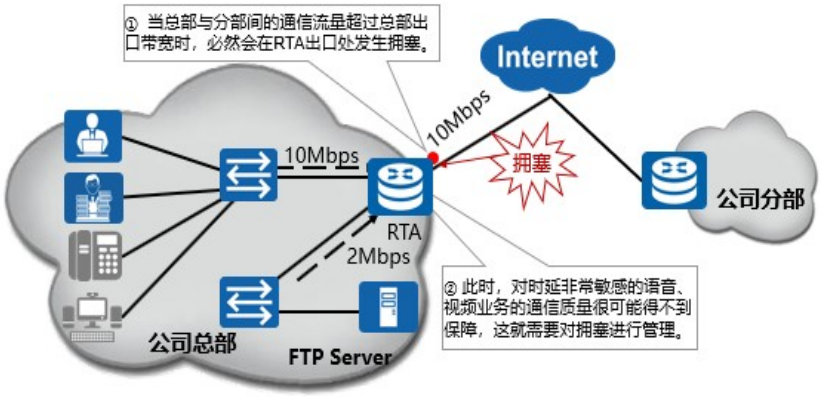
拥塞管理与拥塞避免

前言

- 当网络中间歇性的出现拥塞，且关键报文要求被更优先地转发时，此时就需要进行拥塞管理。通过采用队列技术及不同的调度算法来发送队列中的报文流。
- 如果某些突发的、非关键的报文装满队列，而后续发往该队列的关键报文都被全部丢弃，那么拥塞管理也未起到理想的效果，此时就需要配合使用拥塞避免技术。
- 那么拥塞管理和拥塞避免具体是如何实现的？在实际应用场景中，又该如何配置呢？

拥塞现象的产生



- 拥塞管理通过队列机制来实现：
  - 第一步：将准备从一个接口发出的所有报文放入不同的缓存队列中；
  - 第二步：根据各队列间的调度机制实现不同报文的差分转发。

- 拥塞发生的主要场景：
- 速率不匹配：报文从高速链路进入设备，再由低速链路

转发出去。

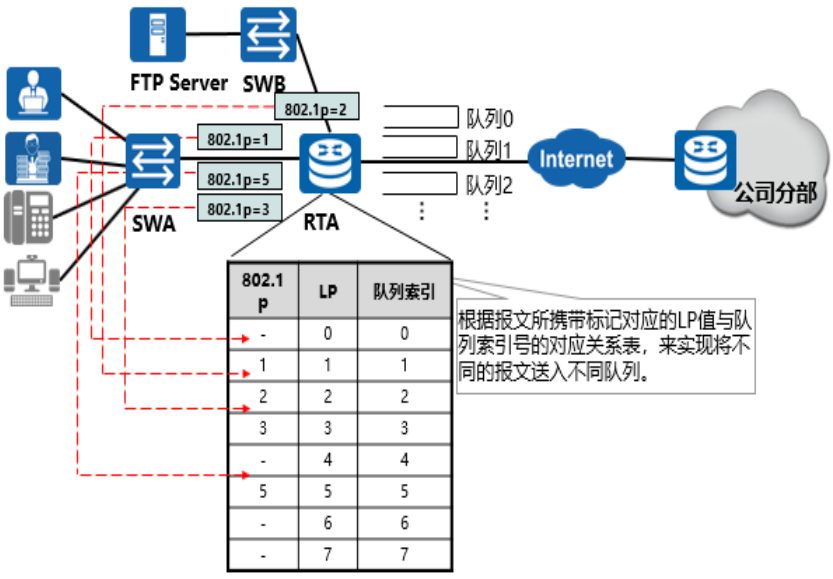
- 汇聚问题：报文从多个接口同时进入设备，由一个没有足够带宽的接口转发出去。

拥塞可能会引发一系列的负面影响：

- 增加了报文传输的时延和抖动。
- 过高的延迟会引起报文重传。
- 使网络的有效吞吐率降低，造成网络资源的损害。
- 加剧耗费大量的网络资源（特别是存储资源），不合理的资源分配甚至可能导致系统陷入资源死锁而崩溃。

由此可见，拥塞使流量不能及时获得资源，是造成服务性能下降的源头。然而在报文交换以及多用户业务并存的复杂环境下，拥塞又是常见的。因此采取有效的避免拥塞以及防止拥塞加剧的方法是必需的，那具体实现的方法是怎样的呢？

