CEN445 – Network Protocols and Algorithms

Chapter 5 – Network Layer

5.4 Quality of Service

Dr. Mostafa Hassan Dahshan
Department of Computer Engineering
College of Computer and Information Sciences
King Saud University
mdahshan@ksu.edu.sa
http://faculty.ksu.edu.sa/mdahshan

Quality of Service (QoS)

- With growth of multimedia networking
- Ad-hoc congestion control not enough
- Need protocols to guarantee QoS
- Match service level with applications needs

Requirements

- Flow: stream of packets from source to dest
  - virtual circuit: flow packets use same path
  - datagram: may follow different paths
- Primary parameters for flow needs
  - reliability
  - delay
  - jitter
  - bandwidth

Requirements

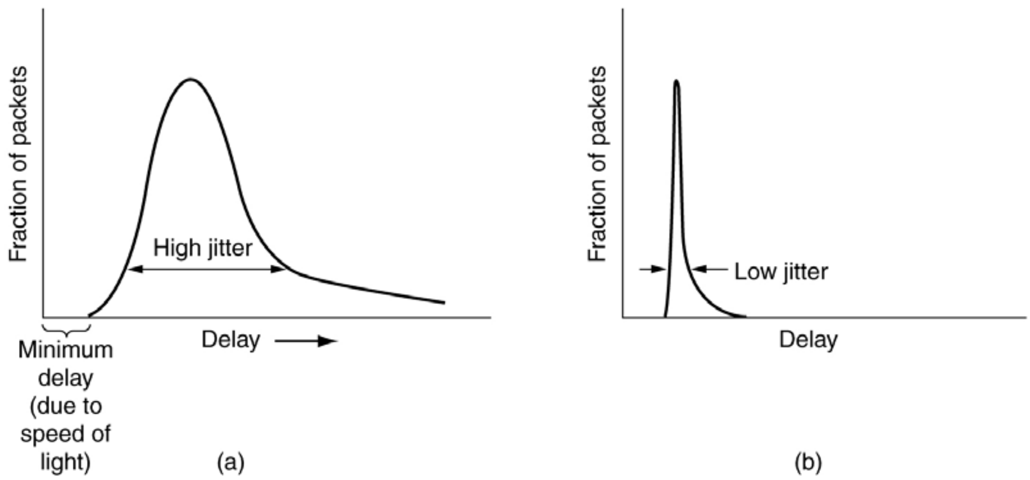
Application	Reliability	Delay	Jitter	Bandwidth
E-mail	High	Low	Low	Low
File transfer	High	Low	Low	Medium
Web access	High	Medium	Low	Medium
Remote login	High	Medium	Medium	Low
Audio on demand	Low	Low	High	Medium
Video on demand	Low	Low	High	High
Telephony	Low	High	High	Low
Videoconferencing	Low	High	High	High

Requirements

- Reliability
  - no bits may be delivered incorrectly
  - achieved using checksum, verified at dest
  - damaged packets are not ACKed; retransmitted
  - important for email, file transfer, web
  - less important for audio, video
- Delay
  - many application have low, medium required
  - real-time audio, video, strict delay required

Requirements

- Jitter
  - variation in delay
  - problem with interactive applications



Requirements

- Bandwidth
  - video is the most demanding application
  - Main categories (as in ATM)
    - constant bit rate: telephony
    - real-time variable bit rate: video conferencing
    - non real-time VBR: watching movie on Internet
    - available bit rate: file transfer

Techniques for Achieving Good Quality of Service

- Overprovisioning
- Buffering
- Traffic Shaping
- The Leaky Bucket Algorithm
- The Token Bucket Algorithm
- Resource Reservation
- Admission Control
- Proportional Routing
- Packet Scheduling

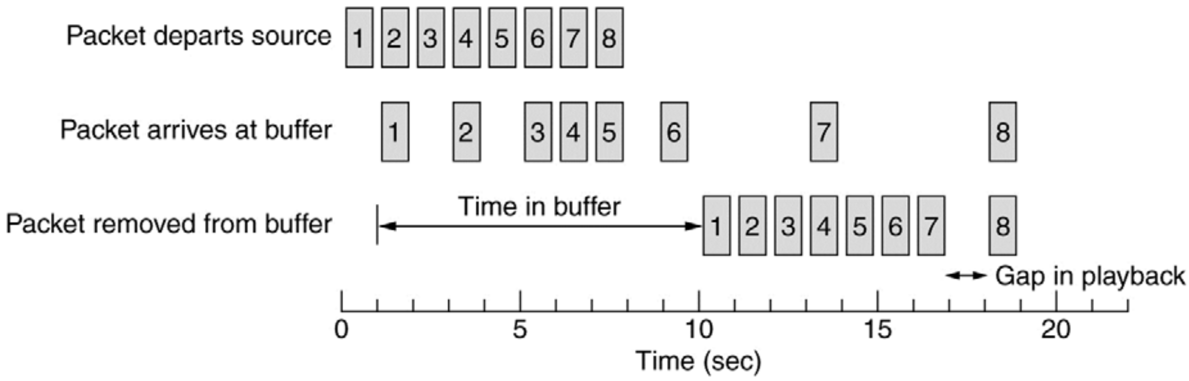
Overprovisioning

- Provide more resources than peak load
  - router capacity, buffer space, bandwidth
- Expensive
- Practical if designers know how much is enough
- Used in telephone system

Buffering

- Buffer flows at receiver before delivery
- No effect on bandwidth, reliability
- Delay is increased but jitter is smoothed
- Buffered data can played at uniform intervals
- Video steaming sites usually buffer 10 sec

