Lecture 11

Networked Game Traffic and Transport Protocol

Assignment 1

isolate traffic
payload size
histogram
activity pattern
periodic pattern

MOBA
RTS
FPS
RPG


DOTA 2


LEAGUE of
LEGENDS®

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


WARCRAFT III

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

SHADOWGUN DEADZONE

XONOTIC

BLACKLIGHT RETRIBUTION

UBERSTRIKE

TEAM FORTRESS 2

LEFT 4 DEAD 2

WORLD OF TANKS

BLACKSHOT

COUNTER STRIKE

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

RPG

RIFT

PATH OF EXILE

WORLD OF WARCRAFT

MapleStory

GUILD WARS 2

AGE of WUSHU

TORCHLIGHT

Metin2

RAGNARÖK
RAGNAROK ONLINE

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 $\longrightarrow$

position = 13 $\longrightarrow$ x

position = 15 $\longrightarrow$

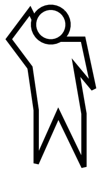
Updated position not delivered to application
until (outdated) lost packet
is received

A’s position = 10 $\longrightarrow$

B’s position = 13 $\longrightarrow$ x

C’s position = 15 $\longrightarrow$

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:// /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

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

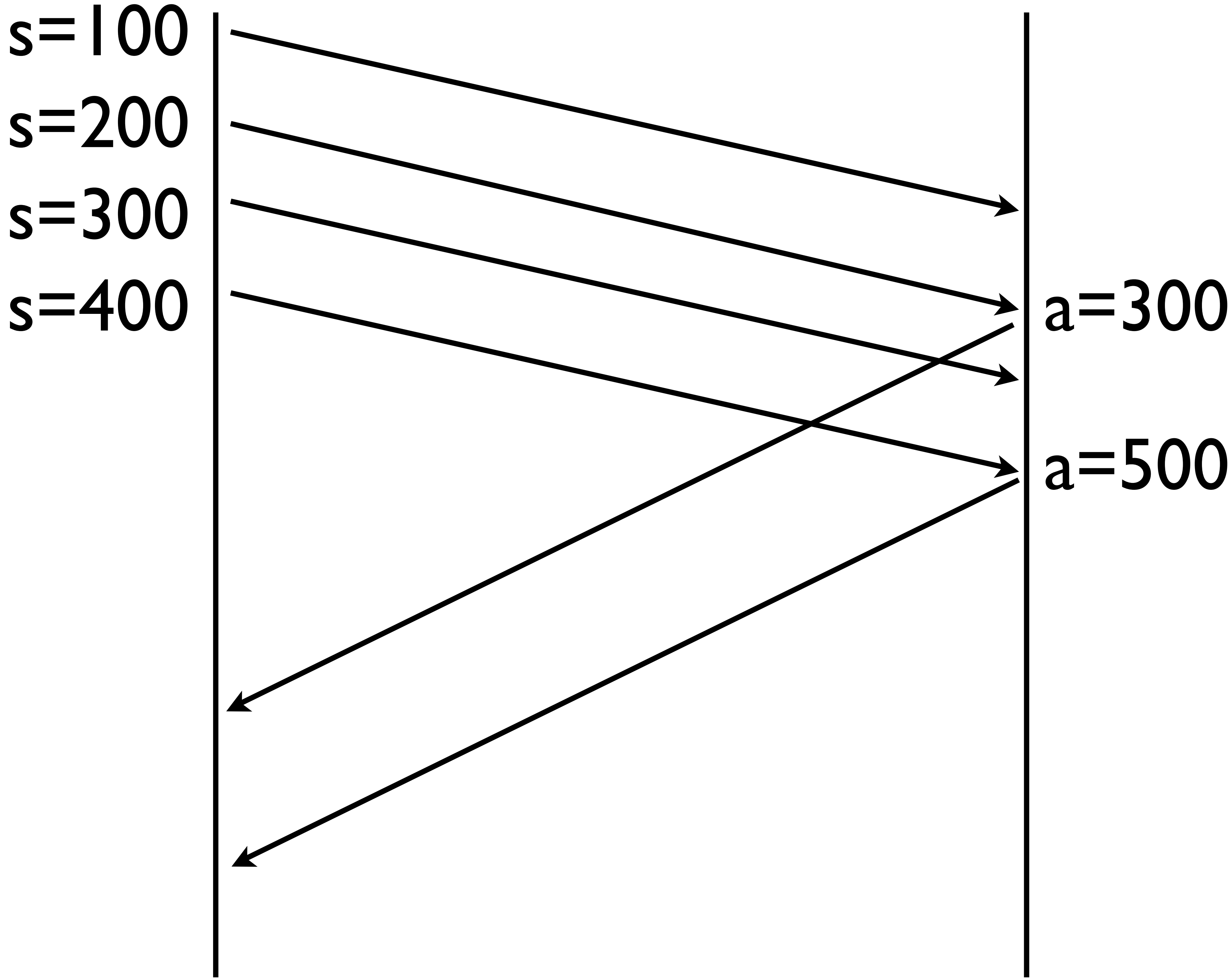
Lessons are still useful to build enet-like UDP library

How slow is TCP, really?

Which part of TCP is the root of slowness?

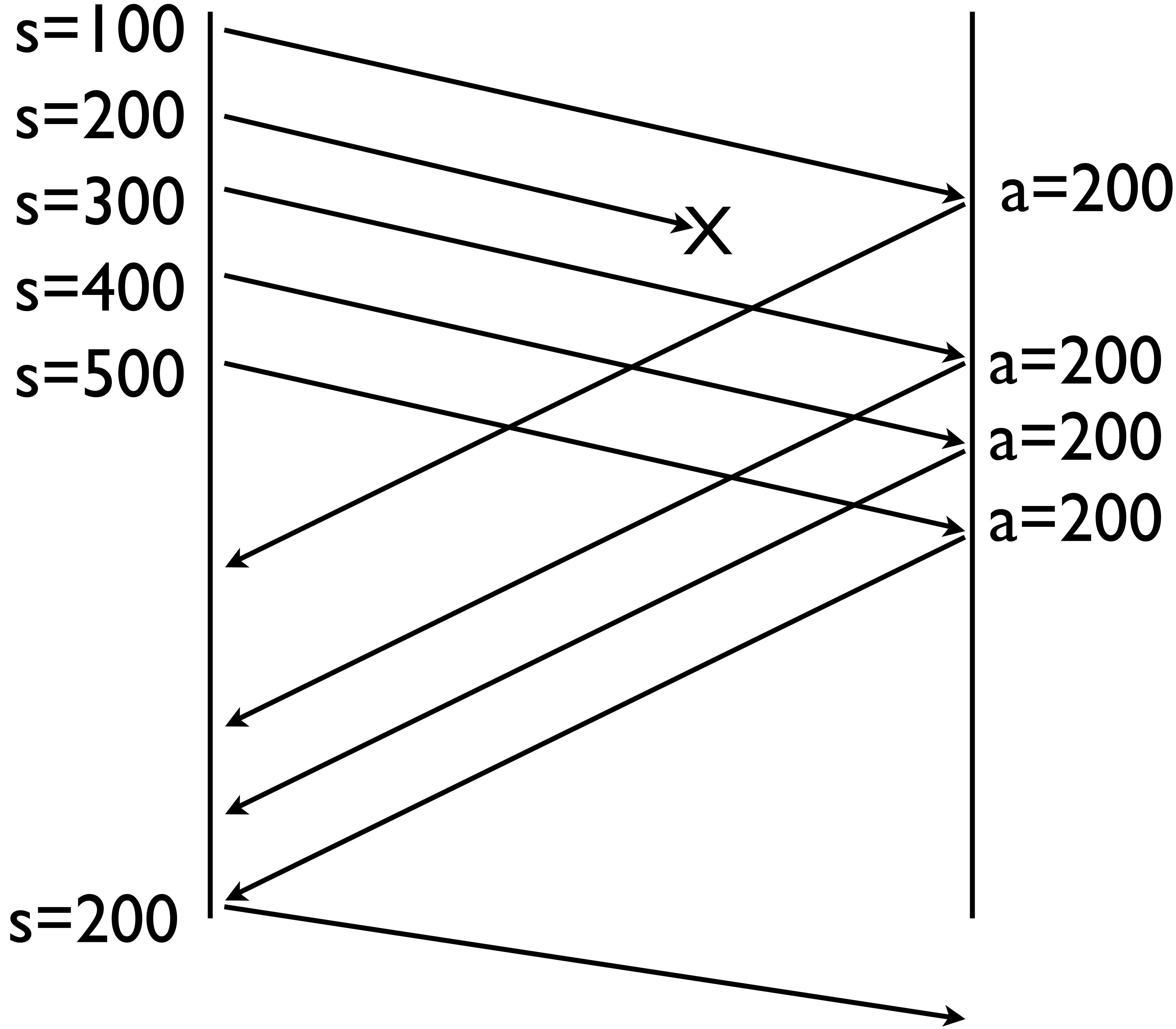
Can we fix TCP?