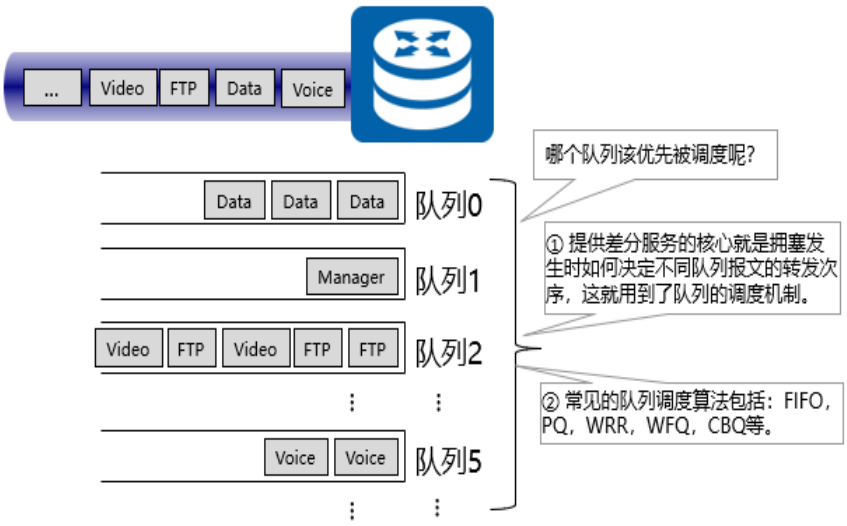
拥塞管理实现的第一步



- LP（本地优先级，又称为内部优先级）：优先级映射实现从数据原始携带的 QoS 优先级到内部优先级或从内部优先级到 QoS 优先级的映射。

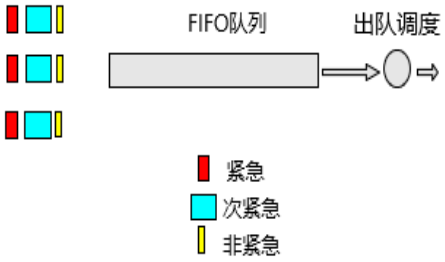
- 对于进入设备的报文，设备将报文携带的优先级或者端口优先级映射为内部优先级，然后根据内部优先级与队列之间的映射关系确定报文进入的队列。

拥塞管理实现的第二步



- 具体差分服务是如何通过队列调度来体现的呢？每种队列调度算法又是如何工作的？各自都有什么优缺点？

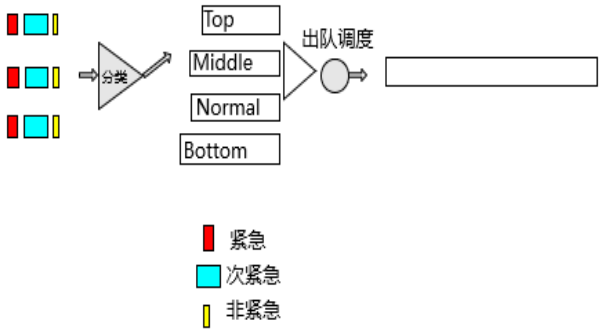
FIFO(First In First Out)



- 优点：实现机制简单且处理速度快。
- 缺点：不能有差别地对待优先级不同的报文。

- FIFO 队列不对报文进行分类，当报文进入接口的速度大于出接口能发送的速度时，FIFO 按报文到达接口的先后顺序让报文进入队列，同时，FIFO 在队列的出口让报文按进队的顺序出队，先进的报文将先出队，后进的报文将后出队。
- FIFO 队列具有处理简单，开销小的优点。但 FIFO 不区分报文类型，采用尽力而为的服务模型，使得对时延敏感的实时应用的延迟得不到保证，关键业务的带宽也不能得到保证。

