# The Delay-Friendliness of TCP

**Eli Brosh**, Salman Baset,
Dan Rubenstein, Henning Schulzrinne

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

SIGMETRICS 2008

# Motivation

- TCP is not designed for **real-time** applications
  - Provides reliable, in-order delivery: not needed by real-time applications
  - Delay is not primary concern
- Motivated design of unreliable protocol alternatives
  - RTP, DCCP, TFRC, and others

# Motivation

- Despite its shortcomings, TCP is widely used by commercial real-time systems
  - Skype and Windows Media Services support TCP
    - Majority of streaming traffic uses TCP [SMZ04,GCXZ05]
  - UDP packets are blocked by many NATs and firewalls
  - A mature, standardized, widely-used protocol

We answer the question of **when** and **why** TCP works for real-time transmission

# Contributions

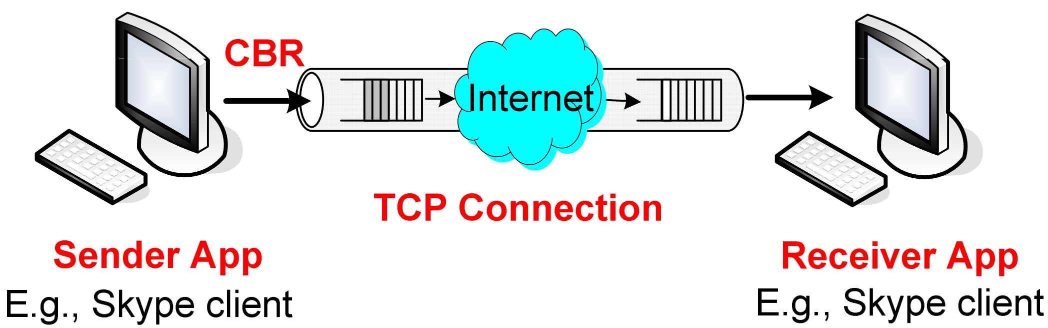
- A discrete-time Markov model for the delay distribution of TCP
- Quantify the feasible region of TCP for VoIP and live video streaming
  - **Insight:** packet sizes play an important role in determining feasible region
- Provide application-level heuristics for reducing delay
  - Also, socket and system-level delay-friendly TCP settings

# Related Work

- Extensive literature on TCP modeling and analysis
  - Model the performance of file transfers [PFTK98,CSA00,..] and video streaming [WKST04,KA06] from the standpoint of **throughput** not delay
- Kernel-level enhancements for reducing TCP delays
  - Adapting TCP send buffer size [GKLW02]
  - Eliminating reliability [MLWL05, MB00]
- Application-level schemes for reducing TCP delays
  - Focuses on **interactive apps** such as telnet and games [GH06, MK07]

# Application Setting

- A media application with a Constant Bit Rate (CBR) source
  - CBR is dominant encoding of media flows [LCKN05]

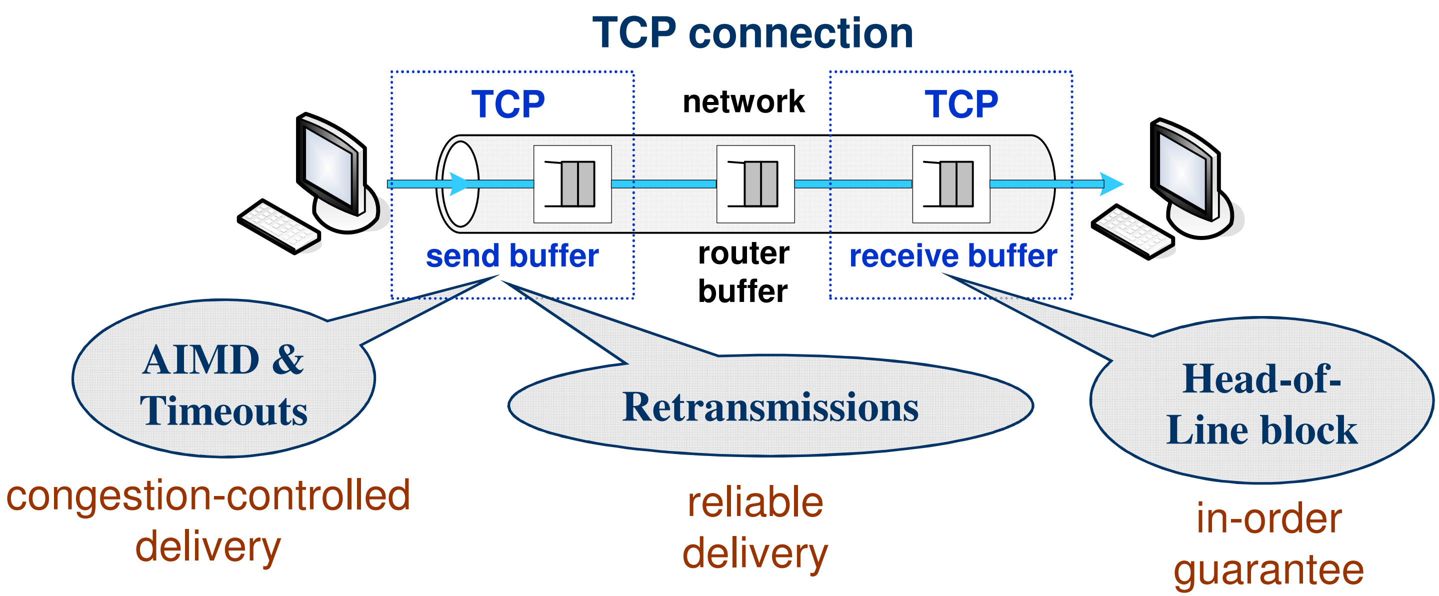



**TCP delay:** the time to send a packet through a TCP connection

- TCP delay distribution determines **late** packets
- Late packets determine the perceived media quality

# TCP Delay Components

- TCP delay = network delay + protocol-induced delay
  - TCP's reaction to network throughput variations



