

# 第五章 网络层（5）

---

袁华: [hyuan@scut.edu.cn](mailto:hyuan@scut.edu.cn)

华南理工大学计算机科学与工程学院

广东省计算机网络重点实验室

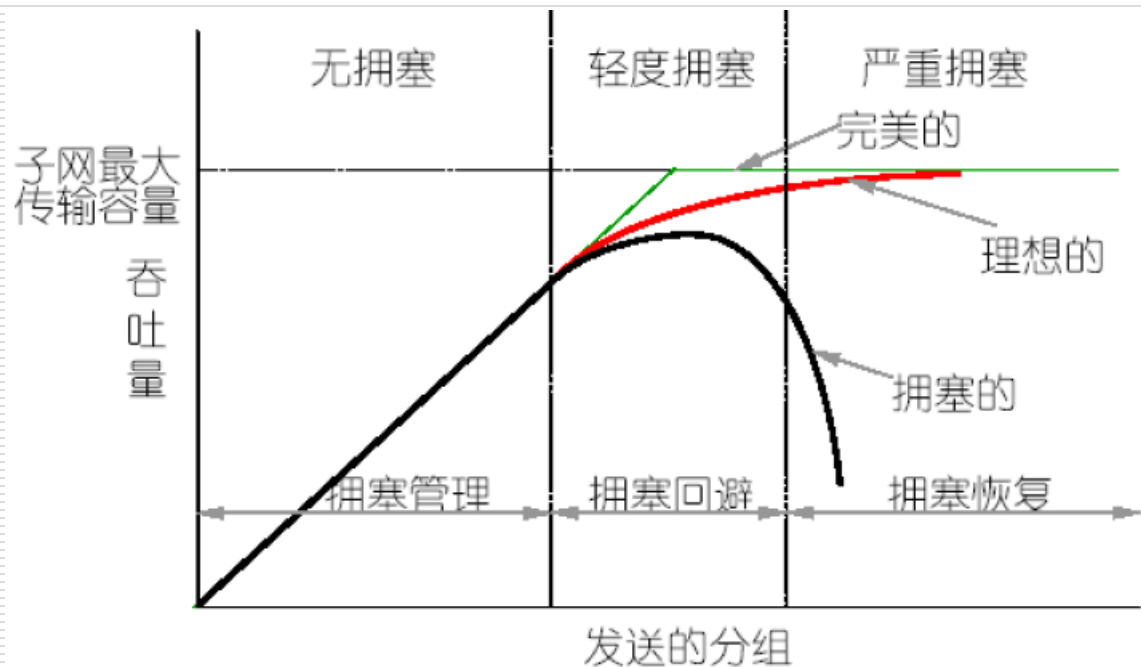
# 本节主要内容 (5.3 P302)

---

- 了解拥塞控制
- 学习流量整形方法 (traffic Shaping)
  - 漏桶 (leaky Bucket)
  - 令牌桶 (token bucket)
- 了解网络互联

# 什么是拥塞? P303

- 当一个子网或子网的一部分出现太多分组的时候，网络的性能急剧下降，这就是拥塞（**Congestion**）



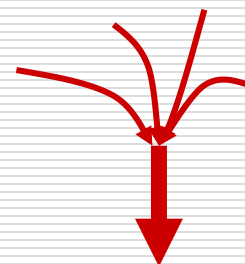
# 导致拥塞的因素 P303

## □ 输入流量速度大于输出线路的容量

■ 如：几根输入线路同时向一个输出线路转发分组

■ 增加内存可以提高性能到某种程度

■ 如果增加内存到无穷会怎样呢？



## □ 慢速的处理器也可能引起拥塞，如线路容量充足，但处理器来不及处理

## □ 线路容量和处理器能力需要平衡

# 拥塞控制 vs. 流控 P303

---

## □ 拥塞控制（Congestion control）

- 任务：确保子网能够承载所到达的流量
- 这是个全局的问题，涉及到主机、路由器，存储转发的过程等方方面面的问题

## □ 流控（Flow control）

- 只与特定的发送方和接收方之间的点到点流量有关
  - 确保一个快速的发送方不会持续地以超过接受方接收能力的速率传输数据
- 一台主机会收到 “slow down” 消息，不管是因为接收者无法处理分组还是网络无法处理

# 拥塞控制的通用原则 P326

## □ 开环（Open loop）

- 试图用良好的设计来解决问题，本质是从一开始就保证问题不会发生
- 开环决策制定不考虑网络的当前状态

不是不明白，这世界变换太快！

## □ 闭环（Closed loop）

- 建立在反馈环路的概念上，分三个步骤：
  - 监视系统，**检测**何时何地发生了拥塞
  - 把这些信息**传递**到能够采取行动的地方
  - 调整系统的运行，以**改正**问题

# 怎么知道拥塞了？--拥塞量度

---

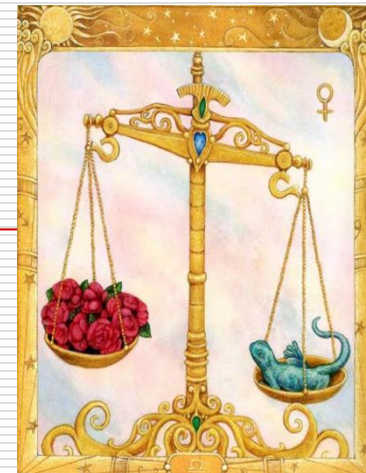
1. 因为缺乏缓存空间而丢弃的分组百分比
  2. 平均队列长度
  3. 超时和重传的分组数
  4. 平均分组延迟
  5. 分组延迟的标准方差 (**standard deviation**)
- 上述这些度量，数值越大表示拥塞的程度越重

# 检测到拥塞后怎么办？--拥塞信息传播

- ❑ 检测到拥塞的路由器发送一个警告分组给流量源，但是。。。。。
- ❑ 其他方法：每个分组可以保留一位或一个域，当拥塞度量超过阈值的时候，路由器填充该位或域，以此警告它的邻居。
- ❑ 其他方法：主机或路由器周期性地向外发送探测分组，显式地询问有关拥塞的情况，然后，在有问题的区域利用回收的信息来路由流量 (类比：交通电台)



# 怎样解决拥塞问题？ P304

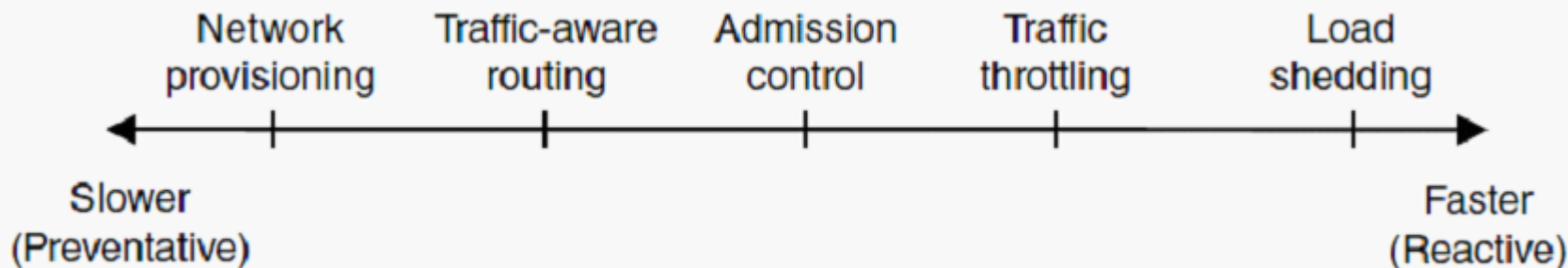


- 拥塞根源：负载 > 资源
- 增加资源
  - 在某些点之间使用更多的通道增加带宽（比如：广深）
  - 把流量分散到多条路径
  - 启用空闲或备份的路由器
- 降低负载
  - 拒绝为某些用户提供服务（比如：春节车票提价，限行）
  - 给某些用户的服务降低等级（比如：黄金周旅游）
  - 让用户更有预见性地安排他们的需求（比如：年假制）

