Edge-Based Differentiated Services

Henrik Lundqvist, Ignacio Mas, Gunnar Karlsson
KTH, Royal Institute of Technology
Stockholm, Sweden
Outline

- Motivation and goals
- TCP fairness
- FEC and parameter settings
- Conclusions and future work
Motivation

- Increasing popularity of real-time applications over Internet
- Quality of service between any pair of hosts
- Simple deployment and low complexity
Goals

- Service differentiation without network support
- Not better or worse efforts, simply different
- Application chooses most appropriate control
- Non-blocking, variable rate for elastic traffic
  - Responds fast to changes in available capacity
  - Rate sharing between flows
- Blocking, fixed rate for streaming traffic
  - Consistency for admitted session
  - Responds slowly to changes in available capacity
EBDS Operation

- Elastic traffic use TCP
- Streaming traffic use probe-based admission control
- Sender probes path to receiver at peak rate, $r_{pr}$
  - Receiver selectively acknowledges packets
- UDP traffic is elastic as an aggregate
  - Flows come and go and might be blocked
  - Accepted calls provide consistent quality
    - In contrast to TCP friendly rate control
- Traffic control gives differentiation and error control gives isolation
  - FEC for streaming traffic
  - Tradeoff between source throughput and redundancy for channel protection
Fairness

- Threshold is set to not allow higher UDP than TCP rate
  
  \[ x_r = \text{MSS} \sqrt{\frac{w_r}{RTT^2 p}} \]

  \[ \Rightarrow \quad \text{tcp}_{-eq} = \frac{\text{MSS}^2 w_r}{RTT^2 U^2_{rate}} \]

- Also defines fairness between different UDP sessions
TCP Fairness Evaluation

- Load increased by higher number of UDP sessions
- “TCP constant” set to 1.0
- Different “shape” of TCP and UDP throughput
  No ideal fairness
Isolation by FEC

- Real-time applications have requirements on loss rate, delay and data rate
- Problem: admission threshold may be higher than tolerable loss
- Solution: loss recovery by forward error correction (FEC)
- FEC uses algebraic coding to generate redundancy information
- Extra packets sent over the network
- Receiver can recover lost packets using the redundancy information
Parameters setting

- Add redundancy if the rate of a streaming flow is lower than the TCP rate
  
  \[ tcp_{\text{eq}} = \frac{MSS^2 w_r}{RTT^2 U_{rate}^2} \]

- Increases the acceptance probability
Fairness between real-time sessions

- Threshold and redundant packets calculated before probing
- Sessions get the required loss rate
- Sessions with high demands are likely to be blocked

<table>
<thead>
<tr>
<th>Class</th>
<th>Rate (kb/s)</th>
<th>Loss req. %</th>
<th>Block lgh.</th>
<th>Thr. %</th>
<th>Red. pkts</th>
<th>Block prob. %</th>
<th>Failed %</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>500</td>
<td>0.5</td>
<td>40</td>
<td>0.8</td>
<td>2</td>
<td>98</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>500</td>
<td>2</td>
<td>-</td>
<td>1.4</td>
<td>0</td>
<td>80</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>300</td>
<td>0.5</td>
<td>10</td>
<td>1.8</td>
<td>2</td>
<td>55</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>300</td>
<td>0.5</td>
<td>40</td>
<td>2.2</td>
<td>3</td>
<td>29</td>
<td>0.4</td>
</tr>
<tr>
<td>4</td>
<td>300</td>
<td>2</td>
<td>10</td>
<td>2.1</td>
<td>1</td>
<td>34</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>300</td>
<td>2</td>
<td>40</td>
<td>2.3</td>
<td>2</td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>0.5</td>
<td>10</td>
<td>10.2</td>
<td>5</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>2</td>
<td>10</td>
<td>11.7</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Conclusions

- Fairness between applications with different requirements
- Simple deployment
- Promising results

Future work:
- Evolution from TCP fairness to other quality of service provisioning