Lecture 10

Transport Protocol for Networked Games

29 March 2010

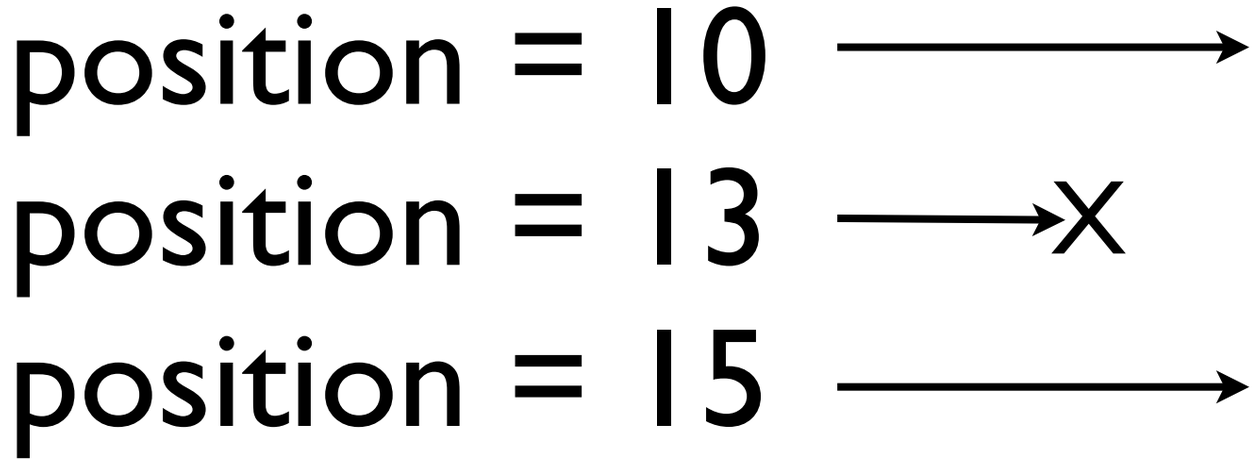
TCP or UDP ?

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



Updated position not delivered to application until (outdated) lost packet is received

A's position = 10 ⟶

B's position = 13 ⟶x

C's position = 15 ⟶

Some messages need not be delivered in sequence.

Gestures from someone far away need not be received reliably.

46%

of ShenZhou Online bandwidth is occupied by TCP header

enet.cubik.org

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.cubik.org

Portable, easy to use, but still, most firewalls block UDP traffic

- **Uses TCP**: WoW, Lineage I/II, Guild Wars, Ragnarok Online, Anarchy Online, Mabinogi

- **Uses UDP**: EverQuest, SW Galaxies, City of Heroes, Ultima Online, Asherons Call, FFXI

Need to study the use of TCP for networked games

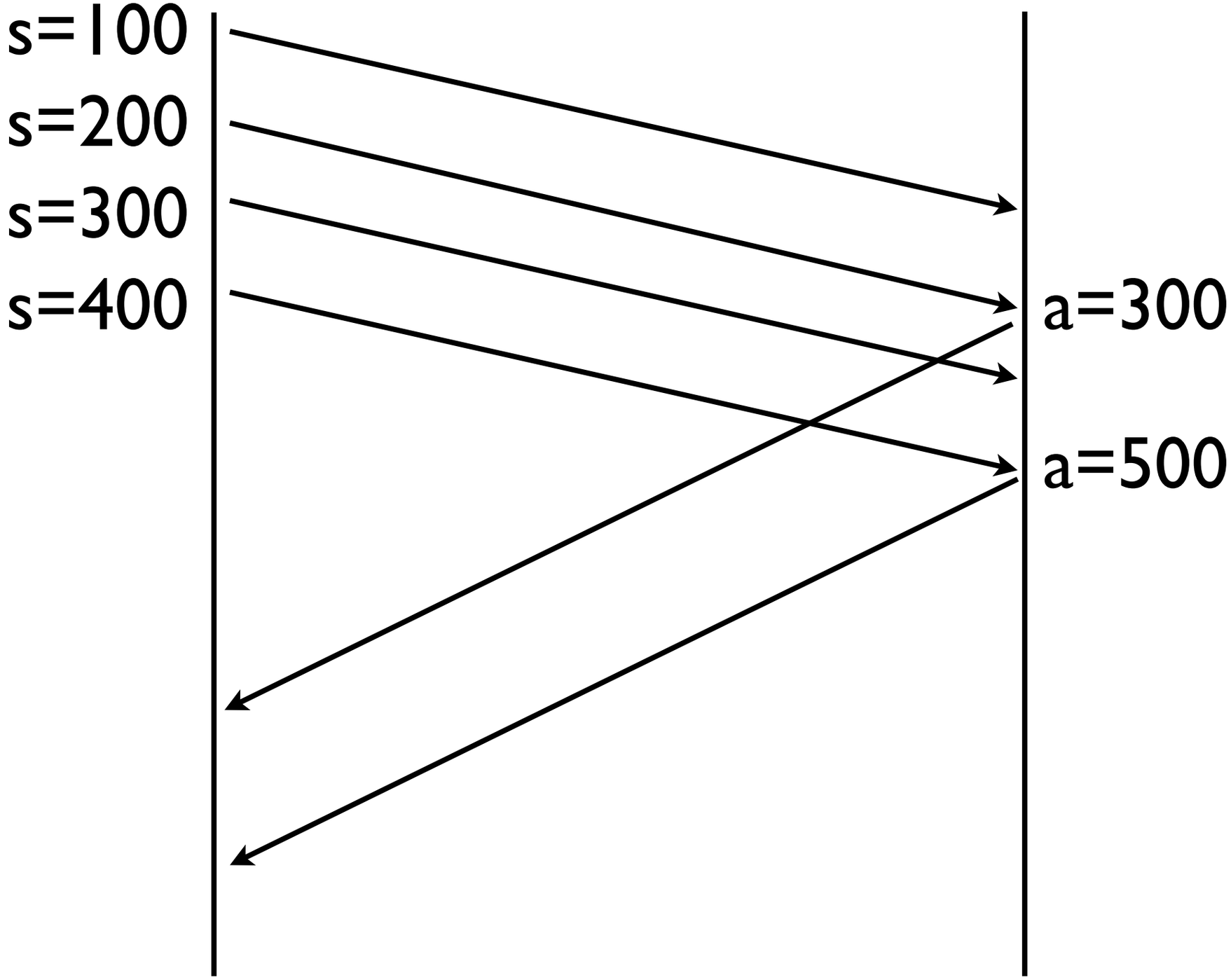
How to provide reliability over UDP?

How slow is TCP, really?

Which part of TCP is the root of slowness?

Can we fix TCP?

