



# An Empirical Evaluation of TCP Performance in Online Games

Kuan-Ta Chen

National Taiwan University

Collaborators: Chun-Ying Huang  
Polly Huang  
Chin-Laung Lei



# Talk Outline



- Motivation
- Trace collection
- Game traffic characteristics
- Analysis of TCP performance
- Design Implications
- Conclusion

# Motivation

- TCP is generally considered **not suitable for real-time transmission**
- All first-person shooting games use UDP
- No consensus has been reached for MMORPGs
  - Heated debate occurs at game developers' forums
  - Each protocol has quite a few proponents

Protocol	MMORPGs
TCP	World of Warcraft, Lineage I/II, Guild Wars, Ragnarok Online, Anarchy Online, Mabinogi
UDP	EverQuest, Star Wars Galaxies, City of Heroes, Ultima Online, Asherons Call, Final Fantasy XI
TCP+UDP	Dark Age of Camelot

# Motivation (cont.)

Q1: Is TCP appropriate for MMORPGs?

Q2: If not, how to design a suitable protocol?

We try to answer these problems by  
empirical evidence.

# Key Contributions / Findings

- TCP is **unwieldy** and **inappropriate** for MMORPGs
- Estimate **how long users will stay** if a more suitable protocol is used (**100 minutes → 135 minutes**)
- Proposed a number of **design guidelines** which exploit the unique characteristics of game traffic

# 神州 ONLINE

## *ShenZhou Online*

- A commercial MMORPG in Taiwan
- Thousands of players online at anytime
- TCP-based client-server architecture

神州城

	姓名	小寬
	職業	劍俠
	名聲	無名小卒
	生日	水曜日
	負重	27 / 75

241 / 241

93 / 93

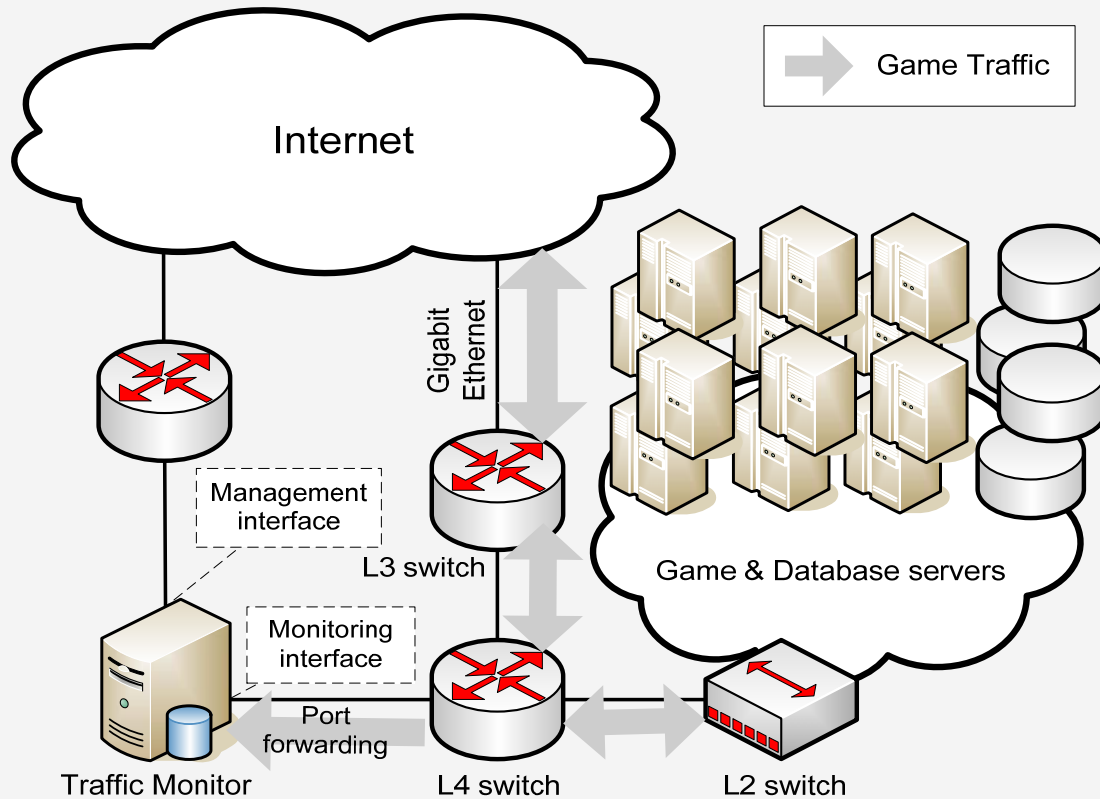
抓圖成功(cap\cap0059.jpg)◦  
冰心雪>今天打蛇才賺為70萬  
抓圖成功(cap\cap0060.jpg)◦  
冰心雪>白鹿還死為一次  
冰心雪>忘為看血 故看電視 QQ"  
抓圖成功(cap\cap0061.jpg)◦