# TCP Delay Model

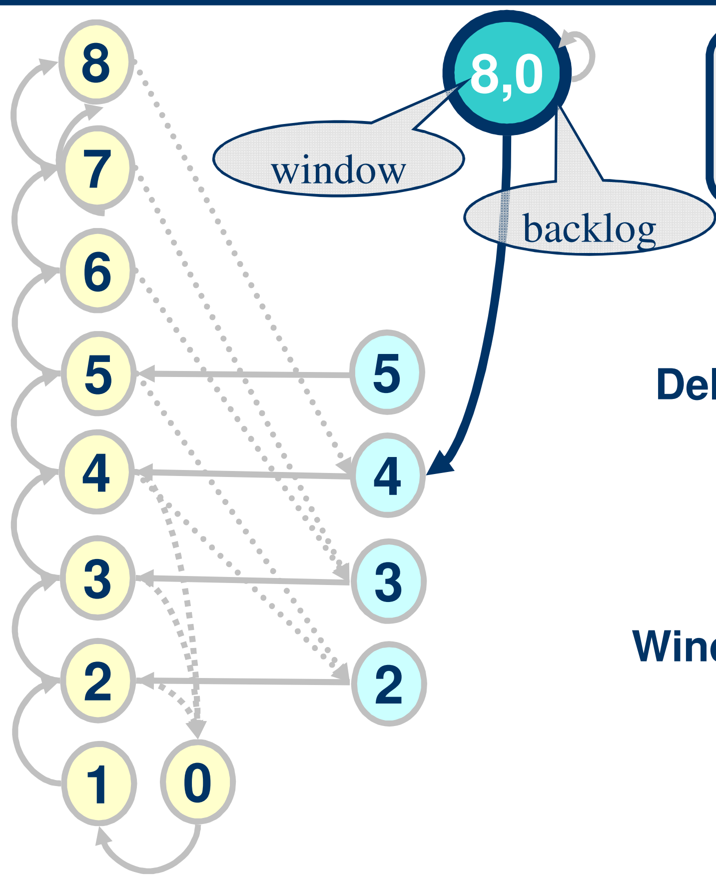
- Extends TCP throughput model of Wierman *et al.* 2003 by capturing
  - **Backlog**: the send buffer occupancy
  - TCP's behavior in **application-limited** periods
- Markov model: states are associated with packet transmissions
  - Each packet transmission is associated with delay
  - Transitions: successful transmission and loss occurrence

$$\text{Packet delay} = \underbrace{\frac{\text{backlog size}}{\text{source's rate}} + \text{head-of-line blocking}}_{\text{protocol-induced delay}} + \text{network delay}$$