A Quick Review of TCP



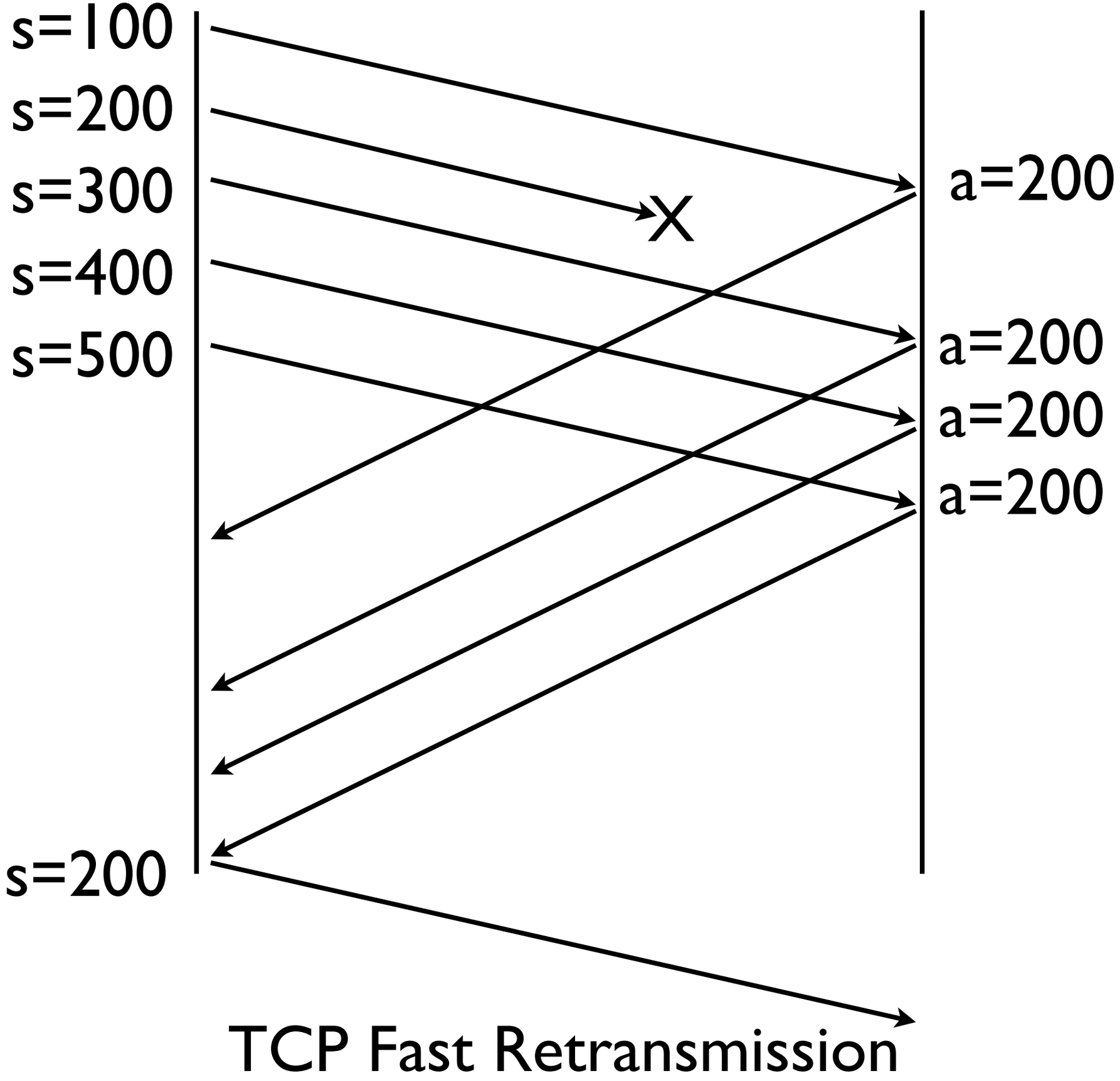
TCP Delayed ACK

TCP Spec: max **500ms** delay

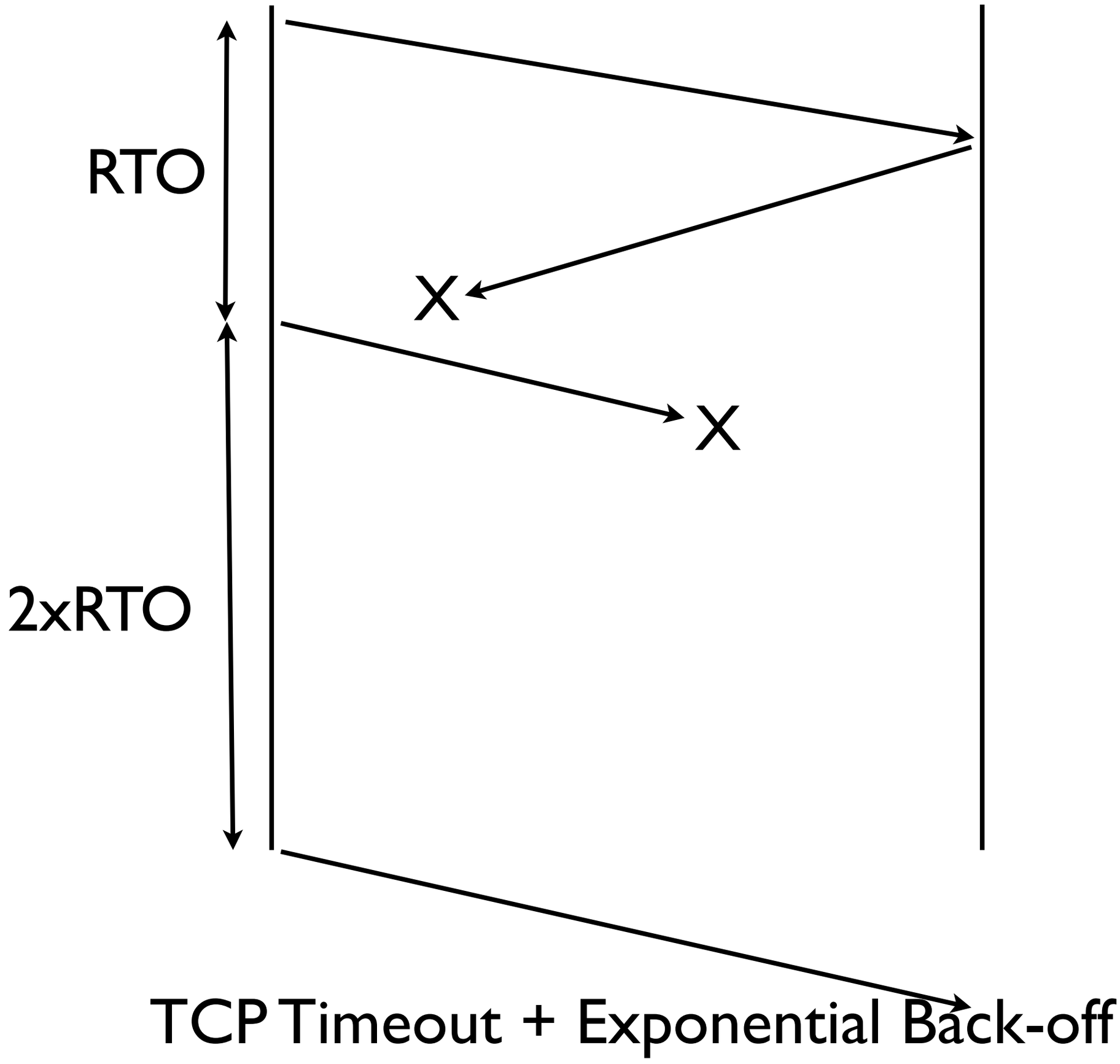
Most implementation: **200ms**

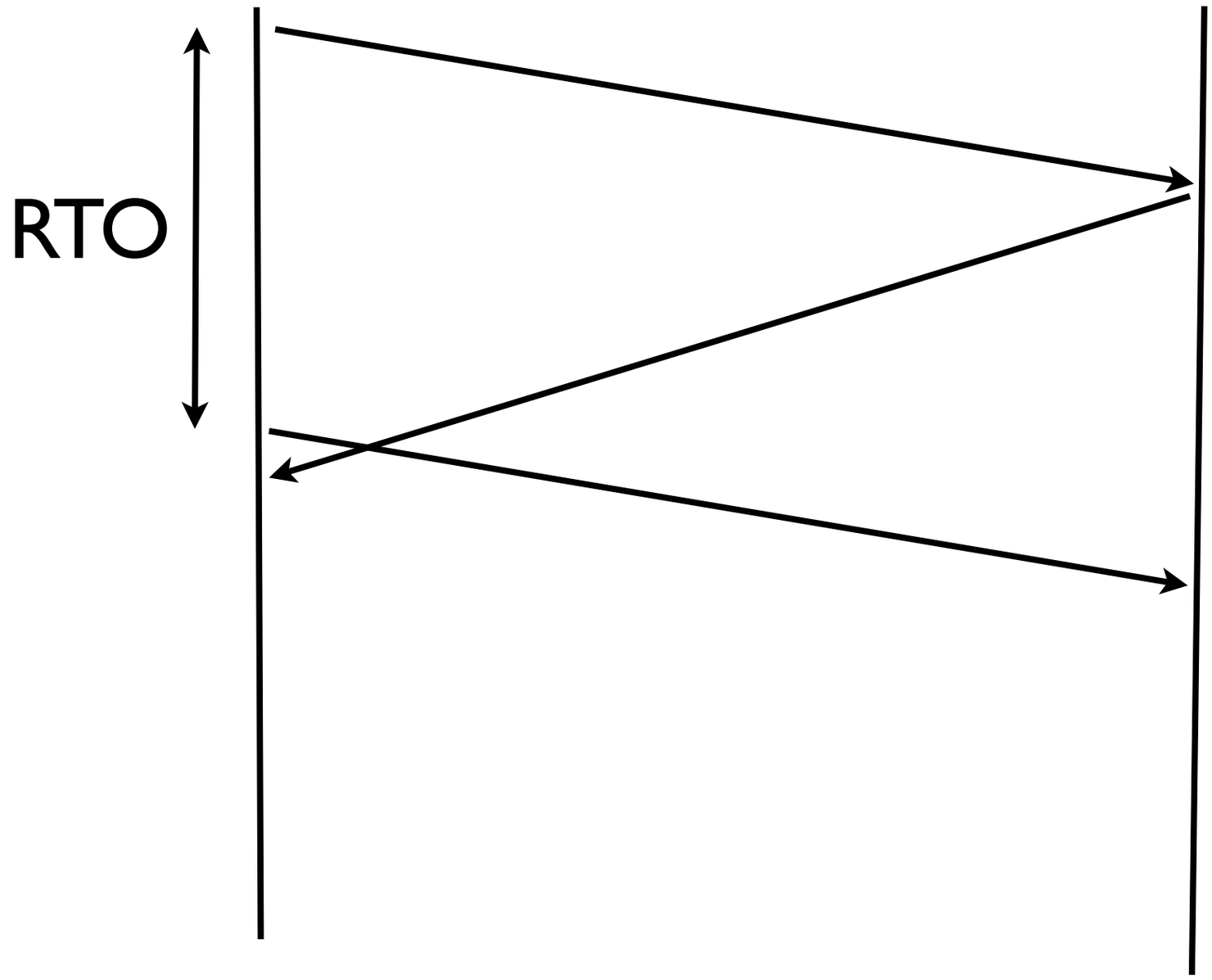
Why delay ACK?

- 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)


s=100
s=200
s=300
s=400
s=500
X
a=200
a=200
a=200
a=200
s=200
TCP Fast Retransmission

