eecs489 COMPUTER NETWORKS

Lecture 36:
QoS, Priority Queueing, VC, WFQ

Circuit Switching

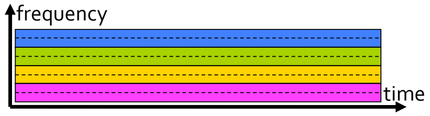
Network resources (e.g., bandwidth) divided into "pieces"

Pieces allocated to and reserved for calls

Resource idle if not used by owner (no sharing)

Ways to divide link bandwidth into "pieces"

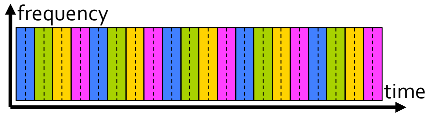
- frequency division multiplexing (FDM)



Example:

4 users 🟦 🟩 🟨 🟪

- time division multiplexing (TDM)



Packet Switching

~~Bandwidth division into "pieces"~~
~~Dedicated allocation~~
~~Resource reservation~~

Each end-to-end data stream divided into packets

Packets from multiple users share network resources

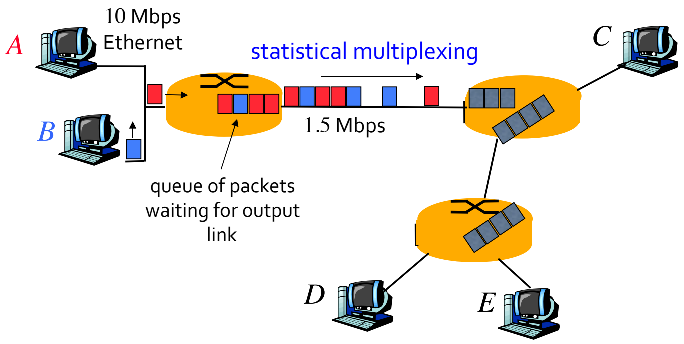
Each packet uses full link bandwidth

Resources used as needed

Resource contention:

- aggregate resource demand can exceed amount available
- congestion: packets queued, wait for link use
- store and forward: packets move one hop at a time
  - each node receives complete packet before forwarding

Packet Switching: Statistical Multiplexing



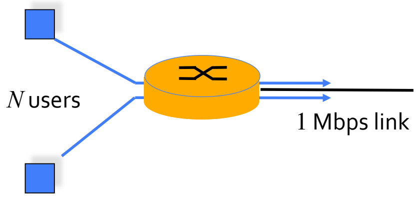
Sequence of A's and B's packets does not have a fixed pattern ⇒ statistical multiplexing

Packet vs. Circuit Switching

Packet switching allows more users to use network!

For example:

- 1 Mbps link
- each user:
  - sends 100 kbps when "active"
  - active 10% of time



circuit-switching: 10 users

packet switching: with 35 users, probability that more than 10 are active at the same time < .0004

Pros and Cons of Packet Switching

Advantages: great for bursty data

- resource sharing
- simpler, no call setup

Disadvantages: excessive congestion, packet delay and loss

- protocols needed for reliable data transfer
- congestion control
- no service guarantee: "best-effort" service

Better than Best-Effort Service

Approach: deploy enough link capacity such that congestion doesn't occur, traffic flows without queueing delay or overflow buffer loss

- advantage: low complexity in network mechanisms
- disadvantage: high bandwidth costs, most of the time bandwidth is under utilized (e.g., 2% average utilization)

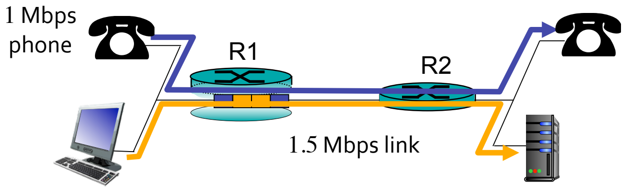
Alternative: multiple classes of service

- partition traffic into classes (not individual connections)
- network treats different classes of traffic differently

Example: HTTP vs. VoIP Traffic

1Mbps VoIP shares 1.5 Mbps link with HTTP

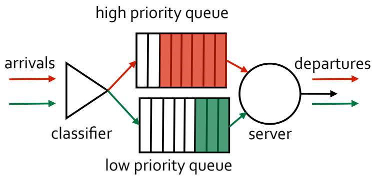
- HTTP bursts can congest router, cause audio loss
- want to give priority to audio over HTTP
  - packets can be differentiated by port number or
  - packets can be marked as belonging to different classes

