主要内容

- 网络资源管理
  - 网络层的拥塞控制
  - 队列调度机制
    - 资源分配–公平排队
    - 队列调度的典型算法
  - 流量监管算法
- 服务质量QoS（选）
  - 区分服务和集成服务
- Linux的流量控制TC（补充）
- 数据中心负载均衡技术: ECMP, MPTCP
- 大作业中期检查
- 完成小作业4

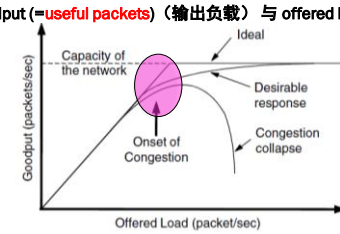
网络层的拥塞控制

拥塞控制

- 拥塞 congestion
  - 网络中存在太多数据包，导致数据包和延迟和丢失，从而降低传输性能
- 处理
  - 网络层
  - 传输层

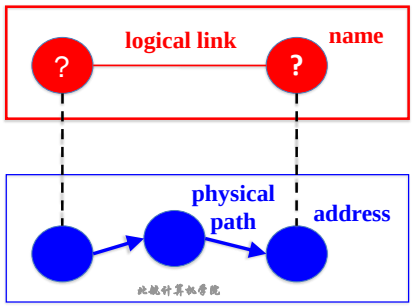
拥塞控制Congestion Control

- 拥塞产生的原因：Congestion results when too much traffic is offered; performance degrades due to loss丢包/retransmissions 重传
  - Goodput (=useful packets)（输出负载） 与 offered load（输入负载）|



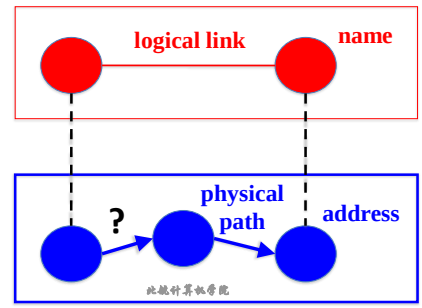
拥塞控制 Congestion Control

What can the *end-points* do to collectively to make good use of shared underlying resources?



队列管理Queue Management

What can the individual *links* do to make good use of shared underlying resources?



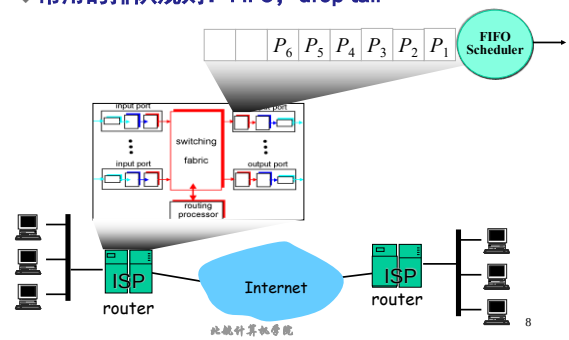
队列管理（Queue Management）

- **调度规则 Scheduling discipline**
  - **发送分组：Which packet to send?**
  - **参数：Some notion of fairness（公平性）？ Priority（优先级）？**
- **丢包策略 Drop policy**
  - **时间：When should you discard a packet?**
  - **丢弃分组：Which packet to discard?**
- **目标：平衡 吞吐量 throughput 和 时延delay**
  - **增加缓冲区是否可行?**

Huge buffers minimize drops, but add to queuing delay (thus *higher RTT, longer slow start, ...*)

路由器的队列管理

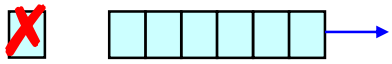
- **常用的排队规则：FIFO，drop-tail**



FIFO Scheduling and Drop-Tail

- "先入先出": Access to the bandwidth: FIFO(First-In First-Out queue)
  - Packets only differentiated when they arrive, 单队列

- "弃尾": Access to the buffer space: drop-tail queuing
  - If the queue is full, drop the incoming packet



Internet路由器上的实现

- FIFO + drop-tail
  - FIFO: scheduling discipline
  - Drop-tail: drop policy
- 存在问题
  - 主机端负责拥塞控制 (e.g., TCP)
  - 不区分不同的流（flows）
  - 无监管（No policing）: send more packets → get more service
  - 同步（Synchronization）: end hosts react to same events

网络层的拥塞处理

- 网络层方法
  - 流量感知路由 Traffic-aware routing
  - 准入控制 Admission control
  - 流量调节 Traffic throttling
  - 负载脱落 Load shedding

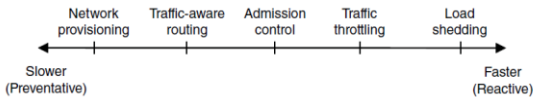
参考：Andrew S.Tanenbaum, Computer Networks，清华大学出版社，（第五版，《计算机网络》）

拥塞控制和流量控制

- 拥塞控制：全局性问题
  - 确保网络能够承载所有到达的流量
  - 主机和路由器
- 流量控制：点到点
  - 确保快速发送方不会持续以超过接收方接收能力的速率传递数据
- 例1：
  - 一个光纤网络100Gbps，一台超级计算机给一个PC传送大文件，PC所在网络1Gbps。
- 例2：
  - 网络线路是1Mbps，有1000台大型计算机，其中一半机器给另一半机器以100kbps的速率传送文件

拥塞控制的方法

- 方法
  - 增加资源
  - 减少负载
- 不同时间尺度

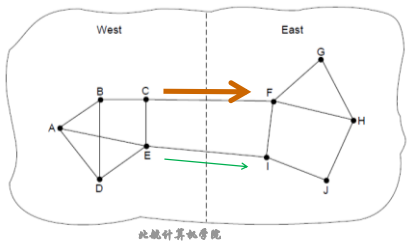


拥塞控制的方法

- 供给 provisioning
  - 建立与流量相匹配的网络：增加带宽，设备升级等
- 流量感知的路由
  - 根据流量模式定制路由
- 准入控制
  - 降低负载
- 负载脱落
  - 网络丢弃无法传递的数据包：良好策略

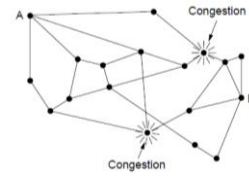
流量感知路由

- Traffic-Aware Routing
  - 根据链路流量进行路由选择，而不是拓扑结构
    - E.g., use *EI* for West-to-East traffic if *CF* is loaded
    - But take care to avoid oscillations（震荡）

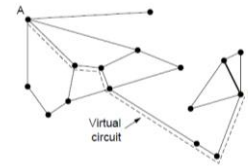


准入控制

- Admission Control
  - Admission control allows a new traffic load only if the network has sufficient capacity, e.g., with virtual circuits 虚电路



网络中有拥塞节点



A→B：选择无拥塞的链路

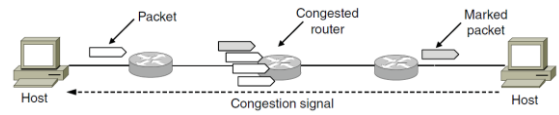
准入控制

需要解决的问题：

- **如何描述流量特性？**
  - 平均速率，突发性
- **如何建立虚电路？**
  - 预约机制: 资源预留
  - 估算网络承载能力
    - 网络过去行为的测量，捕获数据的统计特征
  - 和流量感知路由相结合

流量调节

- Traffic Throttling
  - 拥塞避免 congestion avoidance
  - 通知机制：Congested routers signal hosts to slow down traffic

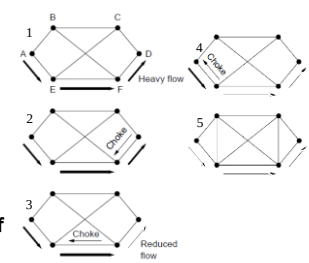


流量限制

- **路由器监测它正在使用的资源**
  - 输出线路利用率：突发流量 vs. 平滑流量
  - 在路由器缓存中排队的数据包
  - 由于没有足够缓存而丢弃的数据包数量
- **估算队列长度**
  - 指数加权移动平均（Exponentially Weighted Moving Average，EWMA）
- **反馈机制：如何通知发送方？**
  - 抑制包（choke packet）
  - 显式拥塞通知
    - ECN (Explicit Congestion Notification) marks packets and receiver returns signal to sender
  - 逐跳后压

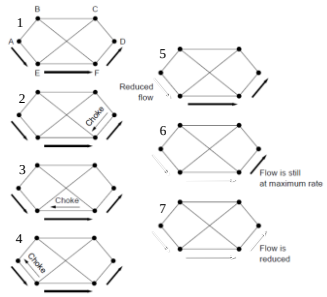
逐跳后压（1）

- Hop-by-Hop Backpressure
  - 端到端 end-to-end
  - 链路到链路 link-by-link
- 链路到链路 link-by-link（如图所示）
  - Link-by-link (right) produces rapid relief



逐跳后压（2）

- 端到端End-to-end
  - takes longer to have an effect, but can better target the cause of congestion



负载脱落

Load Shedding

- 问题：如何选择要丢弃的数据包？
  - 保留旧数据包（葡萄酒策略，旧比新好）：如文件传输
  - 保留新数据包（牛奶策略，新比旧好）：如流媒体
  - 与发送方协作（与应用类型相关，与数据包类型相关）
    - 数据包携带路由信息
    - 数据包包含全帧还是差异帧
- 随机早期检测 RED
  - Random Early Dectecion， Floyd and Jacobson， 1993

拥塞避免机制

- Random Early Detection (RED)
- Explicit Congestion Notification (ECN)

