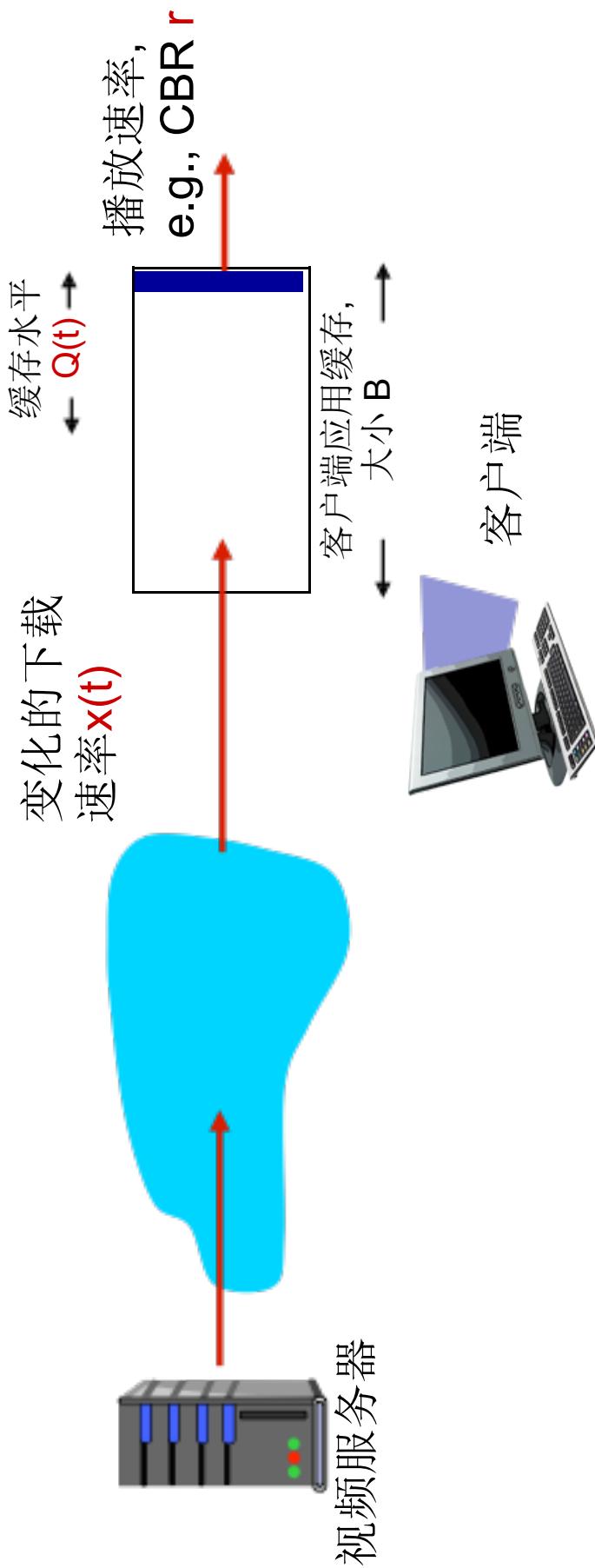


互联网多媒体应用往往需要利用应用层技术来尽
量补偿底层网络的延时和丢包影响。

客户端缓存及播放

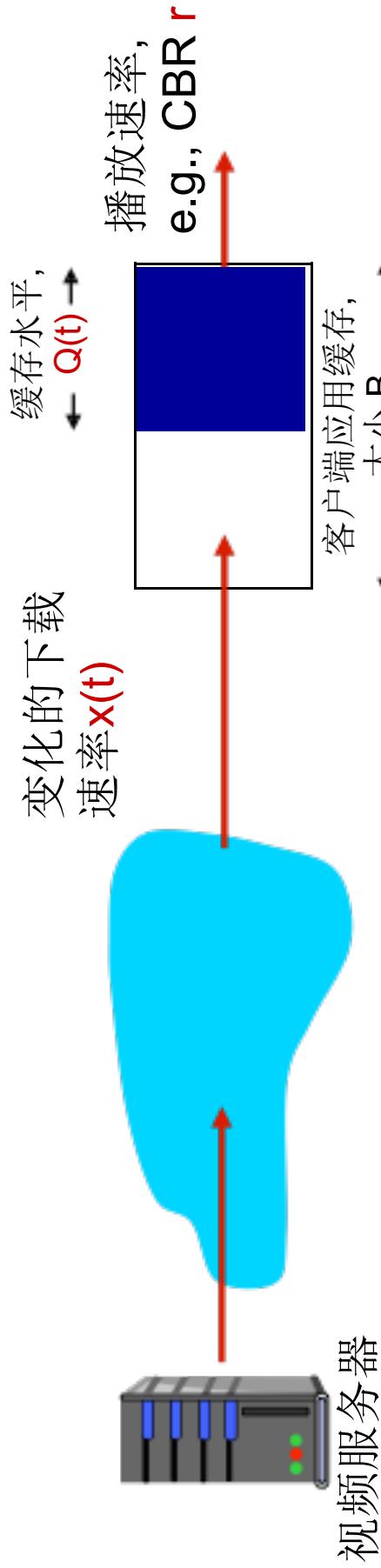


客户端缓存及播放



- 首先缓存流量，直到tp开始播放
- tp时刻开始播放
- 因为下载速率随时间变化，而视频播放速率是常数，缓存水平不断变化

客户端缓存及播放



播放缓存: 平均下载速率 (\bar{x}), 播放速率 (r):

- $\bar{x} < r$: 缓存最终变空 (停止视频播放, 直到缓存下载到新的视频内容)
- $\bar{x} > r$: 缓存不会为空, 只要最初的缓存时延足够大, 可以抵消下载速率的变化
- **平衡初始的缓存时延:** 更大的缓存时延会减少缓存为观看视频空的可能性, 但会造成用户等待更长的时间才能开始观看