congestion control

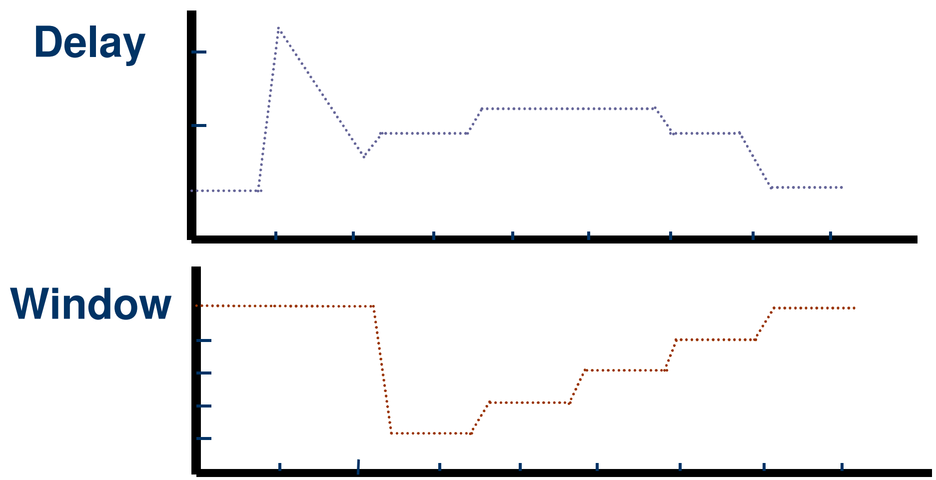
loss recovery latency

# TCP Delay Example

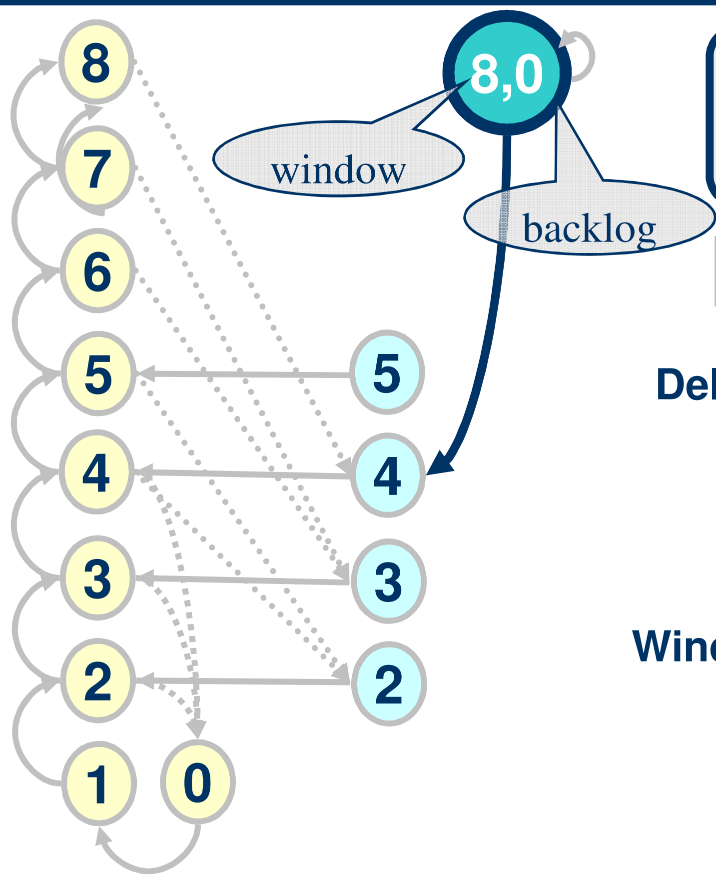



**Application-limited state**

TCP's throughput limited by application's rate



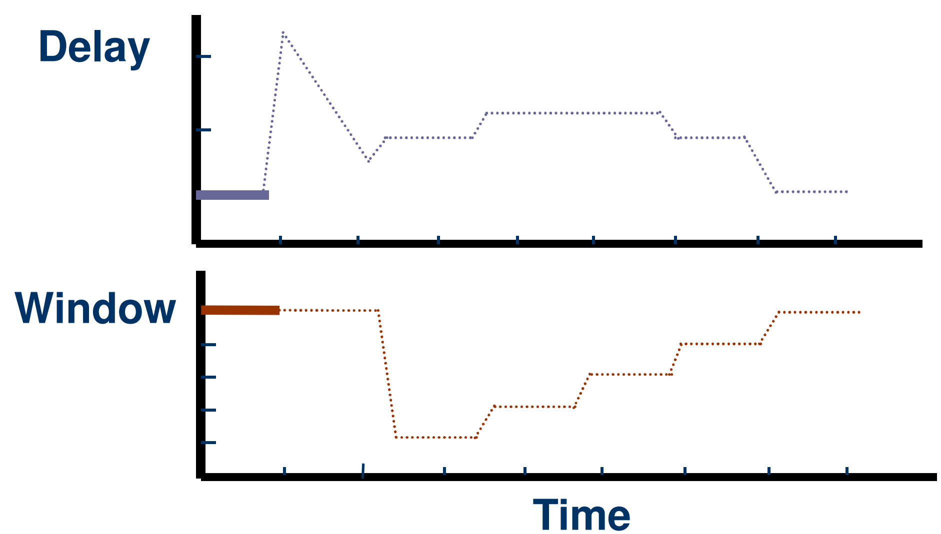
# TCP Delay Example



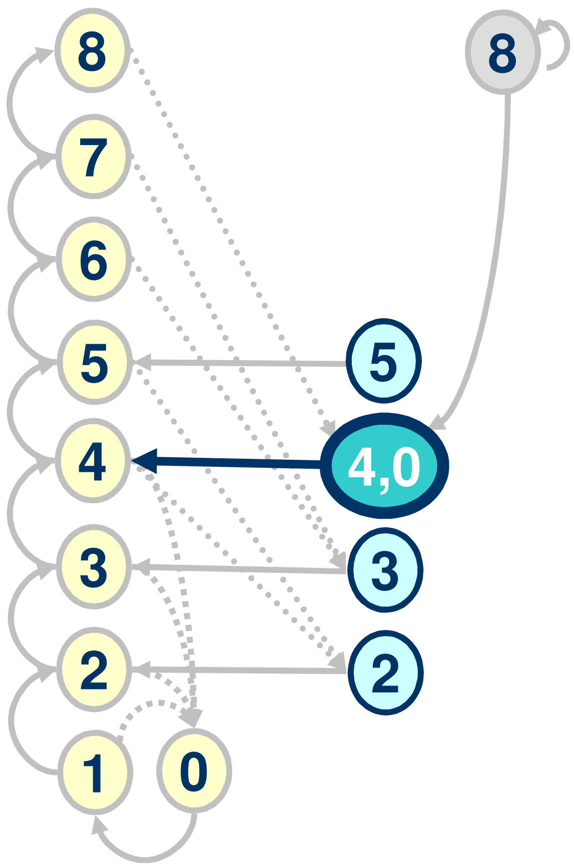

**Application-limited state**

TCP's throughput limited by source's rate

packet delay = network delay

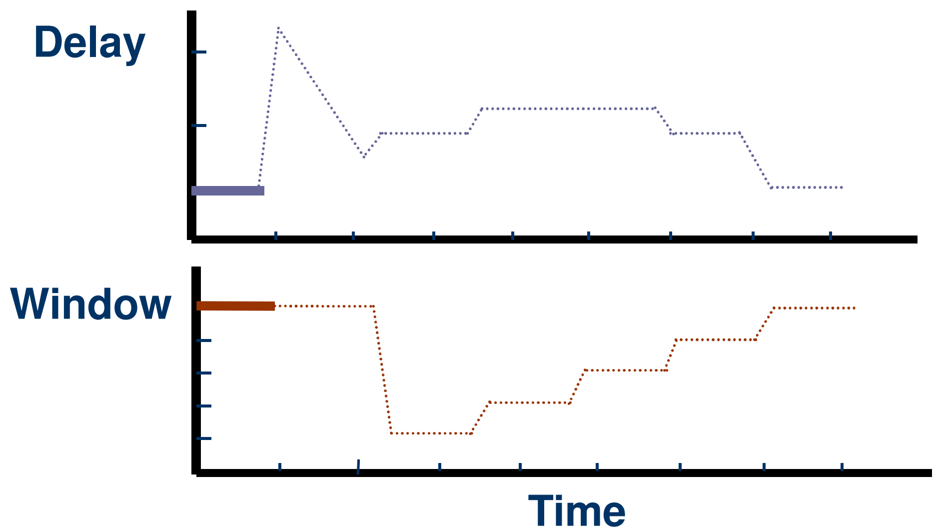