A Quick Review of TCP



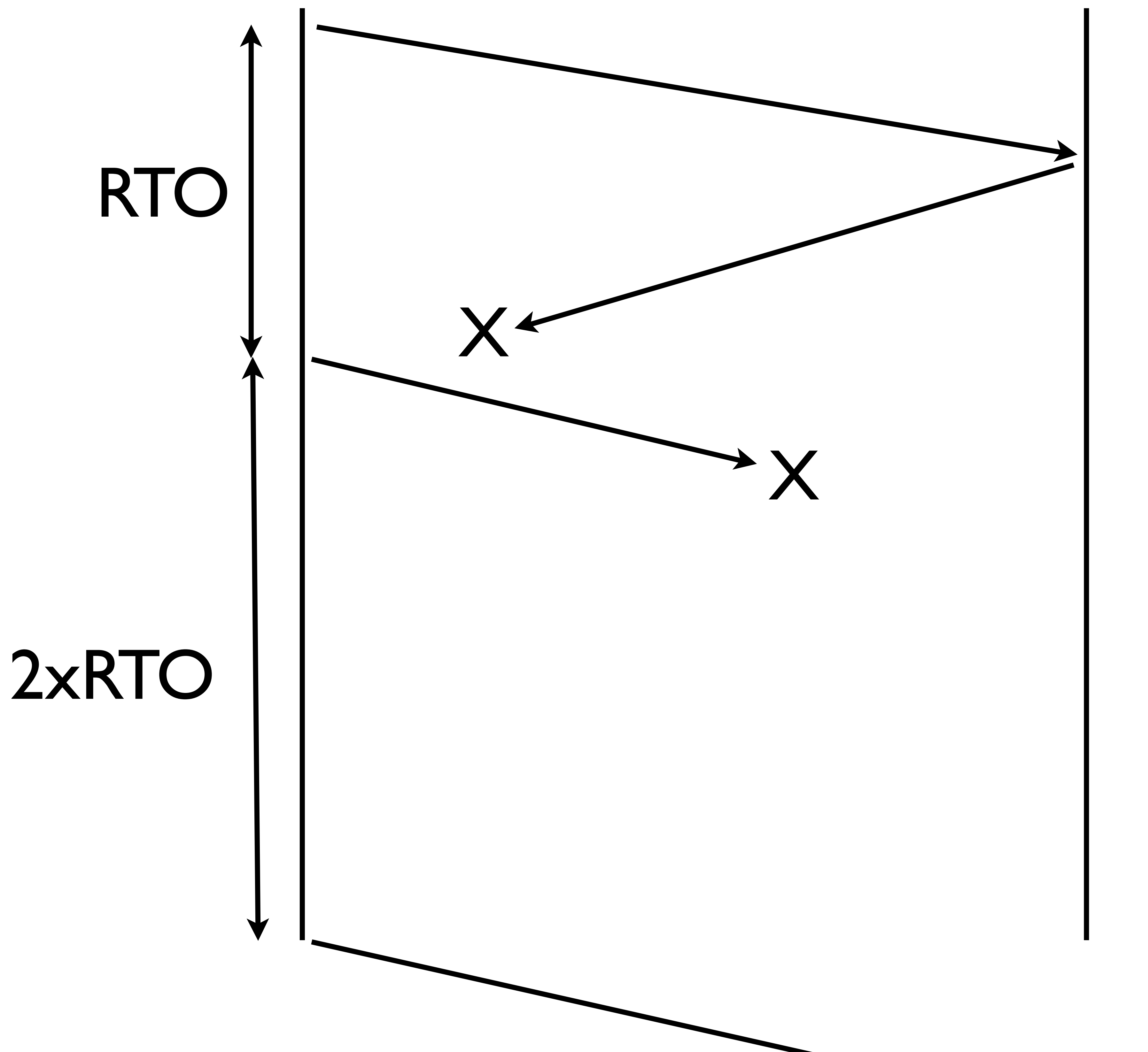
TCP Delayed ACK

TCP Spec: max **500ms** delay
Most implementation: **200ms**

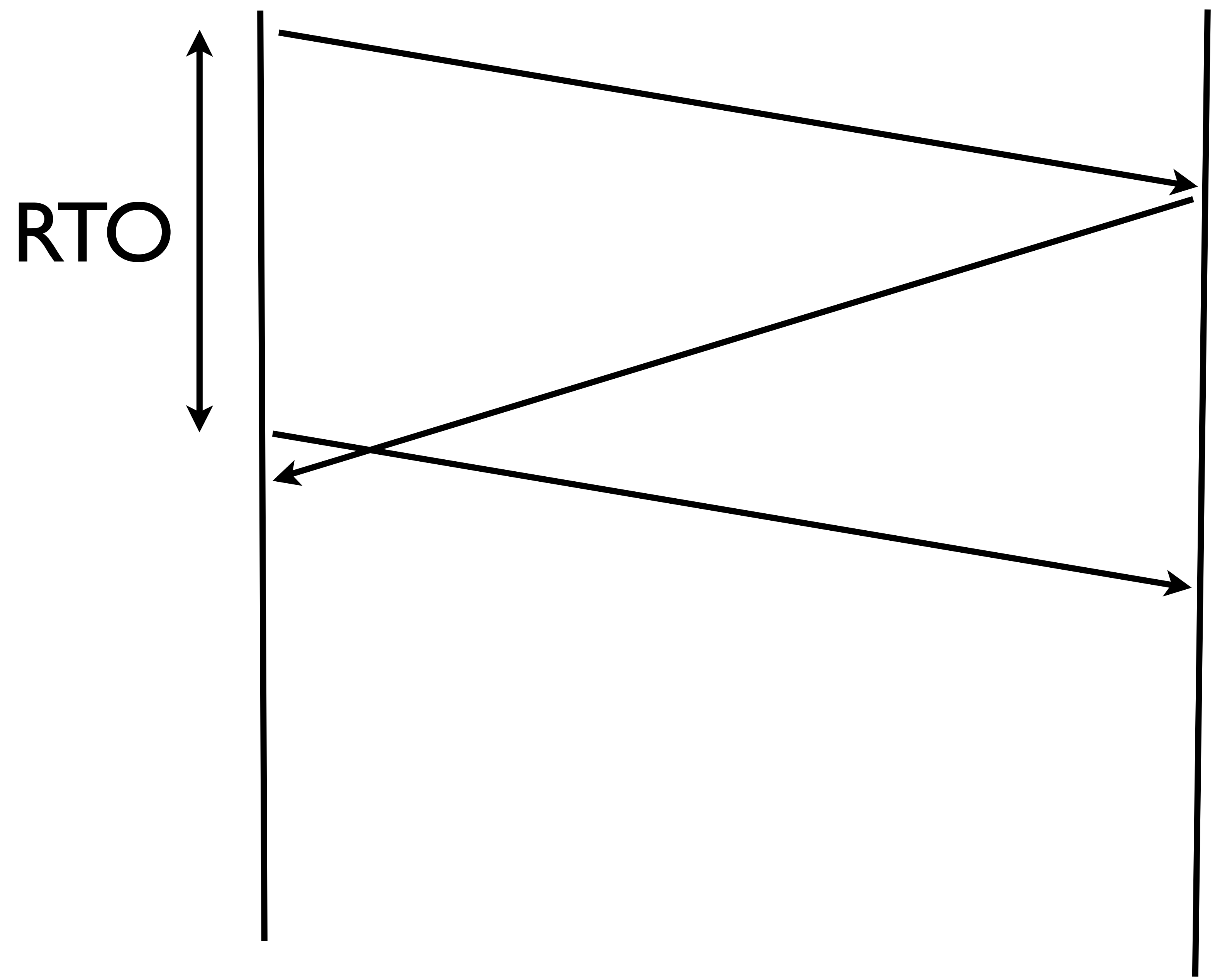


3 dup ACKs within RTO – RTT: TCP Fast Retransmission

Definition of Dup ACKs in 4.4BSD and Stevens: "pure ACK with no data"



