

[ simula . research laboratory ]

# Attacking the Latency Problem in Massive Multiplayer Games

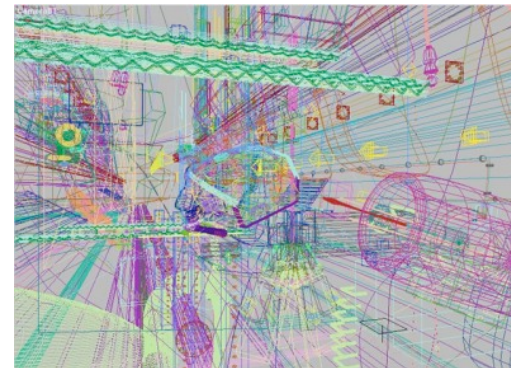
Andreas Petlund, Knut-Helge Vik, Pål Halvorsen, Carsten Griwodz

Simula Research Laboratory, Norway

Email: [griff@ifi.uio.no](mailto:griff@ifi.uio.no)

# Massively Multi-Player Online Games (MMOGs)

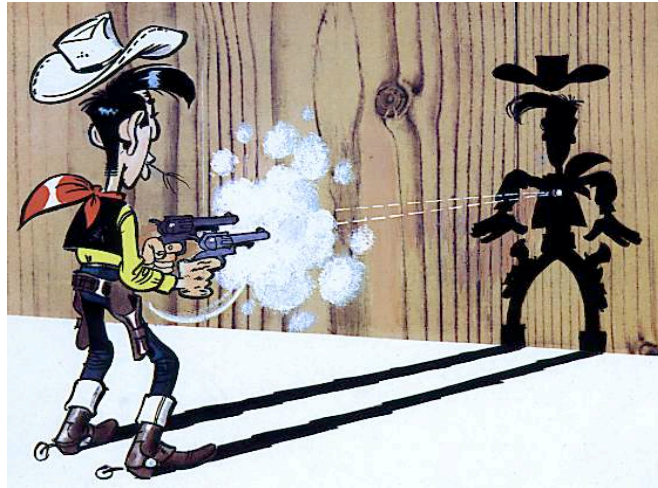
- **Immersive applications**
  - Virtual environments
  - Real-time user interaction
- **Growing user base**
  - 13 million subscribers in 2006
- **Broad usage**
  - Entertainment, education, military, business
- **Large number of concurrent users**
  - Diverse geographical locations
- **Decomposable 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

*Latency is the largest problem for truly large-scale interactive applications*



• **Topic**

- Latency reduction

• **Traces from Funcom's Anarchy Online**

- Single virtual region

• **Depicted scenario**

- Geographically distributed servers
- Skewed connectivity

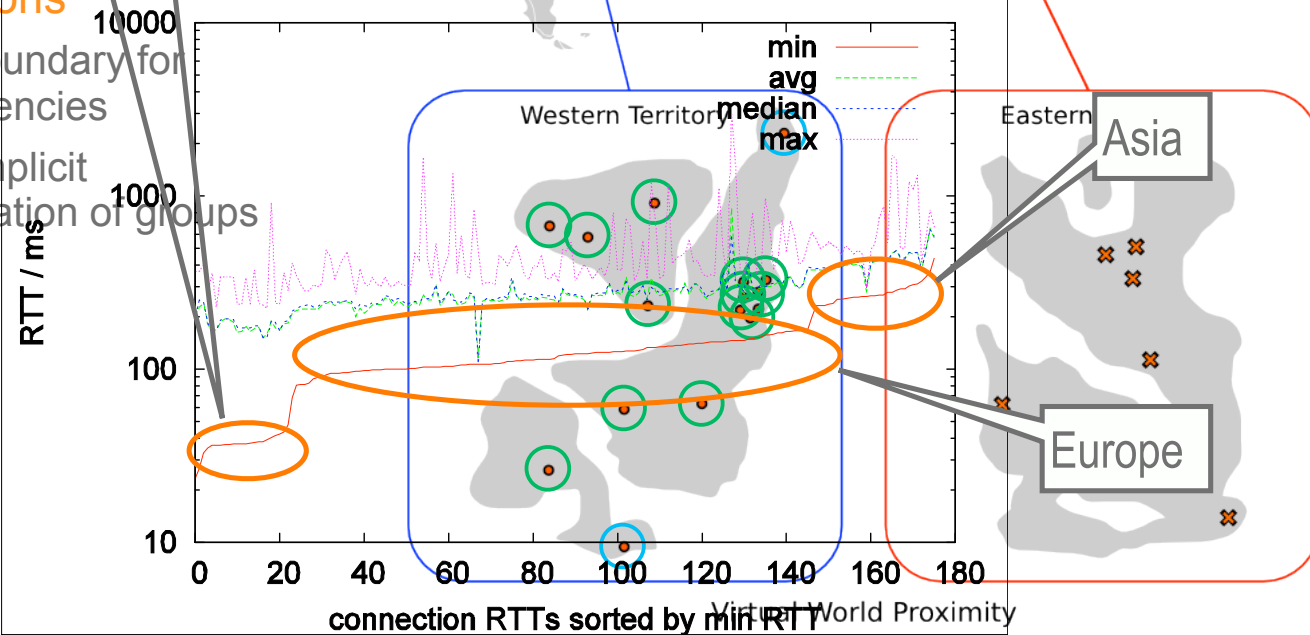
- Based on time of day
- Physical location is unrelated to virtual location