# TCP Delay Example

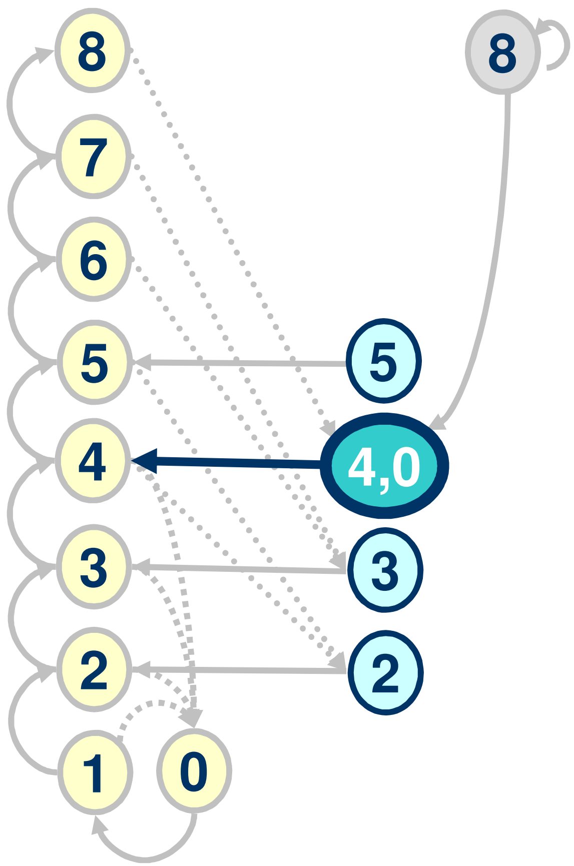



**Fast recovery state**

Retransmission to recover the lost packet



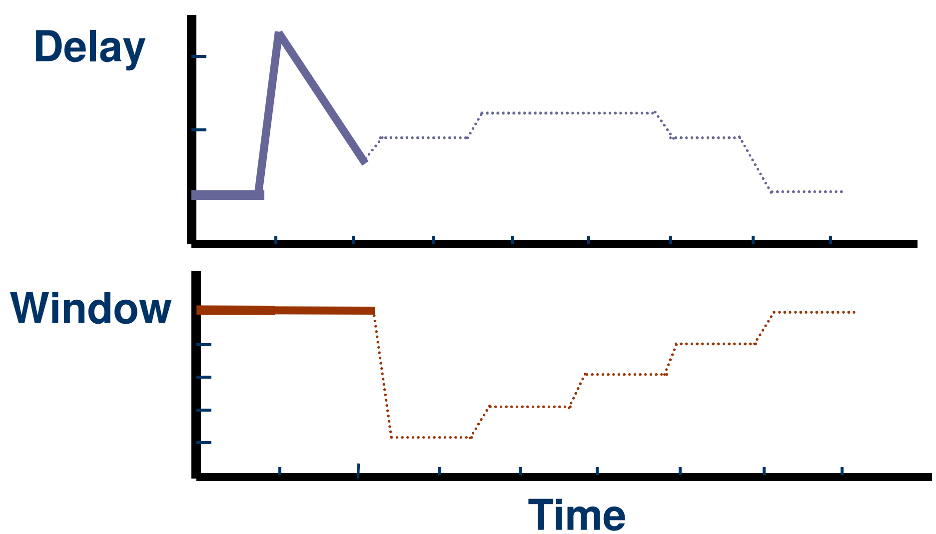
# TCP Delay Example



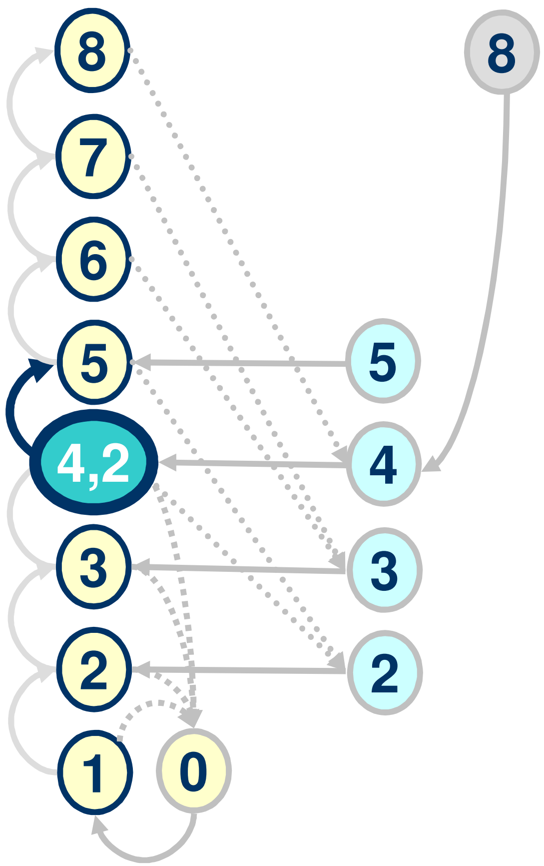

**Fast recovery state**

Retransmission to recover the lost packet

Protocol delay = head-of-line blocking

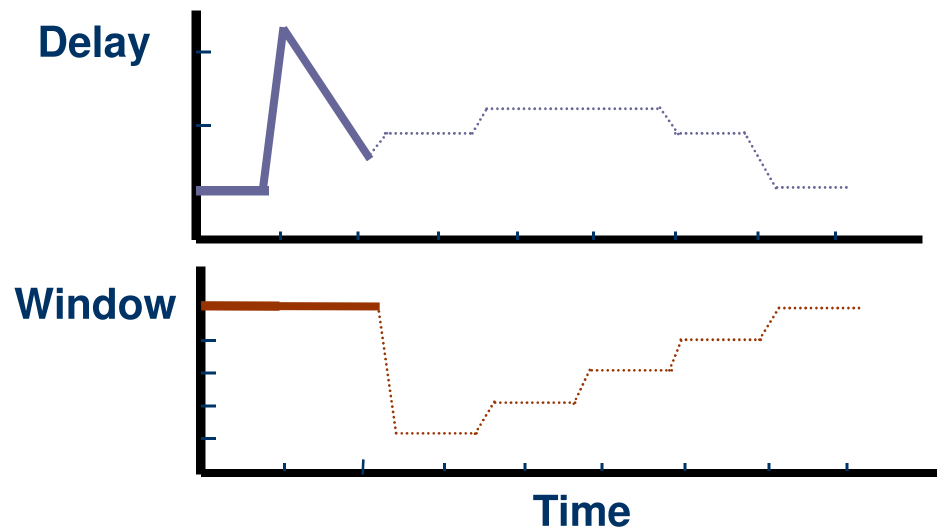


# TCP Delay Example

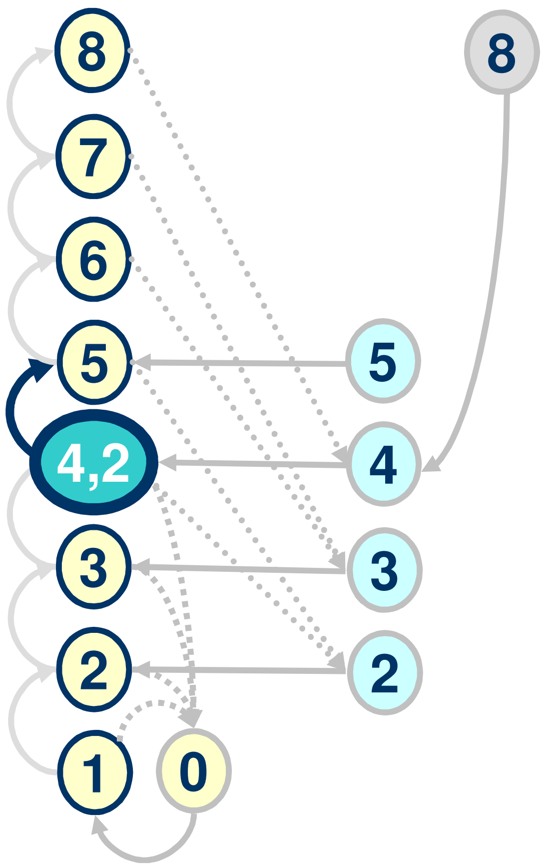



**Congestion avoidance state**

TCP's throughput limited by the network