TCP Timeout + Exponential Back-off



Spurious Retransmission

RTO estimation

$$E_i = 7E_{i-1}/8 + RTT/8$$

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

$$RTO = \max(E_i + 4V_{i,}\ 1s)$$

Linux's RTO estimation

$$E_i = 7E_{i-1}/8 + RTT/8$$

$$V_i = 3V_{i-1}/4 + |RTT - E_{i-1}|/4$$

$$W_i = \min(V_i, 50ms)$$

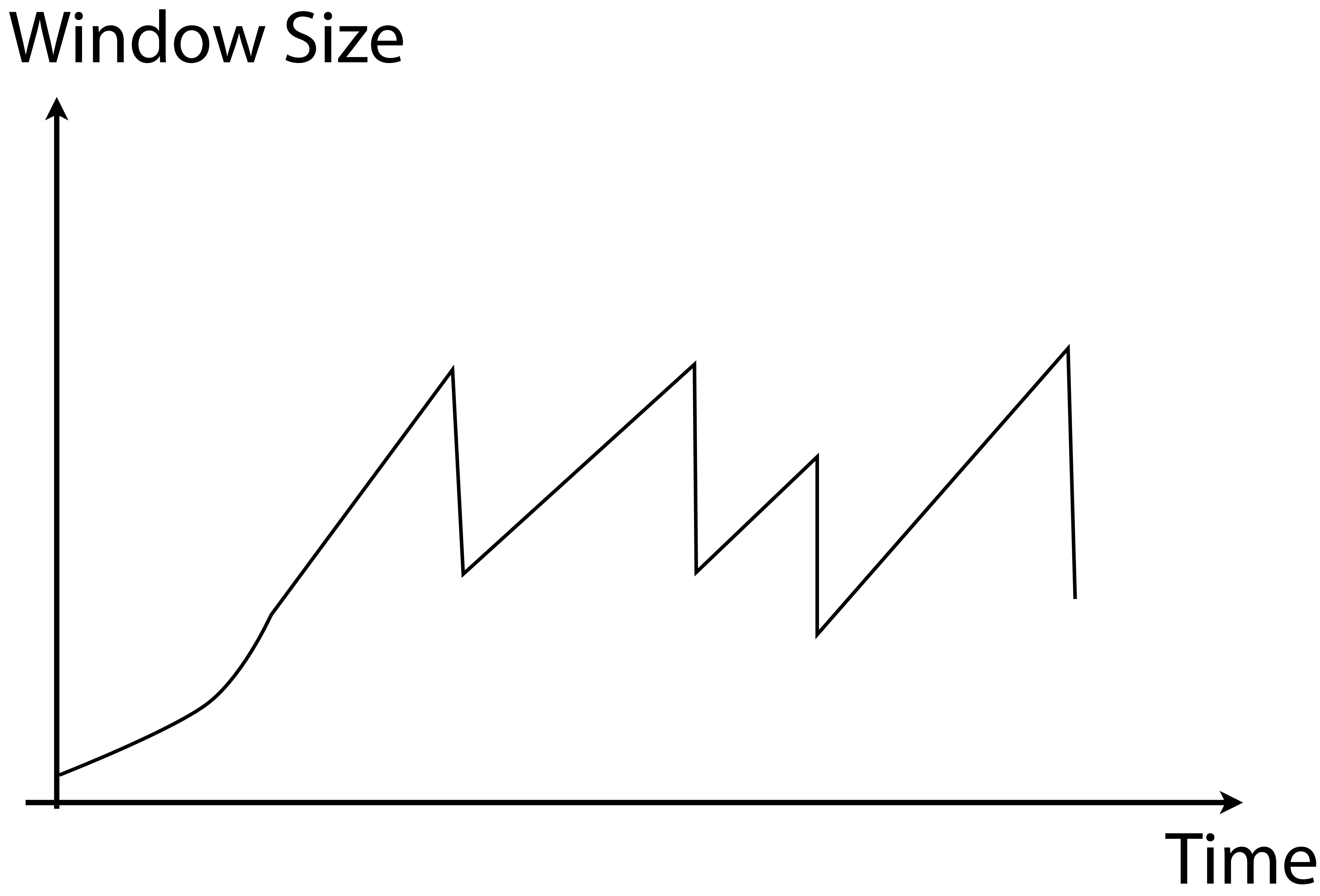
$$RTO = \max(200ms, E_i + W_i)$$

delayed ACK

increase RTT

increase RTO

Congestion Control



TCP Congestion Control

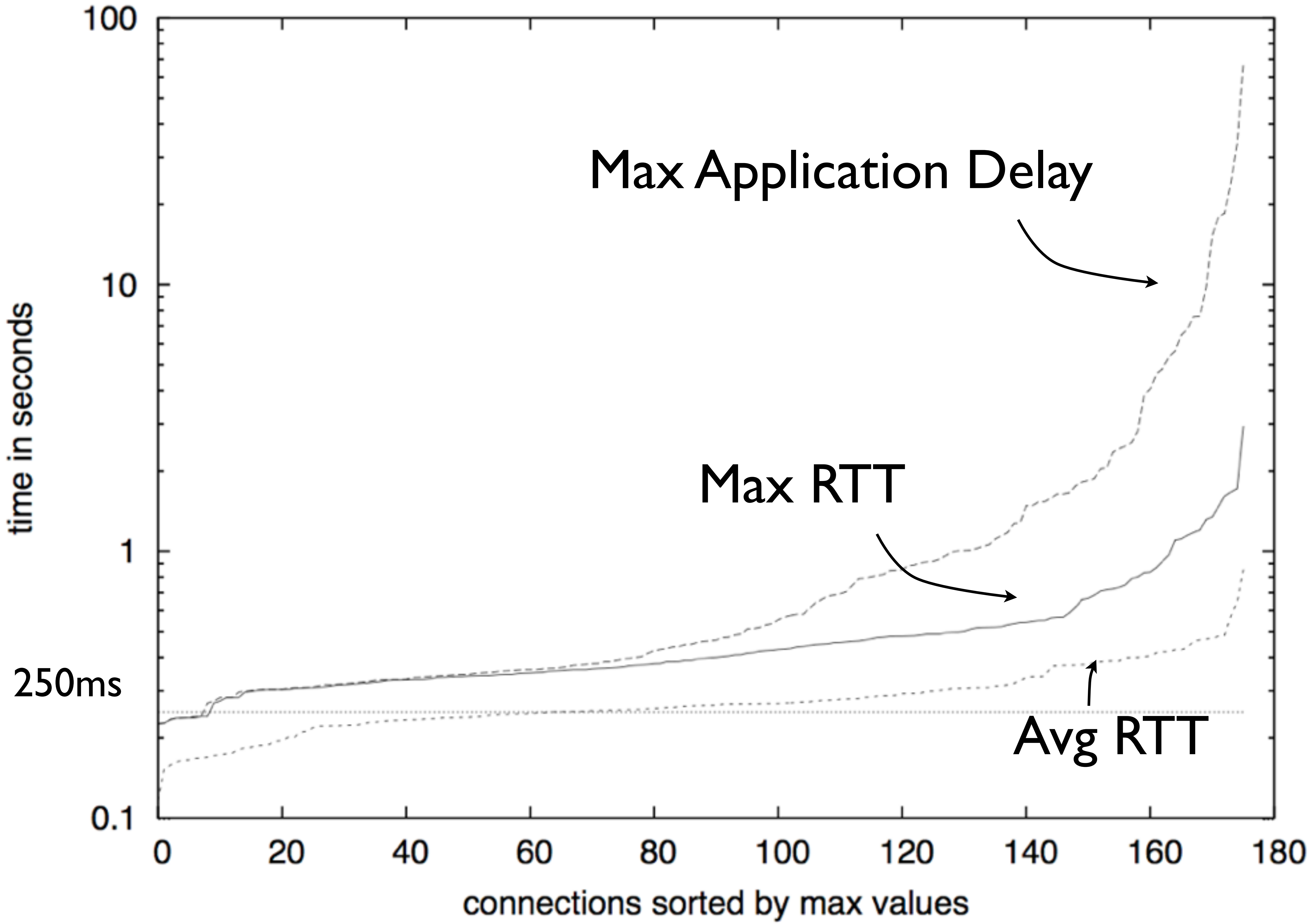
Congestion window resets to 2 after an idle period (> RTO)

What does real game traffic look like?

low packet rate
small packet size

“Thin Streams”


Anarchy online



Max Application Delay
Max RTT
Avg RTT
250ms
100
10
1
0.1
time in seconds
0
20
40
60
80
100
120
140
160
180
connections sorted by max values

