Transport Protocols for Networked Games
TCP or UDP?
Why use TCP?

- TCP provides reliable, in-order delivery
- TCP goes through most firewalls, UDP does not
- TCP manages connection for us
Why not to use TCP?

• TCP incur higher latency
• Don’t always need reliability and in-order delivery
• High header overhead
position = 10  →
position = 13  → X
position = 15  →

Updated position not delivered to application until (outdated) lost packet is received
Gesture from someone far away need not be received reliably.
A study on ShenZhou Online shows that 46% of the bandwidth is occupied by TCP header
enet.cubik.org

A library that provides reliability, sequencing, connection managements over of
Delivery can be stream-oriented (like TCP) or message-oriented (like UDP)
Supports partial reliability

`enet_packet_create ("abc", 4, ENET_PACKET_FLAG_RELIABLE)`
Retransmission triggered by Timeout based on RTT
Data in queue are bundled into one packet if there is space
enet.cubik.org

Portable, easy to use, but still, most firewalls block UDP traffic
• MMORPG that uses **TCP**: WoW, Lineage I/II, Guild Wars, Ragnarok Online, Anarchy Online, Mabinogi
• MMORPG that uses **UDP**: EverQuest, SW Galaxies, City of Heroes, Ultima Online, Asherons Call, FFXI
Need to study the use of TCP for networked games
How to provide reliability over UDP?
How slow is TCP, really?

Which part of TCP is the root of slowness?
A Quick Review of TCP
TCP Delayed ACK
TCP Spec: max 500ms delay
Most implementation: 200ms
Why delay ACK?

- reduce num of ACKs
- in case receiver wants to send data within 200ms (in which case it can piggyback the ACK with data)
- give sender time to buffer more data for sending (avoid silly window syndrome)
TCP Fast Retransmission
Definition of Dup ACKs in 4.4BSD and Al Stevens: “pure ACK, cannot piggyback with data”
TCP Timeout + Exponential Back-off

TO

2x TO

TCP Timeout + Exponential Back-off
Spurious Retransmission
RTO estimation

\[ E_i = 7E_{i-1}/8 + \text{RTT}/8 \]

\[ V_i = 3V_{i-1}/4 + |\text{RTT}-E_{i-1}|/4 \]

\[ \text{RTO} = \max(E_i + 4V_i, 1\text{s}) \]
Linux’s RTO estimation

\[ E_i = \frac{7E_{i-1}}{8} + \frac{RTT}{8} \]

\[ V_i = \frac{3V_{i-1}}{4} + \frac{|RTT-E_{i-1}|}{4} \]

\[ RTO = \max(200\text{ms}, E_i + \min(V_i, 50\text{ms}) \]
Note:
Delayed ACK => increase RTT =>
increase RTO
Congestion Control
TCP Congestion Control
Congestion Window reset to 2 after an idle period (> RTO)
What does real game traffic look like?
About 4 packets / sec
Average Payload: 100 Bytes
Loss Rate 1%
But some experience 6 retransmissions
ShenZhou Online
Similar stats for other games
<table>
<thead>
<tr>
<th>application (platform)</th>
<th>payload size (bytes)</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>average</td>
<td>min</td>
<td>max</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anarchy Online (PC) ‡</td>
<td>98</td>
<td>8</td>
<td>1333</td>
<td></td>
<td></td>
</tr>
<tr>
<td>World of Warcraft (PC)</td>
<td>26</td>
<td>6</td>
<td>1228</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Counter Strike (PC) ‡</td>
<td>36</td>
<td>25</td>
<td>1342</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Halo 3 (Xbox 360) ‡ ‡</td>
<td>247</td>
<td>32</td>
<td>1264</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Halo 3 (Xbox 360) ‡ ‡</td>
<td>270</td>
<td>32</td>
<td>280</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gears of War (Xbox 360) ‡</td>
<td>66</td>
<td>32</td>
<td>705</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tony Hawk’s Project 8 (Xbox 360) ‡</td>
<td>90</td>
<td>32</td>
<td>576</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Drive Unlimited (Xbox 360) ‡</td>
<td>80</td>
<td>34</td>
<td>104</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

† For Halo 3 (beta version), we also show differences between intensity levels.
‡ The presented values are average values over all players (sending and receiving).