拥塞避免

- 拥塞避免(congestion avoidance)
  - 预测拥塞在何时发生，并在分组被丢弃之前降低主机 的发送速率
- 两种方法
  - 路由器为中心router-centric: 如RED Gateways
    - 隐含的，不需要路由器发出显式信号
    - 测量：丢包概率loss probability or 排队时延 queueing delay
  - 主机为中心host-centric: TCP Vegas，TCP BBR
    - TCP/AQM (active queue management)

Active Queue Management（AQM）

- 主动队列管理AQM
  - Design active router queue management to aid congestion control
- 修改路由器和主机
  - DECbit – congestion bit in packet header
- 修改路由器，主机仍采用TCP协议
  - 公平排队 Fair queuing
    - 为每个连接分配缓存
  - RED (Random Early Detection)
    - 在拥塞发生前丢包

Active Queue Management（AQM）

- 为什么在路由器上进行控制？
  - 全局视图：Router has unified view of queuing behavior
  - 队列参数：Routers see actual queue occupancy (distinguish queue delay and propagation delay)
  - 决策：Routers can decide on transient congestion, based on workload

随机早期检测RED：动机

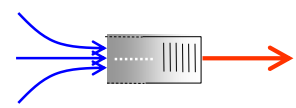
随机早期检测 RED (Random early detection)

- 1993，Sally Floyd和Van Jacobson提出

- 现象：突发流量消耗缓冲区，导致丢包
- TCP通过慢启动降低负载
  - 弃尾排队规则导致突发丢包
  - 重复发送丢失的分组，导致负载和延迟的进一步增加
  - 全局同步问题 Global synchronization
    - 大量TCP连接进入慢启动
    - 流量降低，网络利用率下降
    - 同时离开慢启动阶段之后，导致突发流量

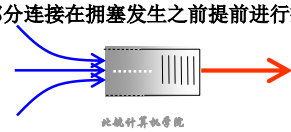
弃尾排队规则导致突发丢包

- TCP拥塞控制指标： 丢包packet loss
  - TCP拥塞控制机制-AIMD（加增乘减）： TCP additive increase drives network into loss
- 突发丢包：Drop-tail leads to *bursty* loss
  - *Congested link*: many packets encounter full queue
  - *Synchronization同步*: many connections lose packets at once



慢反馈问题

- 问题
  - 以缓冲区全满作为反馈指示
  - 而在缓冲区满之前，已经填充了一段时间
- 现象：分组RTT增加
  - 在填充过程中，RTT不断增加
  - RTT的方差增加
- 改进：尽早反馈
  - 使部分连接减速
  - 使部分连接在拥塞发生之前提前进行控制



Random Early Detection (RED)

- 随机早期检测（RED）
  - 有时也称为 Random Early Drop
- 队列调度：FIFO scheduling
- 缓冲区管理：替换 drop tail buffer management
  - 概率丢包：Probabilistically discard packets
  - Probability is computed as a function of average queue length
- 为什么使用平均队列长度？
  - Avoid over responsiveness to transient congestion

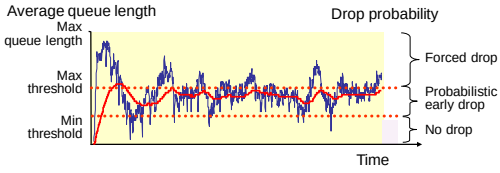
Random Early Detection (RED)

- 随机早期检测（RED）
  - 有时也称为 Random Early Drop
- 队列调度：FIFO scheduling
- 缓冲区管理：替换 drop tail buffer management
  - 概率丢包：Probabilistically discard packets
  - Probability is computed as a function of average queue length（平均队列长度）
- 为什么使用平均队列长度？
  - 避免对短暂性拥塞过度反应

RED的基本方法

- 监测队列长度：平均队列长度
  - Router notices that the queue is getting full（队列满）
  - ... and randomly drops packets to signal congestion
- 确定何时丢弃分组（丢包概率）
  - 丢包概率随队列长度增加而增加
  - 如果缓存低于一定水平，不丢包
  - 否则，设置丢包概率为队列长度的函数。
- 实现
  - 和主机控制算法相结合（TCP Reno）

RED 丢包策略



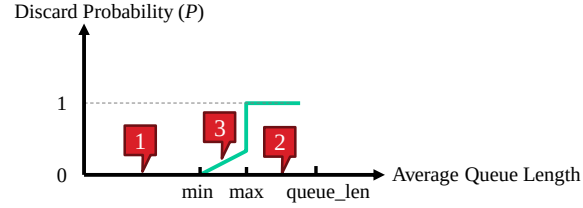
- 最小值min: minimum threshold
- 最大值max: maximum threshold
- avg_len: (weighted) average queue length
- 计算平均队列长度：

  avg_len = (1-w)*avg_len + w*sample_len （指数加权移动平均）

  W= 0.002

RED:丢包策略

1. If (avg_len < min_{th}) → enqueue packet 排队
2. If (avg_len > max_{th}) → drop packet 丢弃
3. If (avg_len >= min_{th} and avg_len < max_{th}) → enqueue packet with probability *P* （以概率 *P* 排队）

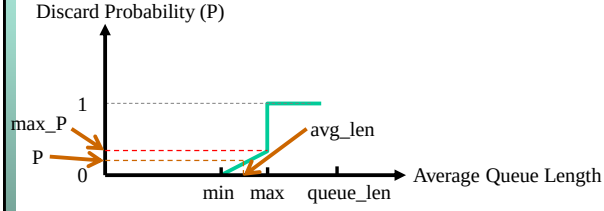


RED:丢包策略

- P = max_P×(avg_len – min)/(max- min)
- Improvements to spread the drops

  P' = P/(1 – count×P), 其中

  · Count：自最后一次丢包后，连续排入队列的分组数量



RED:丢包策略（续）

- 说明：P' 随count值的增加而缓慢增加，避免在单个连接上的多重丢包
- 分组丢弃避免成群发生
  - 由于来自某一个连接的分组到达可能是突发的，因此，成群丢弃分组可能集中在单个连接上
  - 每个RTT内，只丢弃一个分组就足以让连接减少窗口大小
  - 丢弃多个分组会让连接会退到慢启动
- 例： max_P=0.02，count=0， avg_len 位于两个阈值中间，则P， P'为 0.01，则到达分组99%进入队列
  - 随着队列长度增加，P缓慢增加
  - 丢弃大致随时间均匀分布

RED的特性

- 在缓冲区满之前进行丢包
  - 降低一些流（flow）的速率
- 丢包与每个流的速率成正比
  - 高速率的流有更多的分组，因此被选择的概率更高
- 丢包随时间变化
  - 避免TCP发送方的同步问题
- 容忍突发流量
  - 考虑平均队列长度和突发长度
- RED实现机制
  - Cisco, Juniper等路由器上实现RED算法

RED的问题

- RED的参数敏感性
  - 设置参数值：minth, maxth，maxp
  - 何时开始早期丢包？
  - 丢包概率的变化率如何确定？（斜率）
  - 平均队列长度计算的时间尺度？
- RED可能有副作用
  - RED 使用平均队列长度，可能引入较大的时延（ large feedback delay），导致不稳定性
- 参数设置不适当，RED算法性能下降
- 其他相关研究：Adaptive RED (ARED) – Linux 内核 3.3；BLUE, WRED (Cisco)
- 通常用在骨干路由器或核心网络上

拥塞控制研究进展

- 拥塞控制的目标
  - 最大化网络上瓶颈链路的吞吐量
  - 降低网络链路上缓存占用率，降低时延
- 基于AQM的方法如 RED等无法在吞吐量throughout与时延latency之间tradeoff
  - 时延降低，吞吐量也下降，同时伴随丢包
- 解决方法:TCP+AQM
  - DCTCP，数据中心网络中，通过*ECN进行显式拥塞控制*，优化流完成时间FCT等（需要设备支持ECN）
  - 优化TCP：google的TCP BBR算法（ linux内核4.9 ）
    - 主动发现：实时检测当前的传输带宽

Explicit Congestion Notification (ECN) [Floyd and Ramakrishnan 98]

- 传统机制
  - 丢弃分组：packet drop as implicit congestion signal to end systems
  - TCP will slow down
- 适用于块数据传输（ bulk data transfer）
- 不适合延迟敏感的应用
  - 交互式应用： Web, telnet

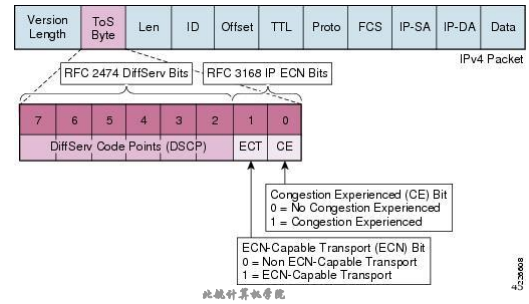
Explicit Congestion Notification (ECN)

显式拥塞通知

- ECN (RFC 3168):
  - 路由器标记：Routers mark packets instead of dropping them
  - 接收方确认：Receiver returns marks to sender in ACK packets
  - 发送方调整：Sender adjusts its window accordingly
- 利用IP头部的两位:
  - ECT: ECN-capable transport (set to 1)：源端能够处理拥塞通知
  - CE: congestion experienced (set to 1)：拥塞指示
- 源主机上运行的TCP将对ECN位的设置进行响应，响应方式与对被丢弃分组的响应一样

IPv4分组中的ECN字段

- IP TOS字段：第6位和第7位
  - 实现：需要路由器、发送方主机和接收方主机支持



TCP对ECN的支持

- ECN使用TCP头部来告知发送端网络正在经历拥塞
- TCP头部中预先定义的保留位
  - ECE(ECN Echo)：ECN响应标志被用来在TCP3次握手时表明一个TCP端是否具备ECN功能，并且表明接收到的TCP包的IP头部的ECN被设置为11。
  - CWR(Congestion Window Reduce)：拥塞窗口减少标志被发送主机设置,表明接收到了设置ECE标志的TCP包。发送端将通过降低发送窗口的大小来降低发送速率。