# 拥塞控制途径？ P304

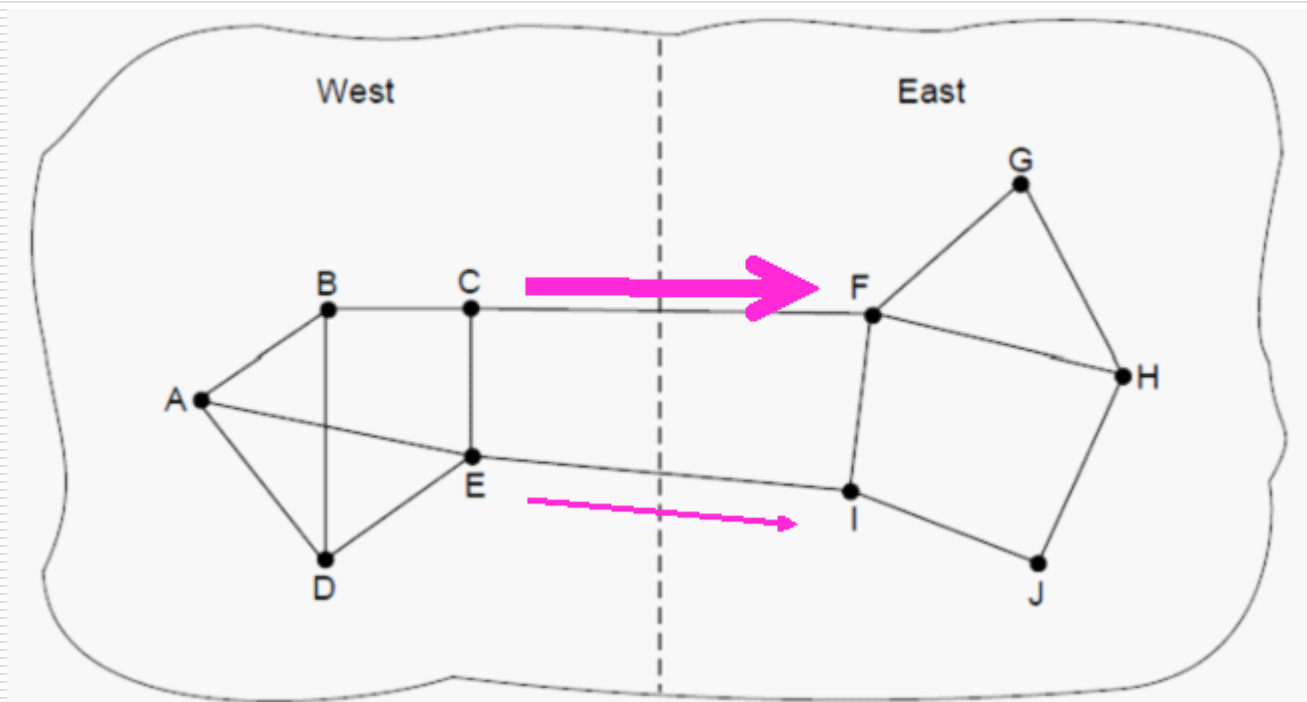
## □ 开环系统中的拥塞控制方法：

- 一开始就将拥塞发生的可能性降到最低，而不是随它发生了之后再去做反应
- 在各个层面上采用适当的策略达成这个目标



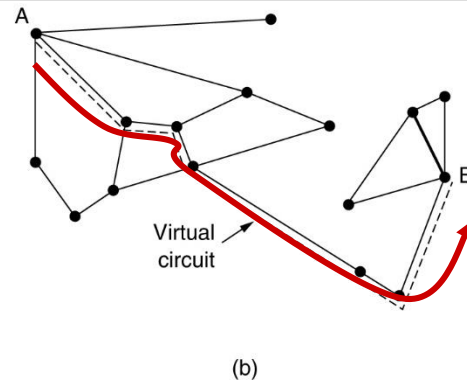
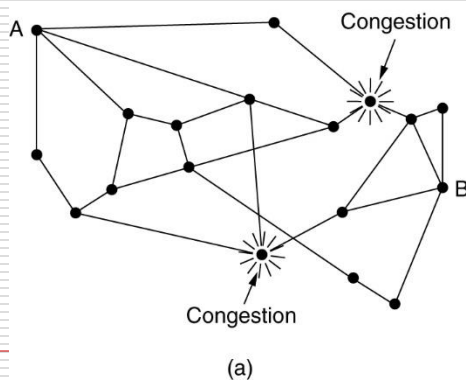
# 流量感知路由P305

## □ 可能导致路由摇摆



# 准入控制 P305

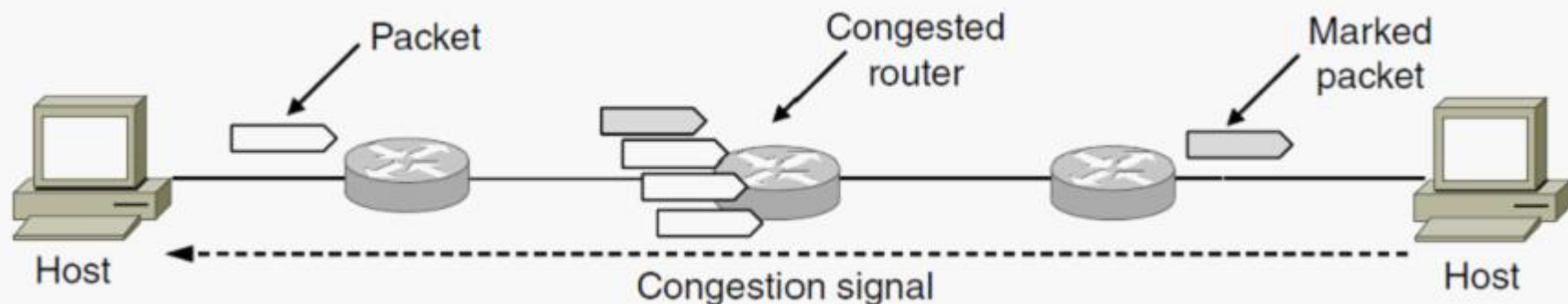
- 准入控制 (**Admission control** , 简单但粗鲁) (**单双号限行**)
  - 防止拥塞进一步恶化和加剧
  - 基本思想: 一旦出现拥塞信号, 则不再创建任何虚电路, 直到问题得到解决
- 绕过问题区域 (**Alternate Routing**)
  - 允许建立新的虚电路, 但是仔细选择线路, 绕开问题区域
- 在主机和子网之间协商, 达成一致的约定, 所以子网会沿途预留资源 (**资源预留**)
  - 流量的容量和形状
  - 所要求的服务质量
  - 其他参数



# 流量调节 P307

□ 拥堵路由器通告源机发送慢下来

■ ECN (Explicit Congestion Notification) marks packets and receiver returns signal to sender

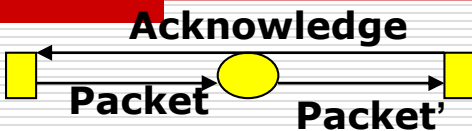


# 数据报子网中的拥塞控制<sub>P308</sub>

---

- 每台路由器可以监视它的输出线路和其它资源的使用情况
- 每条线路和一个实变量u关联在一起，其值位于(0.0-1.0)之间
- 无论何时，只要 **u** 超出了阈值，对应的线路就进入到警告 “**warning**” 状态
- 每个新到达的分组都将被检查，看它的输出线路是否处于 “警告状态”

# 处于警告状态后可以采取的措施<sup>P309</sup>

- 
- **The Warning Bit (警告位)**
    - 分组头部中的一个特殊的位
    - 该位被复制到下一个确认分组中, 被传回源
    - 源机会监视设置了警告位的分组的比例, 相应地调整它的发送速度
  - **Choke Packets (抑制分组)**
    - 路由器给源机发回一个抑制分组, 并在抑制分组中指明原分组的目标地址
    - 当源机收到抑制分组后, 它会以某种百分比减少它发向该目标的流量
    - 在一段固定的时间内, 源机会忽略同样一个目标的抑制分组
    - 过了这段时间, 会继续侦听有无新的抑制分组, 如果没有, 源机将增加流量