6 級

7610

密名

# Trace Collection



Trace Time	# TCP Conn	# Packets	# Bytes
20 hours	112,369	1,356 million	58.5 TB

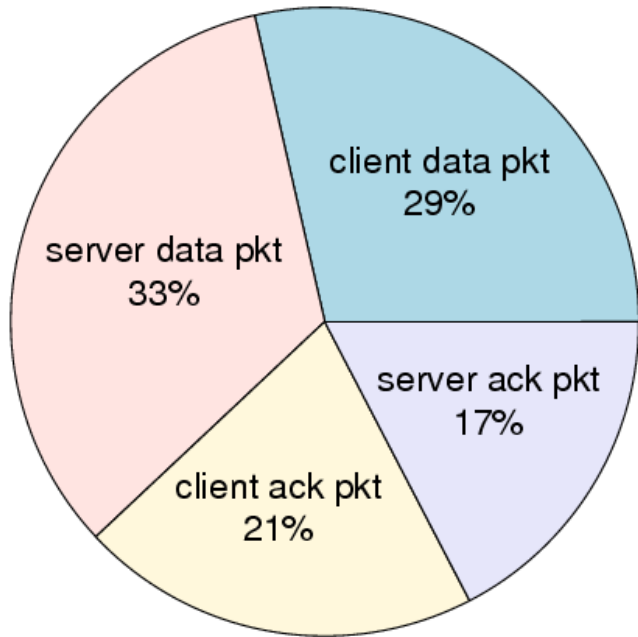
# Game Traffic Characteristics

- **Small packet size**
  - Most of client payload < 32 bytes
  - Average pkt size 84 bytes
- **Low packet rate**
  - Client packet rate is 8 pkt/sec (avg.)
  - Most of server packet rate < 5 pkt /sec

