主动队列管理
—AQM

华中地区网络中心　李之棠

2.10 主动队列管理：AQM

2.10.1 什么是数据流

- 流是在源目主机对之间，通过相同路由而发送的**一系列包**，在路由器中的资源分配上下文中这是一个重要的抽象
- 流过同一路由器，且具有若干相同头部特征的一系列数据包。这些特征可是:源地址、目地址、源端口号、目端口号、协议类型、服务类型TOS、输入端逻辑接口等
- “通道”是端-端联通性的**一种抽象**，“流”对网络路由器是**可见的**。

2.10.2 路由器中面向TCP流分类

- TCP-friendly流
  - 采用了拥塞控制机制的流，即同质或大体上在相似环境下按可比条件（丢包率、RTT、MTU）不会占用比TCP流更多带宽的流
- 非适应（unresponsive）流
  - 不采用拥塞控制机制，不能对拥塞作出反应的流，如**许多多媒体应用和组播应用、UDP、AH、ES 等**
- 有适应性但非TCP-friendly流
  - 使用拥塞控制算法，但非TCP友好。如接收端驱动分层组播（Receiver-driven Layered Multicast RLM）采用的就是这种算法。
  - 使用non-TCP的拥塞控制算法。如修改TCP，使窗口的初始值很大并且保持不变，即所谓的"快速TCP"。

2.10.3 什么是拥塞和拥塞控制？

- 什么是拥塞？
  - 对聚合带宽的需求超过了链路的可用容量
- 拥塞后会发生什么？
  - 性能下降
    - 多个包丢失
    - 链路利用率低下（低吞吐率）
    - 高排队延迟
    - 拥塞崩溃

