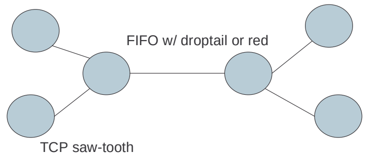
CSE561 - Lecture 08
QoS Network

Xiaowei Yang

The network we see so far



- Best-effort service
  - Web-surfing, email, ftp, file-sharing

Internet “Best Effort” Service

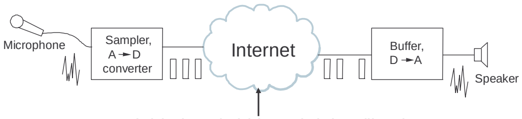
- Our network model so far:
  - IP at routers: a shared, first come first serve (drop tail) queue
  - TCP at hosts: probes for available bandwidth, causing loss
- The mechanisms at routers and hosts determine the kind of service applications will receive from the network
  - TCP causes loss and variable delay, and Internet bandwidth varies!
- Q: What kinds of service do different applications need?
  - The Web is built on top of just the “best-effort” service
  - Want better mechanisms to support demanding applications

Is best-effort good enough?

- How about IP telephony? Video conferencing, gaming, command control?

An Audio Example

- Playback is a real-time service in the sense that the audio must be received by a deadline to be useful



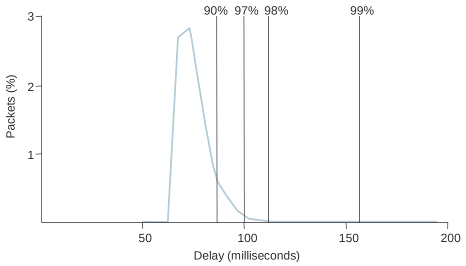
Variable bandwidth and delay (jitter)

- Real-time apps need assurances from the network
- Q: What assurances does playback require?

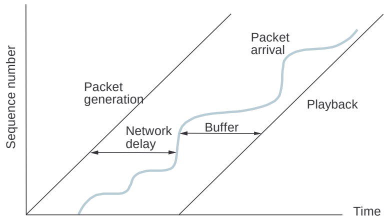
Network Support for Playback

- Bandwidth
  - There must be enough on average
  - But we can tolerate to short term fluctuations
- Delay
  - Ideally it would be fixed
  - But we can tolerate some variation (jitter)
- Loss
  - Ideally there would be none
  - But we can tolerate some losses

Example: Delay and Jitter

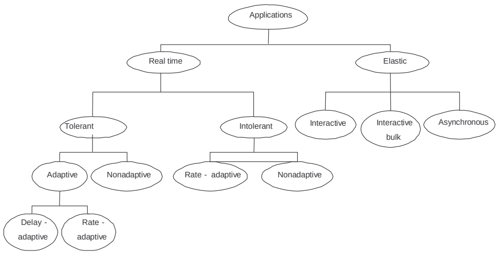


Tolerating Jitter with Buffering



- Buffer before playout so that most late sample will have arrived

Taxonomy of Applications



Problems to work out in this and next lecture

- What services do we want to provide?
- What building blocks are needed to provide them?
- QoS network: provide different levels of service for different types of applications

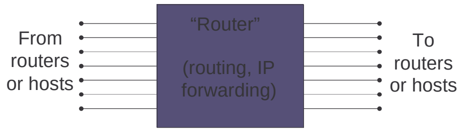
What do we want to provide?

- Guaranteed service
  - Rate: by guarantees we basically mean what rate an application can send packets into the network so that they will get through without loss
  - Delay: what delays these packets will experience
  - there are otherpossible forms of guarantee too such as limited jitter (variation in delay) but they are typically less important.

What building blocks do we need?

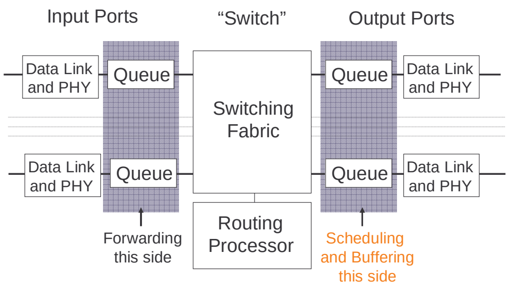
- What is it that FIFO does not provide? Look at previous picture of FIFO queue
  - I get some amount of bandwidth, but it depends on how many other flows are there in the network, and how fast they send
  - We we really want it to be independent of other sources of traffic that are using the network. That is the crucial aspect, some form of isolation between users so that the actions of another user cannot adversely impact me.