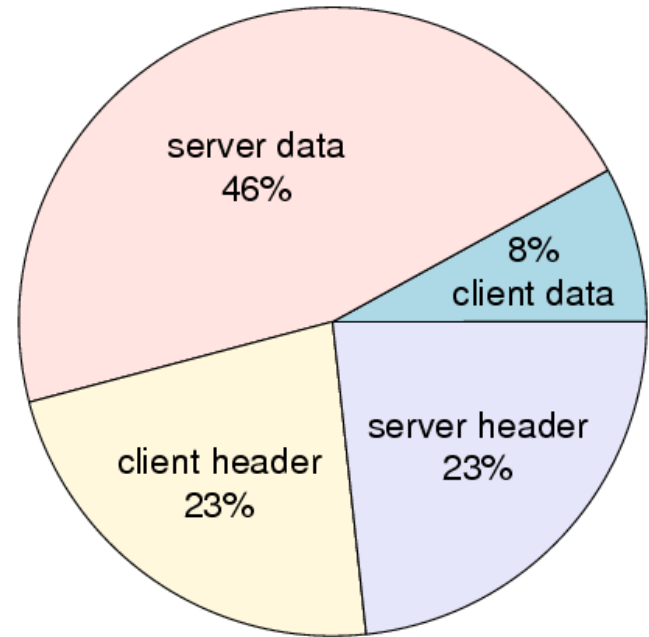
# TCP Performance Analysis

- Protocol overhead
- In-order delivery
- Congestion control
- Loss recovery

# Protocol Overhead



(a) Packet count



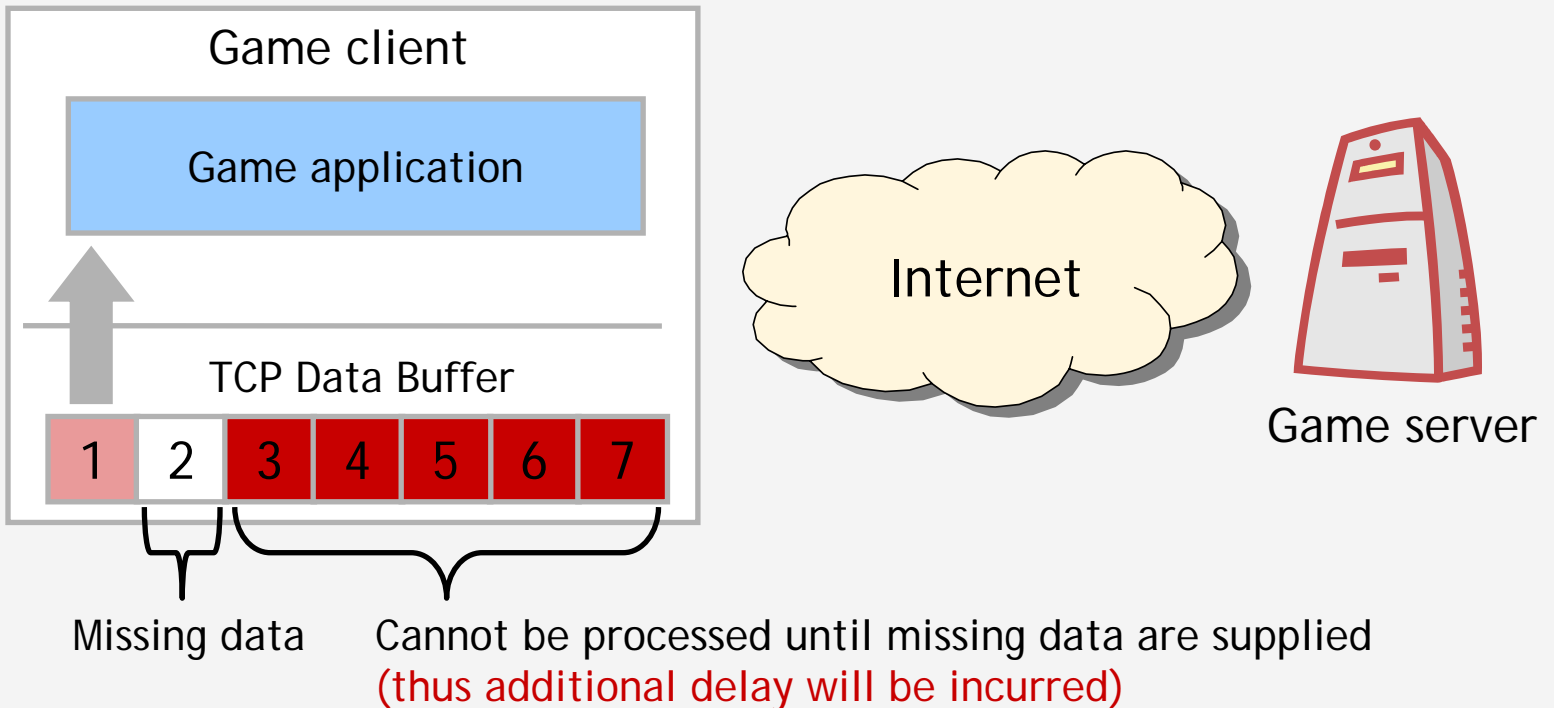
(b) Byte count

38% packets are TCP ACK packets

46% bandwidth are used by TCP/IP headers

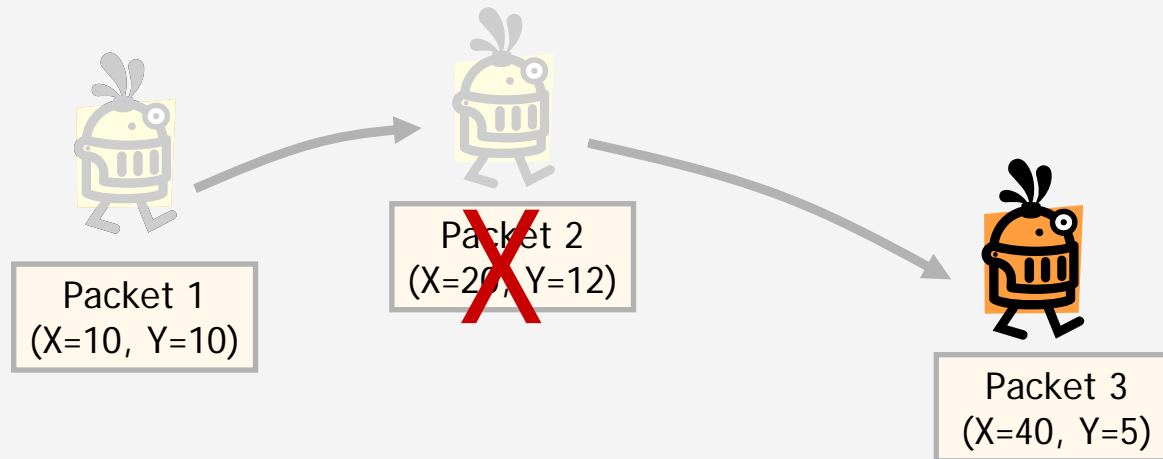
# In-Order Delivery

- TCP enforces byte-level in-order delivery