About **4** packets / sec

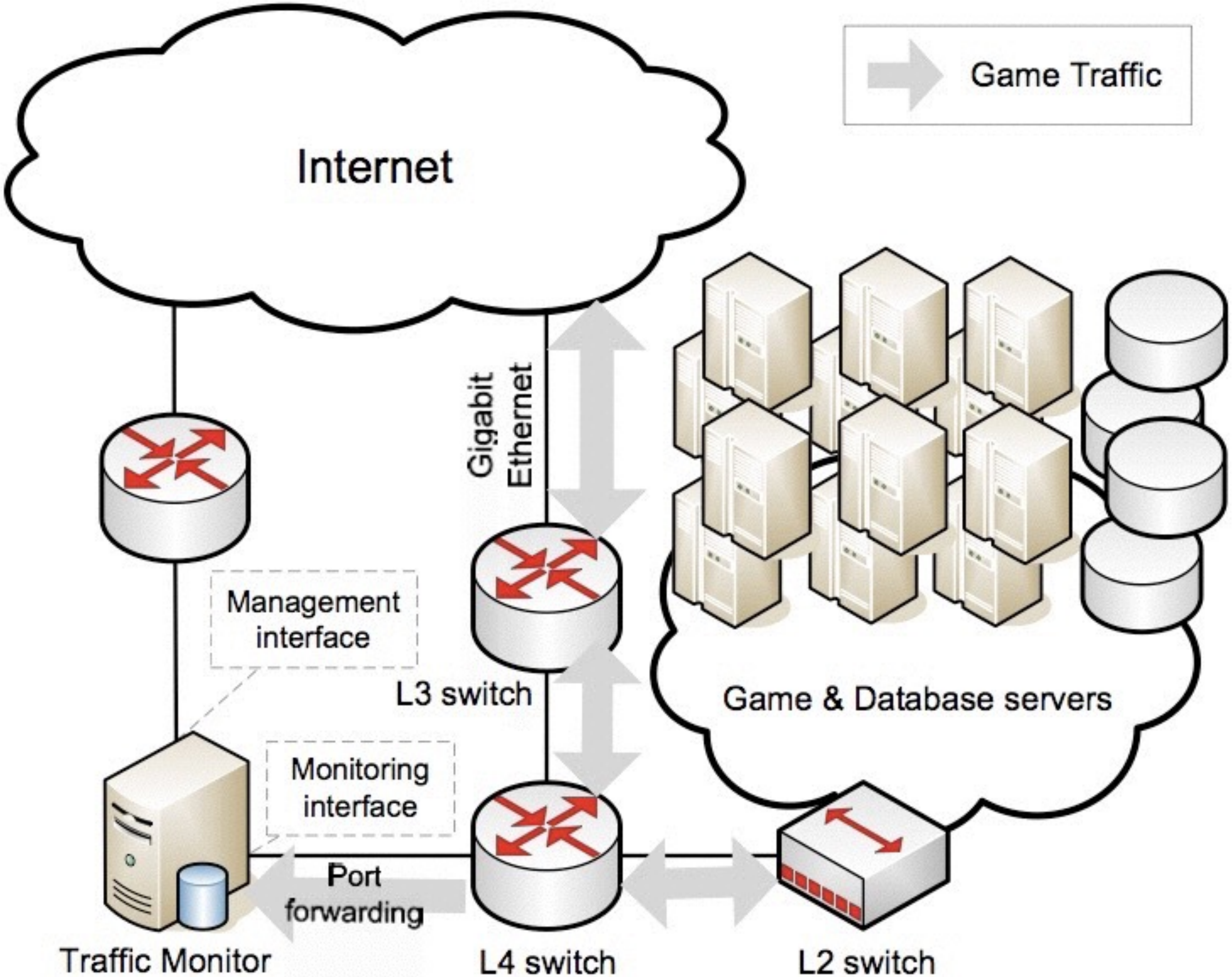
Average Payload:
100 Bytes

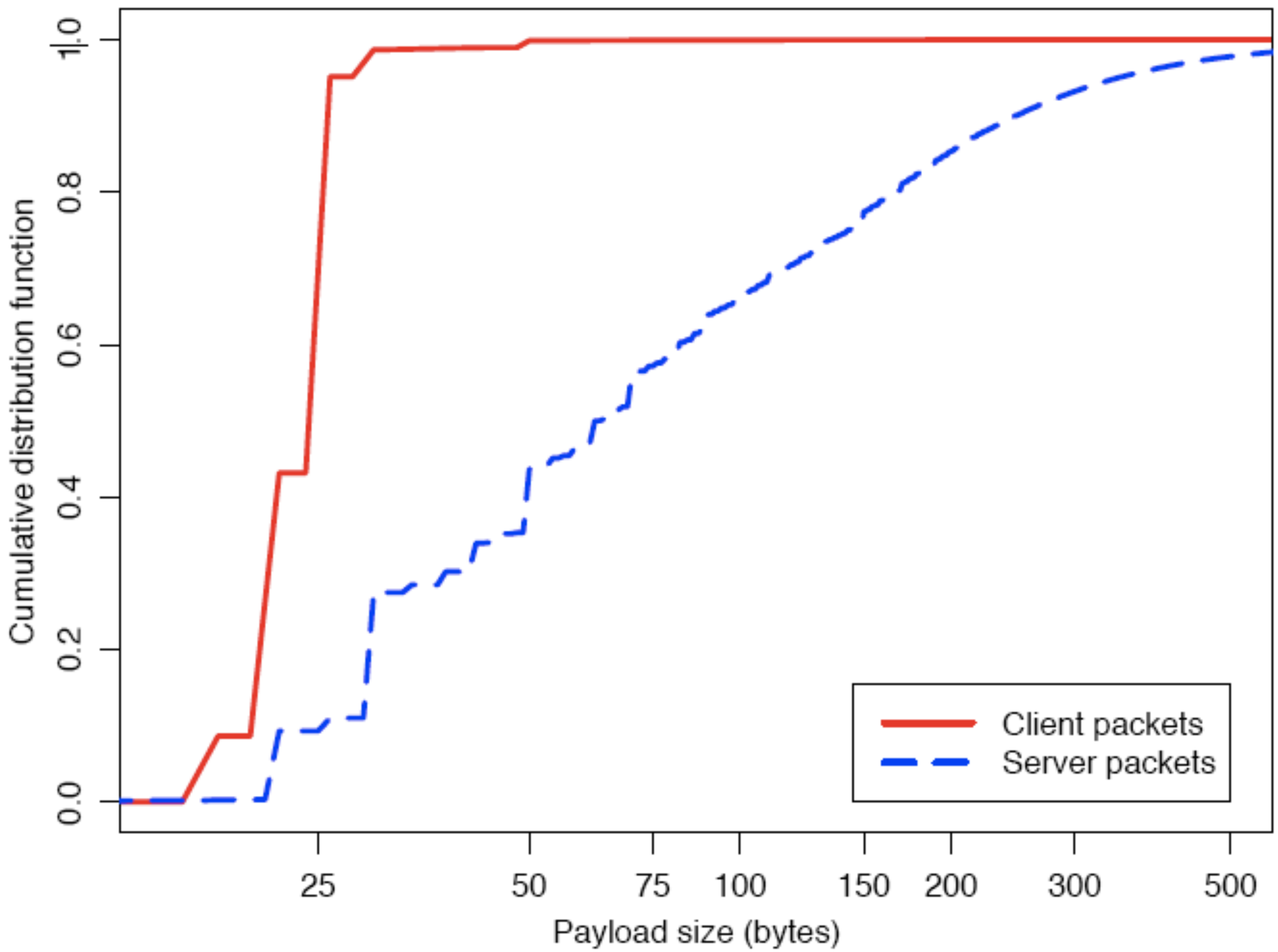
Loss Rate **1%**

But some experience **6** retransmissions

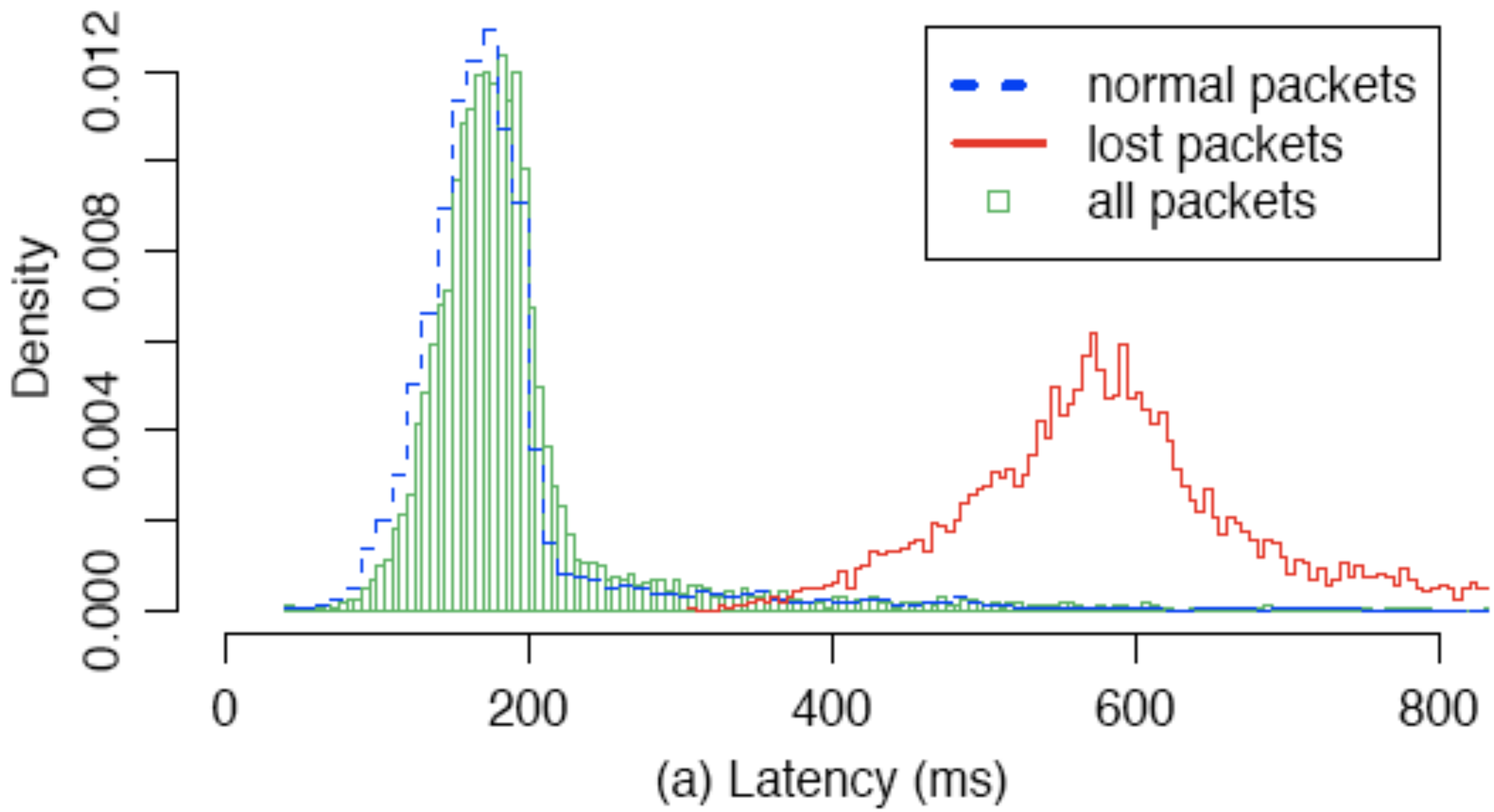
Shen Zhou Online

http://tjgame.enorth.com.cn/images/200307/0903-1.jpg


Game Traffic
Internet
Gigabit
Ethernet
Management
interface
L3 switch
Game & Database servers
Monitoring
interface
Port
forwarding
Traffic Monitor
L4 switch
L2 switch



Cumulative distribution function
0.0
0.2
0.4
0.6
0.8
1.0
25
50
75
100