Congestion window is increased every RTT

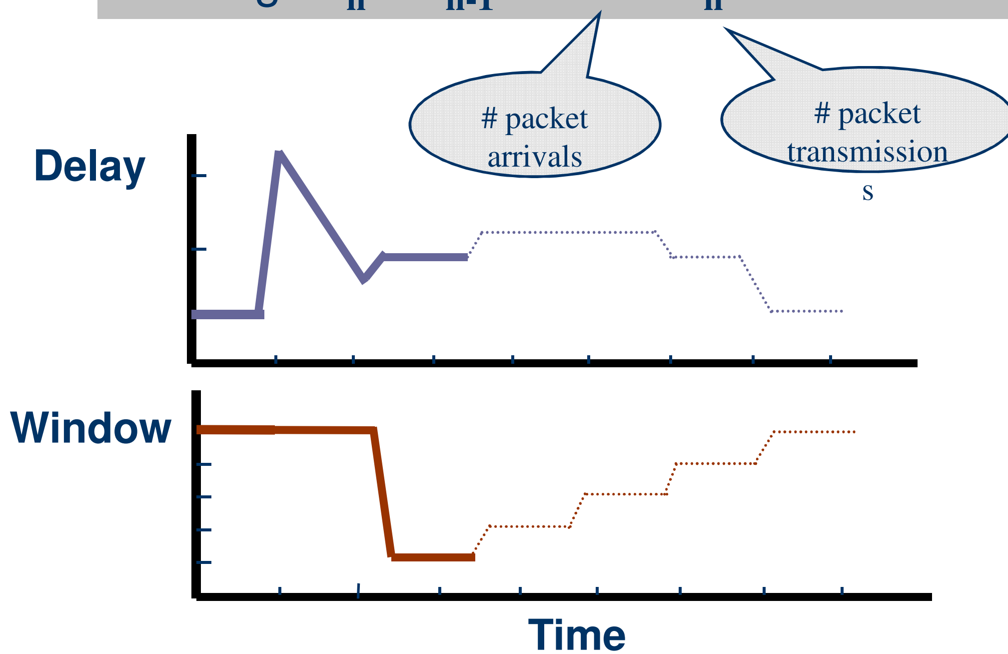


# TCP Delay Example

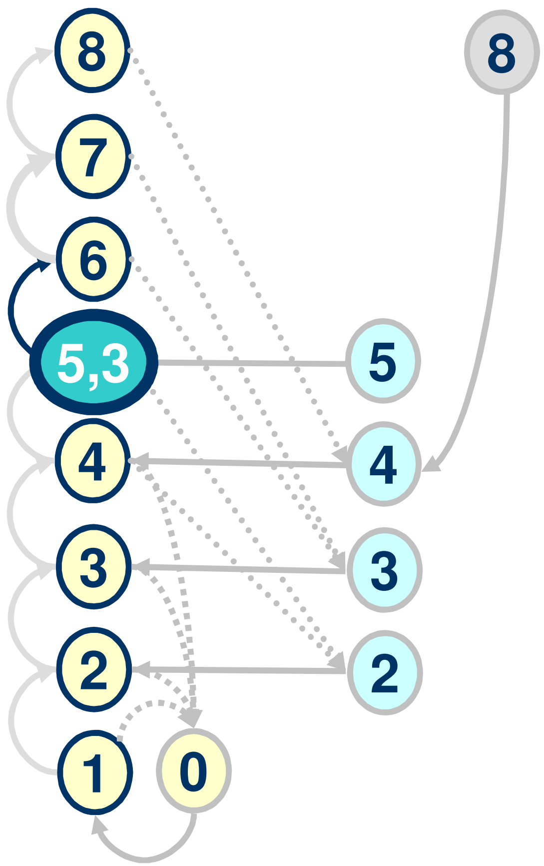



Protocol delay = backlog/source's rate

backlog: $B_n = B_{n-1} + A - W_n$



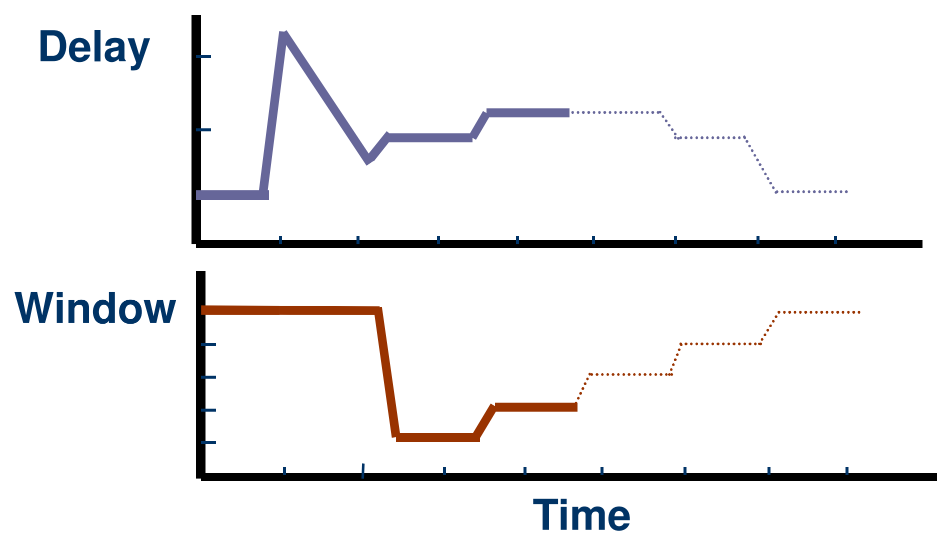
# TCP Delay Example



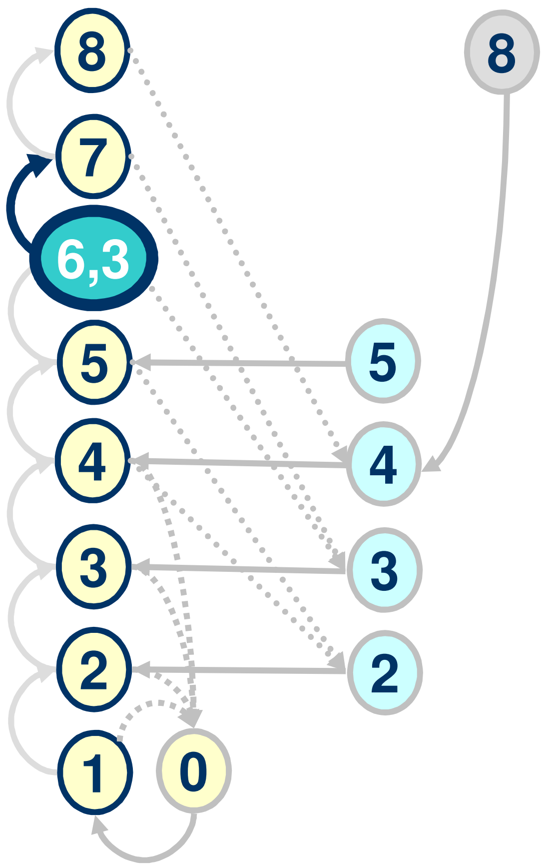

**Congestion avoidance state**

Congestion window is increased every RTT

Protocol delay = backlog/source's rate



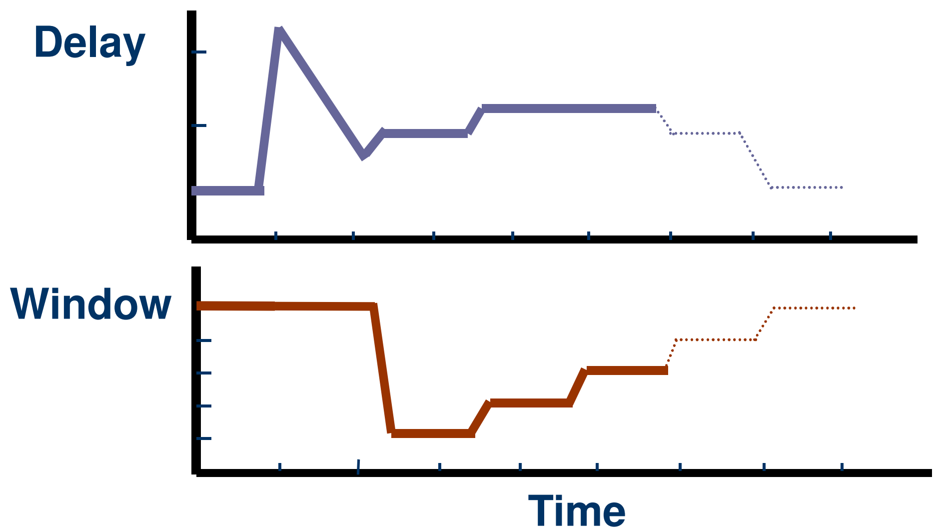