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


RTO
X
X
2xRTO
TCP Timeout + Exponential Back-off



Spurious Retransmission

RTO estimation

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

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

$$RTO = \max(E_i + 4V_i, 1s)$$

Linux's RTO estimation

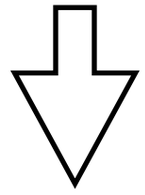
$E_i = 7E_{i-1}/8 + RTT/8$

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

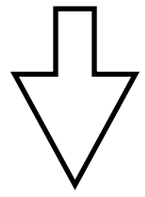
$W_i = \min(V_i, 50ms)$

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

delayed ACK

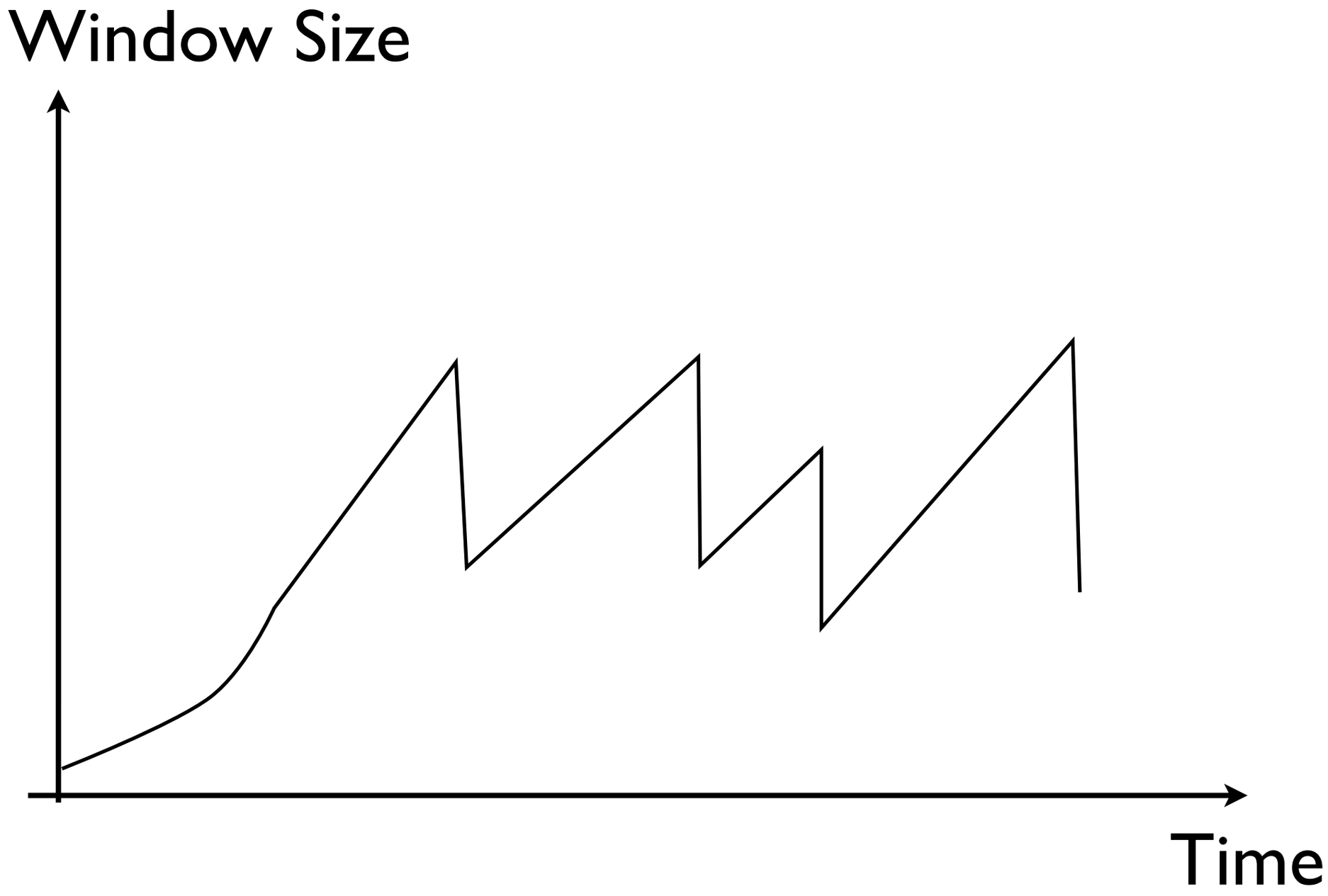


increase RTT



increase RTO

Congestion Control



TCP Congestion Control

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

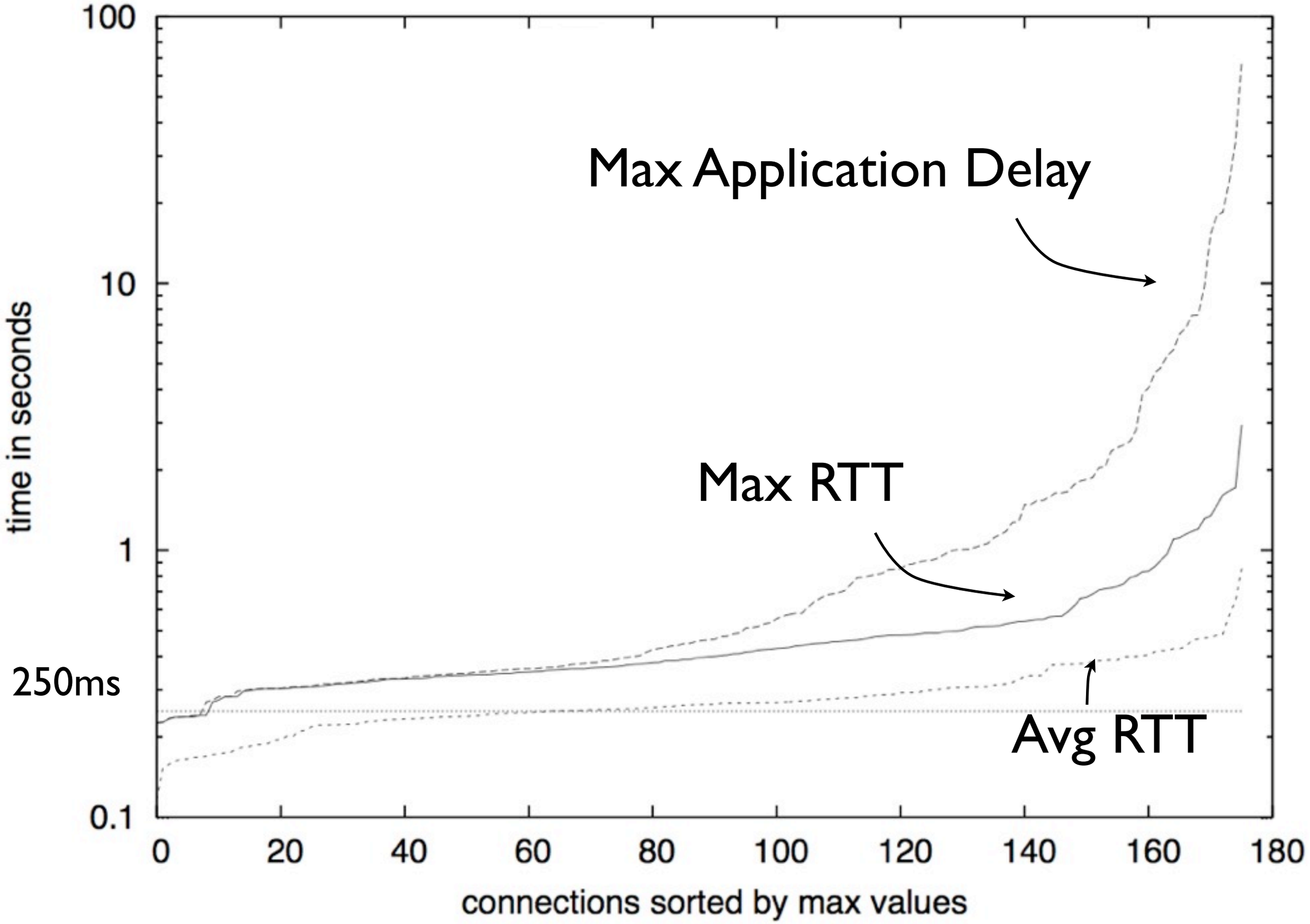
What does real game traffic look like?

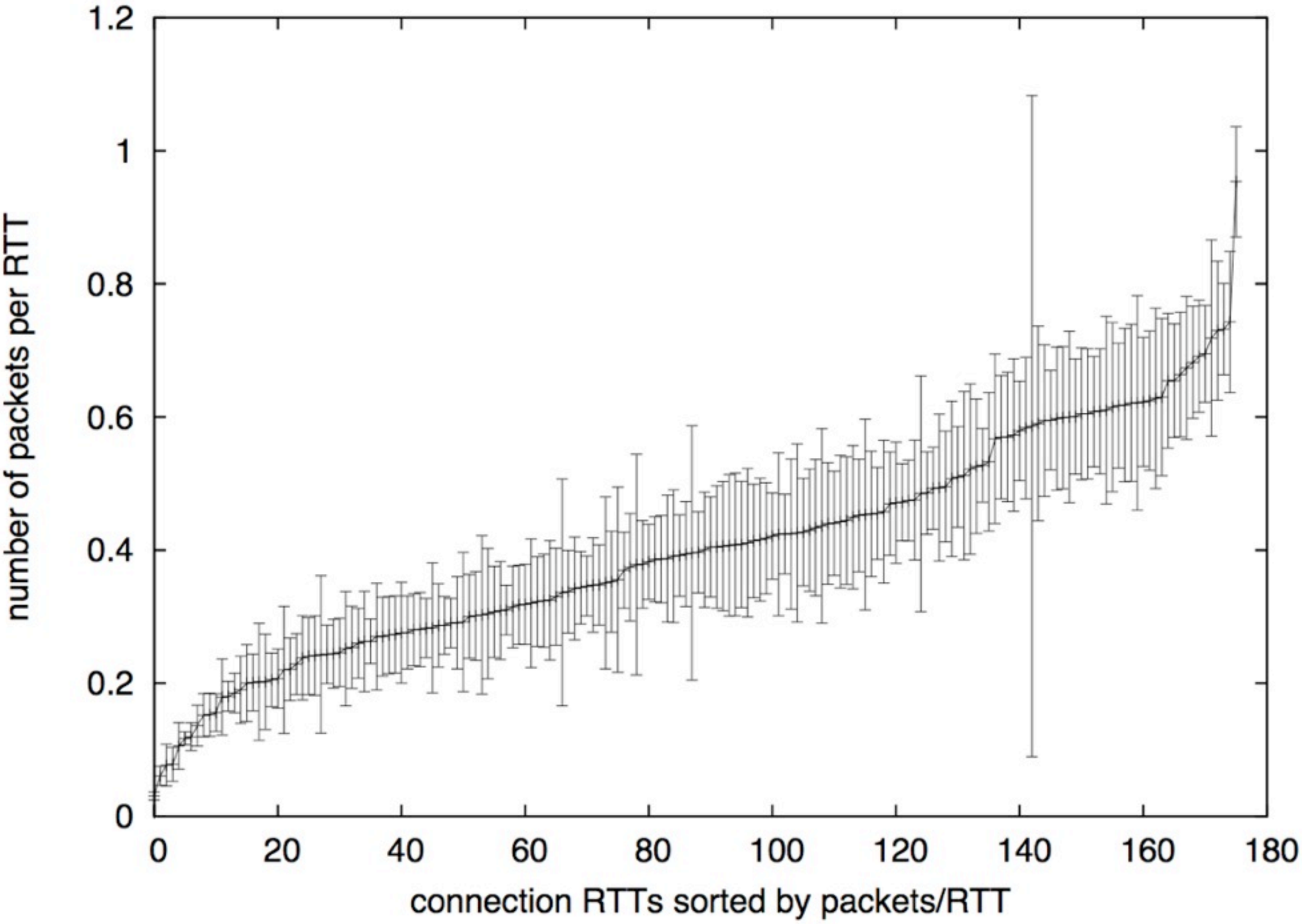
low packet rate

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


1.2
1
0.8
0.6
0.4
0.2
0
number of packets per RTT
0
20
40
60
80
100
120
140
160
180
connection RTTs sorted by packets/RTT