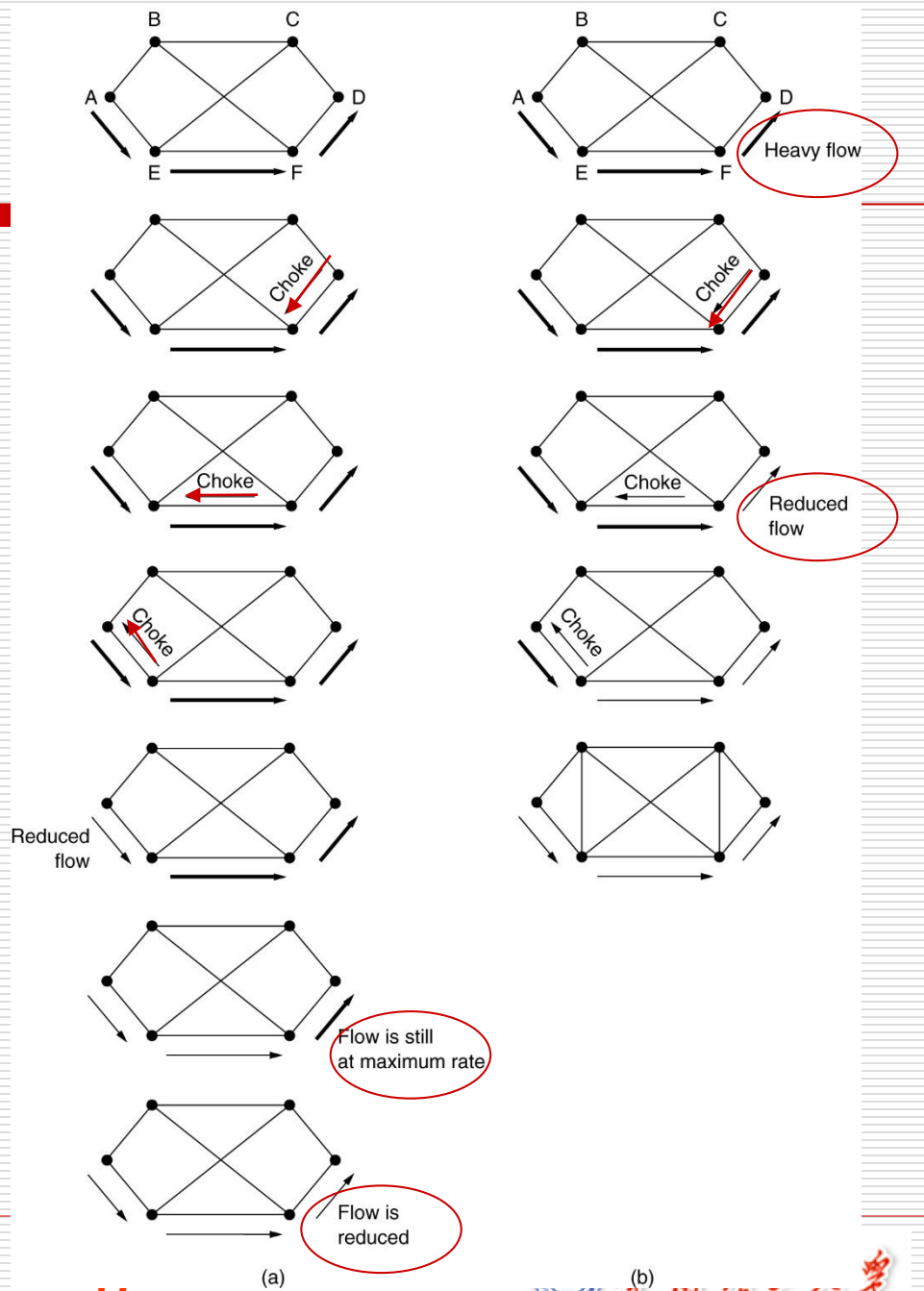
# 处于警告状态后可以采取的措施（续）

## □ 逐跳抑制分组（Hop-by-Hop Choke Packets）

- 当网络速度很高或者路由器离源主机的距离很远的时候，给源主机发送抑制分组并不能很好地起作用，因为反应太慢（图 5.26a）
- 一种改进的方法是让抑制分组路径的每个路由器都采取相应的措施
- 这种逐跳抑制的机制的效果就是，拥塞点上的拥塞很快得到了缓解，但是付出的代价就是，上游路由器需要更多的缓存空间



# 抑制分组机制



# 负载丢弃/载荷脱落P310

- 这是处理拥塞的最极端的方法
- 当路由器收到的分组超载了，一些分组会被丢掉
- 丢掉哪些分组呢？
  - 随机丢弃（ **random** ）
  - 丢弃新到达的（**葡萄酒策略**，适合文件传输类）
  - 丢弃早到达的分组（**牛奶策略**，适合多媒体类）
  - 丢弃不太重要的（ **less important** ）分组
    - 需要发送方在它们的分组中标明优先级

# 随机早期检测RED（防患于未然）

---

- 当情况变得恶化无可救药之前就开始丢弃分组
- 为了确定什么时候开始丢弃分组，路由器维护着最近的队列平均长度
- 当某条线上的队列平均长度超过了某阈值时，该线路被认定是拥塞的，可以采取相应的措施
- 告诉源机关于拥塞的情况有两种可能
  - 发送抑制分组
  - 仅仅丢弃，不做任何报告

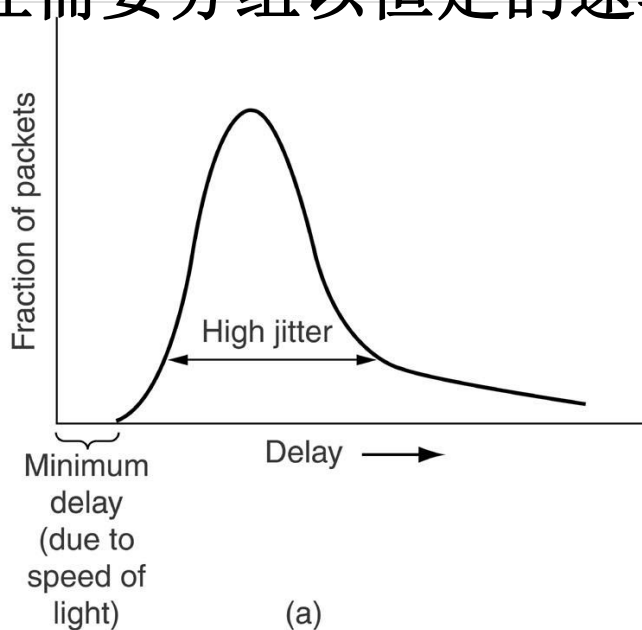
# 应用的QoS需求P312

---

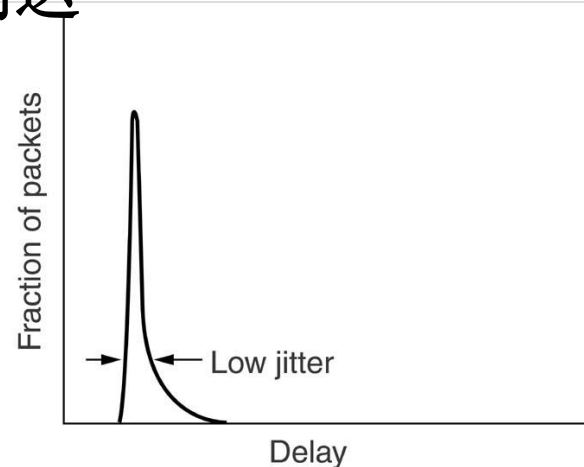
Application	Bandwidth	Delay	Jitter	Loss
Email	Low	Low	Low	Medium
File sharing	High	Low	Low	Medium
Web access	Medium	Medium	Low	Medium
Remote login	Low	Medium	Medium	Medium
Audio on demand	Low	Low	High	Low
Video on demand	High	Low	High	Low
Telephony	Low	High	High	Low
Videoconferencing	High	High	High	Low

# 抖动控制

- 分组到达时间的变化量 (标准方差 **standard deviation**) 叫做抖动 (**jitter**)
- 在实时传输中, 抖动可以引发问题, 因为实时传输往往需要分组以恒定的速率到达



(a)



(b)

# 如何做抖动控制（续）

---

- 通过计算出沿途每一跳的期望传输时间，就可以控制抖动。
  - 当一个分组到达路由器，路由器检查这个分组，看它是来晚了还是来早了
  - 如果分组来早了，那么它可能多停留一些时间，以便回到预定的时间点上
  - 如果分组来晚了，路由器应该尽可能快地将他转发出去
- 当几个分组都要使用同一根输出线路转发出去，哪一个分组优先呢？
  - 路由器总是让偏离预定时间最远的那个分组优先转发

# 流量整形 (P313)

---

- 调节数据传输的平均速率（和突发数据流）
- 算法
  - 漏桶（leaky bucket）
  - 令牌桶（token bucket）
  - 其它：资源预留、准入控制、分组调度等
- 可以减少拥塞