ECN 协议

- ECN protocol (repurposes 4 header bits)
  1. Sender marks "ECN-capable" when sending
  2. If router sees "ECN-capable" and congested, marks packet as "ECN congestion experienced"
  3. If receiver sees "congestion experienced", marks "ECN echo" flag in responses until congestion ACK'd
  4. If sender sees "ECN echo", reduces *cwnd* and marks "congestion window reduced" flag in next packet

DCTCP

- DCTCP是斯坦福和微软一起开发的一个使用RED和ECN的拥塞控制算法，可以有效的降低了缓存队列的占用
  - 充分利用现有的硬件资源，只修改软件
  - 数据中心网络，6000台服务器，采样150TB压缩数据（1个月）
- **现象**
  - Incast
  - Queue buildup
  - Buffer pressure

Incast

- Incast指的是这样一种现象
  - 1个client向N个server同时发送请求，client必须等待N个server的应答全部到达时才能进行下一步动作。N个服务器中的多个同时向client发送应答，多个同时到达的”应答”导致交换机缓存溢出，从而丢包。
  - 这样只有等server发生TCP重传超时才能让client继续。
  - 这个现象同时损害高吞吐量和低延迟。
  - 目前对于Incast的已有研究表明，降低TCP RTO的粒度是比较有效的方案，但这并没有解决所有问题。

Queue buildup

- 由于TCP发送流量的“贪婪性”，可以导致网络流量的大幅振荡，因而表现在交换机队列长度的大幅振荡。在队列长度增高时，会有导致两个副作用
  - small flow丢包产生incast
  - small flow在队列中延迟较长时间（在1Gb网络中是1ms vs 12ms的区别）

Buffer pressure

- 因为许多交换机上的缓存是在端口间共享的。因此，某端口上short flow很容易因为缺少缓存受到其他端口上的large flow的影响。

DCTCP实现

- 交换机
  - 在交换机发现队列长度超过某个阈值时，使用ECN中的CE标记通过的TCP segment。
  - 根据当前队列长度（instantaneous queue size）进行判断，快速响应交换机的队列长度变化
- 接收方
  - 只在对有CE标志的报文的ACK中设置ECE
  - 出现CE标志，或者CE标志消失时立即发送确认
  - 可以确切告知发送方有多少TCP流量（一个序号范围）几乎触发了“拥塞”，这个序号范围的大小标志了拥塞的程度。
- 发送方
  - 接收到的ECE的发送方按照“拥塞程度”缩小拥塞窗口

队列调度机制

Link Scheduling

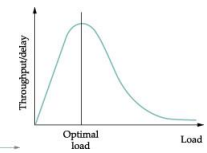
资源分配

- 资源分配机制
  - 路由器使用排队规则来决定如何缓存等待发送的分组
    - 分配带宽（哪些分组被传送）
    - 分配缓冲区（哪些分组被丢弃）
- 网络资源分配方案的有效性（effectively）
  - 尽可能大的吞吐量throughput
  - 尽可能小的时延delay

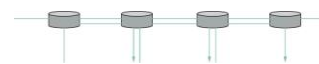
资源分配机制的评价标准

- 问题：增加吞吐量，避免链路空闲，就会增加路由器队列的长度，导致分组的延迟增加。
  - 能力曲线：Power of the network (M/M/1 queuing network)

Power = Throughput/Delay



- 公平性（fairly）
  - *Equal* share vs. *fair* share?
  - 例如：考虑流路径的长度



  - bandwidth shared in proportion to RTT（为什么？）

问题提出

现象：

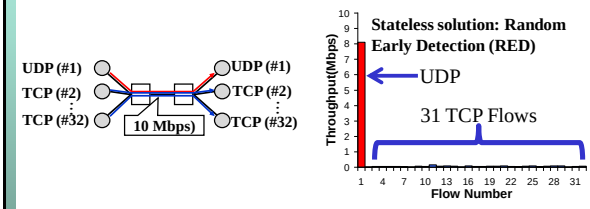
- 发送方最大化发送速率
  - In order to maximize its chances of success, a source has an incentive to maximize the rate at which it transmits.
- 路由器采用FIFO排队方法
  - a FIFO queue is "unfair" – it favors the most greedy flow.
- 难以保证分组的时延

问题：

- 公平性（Fairness）
- 时延保证（Delay Guarantee）

例：RED的问题

- 无保护机制的情况下：丢包的影响
  - Misbehaving flows can hurt other flows



调度机制

问题: 如何调度分组在链路上传输？

服务质量保证：如何避免一个流占用太多的资源，而其他流得不到资源?

典型的方法：

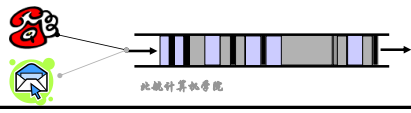
- 先进先出（ FIFO ）
- 基于优先级的调度
- 加权公平排队 (Weighted Fair Queueing ， WFQ)

典型的Internet排队规则

- FIFO + drop-tail
  - 简单，广泛用于Internet
- FIFO (first-in-first-out)
  - 单一类别流量（一个队列）
- Drop-tail
  - 丢弃排在队尾的包，不考虑流类别或优先级
- 区别:
  - FIFO: 调度规则（ scheduling discipline）
  - Drop-tail: 丢包策略（ drop policy）

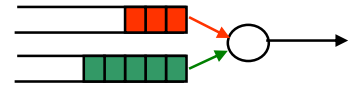
FIFO存在的问题

- FIFO(First-in first-out scheduling) 问题
  - 没有考虑优先级
  - 多个短分组排在长分组后面，使每个分组的平均时延增加。通常，长分组构成的流获得更好的服务
  - 一个“贪婪”的TCP流将挤掉其他退避的TCP流
- 例：两种类型的流量
  - VoIP： 需要低延迟； E-mail 对延迟不敏感
  - 排队： VoIP流在E-mail流后面等待



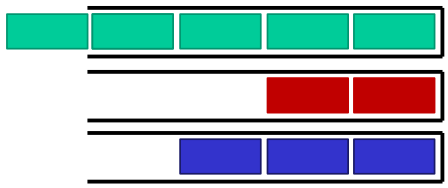
严格优先级调度

- 多种优先级设置
  - Always transmit high-priority traffic, when present
- 隔离高优先级流量
  - Almost like it has a dedicated link
  - Except for (small) delay for packet transmission
- 但低优先级流量可能无法获得传输 ☹



解决方法?

- 轮询：Round-robin among different flows
  - One queue per flow



- 保护和隔离
  - Provides protection/isolation between flows

轮询方法的问题

- 倾斜性：Biased towards flows with large packets

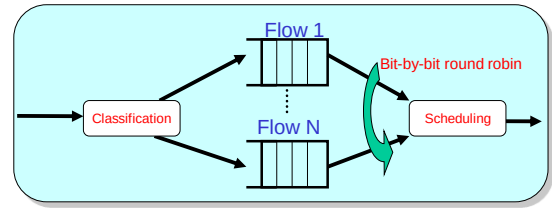
- 复杂性: requires per flow state in the router
  - Must keep track of number of flows
  - Must maintain flow->queue mapping
- 实际上，路由器不保存 per-flow 状态

公平排队（Fair Queueing）

- **公平排队**
  - 路由器为当前处理的每个流维护一个独立的队列
  - 每个入分组放入一个独立的队列，每个队列采用FIFO排队
  - 这些队列按照轮流方式被服务，空队列被跳过。
- **按位轮询是否可行？**
  - Bit-by-bit round-robin
  - 近似的方法

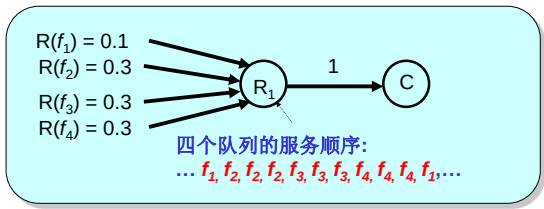
按位公平排队

1. 路由器采用轮询（round-robin）的方式调度这些队列中的每一位.
2. 按位公平排队（Bit-by-Bit Fair Queueing.）



加权按位公平排队

- 在每次轮询中，对于每个流发送不同的位数（给出权重），可以对不同流分配不同的数据率.
- 广义处理机共享 “Generalized Processor Sharing (GPS)”

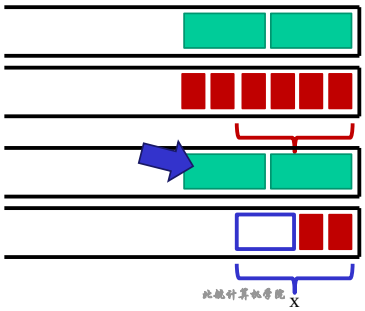


公平排队 Fair Queuing (FQ)

- **定义一个流体（fluid flow）系统**
  - a system in which flows are served bit-by-bit
- **处理分组**
  - 分组按截止期（deadlines）的增序接受服务
- **优点**
  - 每个流严格获得自己“公平”速率
- **FQ 可得到最大-最小公平性（max-min fairness）**