Table 1: Examples of game stream packet sizes.
<table>
<thead>
<tr>
<th>packet interarrival time (ms)</th>
<th>average</th>
<th>median</th>
<th>min</th>
<th>max</th>
<th>1%</th>
<th>99%</th>
<th>avg. bandwidth requirement (pps)</th>
<th>(bps)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>632</td>
<td>449</td>
<td>7</td>
<td>17032</td>
<td>83</td>
<td>4195</td>
<td>1.582</td>
<td>2168</td>
</tr>
<tr>
<td></td>
<td>314</td>
<td>133</td>
<td>0</td>
<td>14855</td>
<td>0</td>
<td>3785</td>
<td>3.185</td>
<td>2046</td>
</tr>
<tr>
<td></td>
<td>124</td>
<td>65</td>
<td>0</td>
<td>66354</td>
<td>34</td>
<td>575</td>
<td>8.064</td>
<td>19604</td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>33</td>
<td>0</td>
<td>1403</td>
<td>32</td>
<td>182</td>
<td>27.778</td>
<td>60223</td>
</tr>
<tr>
<td></td>
<td>67</td>
<td>66</td>
<td>32</td>
<td>716</td>
<td>64</td>
<td>69</td>
<td>14.925</td>
<td>35888</td>
</tr>
<tr>
<td></td>
<td>457</td>
<td>113</td>
<td>3</td>
<td>10155</td>
<td>14</td>
<td>8953</td>
<td>2.188</td>
<td>10264</td>
</tr>
<tr>
<td></td>
<td>308</td>
<td>163</td>
<td>0</td>
<td>4070</td>
<td>53</td>
<td>2332</td>
<td>3.247</td>
<td>5812</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>33</td>
<td>0</td>
<td>298</td>
<td>0</td>
<td>158</td>
<td>25.000</td>
<td>22912</td>
</tr>
</tbody>
</table>

Intensive (the upper row) and moderate (the lower row) action (ending minimum 1000 packets) within the period of the trace.

Net statistics per stream based on packet traces.
“Thin Streams”
Findings 1:
Fast retransmission rarely triggered
In ShenZhou Online traces, fail to trigger fast retransmission because insufficient dup ACK (50%) interrupted by data (50%)
Findings 2: Delay due mostly to timeout
Figure 9: Average latency of dropped packets
Findings 3: Congestion Window reset is frequent
12% - 18% of packets faces window reset
think..
think..
think..

click (tank attack here) →
click (missle launch there) →
click (charge soldiers) →

The last command is delayed as congestion window = 2
How to make TCP (or, transport protocol) go faster in these games?
1. Remove exponential backoff
TCP Timeout
2. Make RTO Smaller
make sure minimum RTO is not 1s
spurious retransmission is not disastrous
3. Make Fast Retransmit Faster
Retransmit after one duplicate ACK
4. Retransmission Bundling
Retransmit all unacknowledge data in queue
5. Redundant Data Bundling
Send any unacknowledged segment in queue as long as there is space. Lost data gets recovered in the next transmission before retransmission.
6. Turn off or reduce Delayed ACKs
Packet interarrival time on average > 200ms
(can’t combine two ACKs into one)
7. Combine Thin Streams into Thicker Stream
Transport for Games
Transport for Games

- remove exponential backoff
Transport for Games

• remove exponential backoff
• reduce RTO
Transport for Games

• remove exponential backoff
• reduce RTO
• make fast retransmit faster
Transport for Games

- remove exponential backoff
- reduce RTO
- make fast retransmit faster
- retransmit aggressively
Transport for Games

• remove exponential backoff
• reduce RTO
• make fast retransmit faster
• retransmit aggressively
• don’t delay ACK
Transport for Games

• remove exponential backoff
• reduce RTO
• make fast retransmit faster
• retransmit aggressively
• don’t delay ACK
• combine into thick streams