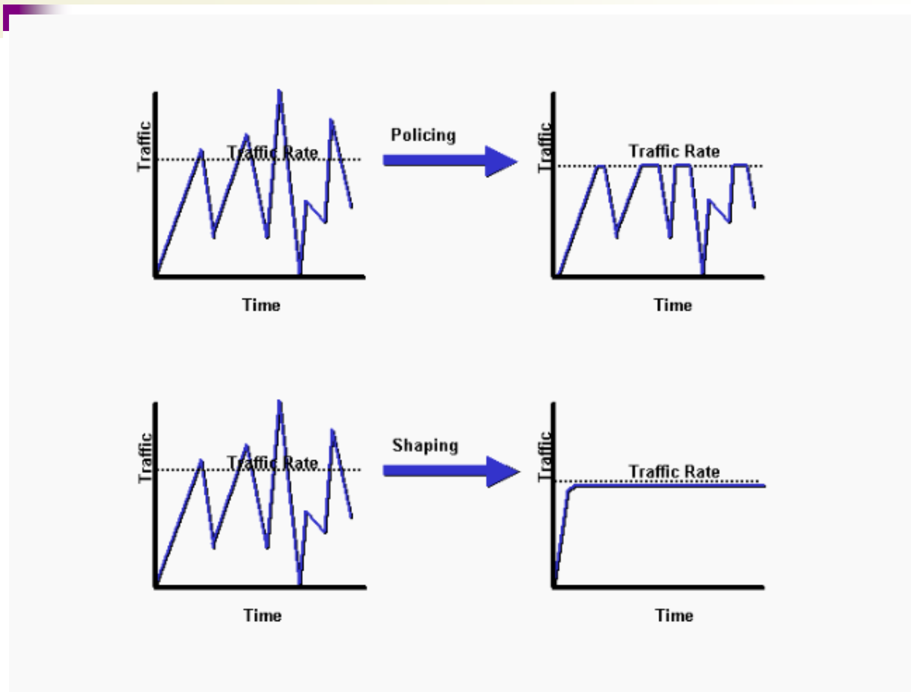
Buffering



Traffic Shaping

- In previous example data transmitted with uniform spacing
- Some applications' data is nonuniform
  - server handling multiple streams
  - allow fast forward, rewind video stream
- Traffic shaping: regulate average rate (burstiness) of data transmission
- According to Service Level Agreement (SLA)

Traffic Shaping



source: Cisco

Traffic Shaping

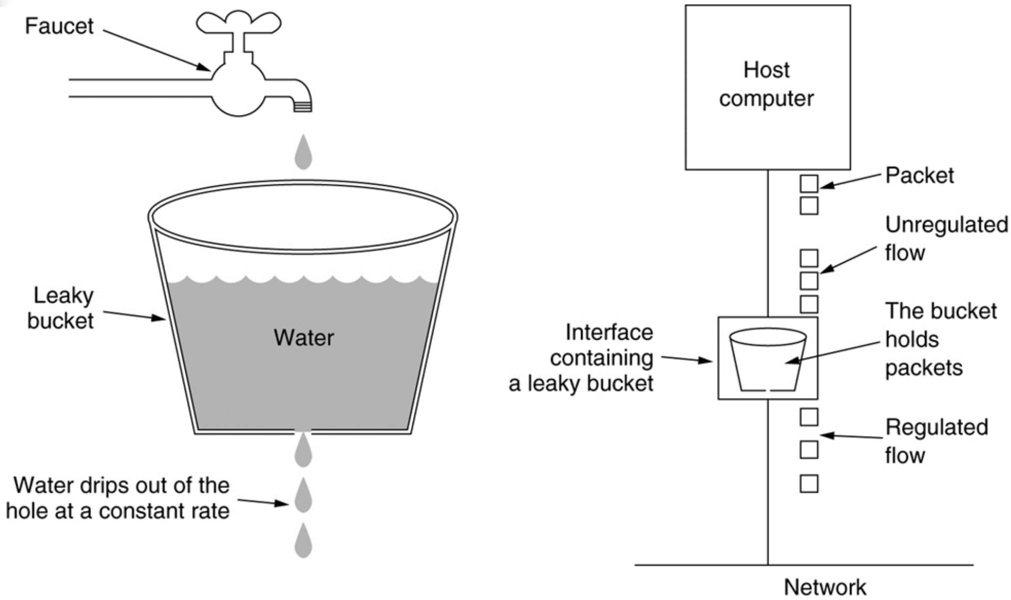
Traffic Policing

- When rate reaches max excess traffic is dropped
- Bursts are propagated
- Doesn't queue packets
- Applies to inbound traffic

Traffic Shaping

- Retain packets in queues, schedule excess for later retransmission
- Smoothed output rate
- Requires memory to queue packets
- Applies to outbound traffic
- Requires scheduling function FQ, WFQ, CBWFQ, …

The Leaky Bucket Algorithm



(a) (b)

No matter the rate at which water enters the bucket, the outflow is at a constant rate

The Leaky Bucket Algorithm

- Network interface has finite internal queue
- Simply it is a single server queueing system with constant service time
- Smooth bursts, reduce congestion chance
- Two approaches
  - packet-counting: for fixed-size packets (ATM)
  - byte-counting: for variable-size packets

The Leaky Bucket Algorithm

- Leaky bucket consists of finite queue
- When packet arrives
  - if there is a room in queue, append packet
  - else, discard packet
- At every clock tic, one packet is transmitted