- Cons: a dropped packet causes a **stall** in subsequent network data until that packet is delivered



# In-Order Delivery is **Not** Always Required

- Server to client: state updates, dialogue messages, responses to users' queries
- Client to server: many commands are **accumulative**, e.g., position updates

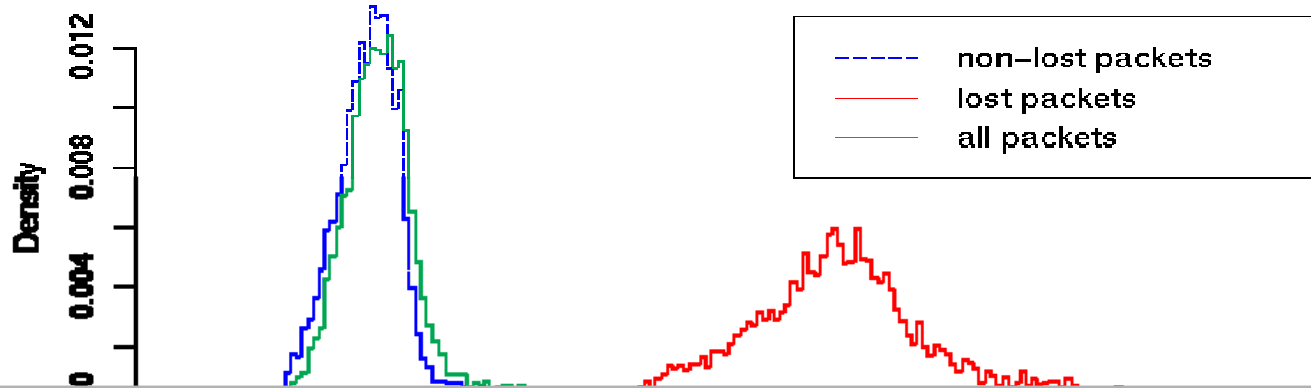


Processing packet 3 **without** waiting for packet 2 is **OK**  
and leads to **more smooth game play**