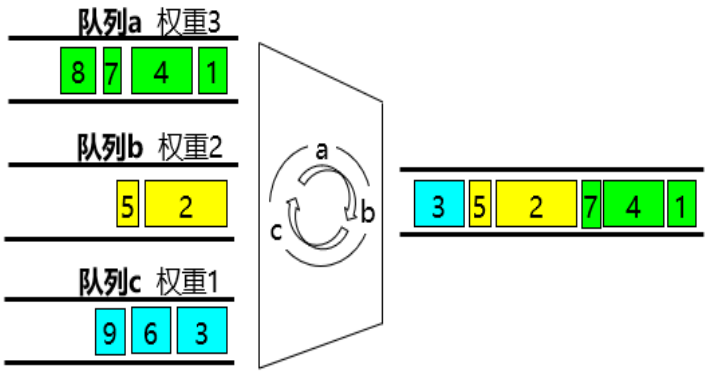
PQ(Priority Queuing)



- 优点：对高优先级的报文提供了优先转发。
- 缺点：低优先级队列可能出现“饿死”现象。

- PQ 队列是针对关键业务应用设计的，且关键业务有一个重要特点，就是需要在拥塞发生时要求优先获得服务以减少响应的延迟。
- PQ 调度机制：分为 4 个队列，分别为高优先队列、中优先队列、正常优先队列和低优先队列，它们的优先级依次降低。在报文出队的时候，PQ 会首先让高优先队列中的报文出队并发送，直到高优先队列中的报文发送完，然后发送中优先队列中的报文，同样，直到发送完，然后是正常优先队列和低优先队列。如此的话，将关键业务的报文放入较高优先级的队列，将非关键业务（如 E-Mail）的报文放入较低优先级的队列，可以保证关键业务的报文被优先传送，非关键业务的报文在处理关键业务数据的空闲间隙被传送。
- 如果高优先级队列中持续有报文等待被发送，那么后面较低优先级队列中的报文就迟迟不能得到发送，出现“饿死”现象。

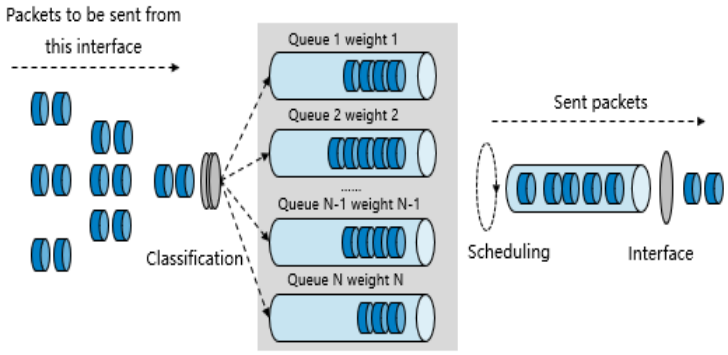
WRR(Weighted Round Robin)



- 优点：避免了PQ调度的“饿死”现象。
- 缺点：基于报文个数来调度，容易出现包长尺寸不同的报文出现不平等调度；低时延业务得不到及时调度。

- WRR（Weight Round Robin）加权循环调度在 RR（Round Robin）调度的基础上演变而来，根据每个队列的权重来轮流调度各队列中的报文流。实际上，RR 调度相当于权值为 1 的 WRR 调度。

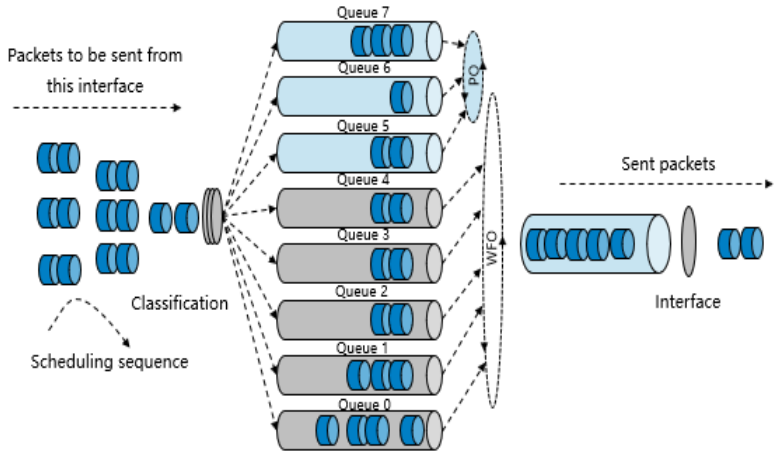
WFQ(Weighted Fair Queuing)