# 漏桶算法P314

## □ 算法描述

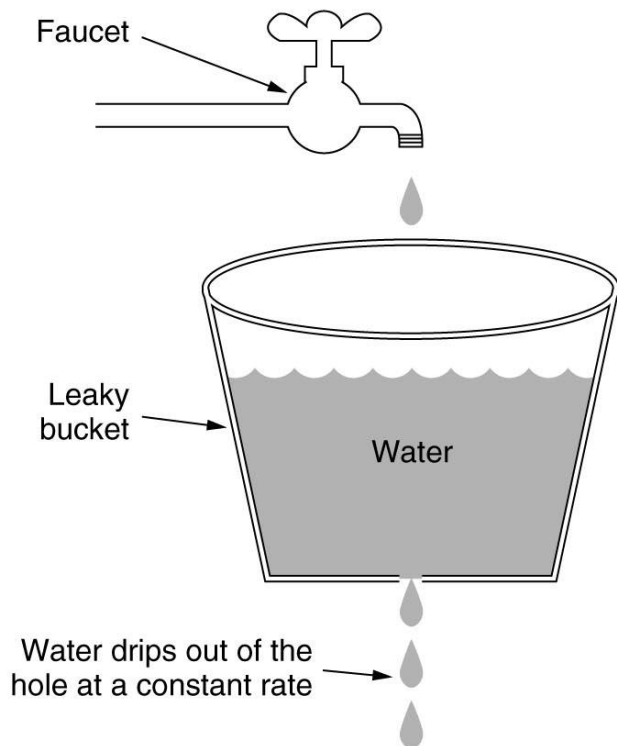
- 每个主机连接到网络的接口中都有一个**漏桶**，即一个**优先长度的内部队列**
- 当桶中有分组的时候，输出速率是恒定的，当桶空的时候，输出速率是0
- 当一个分组到达满的桶的时候，分组将被丢弃（满则溢）
- 每个时钟嘀嗒( **tick** )，仅允许一个分组或固定数量的分组发送出去

## □ 算法效果

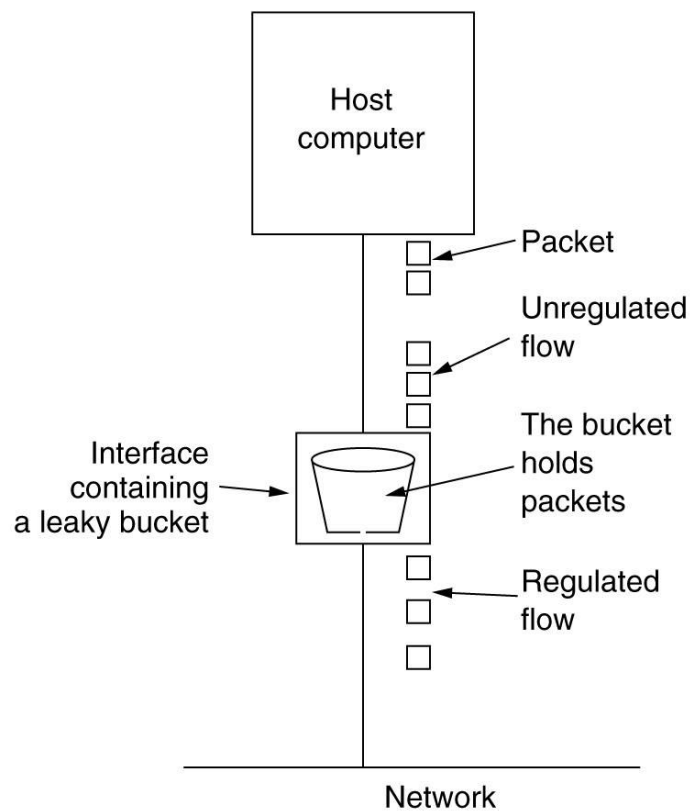
- 主机内用户进程产生的分组流往往是一个**不稳定的流**，漏桶可以让**它输出到网络时变成一个稳定流**，抹平了突发尖峰，极大地减少了发生拥塞的机会



# 漏桶算法 (续)



(a)

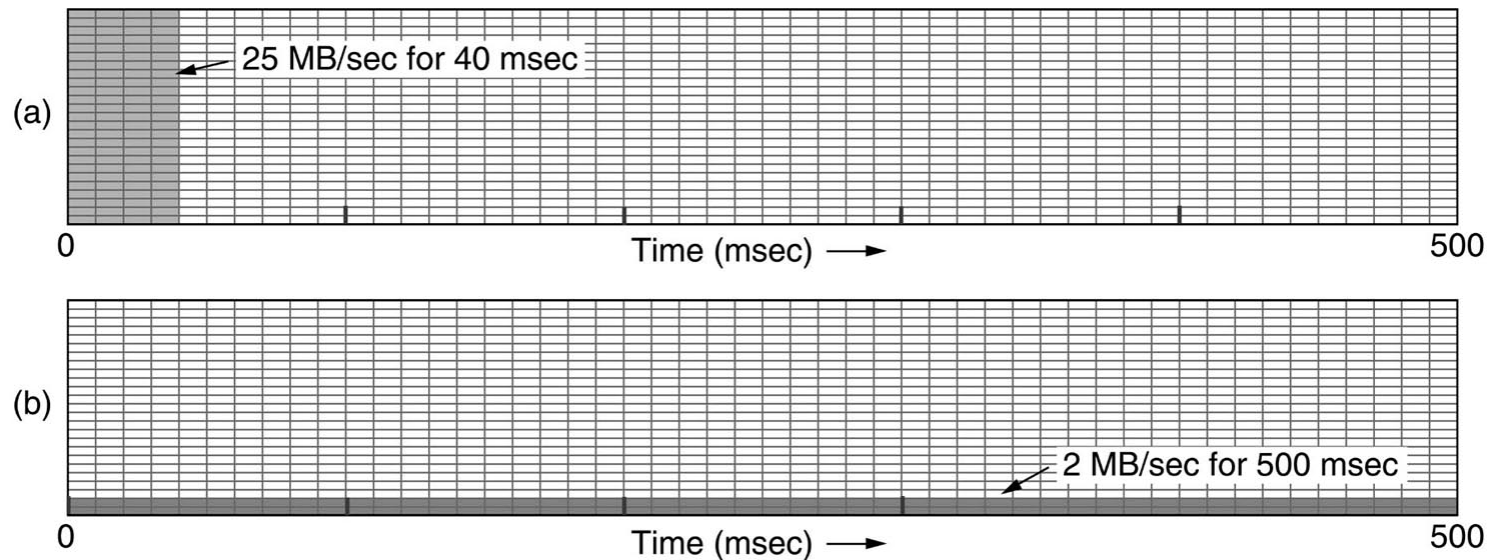


(b)