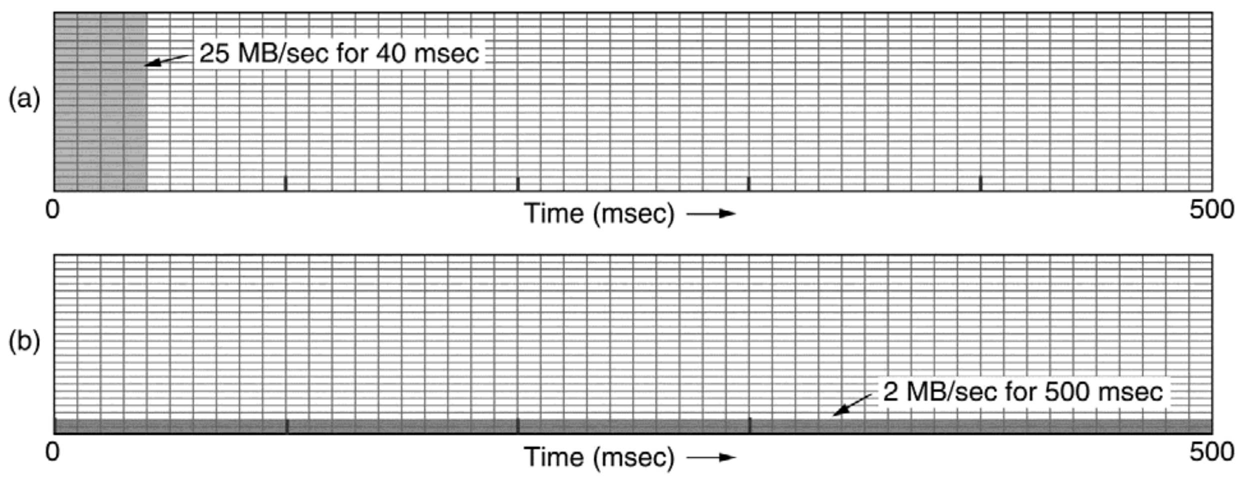
The Leaky Bucket Algorithm

- One variant: byte-counting leaky bucket
- For byte-counting leaky bucket
  - at each tic, counter initialized to n
  - while size of next packet < counter
    - transmit packet
    - counter = counter – size of packet

Example

- Computer send data @ 25 MB/s = 200 Mbps
- Network speed is 200 Mbps
- But routers can accept this rate only for short intervals, until buffer fill up
- For long intervals, they work best at 2 MB/s
- Suppose data come in 1 MB, 40 ms burst/s
- To reduce rate to 2 MB/s, use leaky bucket with rate ρ=2 MB/s & capacity C = 1 MB
- Output will be 2 MB/s for 500 ms = 1 MB

Example

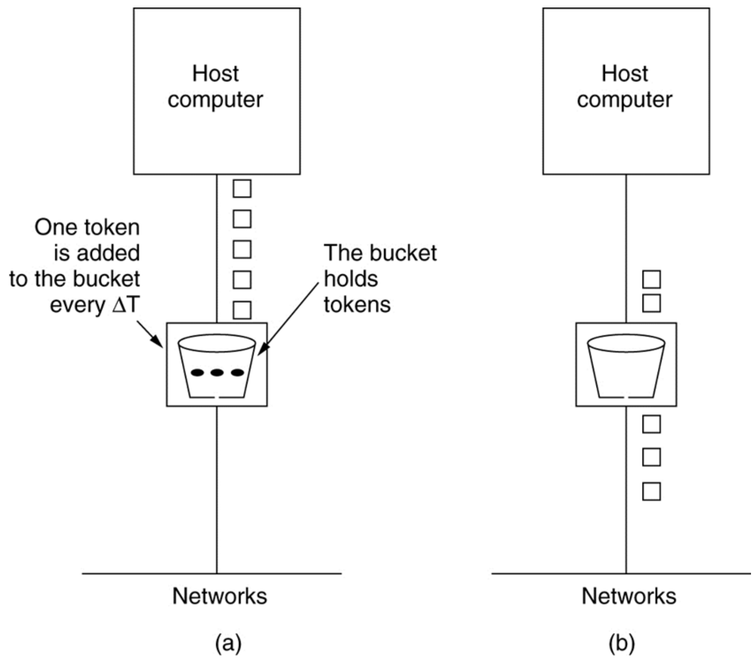


The Token Bucket Algorithm

- Leaky bucket has fixed output rate
- For some applications, better to allow output speed up when large burst arrives
- Token bucket
  - leaky bucket holds tokens
  - generated by a clock 1 token every ΔT sec
  - to transmit 1 packet, capture & destroy 1 token
  - can be used to regulate traffic at hosts
  - or smooth traffic between routers

The Token Bucket Algorithm

Essentially what the token bucket does is: allow bursts, but up to a regulated maximum length



The Token Bucket Algorithm

- Allows idle hosts to save permission to send large bursts later, up to max size of bucket *n*
  - leaky bucket doesn't allow this
- Bursts up to *n* packets can be sent at once
- Throws away tokens when bucket fills up
- But never discards packets
  - leaky bucket discards when bucket fills up

The Token Bucket Algorithm

- A minor variant, token represents *k* bytes instead of 1 packet
- Packet can be transmitted if enough tokens available to cover its size
- For packet counting token bucket
  - every ΔT , counter += 1
  - when packet sent, counter -= 1
- For byte counting token bucket
  - every ΔT , counter += k
  - when packet sent, counter -= packet size