- Server located in North America
- Spanning approximately one hour

• **Virtual regions**

- Mark boundary for dependencies
- Allow implicit specification of groups

RTT statistics for all packets that are never retransmitted



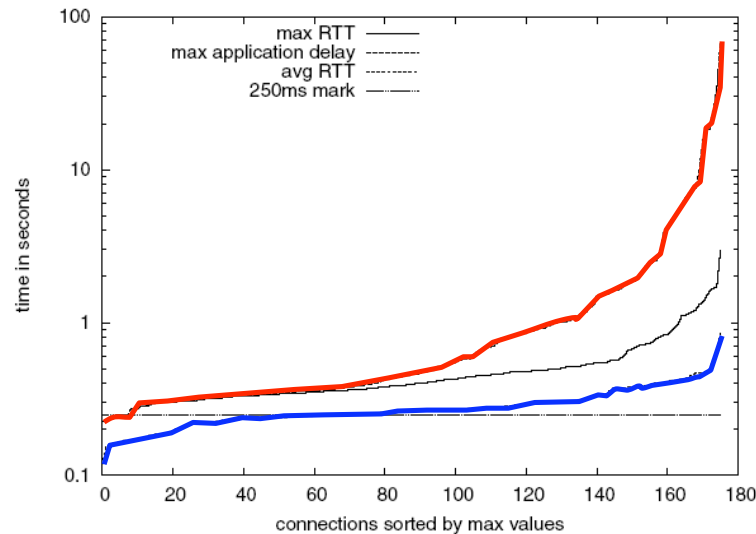


# Thin TCP Streams

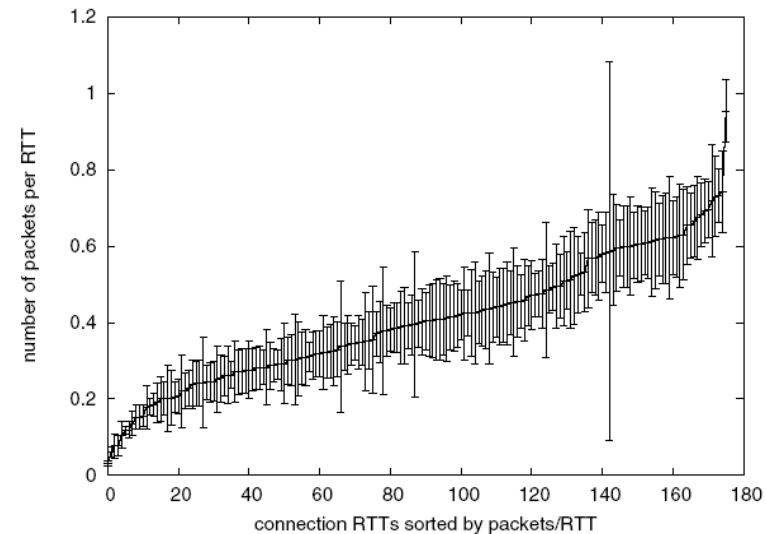
---

# Thin Streams

- Transport protocols being developed for throughput-bound applications
- ↪ BUT, there exist several **low-rate, time-dependent** applications
- Anarchy Online MMORPG Case Study



(a) RTT versus maximum application delay



(b) Packets per RTT with standard deviation

- 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

Application	Average payload size (byte)	Packet interarrival time (ms)	Bandwidth requirements (bps)
<b>Anarchy Online</b>	93	909	1757
<b>Counterstrike</b>	142	81	19764
<b>BZFlag</b>	30	24	31370
<b>Skype</b>	111	30	37906
<b>CASA (radar control)</b>	175	7287	269
<b>Windows remote desktop</b>	111	318	4497
<b>SSH text session</b>	48	323	2825
<b>MPEG-2 streaming</b>	1460	3	~4200000

- 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

Fast retransmit

wait until it's likely that a packet is lost (ACKs for 3 "younger" packets arrived)