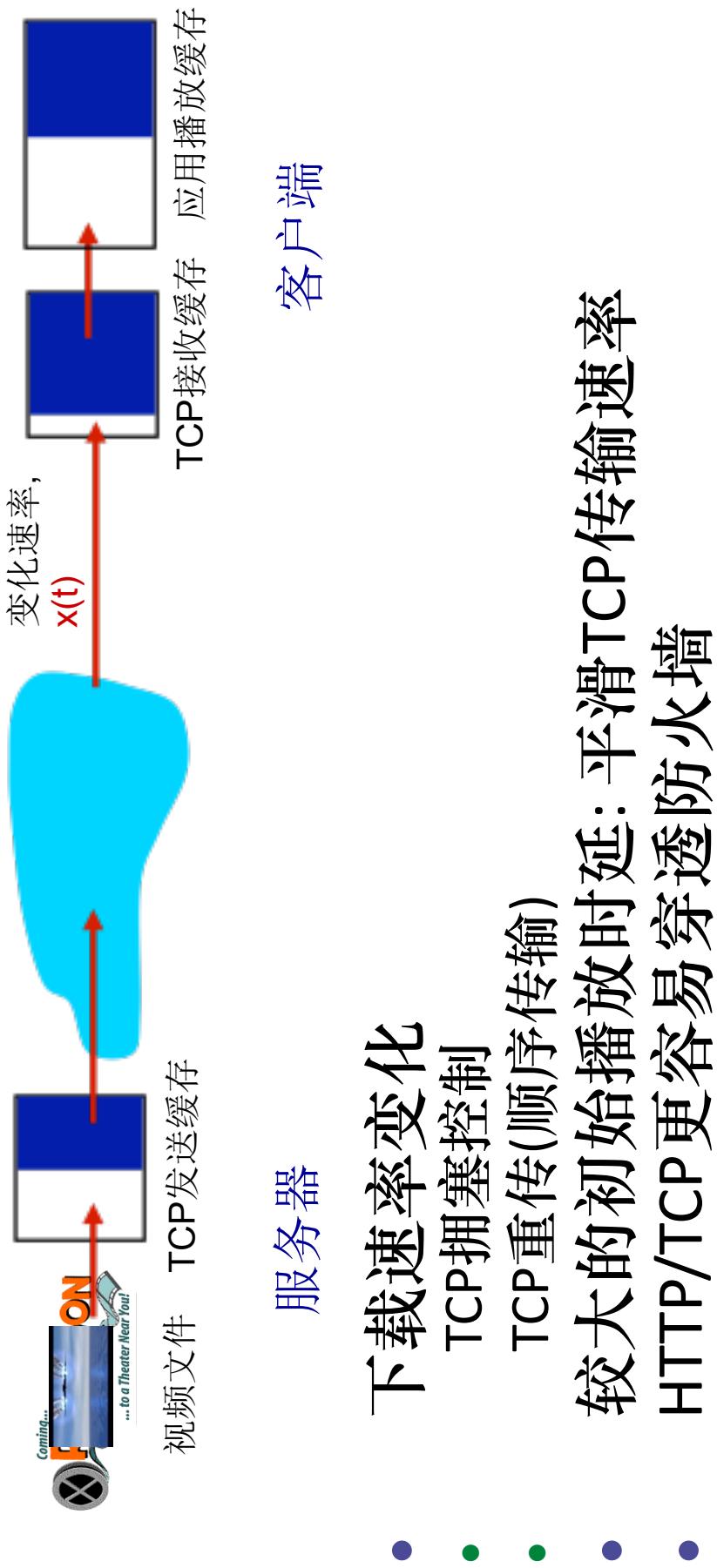
流媒体: UDP



- 服务器以合适的速度向客户端发送流量
- 一般而言: 发送速率 = 编码速率率 = 常数速率
- 传输速率可以响应网络拥塞水平
- 引入较短的播放延迟(2-5 seconds), 补偿网络时延变化的影响
- 错误恢复:
 - 在应用层实现
 - 根据时延允许的情况, 可自适应调整
- RTP [RFC 2326]: 定义不同的多媒体载荷类别
- UDP可能无法穿透防火墙

流媒体: HTTP

- 传输多媒体文件: HTTP GET
- 以TCP的最大可能速率传输



流媒体: DASH



- **DASH: Dynamic Adaptive Streaming over HTTP**
- **服务器:**
 - 视频文件切割为多个数据块
 - 每个数据库以不同的速率编码及存储
 - **说明文件:** 不同数据块有对应的URLs
- **客户端:**
 - 周期性测量服务器到客户端的带宽
 - 参考说明文件, 一次请求一个数据块
 - 根据当前的网速限制, 选择最大的编码速率
 - 基于当时的可用带宽, 不同时间可选择不同编码速率

流媒体: DASH



- *DASH: Dynamic, Adaptive Streaming over HTTP*
- 智能客户端: 客户端决定什么时候请求数据块 (避免缓存为空或者用尽)
- 请求什么编码速率 (当可用带宽更大, 选择更高的视频流)
- 从哪里下载数据块 (可以从距离客户端更近或者可用带宽更多的服务器下载)

- 引言
- 核心问题: 分配资源
- 资源分配问题
- 排队规则
- TCP 拥塞控制
- 拥塞避免机制
- 服务质量(QoS)
- 服务需求
- 应用需求
- 端到端自适应
- 综合服务(RSVP)
- 区分服务(EF、AF)
- 基于等式的拥塞服务
- 总结



互联网架构应如何演化才能更好地支持多媒体？



综合服务(IntServ):

- 对互联网做根本性的变革，能够明确地端到端的性能保证
- 在主机和路由器都需要新的、复杂的软件及新的服务类型

区分服务(DiffServ):

- 只需对互联网基础设施做少量数目有限的服务。

自由放任的方法:

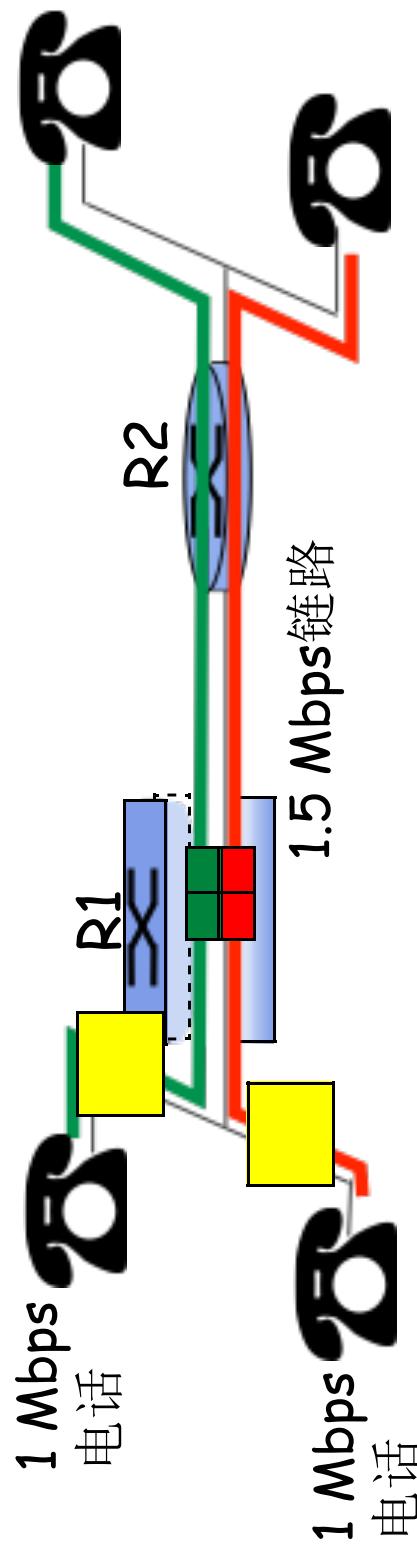
- 不必对支撑互联网的协议做任何根本的改变
- 按需分配带宽
- 内容分发网络，多播覆盖网络



你对此有何看法？

QoS 保证原则

- 基本原则：流量需求不能超过链路容量



原则

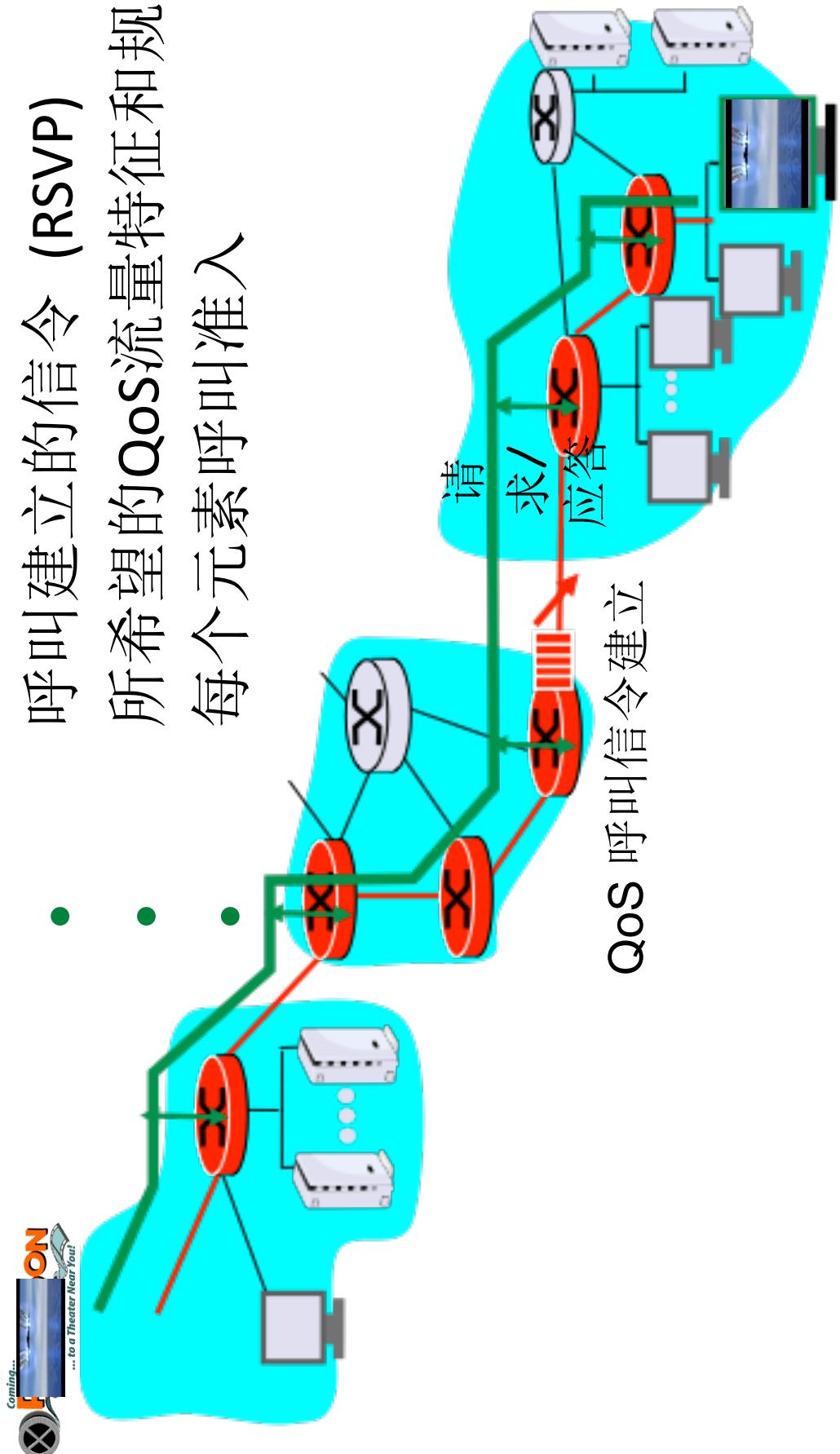
如果网络没有充足的资源，但要保证应用的QoS，则需要一个呼叫准入过程，根据流声明的QoS需求，要么网络接受服务请求，要么拒绝服务请求。