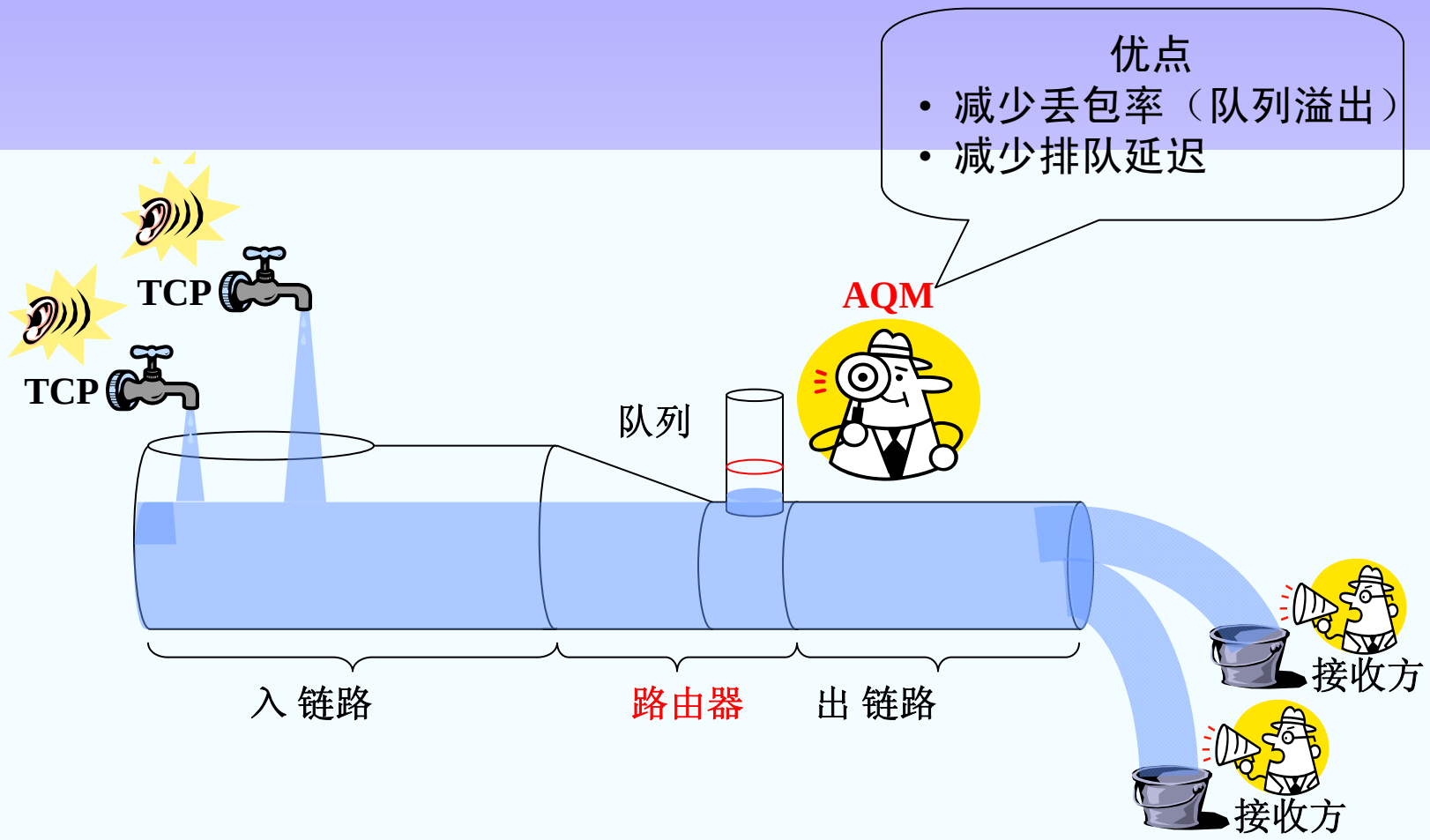
什么是拥塞控制？

- 有2种方法处理拥塞
  - 拥塞避免（提前反应）
    - 在网络过载之前动作
  - 拥塞控制（过后反应）
    - 在网络过载后动作
- 开环控制：主要在电路交换网络（GMPLS）
- 闭环控制：主要用在包交换网路，通过反馈信息：全局&局部
  - 隐式反馈控制
    - 端到端拥塞控制：如TCP:Tahoe（BSD网络1.0版）Reno(BSD网络2.0版：有快速恢复),Vegas.etc)
  - 显式反馈控制
    - 网络援助拥塞控制，如IBM SNA,DECbit,ATM ABR,ICMP source quench,RED,ECN

拥控与流控的区别

- 路由器中拥塞控制的必要性
  - 不遵守TCP拥塞控制机制的应用进一步加剧因特网范围内拥塞崩溃的可能性
  - TCP拥塞控制还存在着自相似、效率、公平性等方面的问题
  - 对端节点拥塞控制机制进行加强和补充
- 拥控：防止发送者把太多的数据发送到网路中，适应瓶颈链路和路由器有限buffer，保护网络
- 流控：防止发送方的发送速度比接收方的接收速度快，适应收发双方的buffer＋cpu能力，保护端点
- 拥塞避免机制的首要任务是提前检测到拥塞

2.10.4 AQM的工作原理



为什么要 AQM ？

- 控制平均队长
- 吸收没有丢弃的突发流包
- 防止非TCP友好流仰仗突发连接的锁外行为
- 避免TCP的全局同步现象
- 减少TCP的超时次数
- 惩罚行为不端的流

2.10.5 几种AQM简介

2.10.5.1 基于队列的AQM

- RED Random Early Detection [Floyd and Jacobson, 1993]
  - 丢包概率与平均队长（缓冲区占用率）成比例
- SRED Stabilized RED [Ott et al., 1999]
  - 丢包概率与瞬时缓冲区占用率和活动流的估计数成比例.
- FRED Flow RED [Lin and Morris, 1997]
  - 使每个流的丢包率与其平均队长和瞬时缓冲区占用率成比例
- BLUE [Feng et al., 2001]
  - 用丢包事件和链路空闲事件来管理拥塞。
  - 当丢包率增加时增加丢包率；当链路空闲时减少丢包率。

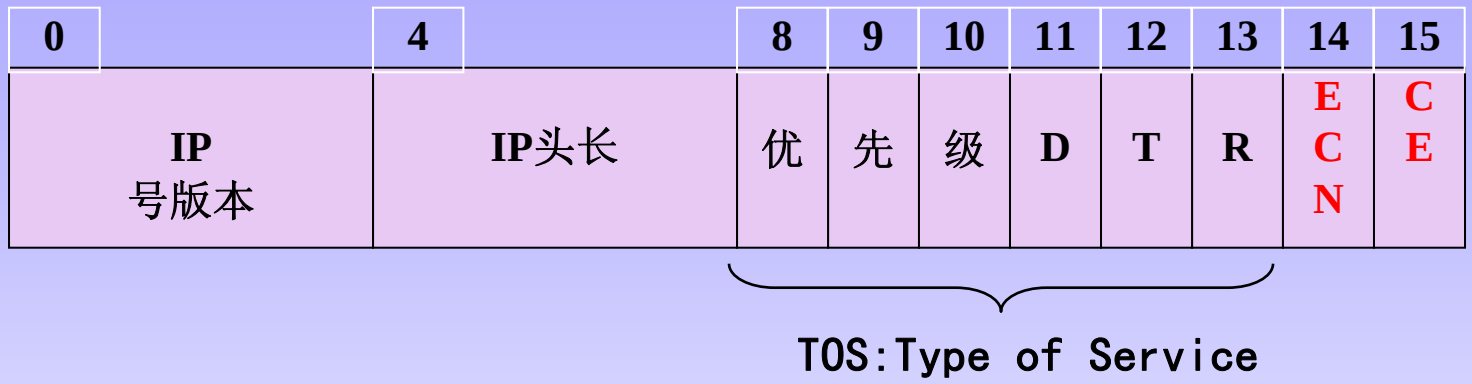
2.10.5.2 基于负载的AQM

- REM Random Exponential Marking [Athuraliya et al., 1999]
  - 通过输入和输出之间速率差及路由器中当前缓冲区占用率来计算拥塞的尺度
  - 源的发送速率与拥塞尺度成反比，于是源达到全局优化平衡
- AVQ Adaptive Virtual Queue [Kunniyur and Srikant, 2001]
  - 维持一个虚拟队列.当其队长溢出时，实际队列中的包就标记或丢弃
  - 修改虚拟容量使所有流达到希望的链路利用率
- PI (Proportional Integral) controller[Hollot et al., 2001]
  - 队列长度的变化决定丢包率，队列调节到由锁希望的队长