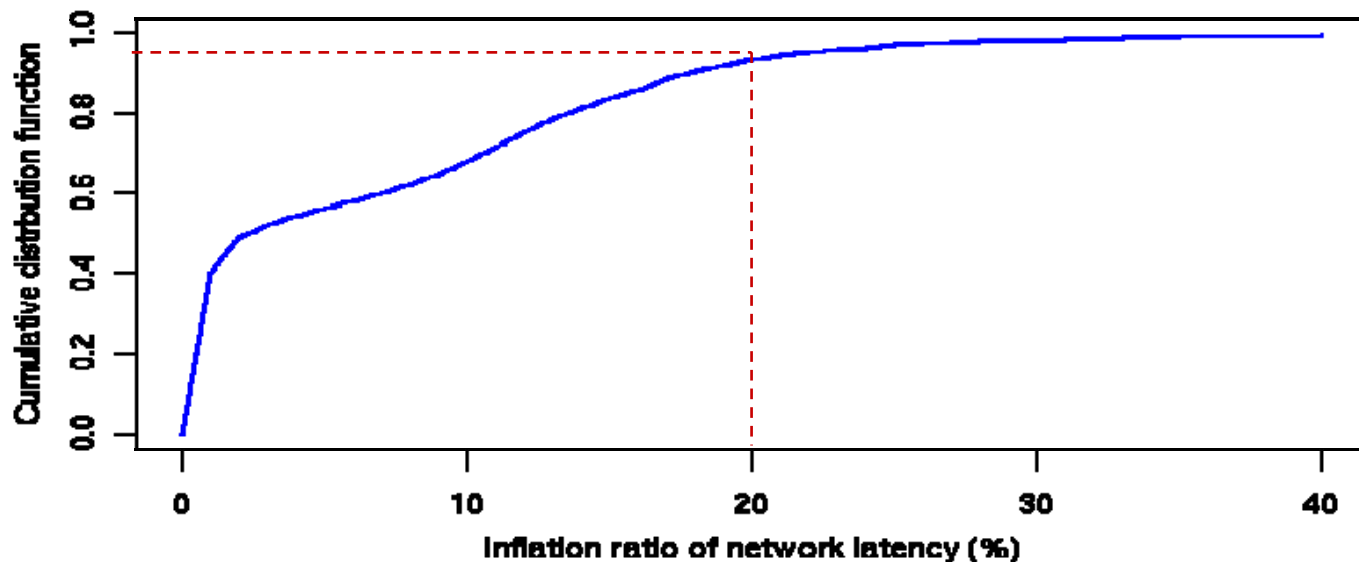
# Overhead of In-Order Delivery

- Assuming an ideal case that all packets can be processed in any order
- The overhead of in-order delivery is measured by
  - The increase of round-trip times (RTT)
  - The increase of RTT jitters (std. dev.)