QoS 保证

资源预留

- 呼叫建立的信令 (RSVP)
- 所希望的QoS流量特征和规范
- 每个元素呼叫准入



IETF综合服务(IntServ)



- 用于在互联网中为各个应用会话提供个性化的QoS保证
- 资源预留：路由器需要维护资源分配的状态信息和QoS需求
- 准许/拒绝新的呼叫请求：

问题:新到的流在不影响已经建立会话的服务质量前提下能否获得性能保证？

呼叫接纳

到达的会话必须：

- 声明QoS需求
- **R-spec:** 定义一个呼叫要求的具体QoS向网络发送的流量特性
- **T-spec:** 定义流量特性
- 信令协议：将 R-spec和T-spec提交到路由器进行资源预留
- **RSVP**



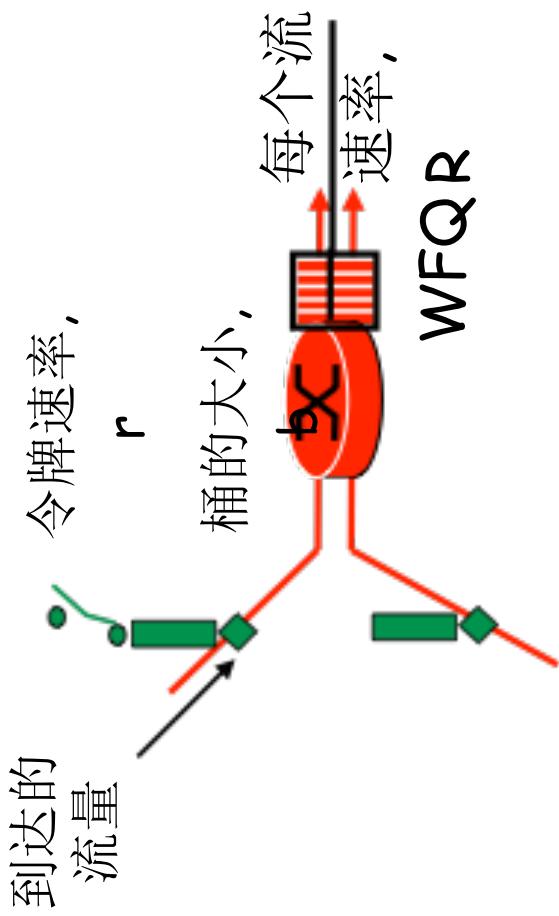
综合服务 (IntServ)[RFC2211, RFC2212]

有服务保证的服务：

- 为分组提供了在路由器中的严格队列监管机制。经受的排队时延的界限。使用漏桶机制来刻画流量特性。

受控的负载服务：

- 一个呼叫将接收与相同流在无负载网络中获得的QoS非常接近的服务质量



$$\frac{D_m}{D_l} =$$

互联网中的信令



通过无连接的IP + 尽力而为服务 = 初始网络协议设计时无信令协议
路由器转发分组

- 新需求：沿着端到端的路径预留资源以向多媒体应用提供QoS保证
- RSVP：资源预留协议 [RFC 2205]
- 允许用户以健壮、高效的方式与网络交互需求(信令)
- 早期的互联网信令协议：ST-II [RFC 1819]

RSVP设计目标

1. 数据的接收方发起并维护用于该流的资源预留
2. 预留不同的资源以适应不同的应用
3. 提供多播树中的带宽预留，以适应多播组成员
4. 利用现有多播/单播路由器，以适应单播/多播路由器的改变
5. 控制协议开销随着接收方数量的增加线性增长
6. 模块化设计以支持不同的基础技术





RSVP所不能做的：

- 没有定义网络向数据流提供预约带宽的方法。
- 只是一个允许应用预约必要链路带宽的协议
- 不指定分组经过哪些路由器
- 这是路由协议的工作
- 信令与路由器分离
- 不参与转发数据分组
- 分离控制(信令)和数据(转发)传输

RSVP: 运行概述

- **发送方、接收方加入到多播组**
- 在RSVP之外完成
- 发送方无须加入组
- **发送方到网络的信令**
 - 链路消息: 使路由器知道发送方的存在
 - 链路撤销: 在服务器器中删除发送方的链路状态
- **接收方到网络的信令**
 - 预留消息: 预留从发送方到接收方的资源
 - 预留撤销: 移除对接收方的资源预留
- **网络到终端系统的信令**
 - 链路错误
 - 预留错误



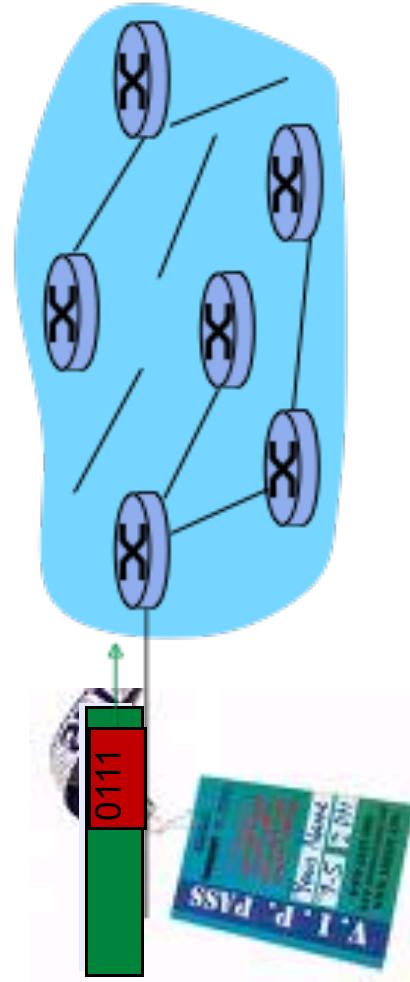
- 引言
- 核心问题: 分配资源
- 资源分配问题
- 排队规则
- TCP 拥塞控制
- 拥塞避免机制
- 服务质量(QoS)
- 服务需求
- 应用需求
- 端到端自适应
- 综合服务(RSVP)
- 区分服务(EF、AF)
- 基于等式的拥塞服务
- 总结