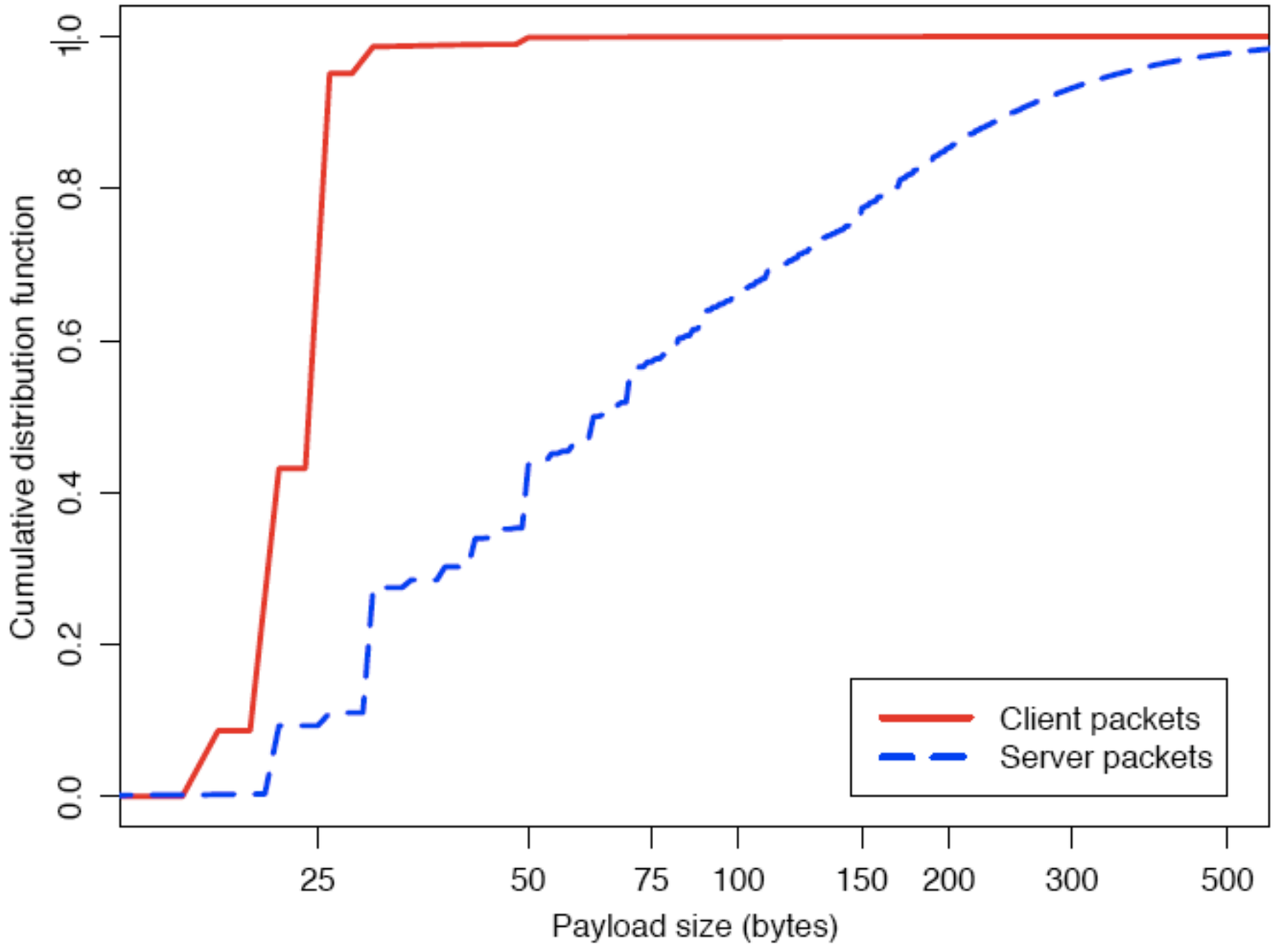
About **4** packets / sec

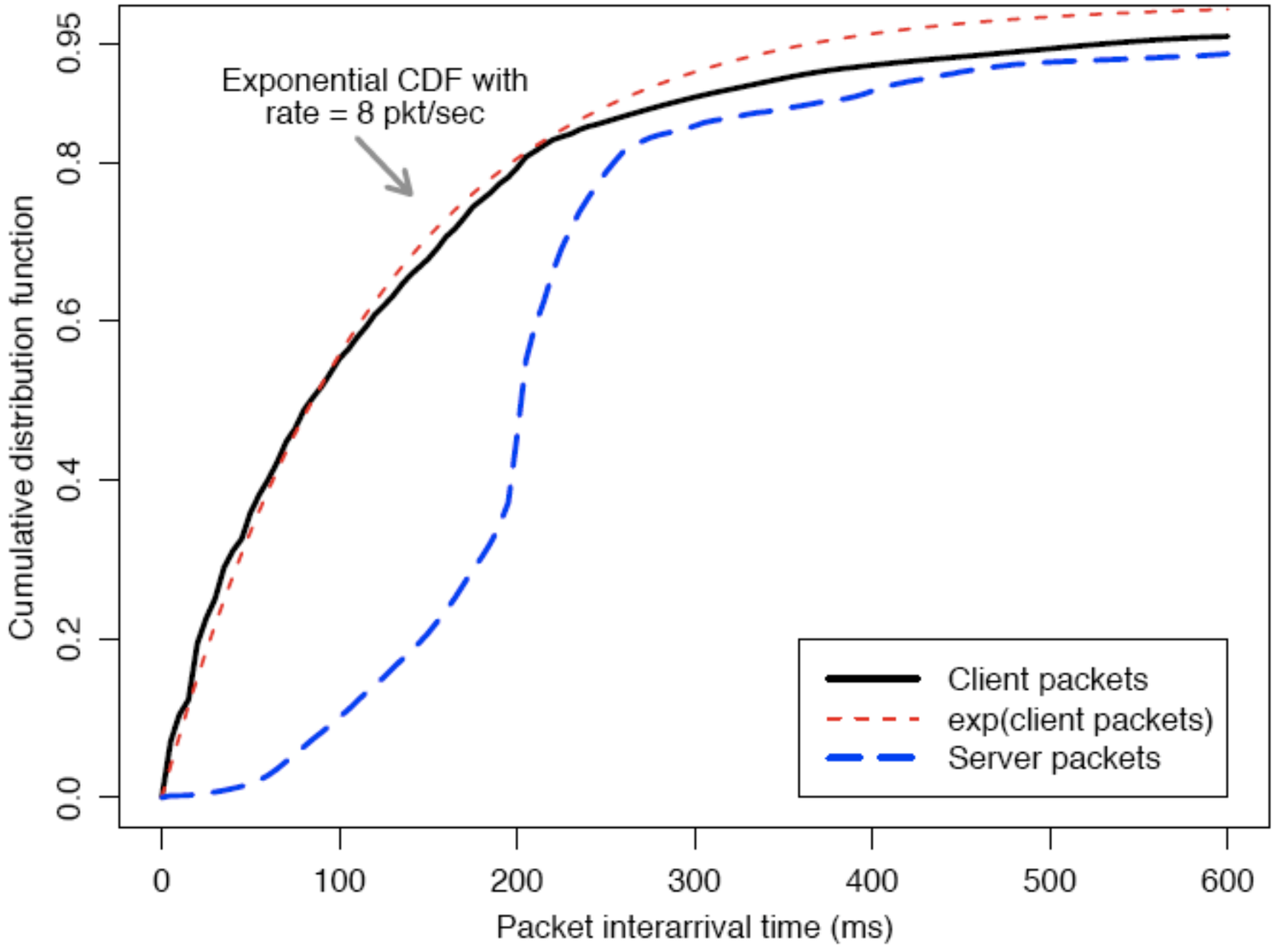
Average Payload:
100 Bytes

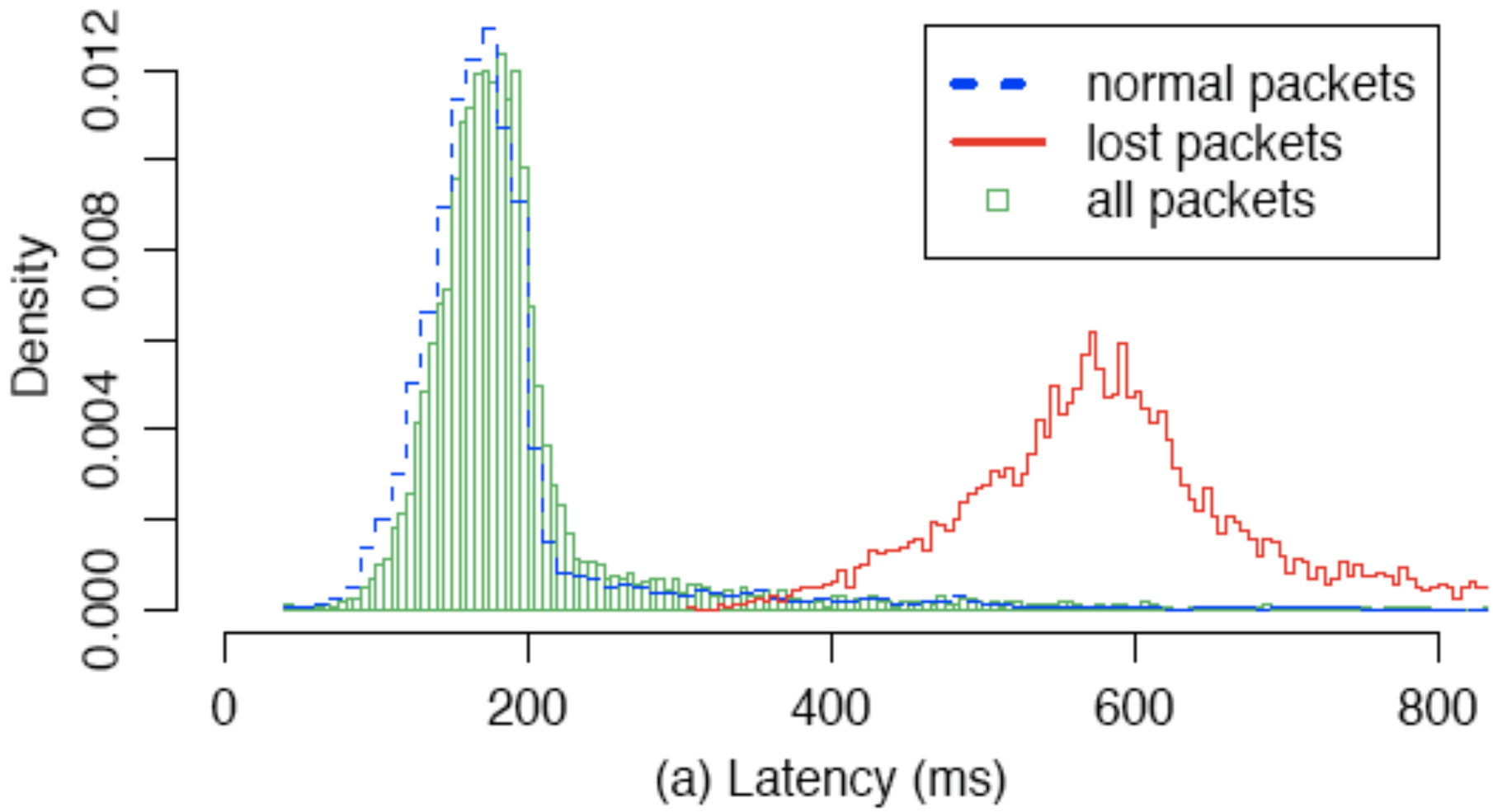
Loss Rate 1%

But some experience **6** retransmissions

ShenZhou Online


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


Exponential CDF with
rate = 8 pkt/sec
Client packets
exp(client packets)
Server packets
Cumulative distribution function
0.0
0.2
0.4
0.6
0.8
0.95
0
100
200
300
400
500
600
Packet interarrival time (ms)


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

