Multimedia Communication

Multimedia Systems(Module 5 Lesson 3)

Summary:

- Beyond Best-Effort
  - Motivating QoS
- Quality of Service (QoS)
- Scheduling and Policing

Sources:

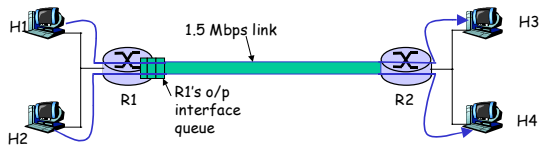
- Chapter 6 from "Computer Networking: A Top-Down Approach Featuring the Internet", by Kurose and Ross

Beyond Best-Effort

In order to take the internet into the realm of supporting quality of service (QoS) several important architectural components have been proposed.

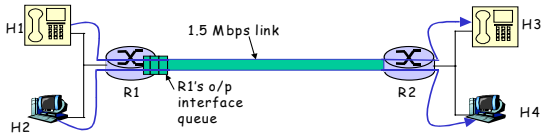
To illustrate these proposals we will use the following network:

- Suppose that two application packet flows originate on hosts H1 and H2 on one LAN and are destined for hosts H3 and H4 on another LAN. The routers on the two LANs are connected by a 1.5 Mbps link. Let's assume the LAN speeds are significantly higher than 1.5 Mbps, and focus on the output queue of router R1; it is here that packet delay and packet loss will occur if the aggregate sending rate of the H1 and H2 exceeds 1.5 Mbps.



Scenario 1: A 1 Mbps Audio App and an FTP transfer

- A 1 Mbps audio application (for example, a CD-quality audio call) shares the 1.5 Mbps link between R1 and R2 with an FTP application that is transferring a file from H2 to H4.
- In the best-effort Internet, the audio and FTP packets are mixed in the output queue at R1 and (typically) transmitted in a first-in-first-out (FIFO) order.
- A burst of packets from the FTP source could potentially fill up the queue, causing IP audio packets to be excessively delayed or lost due to buffer overflow at R1. How should we solve this potential problem?



Scenario 1 (Contd.)

- Given that the FTP application does not have time constraints, our intuition might be to give strict priority to audio packets at R1.
- Under a strict priority scheduling discipline, an audio packet in the R1 output buffer would always be transmitted before any FTP packet in the R1 output buffer.
- The link from R1 to R2 would look like a dedicated link of 1.5 Mbps to the audio traffic, with FTP traffic using the R1-to-R2 link only when no audio traffic is queued.
- In order for R1 to distinguish between the audio and FTP packets in its queue, each packet must be **marked** as belonging to one of these two "classes" of traffic.
- The Type-of-Service (ToS) field in IPv4 can be used for this

Principle 1: Packet marking allows a router to distinguish among packets belonging to different classes of traffic.

Scenario 2:
Scenario 1 with high-priority FTP

- Suppose now that the FTP user has purchased "platinum service" Internet access from its ISP, while the audio user has purchased cheap, low-budget service.
- Should the cheap user's audio packets be given priority over FTP packets in this case? Arguably not.
- It would seem more reasonable to distinguish packets on the basis of the sender's IP address. More generally, we see that it is necessary for a router to *classify* packets according to some criteria.

Principle 1 (modified): Packet classification allows a router to distinguish among packets belonging to different classes of traffic.

Scenario 3:
A misbehaving Audio App and an FTP transfer

- Suppose, the router knows it should give priority to packets from the 1 Mbps audio application.
- Since the outgoing link speed is 1.5 Mbps, even though the FTP packets receive lower priority, they will still, on average, receive 0.5 Mbps of transmission service.
- What happens if the audio application starts sending packets at a rate of 1.5 Mbps or higher (either maliciously or due to an error in the application)?
  - In this case, the FTP packets will starve
- Ideally, one wants a degree of *isolation* among flows, in order to protect one flow from another misbehaving flow.

Principle 2: It is desirable to provide a degree of isolation among traffic flows, so that one flow is not adversely affected by another misbehaving flow.

Scenario 3 (contd.):

- With strict enforcement of the link-level allocation of bandwidth, a flow can use only the amount of bandwidth that has been allocated;
- It cannot utilize bandwidth that is not currently being used by the other applications.
- It is desirable to use bandwidth as efficiently as possible, allowing one flow to use another flow's unused bandwidth at any given point in time.