2.10.5.3 显示拥塞控制

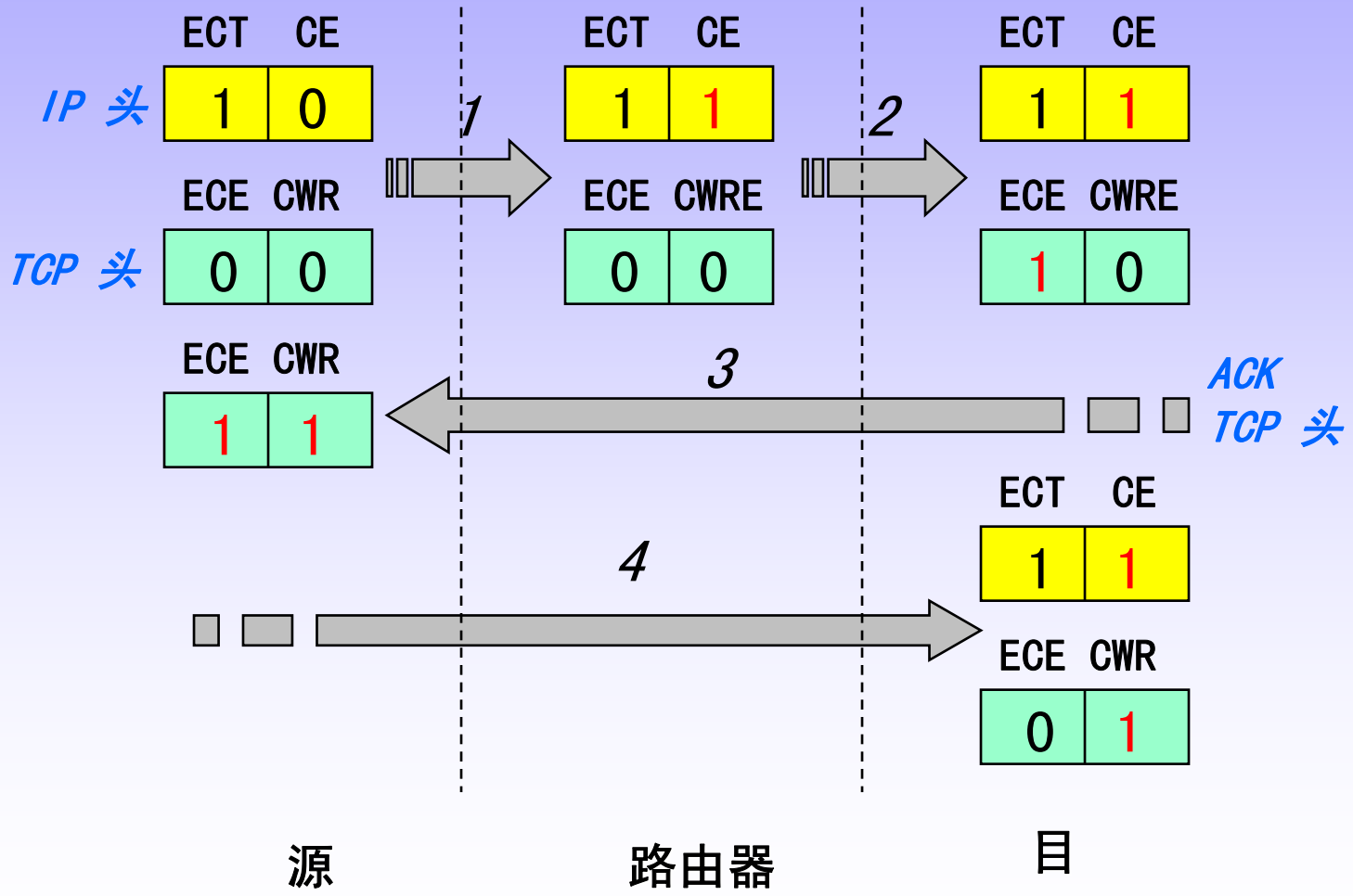
- ECN:Explicit Congestion Notification，IETF RFC 2481
  - 标记包而不是丢弃包
  - 减少超时长度和重传次数
- 一种拥塞通知方式；格式分配如下
  - IP头中设置 2 bits 的ECN域，TOS字段中的第6、7位分别定义为ECT 和 CE（V4的TOS中的前6位是区分服务）
    - ECN Capable Transport(ECT):由源对所有包设定，以指明ECN 能力
    - Congestion Experienced(CE)：由路由器设置作为拥塞的标记（代替丢弃）
  - TCP头中设置2 bits ，ECE 和 CWR标志位
    - ECE：Echo Congestion Experienced ,但接收方收到CE,在所有包上设置ECE，直到收到源的CWR
    - CWR:Congestion Window Reduced，由源设置，以指明它收到了ECE并减少了窗口大小