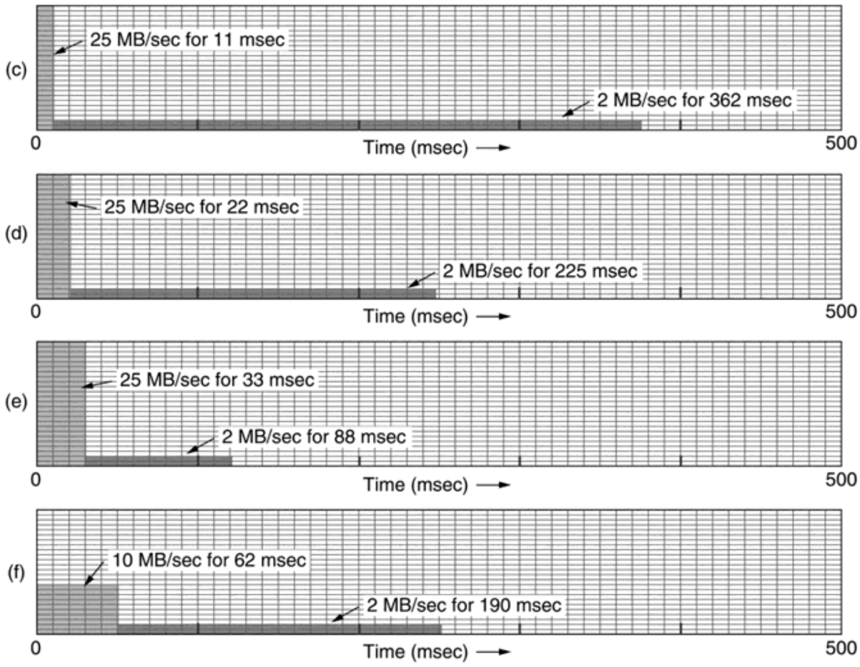
The Token Bucket Algorithm

Output of TB with C of
(c) 250 KB

(d) 500 KB

(e) 750 KB

(f) 500 KB feeding a
10 MB/s leaky bucket



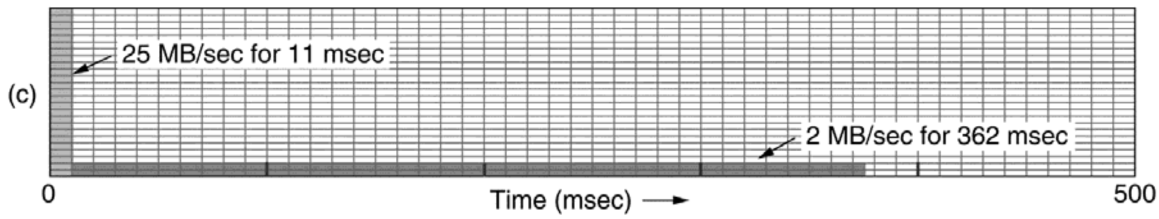
The Token Bucket Algorithm

- Calculating length of maximum rate burst
  - not 1 MB / 25 MB/s. Why?
  - during burst output, more tokens arrive
- Definitions
  - S: burst length (seconds)
  - B: token capacity (bytes)
  - R: token arrival rate (bytes/s)
  - M: maximum output rate (bytes/s)

The Token Bucket Algorithm

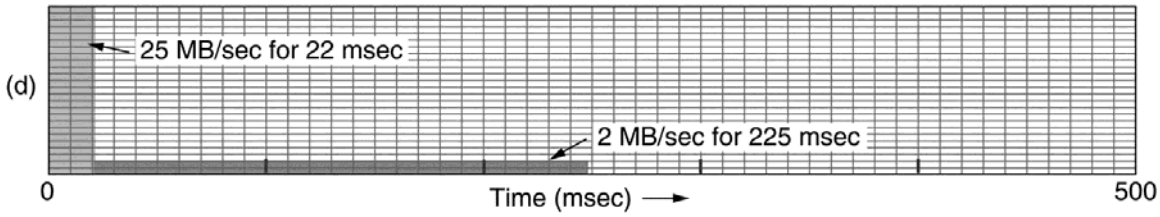
- We can see that
  - output burst contains max of $B + RS$ bytes
  - number of bytes in max-speed burst of length S is MS bytes
- Hence we have:
  - $B + RS = MS$
- To get S:
  - $S = \frac{B}{M-R}$

Example



- $B = 250{,}000$ Bytes $\quad M = 25 \times 10^6$Bps $\quad R = 2 \times 10^6$Bps
- Burst data = 10^6 Bytes
- $S = \frac{250\times10^3}{25\times10^6-2\times10^6} = 0.010869 \approx 11$ ms (we'll ***round*** to nearest ms)
- Bytes transmitted in burst $= MS = 25 \times 10^6 \times 0.011 = 275000$
- Remaining byte in burst = $10^6 - 275000 = 725000$
- Time to transmit remaining bytes using token arrival rate:
- $= \frac{725000}{2\times10^6} = 0.3625 \approx 363$ ms
- Total time to transmit 1 MB = 11+363 = 374 ms

Example

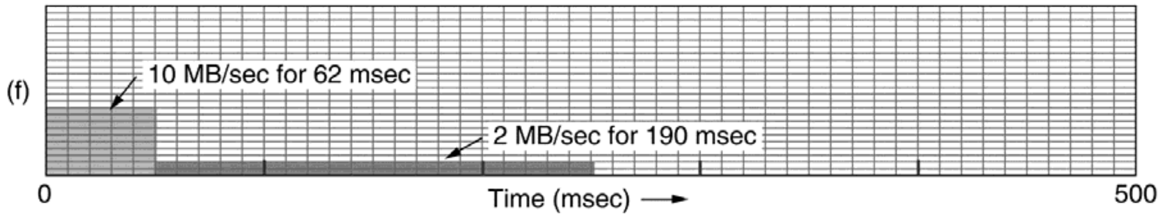


- $B = 500{,}000$ Bytes $\quad M = 25 \times 10^6$Bps $\quad R = 2 \times 10^6$Bps
- Burst data = 10^6 B
- $S = \frac{500\times10^3}{25\times10^6-2\times10^6} = 0.021739 \approx$ 22 ms (we'll ***round*** to nearest ms)
- Bytes transmitted in burst $= MS = 25 \times 10^6 \times 0.022 = 550000$
- Remaining byte in burst = $10^6 - 550000 = 450000$
- Time to transmit remaining bytes using token arrival rate:
- $= \frac{450000}{2\times10^6} = 0.225 = 225$ ms
- Total time to transmit 1 MB = 22+225 = 247 ms

The Token Bucket Algorithm

- Problem: token bucket allows large bursts
- Frequently desirable to reduce peak rate
- One way: insert leaky bucket after TB
- LB rate R_2 > TB rate R_1 and < max rate M i.e. $R_1 < R_2 < M$
- This produces smoother traffic

