Quality of Service

Overview

- Why QoS? When QoS?
- One model: Integrated services
- Contrast to Differentiated Services (more modern; more practical; not covered)

What is QoS?

- Providing guarantees (or rough bounds) on various network properties:
  - Available bandwidth for flows
  - Delay bounds
  - Low jitter (variation in delay)
  - Packet loss

Service provider QoS goals

- Traffic classes for customers for differential pricing (“Gold”, “Silver”, …)
  - Gets particular Service Level Agreement (SLC) about b/w, delay, etc.
  - Costs more. ☺
- SLAs that specify rate guarantees, max rates, priorities, etc.
- Control who gets to use the network (admission control) (maybe, maybe not)

Why a New Service Model?

- What is the **basic objective** of network design?
  - Maximize total bandwidth? Minimize latency?
  - **Shenker argues: Maximize user satisfaction** – the total **utility** given to users
- What does utility vs. bandwidth look like?
  - Must be non-decreasing function
  - Shape depends on application

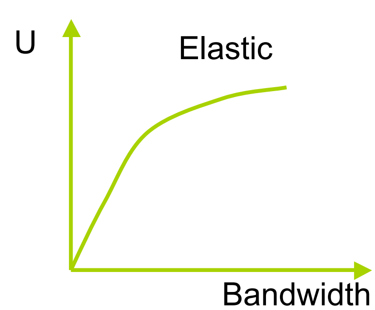
"Today": Elastic apps

- Internet currently (mostly) provides one single class of **"best-effort" service**
  - No assurances about delivery
- Most existing applications are ***elastic***
  - Tolerate delays and losses
  - Can adapt to congestion
- Some "real-time" applications are ***inelastic***

Inelastic Applications

- Continuous media applications
  - **Lower and upper limit** on acceptable performance.
  - BW below which video and audio are not intelligible
  - Internet telephones, teleconferencing with high delay (200 - 300ms) impair human interaction
- Hard real-time applications
  - Require **hard limits on performance**
  - E.g. control applications
- Claim: These apps are not as elastic or adaptive. Don't typically react to congestion. This is a bit questionable, but telephony has some of these attributes.
- Note about jitter: More jitter == more buffering == delay + memory.

Utility curve – Elastic traffic

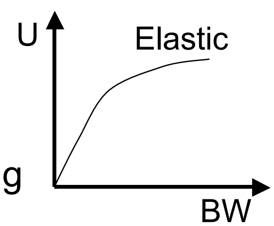


Does equal allocation of bandwidth maximize total utility?

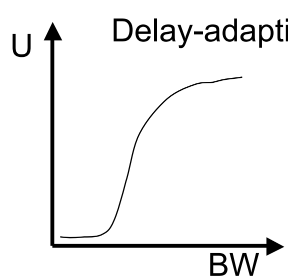
Admission Control

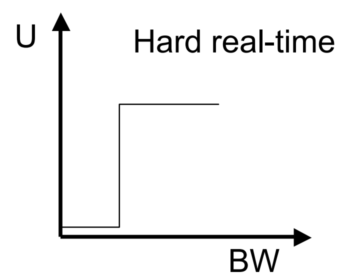
- If U(bandwidth) is concave → elastic applications
  - Incremental utility is decreasing with increasing bandwidth
  - Is always advantageous to have more flows with lower bandwidth
    - No need of admission control;

  This is why the Internet works!



Utility Curves – Inelastic traffic

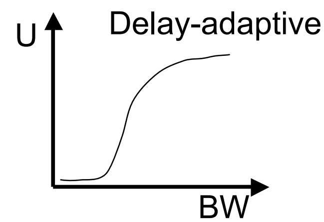




Does equal allocation of bandwidth maximize total utility?

Admission Control

- If U is convex → inelastic applications
  - U(number of flows) is no longer monotonically increasing
  - Need admission control to maximize total utility
- **Admission control** → deciding when the addition of new people would result in reduction of utility
  - Basically avoids overload



So?

