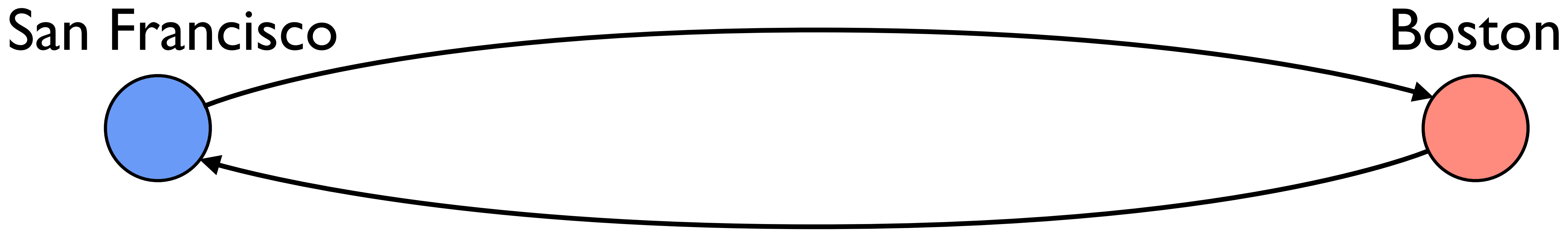
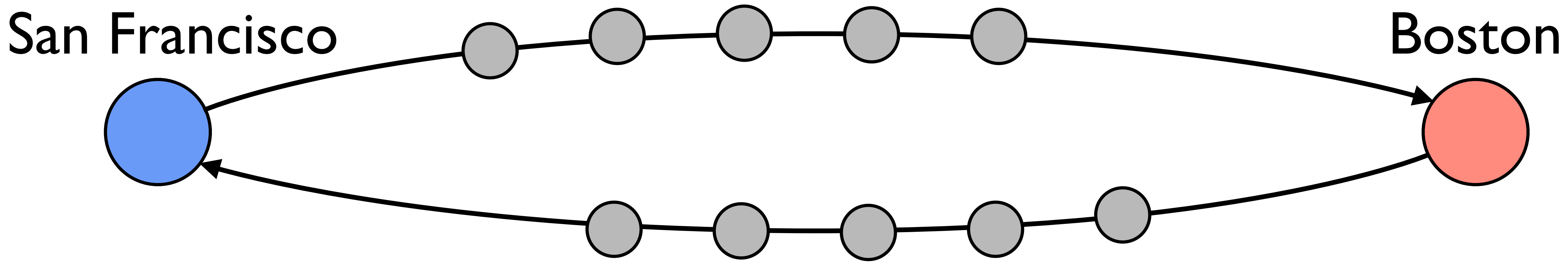
Congestion Control

AIMD, queueing, TCP variants

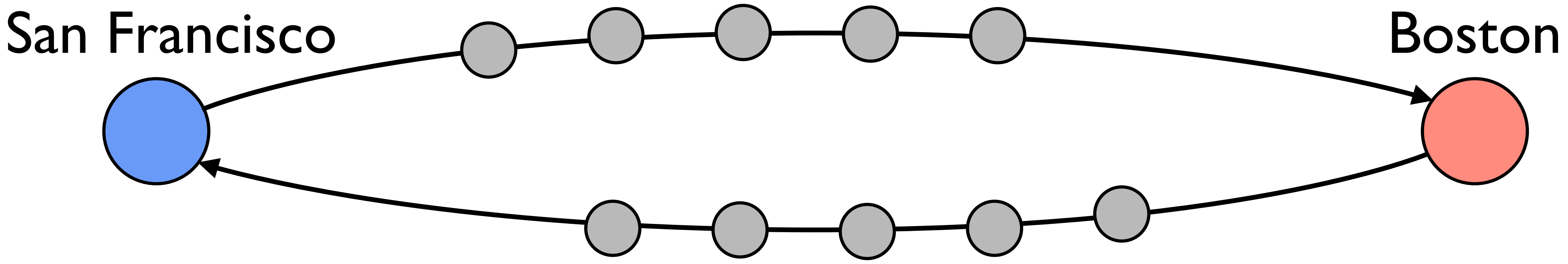
Congestion Control Motivation



Congestion Control Motivation



Congestion Control Motivation



Congestion control: limit outstanding data so it does not congest network, improves overall performance

TCP and AIMD

- TCP uses additive-increase, multiplicative decrease (AIMD)
  - Maintains a *congestion window*, an estimate of how many unacknowledged segments can be sent
  - Increases the congestion window by one segment every RTT
  - Halves the congestion window (or more) on detecting a loss
- A bit of history on why (the Internet collapsed)
- Explanation of AIMD dynamics with packet switching
- Explanation of <u>how</u> TCP achieves and implements AIMD

TCP History

- 1974: 3-way handshake
- 1978: TCP and IP split into TCP/IP
- 1983: January 1, ARPAnet switches to TCP/IP
- 1986: Internet begins to suffer congestion collapse
- 1987-8: Van Jacobson fixes TCP, publishes seminal TCP paper (Tahoe)

TCP Pre-Tahoe

(very similar to your your lab 2)

- Endpoint has the flow control window size
- On connection establishment, send a full window of packets
- Start a retransmit timer for each packet
- Problem: what if window is much larger than what network can support?
  - "In October of '86, the Internet had the first of what became a series of 'congestion collapses'. During this period, the data throughput from LBL to UC Berkeley (sites separated by 400 yards and two IMP hops) dropped from 32 Kbps to 40 bps."