# The Effect of In-Order Delivery on RTT

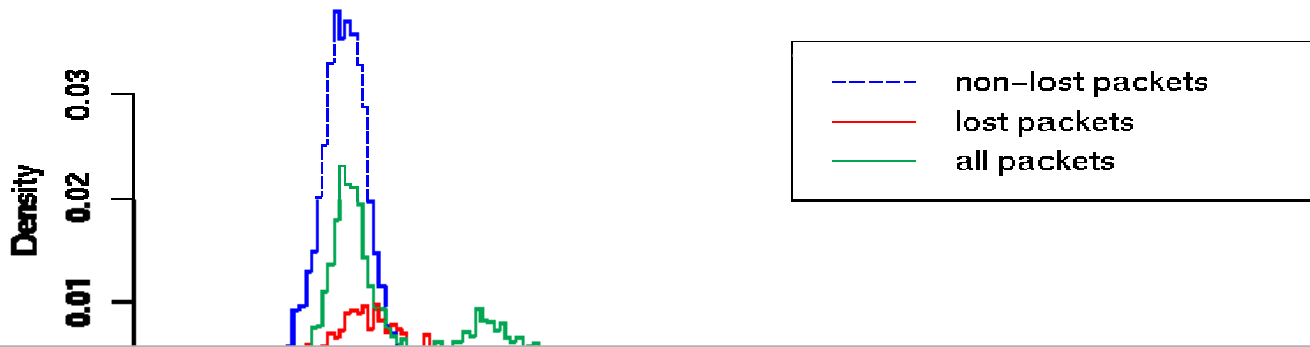


7% connections incurred more than 20% additional average RTTs.

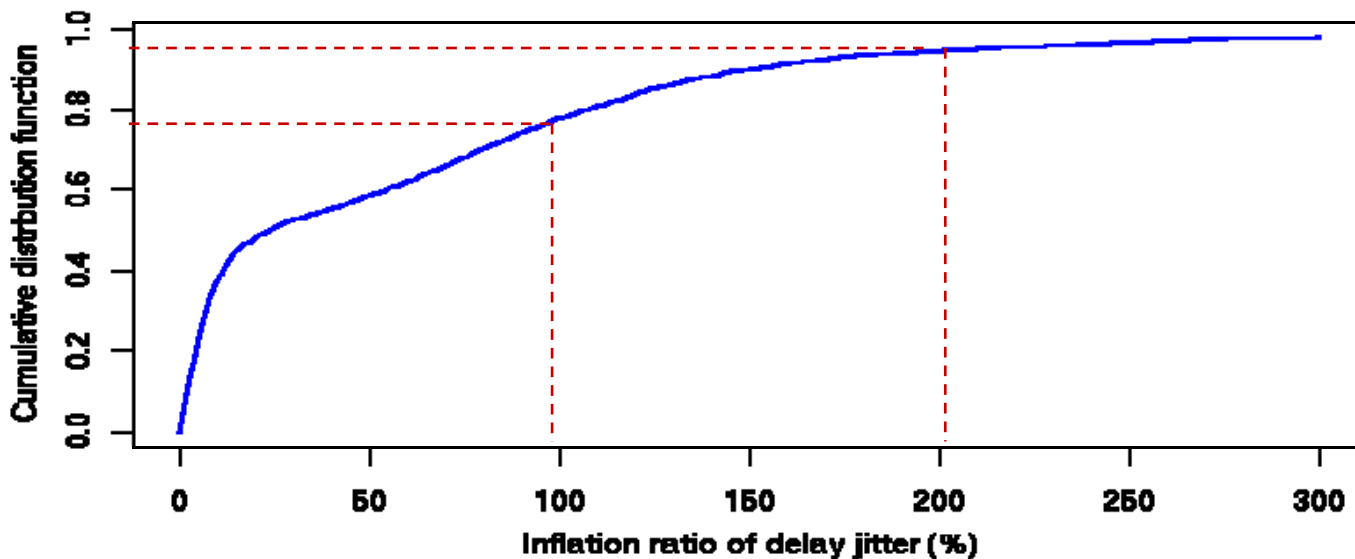


# The Effect of In-Order Delivery on RTT Jitter

(std. dev. of RTTs)



22% connections incurred more than 100% additional RTT jitters.  
6% connections incurred more than 200% additional RTT jitters.