ECN 的表达

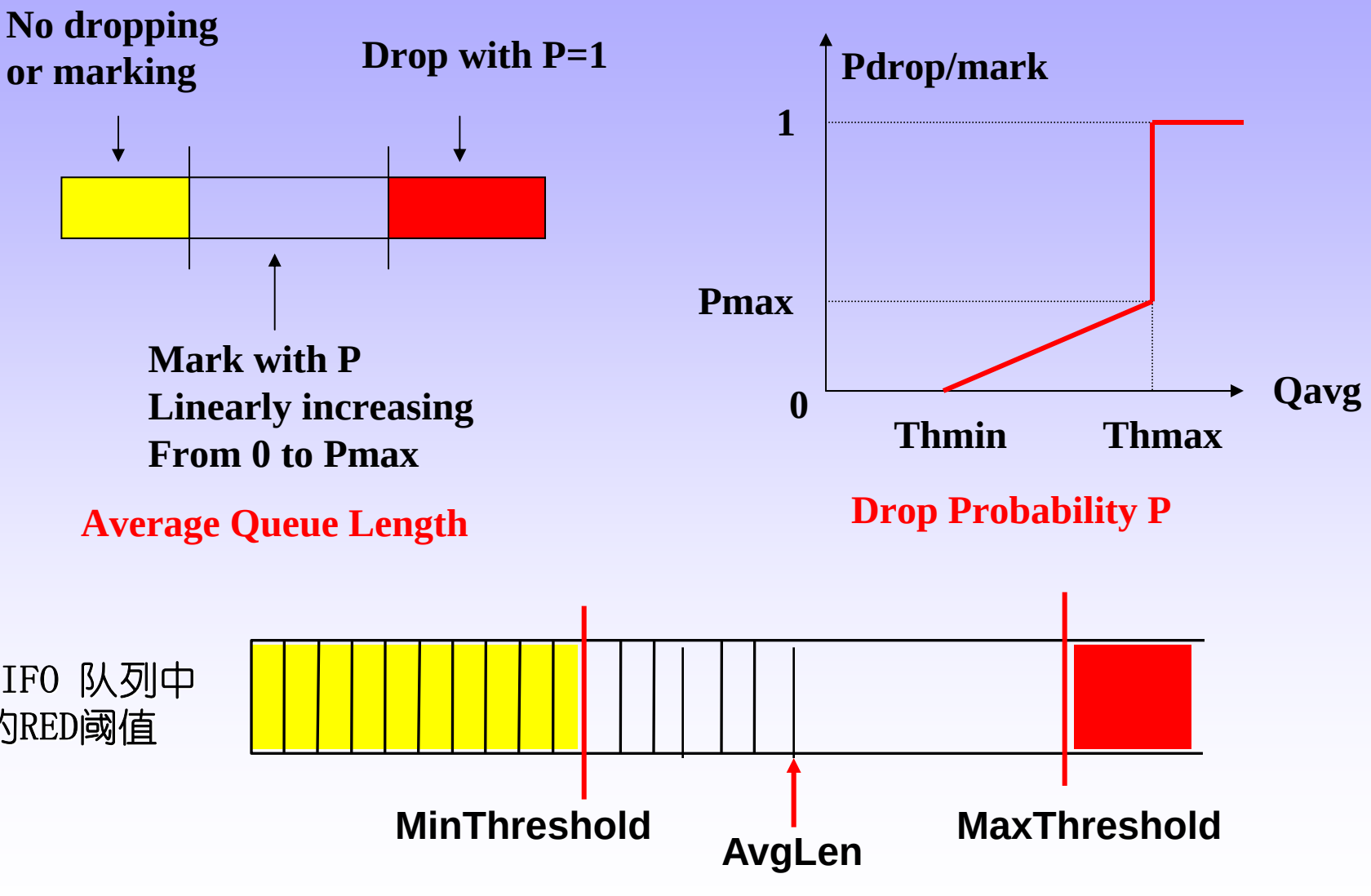


- 需要源与网络合作
  - ♣ 源须用ECN指明开启；目同意使用；
  - ♣ 目须用ECE通知源；源须同丢包一样反应，并置CWR

ENC 的工作过程


ECT CE
IP 头
1 0
ECE CWR
TCP 头
0 0
1
ECT CE
1 1
ECE CWRE
0 0
2
ECT CE
1 1
ECE CWRE
1 0
3
ACK
TCP 头
ECE CWR
1 1
4
ECT CE
1 1
ECE CWR
0 1
源
路由器
目