TCP in 1986

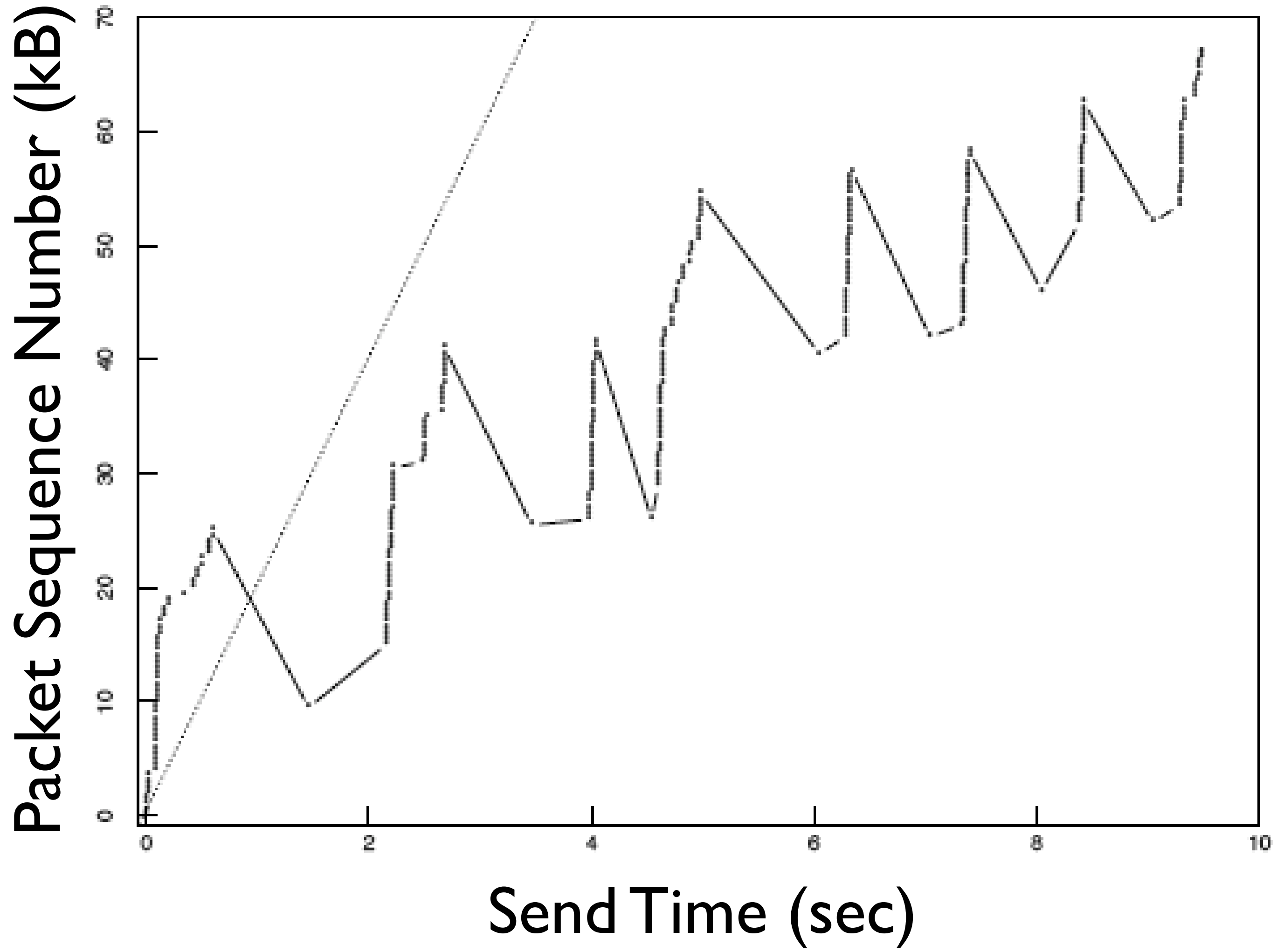
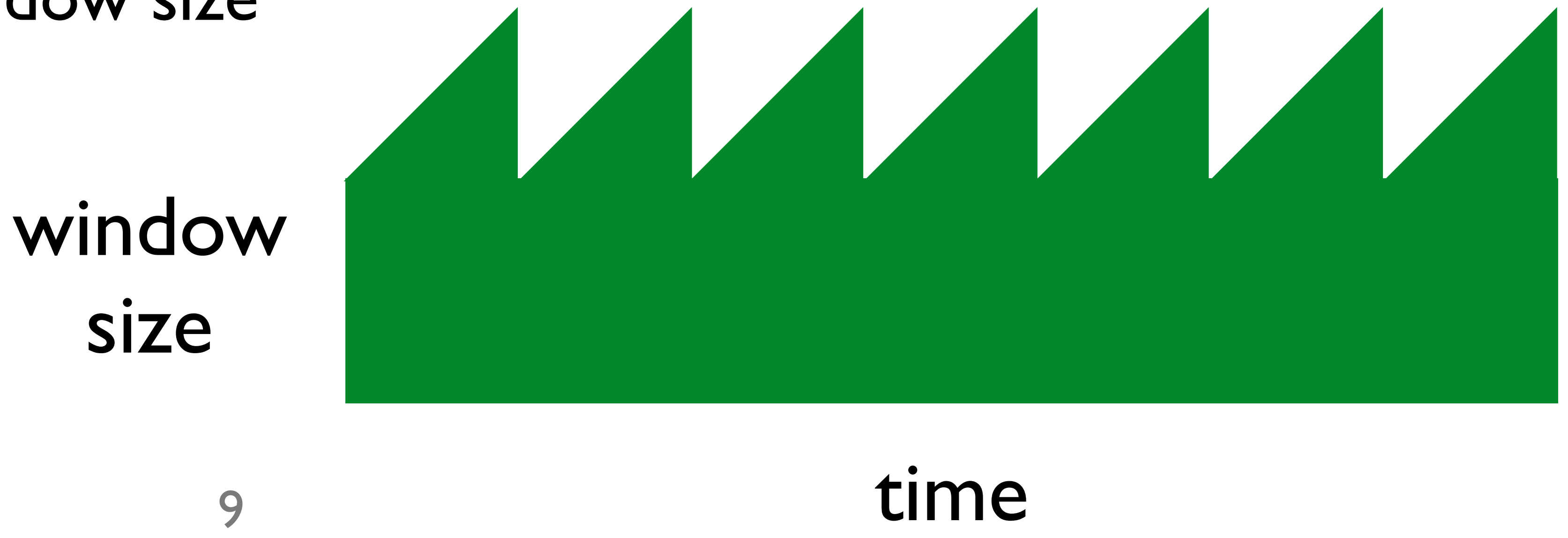


Figure from "Congestion Avoidance and Control", Van Jacobson and Karels. Used with permission.