# Unbounded Congestion Window

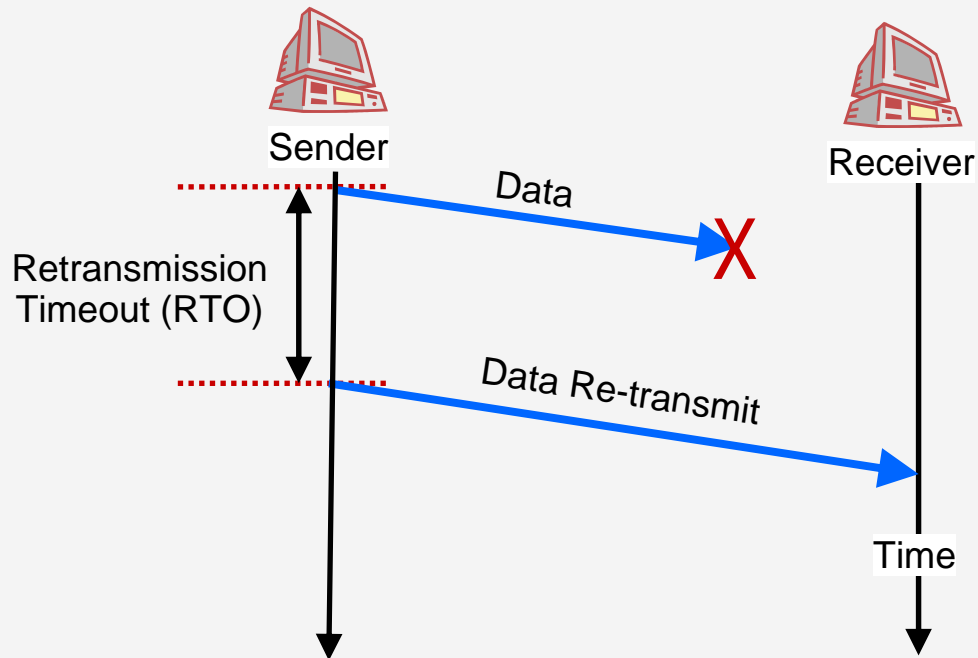
- TCP's congestion control is designed for **network-limited** application
  - Relies on packet loss to adjust its congestion window
- Data generation in online games is often **application-limited**
  - 36% connections never experienced a packet loss
- Congestion window may never be reduced → unreasonably large window
- If the game may occasionally generate a large burst of packets → **unnecessary network congestion**

# Unnecessary Congestion Window Reset

- The “restart after idle periods” policy:  
TCP resets its window to 2 if a connection stops sending data for a short interval (usually  $< 1$  second)
- The policy prevents inappropriate bursts of packets being sent due to an out-of-date window
- But, game packet rate is so low → window is occasionally reset (18% of packets faced a window reset)
- In this case, a series of three commands following a thinking time will be penalized (the 3<sup>rd</sup> packet will be delayed until the first two have been delivered)

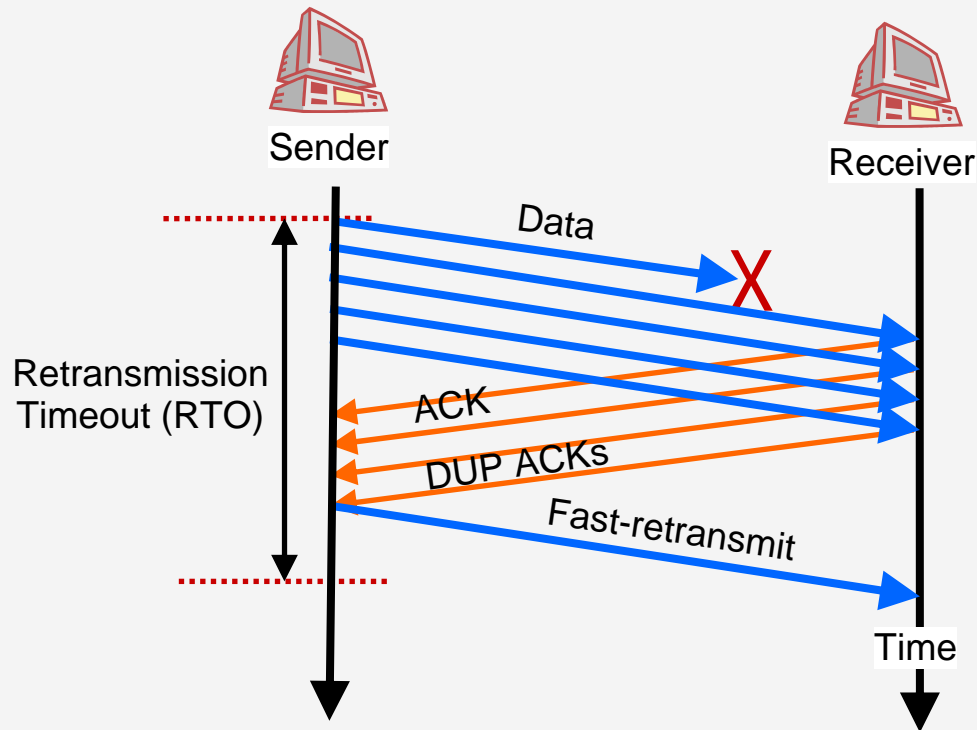
# Loss Recovery -- Retransmission TimeOut

