

# QoS

CSC 790

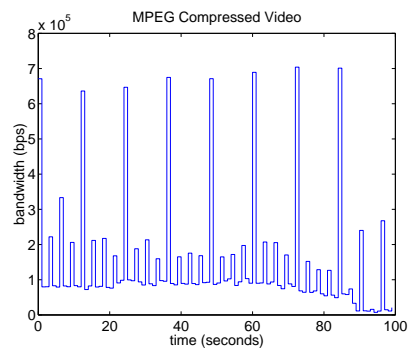
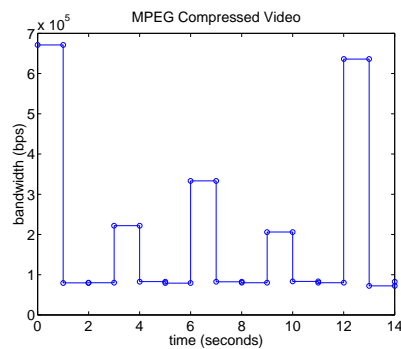
WAKE FOREST  
UNIVERSITY

Department of Computer Science

Fall 2009

## Multimedia Traffic

- Multimedia traffic tends to be highly variable



- High **peak-to-mean** ratios (18:1)
- MPEG has self-similar behavior, slowly decaying autocorrelation
- As a result it is difficult to allocate network resources

## Multimedia Network Applications

---

- Multimedia network applications are typically
  - Delay sensitive, but loss tolerant (marginally)  
*What type of MPEG frame don't you want to lose?*
  - For example, packets that are delayed beyond a few hundred milliseconds are useless (IP telephony)  
*Once delay bound exceeded, what should the network do?*
- Network must provide some service guarantees  
*What type of application is delay tolerant, but loss sensitive?*

## Quality of Service

---

As previously discussed, many applications require certain network performance bounds  $\Rightarrow$  Quality of Service (QoS) guarantees

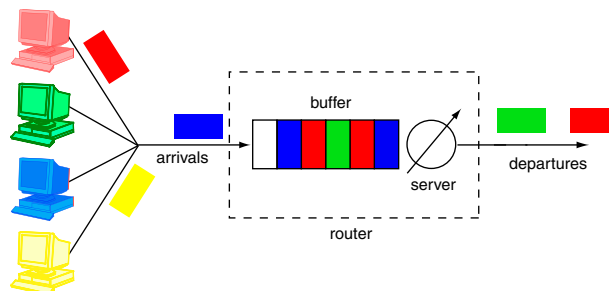
- Delay
  - Most real-time applications are delay sensitive
  - If packet arrives beyond the delay bound, it is useless
- Delay variation
  - Also called **jitter**, relative difference between packet arrivals  
*Do we know of any methods to smooth such arrivals?*
- Loss rate
  - Probability of a lost packet, (for example  $1 \times 10^{-6}$ )  
*What is the problem associated with small loss bounds?*

## Four Principles of Providing QoS Support

1. **Packet classification**
  - Network device must be able to distinguish packets
2. **Resource reservations, flow isolation, and policing**
  - Reserve network resources for certain traffic flows
  - Provide isolation among traffic flows
  - Be certain flow stays within bounds
3. **High resource utilization**
  - Efficiently use network resources
4. **Call admission**
  - Flow declares network requirements
  - Network either accepts or rejects

## Network Resources

Network resources include **buffer space** and **link bandwidth**



- Reserve per flow or per QoS class
- Determining the amount to provide a particular QoS is difficult
  - Limited a priori information
  - Resources are finite