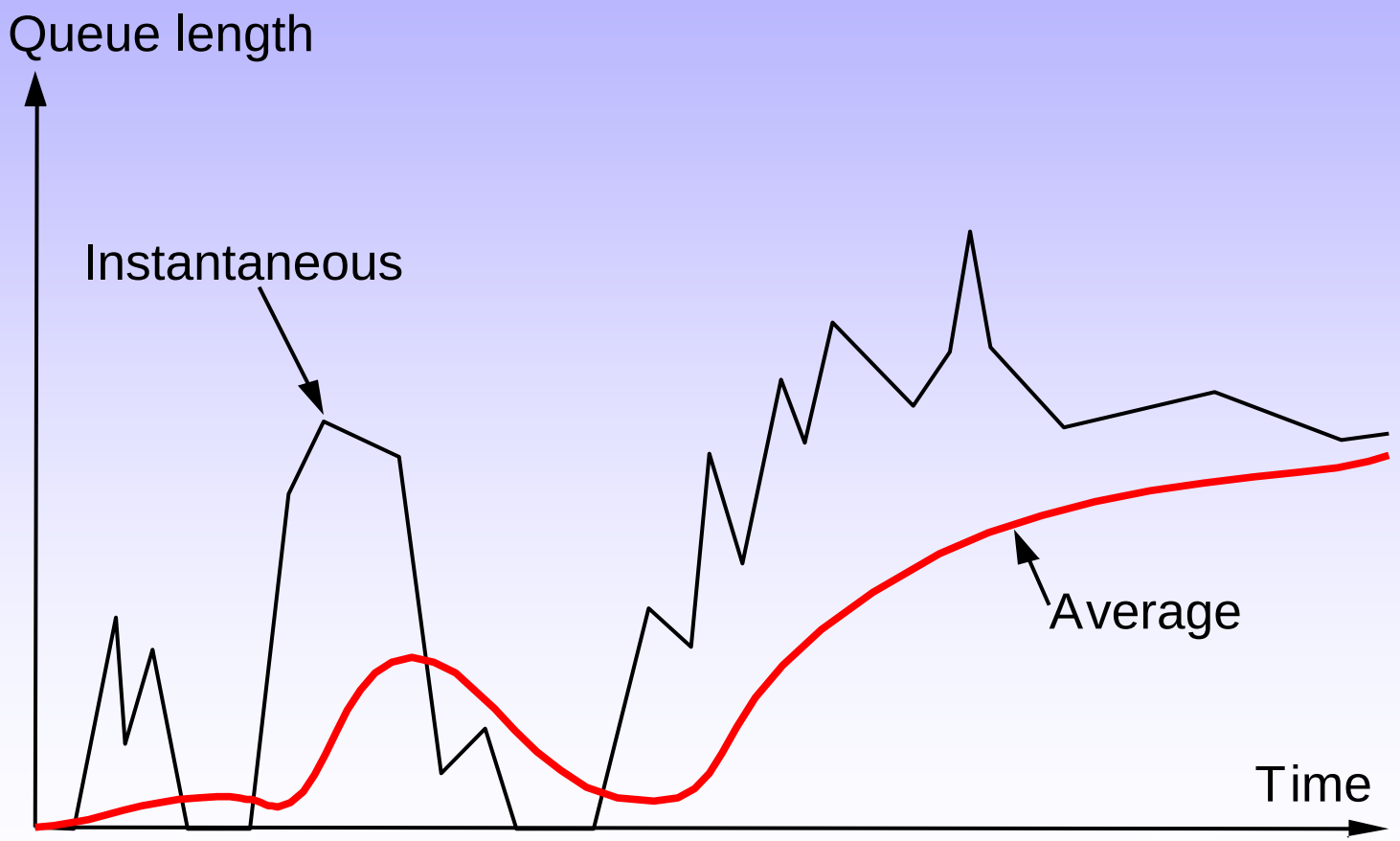
2.10.5.4 RED 簇基本原理



随机早检测机制

- RED：监控R输出端口队列的平均队长来探测拥塞
- Early:在其缓冲被完全填满前就丢少量包，使源方慢发送速率
- RED与DECbit的不同
  - RED不显式发拥塞通知到源，而是通过丢弃一个包来隐式已发生拥塞。源会被随后的超时或重复ACK所提示，这可与TCP配合使用
  - 决定何时丢弃及丢弃哪个包的细节上不同，对FIFO，不是等队列完全排满后将每个刚到达包丢弃，而是当队列超过某阈值时按某概率丢弃到达包--early random drop
- 在适当时候丢弃正进入R的包，算法实施简单
- 目标:最小化丢包率和排队延迟，避免全局同步和队突发业务的偏见（而去尾算法表现出对突发的偏见）

加权动态平均对长



Stabilized RED (SRED)

- 目标：
  - 保持FIFO队列占用稳定，且与活跃流（Active flows）数量无关
  - 鉴别恶意流，方法是到达的包同Buffer中随机选择的包进行比较，若属同一个流，则为“命中”
- 方法：
  - 在流Cache中记录M个流及其额外信息：“命中计数”和“命中包到达时间戳”
  - 函数Hit(t)记录第t个包是否击中，击中，则Hit(t)取值为1，否则为0
  - 如果一个包“命中”且该count值很大，则可认为它属于恶意流，丢包P增大

