Smart Queuing: An Adaptive Approach

By
Tedi Susanto & Jason Sze
{tsusanto,jszea}@cs.sfu.ca

Motivation
- Explosive growth of Internet
- Need to utilize network efficiently
- Control congestion & provide QoS
- Use queuing schemes
- Dynamic/chaotic Internet
- Answer: Smart Queuing

Queuing Schemes (1)
- FIFO
  - Simple and predictable
  - Cannot provide differentiated service
- Priority Queuing
  - Provide differentiated service
  - Starving of low priority traffic
Queuing Schemes (2)

- WFQ
  - Fair
  - Complex and low-speed
- Custom Queuing (CBQ)
  - Guaranteed output bandwidth for each class
  - Problem with misbehaving user in same class

Smart Queuing

- Maintain a set of parameter
  - Number of flows
  - Quality of Service
  - Rates, packet size
- Decide which queuing is the best
  - Fairness, loss, delay
- Dynamically switch to it

Implementation (1)

- Simple IP network
- UDP traffic with QoS
- Different characteristics
  - C1: 25 pkt/s (Hi Pr.)
  - C2: 25 pkt/s (Hi Pr.)
  - C3: 20 pkt/s (Low Pr.)
  - C4: 30 pkt/s (Hi Pr.) [misbehaving]
- Link capacity
  - 56 kbps, ~42 pkt/s

Clients Send Rates
**Smart Settings**

- Modify "ip_output_iface" process model
- Create multiple queuing structure (qm_info)
- Examine current traffic
- Is current method ideal?
  - No: enqueue to new method
- Synchronization issues

**Original ip_output_iface**

**Modified ip_output_iface**
Switching Module (do_stat)

- At each packet received record statistics
- If \( \frac{1}{(\text{curr}_\text{pkt}_\text{time} - \text{last}_\text{pkt}_\text{time})} > \text{allow}_\text{rate} \) then misbehave = true
- For each row, if \( \text{curr}_\text{pkt}_\text{time} - \text{last}_\text{pkt}_\text{time} > \text{timeout} \) then active = false
- Note: allow_rate is fixed according to tos

<table>
<thead>
<tr>
<th>source address</th>
<th>tos</th>
<th>last pkt time</th>
<th>allow rate</th>
<th>active</th>
<th>misbehave</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Switching Module (enqueue)

- Attempted
  - if curr_queue is not empty, put pkt to buffer
  - else if curr_queue is empty && buffer is not empty, flush buffer contents to next_queue
  - else put pkt to next_queue, set curr_queue = next_queue
- Implemented
  - if old queue is not empty, drop packet
  - else put pkt to next_queue, set curr_queue = next_queue
- dequeue: retrieve pkt from current queue and send

Switching Module (switch?)

- For all active users
  - if misbehave then use WFQ
  - else if users with multiple tos then use CQ
  - else use FIFO
- If need to switch
  - increment pkt_counter until it is greater than sensitivity, then switch
  - else reset pkt_counter

Smart Queuing In Action (1):
Packets send from each queue
Smart Queue In Action (2):
Packet dropped by each queue

Delay Comparison:
Client 1

Delay Comparison:
Client 2

Delay Comparison:
Client 3
Delay Comparison:
Client 4

Future Works

- More Verification
- Transition Buffer
- Switching:
  - Lookup tables, Pre-processing via simulation
- Traffic:
  - Incorporate TCP traffic, active queue management (RED)
- Queuing Mechanisms:
  - Incorporate other queuing schemes (i.e. DWRR, VClock)

Conclusion

- Is “Smart Queuing” better?
  - Yes!

References

  www.juniper.net/techcenter/techpapers/200020.html