平衡公平性 Fairness 和利用率 Utilization

◆假设每个队列在每个RTT发送x 位



- Fair, and high utilization
- Fair, but capacity is wasted

Max-Min Fairness

◆最大-最小公平性

❖使得资源分配向量的最小分量的值最大

◆基本方法

❖公平分配给每个用户想要的可以满足的最小需求,然后将没有使用的资源均匀的分配给需要‘大资源’的用户

Max-Min Fairness

◆最大-最小公平性的定义

❖ 资源按照需求递增的顺序进行分配

❖ 不存在用户得到的资源超过自己的需求

❖ 未得到满足的用户等价分享资源

计算 Max-Min Fairness

◆符号集

❖*C* – 链路容量 link capacity

$$\sum_i \min(r_i, f) = C$$

❖*N* – 流的个数 number of flows

❖r_i – 每个流的需要带宽 desired bandwidth of flow *i*

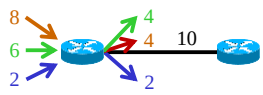
❖*f* – *公平分配带宽 fair share bandwidth*

◆目标：

❖ Each flow can receive at most the fair rate, i.e., $\min(r_i, f)$

Max-Min Fairness Example

◆$r_1 = 8, r_2 = 6, r_3 = 2$



C	N	C/N	Flows w/ r_i <= C/N
10	3	3.33	$r_3 = 2$

min(8, 4) = 4
min(6, 4) = 4
min(2, 4) = 2

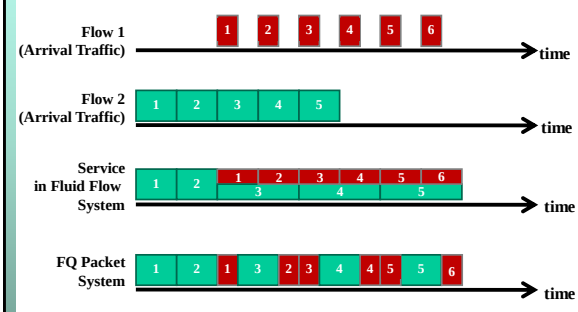
$C = C - \sum$(flows w/ r_i <= C/N)
$N = N$ - |flows w/ r_i <= C/N|
$f = C/N = 8/2 = 4$

FQ 的实现

◆理想的: 按位抢占式调度
- ❖ do preemptive scheduling between packets on bit-by-bit basis

◆实际的近似:
- ❖ serve packets in the order in which they would have finished transmission in the fluid flow system

FQ 的示例



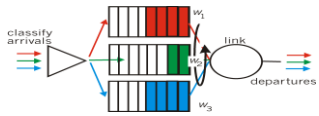
加权公平排队WFQ

◆Weighted fair scheduling（WFQ）
- ❖Assign each queue a fraction of the link bandwidth
- ❖Rotate across queues on a small time scale

50% red, 25% blue, 25% green

加权公平排队WFQ

- 每个队列i被分配一个权值w_i, 该队列在非空时，得到的带宽为 w_iC
  - C: link capacity； $\sum w_i = 1$
- 如果所有队列非空，每个队列得到的带宽为 w_iC;
  - 否则，空闲带宽按比例分配给非空的队列
    - “公平排队”



WFQ 实现

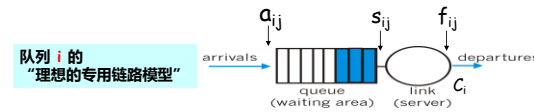
- 虚拟时间Virtual Time:
  - 队列 i 中的分组 j 到达时间为 a_{ij}； 分组大小为L_{ij}
  - 如果队列 i 获得的链路带宽为C_i，分组传输的完成时间为多少？

设：$s_{(ij)}$ 为队列i中分组j开始传输的时间，$f_{(ij)}$ 为传输完成的时间，

对于第 1 个分组，$s_{i1} = a_{i1}$; $f_{i1} = s_{i1} + L_{i1}/C_i$ 时间戳

对于第 j 个分组，$s_{ij} = max\{a_{ij}, f_{i,j-1}\}$; $f_{ij} = s_{ij} + L_{ij}/C_i$

若 $a_{ij} < f_{i,j-1}$， 则第 j 个分组进行排队（无抢占机制）



WFQ 实现（续）

- 不考虑“理想的专用链路”模型，当一些队列为空时，确定“spare capacity”的分布
  - 跟踪空队列和非空队列

设 $B(t) \subset \{1,2,...N\}$ 表示在 t 时刻非空队列的集合

则对于队列 $i \in B(t)$ 实际带宽为：

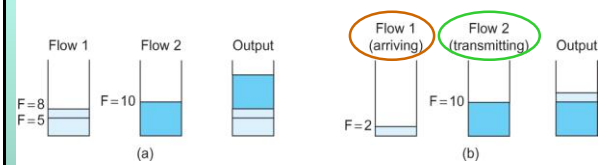
$$C_i(t) = \frac{w_i}{\sum_{k \in B(t)} w_k} C$$

修改“理想的专用链路”模型：

$s_{ij} = max\{a_{ij}, f_{i,j-1}\}$; $f_{ij} = s_{ij} + L_{ij}/C_i(t)$ （时间戳）

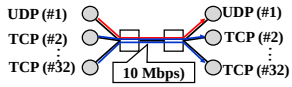
- 基于虚拟完成时间f_{ij}调度分组: 下一个被传输的分组是时间戳最小的分组
- 问题：计算 f_{ij}需要维护哪些状态？

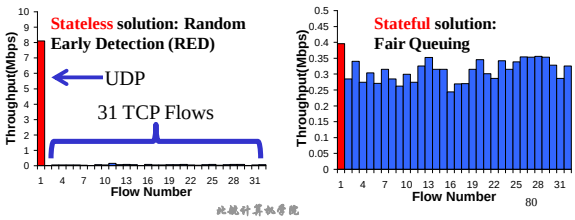
例1 实现公平排队



(a) 具有较早完成时间的分组首先被发送
(b) 正在发送中的分组继续完成

模拟示例





WFQ:总结

- WFQ 是工作保持模式 （work-conserving）
  - Send extra traffic from one queue if others are idle
  - Algorithms account for bytes, not packets
  - Results in higher utilization than non-work conserving
- 按字节（bytes）计算，而非分组（packets）
- WFQ 的公平性： max-min fairness
  - Maximize min rate of each flow
- WFQ的 权衡（Tradeoffs）
  - Complex state: Must keep queue per flow
  - Complex computation

比较

- FIFO
  - One queue, trivial scheduler
- Strict priority
  - One queue per priority level, simple scheduler
- WFQ(Weighted fair) scheduling
  - One queue per class（类别）, and more complex scheduler

流量监管典型算法

流量监管（Policing）

目标: limit traffic to not exceed declared parameters

通信量特性的描述方法

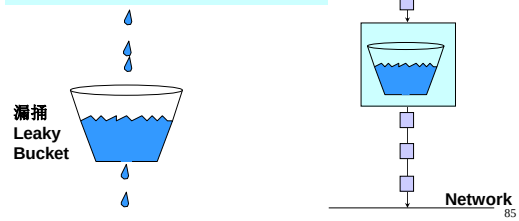
三个主要监管参数:

- *平均速率（Average Rate）:* 单位时间内发送的分组数量（长期）
  - 约束条件
    - 例如，时间间隔: 100分组/秒 比 6000分组/分钟的约束更严格
- *峰值速率（Peak Rate）:* 在较短时间内能够发送的分组数量
  - 例如，平均速率6000分组/分钟，峰值速率 1500 分组/秒
- *突发长度（Burst Size）:* 在极短时间间隔能够连续发送的分组数量

漏桶算法（Leaky Bucket）

监管主机的流量模式（服务器端）

- 在主机－网络接口，允许分组以恒定速率注入网络（滑动窗口？）
- 分组产生具有突发性，但通过漏桶后，变成均匀分组流。



漏桶算法说明

- 流量整形器“traffic shaper”
- 常数服务器时间的单服务器排队系统
- 每个主机连接到网络的接口中维持一个有限长度的内部队列。
  - 当分组到达时，被添加到队尾
  - 当队列满时，丢弃队尾的分组
- 参数
  - 输出速率 ρ
  - 桶容量 C

漏桶算法：例子

- 一台计算机可以25MBps的速率发送分组，假定每秒钟有一次40ms的突发数据.
  - 发送总数据量为1MB

问题：路由器缓冲区有限，最佳工作速率小于2MBps。如何设计漏桶算法？

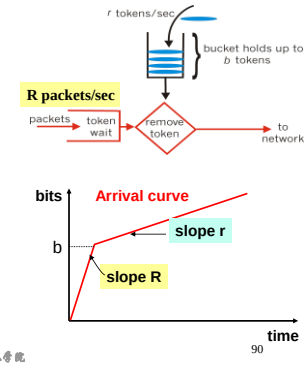
- 两个参数
  - 输出速率ρ = 2MBps
  - 桶容量C=1MB
- 均匀输出分组500ms

例：漏桶设计

- 漏桶容量
  - What should be the capacity of the leaky bucket to avoid loss?
- 例如，数据流突发
  - During the burst, data inflow is at the rate of 1.5 MB/msec and the outflow is at the rate of 0.06 MB/msec.
  - So accumulation is at the rate of 1.44 MB/msec. So at the end of 2 msec, there will be an accumulation of 2.88 MB.
- 避免溢出
  - This is the minimum leaky bucket capacity to avoid buffer overflow and hence data loss.