- timeout is a fallback

- when no ACKs arrive

Exponential backoff

double timeout waiting time, retransmit again

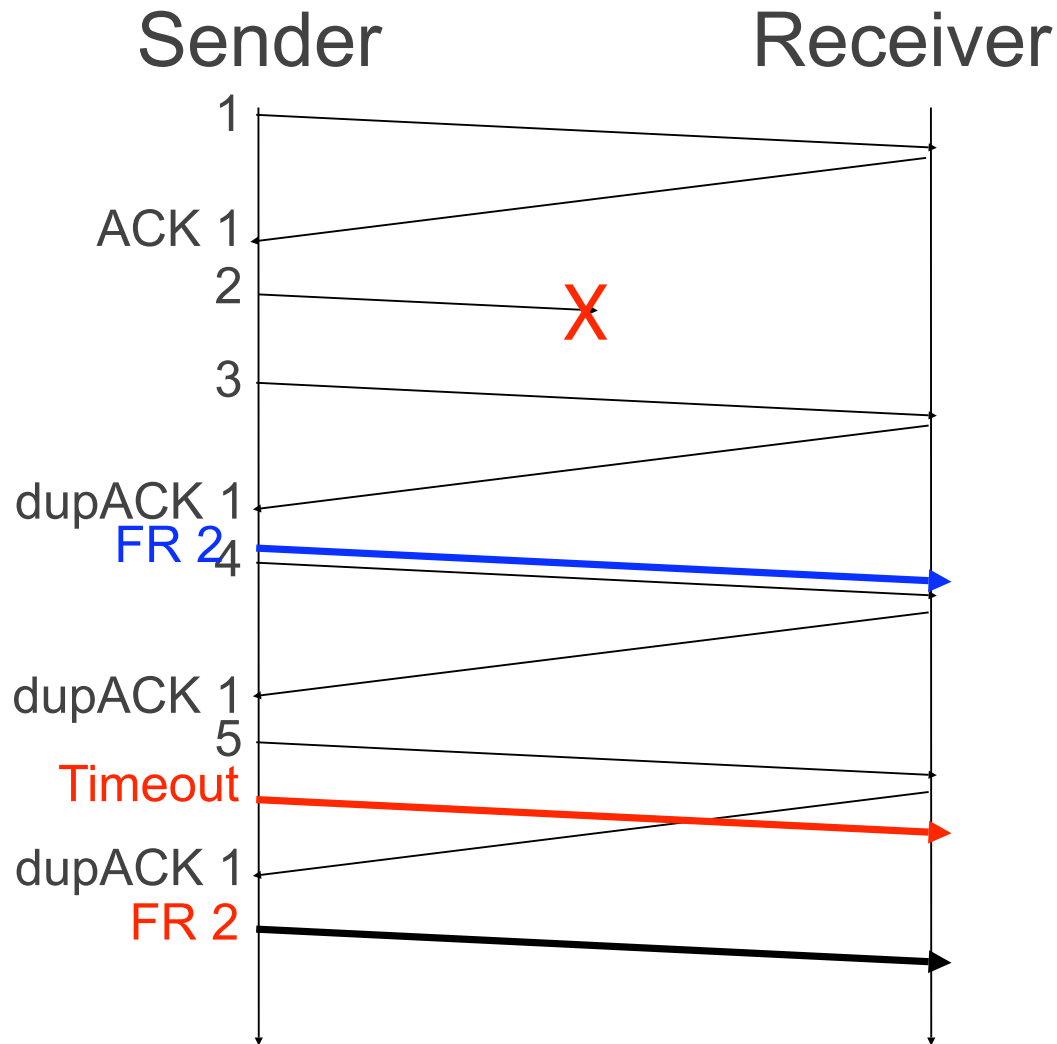


# Thin-stream detection

- 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:  $\text{size\_unacked}(p1) + \text{size}(p2) < \text{MSS}$
- Apply modifications only when criteria are fulfilled.
- Make modifications available on a per-stream basis (using socket options)