Example



- $B = 500{,}000$ Bytes $R_2 = 10 \times 10^6$Bps $R_1 = 2 \times 10^6$Bps
- Burst data = 10^6 B
- $S = \frac{C}{R_2 - R_1} = \frac{500 \times 10^3}{10 \times 10^6 - 2 \times 10^6} = 0.0625$= 62.5 ms
- Bytes transmitted in burst =$R_2 S = 10 \times 10^6 \times 0.0625 = 625000$
- Remaining byte in burst = $10^6 - 625000 = 375000$
- Time to transmit remaining bytes using token arrival rate:
- $= \frac{375000}{2 \times 10^6} = 0.1875 = 187.5$ ms
- Total time to transmit 1 MB = 62.5+187.5 = 250 ms

Resource Reservation

- Need virtual circuit of something similar
- Otherwise, hard to guarantee anything
- Once route is fixed, can reserve resources
  - bandwidth: not oversubscribe output lines
  - buffer space: allocate queue sp for specific flow
  - CPU cycles: required to process packets
    - important to ensure CPU is not overloaded
    - even with load slightly below theoretical capacity, queue can build and delay occurs

Resource Reservation

- Assume packets arrive at random
  - mean arrival rate = λ packets/second
- Assume CPU time required is also random
  - mean processing rate = μ packets/second
- Both arrival & service: Poisson distribution
- From queueing theory, mean packet delay T
  - $T = \frac{1}{\mu} \times \frac{1}{1-\lambda/\mu} = \frac{1}{\mu} \times \frac{1}{1-\rho}$ seconds
  - where $\rho = \frac{\lambda}{\mu}$ is CPU utilization

slowdown due to competing flows

Service time with no competition

Example

- $\lambda = 950{,}000$ packets/s
- $\mu = 1{,}000{,}000$ packets/s
  - i.e. mean packet service time = $1\mu s$
- Then, $\rho = \frac{\lambda}{\mu} = 0.95$
- Mean packet delay
  - $T = \frac{1}{\mu} \times \frac{1}{1-\rho} = \frac{1}{1{,}000{,}000} \times \frac{1}{0.05} = 20\ \mu s$
    instead of $1\mu s$, because of queueing delay
  - if there are 30 routers, > $600\mu s$ of queueing delay

Admission Control

- Now we're at the point where traffic is
  - well shaped
  - can potentially follow single route, in which
  - capacity can be reserved on all routers on path
- When such flow is offered, admit or reject?
  - current capacity?
  - how many commitments made for other flows?
- Compare current (bandwidth, buffer, cycles) with excess capacity in those 3 dimensions?

Admission Control

- Decision is not that simple because
  - few apps know their CPU requirement, unlike bw
    - different way needed to describe flow
  - some apps more tolerant in occasional missed deadline than others
  - some apps may haggle about parameters, others may not
    - movie viewer can drop from 30 fps to 25 fps
    - similarly, pixels/frame, audio bps may be adjustable

Admission Control

- Many parties involved in flow negotiation
  - sender, receiver, all routers along path
- Flows must be described accurately
  - specific parameters that can be negotiated
  - called *flow specification*
- Sender produces flow specification
- Routers check and modify as needed
- At other end, parameters can be established

Admission Control

Parameter	Unit	Notes
Token bucket rate	Bytes/second	Max sustained rate sender may transmit averaged over long interval
Token bucket size	Bytes	If TB rate = 1 Mbps, TB size is 500 KB, bucket take 4 s to fill, after that tokens are lost
Peak data rate	Bytes/second	Must not exceed, even for short intervals
Minimum packet size	Bytes	Important because processing each packet takes some fixed time, no matter how short
Maximum packet size	Bytes	Important due to internal network limitations that may not be exceeded

Example

- How a router turns a flow specification into a set of specific resource reservations?
  - implementation specific, not standardized
- Suppose router can process 100,000 packets/s
- Offered flow 1 MB/s (1×2^{20} bytes/s)
- Min = max packet size = 512 bytes
- Thus, packet rate = $\frac{2^{20}}{512}$ = 2048 packets/s
- Router must reserve 2% CPU for that flow
- If policy is not allocate more than 50% CPU
- Router with 49% will reject the 2% flow

Example

- Tighter flow specification is more useful
- If spec states it needs TB rate 5 MB/s but packets vary between 50 and 1500 bytes
  - packet rate for max = $\frac{5 \times 2^{20}}{1500} \approx$ 3,500 packets/s
  - packet rate for min = $\frac{5 \times 2^{20}}{50} \approx$ 105,000 packets/s
- Router will reject because of latter number
- If min packet size was 1000, might accept

Proportional Routing

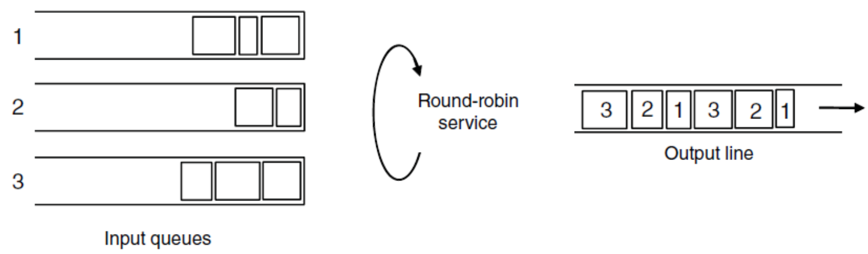
- Most routing algorithms try find best path
- Send all traffic over best path
- To provide higher QoS, split traffic for each destination over multiple paths
- Routers don't have complete overview of network traffic
- Divide traffic equally or in proportion to capacity of outgoing links

Packet Scheduling

- 1 flow may hog capacity, starve other flows
- Processing packets in order of arrival means aggressive sender can capture most capacity
- To prevent this, use packet scheduling
  - fair queueing
  - weighted fair queueing
  - ...