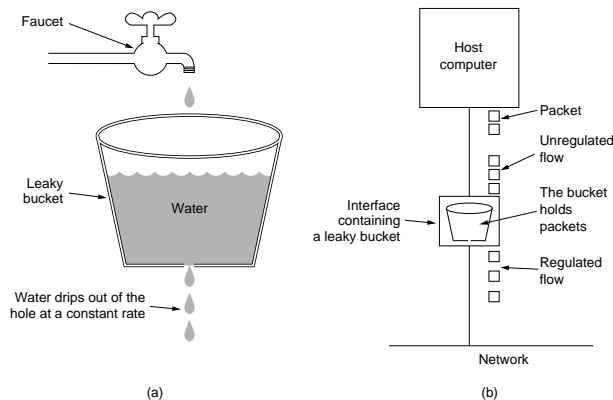
## Traffic Shaping

- Regulating average rate and burstiness of data transmission
- *Can we use a sliding window method?* (for example in TCP)
  - Limits amount transmitted at once
  - Does not limit the rate at which it is sent
- We will consider two traffic shaping mechanisms,
  - **Leaky bucket** and **token bucket**
- Shaping traffic is also referred to as **smoothing**

*At what network layer could this be used?*

## Leaky Bucket Algorithm

- Model - bucket with a hole near the bottom



- Regardless of the rate water enters the bucket
  1. Outflow is constant  $\rho$  if any water present
  2. Zero if no water

- Same *bucket* idea can be applied to cells
- Conceptually, each source is connected to the network via an interface containing a leaky bucket (finite queue)
  - If a cell arrives and the queue is full, the cell is discarded
  - Otherwise it joins queue and awaits transmission
  - Server transmits one cell per *clock-tick*

*What does this smell like?*

- As a result, source can only send one cell per *clock-tick*

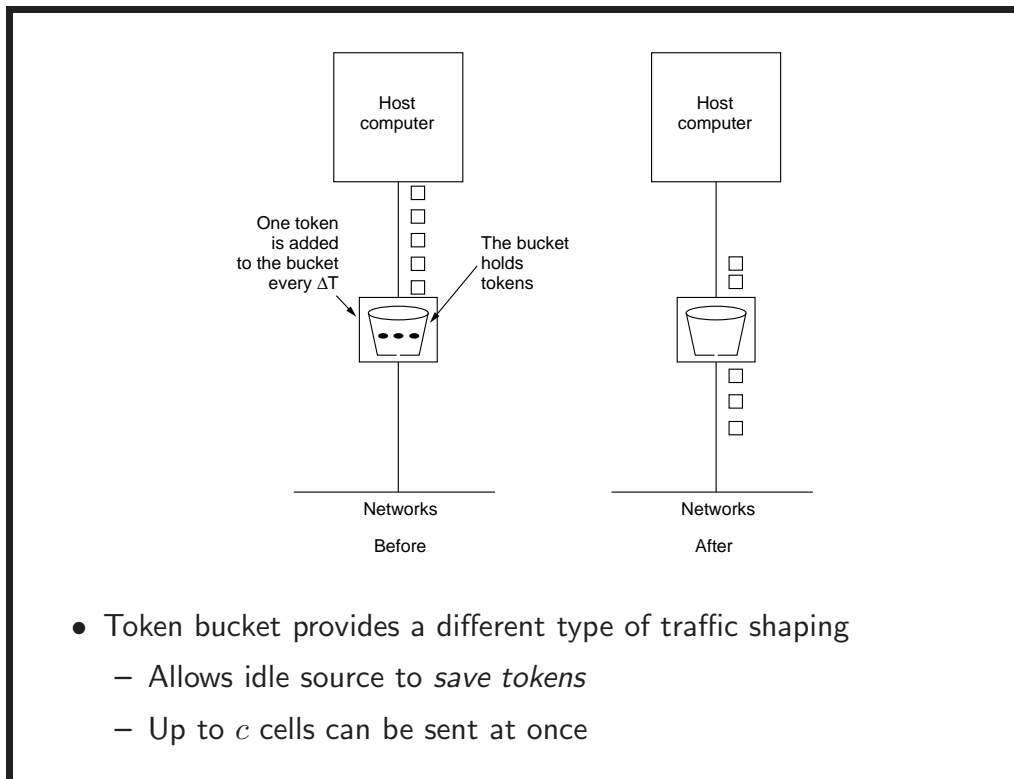
*If a source is bounded by when it can send a packet, how could a higher data rate be achieved?*