AIMD

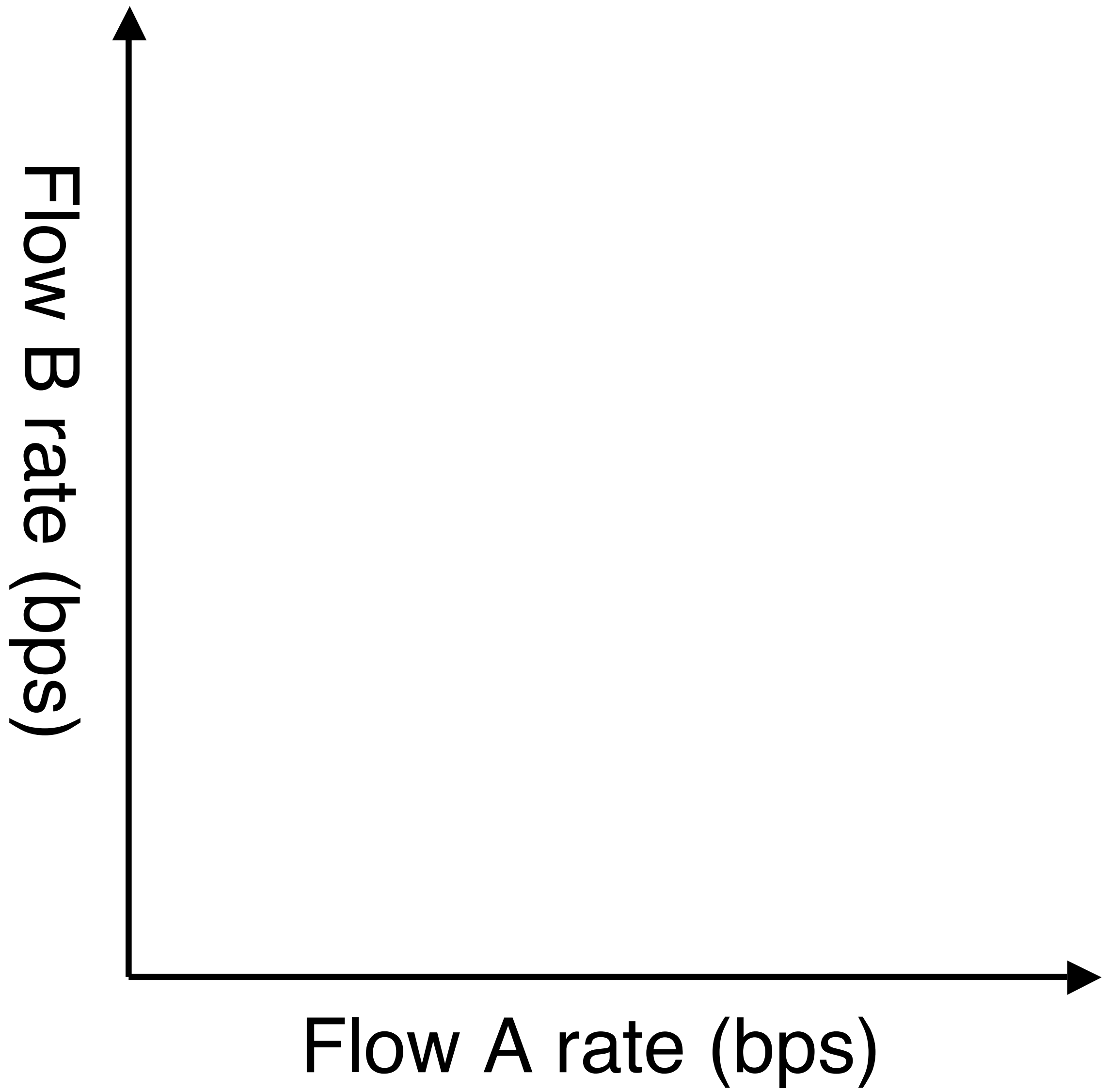
- Additive Increase, Multiplicative Decrease
- On each RTT, increase send window by 1 maximum-sized segment (MSS)
- If a packet is lost, halve the congestion window
- Results in a "sawtooth" window size