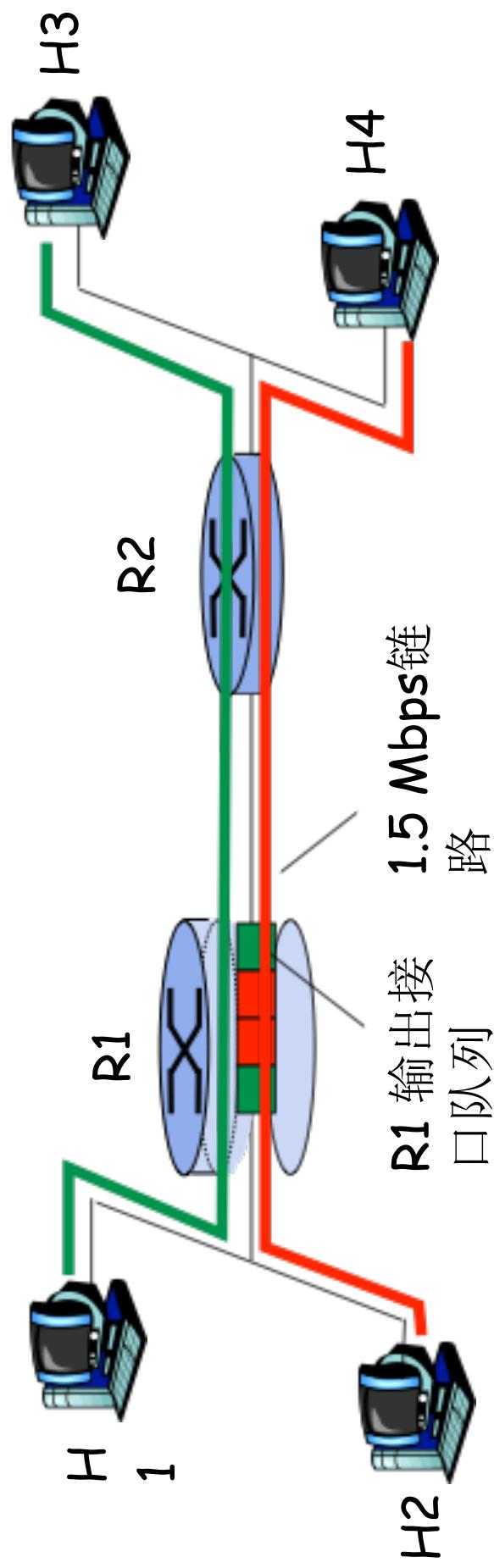
提供多个服务等级



- 当今互联网对所有应用提供尽力而为服务。
- 应用将得到网络能够提供的任何等级的性能
- 强化服务模型：提供多个等级的服务
 - 将流量分为多个类别
 - 网络为不同的流量提供不同等级的服务。（类似于VIP服务与常规服务）
 - 聚合流量：同一种流量类别在网络中得到相同的对待，独立于特定的它们所属的端到端连接。
- IPv4首部中的服务类型字段(ToS)
 -

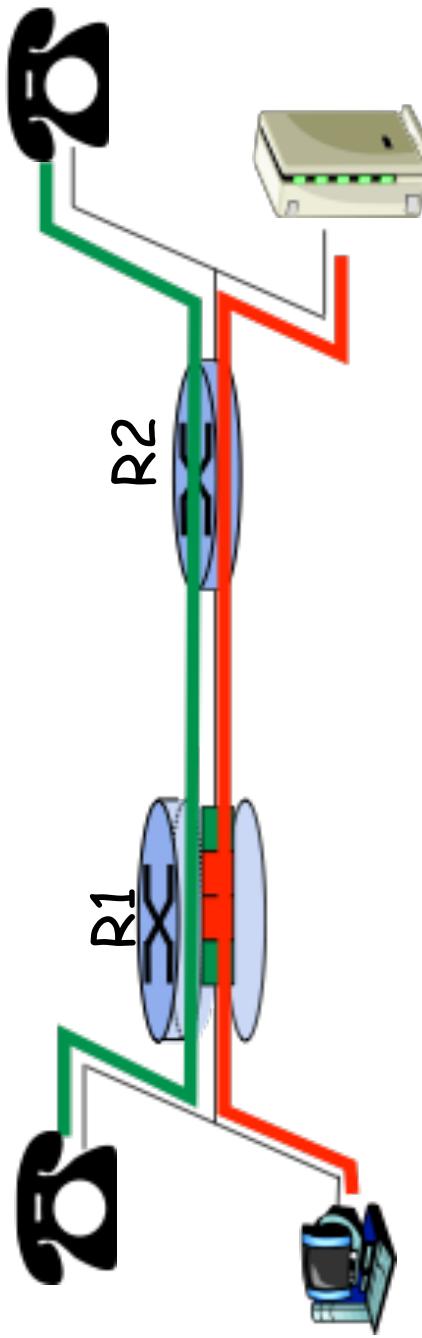


研究运动机：场景



情况 1：音频应用和FTP传输

- 例子：1Mbps IP 电话, FTP 分享 1.5 Mbps 链路。
- FTP源产生的突发分组可能潜在地填充队列导致音频分组延迟或丢失
- 为音频分组分配已严格的规定优先级

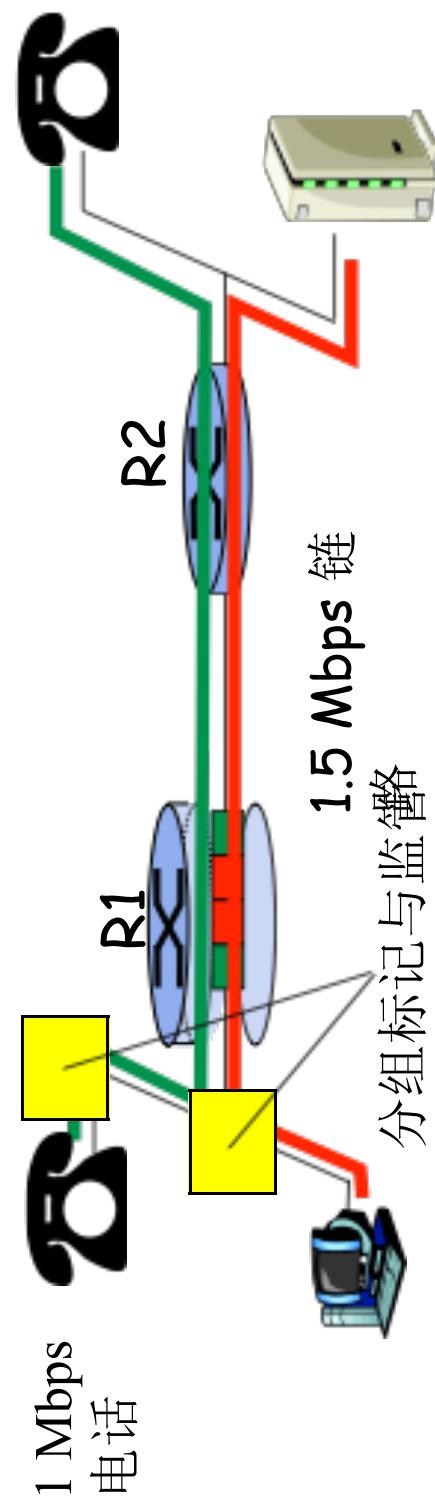


原则2
分组分类使得路由器可以区分属于不同类型流量的分组

QoS 保证原则



- 如果应用程序发生异常怎么办(如音频以比声明速率更高的速率发送)
- 监管: 强迫源按照一定的标准发送流量
- 在网络边缘进行标记与监管:
- 类似于ATM UNI(用户网络接口)