## Token Bucket Algorithm

---

- Leaky bucket enforces a rigid output rate  $\rho$ 
  - However, some applications need a limited *speed-up* in transmission when a burst arrives
  - **Token bucket algorithm** allows limited burst
- Token bucket algorithm design
  - Leaky bucket holds **tokens**
  - Tokens generated by a clock, one token every  $\Delta T$  seconds
  - Bucket has a maximum size (capacity) of  $c$  tokens
  - For a cell to be transmitted, it must remove one token from the bucket

*Bursts are allowed, what is the maximum size?*



- What if packets (variable length) are transmitted, not cells?
  - Let each token represent  $k$  bytes
  - Packet transmitted only if enough tokens are present

*What is a simple implementation of a token bucket?*
- **Composite shapers**  $\Rightarrow$  Leaky bucket + token bucket
  - Token bucket regulates the maximum burst size, would like to also bound the *peak rate*
  - Send the output of the token bucket into a leaky bucket (which governs the output rate)

## Token Bucket Performance

- Let  $s$  = burst length (seconds),  $c$  = bucket capacity (bytes),  $\rho$  = token arrival rate (bytes/second), and  $m$  = maximum source rate (bytes/second)
- *What is the duration of a maximum-rate burst through a token bucket?*

– Maximum bytes sent from the token bucket during a burst is

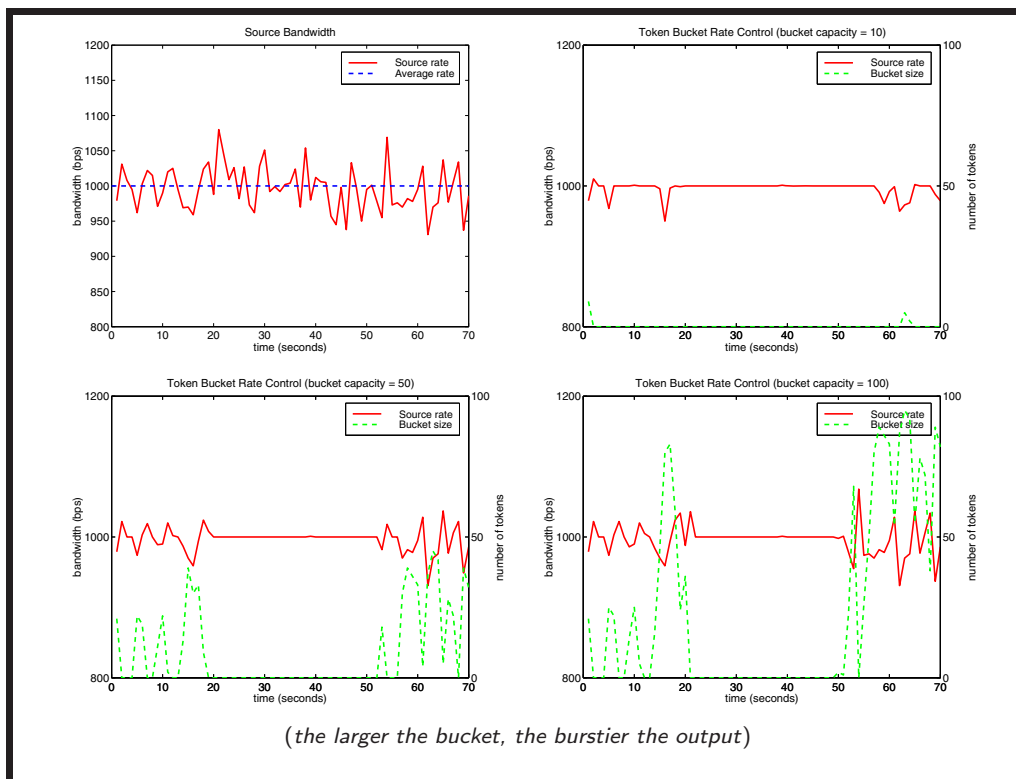
$$c + \rho \cdot s$$

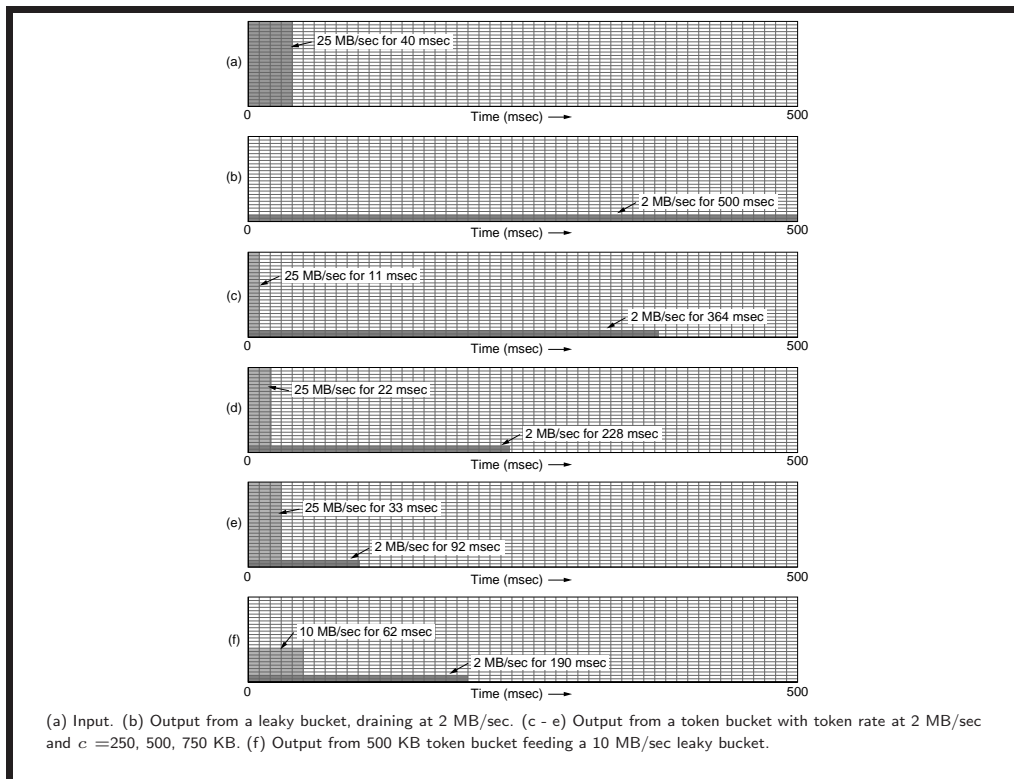
– Maximum bytes the source can send during a burst is

$$m \cdot s$$

– Setting the two equal and solving for  $s$

$$s = \frac{c}{m - \rho}$$





## Current Traffic Shaper Ideas

- Name changes...
  - Leaky buckets are also referred to as *peak rate limiters*
  - Unfortunately, many call a *token bucket* a *leaky bucket* (token bucket is considered a leaky bucket variation).
- Possible traffic shaper modification
  - If a packet arrives and no token is available, it is *marked* then sent (instead of dropped)
  - The *mark* indicates the packet is *out-of-bounds* traffic
  - Routers are allowed to drop *marked* packets if congested