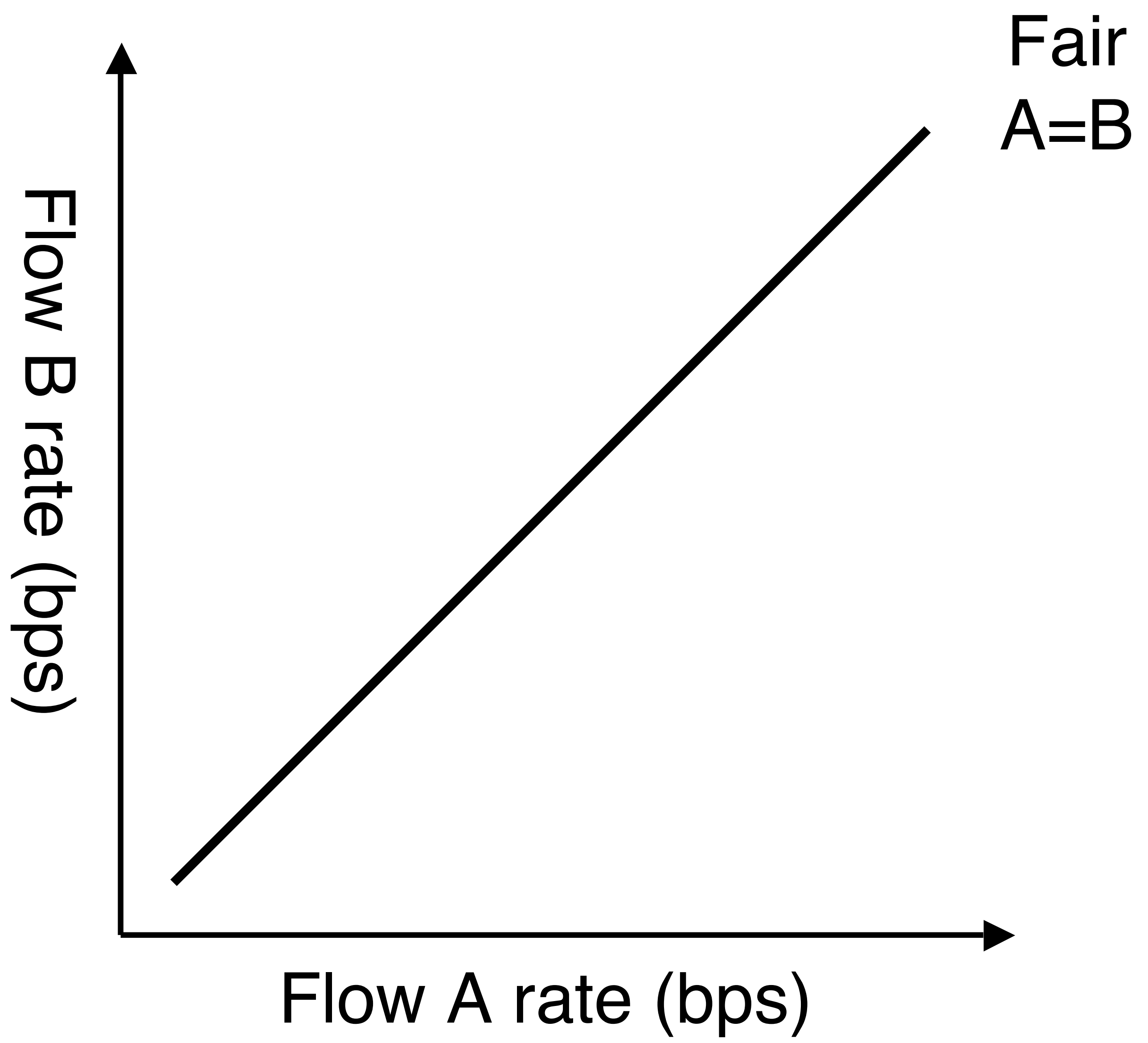
Congestion Control

- Service Provider: maximize link utilization
- User: I get my fair share
- Want network to converge to a state where everyone gets 1/N
- Avoid congestion collapse