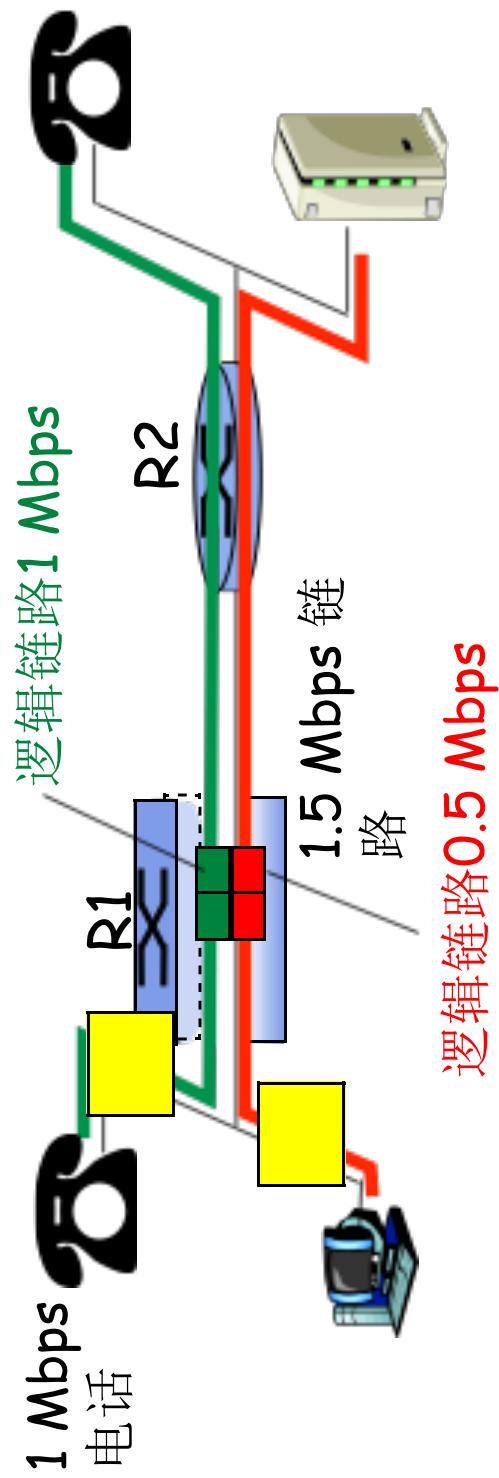


原则3

在不同数据流之间提供保护或隔离，使一类流量不会受到另一类异常流量的负面影响

QoS 保证原则

- 给每个应用流分配固定量的链路带宽：一个流量类型不能利用其他应用现在不使用的带宽，低



原则4

当为不同数据流提供隔离时，希望尽可能有效地利用资源



监管机制



目标：调节流向网络的分组速率

三种重要的监管准则：

- **平均速率：**限制一个流的分组能够发送到网络中的长期平均速率

- **关键问题：**监管的时间间隔。每秒100个分组比每分钟6000个分组约束更严格

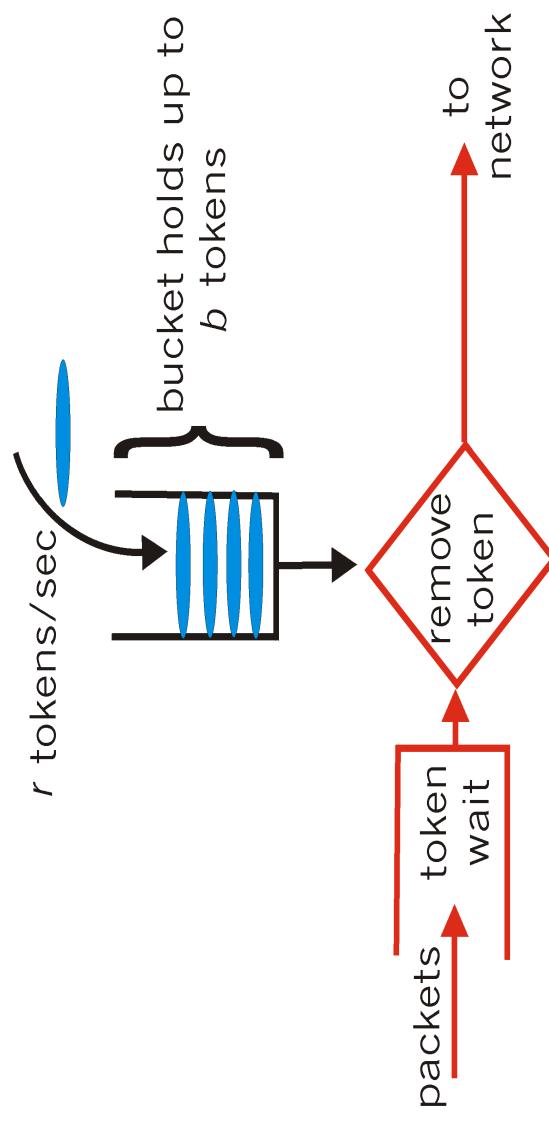
- **峰值速率：**限制在一个较短的时间内能够发送的最大分组数。例如：以每分钟6000个分组的平均速率来监管一个流，但限制峰值速率为每秒1500个分组

- **突发长度：**限制在极短的时间间隔能够发送到网络中的最大分组数

监管机制

漏桶机制： 行限制

对流的最大突发长度与平均速率进

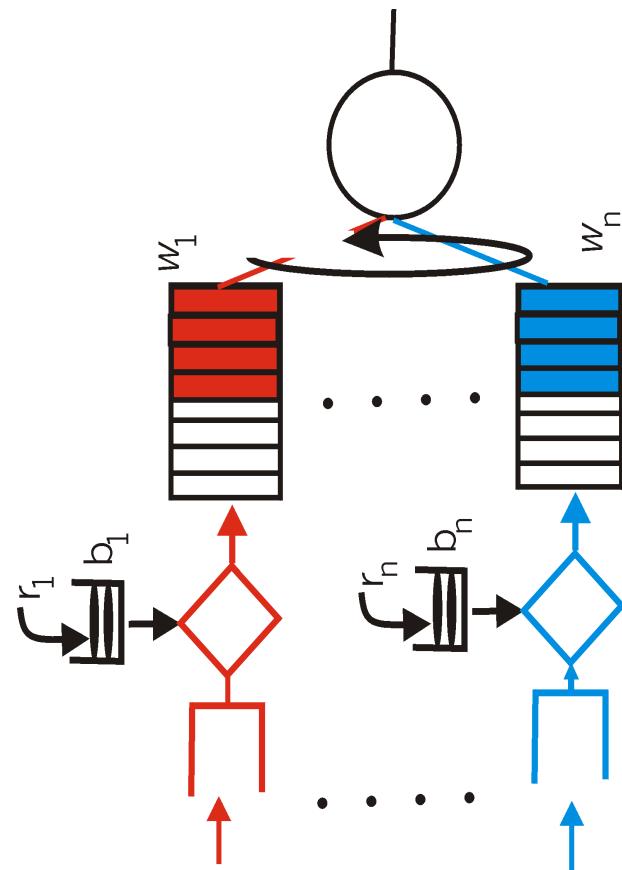
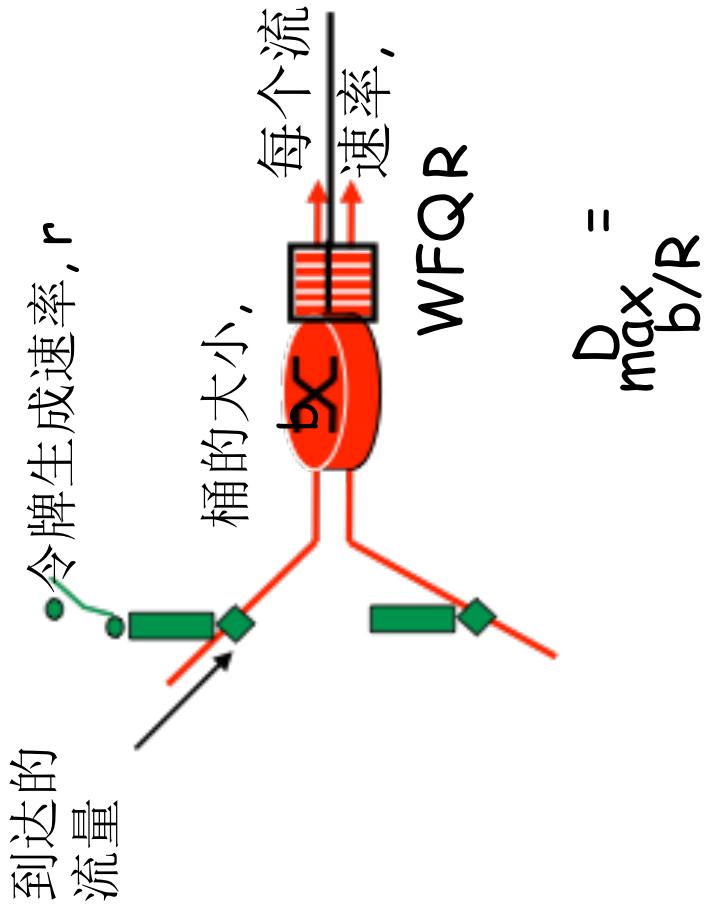