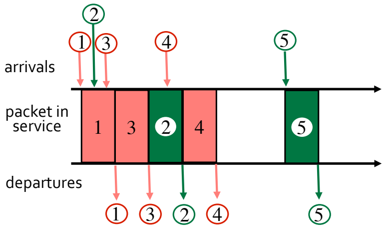


Priority Queueing

Send highest priority queued packet first

- multiple classes, with different priorities
- fairness: gives priority to some connections
- delay bound: higher priority connections have lower delay
- but within the same priority, still operates as FIFO, hence delay not bounded
- relatively cheap to operate ($O(\log N)$), N number of packets in queue



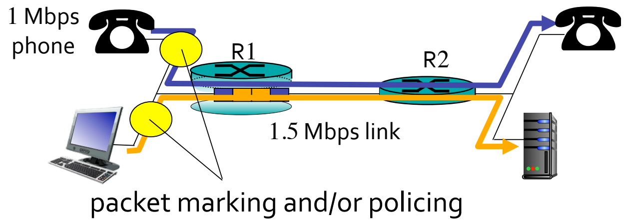


Traffic Metering/Policing

What if applications misbehave (VoIP sends higher than declared rate)?

Marking and/or policing:

- force sources to adhere to bandwidth allocations
- provide protection (isolation) for one class from others
- done at network ingress



Policing Mechanisms

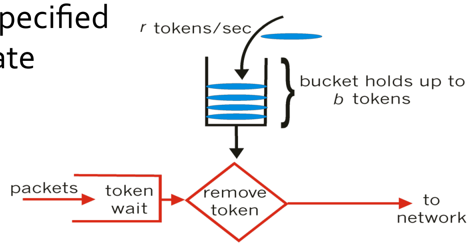
Goal: limit traffic to not exceed declared parameters

Three commonly used criteria:

1. average rate: how many packets can be sent per averaging time interval
    - crucial question: what is the averaging interval length?
    - 100 packets per sec or 6,000 packets per min have the same average!
2. peak rate: packet sent at link speed, inter-packet gap is transmission delay
    - e.g., 6,000 packets per min (ppm) avg.; 1,500 ppsec peak
3. (max.) burst size: maximum number of packets allowed to be sent at peak rate without intervening idle period

Token-Bucket Filter

Limit packet stream to specified burst size and average rate



- bucket can hold at most b tokens
- new tokens generated at the rate of r tokens/sec
- new tokens dropped once bucket is full
- packet can be sent only if there's enough tokens in buffer to cover it
- assuming 1 token is needed per packet, over interval of length t: number of packets metered out is $\leq (rt + b)$

Circuit vs. Packet Switching

Packet switching: data sent through the network in discrete “chunks”

Circuit switching: dedicated circuit per call

- end-to-end resources reserved for calls
  - link bandwidth, switch capacity
  - call setup required
- dedicated resources: no sharing
  - guaranteed performance
  - resource idle if not used by owner

Packet-Switched Networks

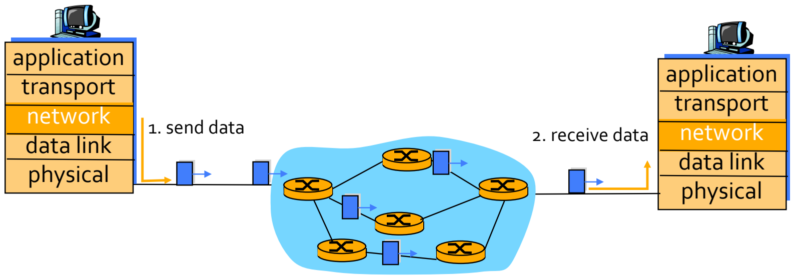
No call setup at network layer

No state to support end-to-end connections at routers

- no network-level concept of “connection”
- route may change during session

Packets forwarded using destination host address

- packets between same source-destination pair may take different paths



Pros and Cons of Packet Switching

Advantages: great for bursty data

- resource sharing
- simpler, no call setup

Disadvantages: excessive congestion, packet delay and loss

- protocols needed for reliable data transfer
- congestion control
- no service guarantee of any kind

How to provide circuit-like quality of service?