# 漏桶算法 (续)

## □ 例子

- 数据产生速率: 25MBps
- 路由器的工作速率: 2MBps
- 数据突发时间持续: 40ms
- 漏桶输出速率: 2MBps
- 漏桶容量: 1MB



# 漏桶的缺点

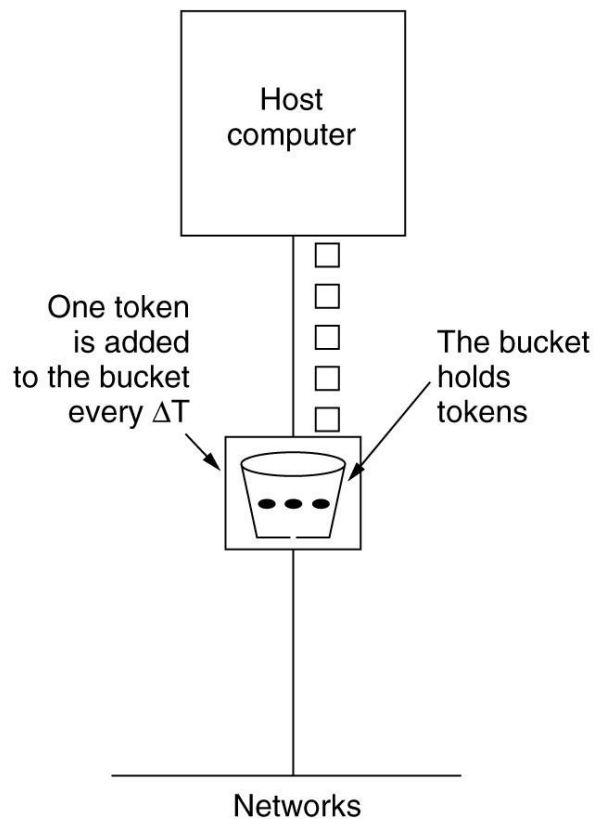
---

- ❑ 当漏桶满了之后，数据将被丢弃
- ❑ 不能大量地突发数据
- ❑ 改进：令牌桶

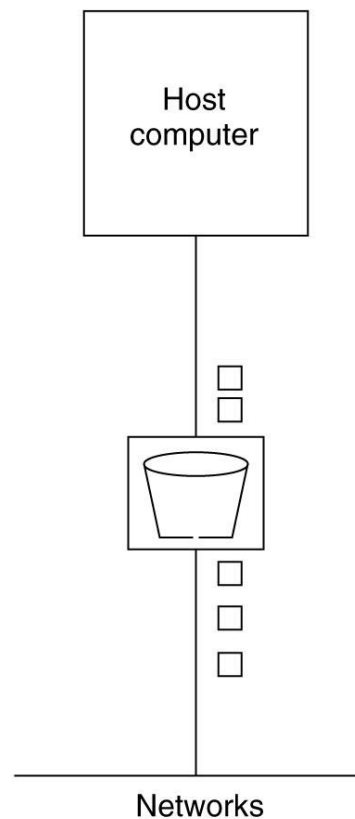
# 令牌桶算法P315

- 当大量数据突发的时候，令牌桶算法允许输出加快到某种程度
- 令牌桶拥有令牌（**tokens**），且以每 $\Delta T$ 秒产生一个令牌的速度往桶中输入令牌
- 一个分组要发送的时候，它必要从桶中取出和获取到一个令牌
- 令牌桶算法允许累积令牌，但最多可以累积 $n$ （令牌桶的容量）个令牌
- 和漏桶算法相比：
  - 令牌桶允许突发，但是最大突发受制于令牌桶容量的限制
  - 当桶满的时候，令牌桶算法丢掉的是令牌（不是分组）

# 令牌桶算法 (续)



(a)



(b)

# 令牌桶算法 (续)

## □ 计算最大突发时间

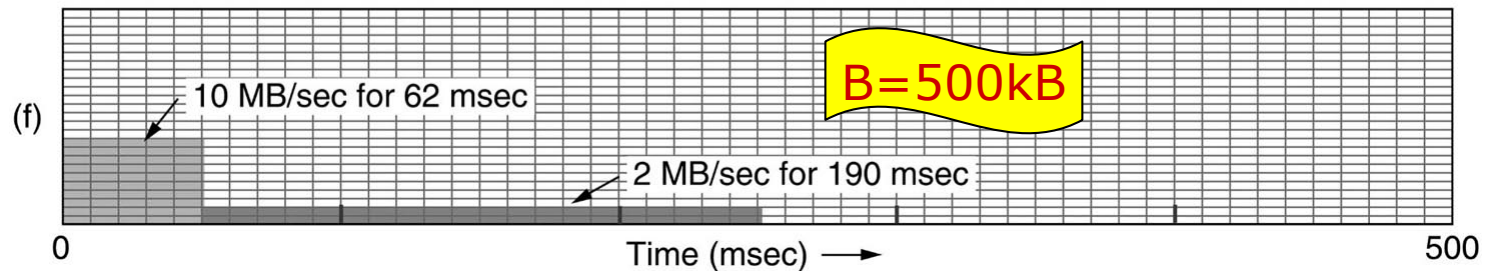
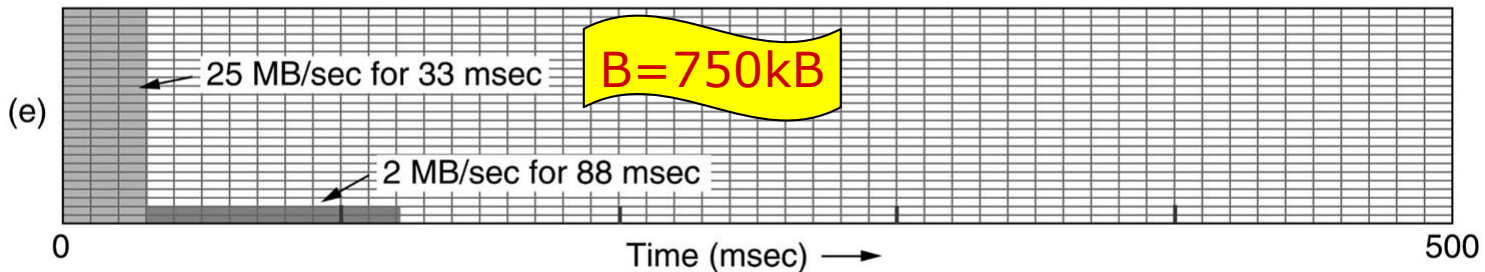
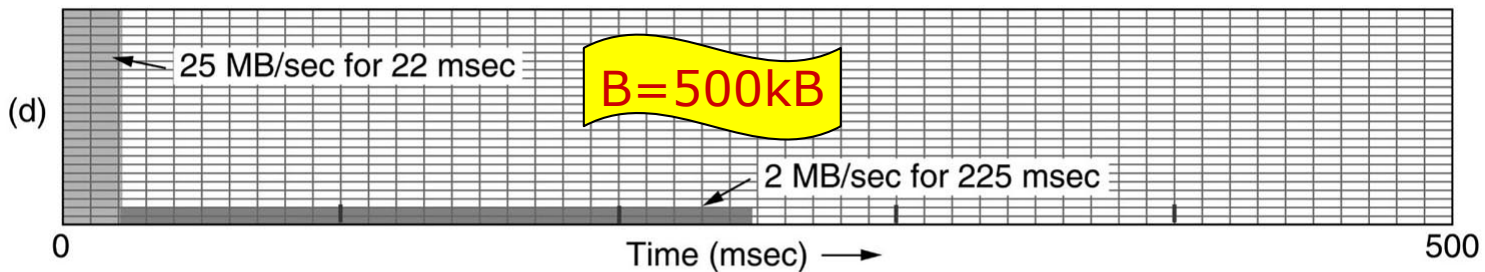
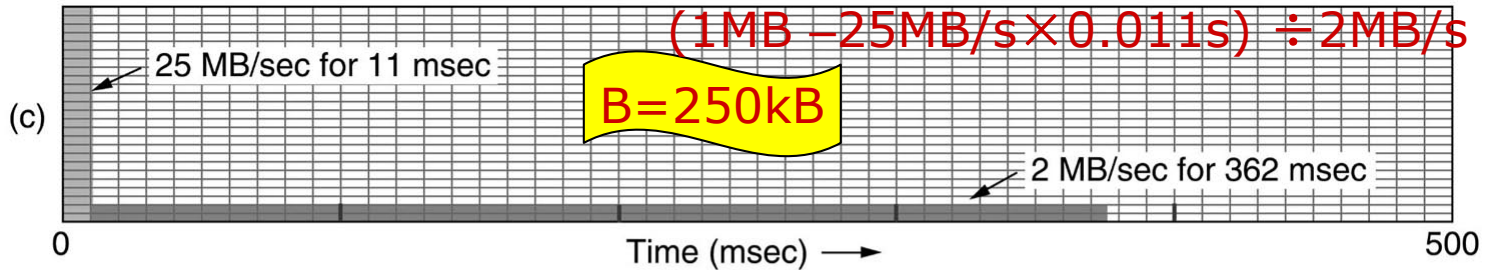
- 突发时间: **S** 秒
- 令牌桶容量: **B** 字节
- 令牌到达的速率: **R** 字节/秒
- 最大输出速率: **M** 字节/秒

$$B + RS = MS$$

$$\rightarrow s = B / (M - R)$$

$M=25\text{MB}$ ,  $R=2\text{MBps}$ , 当 1M 突发数据到达时

$$(1\text{MB} - 25\text{MB/s} \times 0.011\text{s}) \div 2\text{MB/s} = 362\text{ms}$$



# 课堂练习

---

- 一个10Mb/s的网上有一台由令牌桶控制的主机，令牌桶以2Mb/s的速率填充。假定令牌桶有20Mb的容积。（1）问主机能以10Mb/s全速发送数据的最大可能的时间是多少？
- （2）主机能以10Mb/s全速发送数据的最大数据量是多少？



# 参考答案

---

□ 已知:

令牌桶容量:  $B = 20\text{Mb}$

令牌产生速率:  $R = 2\text{Mb/s}$

最大突发速率:  $M = 10\text{ Mb/s}$

设突发时间为  $s$ , 则:

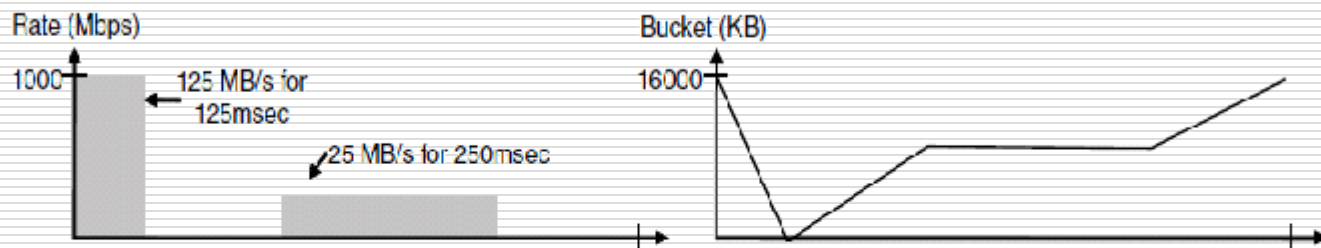
$$s = B / (M - R) = 20 / (10 - 2) = 2.5 \text{ (s)}$$

□ 设最大突发数据量是  $V_{\max}$ , 则:

$$V_{\max} = s \times M = 2.5(\text{s}) \times 10\text{Mb/s} = 25\text{M (bits)}$$

# P315

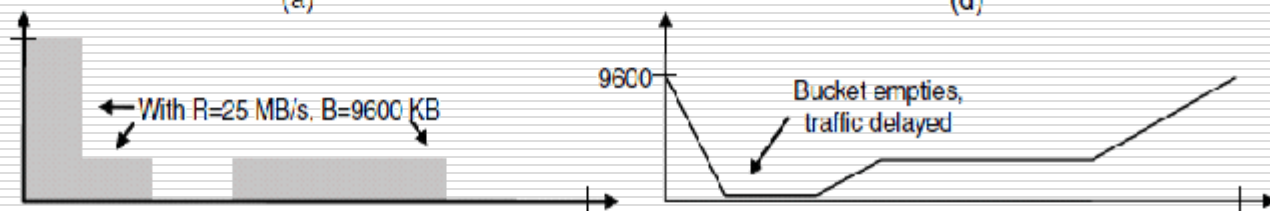
Host traffic  
 $R=200$  Mbps  
 $B=16000$  KB



(a)

(d)

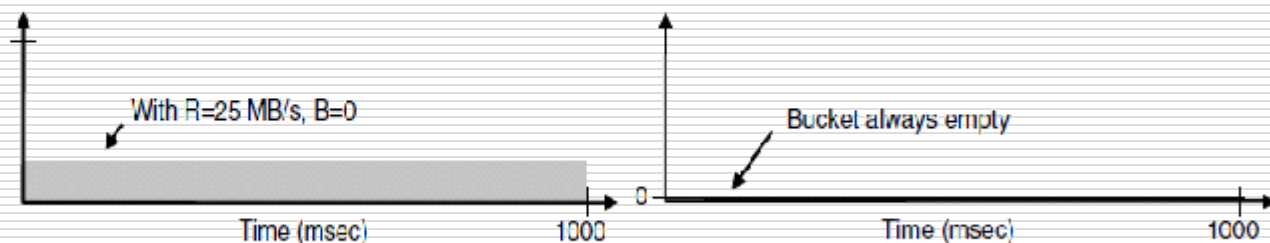
Shaped by  
 $R=200$  Mbps  
 $B=9600$  KB



(b)

(e)

Shaped by  
 $R=200$  Mbps  
 $B=0$  KB



(c)

(f)

# 简明小结

---

- ❑ **Best-effort (BE)**
- ❑ **拥塞控制，保证QoS非常困难**



# 网络互联 (5.5节 P326)

- 到目前为止，我们一直假设面对的网络是同质的网络（**homogeneous**）network.
- 事实上，总存在很多不同的网络
  - 不同的网络的安装基数都很大且在增长（如：TCP/IP, SNA, ATM, Novell NCP/IPX, AppleTalk, wireless, 等）
  - 随着计算机和网络变得越来越便宜，购买设备的决策权正在向下迁移，这容易导致一个大公司安装了不同的网络
  - 不同的网络（如：ATM 和无线网络）使用完全不同的技术

# 网络的不同之处P327

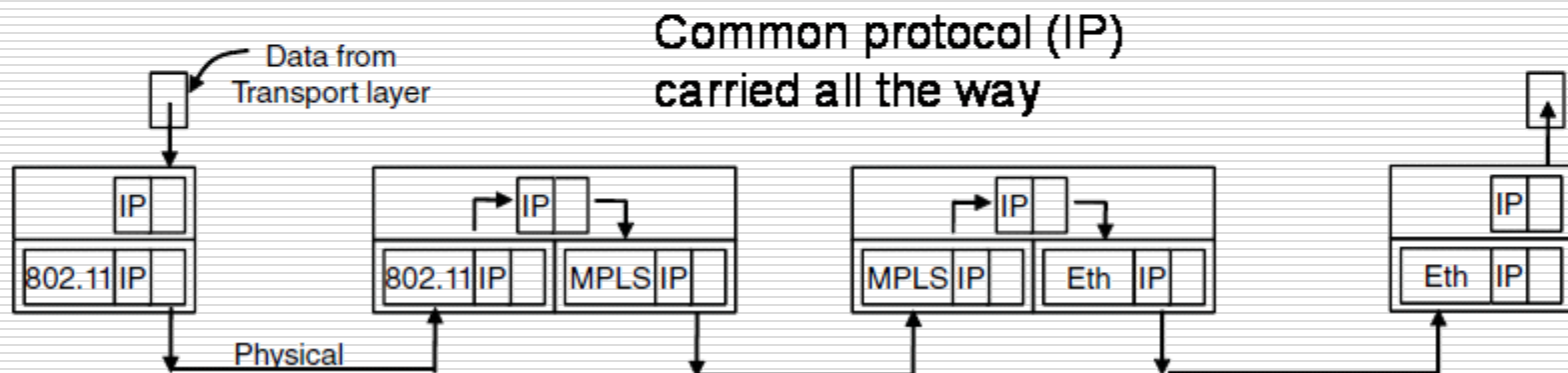
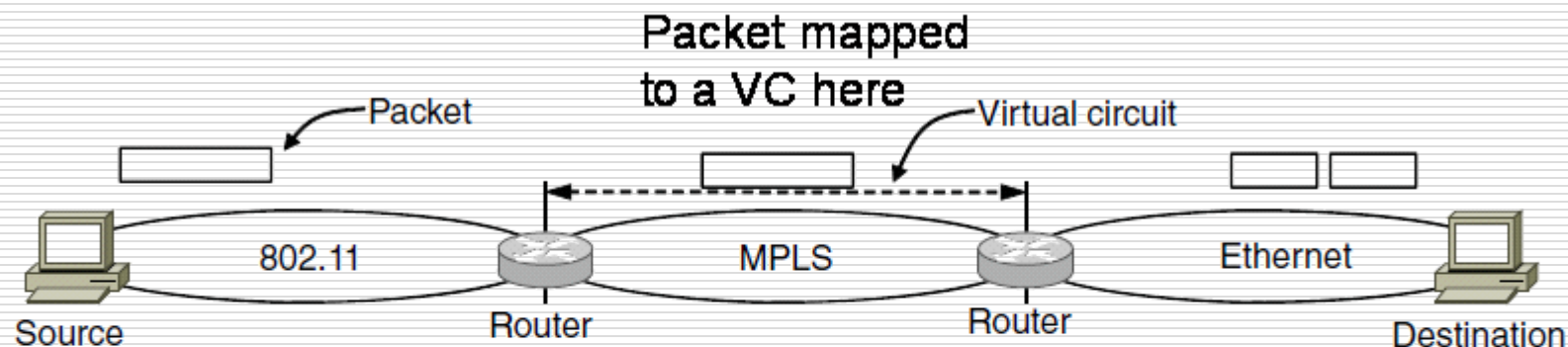
Item	Some Possibilities
Service offered	Connection oriented versus connectionless
Protocols	IP, IPX, SNA, ATM, MPLS, AppleTalk, etc.
Addressing	Flat (802) versus hierarchical (IP)
Multicasting	Present or absent (also broadcasting)
Packet size	Every network has its own maximum
Quality of service	Present or absent; many different kinds
Error handling	Reliable, ordered, and unordered delivery
Flow control	Sliding window, rate control, other, or none
Congestion control	Leaky bucket, token bucket, RED, choke packets, etc.
Security	Privacy rules, encryption, etc.
Parameters	Different timeouts, flow specifications, etc.
Accounting	By connect time, by packet, by byte, or not at all

# 网络是怎样联接起来的？

---

- 网络可以通过不同的设备联接起来：
  - 物理层 - repeaters or hubs
  - 数据链路层 - bridges and switches
    - 可能作小小的协议转换，如从Ethernet 到 FDDI 或到 802.11
  - 网络层 - routers
    - 多协议路由器
  - 传输层 - transport gateways
    - 传输层连接之间的接口，如允许TCP连接和SNA连接粘结起来，分组可畅通无阻
  - 应用层 - application gateways
    - 翻译消息语义，如不同的email格式转换