- 桶中最多装入 b 个令牌
- 令牌以每秒钟 r 个的速率加入到桶中直到桶装满 b 个令牌
- 在任何长度 t 的时间间隔内能够进入到网络中的最大分组数目为 $rt+b$



监管机制

- 漏桶与加权公平排队相结合，控制队列中的最大时延



区分服务(DiffServ)

- 以一种可缩扩和灵活的方式处理不同“类别”的流量
- **可缩扩性：**在网络核心放置简单功能，在网络边缘放置相对复杂功能
- **灵活性：**不定义服务类别，提供功能组件来构造不同的服务



区 分服 务体 系 结构

边缘路由器:



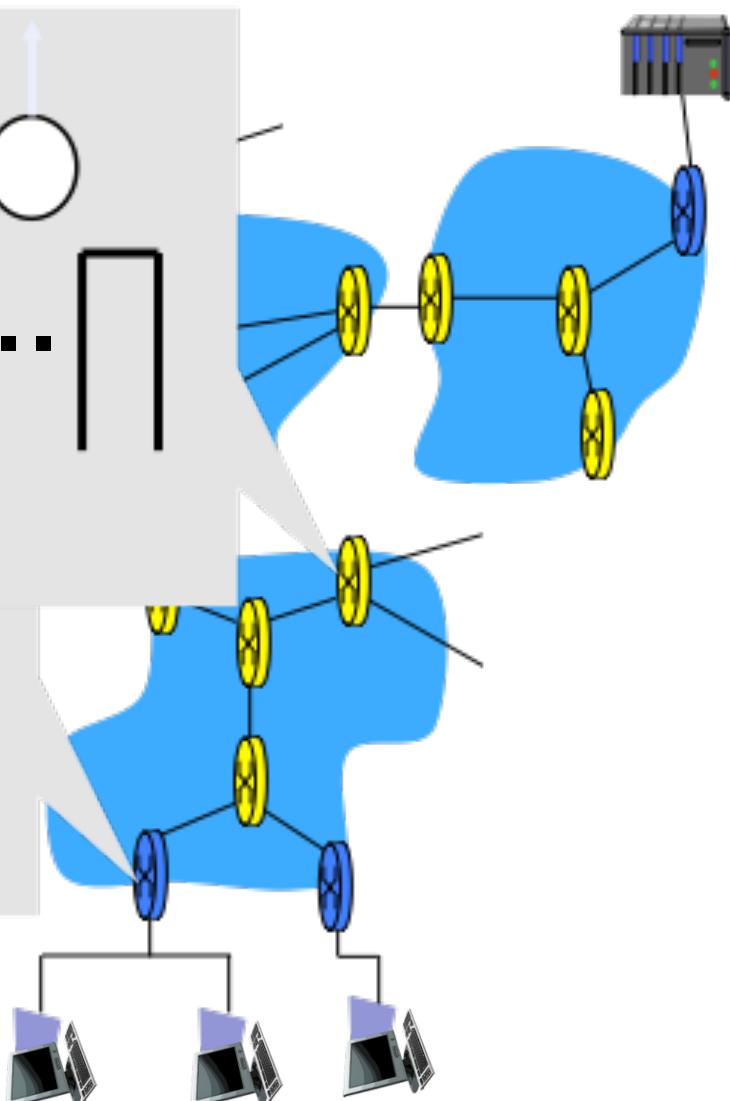
分组分类和流量调节

- 标记分组为 **in-profile** 和 **out-profile**

标记

r
b
—

调度



核心路由器:



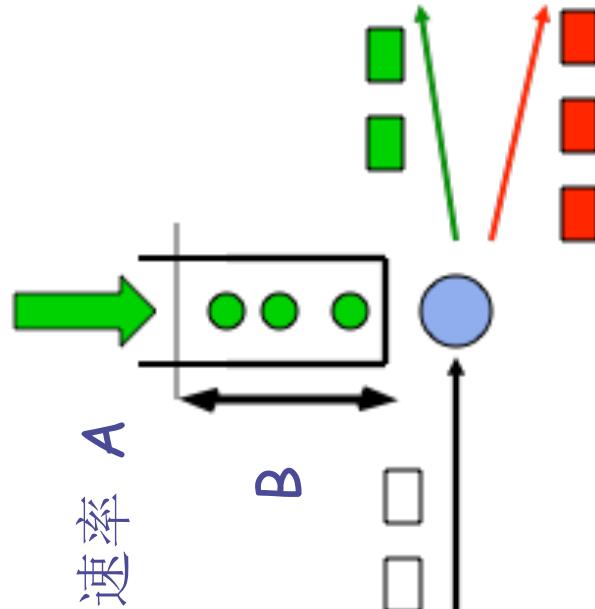
基于流量类别的转发

- 根据分组标记进行缓冲与调度
- 优先考虑 **in-profile** 分组

边缘路由器分组标记



- 流量配置文件：包含对峰值速率和分组流的突发度的限制

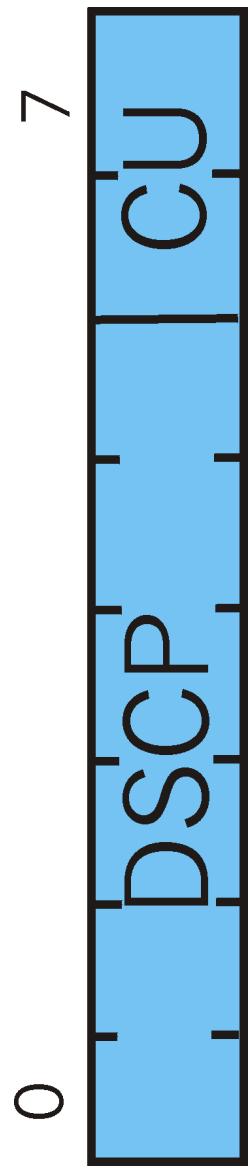


- 符合流量配置文件的分组得到它们的优先级标记并被转发
- 违反流量配置文件的分组可能被打上不同的标记，或被整形，或被丢弃。

分类和调节

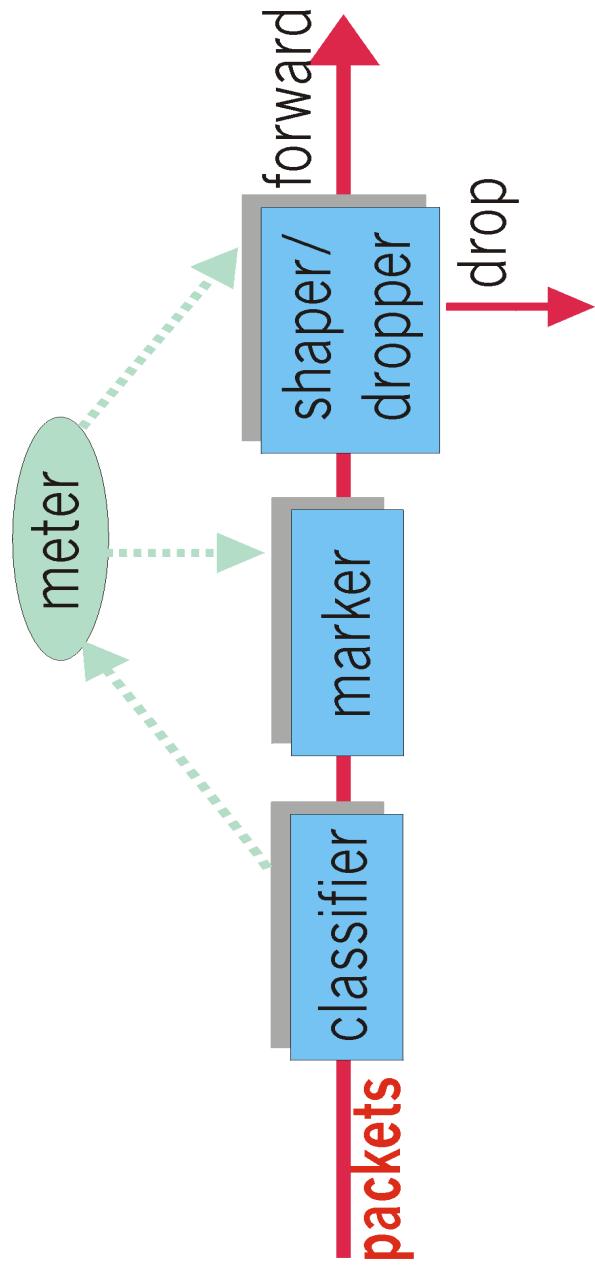


- 分组标记携带有IPv4 服务类型字段或IPv6分组首部DS字段
- 6比特用于区分服务类型(DSCP) 并且确定将接收分组的每跳行为
- 2比特暂时不使用



分类和调节

- 可以限制分组发送速率：
- 用户定义流量配置文件(包括平均速率和突发长度)
- 分组标记，整形功能



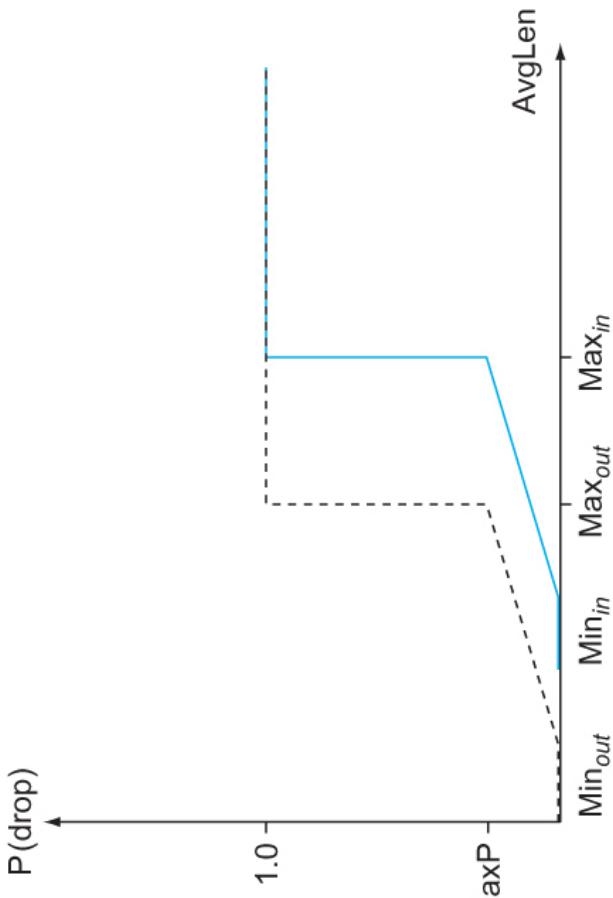
每跳行为(PHB)



- PHB：外部可观察的转发行为的描述，应用特定的DiffServ行为聚合
- PHB使得不同服务类别别的流量能享受到不同网络服务能力
- PHB不限制特定机制以获得这些性能
- 例如：一个PHB不要求为了获得一个特定行为而使用特定的分组排队规则(优先级队列，加权公平排队或先来先服务队列)

每跳行为 (PHB)

- **加速转发PHB(EF):**一类流量离开路由器的速率必须等于或者大于一个已配置的速率
- 为一个流量类别提供了最小保障的链路带宽
- **确保转发PHB(AF):**将流量分为4类
- 每个类都确保提供某种最小数量带宽和缓冲区
- 每一类进一步划分为三个“丢弃优先级”



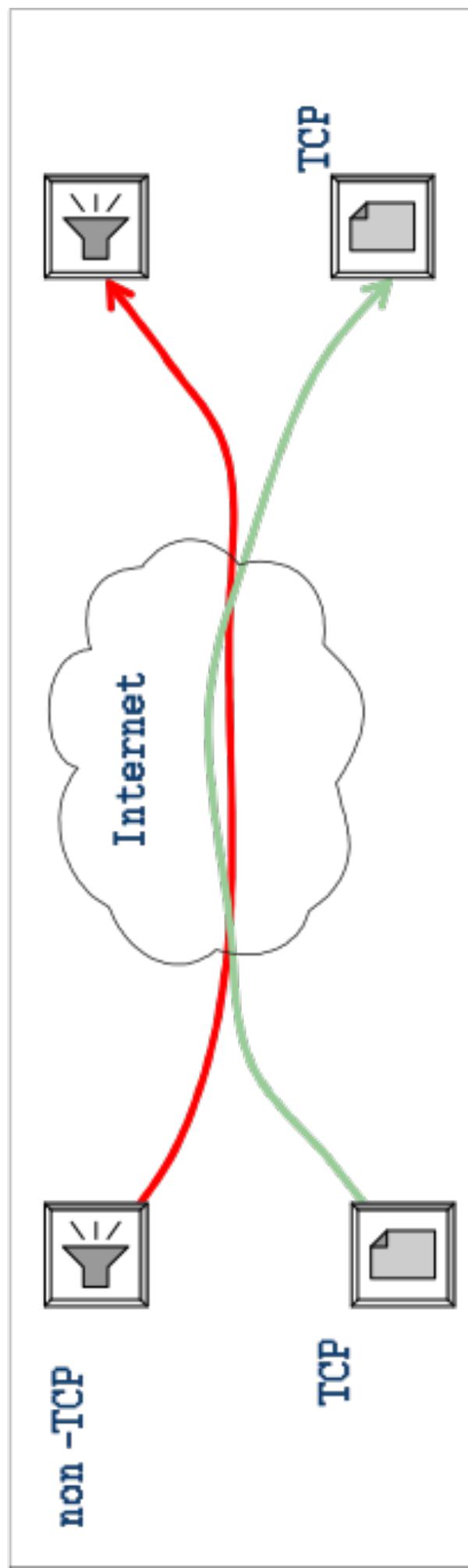
- 引言
- 核心问题: 分配资源
- 资源分配问题
- 排队规则
- TCP 拥塞控制
- 拥塞避免机制
- 服务质量(QoS)
- 服务需求
- 应用需求
- 端到端自适应
- 综合服务(RSVP)
- 区分服务(EF、AF)
- 基于等式的拥塞服务
- 总结



