Network's key role is to allocate its transmission resources to users or applications. Network does resource allocation (e.g., VCs) that can be wasteful of resources and difficult to do allocation of distributed resources. Congestion control vs. resource allocation: Two sides of the same coin. Let network do resource allocation (e.g., VCs) and sources send as much data as they want. It doesn't know about users or applications. How can a connectionless network allocate anything to a user? Connectionless Flows: A sequence of packets between the same source-destination pair following the same route. Flow is visible to routers but not a channel which is an end-to-end abstraction. Flow can be implicitly defined or explicitly established (similar to VC). Routers may maintain soft-state for a flow. Different from VC in that routing is not fixed. Queueing Policies: Many policies have been considered, e.g., FIFO (‘drop tail’), RED, virtual clock, and stochastic fair queueing (per class of traffic). Queueing Goals: Fairness, Efficiency, Stability, Service Differentiation. Queueing Dimensions: Fair to whom (flows, users, etc.), How quickly do you provide feedback, What kind of fairness do you provide, How much state you must keep, How much do you signal congestion. Service Differentiation: Class-based queueing (per class of traffic).
Router-centric: address problem from inside network - routers decide what to forward and what to drop

Router-centric v.s. Host-centric

Host centric: address problem at the edges - hosts observe network conditions and adjust behavior

Not always a clear separation: hosts and routers may collaborate, e.g., routers advise hosts

Reservations:

hosts ask for resources, network responds yes/no

implies router-centric allocation

Reservation-based v.s. Feedback-based

Feedback:

hosts send with no reservation, adjust according to feedback

either router or host centric: explicit (e.g., ICMP source quench) or implicit (e.g., loss) feedback

Flow control: advertised window

Window-based v.s. Rate-based

Both tell sender how much data to transmit

Window-based v.s. Rate-based

Congestion control: cwnd

Mostly host-centric, feedback, window based

In practice, fewer than eight choices

Service Models

Best-effort networks

Router-centric, reservation, rate-based

Networks with flexible Quality of Service

TCP as an example

Bandwidth: which packets get transmitted

Throughput: which packets get transmitted

Dropping: when packets get dropped

Drop-tail: drop policy (which packet to drop next)

First-Fit, First-Come-First-Served

FIFO: first-in-first-out (or FCFS: first-come-first-served)

Queueing Disciplines

Queueing discipline allocates:

buffer space: which packets get dropped

promptness: when packets get transmitted

FIFO:

scheduling discipline (which packet to serve next)

FIFO: first-in-first-out (or FCFS: first-come-first-served)
Some Issues

FIFO

Leaves resources and bandwidth hungry
Allocates buffer space and bandwidth fairly
Flow experience varies, but can share resources
Changes interaction between packets from different flows
Routing, services, queues in round-robin fashion
Allocates a separate group of resources, how currently running

Weighted Fair Queueing (WFQ)

Several granularities at which one can express flows
What constitutes a user?

No policing
Does not provide isolation between different flows
Combination with others
FIFO is like user gets more data through bit shares
Leave responsibility of congestion control to edges
FIFO + drop-tail is the simplest queuing algorithm

Variation: Weighted Fair Queueing (WFQ)

Fair Queueing

[Demers89a]

Bill Cheng

http://merlot.usc.edu/cs551-f12

Fair Queueing Illustration

No isolation between flows
Congestion with others
FIFO lets large user get more data through bit shares
Leave responsibility of congestion control to edges
FIFO + drop-tail is the simplest queuing algorithm

Dimensions

Per-connection
Class-based scheduling
Drop position
Random selection
Overflow selection
Enqueue

FIFO

Head
Tail
First in First out
Logical clock = number of rounds served

For packets:
- $P_i$: length, $A_i$: arrival time, $S_i$: begin transmit (start time)
- $F_i$: finish time

Router maintains a logical clock

Single flow: Suppose clock ticks when a bit is transmitted.

For packet $i$:
- $S_i = \max(F_{i-1}, A_i)$
- $F_i = S_i + P_i$

Multiple flows: Logical clock ticks when a bit from all active flows is transmitted.

$F_i = \max(F_{i-1}, A_i) + P_i$

Logical clock advances more slowly when there are more flows.