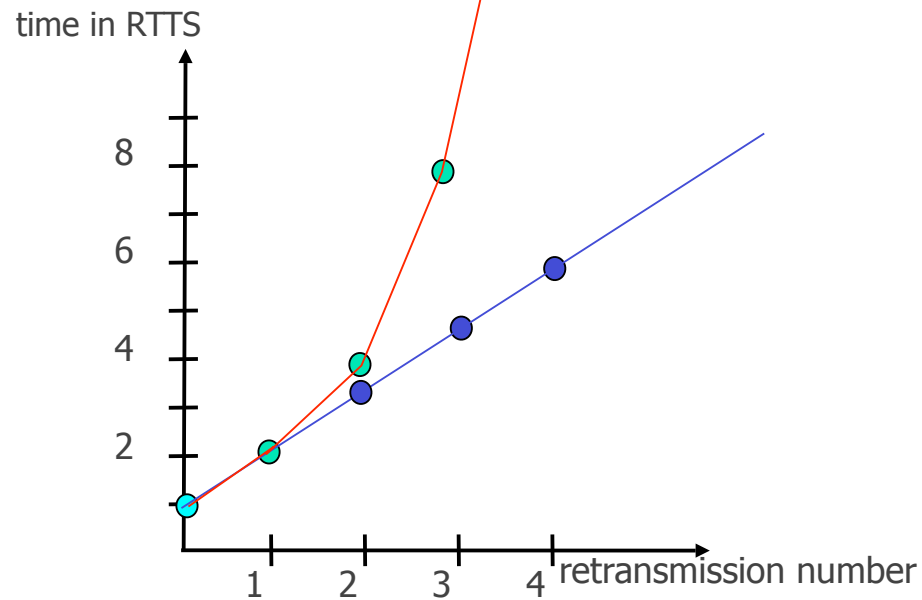


# Fast retransmit with thin-streams



- 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

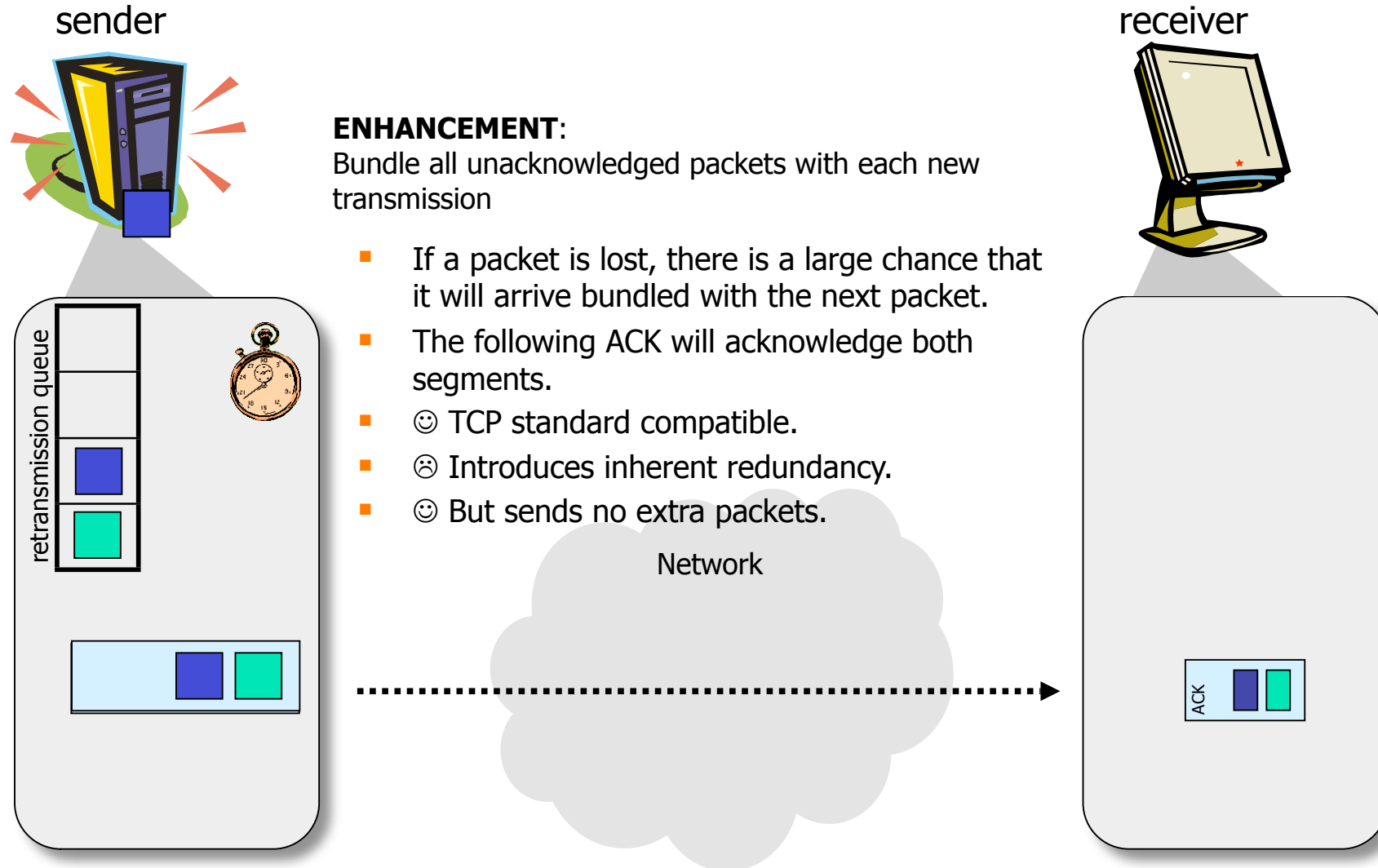


Lost packet

1. retransmission
2. retransmission
3. retransmission
4. retransmission





When thin streams are detected,  
linear timeouts are used.

# Redundant Data Bundling



# Thin stream mechanism applicability

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

	Small Packets 	Large Packets 
High IA 	Typical thin stream RDB, retrans, backoff	Rare faster retransmit, backoff
Low IA 	Rare RDB	FTP, HTTP Thick

# 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.

## Clip 1

[Version 1](#)  
[Version 2](#)

Version 1   Version 2   Equal



## Clip 2

[Version 1](#)  
[Version 2](#)

Version 1   Version 2   Equal



## Clip 3

[Version 1](#)  
[Version 2](#)

