Attacking the Latency Problem in Massive Multiplayer Games

Andreas Petlund, Knut-Helge Vik, Pål Halvorsen, Carsten Griwodz
Simula Research Laboratory, Norway
Email: griff@ifi.uio.no
Massively Multi-Player Online Games (MMOGs)

- **Immersive applications**
  - Virtual environments
  - Real-time user interaction
- **Growing user base**
  - 13 million subscribers in 2006
- **Broad usage**
  - Entertainment, education, military, business

- **Large number of concurrent users**
  - Diverse geographical locations
- **Decomposable systems**
  - Partitioning into virtual regions
- **Intermittent and shifting connectivity**
  - Activity of the user base is dynamic
  - Changes with time of day
• **Latency**
  
  – Impacts perceived quality
  – High latency delays event processing
• **Topic**
  - Latency reduction

• **Depicted scenario**
  - Geographically distributed servers
  - Skewed connectivity based on time of day
  - Physical location is unrelated to virtual location

• **Virtual regions**
  - Mark boundary for dependencies
  - Allow implicit specification of groups

---

**Traces from Funcom’s Anarchy Online**

- Single virtual region
- Server located in North America
- Spanning approximately one hour

---

**RTT statistics for all packets that are never retransmitted**

- North America
- Europe
- Asia

---

(simula research laboratory)
Thin TCP Streams
Transport protocols being developed for throughput-bound applications

BUT, there exist several **low-rate, time-dependent** applications

**Anarchy Online MMORPG Case Study**

- **average delay:** ~250 ms
- **max delay:** 67 seconds (6 retransmissions)
- **packets per second:** < 4 (less than one per RTT)
- **average packet size:** ~93 bytes
- **average bandwidth requirement:** ~1.8 Kbps
## Interactive thin streams over TCP

<table>
<thead>
<tr>
<th>Application</th>
<th>Average payload size (byte)</th>
<th>Packet interarrival time (ms)</th>
<th>Bandwidth requirements (bps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anarchy Online</td>
<td>93</td>
<td>909</td>
<td>1757</td>
</tr>
<tr>
<td>Counterstrike</td>
<td>142</td>
<td>81</td>
<td>19764</td>
</tr>
<tr>
<td>BZFlag</td>
<td>30</td>
<td>24</td>
<td>31370</td>
</tr>
<tr>
<td>Skype</td>
<td>111</td>
<td>30</td>
<td>37906</td>
</tr>
<tr>
<td>CASA (radar control)</td>
<td>175</td>
<td>7287</td>
<td>269</td>
</tr>
<tr>
<td>Windows remote desktop</td>
<td>111</td>
<td>318</td>
<td>4497</td>
</tr>
<tr>
<td>SSH text session</td>
<td>48</td>
<td>323</td>
<td>2825</td>
</tr>
<tr>
<td>MPEG-2 streaming</td>
<td>1460</td>
<td>3</td>
<td>(\sim4200000)</td>
</tr>
</tbody>
</table>

- Thin streams very often represent interactive (time-dependent) applications.
- Analysis shows that thin streams often show very high latencies when loss occurs.
Reasons

- **TCP**
  - congestion controlled
  - flow controlled
  - reliable
  - ordered

- **TCP’s assumptions**
  - all packet loss is congestion loss
  - packet loss at very slow speeds must mean that congestion is very bad

- **TCP’s actions**
  - For ordering
    - don’t deliver to application before errors are corrected
  - For reliability
    - retransmit lost packets
  - To avoid speed reduction
    - wait until it’s likely that a packet is lost (ACKs for 3 “younger” packets arrived)
    - timeout is a fallback
  - when no ACKs arrive
    - double timeout waiting time, retransmit again

- **Fast retransmit**
- **Exponential backoff**
Thin-stream detection

- The source of the extreme latencies:
  - Inability to make use of fast retransmission.
    - Fast retransmit (3 duplicate ACKs).
  - Consequently: “all” retransmissions by timeout.
    - Suffers from exponential backoff.

- We wanted to improve latency for such streams.

- Detecting thin streams:
  - Packets In Flight $\leq 4$
  - Bundling: $\text{size}_\text{unacked}(p1) + \text{size}(p2) < \text{MSS}$

- Apply modifications only when criteria are fulfilled.

- Make modifications available on a per-stream basis (using socket options)
Fast retransmit with thin-streams

− Thin streams often have < 1 packet per RTT.
− Before 3 dupACKs has arrived, a timeout will already have triggered a retransmission.
− When thin streams are detected, we trigger a FR after one dupACK.
Exponential backoff

When thin streams are detected, linear timeouts are used.

Lost packet
1. retransmission
2. retransmission
3. retransmission
4. retransmission
ENHANCEMENT:
Bundle all unacknowledged packets with each new transmission

- If a packet is lost, there is a large chance that it will arrive bundled with the next packet.
- The following ACK will acknowledge both segments.
- 😊 TCP standard compatible.
- 😞 Introduces inherent redundancy.
- 😊 But sends no extra packets.
Thin stream mechanism applicability

- From the properties we have discussed, we can derive four “classes” of streams

<table>
<thead>
<tr>
<th>High IA</th>
<th>Low IA</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="Small Packets" /></td>
<td><img src="#" alt="Large Packets" /></td>
</tr>
<tr>
<td>Typical thin stream RDB, retrans, backoff</td>
<td>Rare faster retransmit, backoff</td>
</tr>
<tr>
<td>Rare RDB</td>
<td>FTP, HTTP Thick</td>
</tr>
</tbody>
</table>
Test setup 1: Skype quality

- Three different audio clips from Skype using TCP.
  - Two instances of each clip:
    - Regular TCP
    - Using the presented modifications.

- Web page where users could compare audio clips and vote.

- As a reference question, for clip 2, we used the same clip twice.
  - To get an impression of the influence of the order of playing.
Data trace analysis: Skype

Transport layer delay  Application layer delay

![Graph showing CDF of latency for Skype with different RTTs and loss rates]
Skype survey results

Comparative tests mod/nomod

Clip 1

Mod
Nomod
Equal

Clip 3

Mod
Nomod
Equal

Reference test:
Same clip played twice

Clip 2

Var1
Var2
Equal
Internet latency improvements
Internet latency improvements
Internet latency improvements
Test setup 2: SSH text session

- Each user was given a set of tasks to perform using a text-based editor (vi or emacs) over a lossy network.
  - 2% loss, 130ms RTT

- The same tasks were to be performed twice.
  - The user was then asked to evaluate the quality of the text session with regard to noticeable delay and interruptions.

- We varied which TCP version (with- or without modifications) was used first.
Data trace analysis: SSH

Transport layer delay

Application layer delay

SSH session replayed: 6 hours - CDF, 2% loss, 130ms RTT

CDF (bytes)
Latency above minimum observed value (ms)

TCP with modifications
Regular TCP
Results from SSH survey

- 80.77% of the test subjects preferred the SSH session using the modified TCP
The road ahead

- Determining tradeoff points of balance for bundling:
  - Map tradeoff between redundancy and latency
  - Packet sizes at different RTTs

- Fairness effects:
  - Large number of modified-TCP streams over a bottleneck
  - Run tests mapping the consequences for each modification

- Community work:
  - We already have a working 2.6.23.8-patch
  - Invited to propose for 2.6.24-rt
  - Get feedback from different communities
Questions?

Thin vs Thick