- bandwidth and delay guarantees needed for multimedia apps

Virtual Circuits (VC)

Datagram network provides network-layer connectionless service

VC network provides network-layer connection-oriented service

Analogous to the transport-layer services, but:

- service is host-to-host, as opposed to socket-to-socket
- implementation in network core

Source-to-destination path behaves much like a telephone circuit

- in terms of performance, and
- network actions along the path

Virtual Circuits

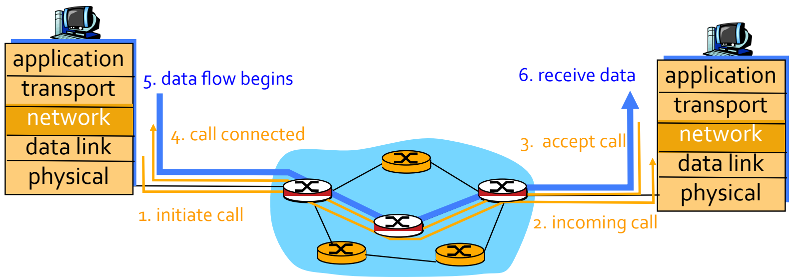
A VC comprises:

1. path from source to destination
   - each call must be set up before data can flow
     - requires signalling protocol
   - fixed path determined at call setup time, remains fixed throughout call
   - every router on path maintains state for each passing connection/flow
   - link, router resources (bandwidth, buffers) may be allocated to VC
2. VC numbers, one number for each link along path
   - each packet carries a VC identifier (not destination host address)
3. entries in forwarding tables in routers along path

Virtual Circuits

Signalling protocol:

- used to setup, maintain, teardown VC
- e.g., ReSource reserVation Protocol (RSVP)



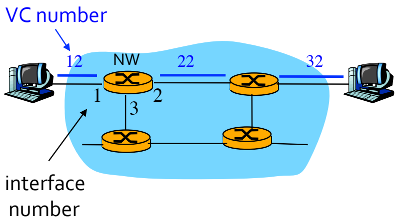
VC Forwarding Table

Packet belonging to a VC carries a VC number

VC number must be changed for each link

New VC number obtained from forwarding table

Examples: MPLS, Frame-relay, ATM, PPP



Forwarding table on router NW:

incoming interface	incoming VC#	outgoing interface	outgoing VC#
1	12	2	22
2	63	1	18
3	7	2	17
1	97	3	87
…	…	…	…

Routers maintain connection state information!

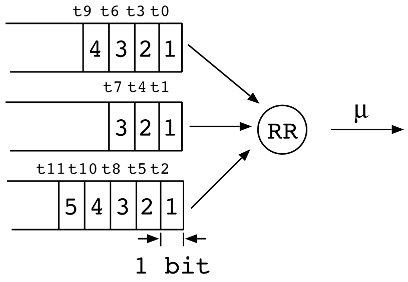
Per-VC Resource Isolation

To provide circuit-like quality of service

- resources allocated to a VC must be isolated from other traffic

Bit-by-bit Round Robin:

- cyclically scan per-VC queues, sending one bit from each VC (if present)
- 1 round, $R(\,)$, is defined as all non-empty queues have been served 1 quantum
  - $R(t_5) = 2$
  - time at Round 3? Round 4?

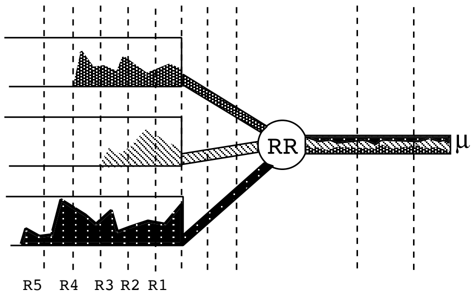


A.k.a. Generalized Processor Sharing (GPS)

Fluid-Flow Approximation

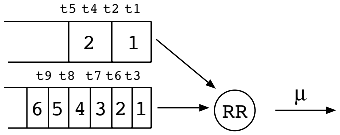
A continuous service model
- instead of thinking of each quantum as serving discrete bits in a given order
- think of each connection as a fluid stream, described by the speed and volume of flow

At each quantum the same amount of fluid from each (non-empty) stream flows out concurrently