Packet Scheduling

- Fair queueing (packet-by-packet RR)
  - separate queues for each output line
  - one queue for each flow
  - when line is idle, scan queues in round robin
  - take first packet on next queue
  - helps in preventing aggressive flows
  - problem: flows with large packets take more BW



Packet Scheduling

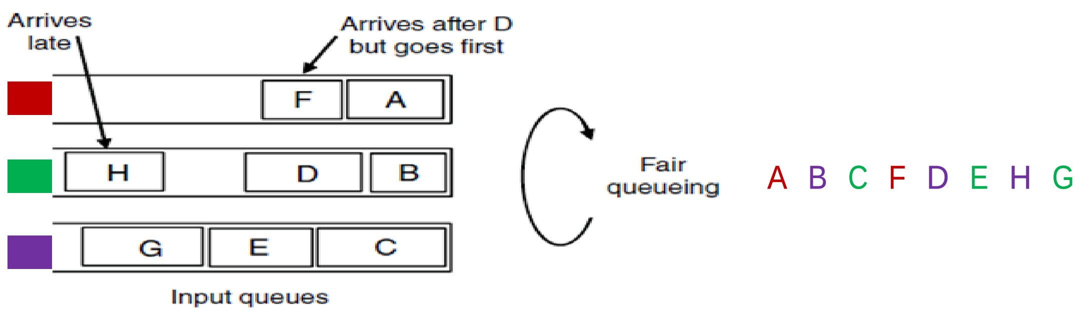
- Fair queueing (byte-by-byte RR)
  - solves the problem with packet-by-packet RR
  - simulate byte-by-byte round-robin
  - scan the queues repeatedly, byte-for-byte
  - find tick on which each packet will be finished
  - packets are then sorted
  - sent in order of their finishing

Packet Scheduling

- Fair queueing (byte-by-byte RR)
  - calculating finish time F_i for packet i
    - A_i: arrival time L_i: packet size
  - $F_i = Max(F_{i-1}, A_i) + L_i$
  - F_i is calculated independently for each queue

Example – Fair Queue

Packet	Arrival	Queue	Length	Finish (Calc)	Finish	Out order
A	0	1	8	Max(0, 0)+8	**8**	1
B	5	2	6	Max(5, 0)+6	**11**	2
C	5	3	10	Max(5, 0)+10	**15**	3
D	8	2	9	Max(8, **11**)+9	**20**	5
E	8	3	8	Max(8, **15**)+8	**23**	6
F	10	1	6	Max(10, **8**)+6	16	4
G	11	3	10	Max(11, **23**)+10	33	8
H	20	2	8	Max(20, **20**)+8	28	7

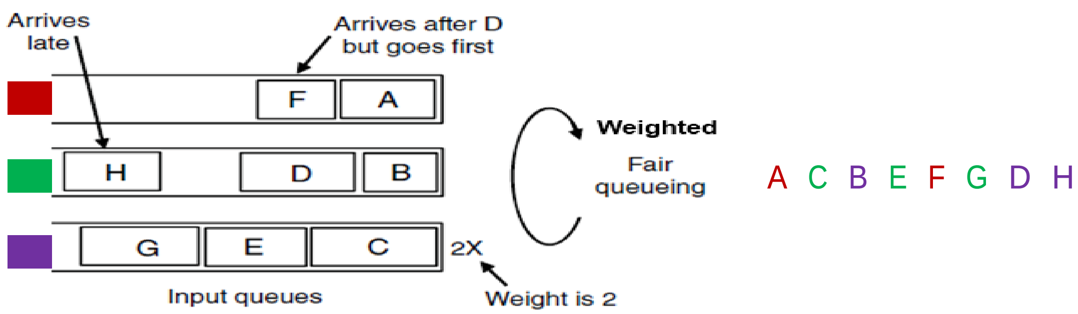


Packet Scheduling

- Fair queueing gives all flows same priority
- Many times, we need to give some flows more bandwidth (e.g. video)
- Weighted Fair Queueing (WFQ)
  - one queue for each flow
  - each flow has different weight
  - $F_i = Max(F_{i-1}, A_i) + \frac{L_i}{W_i}$
  - where W_i is the weight of the queue of flow i

Example – Weighted Fair Queue

Packet	Arrival	Queue	Length	Finish (Calc)	Finish	Out order
A	0	1	8	Max(0, 0)+8	**8**	1
B	5	2	6	Max(5, 0)+6	**11**	3
C	5	3	10	Max(5, 0)+10/2	**10**	2
D	8	2	9	Max(8, **11**)+9	**20**	7
E	8	3	8	Max(8, **10**)+8/2	**14**	4
F	10	1	6	Max(10, **8**)+6	16	5
G	11	3	10	Max(11, **14**)+10/2	19	6
H	20	2	8	Max(20, **20**)+8	28	8



Integrated Services

- Flow-based algorithms, integrated services
- Result of a lot of IETF efforts to devise an architecture for streaming multimedia
- Aimed at both unicast and multicast traffic
  - unicast: single user streaming video from site
  - multicast: collection of digital television stations broadcasting programs as streams of IP packets to many receivers at various locations
  - not suitable for dynamic group membership because bandwidth is reserved in advance