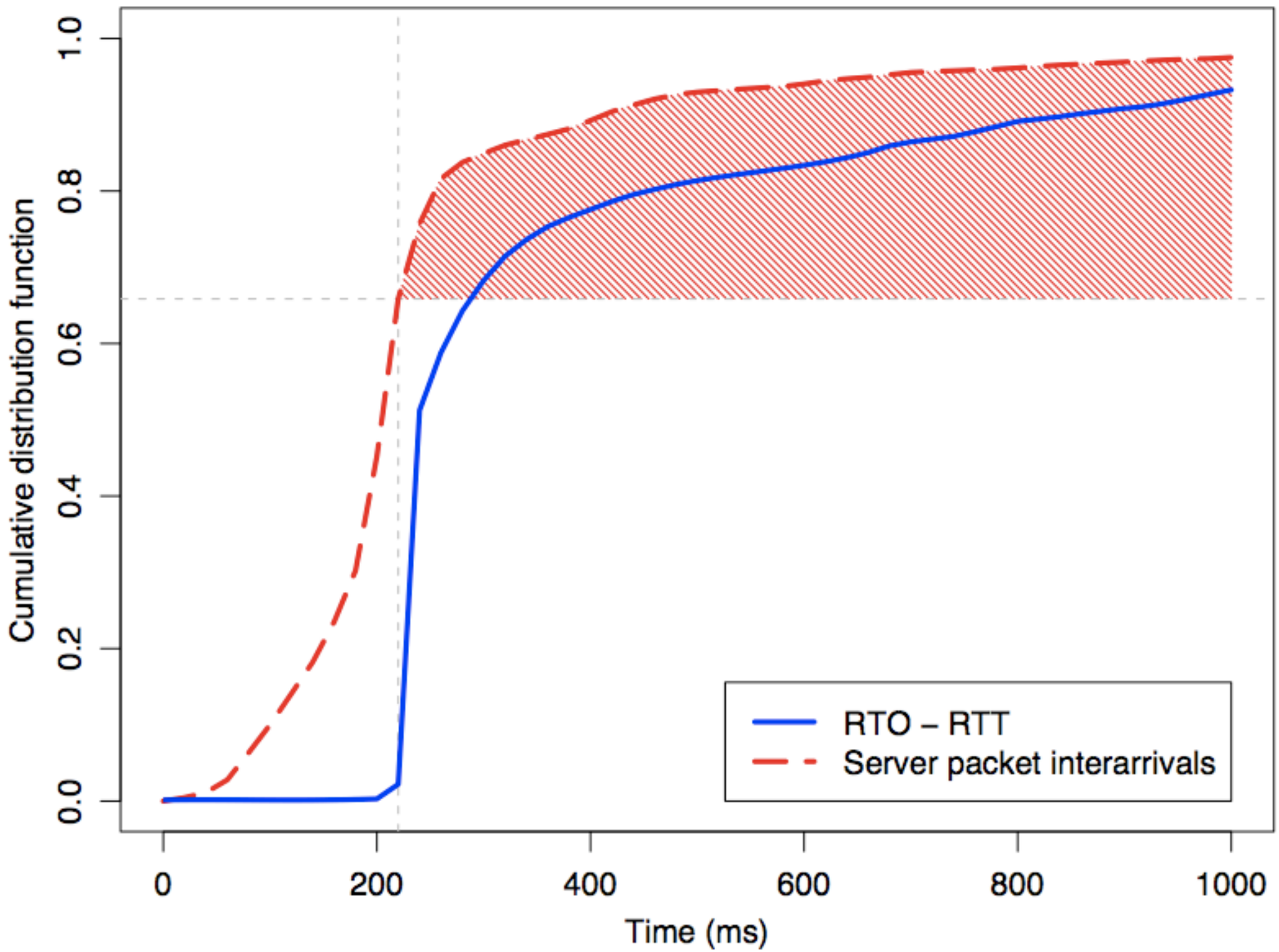
“Thin Streams”

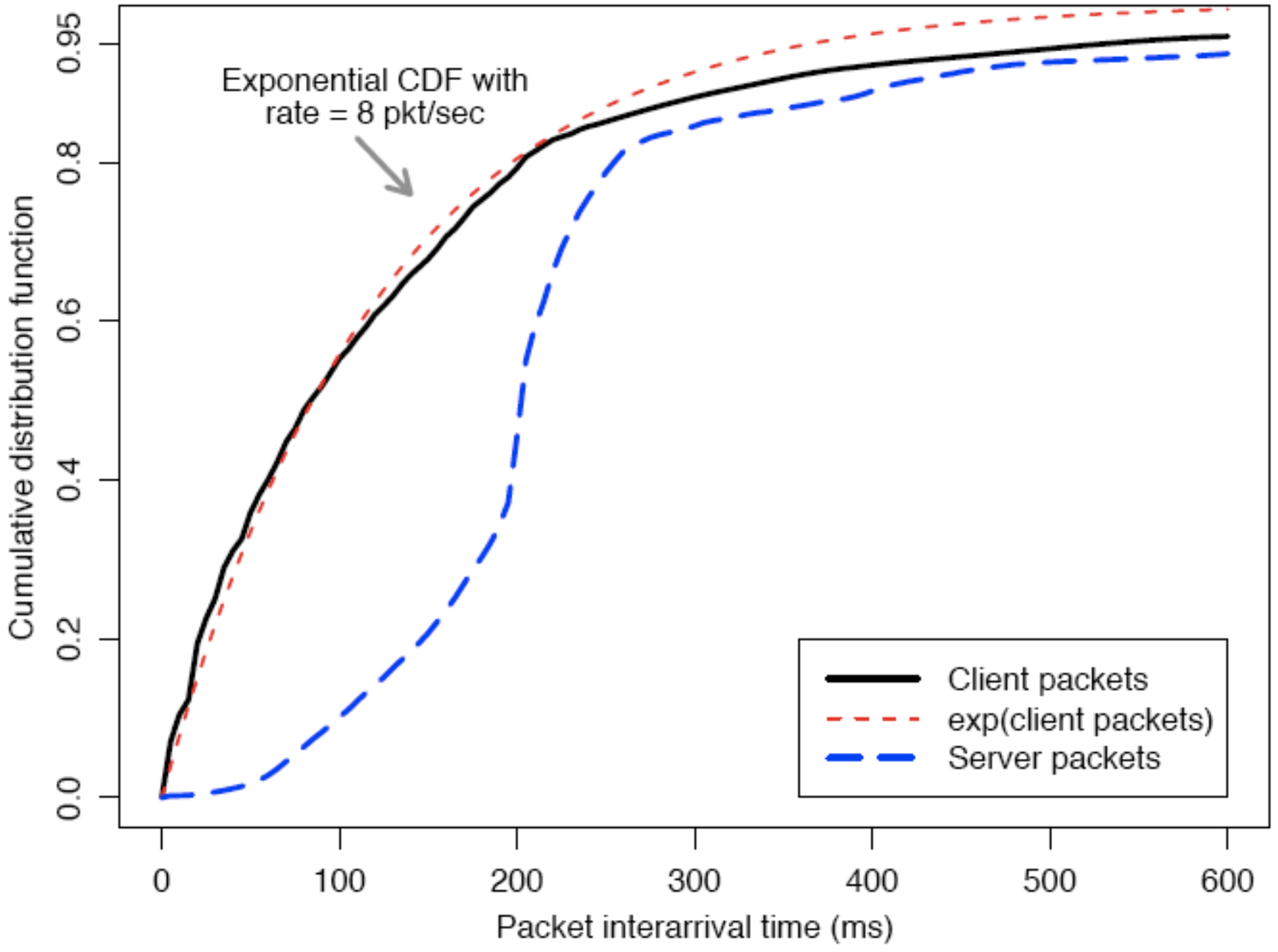
Findings 1:
Fast retransmission rarely triggered

In ShenZhou Online traces, fail to trigger fast retransmission because

insufficient dup ACK (50%)

interrupted by data (50%)


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


Exponential CDF with
rate = 8 pkt/sec
Cumulative distribution function
0.0
0.2
0.4
0.6
0.8
0.95
0
100
200
300
400
500
600
Packet interarrival time (ms)
Client packets
exp(client packets)
Server packets