令牌桶算法（Token Bucket）

- 令牌以恒定速率产生，并放入令牌桶。
- 当分组到达时，若令牌桶中有令牌，分组被传输，否则，缓存分组，直到得到令牌。
- 令牌桶有一定容量，若满了，后续产生的令牌被丢弃。



令牌桶算法说明

- 描述分组流的流量特征：允许有限突发流量
  - 平均速率＋突发性（burstiness）：令牌桶过滤器
- 参数
  - 令牌速率：*r*
  - 桶容量：*b*
- 规则
  - 每个令牌表示发送一个字节的权利
  - 初始化时桶中没有令牌
  - 令牌积累的速率为：*r*
  - 令牌积累的总量不超过 *b*
- 在时间区间 T，能够发送到总数据量不超过：rT+b

例：令牌桶设计

- 令牌桶容量：b = 250KB
- 令牌速率： r = 2MBps
- 最大输出速率 M = 25MBps
- 假设有1MB突发流量达到时，令牌桶处于满的状态。那么，该令牌桶可以以25MBps的速率发送多长时间？然后回到2MBps的数据率，直到所有突发数据全部发送完毕。

提示： 时间 T内允许发送的总数据量＝rT+b

求解：MT= rT+b， 即：T=b/(M-r)

解得T约为11msec (发送的总数据量为：MT)

问题：

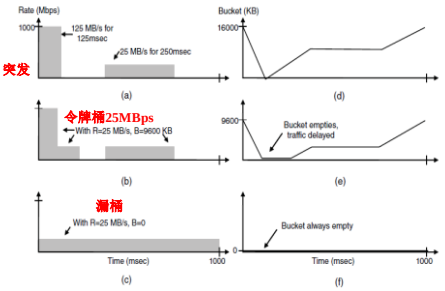
- 如何使流量更加平滑？
  - 令牌桶＋漏桶

例子（《计算机网络》第五版，Tanenbaum） P315

Host traffic
R=200 Mbps
B=16000 KB 突发数据

Shaped by
R=25 MBps
B=9600 KB

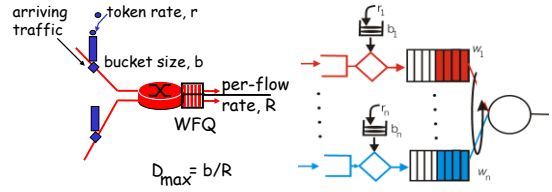
Shaped by
R=200 Mbps
B=0 KB



Smaller bucket size delays traffic and reduces burstiness

问题：队列的最大时延分析

◆令牌桶+WFQ队列调度规则：队列最大时延？



服务质量QoS

Quality of Service Guarantees

Internet的流量特点

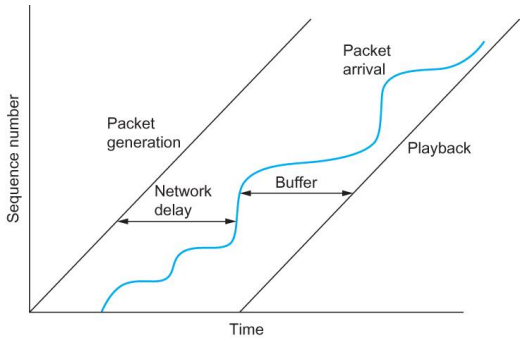
弹性流量

- 通信量可以随着时延和吞吐量 的变化进行调整
- 例如：
  - E-Mail – insensitive to delay changes
  - FTP – Sensitive to changes in throughput
  - SNMP – delay caused by congestion
  - Web (HTTP), TELNET – sensitive to delay
- 关注端到端总时延，而非每个分组的时延

非弹性流量

- 通信量不容易适应时延和吞吐量的变化
  - Real time traffic
- 吞吐量Throughput
  - Minimum may be required
- 时延Delay
  - 时延敏感的应用，如 stock trading
- 抖动Jitter - Delay variation
  - More jitter requires a bigger buffer
  - 例如， teleconferencing requires reasonable upper bound
- 丢包 Packet loss

实时应用实例:音频应用



非弹性（实时）流量的问题

- 在存在可变排队时延和拥塞的网络中，很难满足非弹性通信量的需求
- 需要优先处理
- 应用需求声明
  - 运行前或运行时
  - 使用IP头部字段
  - 资源预留协议RSVP（Resource reservation protocol）
- 同时支持弹性通信量
  - 可以拒绝过分占用资源的服务请求（给弹性流量预留资源）

服务质量（Quality of Service）

- QoS - network-level *guarantee* of quality of service
- 通常包括（考虑实时应用的需求）：
  - 时延 delay
  - 带宽bandwidth
  - 丢包率loss
- 其他需求（可靠性，延迟，抖动，带宽）
- 底层支持：IP协议
  - Don't want to switch to ATM architecture
  - Want to coexist with normal IP traffic
- 面向网络和服务提供商ISP

流（flow）

- Flow

  Clark,D.D, "The design philosophy of DARP Internet Protocols", 1988
- 面向连接的网络
  - 一个连接上的所有数据包
- 无连接网络中
  - 从一个进程发送到另一个进程的所有包
- 每个流的需求
  - 带宽，延时，抖动，丢包率等

QoS 需求

一些常见应用对服务质量（可靠性，时延，抖动，带宽）的需求（严格程度: high， medium，low）

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

"High" means a demanding requirement, e.g., low delay

数据流的分类

- ATM网络将数据流分为四大类
  - 位速率为常数的应用，如电话
  - 位速率可变的实时应用，如压缩的视频会议
  - 位速率可变的非实时应用，如IPTV
  - 最大可用位速率的应用，如FTP

Network Service	Application
Constant bit rate	Telephony
Real-time variable bit rate	Videoconferencing
Non-real-time variable bit rate	Streaming a movie
Available bit rate	File transfer

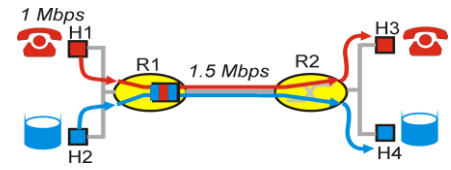
Example of QoS categories from ATM networks

QOS保障的典型方法

- 分类
  - 粗粒度：给多类数据块提供QoS
    - Differentiated Service 区分服务
  - 细粒度：给单独的应用或流提供QoS
    - Integrated Service集成服务；ATM中的QoS
- 主要方法
  - 流量标记： Packet marking
  - 流量监管：Traffic shaping / policing (delay guarantees)
  - 队列调度： Fair sharing (bandwidth guarantees)
  - 资源预留：Reservation systems

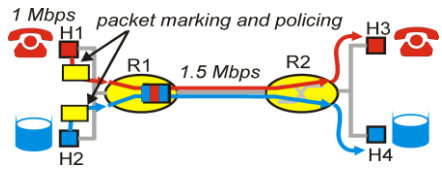
流量标记

- 问题：不同的应用竞争流量
  - E-mail traffic can cause congestion/losses for VoIP
- 方法1: Packet marking
  - So router can distinguish between classes
  - E.g., Type of Service (ToS) bits in IP header



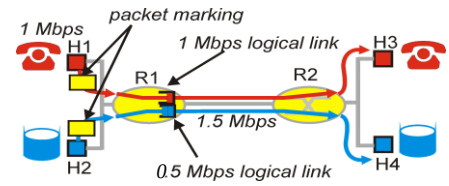
流量监管

- 问题：Applications misbehave
  - VoIP sends packets faster than 1 Mbps
- 方法2:监管traffic Policing
  - Protect one traffic class from another
  - By enforcing a rate limit on the traffic



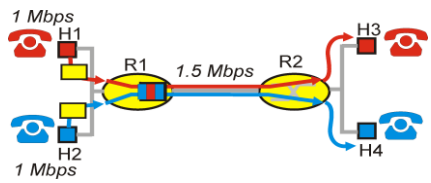
队列调度

- 方法3: Link scheduling
  - Ensure each application gets its share
  - ... while (optionally) using any extra bandwidth
  - E.g., weighted fair scheduling（WFQ）



资源预留

- 问题：Traffic cannot exceed link capacity
  - Deny access, rather than degrade performance
- 方法 4: Admission control 容许控制
  - Application declares its needs in advance
  - Application denied if insufficient resources available



服务质量QoS的目标

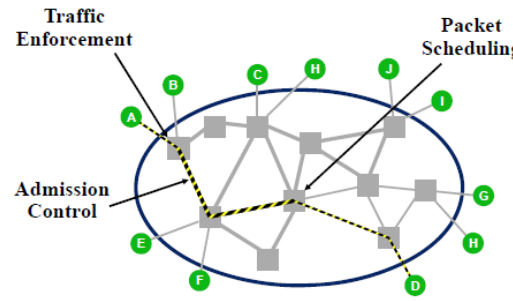
- 性能保障
  - Alternative to best-effort delivery model
- 相关协议和机制
  - Packet classification and marking
  - Traffic shaping
  - Link scheduling
  - Resource reservation and admission control
  - Identifying paths with sufficient resources

QoS体系结构 — 区分服务

Differentiated Services

（补充）

QoS框架



北航计算机学院

111

区分服务（Differentiated Services, Diffserv）

- 区分服务的方法（RFC2474,2475...）
- 粗粒度方法
- *Scalability:*在核心路由器设置简单的功能，在网络“边缘”实现更复杂的操作
- 不定义具体的服务类别，提供功能组件来构造服务
- 因特网的 ISP 和客户商定一个服务等级协定 SLA 。在 SLA 中指明了被支持的服务类别（可包括吞吐量、分组丢失率、时延和时延抖动、网络的可用性等）和每一类所容许的通信量。

北航计算机学院

112

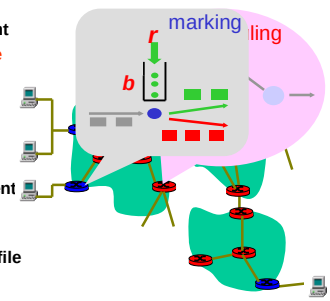
区分服务体系结构

边界路由器:

- per-flow traffic management
- marks packets as in-profile and out-profile

核心路由器:

- per class traffic management
- buffering and scheduling based on marking at edge
- preference given to in-profile packets
- Assured Forwarding



北航计算机学院

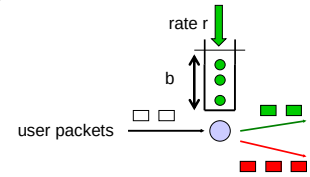
113

边界路由器功能

- 每跳行为PHB 规则
- 流量调节（traffic conditioning） to provide desired service
  - Classifier分类器
    - Separate packets into classes
  - Meter 测定器
    - Measure traffic for conformance to profile
  - Marker 标记器
    - Policing by remarking codepoints if required
  - Shaper 整形器
  - Dropper 丢包器

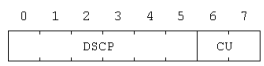
边界路由器标记分组

- 模板profile: pre-negotiated rate r, bucket size b
- packet marking at edge based on per-flow profile
- 类型
  - class-based marking: packets of different classes marked differently
  - intra-class marking: conforming portion of flow marked differently than non-conforming one



边界路由器标记分组（续）

- 分组标记
  - IP头部的TOS字段：Type of Service (TOS) in IPv4
    - 其中TOS的高3位表示IP优先级，高6位表示的DSCP值
  - Traffic Class in IPv6
- 区分服务代码点 Differentiated Service Code Point (DSCP)： 6bits【RFC2474】
  - 确定接收分组的PHB
  - 未使用（2bits）



DSCP: differentiated services codepoint
CU: currently unused

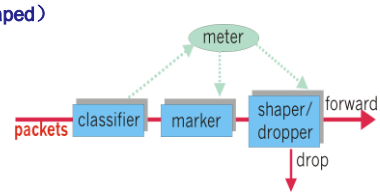
补充：标记方法

- RFC 791 IP ToS字段标记
  - IP头部的type of server字段
- RFC 1349 IP Precedence
  - IP头部中的TOS字段8bit，将其前三位定义为IP Precedence，将其4到7为定义为TOS，最后一位是MBZ（值必为零）
- RFC 2474 DSCP
  - IP头部的TOS字段的前6位定义为DSCP
- 802.1p CoS
  - 802.1Q头中保留了3位的user priority字段，用于标记帧的服务级别，802.1p作为802.1q的扩展，对tag字段中的priority字段（3b）定义，划分了8个服务等级。
- MPLS EXP
  - 在MPLS TAG中的tc字段，保留3位的EXP（实验）字段，用于实验性用途，该字段主要被应用于MPLS报文的类型进行标记

分类Classification 和调节Conditioning

限制一些类别流量的注入速率:

- 用户声明流量模板（profile）：
  - e.g., rate, burst size
- 流量测定（traffic metered）, 若违背，进行整形（shaped）



转发Forwarding (PHB)

- 逐跳行为Forwarding Per-hop Behavior (PHB)
  - 外部可观测的转发行为的描述
    - 产生不同性能、不同服务类别的流量
  - PHB具有单跳性和节点独立性的特点，每个节点具有独立的PHB策略，上下游之间没有影响
- PHB 没有描述保证所需PHB性能行为的特定机制
  - 可以使用不同资源分配策略
- 性能差别可测量

  例如:
  - Class A gets x% of outgoing link bandwidth over time intervals of a specified length
  - Class A packets leave first before packets from class B

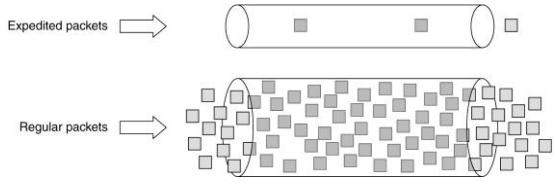
转发Forwarding (PHB) 分类

- Default PHB
  - 默认的PHB，提供尽力而为的服务
- Class Selector PHB
  - class selector代码点的值越高，其重要性和优先级就越高
- EF PHB：快速转发（Expedited Forwarding，EF）
  - 提供了低延迟、低抖动、低丢包率和保证带宽的优先转发服务
- AF PHB：确保转发（Assured Forwarding，AF）
  - 提供有保证的带宽服务

EF PHB —快速转发

- 快速转发（Expedited Forwarding） PHB，即 EF PHB, 或EF
  - 一个流量类中的分组离开路由器的速率等于或大于某个特定速率
  - 为一个类别提供具有最小保证的链路带宽的逻辑链路
  - Premium service奖赏服务 （RFC2598）
  - 实现：路由器每个输出线路有两个输出队列（常规类别分组和快速类别分组），采用不同的排队规则，如WFQ

EF PHB图例

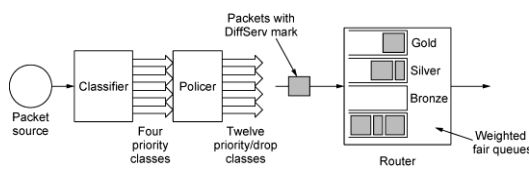


Expedited packets experience a traffic-free network

AF PHB

- 确保转发（Assured Forwarding） PHB，即AF PHB，或AF，（RFC2597）
  - AF用DSCP的比特0~2将通信量划分为四个等级，并给每一种等级提供最低数量的带宽和缓存空间。对于其中的每一个等级再用DSCP的比特3~5划分出三个“丢弃优先级”。共十二种服务类别
  - 每类中对应三个“丢弃优先级”中的一个。当发生网络拥塞时，对于每个等级的AF，路由器首先把“丢弃优先级”较高的分组丢弃。
  - 实现：分类器→标记器→整形器，前三个步骤在发送主机上执行后，输出到入口路由器上。这些功能可以由特殊的网络软件完成，不需要修改现有的应用系统

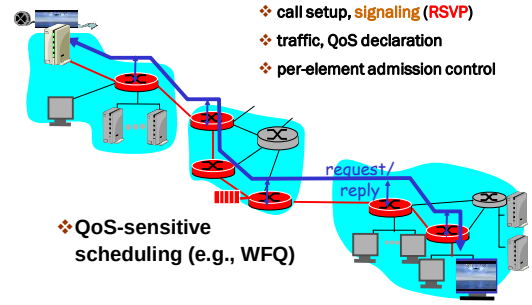
确保转发



A possible implementation of assured forwarding.

QoS 保证的应用场景

- 资源预留(Resource reservation)
  - call setup, signaling (RSVP)
  - traffic, QoS declaration
  - per-element admission control



- QoS-sensitive scheduling (e.g., WFQ)

IP数据流分类

- 按接口的信任模式
  - 设备信任某个接口接收到的数据包中的COS值，IP 优先级，DSCP值等，映射到内部DSCP值，并按默认映射方式对数据流进行操作
- 按接口的手工分类
  - 手工配置映射方式
- 按数据包（基于访问控制表ACL）
  - 将设备建立好的ACL和要建立的CLASS进行关联，并配置策略表进行控制
- NBAR(network based application recognition)基于网络应用的识别

区分服务的问题

- 如何提供端到端的区分服务？
- 如何进行监管和认证，以防止欺骗？
- 是否具有可观测的服务区别？
  - 目前，端到端的时延主要由接入速率和路由器的跳数造成，而不是路由器中排队时延造成。

体验质量QoE

- QoE（Quality of Experience）用户体验质量
  - 终端用户对应用或者服务整体的主观可接受程度（ITU-T）
- 与QoS的关系

表 1 –NP, QoS 和 QoE 特性总结

体验质量 (QoE)	服务质量 (QoS)	网络性能 (NP)
面向用户		面向供应商
用户行为属性	服务属性	连接/流量属性
关注用户预期的效果	关注用户可观测的影响	关注规划、开发(设计)、操作和维护
用户主体	在服务接入点之间（上）	端到端或网络部件的能力

- QoE涉及所有参与移动通信完整价值链的各个方面
  - 包括用户、运营商、内容提供商或应用提供商、设备制造商或系统集成商、终端设备和应用软件。

QoE的技术因素	QoE的非技术因素
端到端的QoS保证机制	用户主观行为和运营商服务质量
端到端的QoS保证机制	用户主观行为和运营商服务质量
端到端的业务质量KQI	业务的便利和快捷性
用户接通和传输能力（网络KPI）	服务内容
网络/服务覆盖情况	价格
终端功能性能	客服支撑
	用户耐受力和行为习惯

QoS体系结构 — 集成服务

Integrated Services

（补充）

集成服务（Integrated Services，Intserv）

- IETF [RFC2211, RFC2212]，1995-1997年
- 单播和组播
- 概念：IP网络上对每个应用会话提供QoS保障的体系结构，提供端到端的QoS保障
  - 细粒度方法
- 资源预留（resource reservation）
  - 路由器维护每个被分配资源和QoS请求的状态信息
- 呼叫建立（准入控制）
  - admit/deny new call setup requests

问题：如何确保有足够的资源满足新的会话的QoS需求，而不会与其他QoS冲突？

流(Flow)

- IP packet can be associated with a flow
  - 单个用户活动中的IP分组流
  - 相同的 QoS需求
    - E.g. one transport connection or one video stream
  - 单向性 Unidirectional
  - 多个接收方
    - Multicast
  - 流（flow）的组成
    - Source and destination IP address， port numbers, protocol type
    - 例如（src IP，dest IP，src port，dest port，protocol)
    - 或 （src prefix，dest prefix）
  - IPv6 header flow identifier can be used

流说明

- 流的通信量特性
  - T-spec: defines traffic characteristics
    - 如：采用令牌桶过滤器描述源端的带宽特性
    - 令牌桶速率，令牌桶大小，峰值数据率等
- QoS请求声明
  - R-spec: defines the QOS being requested
  - 如指定延迟目标或界限等