- 优点：可完全按照权重分配带宽；自动分类，配置简单。
- 缺点：低时延业务仍得不到及时调度；无法实现用户自定义分类规则。

- WFQ 对报文按流特征进行分类，对于 IP 网络，相同源 IP 地址、目的 IP 地址、源端口号、目的端口号、协议号、ToS 的报文属于同一个流，而对于 MPLS 网络，具有相同的标签和 EXP 域值的报文属于同一个流。每一个流被分配到一个队列，该过程称为散列，采用 HASH 算法来自动完成，这种方式会尽量将不同特征的流分入不同的队列中。WFQ 允许的队列数目是有限的，用户可以根据需要配置该值。
- 在出队的时候，WFQ 按流的优先级来分配每个流应占有的出口带宽。优先级的数值越小，所得的带宽越少。优先级的数值越大，所得的带宽越多。这样就保证了相同优先级业务之间的公平，体现了不同优先级业务之间的权值。
- WFQ 优点在于配置简单，但由于流是自动分类，无法手工干预，故缺乏一定的灵活性；且受资源限制，当多个流进入同一个队列时无法提供精确服务，无法保证每个流获得的实际资源量。WFQ 均衡各个流的延迟与抖动，同样也不适合延迟敏感的业务应用。
- 通过以上分析，会发现如果所有队列都应用一种调度算法都存在各自的优缺点，且不能很好地满足业务需求，但通过分析会发现有些调度算法之间的优缺点正好是互补的，试想：是否可以设置不同的队列应用不同的调度算法，这样是否就能很大程度满足业务需求呢？

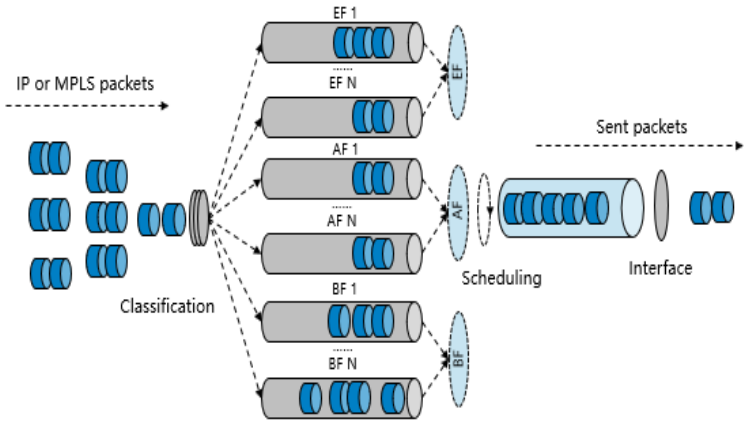
PQ+WFQ



- 优点：可保证低时延业务得到及时调度；实现按权重分配带宽等。
- 缺点：无法实现根据用户自定义灵活分类报文的需求。

- PQ+WFQ 调度过程：
- 如图，在进行调度时，首先按照 PQ 方式优先调度 Queue7、Queue6 和 Queue5 队列中的报文流，只有这些队列中的报文流全部调度完毕后，才开始以 WFQ 方式调度 Queue4、Queue3、Queue2、Queue1 和 Queue0 队列中的报文流。其中，Queue4、Queue3、Queue2、Queue1 和 Queue0 队列包含自己的权值。
- 重要的协议报文以及有低时延需求的业务报文应放入 PQ 调度队列中，得到优先调度的机会，其他报文则可放入以 WFQ 方式调度的各队列中。
- PQ+WFQ 优缺点：
- 其集合了 PQ 调度和 WFQ 调度各自的优缺点。单纯采用 PQ 调度时，低优先级队列中的报文流可能会长期得不到带宽，而单纯采用 WFQ 调度时，低延时需求业务可能得不到及时调度，而如果将两种调度方式结合起来形成 PQ+WFQ 调度方式，其不仅能发挥两种调度的优势，而且能互补一些各自特有的缺点。

CBQ(Class-based Queueing)