*What is the advantage of this modification?*

## Scheduling and Policing Mechanisms

Packets of different flows are multiplexed together for transmission

- **Link scheduling mechanism** the manner packets are served



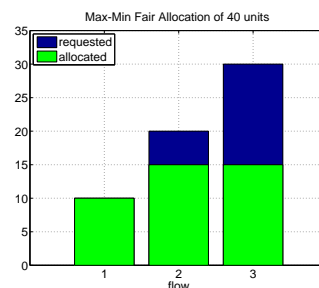
- Will assume finite queue length and fixed server rate
  - **Drop policy** - which packet is dropped if buffer is full  
*Why not drop the packet that finds queue full?*
- Four scheduling mechanisms
  - FIFO, priority queue, round robin, and weighted fair queue
- Consider performance (work done), predictability and fairness

## Scheduler Fairness

- Several definitions, but weighted max-min fair is often assumed
  - *Generally speaking*, small users get what they want and large users share the rest
- A formal definition of max-min fair:

A vector of rates  $r$  is max-min fair if it is feasible and for each flow  $i$ ,  $r_i$  cannot be increased while maintaining feasibility without decreasing  $r_j$  for which  $r_j \leq r_i$

- Allocate 40 units to three sources, demanding 10, 20, and 30 units



- Weighted max-min fair

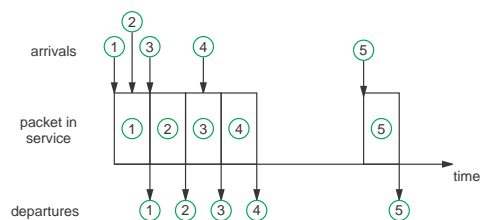
A vector of rates and weights  $(r, w)$  is max-min fair if it is feasible and for each flow  $i$ ,  $r_i$  cannot be increased while maintaining feasibility without decreasing  $r_j$  for which  $\frac{r_j}{w_j} \leq \frac{r_i}{w_i}$

- These definitions also apply to multiple hop connection allocations

*What is special about weighted max-min fair?*

## FIFO

- Packets are served in the order they arrive

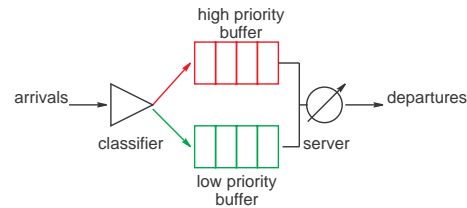


- Buffer full  $\Rightarrow$  arriving packet is dropped (FIFO with tail drop)

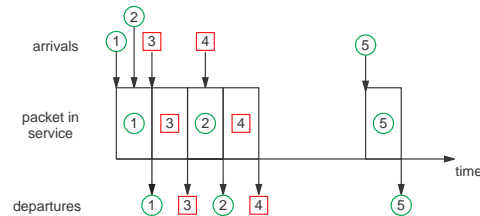
*Advantages and disadvantages of FIFO scheduling?*

## Priority Queue

- Classifier distinguishes between high/low priority



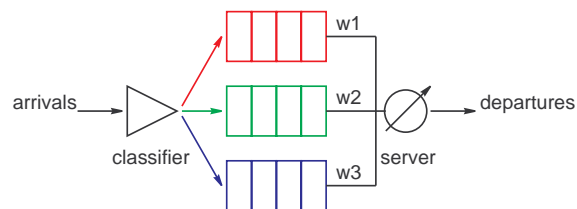
- Service higher priority packets first



*Advantages and disadvantages of priority queueing?*

## Round Robin

- Cells are sorted into classes (one buffer per class)
- Instead of a strict priority, the server alternates among classes



- Simplest form, schedule a cell from class  $1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow \dots$
- This is a **work-conserving queueing discipline**, the server will never allow the link to remain idle if there is a packet in a queue
  - If there is no cell for a given class, move to the next class