## □ IP是现代网络的基础

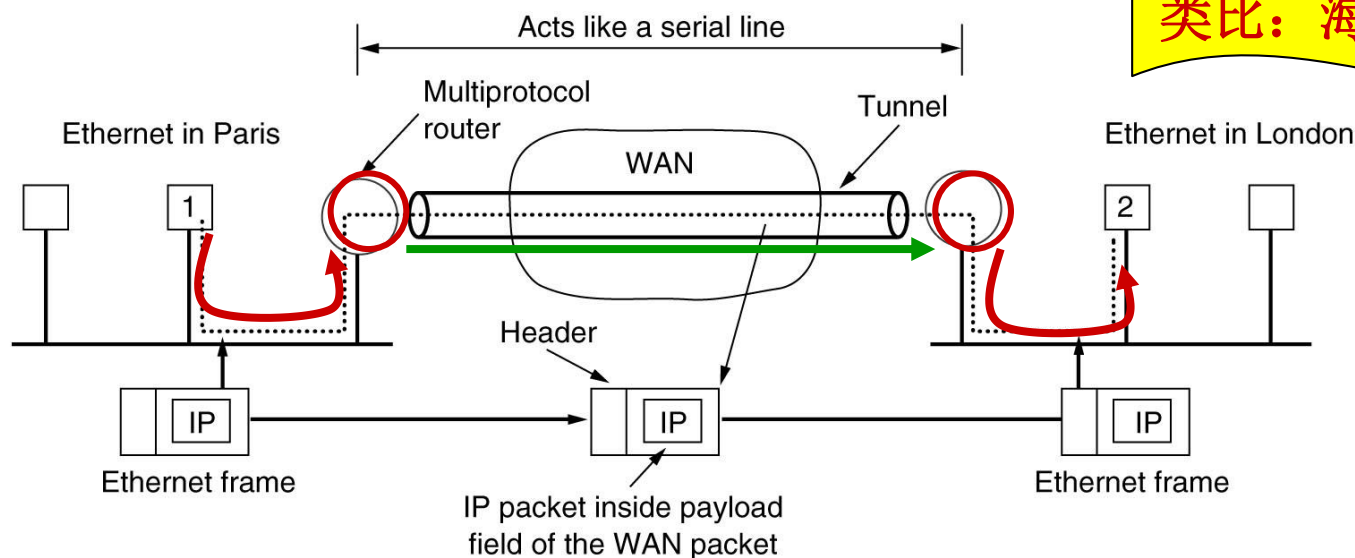




# 隧道技术 (5.5.5)

□ **隧道 (Tunneling)** --一种通用的特殊的网络互连方式

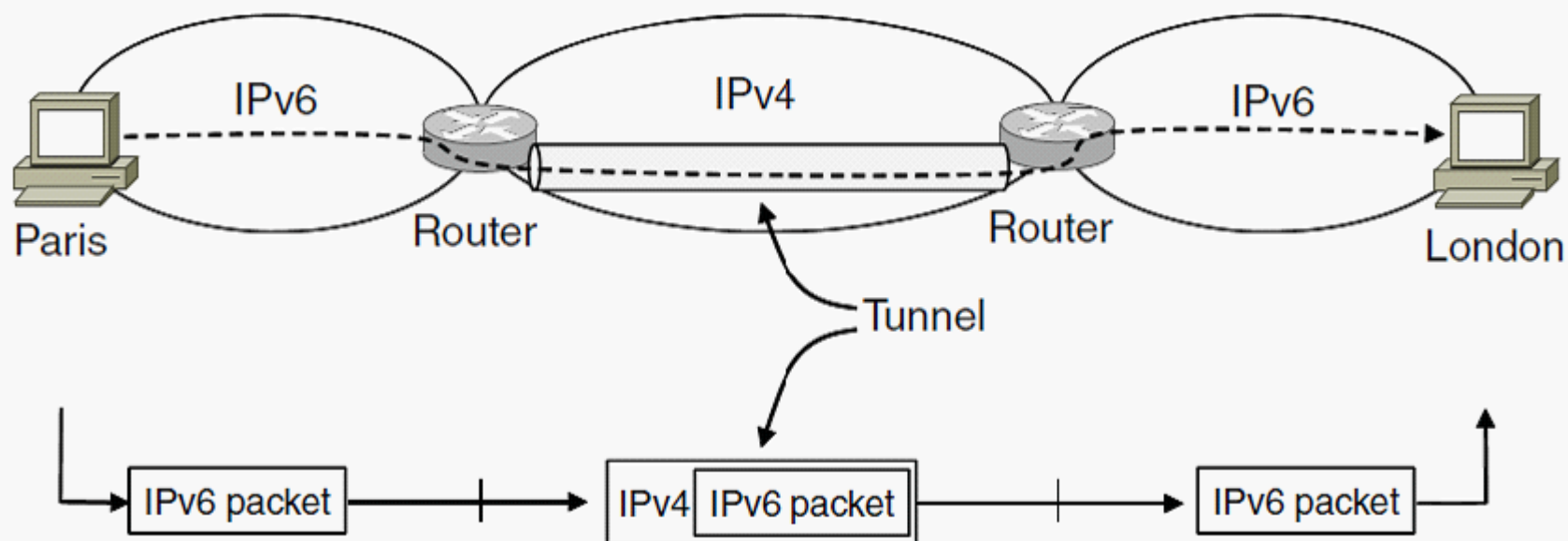
- 源和目的都处于同种网络，但是中经过不同类型的网络
- 中间的WAN部分可被看成一个大的隧道，从一个多协议路由器延伸到另一个多协议路由器



类比：海口海底火车



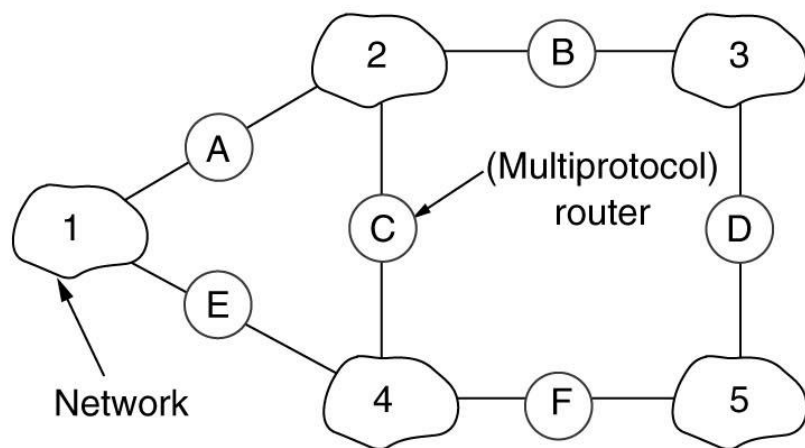
# 一个隧道的例子



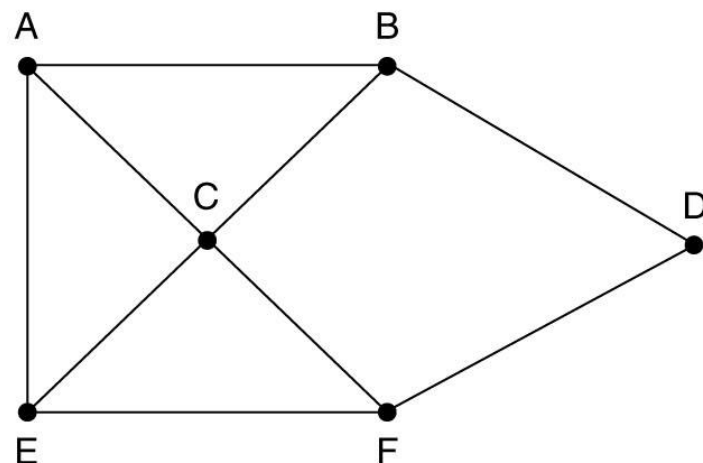
# 互联网路由 P332

- 互联网路由类似于单个子网内部的路由，但是，前者比后者更加复杂一些
- 两级路由算法
  - 每个网络内部采用内部网关协议（IGP, **interior gateway protocol**）
  - 网络之间使用外部网关协议（BGP, **exterior gateway protocol**）
  - 互联网上的每个网络都是独立于所有其他的网络，所以每个网络通常称作一个自治系统（**Autonomous System, AS**）
- 互联网路由和网络内部路由的差别是：
  - 互联网路由可能需要跨越国际边界，不同的法律可能会介入进来