基于等式的拥塞控制: 动机

- 只有所有数据流都配合，拥塞控制才能有效
- 不受限制的UDP流量会影响TCP吞吐率

使用什么方式规范UDP流，使其响应网络拥塞？



多媒体传输: TCP vs UDP

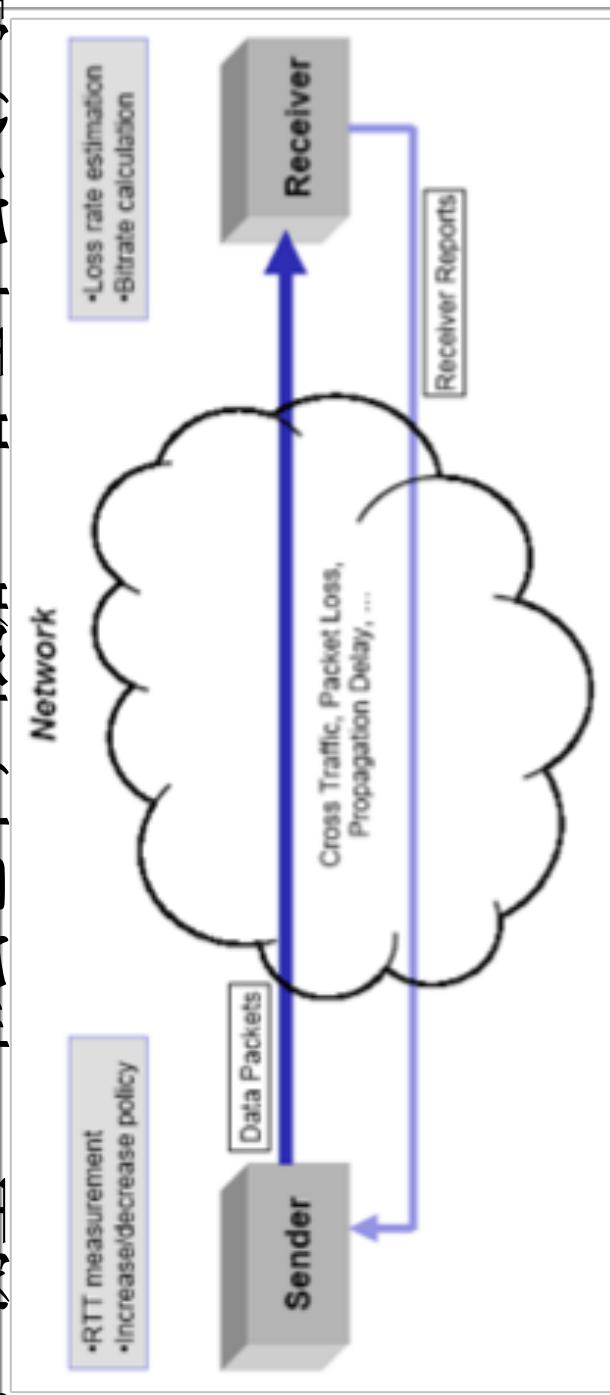


- 流媒体很少使用TCP
 - TCP重传机制引入额外时延(>500ms)
 - 当接收端收到重传数据，已经超过播放时间点
 - AIMD引入时延抖动和吞吐率变化过快
 - 无必要引入TCP流控：接收端需要以编码速率处理数据
- 为什么关注TCP
 - 互联网流量主要使用TCP
 - 非TCP流量已经非常影响TCP流量
 - 一般而言，多媒体应用主要使用UDP
 - 如何使UDP流量做到TCP友好？



基于等式的拥塞控制: 基本想法

- 定义: 在同样的网络条件下, 某个链接的长期吞吐率不超过一个TCP流的吞吐率
- 使UDP流看起来像TCP流, 避免过于激进的发送流量
- 激进的数据流可能受到路由器主动缓存控制机制的打压
- 不希望是其他TCP流吞吐率过低
- 目标 缓慢适应拥塞窗口
- 公平竞争条件下, 实现TCP友好
- 方法: 测量RTT和丢包率, 根据TCP吞吐率公式, 控制发送速率





- 引言
 - 核心问题: 分配资源
 - 资源分配问题
 - 排队规则
 - TCP 拥塞控制
 - 拥塞避免机制
 - 服务质量
 - 总结
- 

拥塞控制



- 拥塞是不可避免的
- Internet中不会预先进行资源分配
- 基于主机vs.基于路由器的拥塞控制
- TCP 拥塞控制
 - 采用分组丢失作为拥塞的判别标示
 - 发送方采用CongestionWindow 适应网络累次增加, 成倍减少
 - 慢启动, 缓慢开始重启

TCP 设计：问题及解决方案



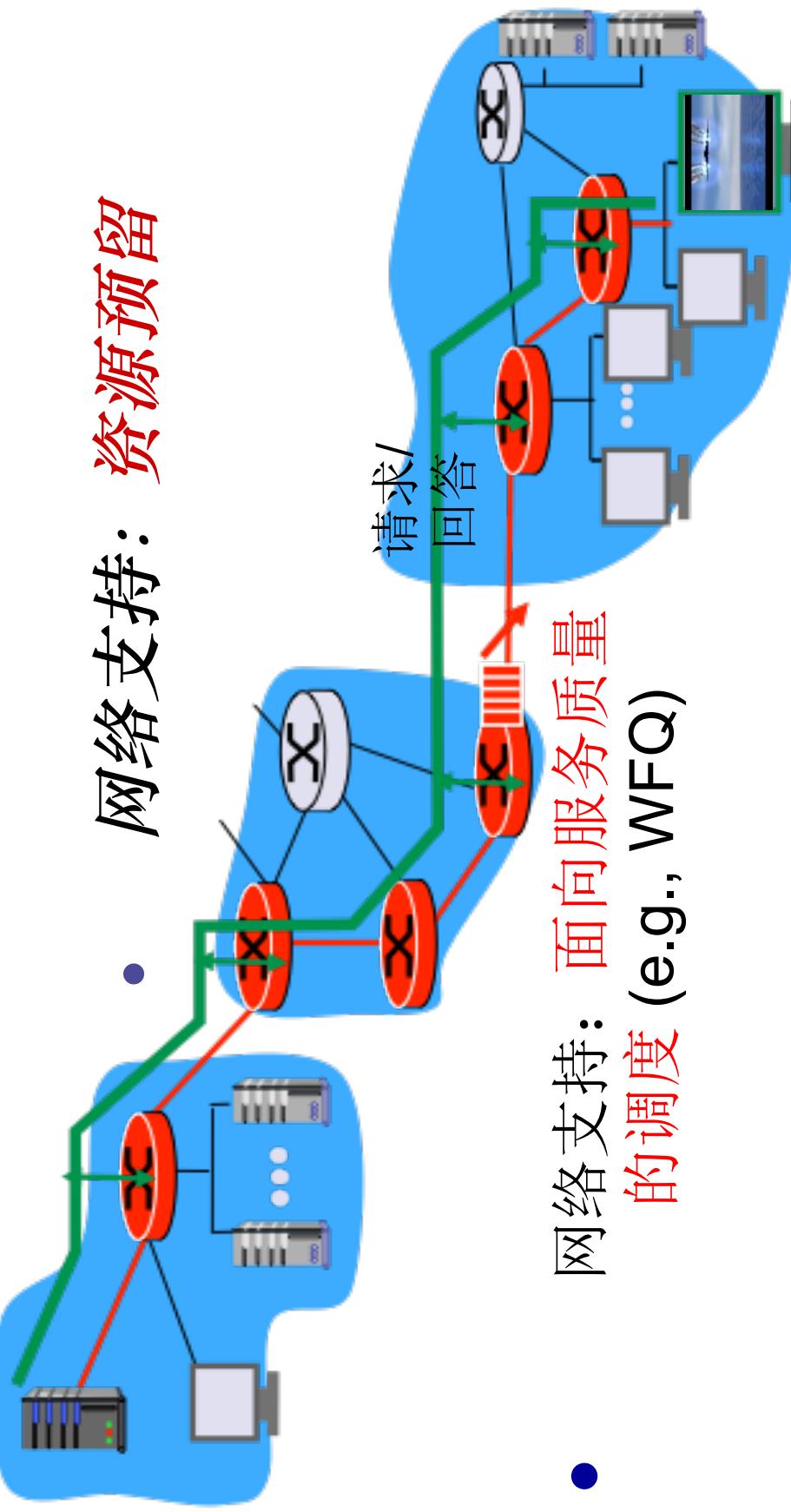
No.	问题及挑战	解决方案	章节
1	连接建立	建立：三次握手 终止：四次握手	5.2.3
2	超时定时器问题	采用Jacobson/ Karels算法估计RTT	5.2.6
3	分组乱序到达	基于窗口的缓存管理	5.2.4
4	流量控制	通过AdvertisedWindow通告实现基于窗口的流量控制	5.2.4
5	拥塞控制	基于CongestionWindow实现拥塞控制	6.3
6	协议扩展	TCP首部的Seq和AdvertisedWindow字段扩展	5.2.4
7	傻瓜窗口症状	Nagle 算法：基于ACK自计时	5.2.5

服务质量保证

- 服务器：多速率视频编码



• 网络支持：资源预留



• 网络支持：面向服务质量的调度 (e.g., WFQ)

- 客户端：自适应速率调整



谢谢！

华中科技大学
电子信息与通信学院
Email: itec@hust.edu.cn
网址: <http://itec.hust.edu.cn>





- Chapter 6 in L. L. Peterson and B. S. Davie, *Computer Networking: A System Approach (5th edition)*, Morgan Kaufmann, 2012
- Chapter 3/7 in James F. Kurose and Keith W. Ross, *Computer Networking: A Top-Down Approach (6th edition)*, Pearson Education Inc., 2012
- 吴功宜, 计算机网络 (第3版), 清华大学出版社社, 2011

附录