Flow RED

- 对每活跃流（per-active-flow）进行记帐（accounting），从而对使用不同带宽的流作出不同的标记包
- RED的改进版本，主要目的是解决RED算法的公平性问题
  - RED是基于带宽使用的比例，随机地选择一个流标记其包，来通知拥塞；
  - 而FRED是从缓冲区中有较多排队包的那些流中选择一个，向其发出拥塞通知。

ARED: Adaptive RED

- 拥塞通知时应充分考虑到瓶颈链路上流的数量
  - 而RED并没有考虑到这一点。
  - ARED提出了一种自动配置机制，根据流量的变化来配置适当的参数。
  - 根据平均队长是否在min_th和max_th之间，对maxp采用积式增加和减少（Multiplicative Increase Multiplicative Decrease，MIMD）从而尽量保持平均队长在min_th和max_th之间

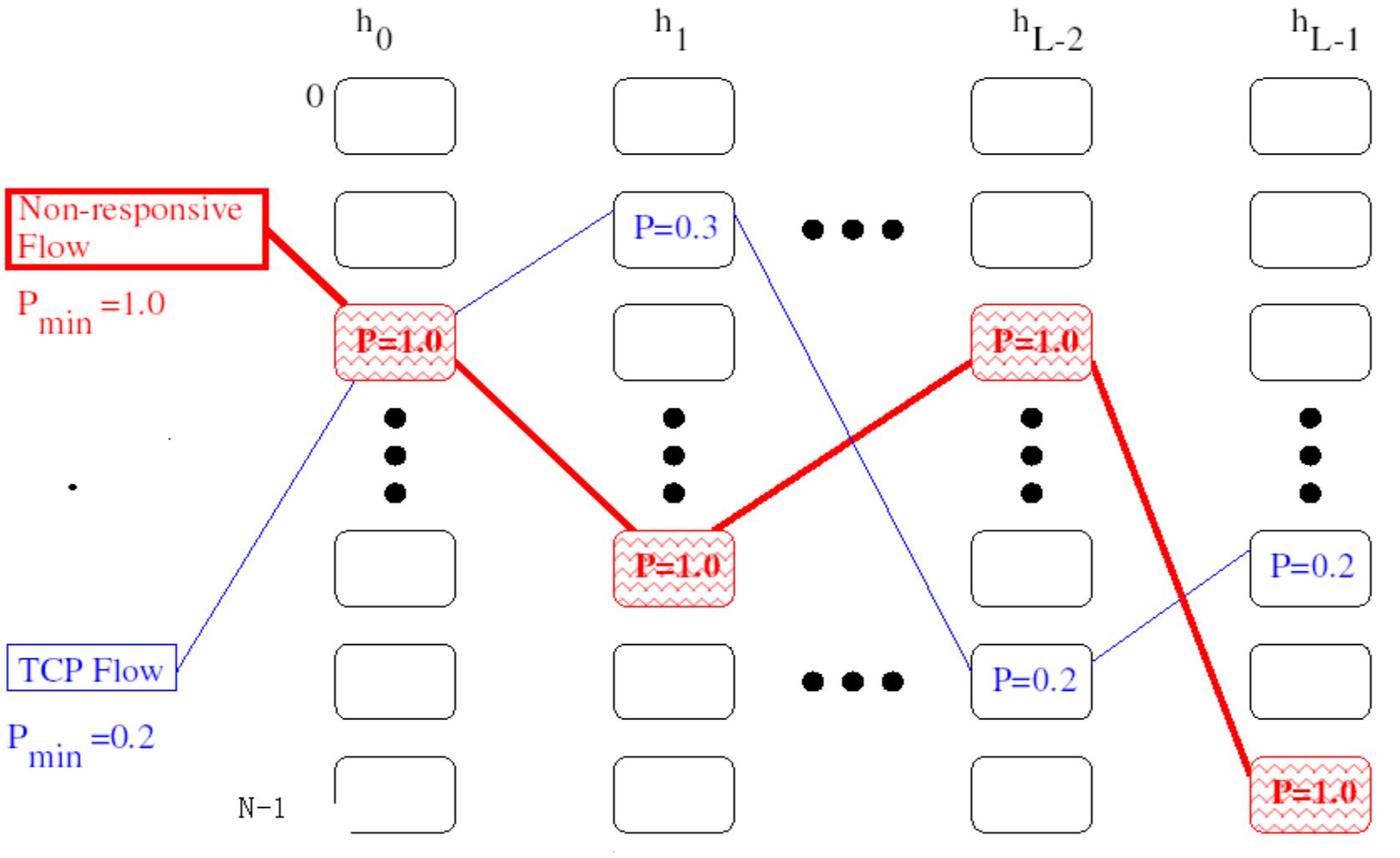
2.10.5.5 BLUE

- **用丢包事件和链路空闲事件**来管理拥塞
  - 如果由于队列溢出导致持续的丢包，BLUE就增加Pm，**因而增加了向源端发送拥塞通知的速度**
  - 如果由于链路空闲导致队列空，**BLUE就减小Pm**
- BLUE最大贡献在于：
  - **使用相对较小缓冲区就能够完成拥塞控制**
  - 减少端到端延迟，提高TCP流的吞吐量
  - 标记包比率相对较稳定，使得队长也相对稳定，减少了延迟抖动