# TCP Delay Example



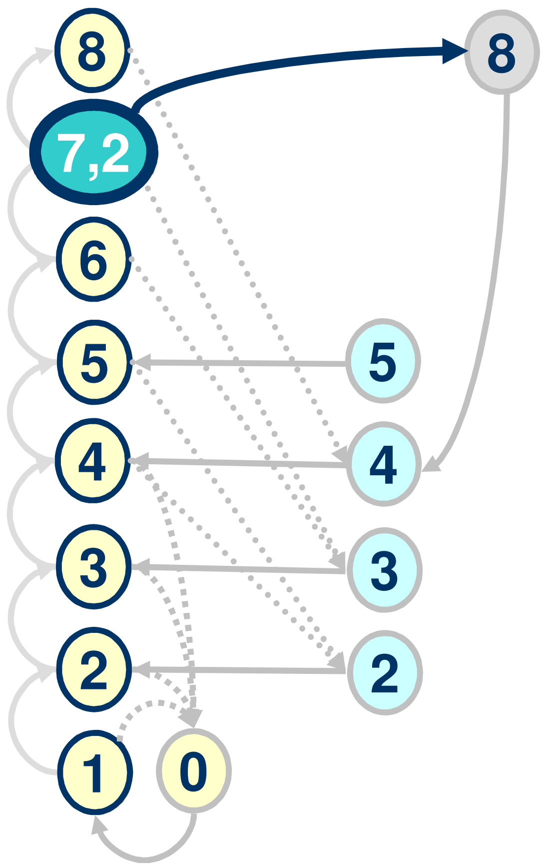

**Congestion avoidance state**

Congestion window is increased every RTT

Protocol delay = backlog/source's rate



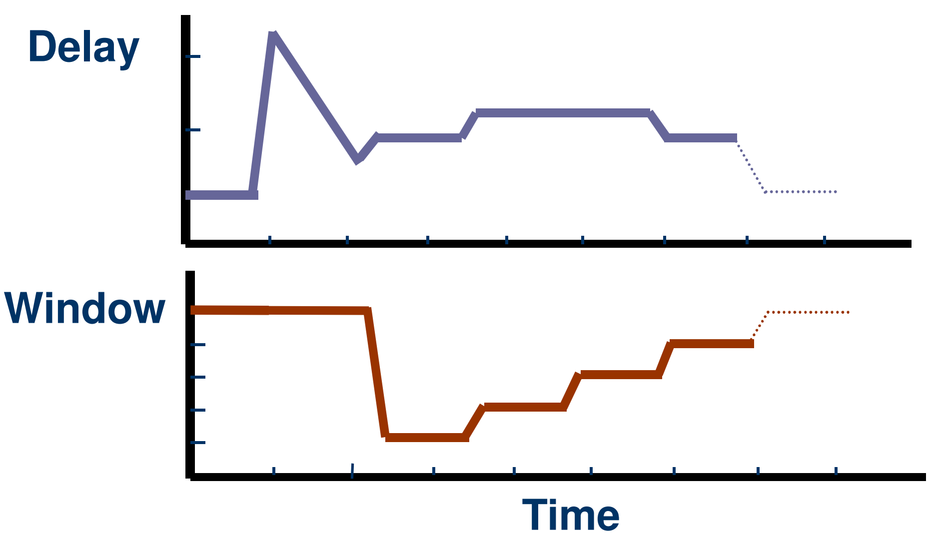
# TCP Delay Example



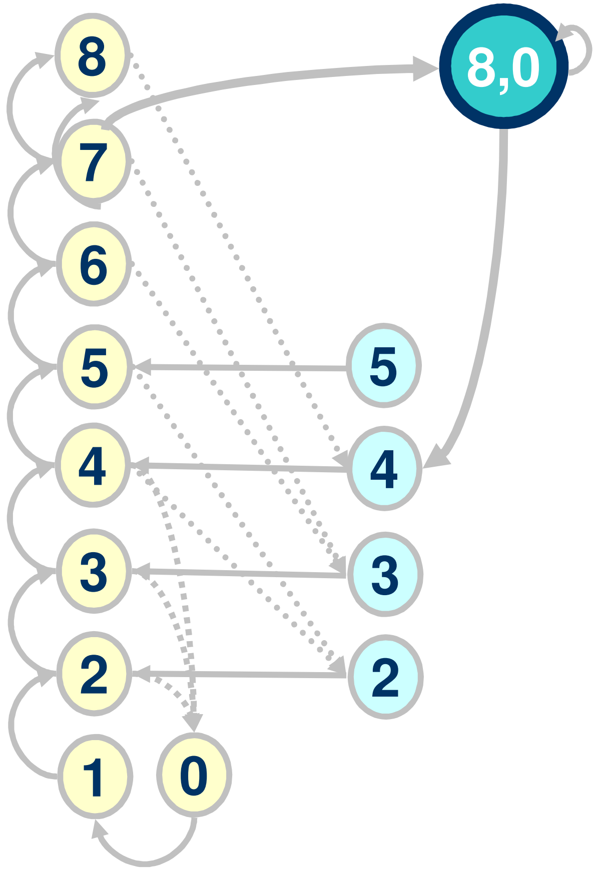

**Congestion avoidance state**

Congestion window is increased every RTT

Protocol delay = backlog/source's rate

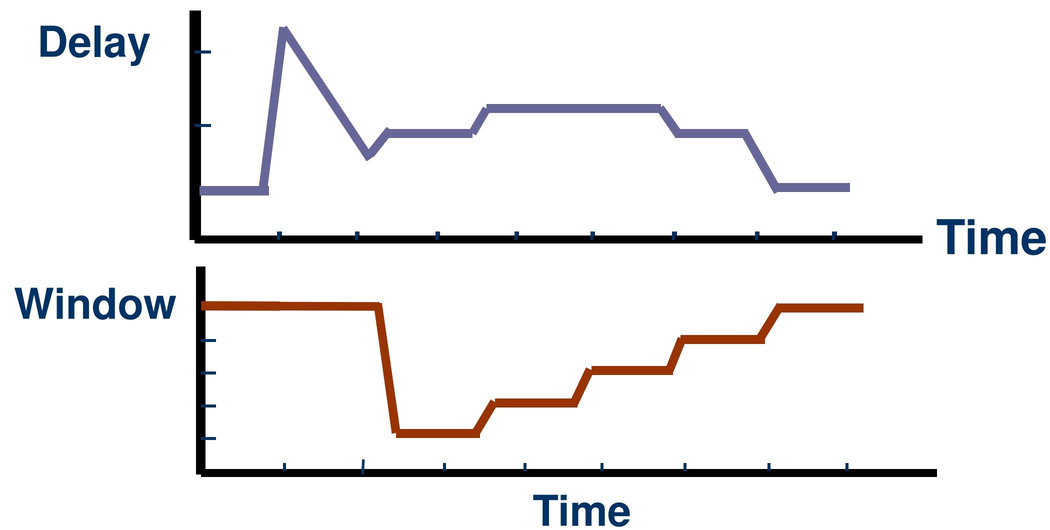


# TCP Delay Example



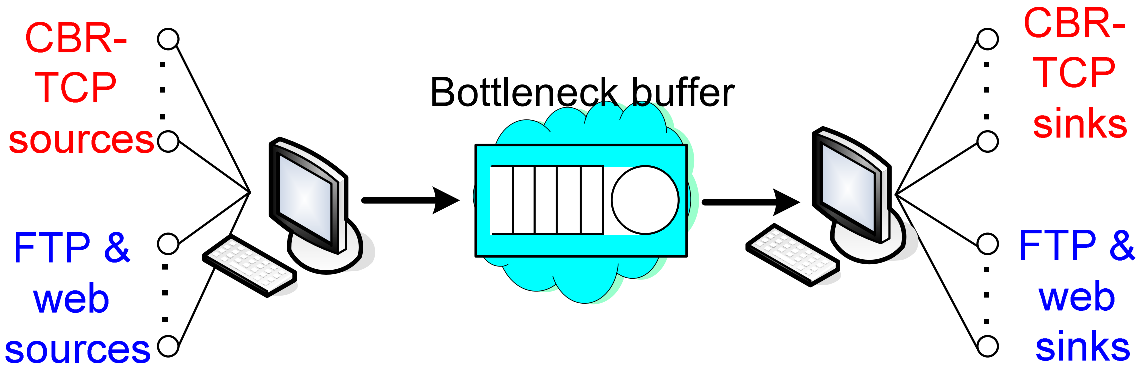