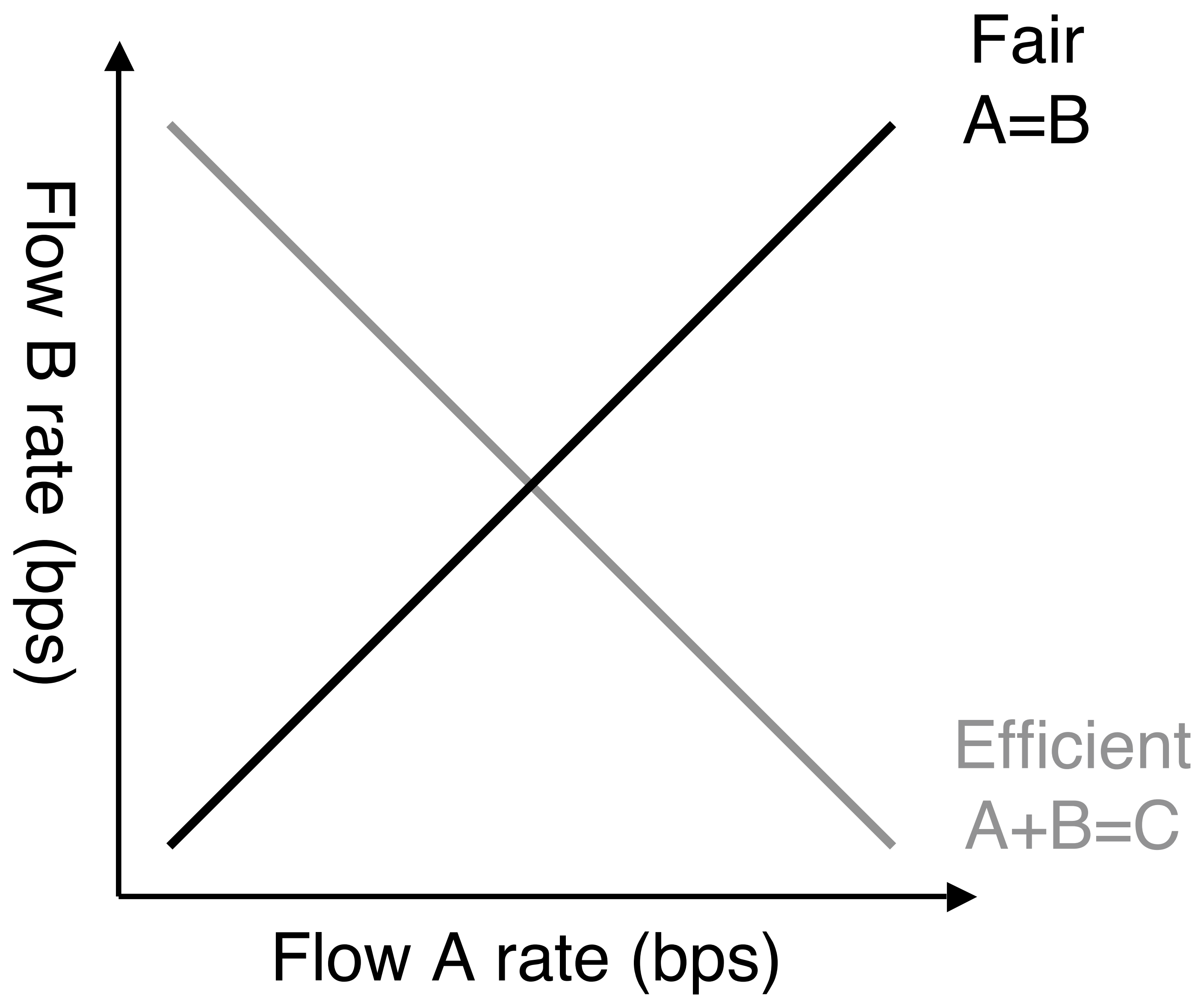
Chiu Jain Plot



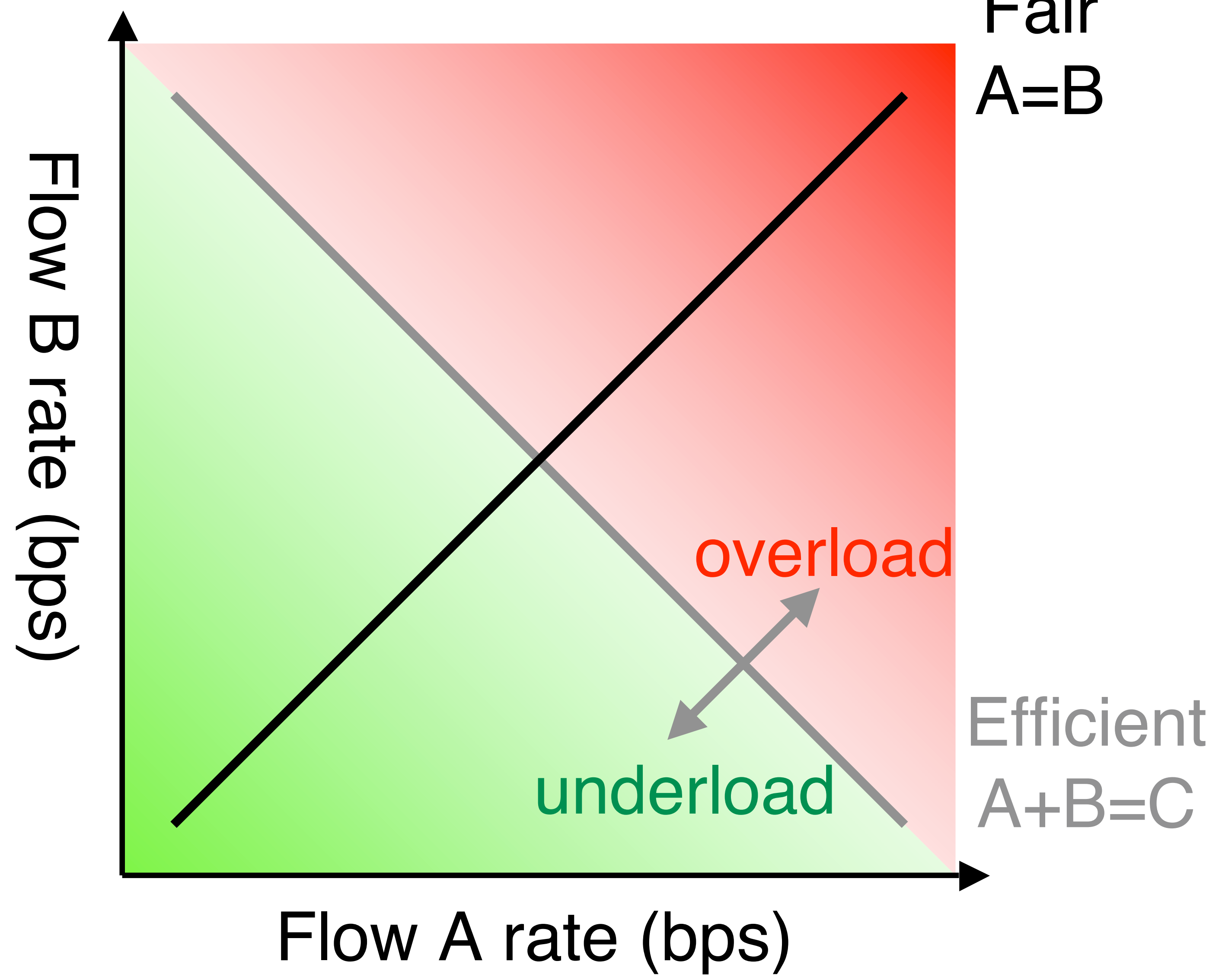
Chiu Jain Plot



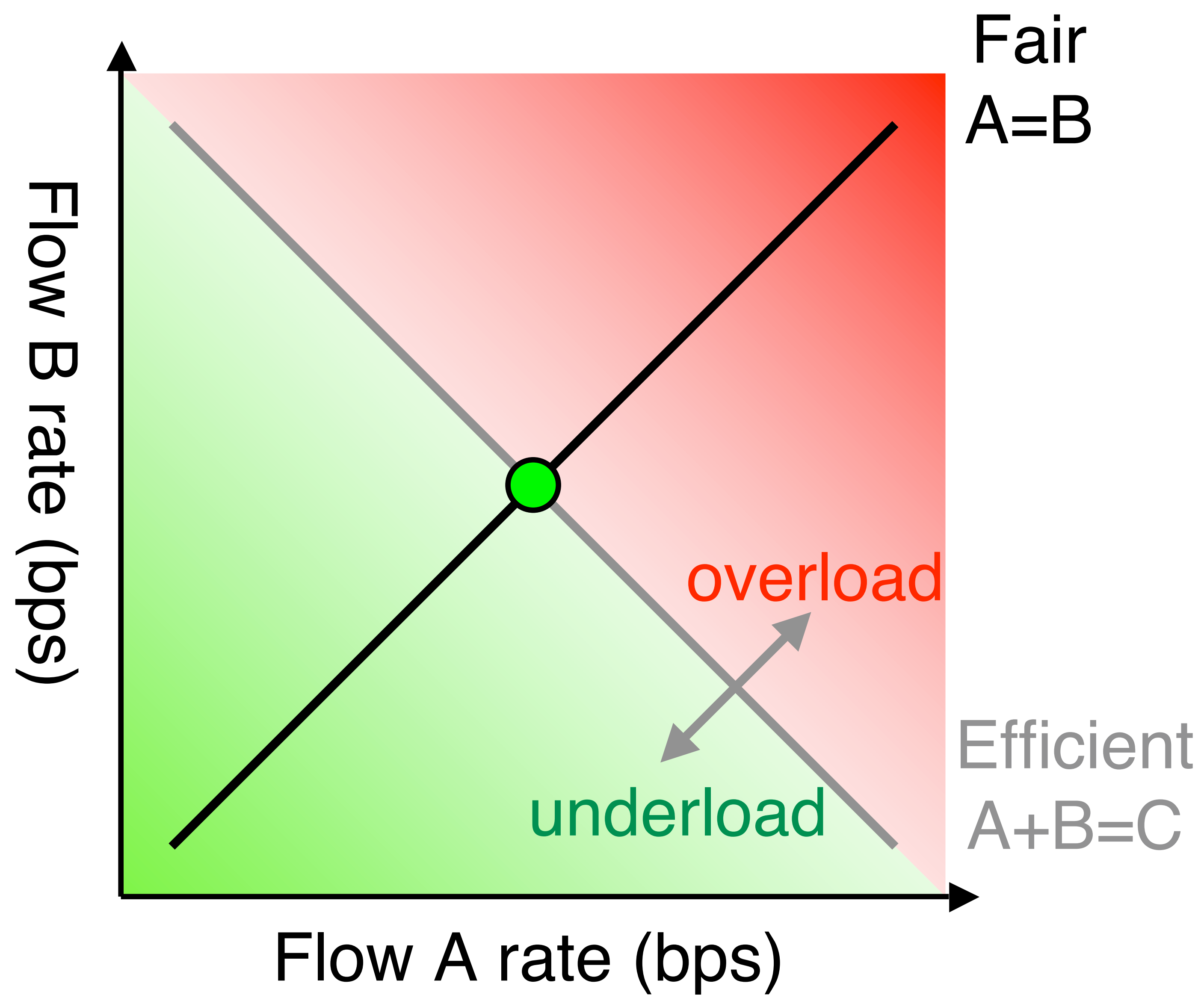
Chiu Jain Plot



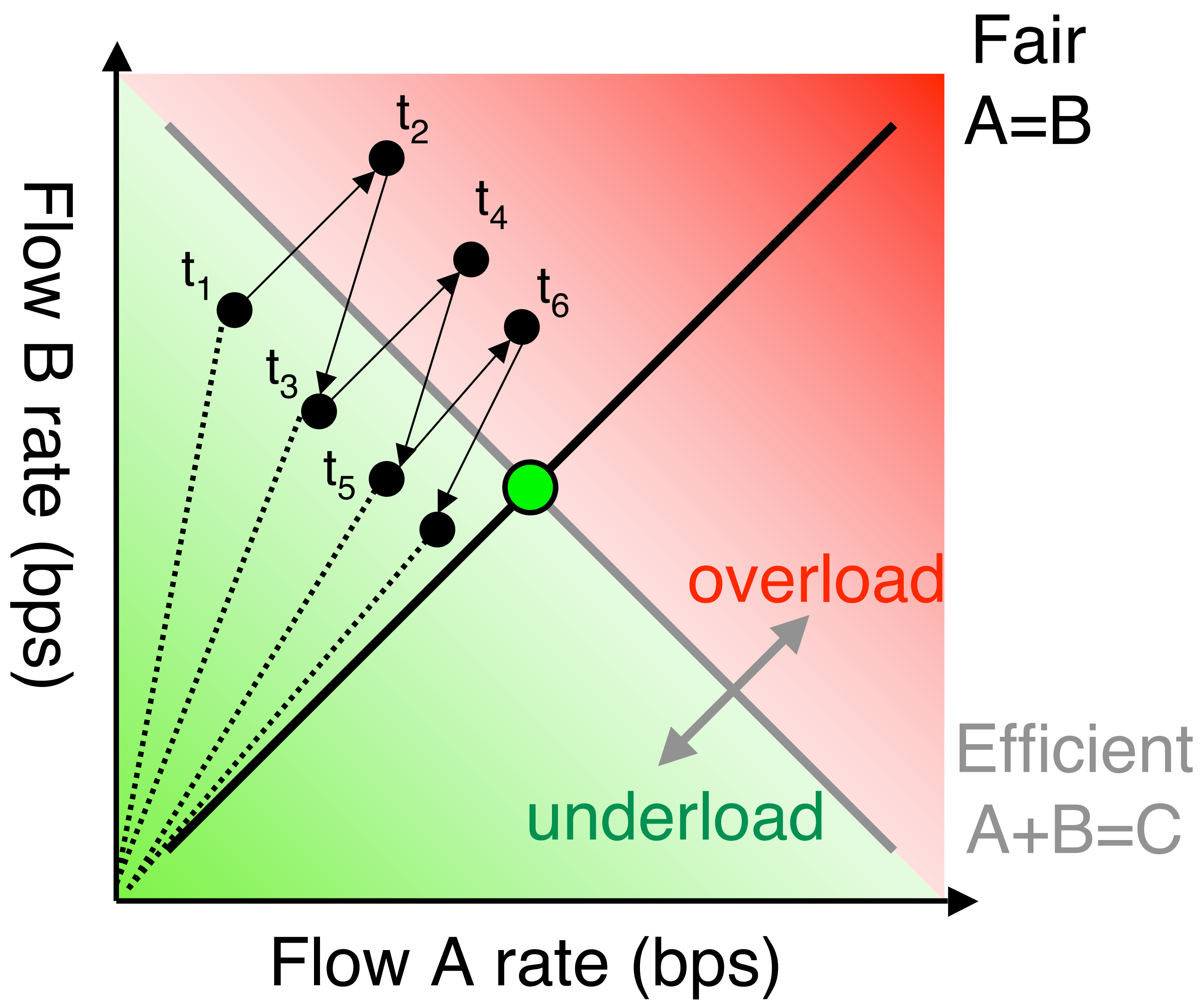
Chiu Jain Plot



Chiu Jain Plot



Chiu Jain Plot

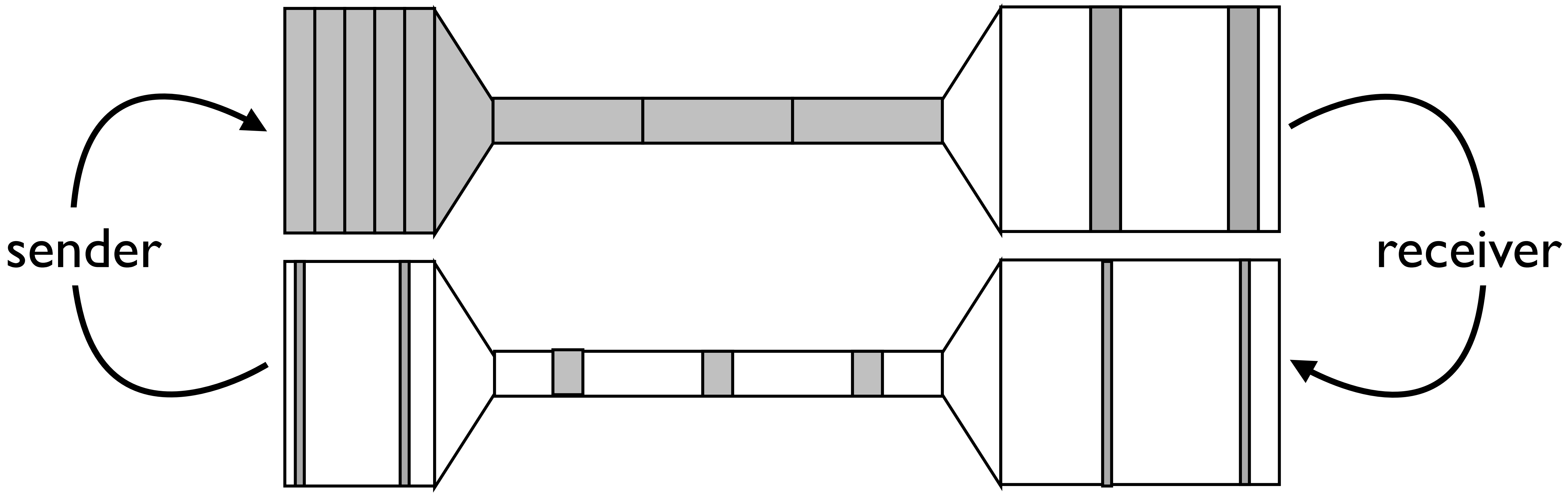


Video!

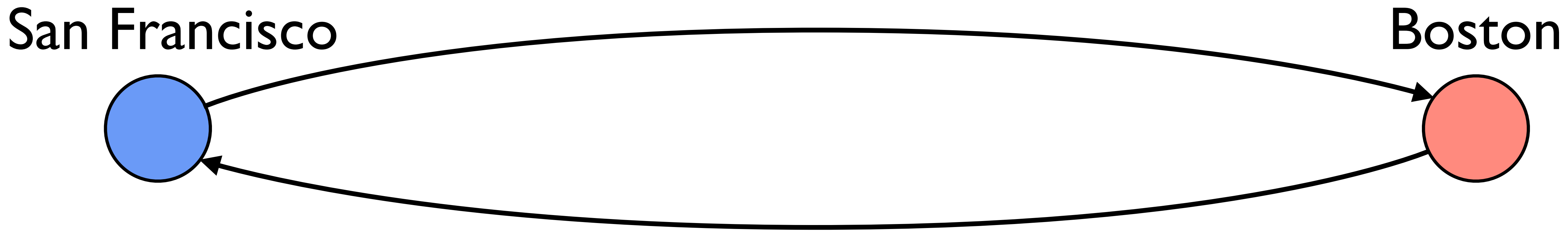
http://guido.appenzeller.net/anims/

Self-Clocking

- In case of a bottleneck link, sender receives acks properly spaced in time



Congestion Window Size



Optimal congestion window size is the bandwidth-delay product

So How Do You Implement It?

Three Questions

- When should you send new data?
- When should you send data retransmissions?
- When should you send acknowledgments?

Three Questions

- When should you send new data?
- When should you send data retransmissions?
- When should you send acknowledgments?

Congestion Window (TCP Tahoe)

- Flow control window is only about endpoint
- Have TCP estimate a *congestion window* for the network
- Sender window = min(flow window, congestion window)
- Separate congestion control into two states
  - Slow start: on connection startup or packet timeout
  - Congestion avoidance: steady operation

Slow Start Benefits

- Slow start
  - Window starts at Maximum Segment Size (MSS)
  - Increase window by MSS for each acknowledged packet
- Exponentially grow congestion window to sense network capacity
- "Slow" compared to prior approach