What's in a Router?



- By convention, draw input ports on left, output on right. (But in reality a single physical port handles both directions.)

Model of a Router



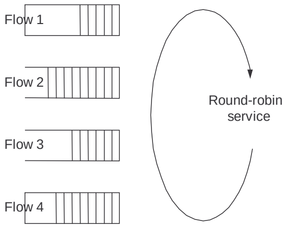
Scheduling and Buffer Management

- Two different functions implemented at the queue
- A scheduling discipline
  - This is the order in which we send queued packets
  - Examples: FIFO or priority-based
- A buffer management policy
  - This decides which packets get dropped or queued
  - Examples: Drop tail or random drop

Fair Queuing (FQ)

- FIFO is not guaranteed (or likely) to be fair
  - Flows jostle each other and hosts must play by the rules
  - Routers don't discriminate traffic from different sources
- Fair Queuing is an alternative scheduling algorithm
  - Maintain one queue per traffic source (flow) and send packets from each queue in turn
    - Actually, not quite, since packets are different sizes
    - 1000 bytes, 500 bytes two flows
    - 2/3 bandwidth, 1/3
  - Provides each flow with its "fair share" of the bandwidth

Fair Queuing

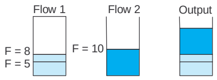


How to approximate bit-by-bit round robin

- Round robin bit streams (with two flows)
- Compute the finish time of a packet in the round robin case. And schedule the packet according to its finish time
- F_{i} = max(F_{i-1}, A_i) + P_i/rate
- WFQ:
  - Rates vary for different flows

Fair Queuing

- with finish times for each packet
  - finish (F) = arrive + length/rate; rate depends on # of flows
  - Send in order of finish times, except Want to share bandwidth
  - At the "bit" level, but in reality must send whole packets
- Approximate don't preempt (stop) transmission if a new packet arrives that should go first



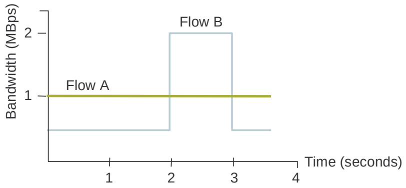
- More generally, assign weights to queues (Weighted FQ, WFQ)

What building blocks do we need?

- Scheduling: WFQ
- Applications need to tell the network their traffic characteristics

Specifying Bandwidth Needs

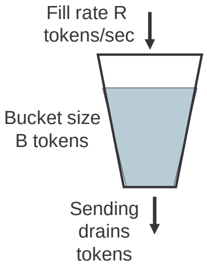
- Problem: Many applications have variable bandwidth demands



- Same average, but very different needs over time. One number. So how do we describe bandwidth to the network?

Token Buckets

- Common, simple descriptor
- Use tokens to send bits
- Average bandwidth is R bps
- Maximum burst is B bits



What building blocks do we need?

- Scheduling: WFQ
- Flowspecs: Applications need to tell the network their traffic characteristics
- Admission control & reservation
  - With a fifo queue, a (r,b) flow
  - With WFQ, what can we get?
    - Better, 1/N
  - With admission control
    - Say no when sum of rates exceed the capacity
    - 1/N * c > r

What building blocks do we need?

- Scheduling: WFQ
- Flowspecs: Applications need to describe their traffic characteristics
- Admission control

What can we provide?

- WFQ, (r,b) flow, admission control, reservation protocol
  - Single node, fluid model
    - Assign weight =Rate r, delay b/r
  - Multiple node, fluid model
    - Assign weight as before
    - Max delay jitter b/r, because you pay burst once
    - If constant bit rate at r, sees no queueing delay
    - (r * t + b) / r
- Guaranteed service
  - Rate, bounded delay

Key Concepts

- Different apps need different network support
  - Elastic versus real-time applications
  - Adaptation is a key technique, e.g, playout buffer
- Building blocks
  - Scheduling algorithm: wfq
  - Flow descriptor: token buckets
  - Admission control
- One way to implement QoS network
  - Guaranteed service network
  - Next: intserv & diffserv (IETF standards about how to provide QoS)