- Right answer depends on a lot of factors:
  - Cost of complexity vs. cost of bandwidth
  - Can applications become adaptive?
- Well worth thinking about!
  - Even if the answer is "best effort is mostly okay"
- Important features:
  - Maximizing V doesn't necessarily maximize Ui
    - In fact, it almost can't! It takes away from elastic Us to add to inelastic Us
  - Keep in mind: Only so much you can do if underprovisioned
    - Much depends on the cost of adding b/w vs. the user benefit
      - Should you add capacity to support traffic? (VoIP? BitTorrent?)
  - Internet economics are not directly passed on to customer
    - Makes some economic models of reservations hard

Components of Integrated Services

1. **Type of commitment**
   What does the network promise?
2. Packet scheduling
   How does the network meet promises?
3. Service interface
   How does the application describe what it wants?
4. Establishing the guarantee
   How is the promise communicated to/from the network
   How is admission of new applications controlled?

1. Type of commitment

What kind of promises/services should network offer?

➡ Depends on the **characteristics of the applications** that will use the network ….

Playback Applications

- Sample signal → packetize → transmit → buffer → playback
  - Fits most multimedia applications
- Performance concern:
  - Jitter – variation in end-to-end delay
    - Delay = fixed + variable = (propagation + packetization) + queuing
- Solution:
  - Playback point – delay introduced by buffer to hide network jitter

Characteristics of Playback Applications

- In general lower delay is preferable.
- Doesn't matter when packet arrives as long as it is before playback point
- Network guarantees (e.g. bound on jitter) would make it easier to set playback point
- Applications can tolerate some loss

Application Variation

- Rigid & adaptive applications
  - Rigid – set fixed playback point
  - Adaptive – adapt playback point
    - Gamble that network conditions will be the same as in the past
    - Are prepared to deal with errors in their estimate
    - Will have an earlier playback point than rigid applications
- Tolerant & intolerant applications
  - Tolerance to brief interruptions in service
- 4 combinations

Applications Variations

Really only two classes of applications

1) **Intolerant and rigid**
2) **Tolerant and adaptive**

Other combinations make little sense

3) Intolerant and adaptive
   - Cannot adapt without interruption
4) Tolerant and rigid
   - Missed opportunity to improve delay

So what service classes should the network offer?

Type of Commitments

- **Guaranteed** service
  - For **intolerant and rigid** applications
  - Fixed guarantee, network meets commitment as long as clients send at match traffic agreement
- **Predicted** service
  - For **tolerant and adaptive** applications
  - Two components
    - If conditions do not change, commit to current service
    - If conditions change, take steps to deliver consistent performance (help apps minimize playback delay)
    - Implicit assumption – network does not change much over time
- **Datagram/best effort service**

Components of Integrated Services

1. Type of commitment

   What does the network promise?

2. **Packet scheduling**

   How does the network meet promises?

3. **Service interface**

   How does the application describe what it wants?

4. Establishing the guarantee

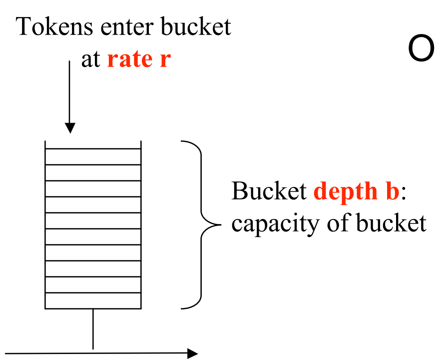
   How is the promise communicated to/from the network

   How is admission of new applications controlled?

Scheduling for Guaranteed Traffic

- Use **token bucket filter** to characterize traffic
  - Described by rate *r* and bucket depth *b*
- Use **WFQ** at the routers
- Parekh's bound for worst case queuing delay = b/r