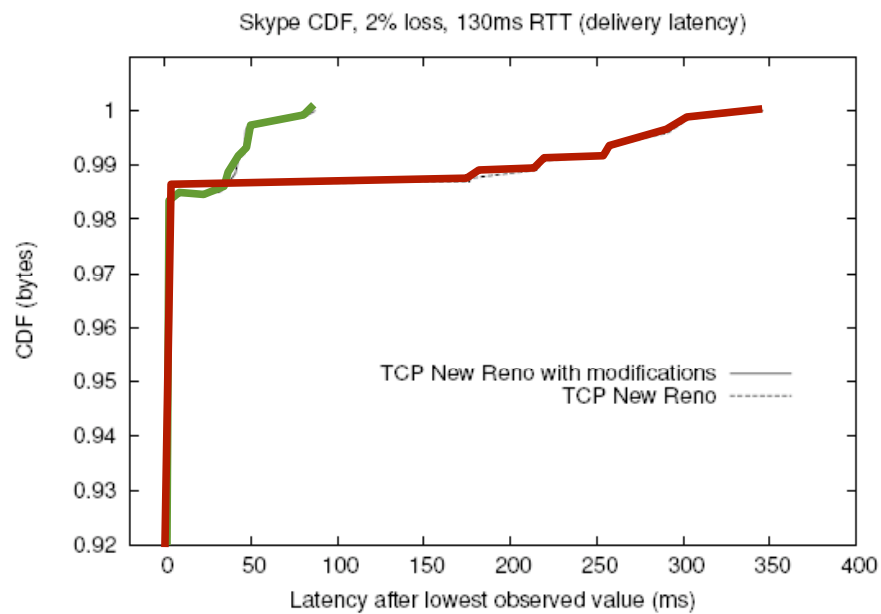
Version 1   Version 2   Equal



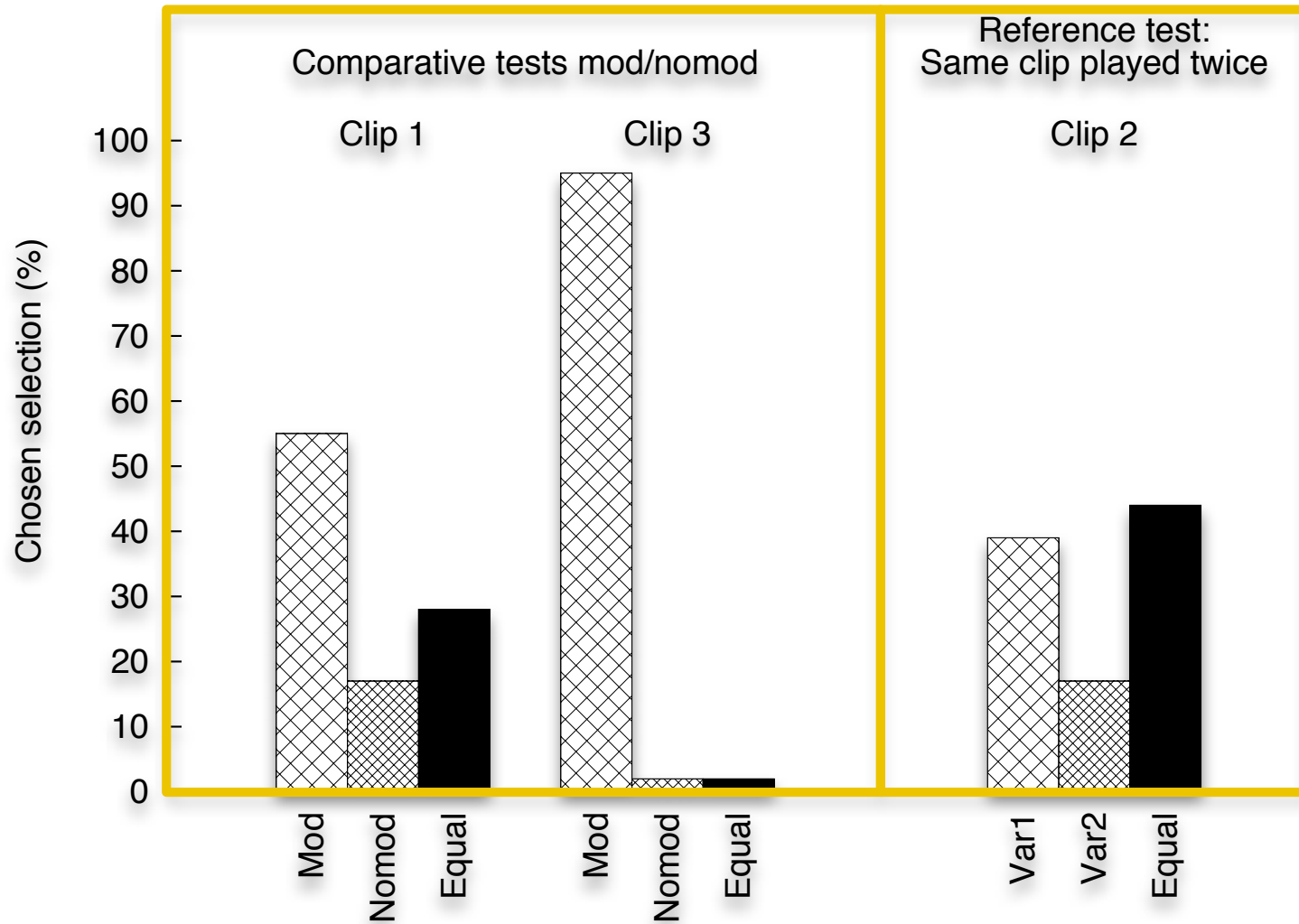
# Data trace analysis: Skype

Transport layer delay

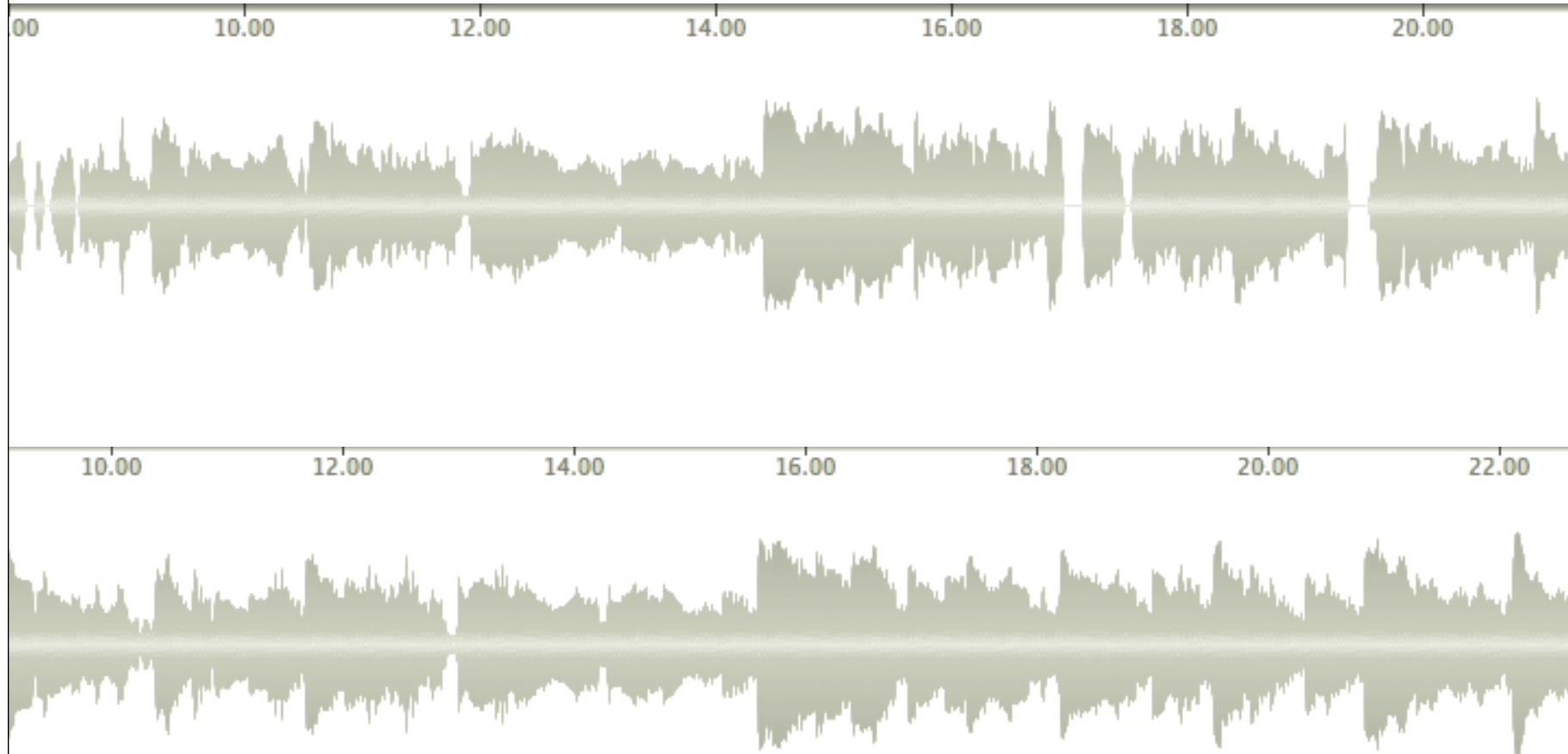
Application layer delay



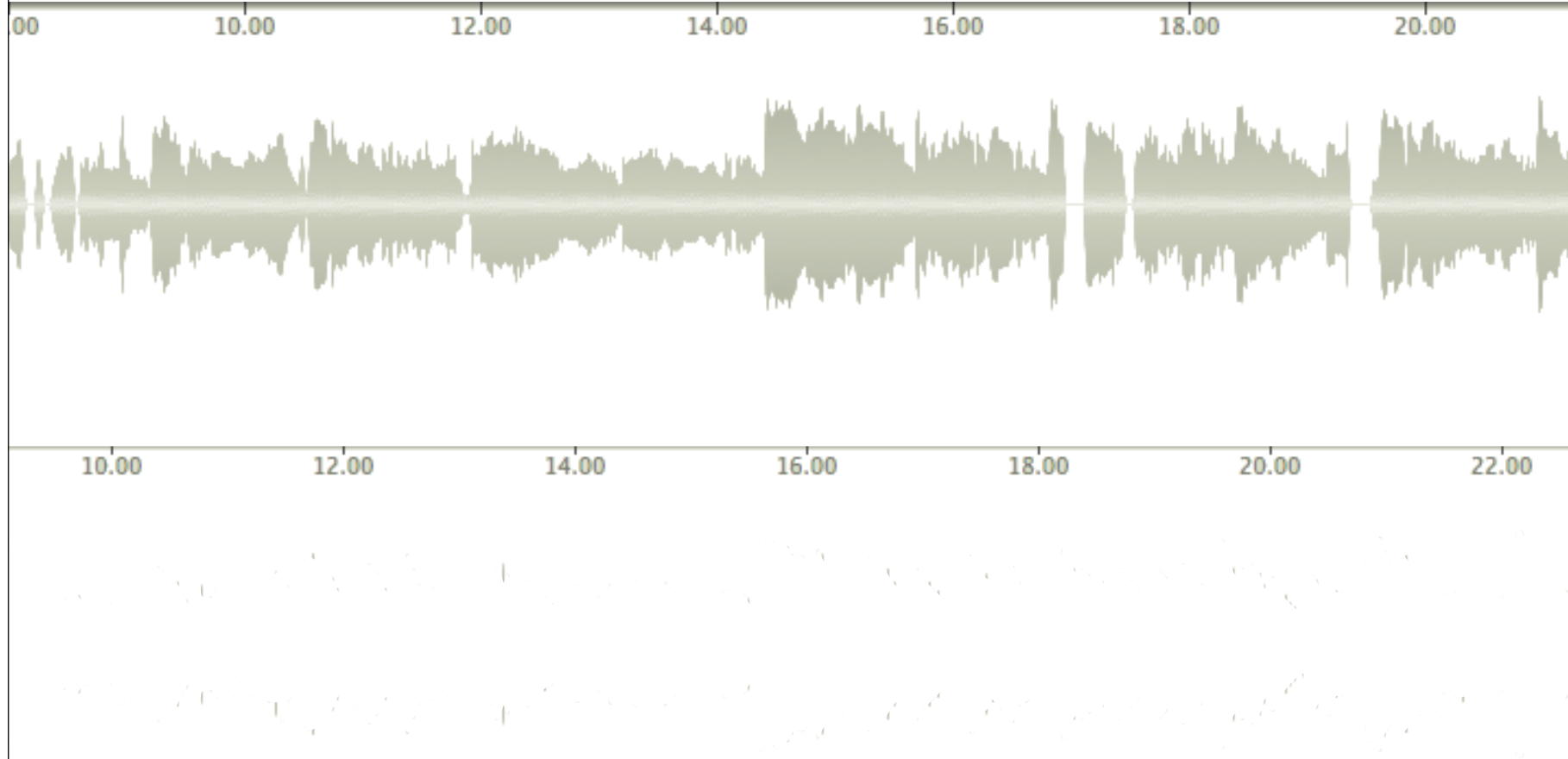