150
200
300
500
Payload size (bytes)
Client packets
Server packets


normal packets
lost packets
all packets
Density
0.000
0.004
0.008
0.012
0
200
400
600
800
(a) Latency (ms)

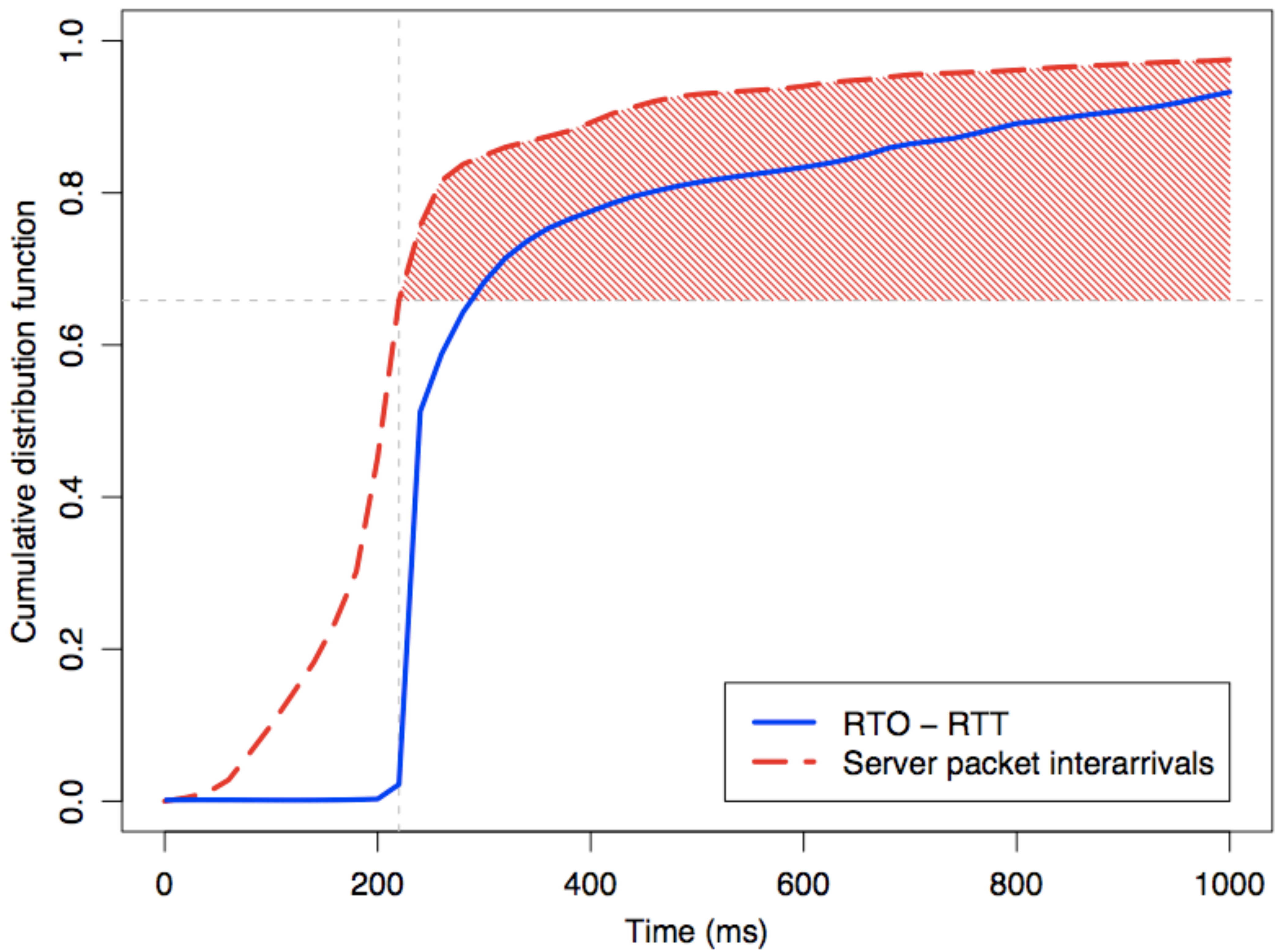
Findings 1:

Fast retransmission rarely triggered

In ShenZhou Online traces, fail to trigger fast retransmission because

insufficient dup ACK (50%)

interrupted by data (50%)



Cumulative distribution function
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000
Time (ms)
RTO – RTT
Server packet interarrivals

Findings 2:

Delay due mostly to timeout

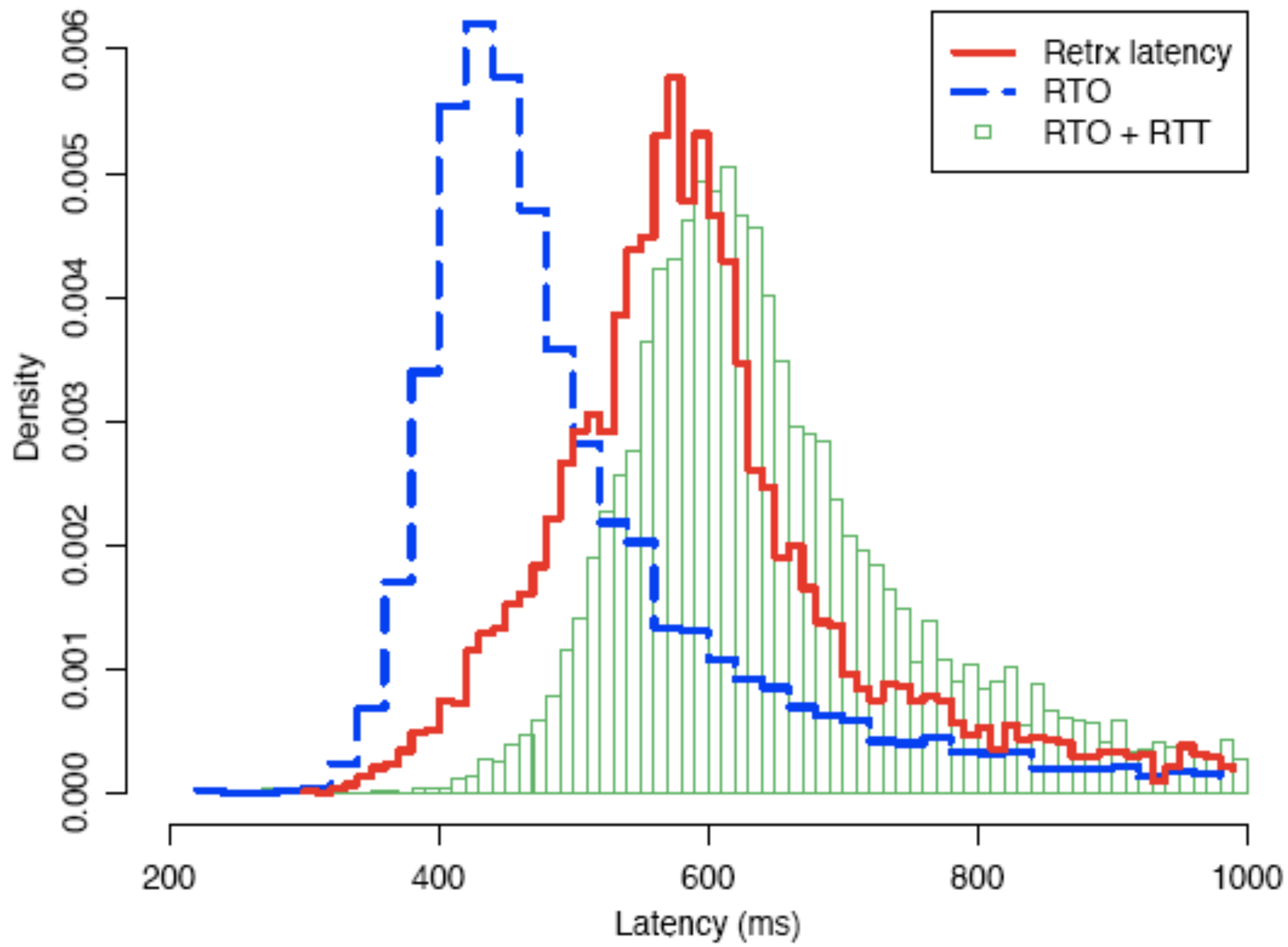
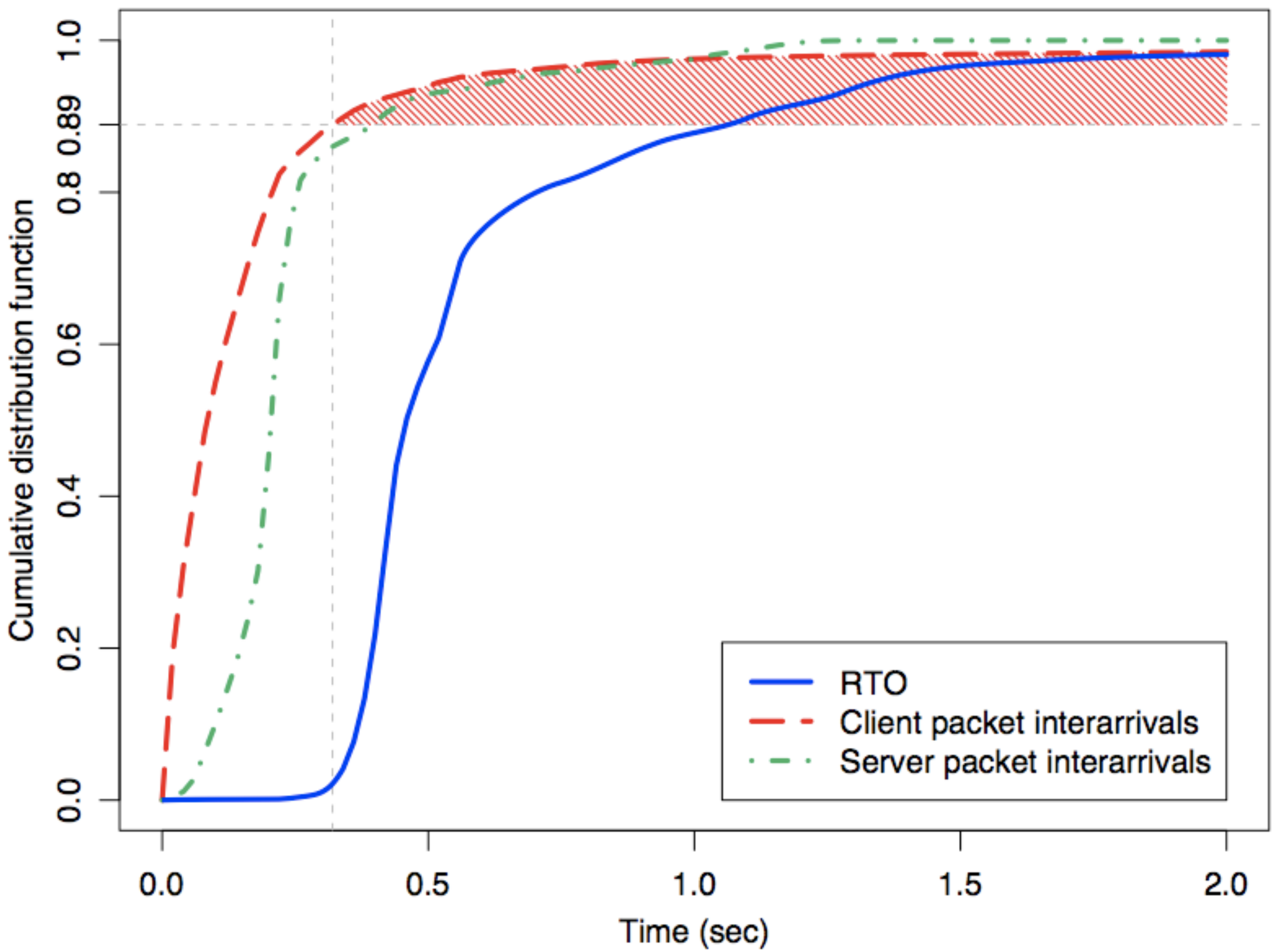


Figure 9: Average latency of dropped packets

Findings 3:
Congestion window reset is frequent



Cumulative distribution function
0.0
0.2
0.4
0.6
0.8
0.89
1.0
0.0
0.5
1.0
1.5
2.0
Time (sec)
RTO
Client packet interarrivals
Server packet interarrivals

12% - 18% of packets faces window reset

think..

think..

think..

click (tank attack here)

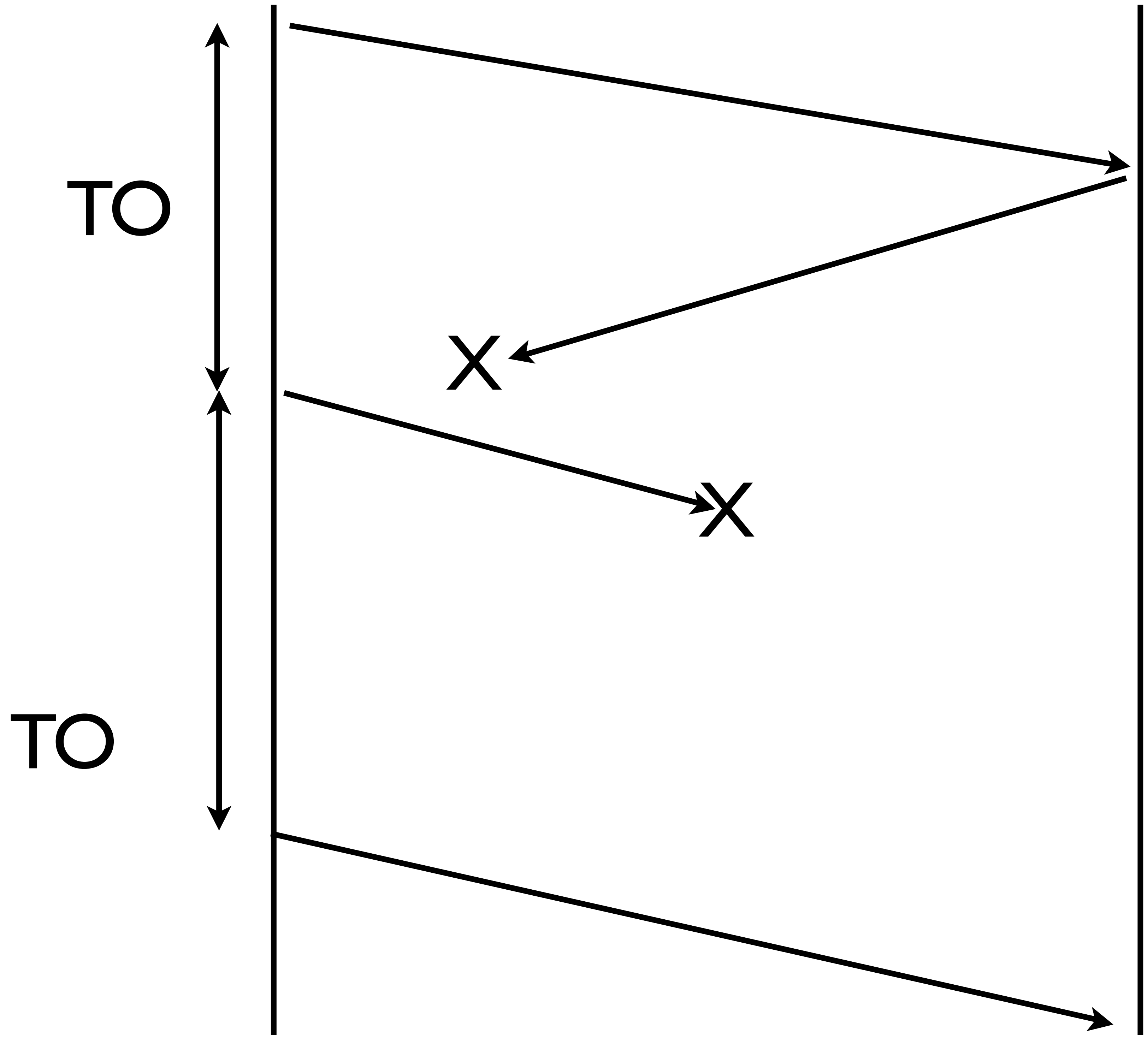
click (missile launch there)