- Originally, TCP only relies on a **timer** to detect packet loss and resend packets



# Loss Recovery -- Fast Retransmit

- TCP Tahoe/Reno improves loss detection with **duplicate acknowledgement** packets



# The Failure of Fast Retransmit

- **Insufficient** duplicate acks

- The game packet rate is too low

- More than **99%** dropped packets were not detected by fast retransmit

- Average latency for **non-dropped** packets was **180 ms**;  
Average latency for **dropped** packets was **700 ms**

- This is why in-order delivery incurred so much overhead in transmission latency and delay jitters

— The definition of “duplicate acks” requires each ack packet not contain data

- The fast retransmit may not be triggered even sufficient dup acks are generated

# Performance Impact on User Experience

- Based on the model describing the relationship between **player departure rate** and RTT, RTT jitter, and packet loss rate [INFOCOM 2006]

$$\log(\text{departure rate}) \propto 19.2 \times rtt + 4.54 \times rtt.jitter + 0.7 \times \log(closs) + 0.45 \times \log(sloss)$$

- RTT jitter is raised from 77ms to 124ms due to in-order delivery  
 $\exp((0.124 - 0.077) \times 4.54) \approx 1.24$
- Assuming the additional RTT jitters due to TCP in-order delivery can be avoided, the **player departure rate** is expected to decrease by **20%** ( $1/1.24 \approx 0.8$ )
- This corresponds to an increase of average game playing time from **100 minutes** to **135 minutes**

# Protocol Design Guidelines (1)

- Supporting both **reliable** and **unreliable** delivery
  - Some packets can be **safely discarded** without affecting gaming experience
  - E.g., A gesture of a character faraway from the notified character may need not to be reliably transmitted
- Supporting both **in-order** and **out-of-order** delivery
  - Only ordering packets whenever absolutely necessary
  - E.g., Ordering is irrelevant for repeated attack actions (fight the same enemy with the same weapon)

# Protocol Design Guidelines (2)

## ■ **Accumulative** delivery

- A new messages can override all the previous ones
- Missing some packets in a series of accumulative commands does not matter (unless the last one in series is also dropped)
- E.g., position updates

## ■ Multiple streams

- Make messages as **independent** as possible
- Messages need to be ordered only when they are in the same stream
- E.g., chat messages are independent of game play commands

# Protocol Design Guidelines (3)

- Coordinated congestion control
  - Tens of thousands of flows are not unusual
  - Difficult for all the flows to achieve an efficient bandwidth sharing by competition
  - Make congestion control in a **coordinated, rather than competitive**, manner
  - E.g., avoid dispatching game message **synchronously** for all game clients → alleviate traffic burstiness → reduce overall packet loss rate

# Summary

- TCP is designed for uni-directional bulk transfer, but game traffic has many unique features:
  - Tiny packets
  - Low packet rate
  - Application-limited traffic generation
  - Bi-directional traffic
- TCP is unwieldy and inappropriate for MMORPGs
- The degraded performance did impact users' willingness to continue playing a game
- A number of design guidelines have been proposed

**Thank You!**



Kuan-Ta Chen