*Is round robin sufficient for QoS?*

## Weighted Fair Queueing

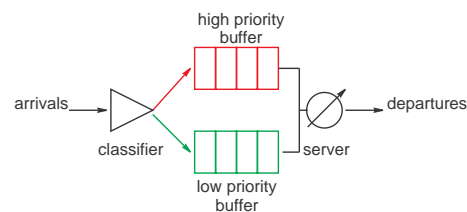
- Like round robin, cells are sorted into classes
- Server serves cells in a circular manner, and is work conserving
- Each class  $i$  may receive a different amount of service, depending on a weight  $w_i$  (differential service)
- During any interval of time class  $i$  is **guaranteed** a fraction of the server capacity  $c$  equal to

$$r_i = c \cdot \frac{w_i}{\sum w_j}$$

*In terms of bandwidth, is this an upper or lower bound?*

*What are the weights for round robin?*

- For example, if  $w_{\text{high}} = 2$ ,  $w_{\text{low}} = 1$ , and  $c = 6$



- High class cells would be given  $\frac{2}{3}$  of the bandwidth, while low class cells would be given  $\frac{1}{3}$
- *Four high class cells served for every two low class cells*
- Also provides a means for congestion control
  - Output rate of a class is governed by weight
- Different types of fairness are possible

*What are the network resources? How are they allocated?*

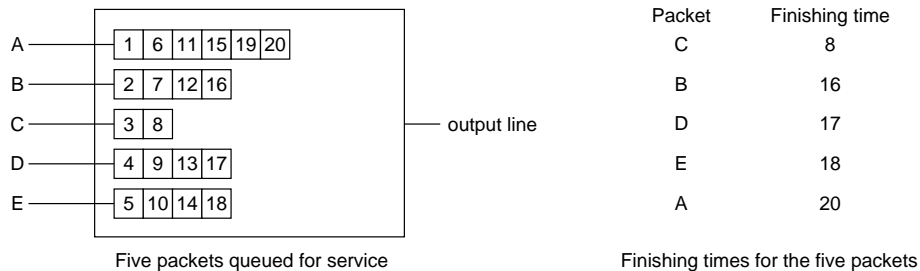
## WFQ and Packets

Consider the impact of variable length packets on WFQ

- To truly allocate bandwidth (server time) in a fair manner, the packet length must be considered
  - Otherwise, what is an easy method that achieves a higher rate?*
- Therefore we need a bit-by-bit WFQ
  - Transmit a bit from class 1, a bit from class 2, ...
  - This is called General Processor Sharing (GPS)
- Not feasible to interleave bits in GPS fashion
  - Need simulate this behavior
  - *None will be as fair as GPS, but we can bound unfairness*

## WFQ and Virtual Clock

- Compute the time packets (at the head of each class) would complete service under bit-by-bit service
- Assume a virtual clock that ticks once each time a bit is sent



- Schedule packets based on the earliest virtual clock finish time
  - Example assumes all classes have the same weight, how would you incorporate different weights?*

## Virtual Clock Performance

- Performance of schedulers is often based on *work* required
  - Work is the maximum amount of time of the associated complexities required to enqueue and dequeue a packet
- Consider the work required for virtual clock
  - Managing the queue required at least  $O(n) = n \log(n)$  work

## Deficit Round Robin

- Another approximation of WFQ (*like virtual clock*)
  - Proceeds in rounds (one pass through all queues)
  - Each flow  $i$  is allocated  $q_i$  bits each round
  - Let  $q = \min_{\forall i} \{q_i\}$  therefore share allocated to  $i$  is  $f_i = \frac{q_i}{q}$
- Initialize  $r_i = 0$  then for each round
  - Queue  $i$  can transmit up to  $(q_i + r_i)$  bytes (*complete packets*)
  - If no more packets in  $i$  after service then  $r_i = 0$ , otherwise  $r_i = (q_i - \sum p_i)$

## Example DRR

---

- Assume three flows and  $q = 1000$ 
  - Initially in the queues  $p_1 = 1500$ ,  $p_2 = 800$ , and  $p_3 = 1200$

- Packets are served as follows

- Round 1

flow	Result
1	packet <b>not</b> sent, $r_1 = 1000$
2	packet sent, $r_2 = 200$
3	packet <b>not</b> sent, $r_3 = 1000$

- Round 2

flow	Result
1	packet sent, $r_1 = 500$
2	empty buffer $r_2 = 0$
3	packet sent, $r_3 = 800$

## DRR Performance

---

- The size of  $q$  (minimum bits allocated) does impact performance
  - If  $q$  is equal to maximum packet size, guaranteed to send at least one packet per round (assuming packets are queued)
  - Smaller the  $q$  the more rounds required to send a packet...