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



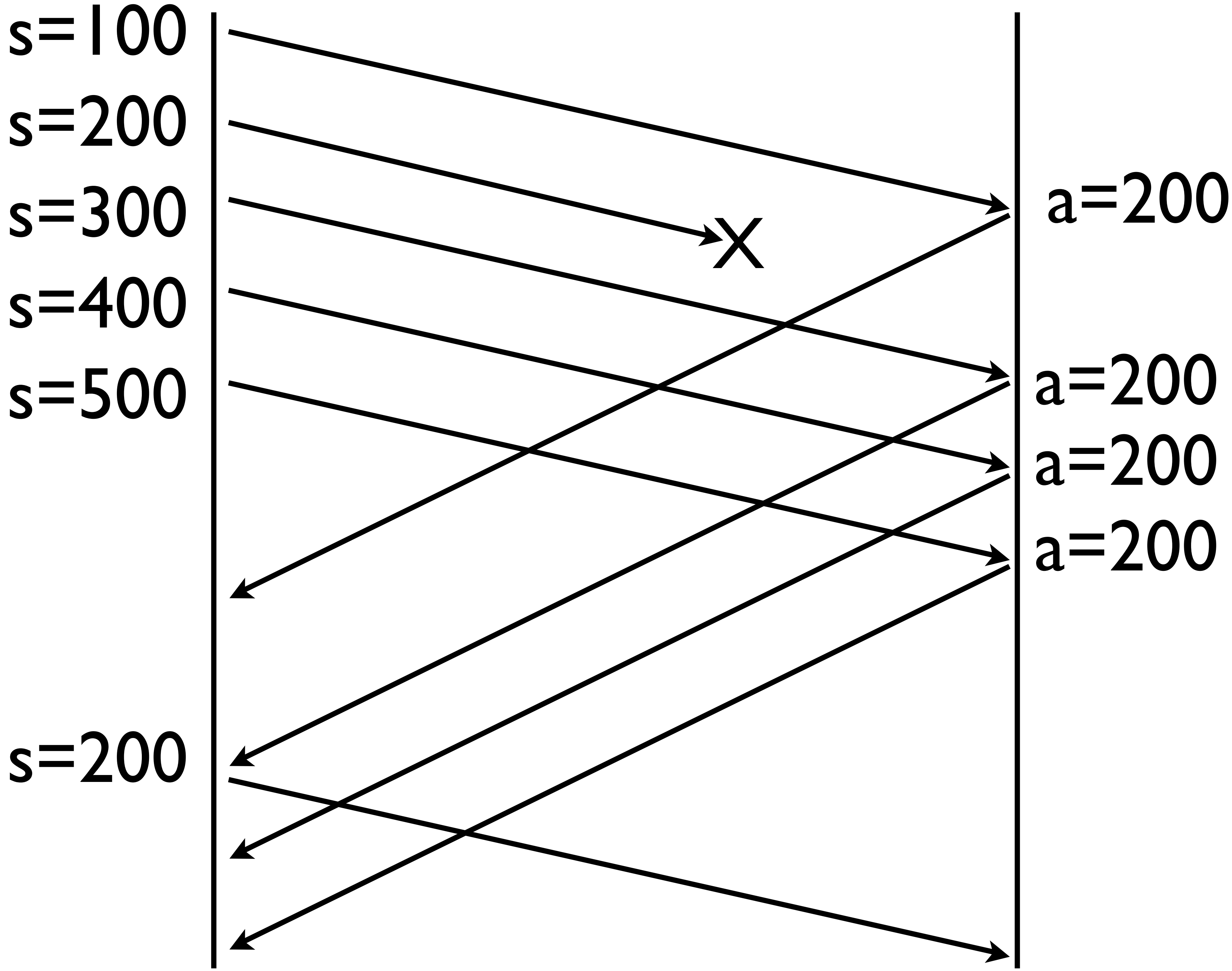
TCP Timeout

2. Make RTO Smaller

make sure minimum
RTO is not 1s

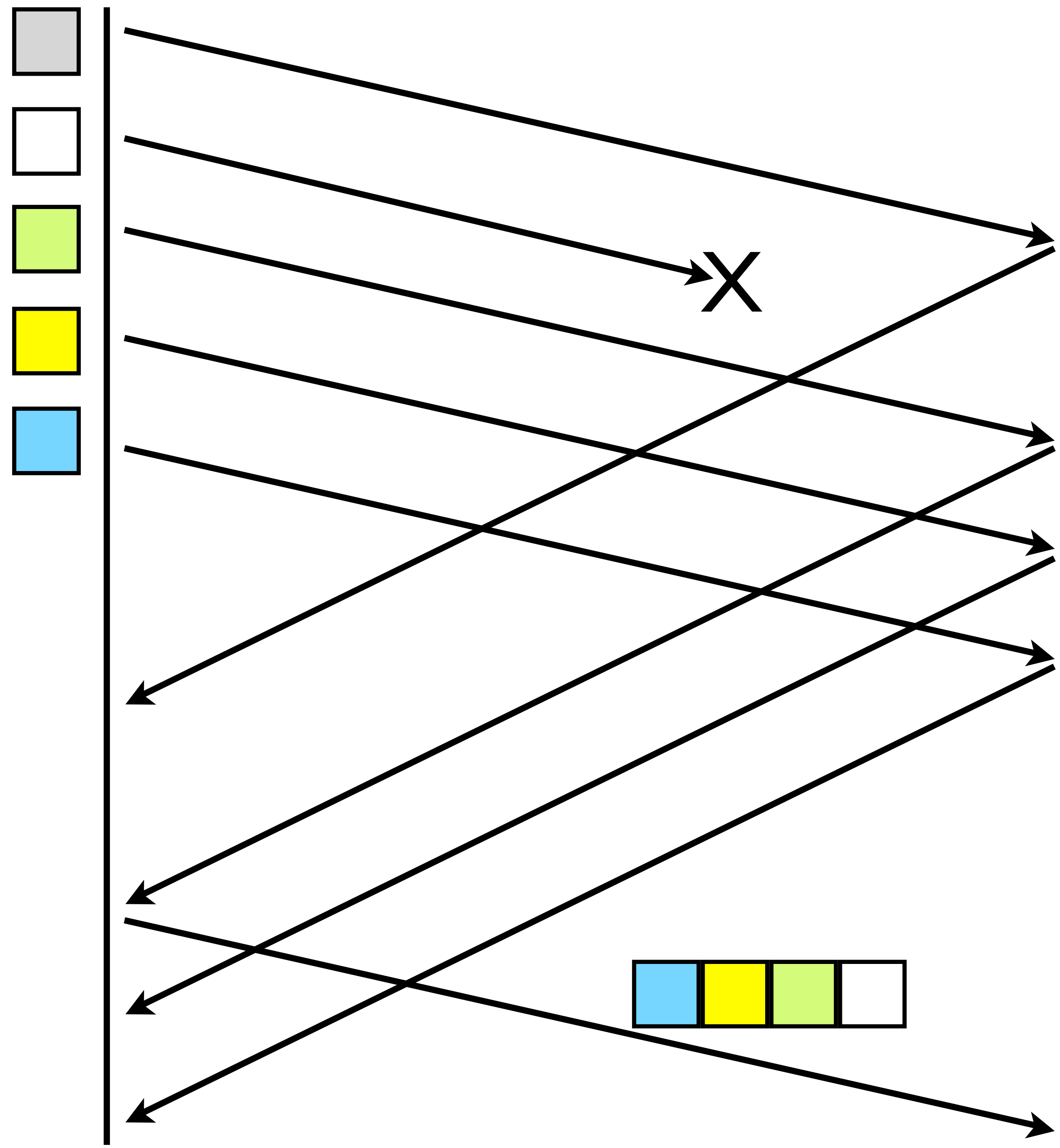
spurious retransmission
is not disastrous