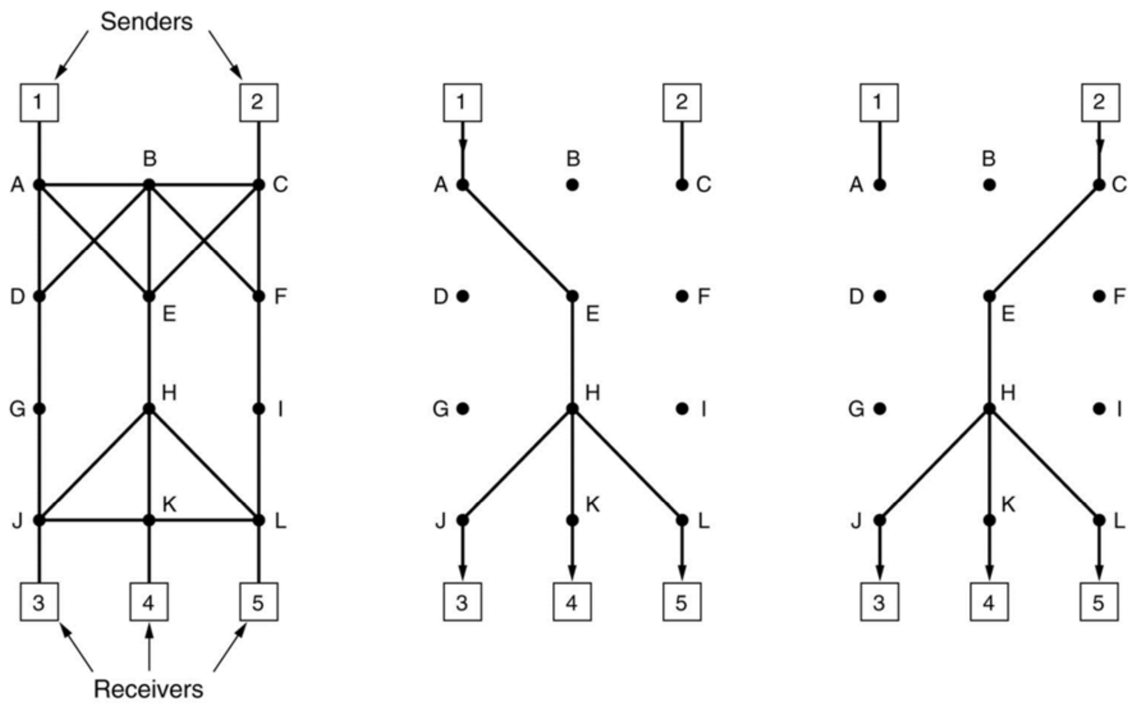
RSVP – Resource reSerVation Protocol

- Main part of IntServ architecture
- Described in RFCs 2205-2210
- Used for making reservations
- Allows multiple senders to transfer to multiple groups of receivers
- Individual receivers can switch channels
- Optimizes bandwidth use while eliminating congestion

RSVP – Resource reSerVation Protocol

- Multicast routing using spanning trees
- Each group is assigned group address
- Sender puts group address in its packets
- MC routing algorithm builds ST for group
- Routing algorithm not part of RSVP

RSVP – Resource reSerVation Protocol



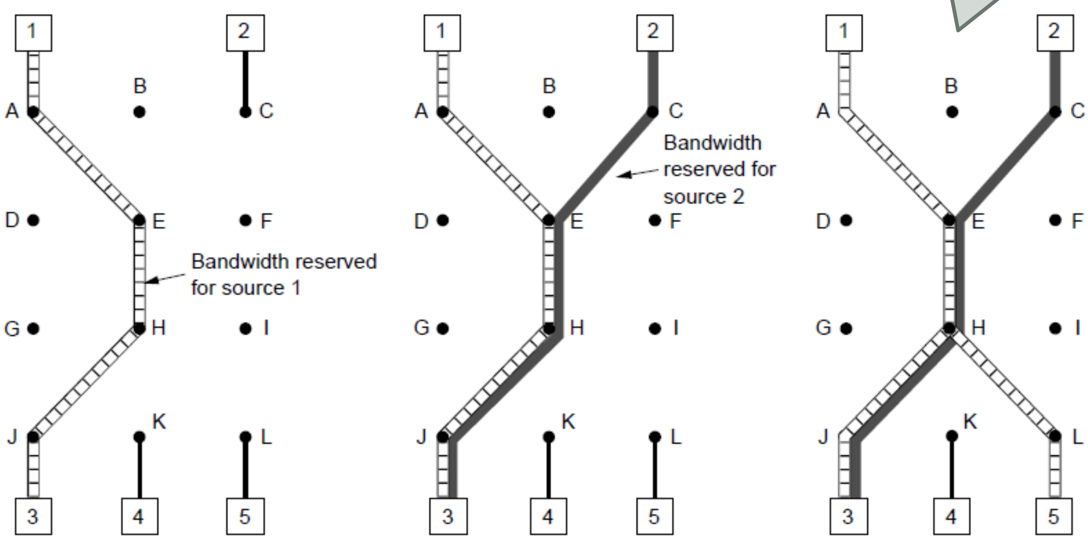
(a) (b) (c)

RSVP – Resource reSerVation Protocol

- To reduce cong, any receiver in group can send a reserv message up the tree to sender
- Message is propagated using reverse path forwarding (discussed earlier)
- Each hop, router reserves necessary BW
- If insufficient bandwidth, report failure
- When message gets back to source, BW is reserved all the way from sender to receiver

RSVP – Example

Capacity reserved must be large enough to satisfy the greediest receiver



R3 reserves flow from S1

R3 reserves flow from S2

R5 reserves flow from S1; merged with R3 at H

RSVP – Resource reSerVation Protocol

- Receiver can specify multiple sources
- Can specify if choices are fixed or dynamic
- Router use info for to optimize BW planning
  - two receivers are only set up to share a path if they both agree not to change sources later on
- BW reserv is decoupled from source choice
- Once BW reserved, can switch source, keep portion of reserved path for new source

Differentiated Services

- Flow-based protocols (IntServ) offer good QoS because needed resources are reserved
- The have downside:
  - require advance setup
  - don’t scale well for large networks w M’s flows
  - maintain per-flow state, vulnerable router crash
  - require substantial change to router code
- Thus, not widely implemented