SFB：Stochastic Fair BLUE

- 越来越多的应用不采用TCP的拥塞控制机制，在竞争带宽时使适应流陷入“饥饿”
- SFB：采用记帐的方法鉴别非适应流并对它们进行速度限制
  - 维持N×L层记帐“柜子”（accounting bins），L个Hash函数与每层柜子关联
  - 每个Hash函数将一个流映射到该层的一个柜子。每个柜子有一个标记包的概率Pm（基于队列占用）
    - 若映射到某个柜子的包的数量超过了一定的阈值，则该柜子的Pm便会增加；
    - 如果该柜子中包的数量降为0，Pm便会下降。
    - 取该包所映射到的L个柜子的标记包概率的最小值Pmin，如果该值为1，就认为该包所属的流是非适应流，从而对其限速。否则以Pmin的概率来标记包

SFB 的例子


h0
h1
hL-2
hL-1
0
N-1
Non-responsive Flow
Pmin =1.0
TCP Flow
Pmin =0.2
P=1.0
P=0.3
P=0.2

2.10.5.6 GREEN

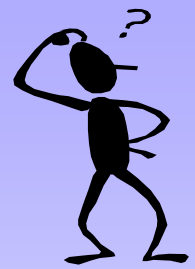
- 包到达路由器的**速度来判断拥塞程度**，根据拥塞程度来调整发送拥塞通知的速度
- 如果包到达的速度超过了链路的容量，那么在某一时间间隔内，拥塞通知速度增加一次；
- 反之，如果包到达的速度小于了链路的容量，那么在某一时间间隔内，拥塞通知速度减少一次

2.10.5.7 CHOKe

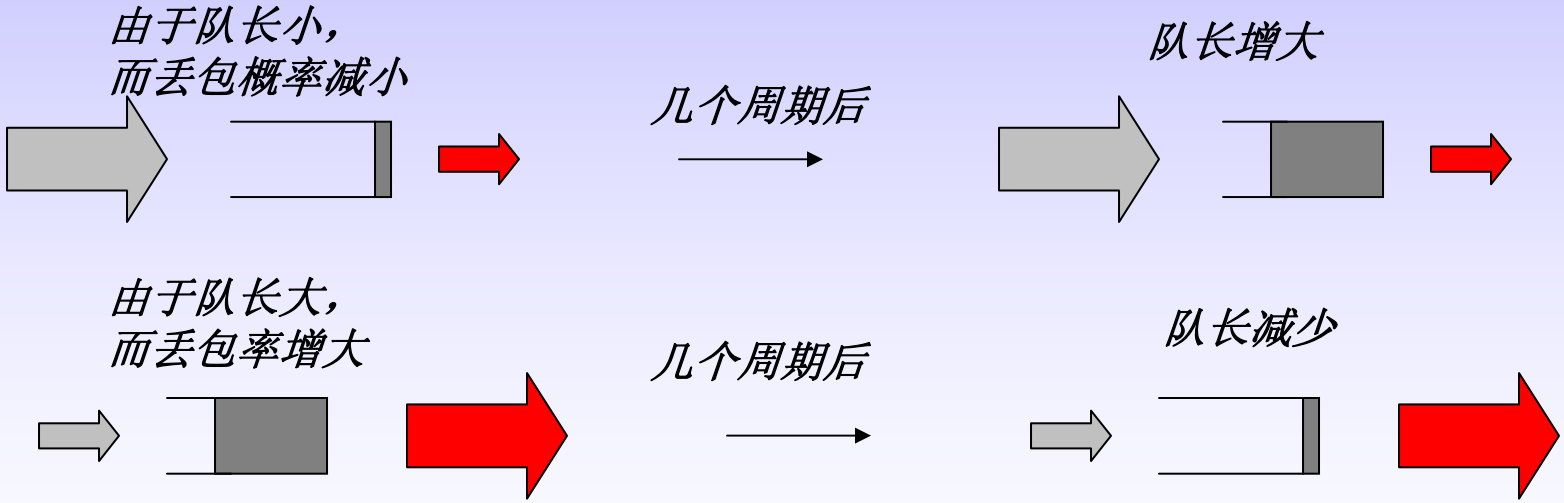
- CHOose and Keep for responsive flows
- 保护适应流，惩罚非适应流，其基本思想是
  - 当一个包到达拥塞的路由器时，从FIFO队列中随机地挑出一个包进行比较。如果它们属于同一个流，则这两个包都被丢弃；
  - 否则，被挑出包依然留下，而刚到达的包则依某种概率被丢弃，此概率的计算和RED中一样
  - CHOKe算法是基于以下原因：FIFO队列可能会更多地包含非适应流包，也就意味着这种流的包更有可能被选中进行比较。从而对非适应流进行了惩罚
- CHOKe基本上继承了RED的优点，只是少量增加了一些额外开销