Findings 2:
Delay due mostly to timeout

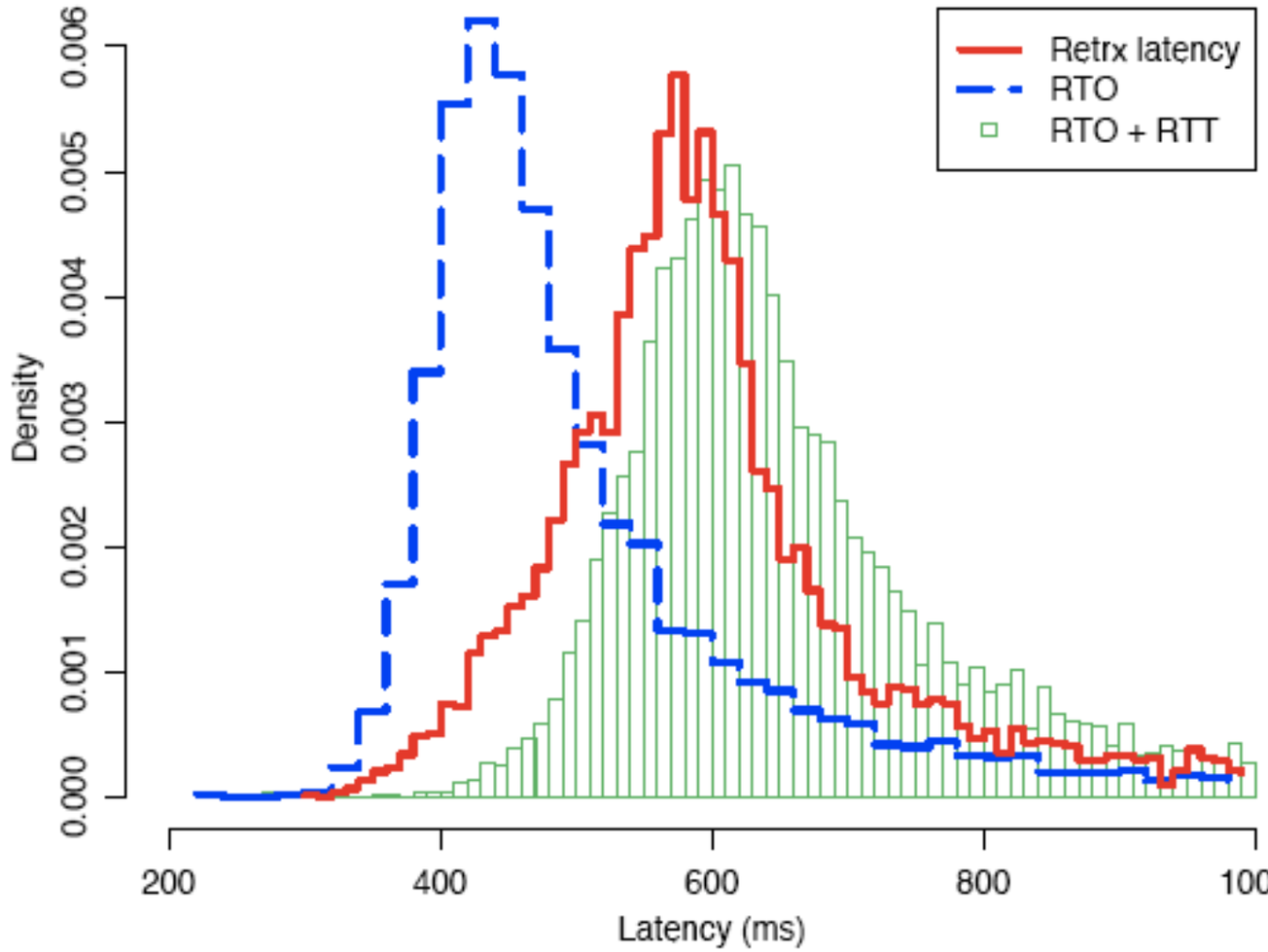
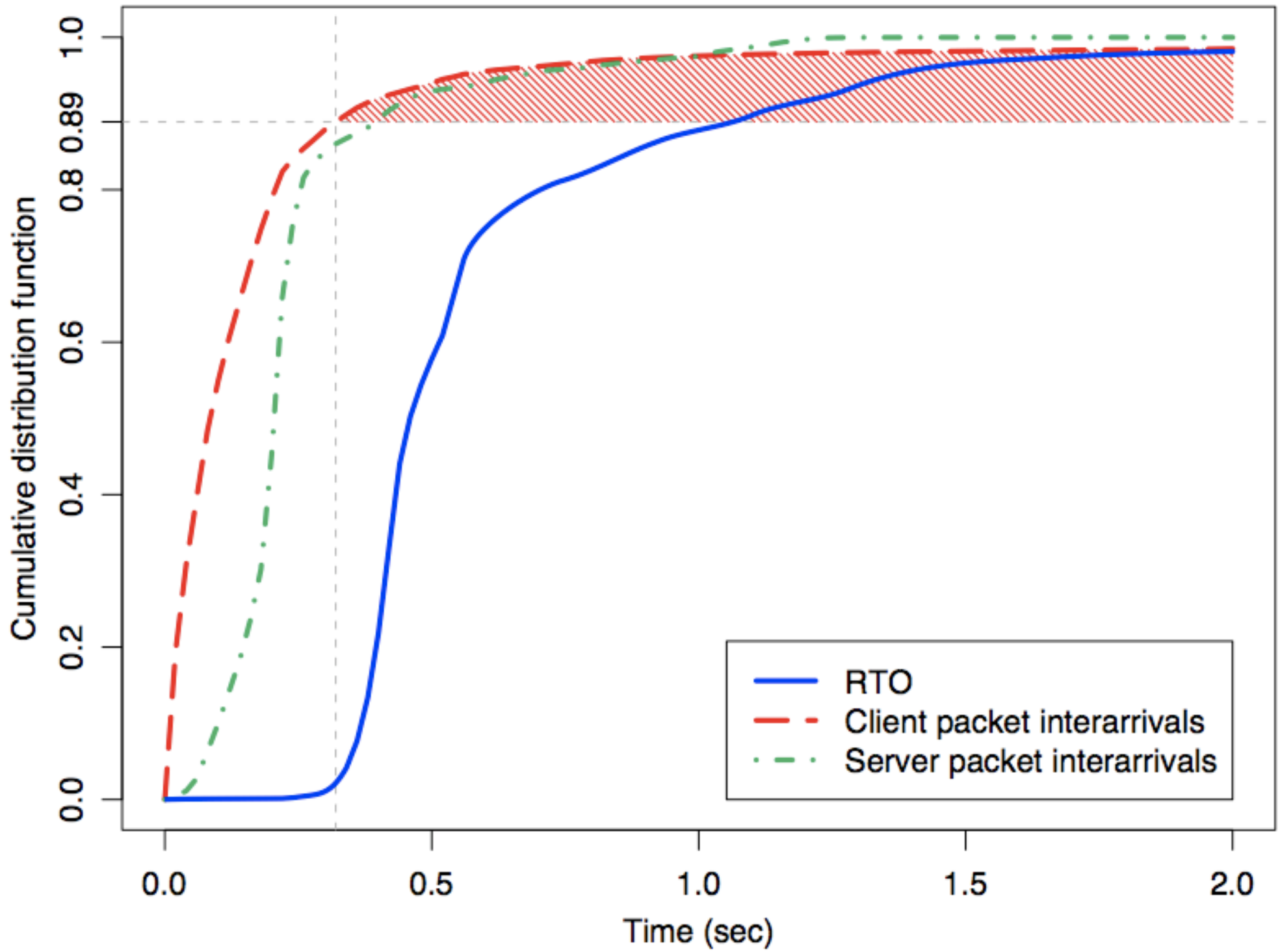