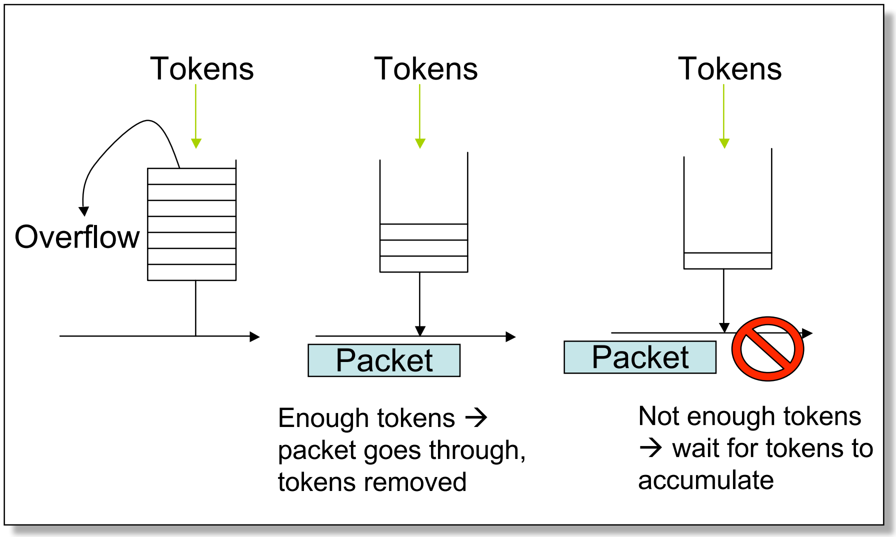
Token Bucket Filter



Operation:

- If bucket fills, tokens are discarded
- Sending a packet of size P uses P tokens
- If bucket has P tokens, packet sent at max rate, else must wait for tokens to accumulate

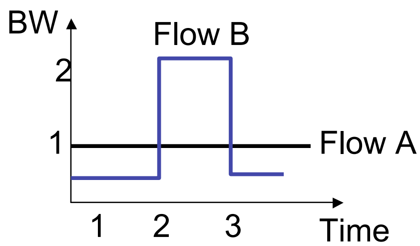
Token Bucket Operation



Token Bucket Characteristics

- On the long run, rate is limited to r
- On the short run, a burst of size b can be sent
- Amount of traffic entering at interval T is bounded by:
  - Traffic = b + r*T
- Information useful to admission algorithm

Token Bucket Specs



Flow A: r = 1 MBps, B=1 byte

Flow B: r = 1 MBps, B=1MB

Possible Token Bucket Uses

- Shaping, policing, marking
  - Delay pkts from entering net (shaping)
  - Drop pkts that arrive without tokens (policing)
  - Let all pkts pass through, mark ones without tokens
    - Network drops pkts without tokens in time of congestion

Guarantee Proven by Parekh

- Given:
  - Flow *i* shaped with token bucket and leaky bucket rate control (depth *b* and rate *r*)
  - Network nodes do WFQ
- Cumulative queuing delay D_i suffered by flow *i* has upper bound
  - $\mathbf{D_i < b/r}$, (where r may be much larger than average rate)
  - Assumes that Σr < link speed at any router
  - All sources limiting themselves to *r* will result in no network queuing

Predicted Service

Goals: Isolation
  - Isolates well-behaved from misbehaving sources
- Sharing
  - Mixing of different sources in a way beneficial to all

- Mechanisms: WFQ
  - Great isolation but no sharing
- FIFO
  - Great sharing but no isolation
- Principle: Mixing with FIFO shares jitter better than WFQ
- Reality: Complexity…

Predicted Service

- FIFO jitter increases with the number of hops
  - Use opportunity for sharing across hops
- FIFO+
  - At each hop: measure average delay for class at that router
  - For each packet: compute difference of average delay and delay of that packet in queue
  - Add/subtract difference in packet header
  - Packet inserted into queues expected arrival time instead of actual
    - More complex queue management!
- Slightly decreases mean delay and significantly decreases jitter

Key Principles of QoS

- Explicit vs. Implicit signaling
  - Explicit has proven very difficult, particularly w/unmetered pricing
    - Economic incentives are critical! ISPs must be able to profit from service differentiation, etc. Part of the reason IntServ didn't take off, but DiffServ has found some use.
- Isolation
  - Fair queueing + token buckets => e2e delays
- Jitter sharing
  - Benefits of stat mux. Helps reduce max jitter of one flow by slightly increasing jitter of all flows
- Admission control
  - Utility functions
- QoS vs. provisioning