集成服务: 处理过程

- Flow 从网络中请求服务
  - 声明其QOS需求：R-spec
  - 流量特征描述： T-spec
- 路由器决定是否支持请求
  - admission control 准入控制
- 若支持，在路由器上基于流flow信息对流量分类和调度
  - 调度方法：WFQ or variants

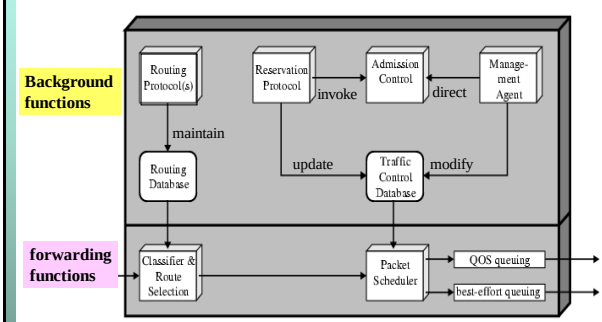
集成服务: 功能

- Admission control准入控制（许可控制，容许控制）
  - For QoS, reservation required for new flow
  - RSVP used
- Routing algorithm 路由算法
  - Decision based on QoS parameters
  - OSPF used
- Queuing discipline 排队规则
  - Take account of different flow requirements
- Discard policy 丢包策略
  - Manage congestion
  - Meet QoS

资源预留协议RSVP

- 运行在从源端到目的端的每一个设备上，为每一条流提供端到端的QoS保障
- 从第一跳开始用PATH消息逐跳进行资源请求，到达目的地后再用RESV消息反向逐跳进行资源预留。
- 提供确定的QoS保障，不会随网络状态的变化而影响
- 可以提供端到端的QoS服务
- 需要跟踪每一个流的状态，难以扩展

集成服务在路由器中的实现



- 分类：路由器和交换机对报文进行分类
  - 自动分类
    - 根据接口上配置的信任类型分类，包括信任端口优先级或者信任报文优先级
  - 手动分类
    - 通过acl对报文的ip地址，端口号，mac地址，入接口，协议类型，vlan号，cos，exp，ip precedence，dscp等进行分类
- 流量监管：
  - 监督进入网络的流量大小，对超出的流量进行处理
  - 用令牌桶技术进行流量控制，可以采取放行（pass）、丢弃（discard）、重标记（remark）等操作
  - 标记分组，流量整形，接口限速

集成服务类型

有保证服务（Guaranteed service）[RFC2212]

- 为分组提供排队时延的严格界限例如：源端的流量特征是令牌桶（参数为r，b），所需的服务为确定的分组传输速率R
- 如何实现？
  - WFQ排队
- 影响因素
  - 分组化：基于分组的调度
  - 链路峰值速率的限制
  - 分组传输时间的变化

受控负载（Controlled load service）

- Tightly approximates to best efforts under unloaded conditions
- 即：分组获得的服务如同经过一个轻负载的网络：无丢包，无排队时延
  - No upper bound on queuing delay
  - Very high percentage delivered
- 延迟自适应（delay-adaptive）的实时多媒体应用

集成服务局限性

- 缺乏可扩展性（Scalability）:状态信息的数量与流的数目成正比。因此在大型网络中，按每个流进行资源预留会产生很大的开销。
- 缺乏灵活的服务模型: 集成服务所定义的服务质量等级数量太少，不够灵活。
- 集成服务体系结构复杂。若要得到有保证的服务，所有的路由器都必须装有 RSVP、准入（容许）控制、分类器和调度器。

资源预留协议RSVP

（补充）

Signaling in the Internet

connectionless (stateless) forwarding by IP routers + best effort service = no network signaling protocols in initial IP design

- 新的需求: reserve resources along end-to-end path (end system, routers) for QoS for multimedia applications
- RSVP: Resource Reservation Protocol [RFC 2205]
  - " … allow users to communicate requirements to network in robust and efficient way." i.e., signaling !
- earlier Internet Signaling protocol: ST-II [RFC 1819]

RSVP 特点 (rfc2205)

- 支持单播和多播（Unicast and Multicast）
- 单工方式Simplex
  - Unidirectional data flow
  - Separate reservations in two directions
- 接收方发起
  - Receiver knows which subset of source transmissions it wants
- 软状态维护
  - Responsibility of end users
- 提供不同的资源预留模式
  - Users specify how reservations for groups are aggregated
- 支持IPv4 (ToS field)和 IPv6 (Flow label field)

RSVP的操作

- 发送方和接收方加入多播组
  - done outside of RSVP
  - senders need not join group
- sender-to-network signaling
  - *path message:* make sender presence known to routers
  - path teardown: delete sender's path state from routers
- receiver-to-network signaling
  - *reservation message:* reserve resources from sender(s) to receiver
  - reservation teardown: remove receiver reservations
- network-to-end-system signaling
  - path error
  - reservation error

RSVP 操作说明

- 发送方: sends PATH message via the data delivery path
  - Set up the path state each router including the address of previous hop （timer：30s）
- 接收方: sends RESV message on the reverse path
  - Specifies the reservation style, QoS desired (RSpec)
  - Set up the reservation state at each router
- 注意：
  - 接收方发起资源预留请求
  - 资源预留与路由分离

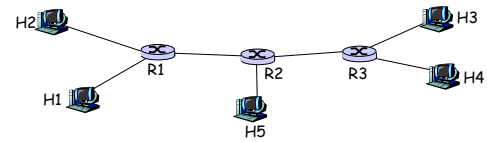
PATH 消息

sender-to-network signaling

- 消息内容:
  - *address:* 单播目的地址，或多播组地址
  - *flowspec:* TSpec，如带宽需求描述等
  - *filter flag:* 源端过滤分组（上行路径）
  - *previous hop:* 上行路由器ID
  - *refresh time:* 刷新时间
- path 消息: 包括源端信息，以及“reverse-path-to-sender”信息
  - later upstream forwarding of receiver reservations

例子: 简单的电话会议

- 如图所示
  - H1, H2, H3, H4, H5 既是发送方，也是接收方
  - 多播组： m1
  - 无过滤器：任意发送方的分组都被转发
  - 带宽: *b*
  - 只有一个多播路由树（组共享树）



RESV消息:

- RESV消息内容（ receiver-to-network signaling ）:
  - *需要预留的带宽*
  - *过滤器类型:*
    - no (wildcard) filter: 任何分组可以使用预留资源发送到多播组
    - fixed filter: 只有来自特定发送方集合的分组可以使用预留资源
    - dynamic filter: 发送方集合可以随时间进行动态改变.
  - *过滤器描述*
- RESV消息:从接收方逆向流到发送方（receiver-to-senders ）, 预留资源, 在路由器上建立与接收方相关的状态。

Linux的流量控制TC

（课外补充）

Linux的流量控制TC

- Linux内核实现相关的流量控制功能
  - 流量控制器TC（Traffic Control）用于Linux内核的流量控制，在输出端口建立处理数据包的队列，并定义队列中的数据包被发送方式
- 流量控制的主要步骤
  - 为网卡配置一个队列
  - 在该队列上建立分类
  - 根据需要建立子队列和子分类
  - 为每个分类建立过滤器
- 可用于搭建高性能Linux网关
  - 使用linux的两个工具实现QoS： TC 和 Iptables

TC的流量控制流程

- Linux流量控制TC主要是在输出接口排队时进行处理和实现的
- 接收包从输入接口（Input Interface）进来后，经过流量监管（Ingress Policing）丢弃不符合规定的数据包
- 由输入多路分配器（Input De-Multiplexing）进行判断选择：如果接收包的目的是本主机，那么将该包送给上层处理；否则需要进行转发，将接收包交到转发块（Forwarding Block）处理。
- 转发块同时也接收本主机上层（TCP、UDP等）产生的包。转发块通过查看路由表，决定所处理包的下一跳。
- 然后，对包进行排队以便将它们传送到输出接口（Output Interface）。
- 一般我们只能限制网卡发送的数据包，不能限制网卡接收的数据包，所以我们可以通过改变发送次序来控制传输速率。

Traffic Control的功能组成

- Qdiscs (queuing disciplines)
  - 排队规则, 决定进入该队列的数据包的流量控制
  - 队列分为两类：无分类CLASSLESS QDISC和 可分类CLASSFUL QDISC
- Classes (within a queuing discipline): 分类器
  - 对设备的流量控制的进行分类，设置不同种类的流量控制策略
  - Class必须属于可分类队列, 不同的Class下可以挂载其它队列，或作为终结点，什么都不做。
- Filters: 过滤器
  - 对网络上的数据包，根据需求的不同对不同的数据包设置不同的流量控制策略。

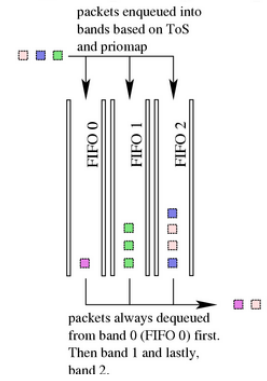
排队规则

- qdisc（排队规则），queueing discipline
  - 内核如果需要通过某个网络接口发送数据包，它都需要按照为这个接口配置的qdisc（排队规则）把数据包加入队列。
  - 内核从qdisc里面取出数据包，把它们交给网络适配器驱动模块。
- linux默认使用的是fifo_fast
- 无分类调度算法（classless qdisc）
  - choke, codel, p/bfifo,fq, fq_codel, gred, hhf, ingress,mqprio, multiq, netem, pfifo_fast, pie, red, rr, sfb, sfq, tbf。
- 可分类调度算法（classful qdisc）
  - ATM, CBQ, DRR, DSMARK, HFSC, HTB, PRIO, QFQ

