Lecture 11
Networked Game Traffic and Transport Protocol
Assignment 1
isolate traffic
payload size
histogram
activity pattern
periodic pattern
MOBA
RTS
FPS
RPG
UDP
bandwidth
in: 40 - 160 kbps
out: 15 - 40 kbps
payload size
in: 100 - 450 bytes
out: 50 - 150 bytes
packet rate
25 - 30 packets/seconds
gap in between: 30-40ms
TCP
lower bandwidth

in: 4.8 kbps
out: 6 kbps

an order of magnitude smaller than MOBA!
payload size
< 25 bytes

an order of magnitude smaller than MOBA!
packet rate

~10 packets/seconds
UDP
bandwidth
in: 20 - 100 kbps
out: 8 - 100 kbps

comparable to MOBA,
with higher outgoing throughput
slightly larger payload size

in: 50 - 300 bytes

out: 30 - 100 bytes
smaller packet rate

in: 20 - 120 packets/seconds
out: 10 - 90 packets / seconds

I expected this to be smaller
TCP
much lower bandwidth
in: 5 - 16 kbps
out: 1 - 8 kbps
larger payload size
in: 100 - 300 bytes
out: 20 - 160 bytes
smaller packet rate

in: 1 - 15 packets/seconds

out: 1 - 15 packets/seconds
1.6 packet / second

2 packet / second
What you found:

RPG have smaller packets and smaller update rate.
what about periodicity?
For many games, server updates are periodic. (50 - 200ms interval)
Summary
low bandwidth
small packets
low frequency
predictable
Both UDP and TCP are used.
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 incurs higher latency
- Don’t always need reliability and in-order delivery
- High header overhead
position = 10
position = 13
position = 15

Updated position not delivered to application until (outdated) lost packet is received
A’s position = 10
B’s position = 13
C’s position = 15

Some messages need not be delivered in sequence.
Gestures from someone far away need not be received reliably.
TCP header is >= 20 bytes high overhead for small packets (46% in Shenzhou Online)
https://github.com/lsalzman/enet

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

```c
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

Portable, easy to use, but still, most firewalls block UDP traffic
Need to study the use of TCP for networked games
Lessons are still useful to build enet-like UDP library
How slow is TCP, really?

Which part of TCP is the root of slowness?

Can we fix TCP?
A Quick Review of TCP
TCP Delayed ACK
TCP Spec: max 500ms delay
Most implementation: 200ms
3 dup ACKs within RTO – RTT: TCP Fast Retransmission
Definition of Dup ACKs in 4.4BSD and Stevens: “pure ACK with no data”
TCP Timeout + Exponential Back-off

RTO

2xRTO

TCP Timeout + Exponential Back-off
Spurious Retransmission
RTO estimation

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

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

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

\[ E_i = \frac{7E_{i-1}}{8} + \frac{RTT}{8} \]
\[ V_i = \frac{3V_{i-1}}{4} + \frac{|RTT - E_{i-1}|}{4} \]
\[ W_i = \min(V_i, 50\text{ms}) \]
\[ \text{RTO} = \max(200\text{ms}, E_i + W_i) \]
delayed ACK

\[\downarrow\]

increase RTT

\[\downarrow\]

increase RTO
Congestion Control
TCP Congestion Control
Congestion window resets to 2 after an idle period (> RTO)
What does real game traffic look like?
low packet rate
small packet size
“Thin Streams”
About 4 packets / sec
Average Payload:
100 Bytes
Loss Rate 1%
But some experience 6 retransmissions
Shen Zhou Online

http://tjgame.enorth.com.cn/images/200307/0903-1.jpg
The graph shows the cumulative distribution function (CDF) of payload sizes for client and server packets. The CDF is plotted against payload size in bytes. The red line represents client packets, and the blue dashed line represents server packets. The x-axis represents payload size in bytes, ranging from 0 to 500. The y-axis represents the cumulative distribution function, ranging from 0.0 to 1.0.
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 (missile 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 unacknowledged 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 anyway)
7. Combine Thin Streams into Thicker Stream
TCP for Games

- remove exponential backoff
- reduce RTO
- make fast retransmit faster
- retransmit aggressively
- don’t delay ACK
- combine into thick streams
With Linux kernel, 
TCP_THIN_LINEAR_TIMEOUTS
TCP_THIN_DUPACK