CSE561 – Quality of Service I

David Wetherall
djw@cs.washington.edu
QOS I (Fair Queueing)

- Focus:
  - How to provide “better than best effort”

- Leftovers: TCP
- Application needs
- Traffic shaping
- Fair queueing
TCP

- 88, Tahoe, slow-start and cong avoid, the original fixes
- 90, Reno, fast retransmit & fast recovery
  - recover from loss using duplicate ack signals w/o timeout
- 94, Vegas, experiment with delay-based signaling
- 95, NewReno, improved Reno for multiple losses
- 96, TCP with SACK, cleaner/better then NewReno

- '02 XCP, example of host and router control theory redesign
- >05 TCP BIC/CUBIC (Linux), modified cong avoid for LFN
- >05 Compound TCP (Microsoft), delay and loss based
- 06, TCP FAST, delay based, control theory
TCP w/ Slow Start + Cong Avoid (Tahoe)
TCP Tahoe + Fast Retransmit

djw // CSE 561, Spring 2010, L12
TCP Reno (Tahoe + F.Retrans/F.Rec)
QOS Framework

• QOS gives “stronger than best effort guarantees”. We need:

  • 1. understand what network services applications need
     – → network services
  • 2. characterize application traffic entering the network
     – → Flow specifications or SLAs
  • 3. decide whether to accept offered traffic
     – → admission control
  • 4. differentially process traffic in the network
     – → packet scheduling
Applications Needs

• May vary in terms of (typically) Bandwidth, Delay, Jitter, Loss
  – VoIP: low bandwidth and low delay/jitter, some loss OK
  – P2P: high bandwidth, high delay/jitter OK, no loss (transport)
  – Streaming: adequate bandwidth, high delay OK, jitter bad

• Leads to notion of network services:
  – Constant bit rate (CBR) real-time, e.g., VoIP
  – Variable bit rate (VBR) real-time, e.g., videoconference
  – Variable bit rate non-real-time, streaming movie
  – Available bit rate, e.g., P2P
Specifying Bandwidth Needs

• Problem: Many applications have variable bandwidth demands

![Graph showing 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

![Diagram of token bucket concept]

- Fill rate $R$ tokens/sec
- Bucket size $B$ tokens
- Sending drains tokens
# Network Roadmap – Various Mechanisms

<table>
<thead>
<tr>
<th>Simple to build, Weak assurances</th>
<th>Complex to build, Strong assurances</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIFO with Drop Tail</td>
<td>Classic Best Effort</td>
</tr>
<tr>
<td>FIFO with RED</td>
<td>Congestion Avoidance</td>
</tr>
<tr>
<td>Weighted Fair Queuing</td>
<td>Per Flow Fairness</td>
</tr>
<tr>
<td>Differentiated Services</td>
<td>Aggregate Guarantees</td>
</tr>
<tr>
<td>Integrated Services</td>
<td>Per Flow Guarantees</td>
</tr>
</tbody>
</table>
Fairer Queuing: Round Robin (Nagle)
Weighted Fair Queuing (WFQ)

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

\[
\begin{array}{ccc}
\text{Flow 1} & \text{Flow 2} & \text{Output} \\
F = 8 & F = 10 & \\
F = 5 & &
\end{array}
\]

- More generally, assign weights to queues (Weighted FQ, WFQ)
Deficit Round Robin (Varghese, 95)

- WFQ has complexity $O(\log N)$ to pick which packet goes next
  - Disadvantage for high speed implementation
- Deficit Round Robin is a $O(1)$ approximation
  - Fix the number of queues
  - Give them a quantum of service in round robin order
  - Skip queues until they build up enough credit for a large packet

- Gives both efficiency and fairness