排队规则（续1）

- 无分类QDISC（CLASSLESS QDISC）
  - [p | b]fifo,最简单的排队规则，先进先出。只有一个参数：limit，用来设置队列的长度，pfifo是以数据包的个数为单位；bfifo是以字节数为单位。
  - pfifo_fast，系统的标准qdisc(排队规则)，它的队列包括三个通道（band）。在每个通道里面，使用FIFO规则。而三个通道（band）的优先级也不相同，band 0 的优先级最高，band 2 的最低。
  - red，Random Early Detection（随机早期探测）
  - sfq，Stochastic Fairness Queueing（随机公平排队）
  - tbf，Token Bucket Filter（令牌桶过滤器）

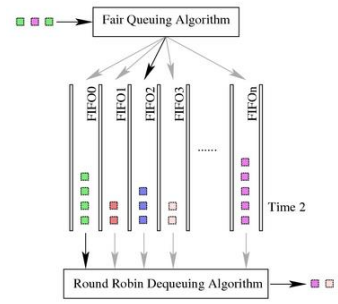
pfifo_fast queuing discipline



随机公平排队SFQ

- SFQ 使用随机算法尽力让每一个流（flow） 都能公平地将自己的数据发送出去。
- SFQ 会使用一个**散列函数**来完成随机分配，根据散列函数的值将数据包分配到由它自己维护的各 FIFO 队列中，并循环发送这些 FIFO 队列中的数据，以此实现随机发送。
- 由于一直使用固定的散列算法可能会造成不公平的情况，所以散列算法每隔一段固定的时间就会更改一次，用户可以修改 perturb 参数的值来控制这一时间间隔
- 存在问题
  - 一些客户端（如 Kazaa，eMule 等）会开启大量 TCP 会话占用更多带宽

Stochastic Fair Queuing (SFQ)



排队规则（续2）

- **可分类 QDISC (CLASSFUL QDISC)**
  - **CBQ，Class Based Queueing（基于类别排队）**
    - **限制（shaping）带宽，管理带宽优先级别**
  - **HTB，Hierarchy Token Bucket（分层令牌桶）**
    - **保证每个类别的带宽，允许特定的类可以租用其他类的带宽。**
    - **可以通过TBF（Token Bucket Filter）实现带宽限制，也能够划分类别的优先级。**
  - **PRIO，PRIO qdisc 不能限制带宽，因为属于不同类别的数据包是顺序离队的。**
    - **使用PRIO qdisc 可以很容易对流量进行优先级管理，只有属于高优先级类别的数据包全部发送完毕，才能发送属于低优先级类别的数据包。**
    - **为了方便管理，需要使用Iptables 或者 Ipchains 处理数据包的服务类型（Type Of Service，TOS）**

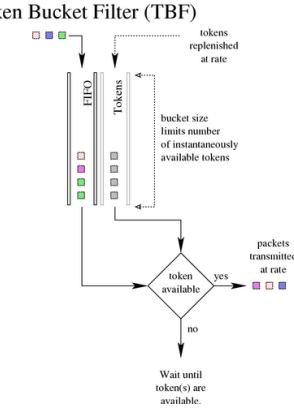
Linux 流量控制算法:SFQ

- SFQ(Stochastic Fairness Queueing 随机公平队列)
  - 流量被分到多个的 FIFO 队列中，每个队列对应一个 TCP会话或者 UDP 流。分组按简单轮询照方式发送，每个会话都按顺序得到发送机会。
  - “随机”：使用散列算法把所有的会话映射到有限的几个队列中去 。多个会话共享发送机共享带宽
  - SFQ 可以改变散列算法，降低冲突，把这种效应控制在几秒钟之内 (时间由参数设定)

Linux 流量控制算法:TBF

- 令牌桶过滤器 (TBF) ：只允许以不超过事先设定的速率到来的数据包通过，但可能允许短暂突发流量超过设定值
  - 对于网络和处理器的影响都很小，实现是针对数据的字节数进行的，而不是针对数据包进行。常用于网关限速 .
  - 算法关联到两个流上：令牌流和数据流

Token Bucket Filter (TBF)



TC中的队列标识

- **在TC中，使用"major:minor"句柄标识队列和类别，其中major和minor都是数字**
  - **队列：minor为0，即表示为“major:0”，可以简写为"major: “**
    - **例如，队列1:0可以简写为1:。需要注意的是，major在一个网卡的所有队列中必须是惟一的。**
  - **类别：其major必须和它的父类别或父队列的major相同，而minor在一个队列内部则必须是惟一的**
    - **例如，如果队列2:包含两个类别，可表示为2:1和2:2。**

linux下的网络带宽限速

- 建立队列
  - 针对网络物理设备（如以太网卡eth0）绑定一个队列qdisc

tc qdisc add dev ens33 root handle 1: cbq bandwidth 100kbit avpkt 1000 cell 8 mpu 64

说明:

该命令的含义是：将一个cbq队列绑定到网络物理设备ens33上，其编号为1:0；网络物理设备ens33的实际带宽为100kbit，包的平均大小为1000字节；包间隔发送单元的大小为8字节，最小传输包大小为64字节。

linux下的网络带宽限速（续1）

- **建立分类：在该队列上建立分类class**

tc class add dev ens33 parent 1:0 classid 1:1 cbq bandwidth 100kbit rate 100kbit maxburst 20 allot 1514 prio 8 avpkt 1000 cell 8 weight 10kbit

说明：

创建根分类1:1，该队列的最大可用带宽为100kbit，实际分配的带宽为100kbit，可接收冲突的发送最长包数目为20字节，最大传输单元加MAC头的大小为1514字节，优先级别为8，包的平均大小为1000字节，包间隔发送单元的大小为8字节，相当于实际带宽的加权速率为10kbit。

tc class add dev ens33 parent 1:1 classid 1:2 cbq bandwidth 100kbit rate 80kbit maxburst 20 allot 1514 prio 2 avpkt 1000 cell 8 weight 8kbit split 1:0 bounded

说明：

创建分类1:2，其父类为1:1，该队列的最大可用带宽为10kbit，实际分配的带宽为80kbit，可接收冲突的发送最长包数目为20字节,最大传输单元加MAC头的大小为1514字节，优先级别为2，包的平均大小为1000字节，包间隔发送单元的大小为8字节，相当于实际带宽的加权速率为8kbit，分类的分离点为1:0，且不可借用未使用带宽。

linux下的网络带宽限速（续2）

◆**建立过滤器**

- ❖ **为每一分类建立一个基于路由的过滤器filter**
- ❖ **与过滤器相配合，建立特定的路由表**

tc filter add dev ens33 parent 1:0 protocol ip prio 100 route

应用路由分类器到cbq队列的根，父分类编号为1:0；过滤协议为ip，优先级别为100，过滤器为基于路由表。

tc filter add dev ens33 parent 1:0 protocol ip prio 100 route to 2 flowid 1:2

建立路由映射分类 1:2。

◆**建立路由**

ip route add 119.0.0.0/8 dev ens33 via 192.168.17.2 realm 2

发往子网119.0.0.0/8的数据包通过分类2转发（分类2的速率80kbit），限速部分上传流的ip。

例2 限制应用带宽

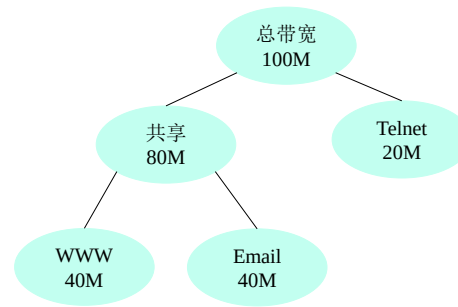
◆ **上网流量限流**

◆ **Linux NAT 网关，并对数据包进行限制**

◆ **Linux服务器限速**

◆ **限制应用带宽**

- ❖ **例如：假设eth0出口有100mbit/s的带宽，分配给WWW、E-mail和Telnet三种数据流量，其中分配给WWW的带宽为40Mbit/s，分配给Email的带宽为40Mbit/s，分配给Telnet的带宽为20Mbit/S。**



例2 限制应用带宽（续）

- 三种数据流WWW、E-mail和Telnet，其中的Telnet独立分配20Mbit/s的带宽，WWW和SMTP各自分配40Mbit/s的带宽。同时，它们又是共享的关系，即它们可以互相借用带宽
- 需要的TC命令如下:

```
#tc qdisc add dev eth0 root handle 1: htb default 21
#tc class add dev eth0 partent 1: classid 1:1 htb rate 20mbit ceil 20mbit
#tc class add dev eth0 parent 1: classid 1:2 htb rate 80mbit ceil 80mbit
#tc class add dev eth0 parent 1: classid 1:21 htb rate 40mbit ceil 20mbit
#tc class add dev eth0 parent 1:2 classid 1:22 htb rate 40mbit ceil 80mbit
#tc filter add dev eth0 protocol parent 10 prio 1 u32 match ip dport 80 0xffff flowid 1:21
#tc filter add dev eth0 protocol parent 1:0 prio 1 u32 match ip dport 25 0xffff flowid 1:22
#tc filter add dev eth0 protocol parent 1:0 prio 1 u32 match ip dport 23 0xffff flowid 1:1
```


完成小作业（4）

- **专题4“拥塞控制”**

1.任意选择1篇论文进行阅读

2.每人独立完成论文评论（paper review），评论内容要求：

- 作者主要观点和要解决的问题
- 研究方法评论（关键技术，优点和局限性）
- 论文的主要贡献
- 其他
- 注意：不是翻译，篇幅不限

3.作业提交（两个文档）

- .docx文件
- .pptx文件（约 10 页左右，请勿超过15页，课堂讨论用）