2.10.6 基于负载的AQM

2.10.6.1 基于队列AQM的缺点



- 对当前队列到达率和丢弃率非常敏感
- 平均队列长度的长周期导致很慢的响应时间
- 参数构造困难



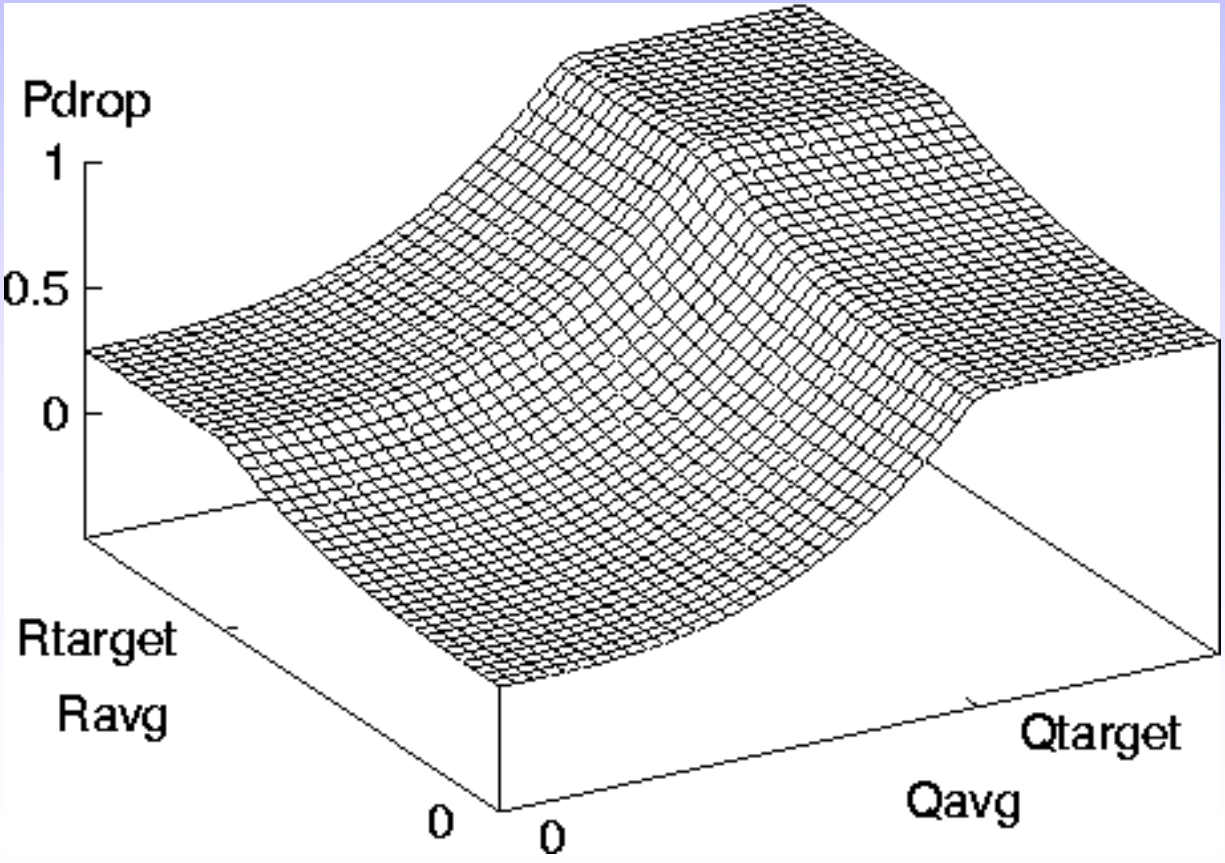
- 只用队长来指示拥塞足够了吗？能否用**负载信息**来更精确指示拥塞？
- 用负载信息后，队长信息还重要吗？能达到高响应和高吞吐率吗？

2.10.6.2 LDC: Load/Delay Control

- 负载因子
  - R = 队列的到达率/ 队列的服务率
- **特点**：同其他load-based 方法不同,LDC不需要 ECN
- **思想**：用队长和负载因子2种信息来多重指示拥塞
  - 长期：队长给出更稳定（但较慢）拥塞指示
  - 短期：负载因子能加快响应
- **目标**:
  - 使排队延迟保持在目标值之下，队列的达到率在队列的服务率之下
  - 提供更好的负载(input/output rate)响应和更直观参数
  - 保持 RED 优点：惩罚恶意流，并避免全局同步

LDC 的丢包概率

$$P_{\text{drop}} = \frac{3}{4}\min\left(1, \left(\frac{Q_{\text{avg}}}{Q_{\text{target}}}\right)^{n}\right) + \frac{1}{4}\min\left(1, \left(\frac{R_{\text{avg}}}{R_{\text{target}}}\right)^{n}\right)$$



Thank you!