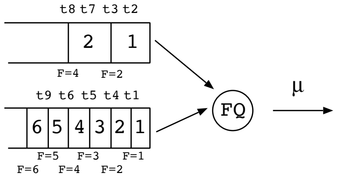


Packetized Scheduling

Packet-by-packet Round Robin:
- cyclically scan per-flow queues, sending one packet from each flow (if present)
- Problem: gives bigger share to flows with big packets



Packet-by-packet Fair Queueing:
- compute F: finish round, the round a packet finishes service
- simulates fluid-flow RR in the computation of F's
- serve packets with the smallest F first

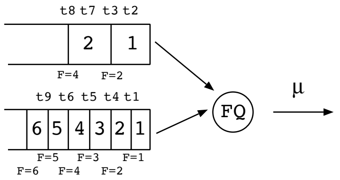


Start and Finish Rounds

When does packet i finish service?

$F^{\alpha}_{i} = S^{\alpha}_{i} + P^{\alpha}_{i}$,

where P^{α}_{i} is the service time (in rounds) of packet i and S^{α}_{i} the service start round



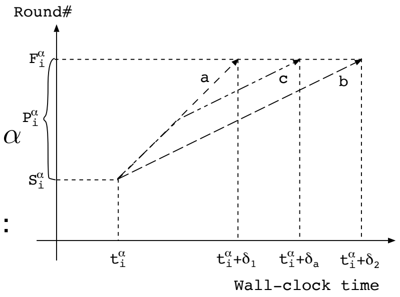
At what round does packet i of flow α start seeing service?

$S^{\alpha}_{i} = \mathrm{MAX}(F^{\alpha}_{i-1}, A^{\alpha}_{i})$

- $S^{\alpha}_{i} = F^{\alpha}_{i-1}$ if there is a queue, A^{α}_{i} otherwise
- $A^{\alpha}_{i} = R(t^{\alpha}_{i})$: round at the time packet i arrives

Round# vs. Wall-Clock Time

Let:
- time: wall-clock time
- round: virtual-clock time
- $\mu = 1$ unit
- t^{α}_{i}: arrival time of packet i of flow α
- $N_{ac}(t)$: #active flows at time t



Computing the rate of change:

a: $N_{ac} = 1$, $\partial R/\partial t = \mu/N_{ac}(t) = 1$,

b: $N_{ac} = 2$, $\partial R/\partial t = \tfrac{1}{2}$, $\delta_2 = 2*\delta_1$

c: at the beginning, $N_{ac} = 1$, $\partial R/\partial t = 1$,
halfway serving packet i, a packet belonging to another flow arrives, $N_{ac} = 2$, $\partial R/\partial t = \tfrac{1}{2}$

As $N_{ac}(t)$ changes, finish round stays the same, actual time stretches

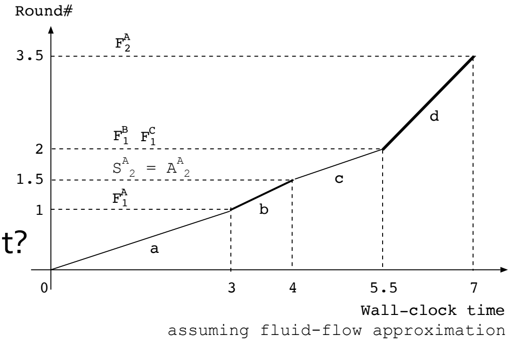
Round Computation Example

Scenario:

- flows A has 1 packet of size 1 arriving at time t_0
- flows B and C each has 1 packet of size 2 arriving at time t_0
- flow A has another packet of size 2 arriving at time t_4

Slope ($\partial R/\partial t$):
a = ⅓, b = ½,
c = ⅓, d = 1

What is the arrival round of A's 2nd packet?
$R(t^A{}_2) = 1.5$



Arrival Round Computation

When packet i of an active flow arrives, its finish round is computed as $F^\alpha{}_i = F^\alpha{}_{i-1} + P^\alpha{}_i$, where $F^\alpha{}_{i-1}$ is the finish round of the last packet in α's queue

If flow α is inactive, there's no packet in its queue, $F^\alpha{}_i = A^\alpha{}_i + P^\alpha{}_i$, how do we compute $A^\alpha{}_i$?