*If  $q = 1$  is this really virtual clock?*

- DRR can also result in bursty traffic
  - A flow can potentially send multiple packets

## Token Bucket + WFQ $\Rightarrow$ Provable Maximum Delay

- Assume traffic flow  $i$  is policed using a token bucket with parameters  $\rho_i$  (token replenish rate) and  $c_i$  (bucket capacity)
- Traffic flow  $i$  is then sent to a WFQ scheduler and has weight  $w_i$  and scheduler capacity  $s$
- The maximum delay the flow will experience is

$$\frac{c_i}{s \cdot \frac{w_i}{\sum w_j}}$$

- Token buckets and WFQ are used to provide QoS guarantees

*So what exactly is the guarantee?*

## End-to-End Delay

- Consider a rate limited flow  $i$  with parameters  $(c_i, \rho_i)$ 
  - Path consists of  $n$  links, where link  $j$  has rate  $R_j$
  - Let  $r_i$  be rate reserved for flow  $i$  and  $r_i^{min} = \min_{i=1 \dots n} \{r_i\}$
  - Let  $p_i^{max}$  be largest packet for flow  $i$ , and  $P_j$  be largest packet traversing link  $j$  (all flows using link)
  - Assume  $r_i^{min} \geq \rho_i$  for stability (rate allocated  $>$  replenish rate)
- End-to-end delay is for flow  $i$  passing through  $n$  hops is

$$d_i \leq \frac{c_i}{r_i^{min}} + \sum_{j=1}^{n-1} \frac{p_i^{max}}{r_i^{min}} + \sum_{j=1}^n \frac{P_j}{R_j}$$

- First term is the delay of a GPS system
- Second term is the delay a packet may experience if it arrives just after it would have been served

- Third term is delay associated with sequentially serving packets
- For example, assume a connection with the following
  - Policed using leaky bucket with  $c = 16\text{KB}$  and  $\rho = 150\text{kbps}$
  - Connection consists of 10 hops, all 45Mbps
  - Largest packet is 8KB
  - Assuming total propagation delay is 30msec, *what rate guarantees end-to-end delay of 100msec?*

$$0.07 = \frac{8 \cdot 16 \cdot 1024}{r_i^{\min}} + \frac{8 \cdot (10 - 1) \cdot 8 \cdot 1024}{r_i^{\min}} + \frac{8 \cdot 10 \cdot 8 \cdot 1024}{45 \times 10^6}$$

$$r_i^{\min} = 11\text{Mbps}$$

## Problems with Delay Analysis

---

- To bound delay, need to select correct  $r_i^{\min}$ 
  - Lower delays require higher minimum rates
- Requires all sources to be leaky bucket regulated
  - Correctly assigning parameters can be difficult
- WFQ couples delay with bandwidth
  - Low delay requires higher bandwidth
  - Can waste bandwidth for low-bandwidth low delay sources

*What about a queueing analysis of the network?*

## Flow Specification

- Before a session can start the network and the user must agree on the specification of the traffic
- Flow specification consists of
  - $T_{spec}$  - Describes the traffic characteristics
  - $R_{spec}$  - Describes the QoS provided
- Before a connection is established
  - Flow specification is presented to the subnet
  - Subnet can accept or reject
  - If both sides agree, then session is started