- 优点：提供了自定义类的支持；可为不同的业务定义不同的调度策略。
- 缺点：由于涉及到复杂的流分类，故启用CBQ会耗费一定的系统资源。

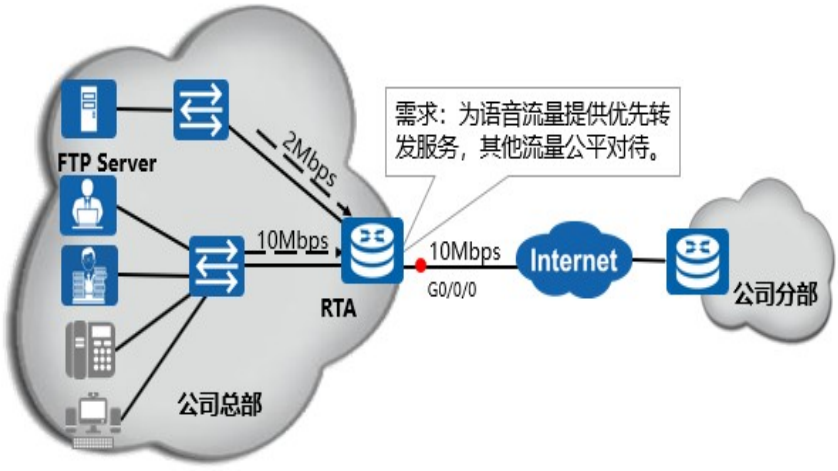
- CBQ（Class-based Queueing）基于类的加权公平队列是对 WFQ 功能的扩展，为用户提供了自定义类的支持。CBQ 首先根据 IP 优先级或者 DSCP 优先级、入接口、IP 报文的五元组等规则来对报文进行分类，然后让不同类别的报文进入不同的队列。对于不匹配任何类别的报文，会送入系统定义的缺省类。
- CBQ 提供三类队列：
- EF 队列：满足低时延业务。
- EF 队列拥有绝对优先级，仅当 EF 队列中的报文调度完毕后，才会调度其他队列中的报文。
- AF 队列：满足需要带宽保证的关键数据业务。
- 每个 AF 队列分别对应一类报文，用户可以设定每类报文占用的带宽。当系统调度报文出队的时候，会按用户为各类报文设定的带宽将报文进行出队发送，可实现各个类的队列的公平调度。
- BE 队列：满足不需要严格 QoS 保证的尽力发送业务。
- 当报文不匹配用户设定的所有类别时，报文会被送入系

统定义的缺省 BE（Best Effort，尽力传送）类。BE 队列使用接口剩余带宽和 WFQ 调度方式进行发送。

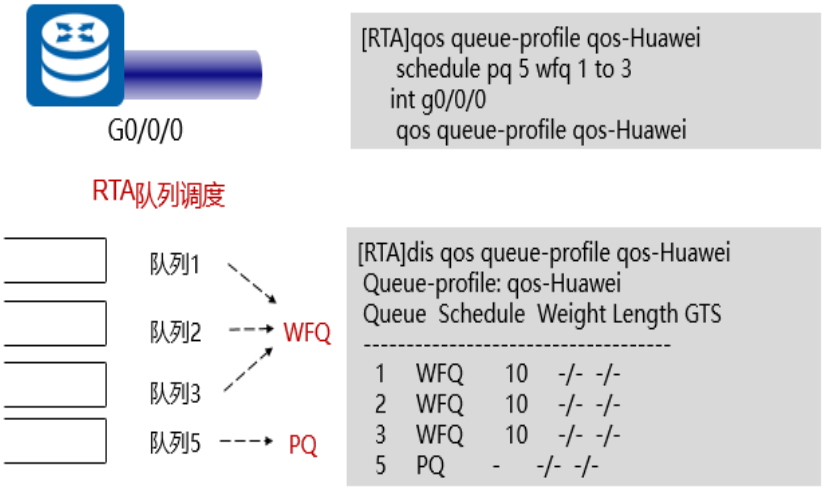
队列调度算法的比较

类型	优点	缺点
FIFO	实现简单，处理速度快	不能有差别地对待优先级不同的报文
PQ	低延迟业务能得到保障	低优先级队列可能出现“饿死”现象
WRR	避免了低优先级队列的“饿死”现象	不平等调度；低时延业务得不到保障
WFQ	按权重实现公平调度；自动分类，配置简单	低时延业务得不到保障；无法支持自定义类
PQ+WFQ	低时延业务能得到保障；按权重实现公平调度等	无法支持自定义类
CBQ	支持自定义类	耗费较多的系统资源