# Skype survey results



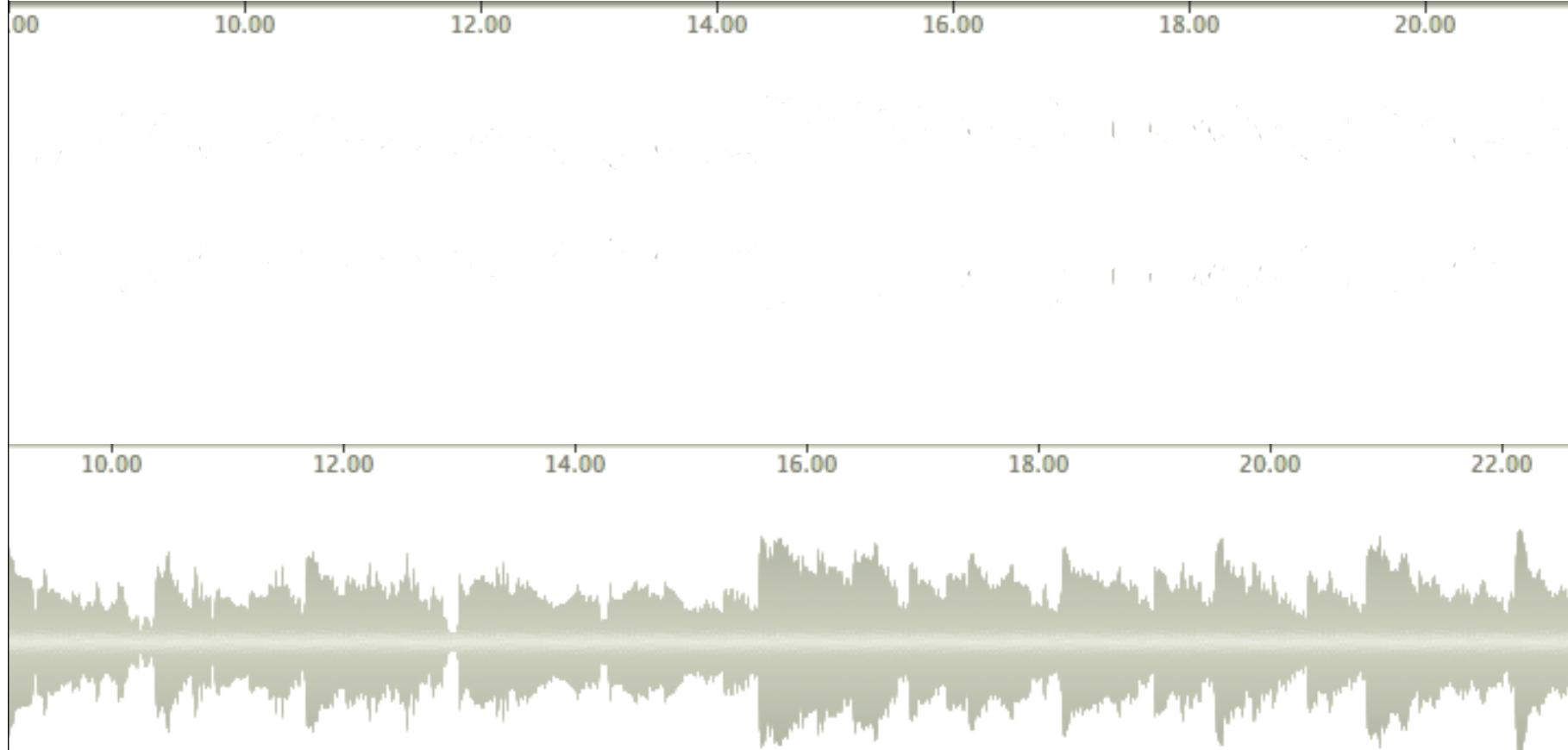
# Internet latency improvements



# Internet latency improvements



# Internet latency improvements

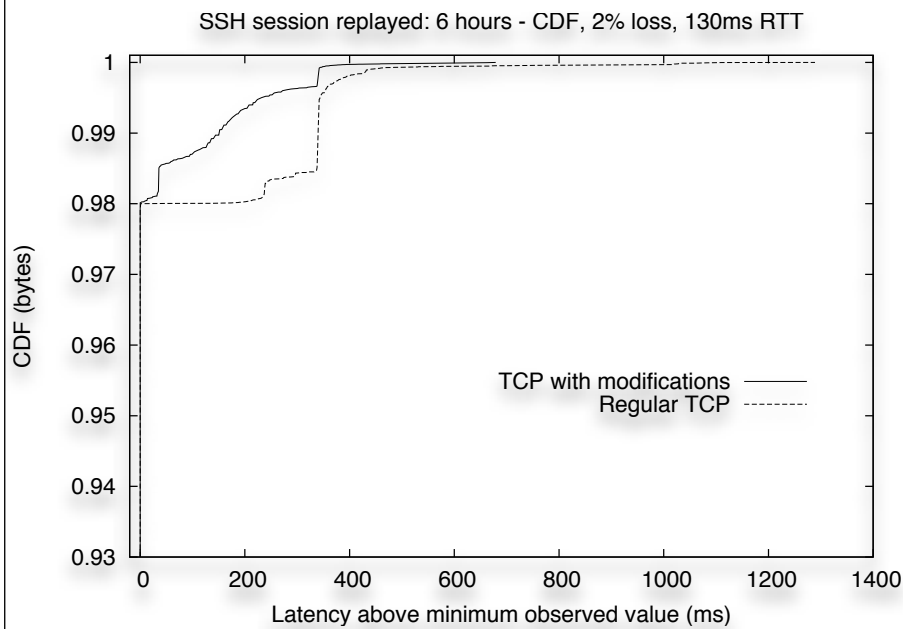


## Test setup 2: SSH text session

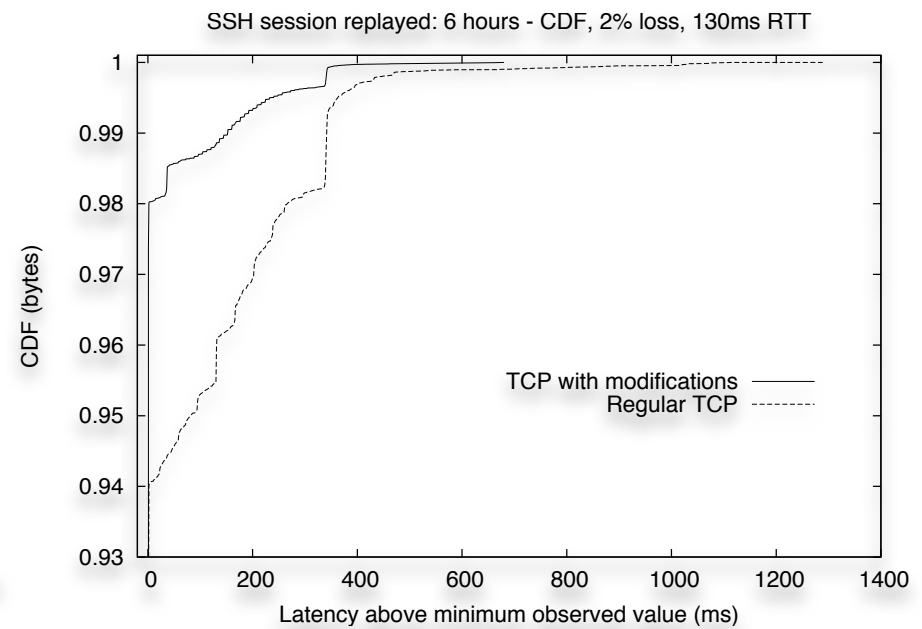