*This is a form of...*

- $T_{spec}$ 
  - Maximum packet size
  - Token bucket rate
  - Token bucket size
  - Maximum transmission rate

*In other words, this will bound the...*

- $R_{spec}$ 
  - Loss rate
    - Would this be packet loss probability?*
  - Loss interval
  - Burst loss sensitivity
  - Maximum delay

## Other Issues with QoS Networks

Four principles of QoS are important, however other QoS problems still exist (we will review two)

1. Mapping network resources needs to QoS requirements
  - Suppose you want to stream an MPEG-2 video over the network, how much bandwidth/buffer-space is required?
  - We know how to reserve resources, but **how much**?
  - There is no *easy* method for mapping a  $T_{spec}$  to a  $R_{spec}$
2. Allocation fairness
  - Resources are finite and must be *fairly* allocated
  - What is *fair*?

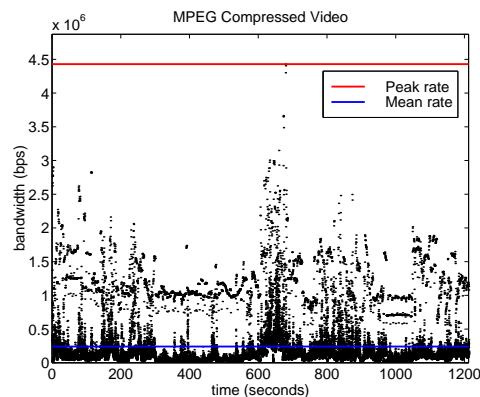
## Another Look at QoS Guarantees

There are three different types of QoS guarantees

1. Deterministic (100%)
  - Based on *peak* demand
  - Simple to determine allocation, conservative
2. Statistical (<100%)
  - Based on peak and mean
  - Less predictable
  - Complex to determine the allocation, more efficient
3. Best effort
  - No guarantees, best-effort service

## Resource Allocation

- Given a QoS requirement and model of the traffic
  - Can determine the amount of bandwidth to allocate
  - However, certain types of traffic are not easily modeled  
*What is a traffic model? What does it provide?*
- Multimedia traffic (specifically video) is difficult to model
  - The amount of *a priori* information is important
  - Compare streaming VoD, interactive, and live
- *Typically* want to allocate only once, but stream demand (amount of bandwidth required) changes over time



- Consider allocating bandwidth for a MPEG-2 video
  - Allocating *peak rate* provides zero losses but is inefficient  
*Why is this inefficient? Can video withstand non-zero loss?*
  - Allocating the *average rate* is too low, high losses

- We seek the **effective bandwidth**
  - Single allocation that yields a certain QoS
  - Formula does exist to determine this amount
  - Traffic must follow a known stochastic model
- Previous allocations were static, may be possible to reallocate
  - Dynamically allocate resources during transmission
  - Allows better utilization of resources
  - Can allocate for live/interactive multimedia

*What are the disadvantages of dynamic allocation methods?*

## Fairness

---

*No one will receive all the resources they need all the time*

- Resources are finite, so need fair allocations
  - Consider WFQ, how do you assign  $w_i$  for each user?
  - Similarly if a router is congested, who should suffer?
- Classical notions of fairness in networks centered around *fair share*
  - All users received the same amount, if you need less the remainder shared among those who need more
  - Consider 3 users and a resource supply of 7

User	Need	Allocated
1	1	1
2	4	3
3	4	3

- This idea is acceptable if all the applications are the same

## QoS Fair Allocations

---

Classical fairness is not appropriate for current networks

- A variety of traffic is present (audio, video, text, ...)
- Assume a ftp program and HDTV application need the same maximum amount of bandwidth, is a fair-share allocation best?
- Certain applications are more sensitive to bandwidth reductions
  - Should be considered when allocating bandwidth
  - This is a *QoS-fair* allocation
  - Focus on the QoS each user receives, not the amount

*What prevents all users stating their applications are highly sensitive to bandwidth reductions?*