CSE 123b
Communications Software

Spring 2002

Lecture 10: Quality of Service

Stefan Savage
Today’s class: Quality of Service

- What’s wrong with “Best Effort” service?
- What kinds of service do applications need?
- Integrated services architecture
- Differentiated service architecture
The Problem(s)

- Best effort service model (send and pray)
  - Statistical multiplexing provides efficient use of bandwidth for bursty applications, but...
  - No isolation during contention

- Smart hosts-dumb routers architecture
  - Congestion control at end-hosts
  - Resource control feedback occurs slowly (min 1 RTT)
  - Must trust end hosts to behave well
Motivation: Multimedia

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

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

Variable bandwidth and delay (jitter)
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

![Graph showing delay and packet loss](image-url)
Tolerating Jitter with Buffering

- Buffer before playout so that most late samples will have arrived
Application requirements

- Application variations
  - Rigid – fixed playback point (low jitter)
  - Adaptive – application can vary playback point (jitter ok)
  - Tolerant – can tolerate interrupt/degradation
  - Intolerant – can’t

- In reality only two classes
  - Rigid and intolerant applications (e.g. telemedicine)
  - Adaptive and tolerant (e.g. RealPlayer)
Another way to look at it...

- Elastic

Utility vs. Bandwidth

- Delay-adaptive

Utility vs. Bandwidth

- Hard real-time

Utility vs. Bandwidth
The Integrated Services solution

- **Change service model**
  - Multiple service classes; service specification
  - Guaranteed (send and pay), predicted, best-effort
  - Network directly supports *per-application* service requests

- **Change implementation**
  - Hosts explicitly reserve network capacity
  - Routers implement admission control
  - Routers use explicit **scheduling** mechanisms to provide *isolation* between admitted flows
  - Requires *per-flow state*
Intserv components

- Service classes
  - What are application demands?
  - What should the network promise?
  - What does the application promise?

- Service interface
  - How are service requirements described?

- Service mechanisms
  - How are service guarantees enforced?
Service interface

- Specify service class
- Specify “flowspec” for application data flow
  - Tspec: describes the flow’s traffic characteristics
    - Average bandwidth + burstiness
  - Rspec: describes the service requested from the network
    - Delay target
- Send request to network
  - Network can say “no”
  - If network says yes, then try to make sure it happens
Token Buckets

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

Fill rate R tokens/sec
Bucket size B tokens
Sending drains tokens
Service mechanisms

- Admission control protocol
  - RSVP

- Shaping/policing
  - Making sure that everyone only sends as much as they requested

- Per-flow scheduling policy in routers
  - Fair Queuing
    » Each flow gets equal access to bandwidth
  - Weighted Fair Queuing
    » Each flow gets access to bandwidth proportional to weight
Using Token bucket traffic conditioning

- Operation
  - Token bucket size: $b$ (maximum number of tokens)
  - Token bucket rate: $r$ (rate at which new tokens are provided)
- Need $k$ tokens to send a packet of $k$ bytes

- Applications
  - **Shaping** (delay packets until $k$ tokens appear)
    - Long term rate is limited to $r$, short term bursts to $b$
    - Over some interval $T$, traffic is limited to $b + r^*T$
  - **Policing** (drop packets if insufficient tokens)
  - **Buffer** management (mark packets and transmit if $> k$)
    - Drop marked (i.e. out-of-spec) packets during congestion
Token Bucket Operation

Tokens

Overflow

Tokens

Token Bucket Operation

Tokens

Packet

Enough tokens $\rightarrow$ packet goes through, tokens removed

Tokens

Packet

Not enough tokens $\rightarrow$ wait for tokens to accumulate
New elements in the router

- Admission Control
- Classifier
- Forwarding Table
- Per Flow QoS Table
- RSVP messages
- Data In
- Data Out
- Routing Messages
- Control Plane
- Data Plane
Integrated Services Example

- Example: guarantee 1MBps and < 100 ms delay to a flow
Integrated Services Example

- Allocate resources - perform per-flow admission control

Sender

Receiver

Yes, you may.

May I?
Integrated Services Example

- Install per-flow state
Integrated Services Example

- Install per flow state
Integrated Services
Example: Data Path

- Per-flow classification
Integrated Services
Example: Data Path

- Per-flow buffer management
Integrated Services Example

- Per-flow scheduling (weighted-fair-queuing: WFQ)
Guaranteed service

- Edge routers shape traffic according to $r,b$
- Routers implement WFQ
  - Weight set according to share $r$ is of total link capacity

- Parekh & Gallager 91,92 prove bound on delay is $b/r$!
  - Even if other flows don’t behave
- Why not make all service guaranteed service?
Issues?