Principle 3: While providing isolation among flows, it is desirable to use resources (for example, link bandwidth and buffers) as efficiently as possible.

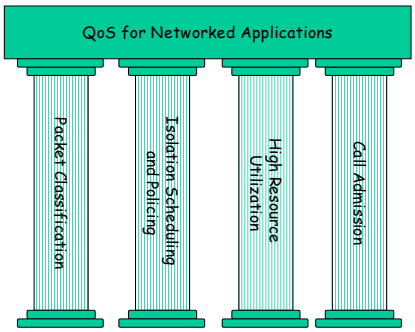
Scenario 4:

Two 1 Mbps Audio apps over an overloaded 1.5 Mbps Link

- The combined data rate of the two flows (2 Mbps) exceeds the link capacity. Even with classification and marking (Principle 1), isolation of flows (Principle 2), and sharing of unused bandwidth (Principle 3), of which there is none, this is clearly a losing proposition.
- Implicit with the need to provide a guaranteed QoS to a flow is the need for the flow to declare its QoS requirements. This process of having a flow declare its QoS requirement, and then having the network either accept the flow (at the required QoS) or block the flow is referred to as the **call admission** process.

Principle 4: *A call admission process is needed in which flows declare their QoS requirements and are then either admitted to the network (at the required QoS) or blocked from the network (if the required QoS cannot be provided by the network).*

Providing QoS Support : Summary



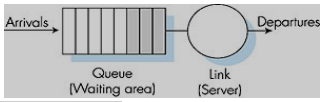
Scheduling and Policing

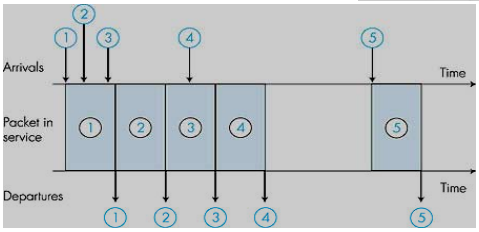
We have seen the principles (*policies*) that govern the support of QoS, now we see *mechanisms* that accomplish this.

Scheduling Mechanisms (at the link):

- FCFS

FIFO Queue Abstraction →

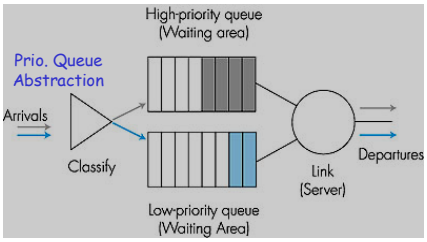


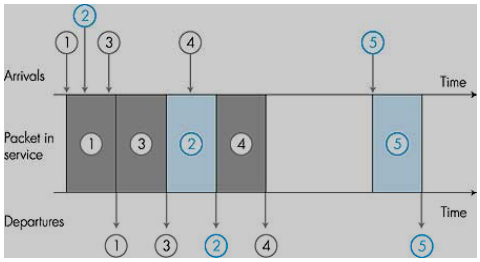


← FIFO Queue Operation

Scheduling Mechanism 2

- Priority Queueing

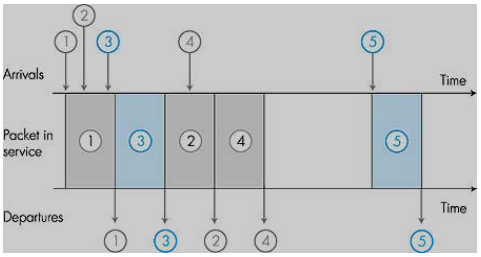




← Prio. Queue Operation

Scheduling Mechanism 3

- RR and Weighted Fair Queuing(WFQ)



← RR Operation

WFQ

- A generalized abstraction of round robin queuing that has found considerable use in QoS architectures is the so-called weighted fair queuing (WFQ) discipline
- WFQ differs from round robin in that each class may receive a *differential* amount of service in any interval of time.
- Each class, i, is assigned a weight, w_i. During any interval of time during which there are class i packets to send, class i will be guaranteed to receive a fraction of service equal to $w_i/(\Sigma w_j)$
- In the worst case, even if all classes have queued packets, class i will still be guaranteed to receive a fraction $w_i/(w_j)$ of the bandwidth.
- Thus, for a link with transmission rate R, class i will always achieve a throughput of at least $R \cdot w_i/(\Sigma w_j)$.