Figure 9: Average latency of dropped packets

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) ⟶

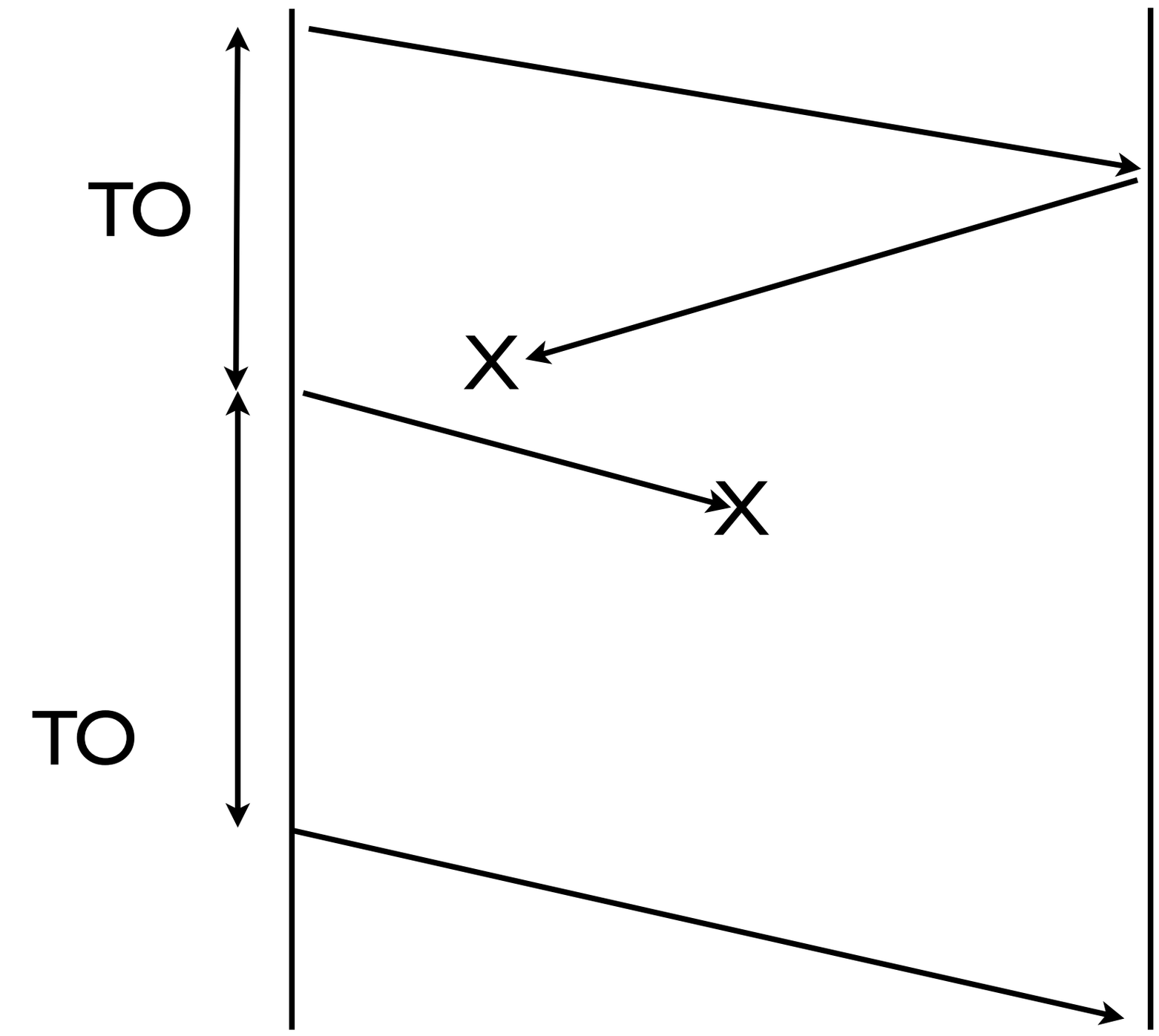
click (missile launch there) ⟶

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



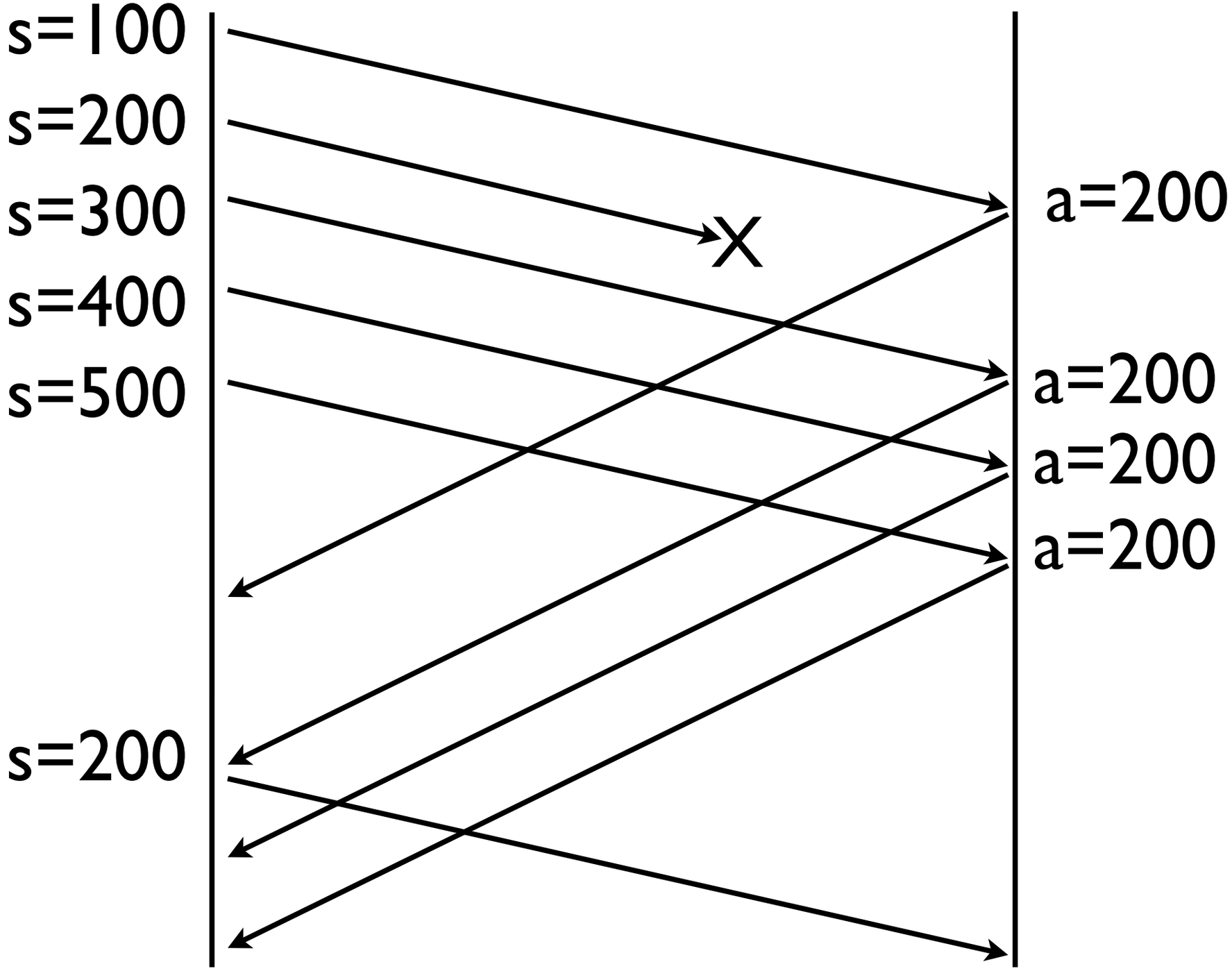
TCP Timeout

2. Make RTO Smaller

make sure minimum RTO is not 1s

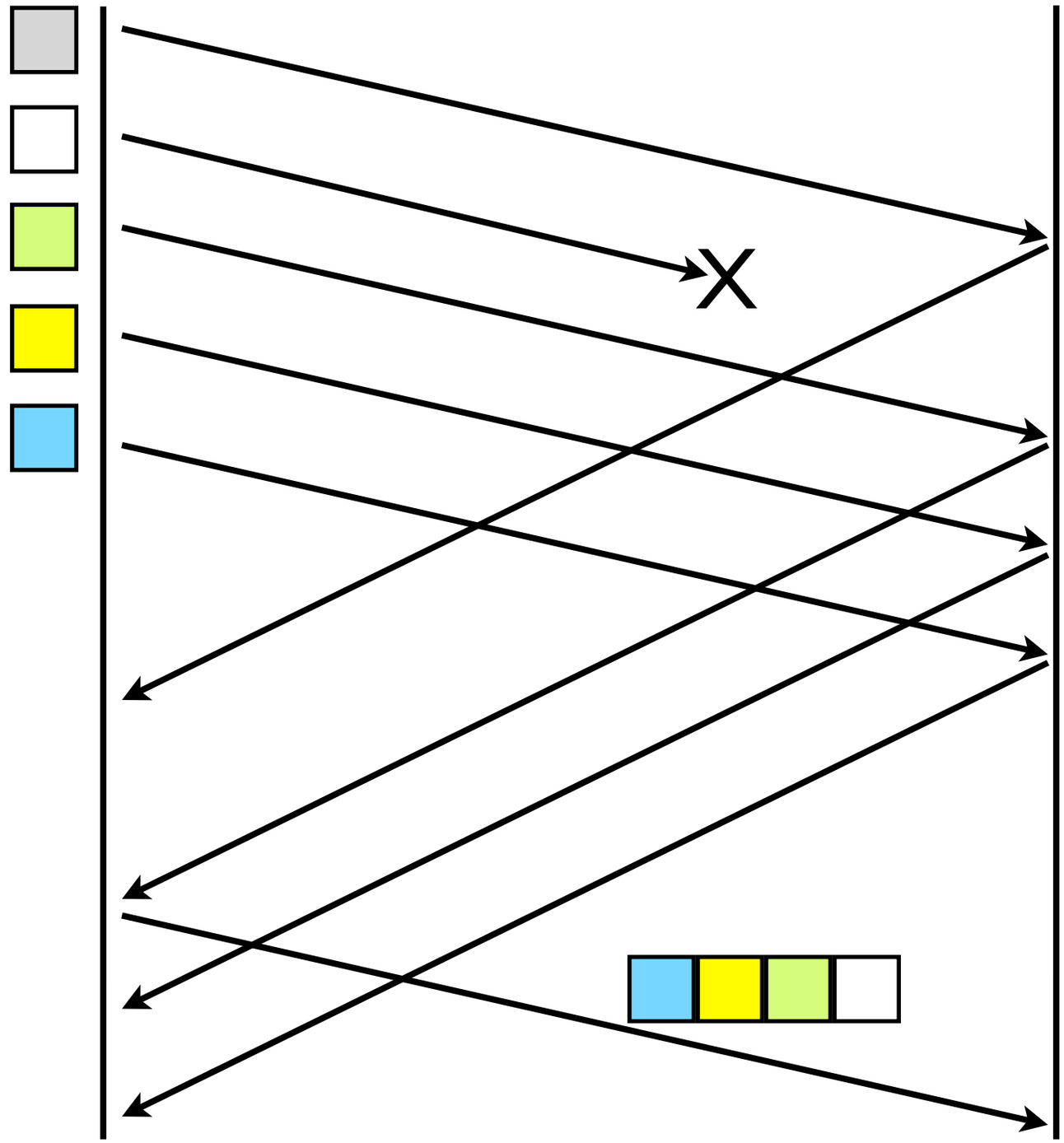
spurious retransmission is not disastrous