- End-to-end approach
  - Inter-administrative domain issues
- Scalability (per-flow state)
- Efficiency
- Need to modify applications
- How much bandwidth does my application need?
Differentiated Services

- **Goal**
  - Provided different classes of service for traffic

- **Economic motivation**
  - Extract value through multi-tiered service (like airlines)
    - Best-effort service is bulk of traffic, but premium service important for capitalization
    - Aside: impact of flat rate vs usage-based pricing?
  - Bilateral-settlement, not end-to-end

- **Technical motivation**
  - Scalability/simplicity
  - No per-flow state; traffic-aggregates only
Basic architecture

- **Single network** (for now)
- **Edge routers**
  - Classify traffic as it enters network
  - Possibly shaping/policing to fit a user profile (pre-negotiated)
  - Marks packets to assign them to different traffic classes
- **Core routers**
  - Schedule/drop packets according to markings
  - State proportional to # classes; No per-flow state or signaling
- **The Network Engineer**
  - Provisions network so there is sufficient bandwidth to accommodate service
**Terminology**

- **Differentiated Services Code Point (DSCP)**
  - Particular value for 6 bit packet header indicating how packet should be handled; index into table

- **Behavior Aggregate (BA)**
  - Collection of packets on a link with the same DSCP

- **Per-Hop Behavior (PHB)**
  - Forwarding behavior applied by router to packets according to DSCP; not end-to-end

- **DS Domain**
  - Contiguous network using a common set of PHBs and provisioning policy
DS header field

6 bits stolen from ToS field (other two for ECN)
DiffServ architecture

- **Edge router**
  - Shape & police traffic
  - Set particular DiffServ Code Point (DSCP) in DS header field
- **Core router**
  - Implement Per-Hop Behavior for each BA
Standard PHBs

- **Expedited Forwarding** (Jacobson)
  - “Virtual Pipe” between ingress and egress routers
  - Pipe has capacity C (negotiated with ISP)
  - If user sends at rate \( \leq C \), network guarantees low delay and loss
  - If user sends \( > C \), excess traffic may be delayed or dropped

- **Assured Forwarding** (Clark)
  - “Better” service to anywhere
  - User profile indicates how much assured traffic is allowed
  - Up to that amount, network provides lower loss vs best-effort
  - Out-of-profile traffic is converted to best-effort
Traffic classified based on packet header (including DSCP)
Traffic Conditioning for Expedited Forwarding

- Shape packets according to user profile
- Marked outbound packets with EF to indicate premium service

![Diagram](image-url)

- Wait for token
- Set EF bit
- Drop on overflow

User Profile (token bucket)

- \( r \) bps
- \( b \) bits

Packet input → Wait for token → Set EF bit → Packet output
Traffic Conditioning for Assured Forwarding

- Meter packets according to user profile
- Mark in-profile packets with AF
- Out-of-profile packets send with AF cleared

Diagram:

- Packet input
- Test if token
- Set AF "in" bit
- No token
- User Profile (token bucket)
- b bits
- r bps
- Packet output
Core Router Scheduling/Buffer Management

- Two strict priority queues (one for EF, one for AF)
  - What if we used WFQ?
- Lower priority queue implements preferential dropping for in-profile AF traffic (RED with In/Out)

```
What DSCP?  EF  High-priority Q  Packets out
  ↓            ↓
  AF  If “in” set incr in_cnt  Low-priority Q
       ↓
       RIO queue management
       ↓
       If “in” set decr in_cnt
```
## Comparison to Best-Effort and Intserv

<table>
<thead>
<tr>
<th></th>
<th>Best-Effort</th>
<th>Diffserv</th>
<th>Intserv</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Service</strong></td>
<td>Connectivity</td>
<td>Per aggregate isolation</td>
<td>Per flow isolation</td>
</tr>
<tr>
<td></td>
<td>No isolation</td>
<td>Per aggregate guarantee</td>
<td>Per flow guarantee</td>
</tr>
<tr>
<td></td>
<td>No guarantees</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Service scope</strong></td>
<td>End-to-end</td>
<td>Domain</td>
<td>End-to-end</td>
</tr>
<tr>
<td><strong>Complexity</strong></td>
<td>No setup</td>
<td>Long term setup</td>
<td>Per flow steup</td>
</tr>
<tr>
<td><strong>Scalability</strong></td>
<td>Highly scalable</td>
<td>Scalable</td>
<td>Not scalable (each router maintains per flow state)</td>
</tr>
<tr>
<td></td>
<td>(nodes maintain only routing state)</td>
<td>(edge routers maintains per aggregate state; core routers per class state)</td>
<td></td>
</tr>
</tbody>
</table>
Summary

- Some applications need improved service
  - Multimedia, teleoperation, etc.

- Two approaches
  - IntServ: reserve bandwidth end-to-end, and schedule to provide guaranteed service
  - Diffserv: no guarantee, simply differentiate between different classes of service; treat some traffic better than other traffic

- Token bucket abstraction
  - Useful for describing and controlling traffic
For next time...

- Midterm
  - Sanjeev will have additional office hours this week
  - 4pm-5pm Wednesday: AP&M 3337A

- Keep working on projects
  - John-Paul will have additional office hours
  - 1pm-2pm Friday: AP&M 3337D