3. Make Fast Retransmit Faster



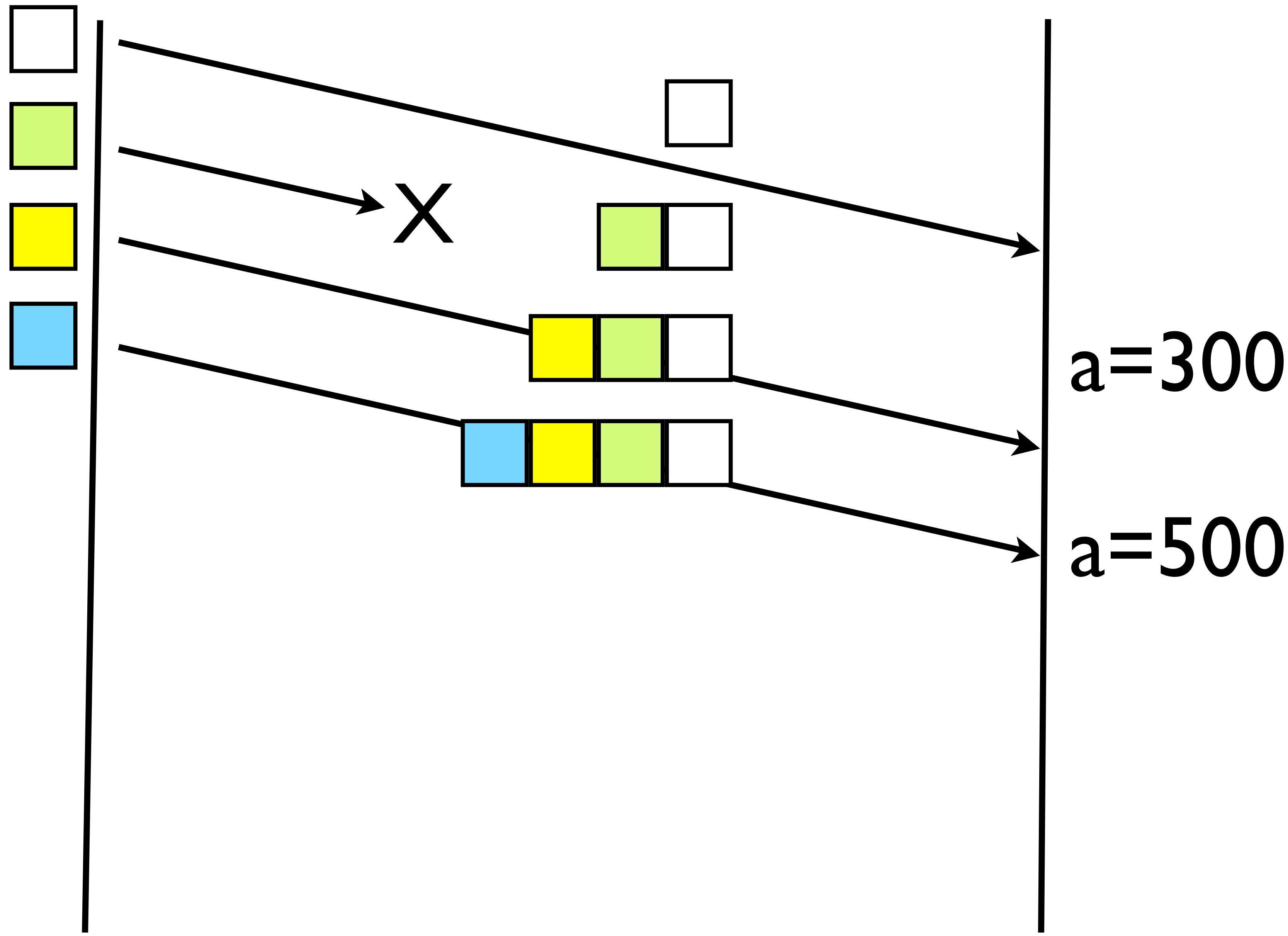
Retransmit after one duplicate ACK

4. Retransmission Bundling



Retransmit all unacknowledged data in queue

5. Redundant Data Bundling

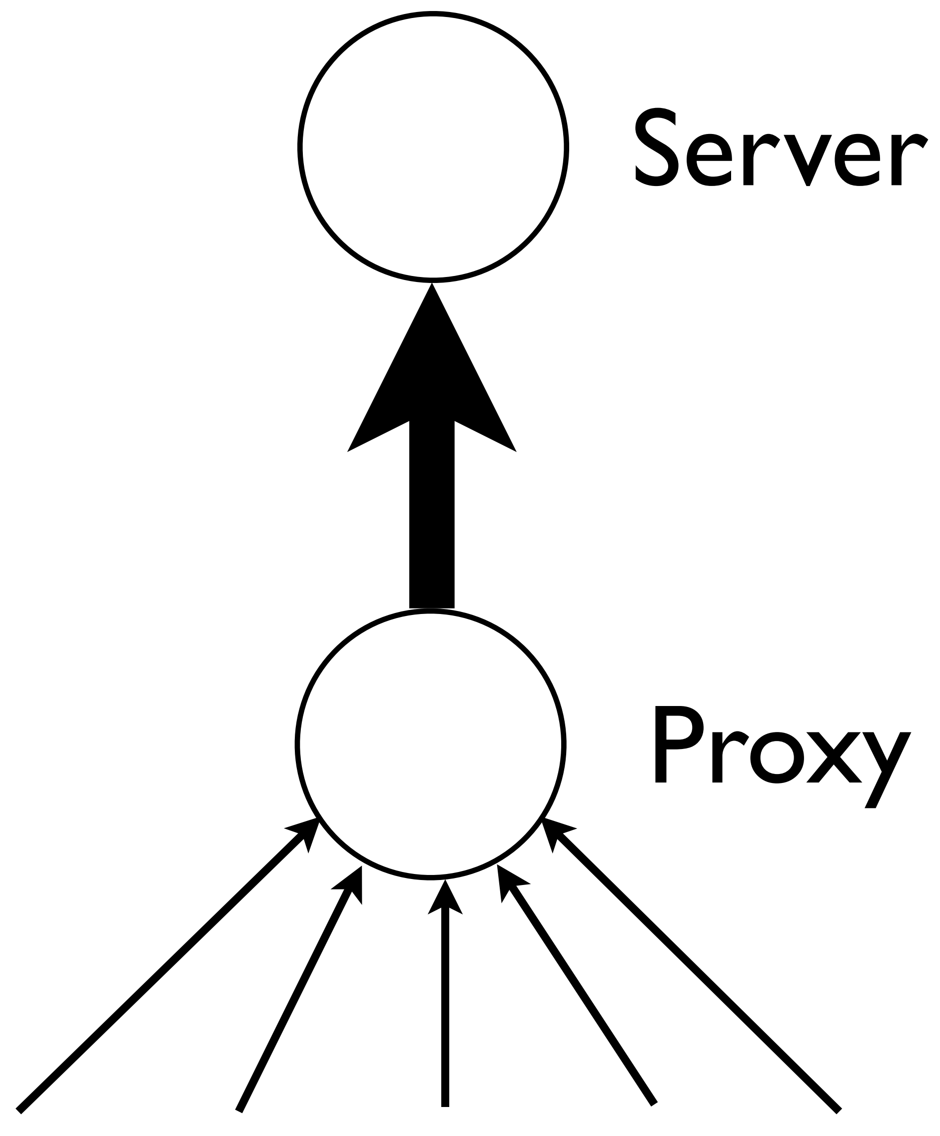


Send any unacknowledged segment in queue as long as there is space. Lost data gets recovered in the next transmission before retransmission.

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


Server
Proxy

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