拥塞管理的配置需求（PQ+WFQ）



拥塞管理的配置实现（PQ+WFQ）



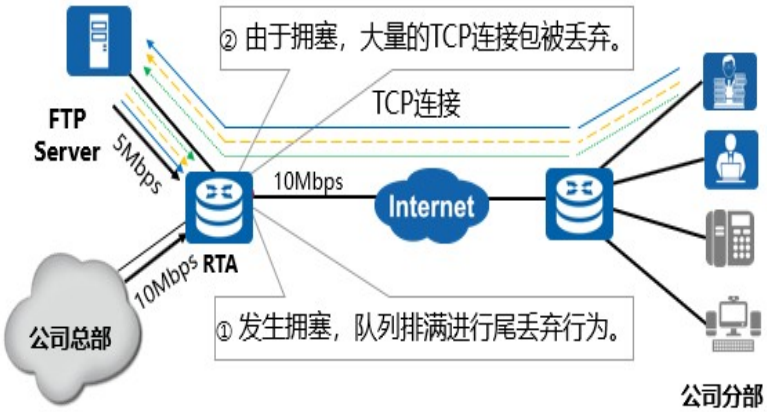
- 此页配置参数与前文标题为“拥塞管理实现的第一步”的一页内容相对应。

队列被装满后的传统处理方式

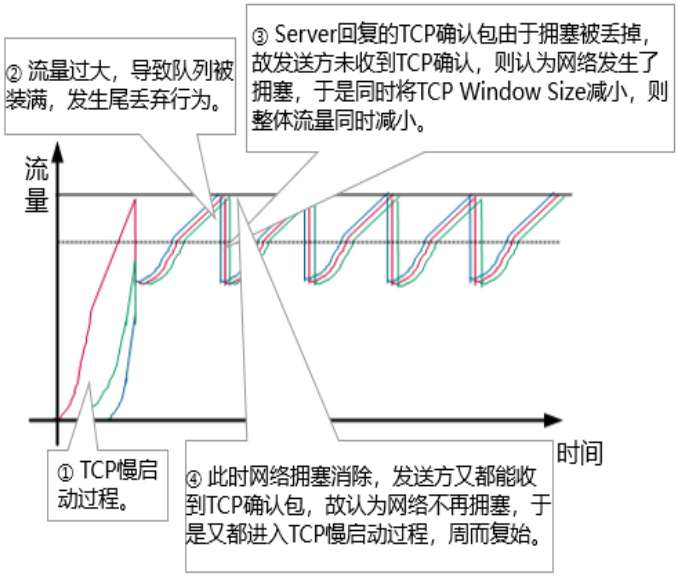
- 由于每个队列长度有限，当某一队列已经被装满时，传统的处理方法会将后续向该队列发送的报文全部丢弃，直至拥塞解除，这种处理方式称为尾丢弃（Tail Drop）。

- 思考：尾丢弃这种方式有什么利弊？

尾丢弃的缺点一：引发TCP全局同步现象 (1)



尾丢弃的缺点一：引发TCP全局同步现象 (2)



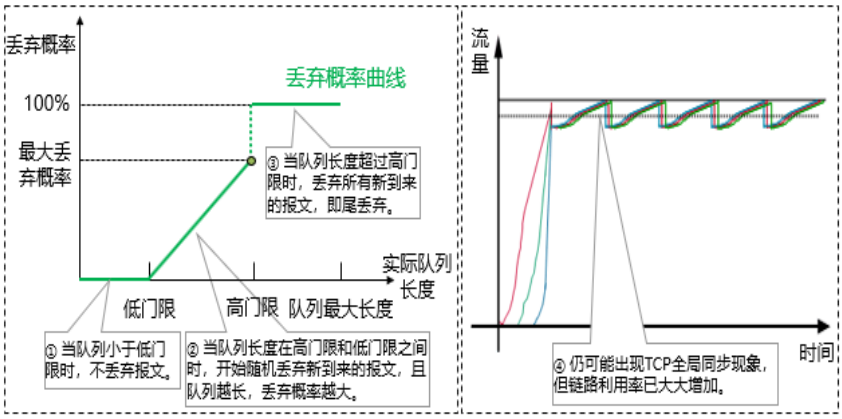
- TCP 全局同步：对于 TCP 报文，如果大量的报文被丢弃，将造成 TCP 超时，从而引发 TCP 慢启动，使得 TCP 减少报文的发送。当队列同时丢弃多个 TCP 连接的报文时，将造成多个 TCP 连接同时进入拥塞避免和慢启动状态以调整并降低