Policing

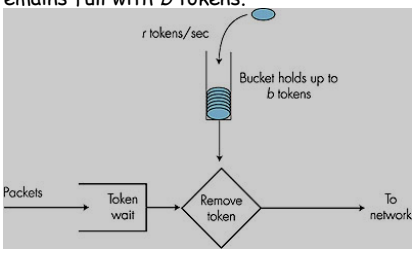
We can identify three important policing criteria, each differing from the other according to the time scale over which the packet flow is policed:

- *Average rate*: The network may wish to limit the long-term average rate (packets per time interval) at which a flow's packets can be sent into the network.
  - A crucial issue here is the interval of time over which the average rate will be policed. A flow whose average rate is limited to 100 packets per second is *more constrained* than a source that is limited to 6,000 packets per minute, even though both have the same average rate over a long enough interval of time.
- *Peak rate*: While the average rate-constraint limits the amount of traffic that can be sent into the network over a relatively long period of time, a peak-rate constraint limits the maximum number of packets that can be sent over a shorter period of time.
- *Burst Size*: The network may also wish to limit the maximum number of packets (the "burst" of packets) that can be sent into the network over an extremely short interval of time. It limits the number of packets that can be instantaneously sent into the network.

Policing: The Leaky Bucket

The leaky bucket mechanism is an abstraction that can be used to characterize policing limits:

- A leaky bucket (r,b), consists of a bucket that can hold up to b tokens. Tokens are added to this bucket as follows:
  - New tokens, which may potentially be added to the bucket, are always being generated at a rate of r tokens per second.
  - If the bucket is filled with less than b tokens when a token is generated, the newly generated token is added to the bucket; otherwise the newly generated token is ignored, and the token bucket remains full with b tokens.



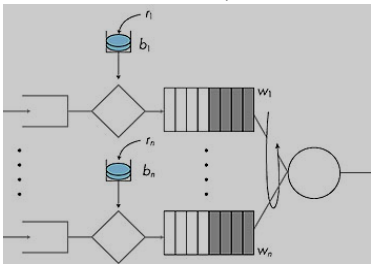
The Leaky Bucket (Contd.)

How to use the leaky bucket concept in policing?

- Suppose that before a packet is transmitted into the network, it must first remove a token from the token bucket.
- If the token bucket is empty, the packet must wait for a token.
- Let us now consider how this behavior polices a traffic flow.
  - Because there can be at most b tokens in the bucket, the maximum burst size for a leaky-bucket-policed flow is b packets.
  - The token generation rate is r, therefore, the maximum number of packets that can enter the network in *any* interval of time of length t is $rt + b$.
  - Thus, the token generation rate, r, serves to limit the long-term average rate at which the packets can enter the network.

Leaky Bucket + WFQ => Provable Maximum Delay in queue

- Consider a router's output that multiplexes n flows, each policed by a leaky bucket with parameters
  (b_i, r_i) for $i = 1, \ldots, n$, using WFQ scheduling.
- Each flow, i, is guaranteed to receive a share of the link bandwidth equal to at least $R \cdot w_i/(\Sigma\, w_j)$, where R is the transmission rate of the link in packets/sec.



Leaky Bucket + WFQ (Contd.)

- Suppose that flow 1's token bucket is initially full. A burst of b_1 packets then arrives to the leaky bucket policer for flow 1. These packets consume all of the tokens (without wait) from the leaky bucket and then join the WFQ waiting area for flow 1.
- Since these b_1 packets are served at a rate of at least $R \cdot w_1/(\Sigma\, w_j)$ packet/sec., the last of these packets will then have a **maximum delay**, d_{max}, until its transmission is completed, where:

$$d_{max} = \frac{b_1}{R \cdot w_1/\Sigma w_j}$$

- That is, if there are b_1 packets in the queue and packets are being serviced (removed) from the queue at a rate of at least $R \cdot w_1/(\Sigma\, w_j)$ packets per second, then the amount of time until the last packet is transmitted cannot be more than $b_1/(R \cdot w_1/(\Sigma\, w_j))$.