Application-limited state

Packet delay = network delay

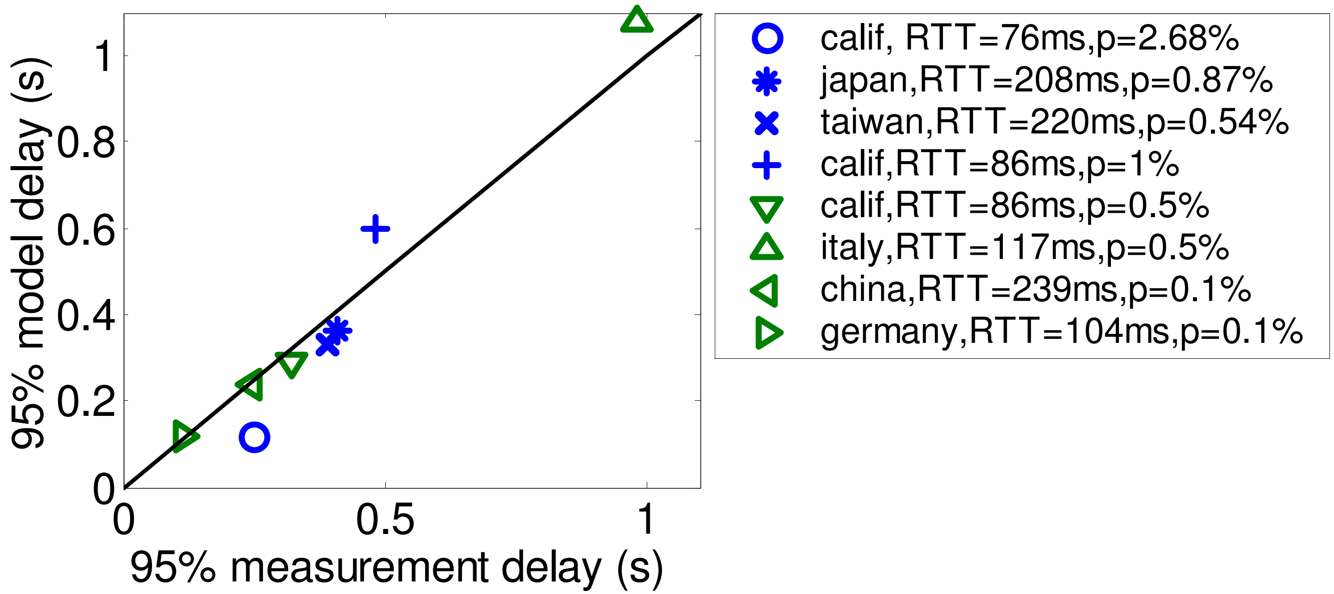


# Model Validation Settings

- **Controlled environment**
  - Drop-tail bottleneck router
    - Loss rates: 0.1% -10%, RTTs: 20-300ms
- **Internet experiments**
  - Planet Lab and residential machines



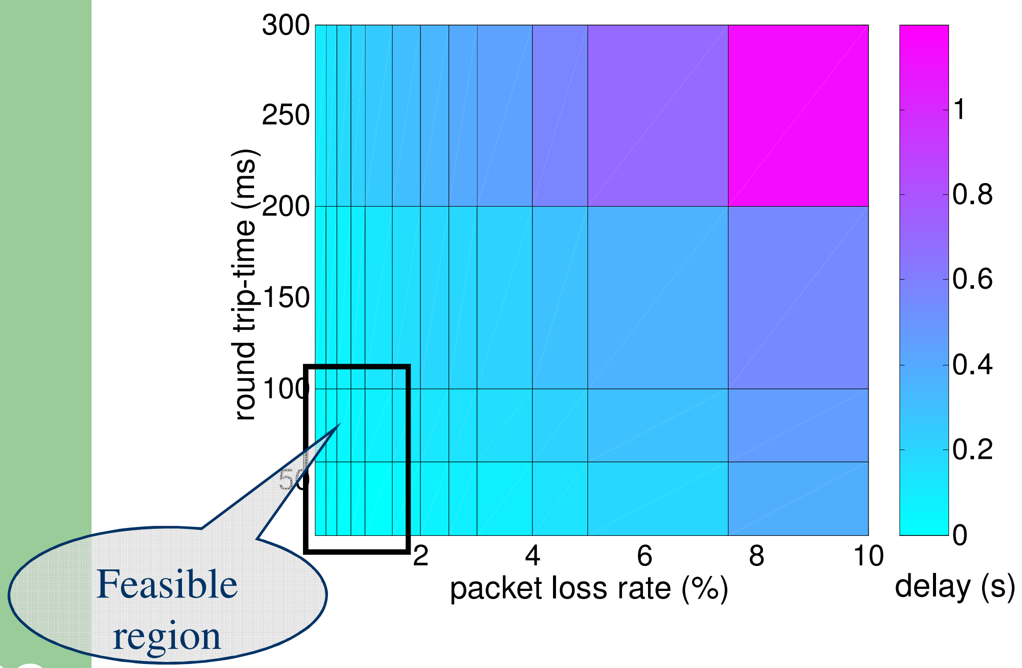
# Validation Using Internet Experiments




- Good match for Internet experiments and controlled environment
  - Prediction error ≤ 25%