3. Make Fast Retransmit Faster



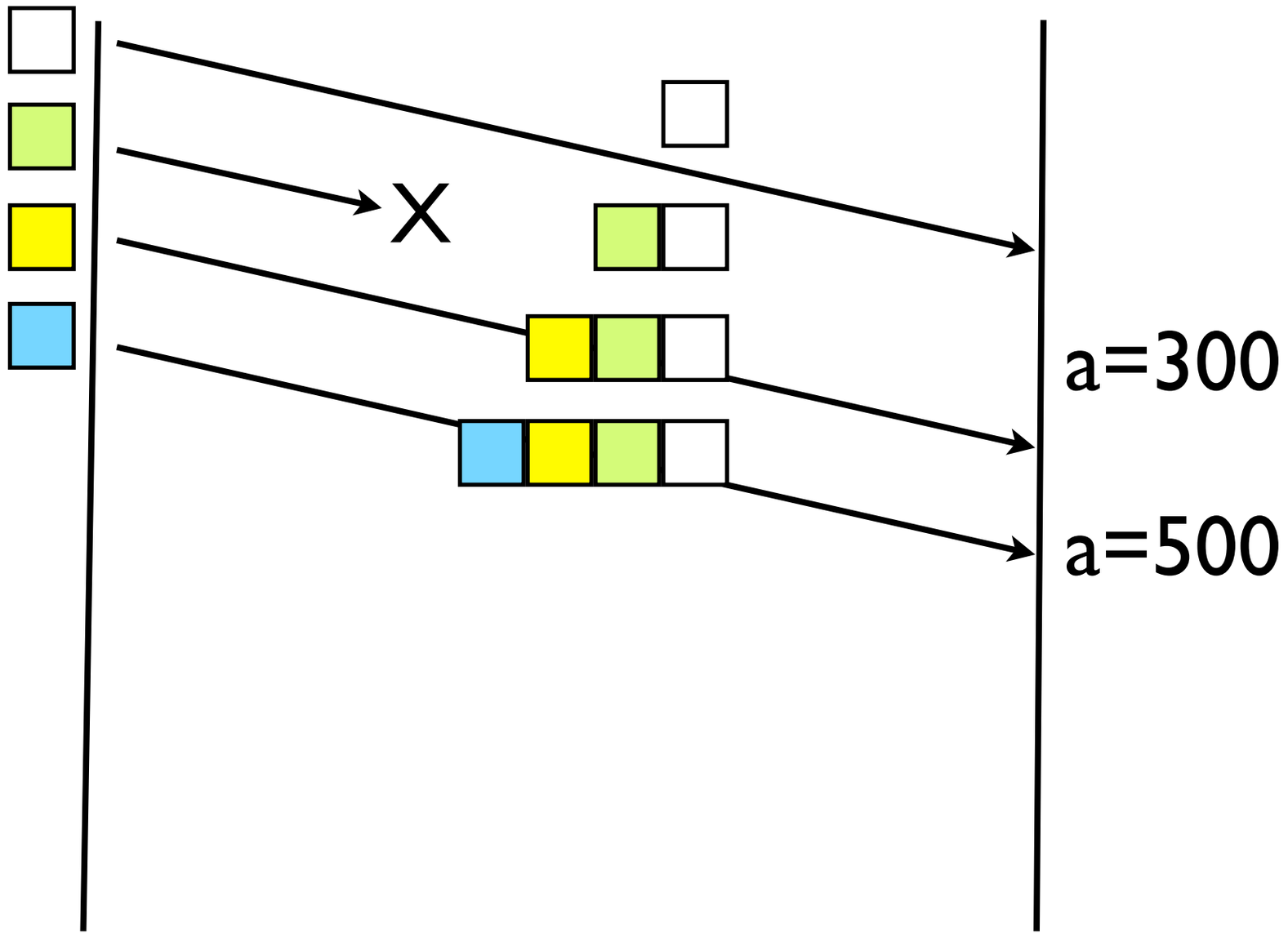
Retransmit after one duplicate ACK

4. Retransmission Bundling



Retransmit all unacknowledge data in queue

5. Redundant Data Bundling

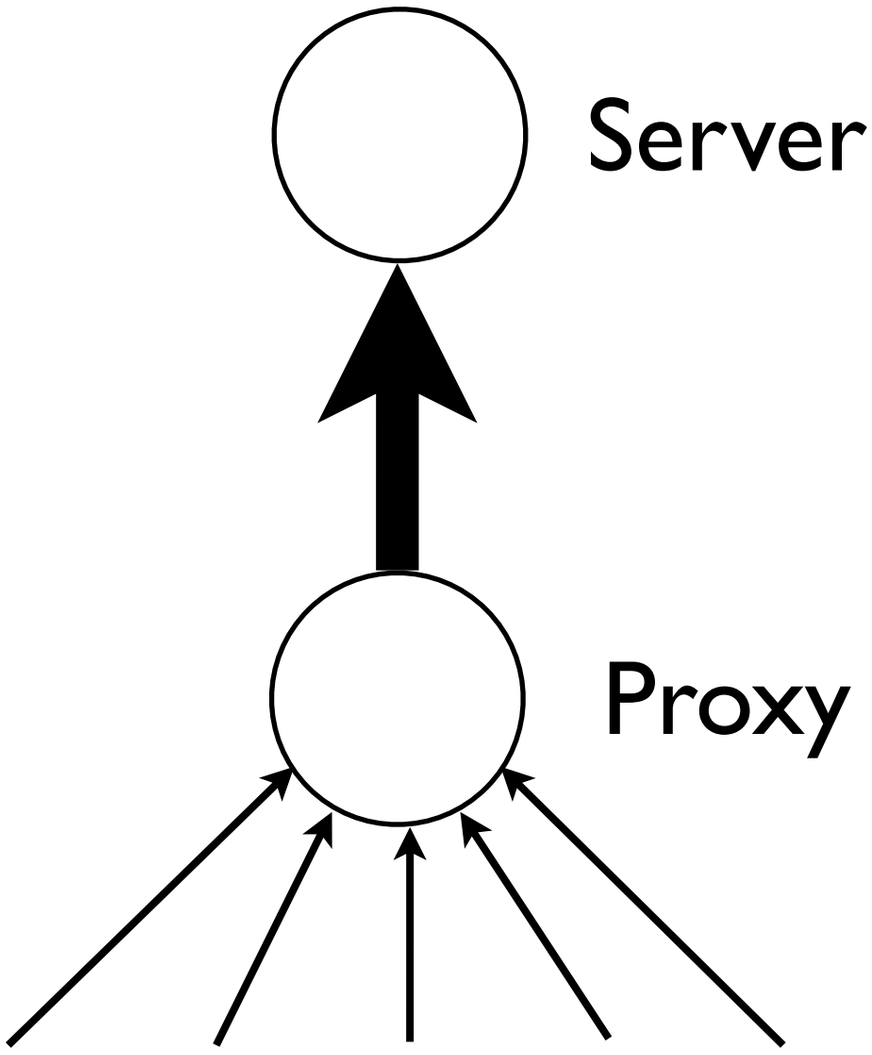


Send any unacknowledged segment in queue as long as there is space. Lost data gets recovered in the next transmission before retransmission.

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


Server
Proxy

TCP for Games

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

Beyond TCP and UDP?

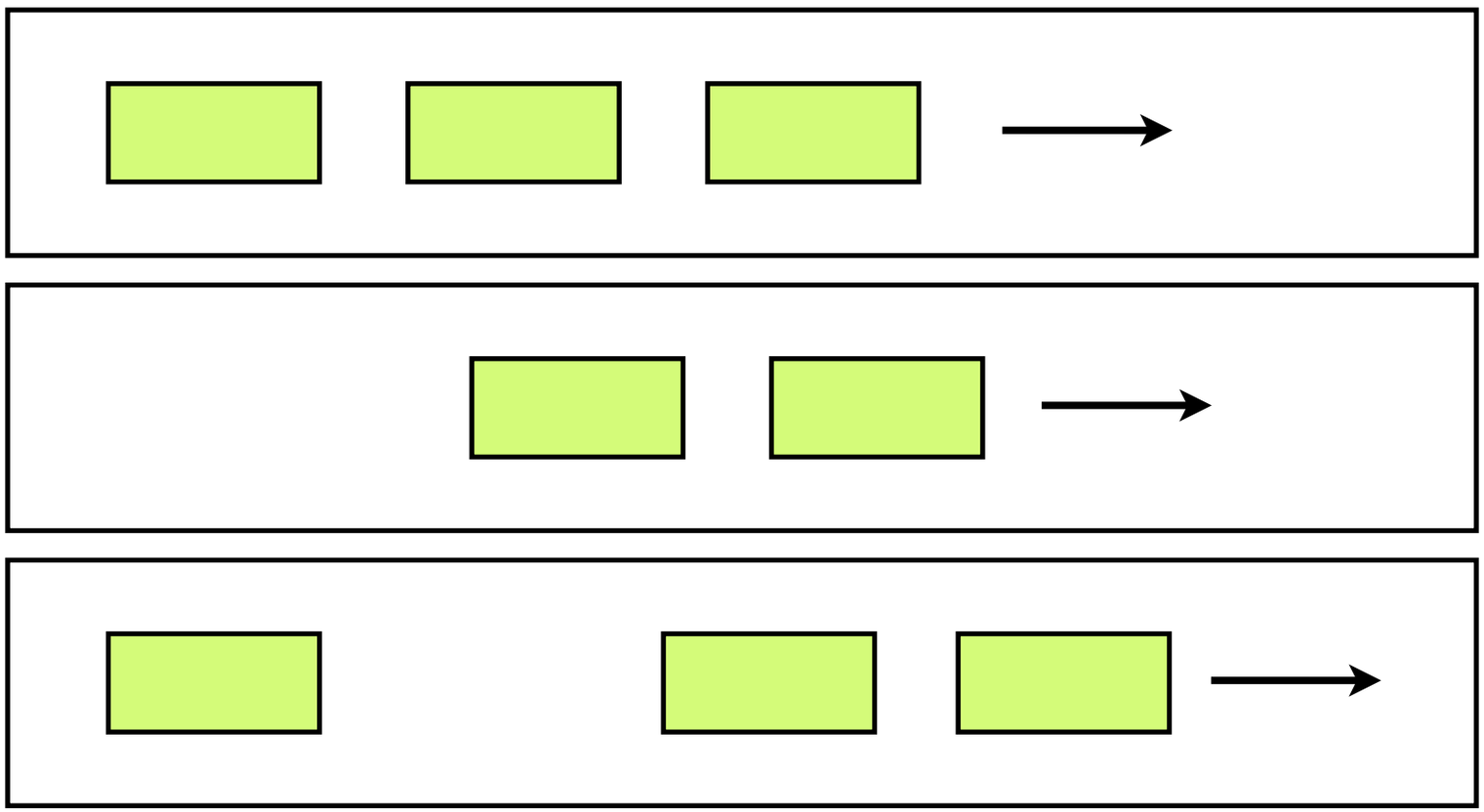
Ideal Transport Protocol for Games

- optional reliability
- optional order-of-delivery
- flexibility in organizing messages into different classes with different requirements

SCTP

Stream Control Transport Protocol

Multi-streaming: multiple independent streams



A stream can be either reliable or non-reliable

Data from multiple streams can be bundled into one packets

Message-oriented (like UDP)

Message can be flagged for unordered delivery

Ideal Transport Protocol for Games

- optional reliability
- optional order-of-delivery
- flexibility in organizing messages into different classes with different requirements

SCTP for Games?

States of SCTP

- not natively available in Windows
- available in many UNIXes