# 互联网路由 (续)



(a)



(b)

# 分段 ( 5.5.7 P332 )

---

- 每个网络都会限制其分组的最大长度
  - 硬件 (例如, 受制于TDM 时隙)
  - 操作系统 (例如: 所有的缓存都是512字节)
  - 协议(例如: 分组长度域中的位数)
  - 遵从某个国家标准或国际标准
  - 期望因错误而重传的次数减少到某种程度
  - 期望防止分组占用的信道时间太长
- 所以, 网络设计者无法自由选择他们所期望的最大分组长度
- 最大净荷的长度从 48 字节 (ATM cells) 到 65,515 字节 (IP packets)不等

# 分段（续）

---

- 显然，当一个较大的分组想要通过一个最大分组长度较小的网络时，问题来了：一个大集装箱怎么放到一台自行车上来运输呢？
- 怎么办？——分段（**蚂蚁搬家**）
  - 分段（**Fragmentation**）是将一个分组切分成几个小的分组，已通过网络
  - 分割容易做到，问题是怎样复原，组装？
  - 两种不同的分段策略
    - 透明分段（**transparent**）
    - 非透明分段（**non-transparent**）

# 分段（续P362~363）

---

## □ 透明分段

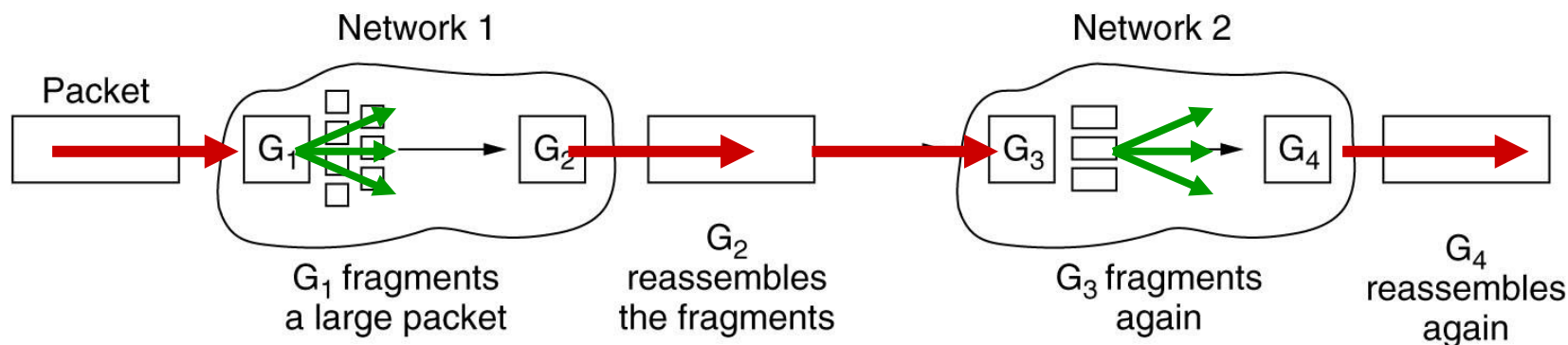
- 分段行为对其他网络来说是不可见的，换句话说，在该网络分段的分组，在离开这个网络的时候需要将它重组恢复

## □ 非透明分段

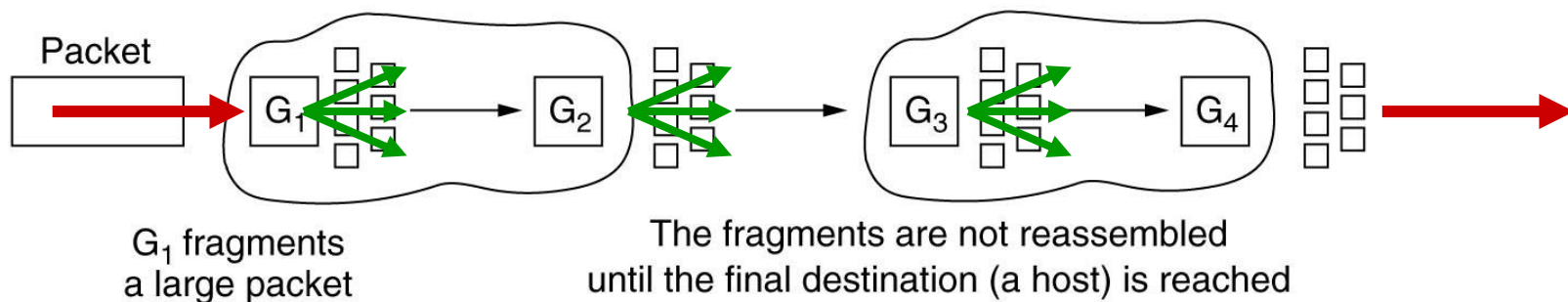
- 每个网络遇到不能承载的分组即对其进行分段，不负责重组恢复，**目的机**收到分割后的分组，完成重组恢复

# 分段 (续)

## 透明分段



(a)



(b)

# 分段遇到的问题

---

## □ 透明分段

- 出口的网关必须知道什么时候它收到了所有的分片
- 所有的分组都必须从同一个网关离开
- 进行分段和重组需要消耗资源，如果不断通过一系列的“小”网络，开销会很巨大

## □ 非透明分段

- 要求每个主机有重组功能
- 总开销增大，因为分片都需要头部信息

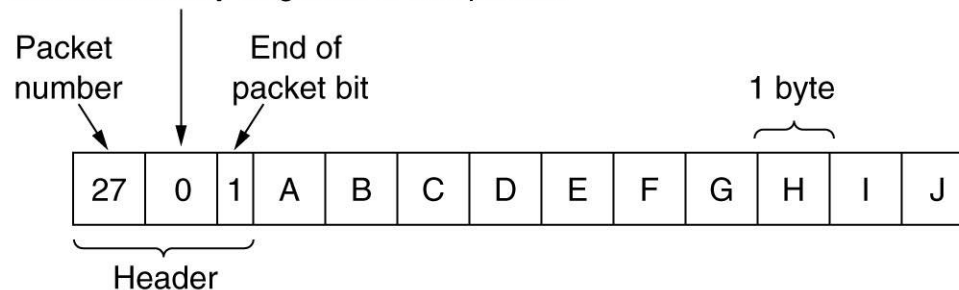


# 分段编号方法P333

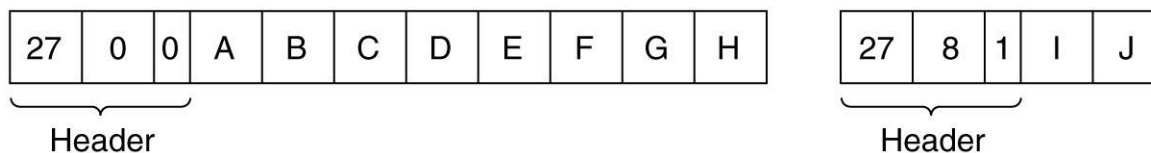
- 定义一个足够小的基本分片长度值，以便基本的分片能通过每一个网络（IPv6是这样做的，发前试探）
- 当原始分组被分割的时候，所有的分片的长度等于基本长度值，只除了最后一个分片（更短）
- 一个互联网分组可能包含多个分片，所以，在分片的头部必须提供这些信息：
  - 初始的分组号
  - 第一个基本分片的编号
  - 一个位（bit），表明该分片是否是原始分组分割后的最后一个分片

# 分段实例 P364

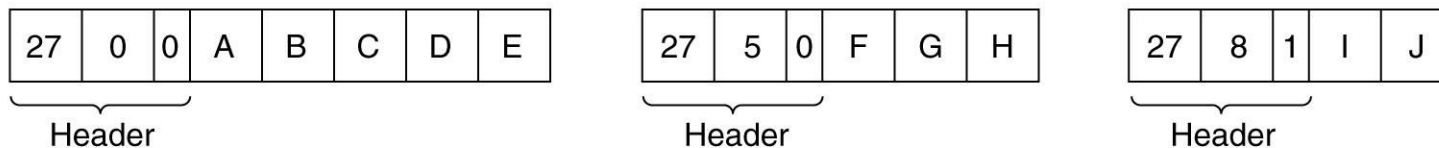
Number of the first elementary fragment in this packet



(a)



(b)



(c)

# 本节小结

---

## □ 拥塞控制

- 根源

- 方法

## □ 流量整形

- 漏桶Leaky bucket

- 令牌桶Token bucket

## □ 网络互联

---

# Thank you all!