# When Does TCP Work?

- Voice over IP (VoIP) application



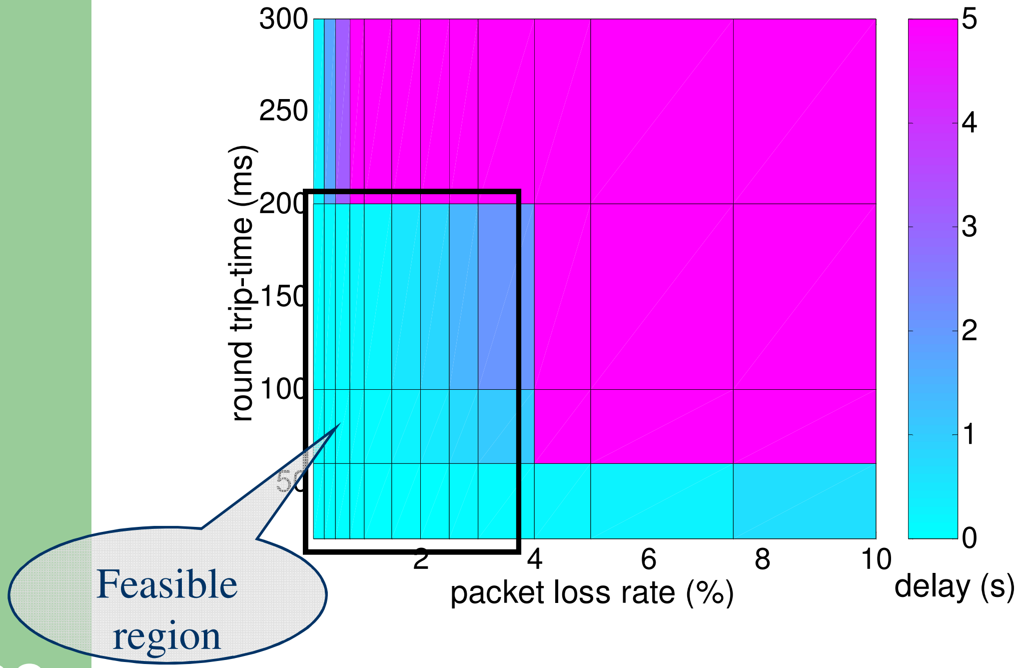

| | |
|---|---|
| Packet size | 160 bytes |
| Bit rate | 64 kb/s |
| **Delay tolerance** | **< 200ms** |

Feasible region:

**RTT≤100ms, Loss≤2%**

# When Does TCP Work?

- Live video streaming application




| | |
|---|---|
| Packet size | 1400 bytes |
| Bit rate | 573 kb/s |
| **Delay tolerance** | **≈ 5s** |

Feasible region:

**RTT≤200ms, Loss≤3%**

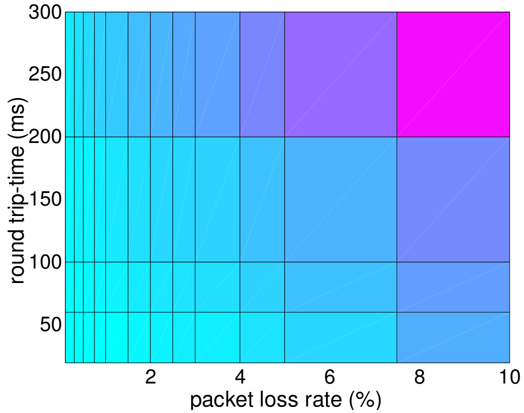
# Feasible Region Comparison

- Voice region is larger than video region.
  why?
  - Voice flows use **lower** bit-rates
  - Voice flows gain from TCP's **bias** in favor of flows with **small packets**

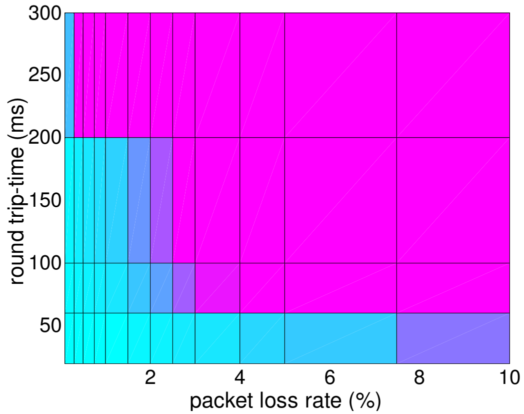
intuitive

model

VoIP




video streaming



# The Effect of Packet Size on Delay (1)

- Most TCP implementations use packet-based congestion control (ACK-counting)
  - TCP regulates rate based on number of sent packets
  - Magnitude of rate fluctuations (in bytes) is smaller for flows with small packets

Bias in favor of flows with small packets (e.g., VoIP)

# The Effect of Packet Size on Delay (2)

VoIP delay dominated by loss recovery latency

protocol-induced delay = congestion control delay + loss recovery latency

small due to TCP's bias

≈ RTT

Video streaming delay dominated by congestion control

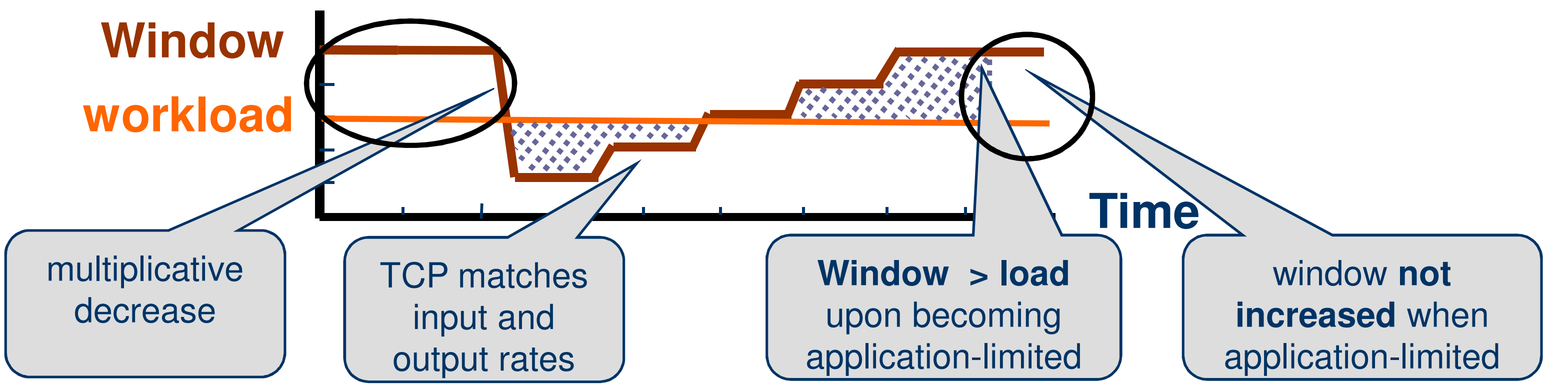
protocol-induced delay = congestion control delay + loss recovery latency

# Why does TCP work well?

(1) TCP's bias in favor of flows with small packets

(2) Congestion window overestimates application's load

- Loss recovery efficiency depends on window size




Low likelihood of timeouts → low delay

# Application-level Heuristics for Reducing TCP Delay

- Packet splitting
  - Exploits TCP’s bias in favor of small packet flows
  - Masquerades a flow with large packets as one with small packets
- Parallel Connections
  - Stripe load across multiple connections
- Effective for video streaming but ineffective for VoIP
  - Parallel connections outperforms packet splitting in terms of performance and fairness

# Conclusions

- Demonstrate why real-time traffic over TCP is feasible in today’s Internet
- Feasible region of TCP for VoIP is larger than that for video streaming
  - As long as packet-based congestion control is enforced
- Parallel connections has better performance and is more fair than packet splitting

Thanks you!

Questions?