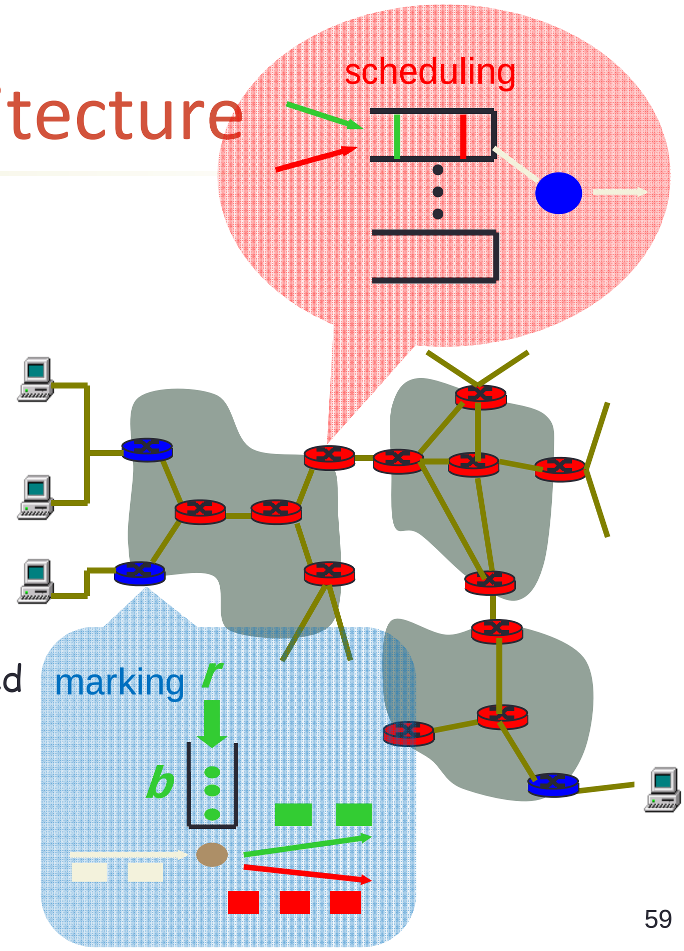
Differentiated Services

- Simpler approach to QoS devised by IETF
- Can be largely and easily implemented
- Class-based instead of flow-based QoS
- DiffServ or DS, RFCs 2474, 2475 and others
- Advantages
  - no advance setup
  - no resource reservation
  - no time-consuming end-to-end negotiation per flow as IntServ

Differentiated Services

- Can be offered by set of routers forming an administrative domain (ISP or Telco)
- Admin define set of service classes
- Customer signs up for DS
- Packets carry Type of Service (DSCP) field
- Better service for some (premium) classes
- Traffic in class may be required to conform to specific shape (e.g. LB)
- Extra charge if > N premium packets/month

Diffserv Architecture



Edge router:

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

Core router:

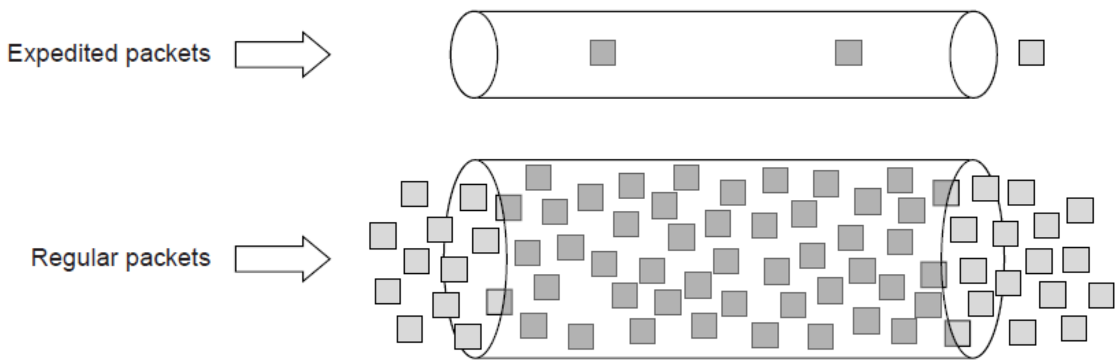
- per class traffic management
- buffering and scheduling based on marking at edge
- preference given to in-profile packets

From: Computer Networking: A Top Down Approach 5th edition. Jim Kurose, Keith Ross, Addison-Wesley, April 2009.

Differentiated Services

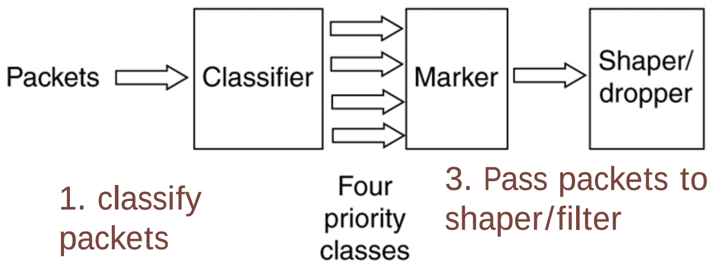
- Per-Hop-Behavior: PHB
  - Default PHP: best effort
  - Expedited Forwarding PHB
    - highest priority
    - reserved BW ≥ required
  - Assured Forwarding PHB
    - four priority classes
    - three discard probabilities for each class

Expedited Forwarding



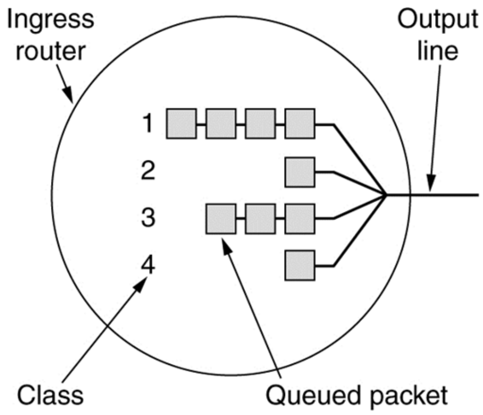
Assured Forwarding

2. Mark packets according to class



1. classify packets

3. Pass packets to shaper/filter



Sending host or Edge router

External References

- Comparing Traffic Policing and Traffic Shaping for Bandwidth Limiting, www.cisco.com/en/US/tech/tk543/tk545/technologies_tech_note09186a00800a3a25.shtml
- Course notes of Dr. Mohammed Arafah
- Fair Queueing, http://webpages.cs.luc.edu/~pld/courses/netmgmt/spr08/fair_queuing.html