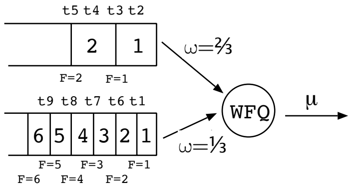
If flow α has been inactive for Δt time and there has been N_{ac} flows during the whole time, we can perform round catch up: $A^\alpha{}_i = F^\alpha{}_{i-1} + \Delta t(1/N_{ac})$

Iterated deletion: if N_{ac} has changed, one or more times, over Δt, round catch up must be computed in piecewise fashion, every time N_{ac} changes ⇒ expensive

Weighted Fair Queueing

Weighted-Fair Queueing (WFQ):

- generalized Round Robin
- each VC/flow/class gets weighted amount of service in each cycle
- $P^\alpha{}_i = L^\alpha{}_i/(\omega\mu)$, $L^\alpha{}_i$ size of packet



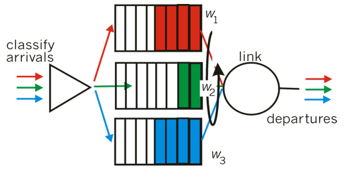
(Weighted) Fair Queueing

Credit accumulation:

- allows a flow to have a bigger share if it has been idle
- discouraged because it can be abused: accumulate credits for a long time, then send a big burst of data

Characteristics of (W)FQ:

- max-min fair
- bounded delay
- expensive to implement



Max-Min Fair

In words: max-min fair share maximizes minimum share of flows whose demands have not been fully satisfied

1. no flow gets more than its request
2. no other allocation satisfying condition 1 has a higher minimum allocation
3. condition 2 remains true as we remove the flow with minimal request

Max-Min Fair

Let:

μ_{total}: total resource (e.g., bandwidth) available

μ_i: total resource given to (flow) i

μ_{fair}: fair share of resource

ρ_i: request for resource by (flow) i

Max-Min fair share is $\mu_i = \mathrm{MIN}(\rho_i, \mu_{fair})$

$$\mu_{total} = \sum \mu_i, i = 1 \text{ to } n$$

Max-Min Fair Share Example

Let:

$\mu_{total} = 30$

i	ρ_i	μ_i
A	12	11
B	11	11
C	8	8

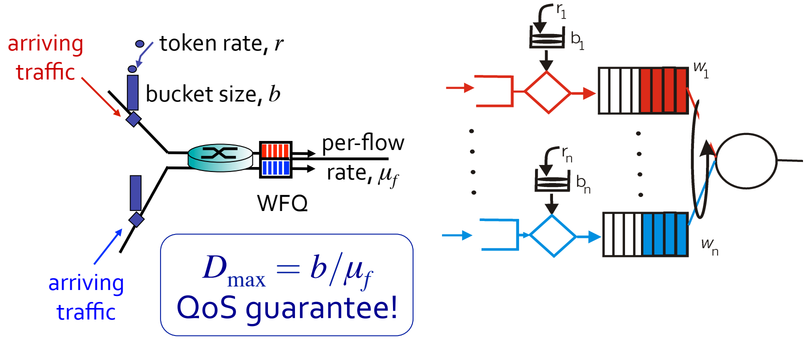
Initialy $\mu_{fair} = 10$

$\rho_C = 8$, so unused resource $(10 - 8 = 2)$ is divided evenly between flows whose demands have not been fully met

Thus, μ_{fair} for A and B $= 10 + 2/2 = 11$

Providing Delay Guarantee

Token bucket filter and WFQ combined provides guaranteed upper bound on delay

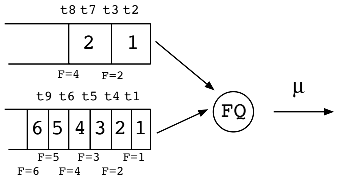


$$D_{\max} = b/\mu_f$$

QoS guarantee!

Same inefficiency issue as with circuit switching: allocating non-sharable bandwidth to flow leads to low utilization if flows don’t use their allocations

Limitations of (W)FQ



Round computation expensive: must re-compute R every time number of active flows changes

Unless packet transmission can be pre-empted, fairness is "quantized" by minimum packet size

- once a big packet starts transmission, newly arriving packets with smaller finish times must wait for completion of transmission
- flows with relatively smaller packets will suffer this more than flows with larger packets

Work Conservation

Work-conserving schedulers:

- doesn't go idle whenever there is packet in queue
- makes traffic burstier
- could require more buffer space downstream

Non-work conserving schedulers:

- only serve packets whose service times have arrived
- more work to determine whether packets' service times have arrived
- smooth out traffic by idling link and pacing out packets