流量，这就被称为 TCP 全局同步现象。这样多个 TCP 连接发往队列的报文将同时减少，而后又会在某个时间同时出现流量高峰，如此反复，使网络资源利用率低。

- 思考：该如何来避免 TCP 全局同步现象？问题的核心是什么？

解决办法：RED

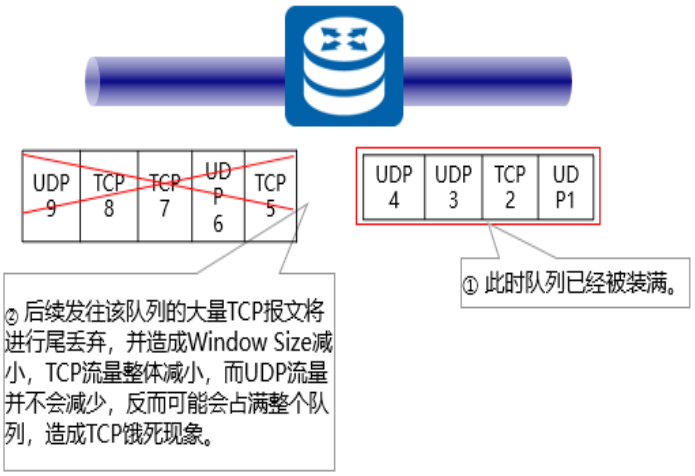
- 为避免TCP全局同步，可在队列未装满时先随机丢弃一部分报文。通过预先降低一部分TCP连接的传输速率来尽可能延缓TCP全局同步的到来。这种预先随机丢弃报文的行为被称为早期随机检测（RED）。



- 为避免 TCP 全局同步现象，出现了 RED（Random Early Detection）技术。RED 通过随机地丢弃数据报文，让多个 TCP 连接不同时降低发送速度，从而避免了 TCP 的全局同步现象。使 TCP 速率及网络流量都趋于稳定。
- RED 为每个队列的长度都设定了阈值门限，并规定：
- 当队列的长度小于低门限时，不丢弃报文。
- 当队列的长度大于高门限时，丢弃所有收到的报文。
- 当队列的长度在低门限和高门限之间时，开始随机丢弃到来的报文。方法是为每个到来的报文赋予一个随机数，并用该随机数与当前队列的丢弃概率比较，如果大于丢弃概率则报文被丢弃。队列越长，报文被丢弃的概率越高。
- 思考：尾丢弃对于丢弃 TCP 报文造成全局同步现象，还