- Each user was given a set of tasks to perform using a text-based editor (vi or emacs) over a lossy network.
  - 2% loss, 130ms RTT
- The same tasks were to be performed twice.
  - The user was then asked to evaluate the quality of the text session with regard to noticeable delay and interruptions.
- We varied which TCP version (with- or without modifications) was used first.

# Data trace analysis: SSH

## Transport layer delay

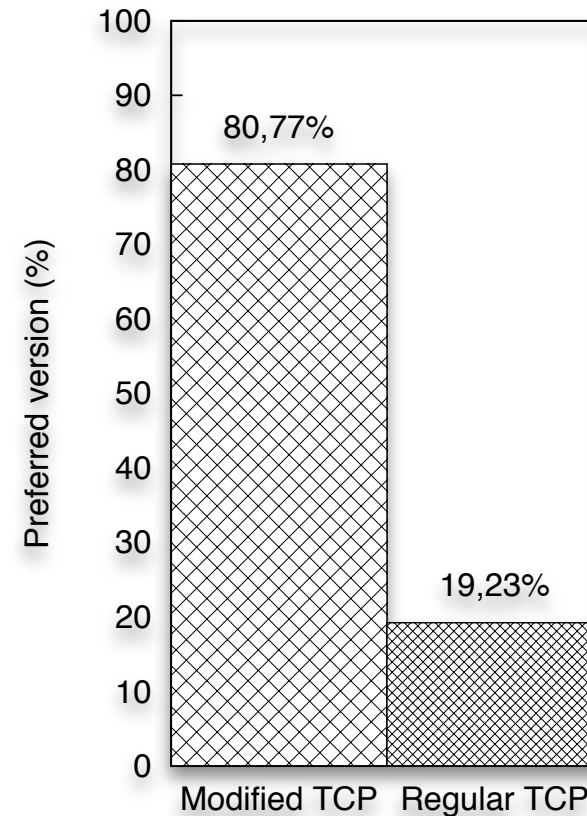


## Application layer delay



# 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