Bit-by-bit Round-Robin

Transmit earliest first

While we cannot actually perform bit-by-bit interleaving, we can compute the next packet if there are no more packets. We cannot compute the next packet if there are more packets.

Fair Queuing

Sort flows based on $x_i$ initially, assign $C/N$ to each flow.

$X_1, X_3$ e.g., three queues, $X, Y, and Z$, packet $X_1$ arrives at queue $X$ at real time 1, packet $X_3$ arrives at queue $X$ at real time 3, etc.

All packets are of size 1 ($P_i = 1, \forall i$), real arrival times are in real time.

Max-min Fairness

A fair service maximizes the service of the customer receiving the poorest service.

$x_i$ is the request of flow $i$

Total capacity C divided among N flows.

$X_1, X_3$ e.g., three queues, $X, Y, and Z$, packet $X_1$ arrives at queue $X$ at real time 1, packet $X_3$ arrives at queue $X$ at real time 3, etc.

Fair Queuing Example

How does the logical clock advance? How do you map real time to logical time?

Multiplying flows: Logical clock ticks when a bit from all active flows is transmitted.

$P_i$: Length, $A_i$: arrival time, $S_i$: begin transmit (start time)
Fair Queuing Example (Cont...)

Computer Communications - CSCI 551

Copyright © William C. Cheng

Packets:

Arrival/finish times:

$A_X1 = 1$

$A_Z1 = 1$

2 flows,

slope = $1/2$

Real time

Logical time

X1

Z1

Y2

X3

Z4

Y5

$F_X1 = F_Z1 = 2$

next arrival

is Y2 (arrives at real time 2)

next event on the X-axis is the next packet arrival

next event on the Y-axis is the next packet departure

To calculate next event:

1) line of best fit

2) next event will be a departure event at logical time $y_1$

next event will occur at $(x_1, y_1)$ where

$(y_1 - y_0)/(x_1 - x_0) = r$

solve for $y_1$, the logical arrival time of this arriving packet

from logical arrival time, you can easily calculate the logical finish time using the bit-by-bit RR equation

If next event will be a departure event at logical time $y_1$

next event will occur at $(x_1, y_1)$ where

$(y_1 - y_0)/(x_1 - x_0) = r$

solve for $x_1$, to make sure that there is no arrival between real time $x_0$ and $x_1$

verify that $y_1$ is the logical finish time of the departing packet

Current coordinate is $(x_0, y_0)$ and slope is $r$.
Aim: give less delay to those using less than their fair share

Delay Allocation

Computer Communications - CSCI 551
Copyright © William C. Cheng

Advance finish times for sources whose queues drain temporarily:

\[ B_i = P_i + \max(F_{i-1}, A_i - \delta) \]

Schedule earliest \( B_i \) first

If \( A_i < F_{i-1} \), conversation is active and \( \delta \) does not affect it:

\[ F_i = P_i + F_{i-1} \]

\( \delta \) gives added promptness:

If \( A_i > F_{i-1} \), conversation is inactive and \( \delta \) determines how much history to take into account.

FQ is a scheduling policy, not a drop policy.

Notes on FQ

Still achieves statistical multiplexing - one flow can fill entire pipe if no contenders - FQ is work conserving.

WFQ is a possible variation - need to learn about weights offline. Default is one bit per flow, but sending more bits is possible.

Weighted Fair Queuing Example

X1, X3

Weights for X, Y, and Z are 1, 2, and 1, respectively.

Y2, Y5

Z1, Z4

Z2, Z6

X1, X3

More Notes on FQ

Principle:

Still achieve statistical multiplexing - one flow can fill entire pipe if no contenders - FQ is work conserving.

WFQ is a possible variation - need to learn about weights offline. Default is one bit per flow, but sending more bits is possible.

Delay Allocation

\( f_{\text{out}} \) is a possible variation - need to learn about weights.

WFQ is a scheduling policy, not a drop policy.

Advantages:

If \( A_i > F_{i-1} \) conversation is active and \( \delta \) does not affect it:

Schedule earliest \( B_i \) first:

\[ B_i = P_i + \max(F_{i-1}, A_i - \delta) \]

Limitations:

If \( A_i < F_{i-1} \), conversation is active and \( \delta \) does not affect it:

\[ F_i = P_i + F_{i-1} \]

Advantage: less delay to those using less than their fair share.