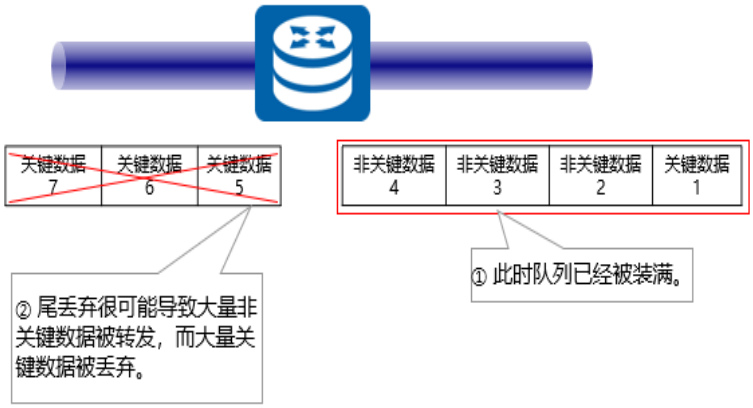
会对其造成其他的影响么？

尾丢弃的缺点二：引起TCP饿死现象



- 导致原因：尾丢弃无法对流量进行区分丢弃。

尾丢弃的缺点三：无差别地丢弃

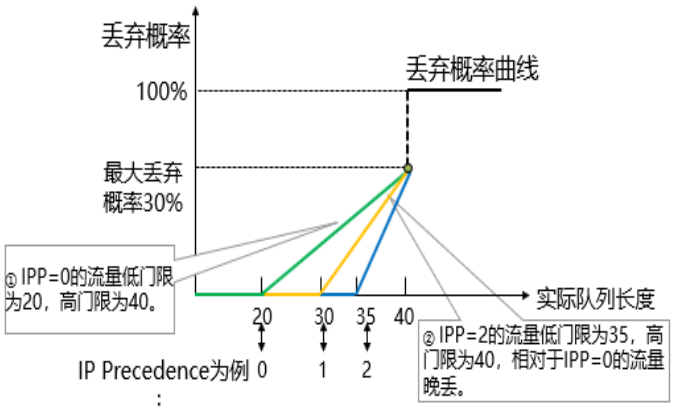


- 导致原因：尾丢弃无法对流量进行区分丢弃。

- RED 技术是否可以解决尾丢弃的缺点二和三？为什么？

解决办法：WRED

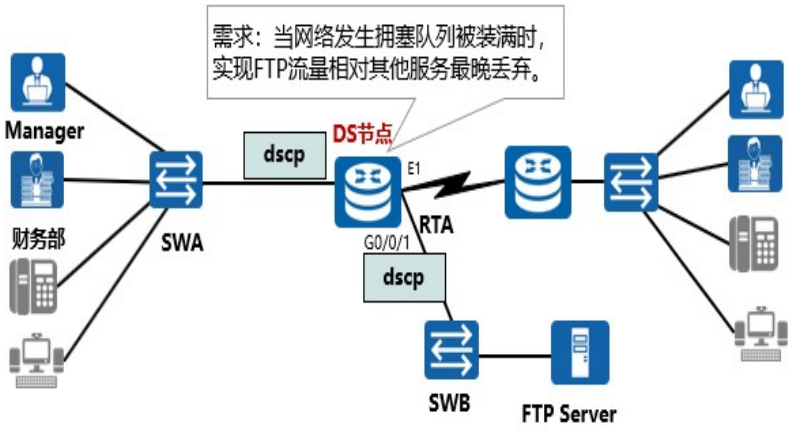
- WRED技术可以通过对不同优先级数据包或队列设置相应的丢弃策略，以实现对不同流量进行区分丢弃。



- WRED可以弥补尾丢弃的三个缺点，且大大提高了链路带宽利用率。

- 基于 RED 技术，又实现了 WRED（Weighted Random Early Detection）技术，可实现每一种优先级都能独立设置报文的丢包的高门限、低门限及丢包率，报文到达低门限时，开始丢包，到达高门限时丢弃所有的报文，随着门限的增高，丢包率不断增加，最高丢包率不超过设置的最大丢包率，直至到达高门限，报文全部丢弃。这样按照一定的丢弃概率主动丢弃队列中的报文，从一定程度上避免了尾丢弃带来的所有缺点。

WRED配置需求



WRED配置实现

流量类型	DSCP值	LP	队列
Voice	40	5	5 (PQ)
Video	24	3	3 (WFQ)
FTP	16	2	2 (WFQ)
Manager	8	1	1 (WFQ)

尾丢弃			
低门限	60	70	50
高门限	80	90	70
丢弃最大概率	20	10	10

```
[RTA]drop-profile manager
    wred dscp
     dscp 8 low-limit 50 high-limit 70 discard-percentage 10
    drop-profile ftp
     wred dscp
     dscp 16 low-limit 70 high-limit 90 discard-percentage 10
    drop-profile video
     wred dscp
     dscp 24 low-limit 60 high-limit 80 discard-percentage 20
    qos queue-profile qos-Huawei
     queue 1 drop-profile manager
     queue 2 drop-profile ftp
     queue 3 drop-profile video
    interface E1
     qos queue-profile qos-Huawei
```

思考题

1. 拥塞管理机制的实现过程分为哪两步?
2. 常用的队列技术有哪些?
3. RED技术可以解决尾丢弃以下哪些缺点（ ）?

   A. TCP全局同步现象

   B. TCP饿死现象

   C. 无差别的丢弃

- 1、答案：拥塞管理机制的实现过程分为：
- 第一步：将准备从一个接口发出的所有报文放入不同的队列中；
- 第二步：根据各队列间的调度机制实现不同报文的差分转发。
- 2、答案：FIFO、PQ、WFQ、PQ+WFQ、CBQ 等。
- 3、答案：A。
-