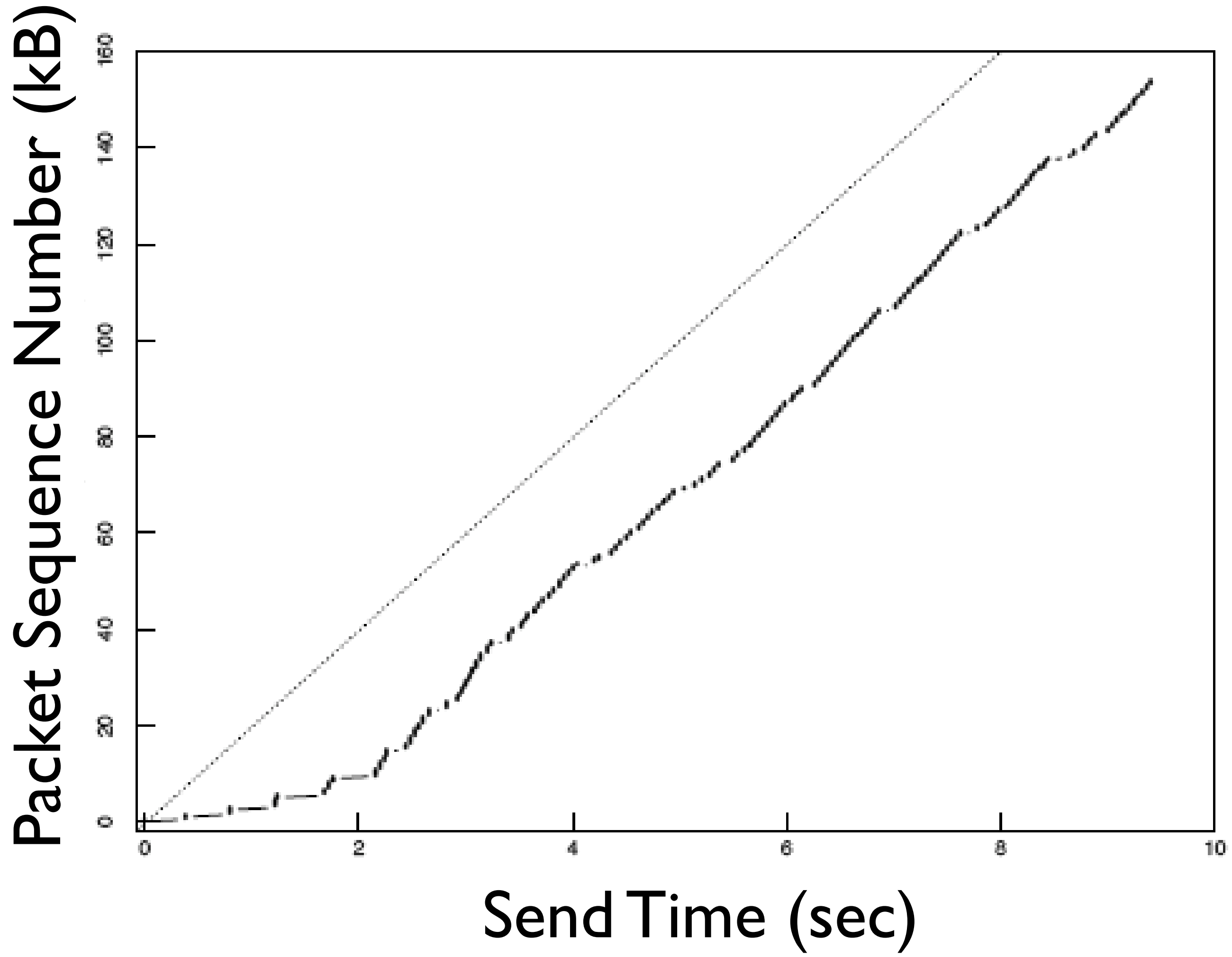


Figure from "Congestion Avoidance and Control", Van Jacobson and Karels. Used with permission.

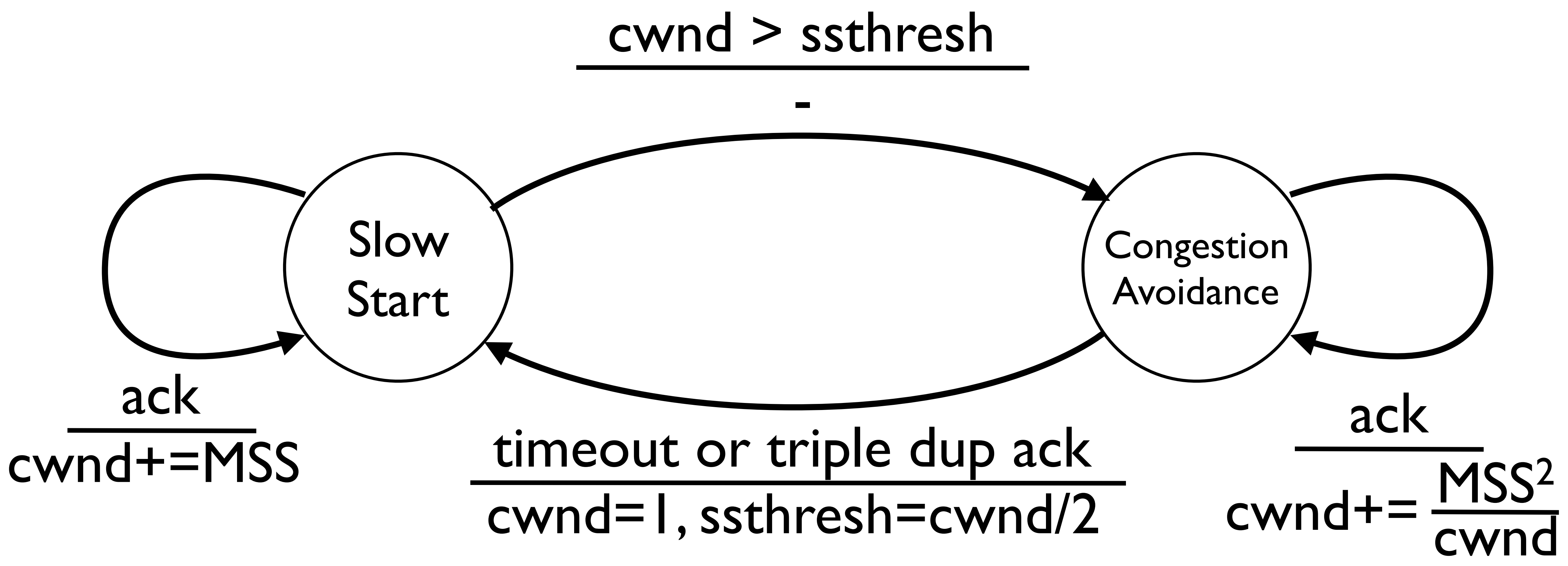
Congestion Avoidance

- Slow start
  - Increase congestion window by MSS for each acknowledgment
  - Exponential increase
- Congestion avoidance
  - Increase by MSS^2/congestion window for each acknowledgment
  - Behavior: increase by MSS each round trip time
  - Linear (additive) increase

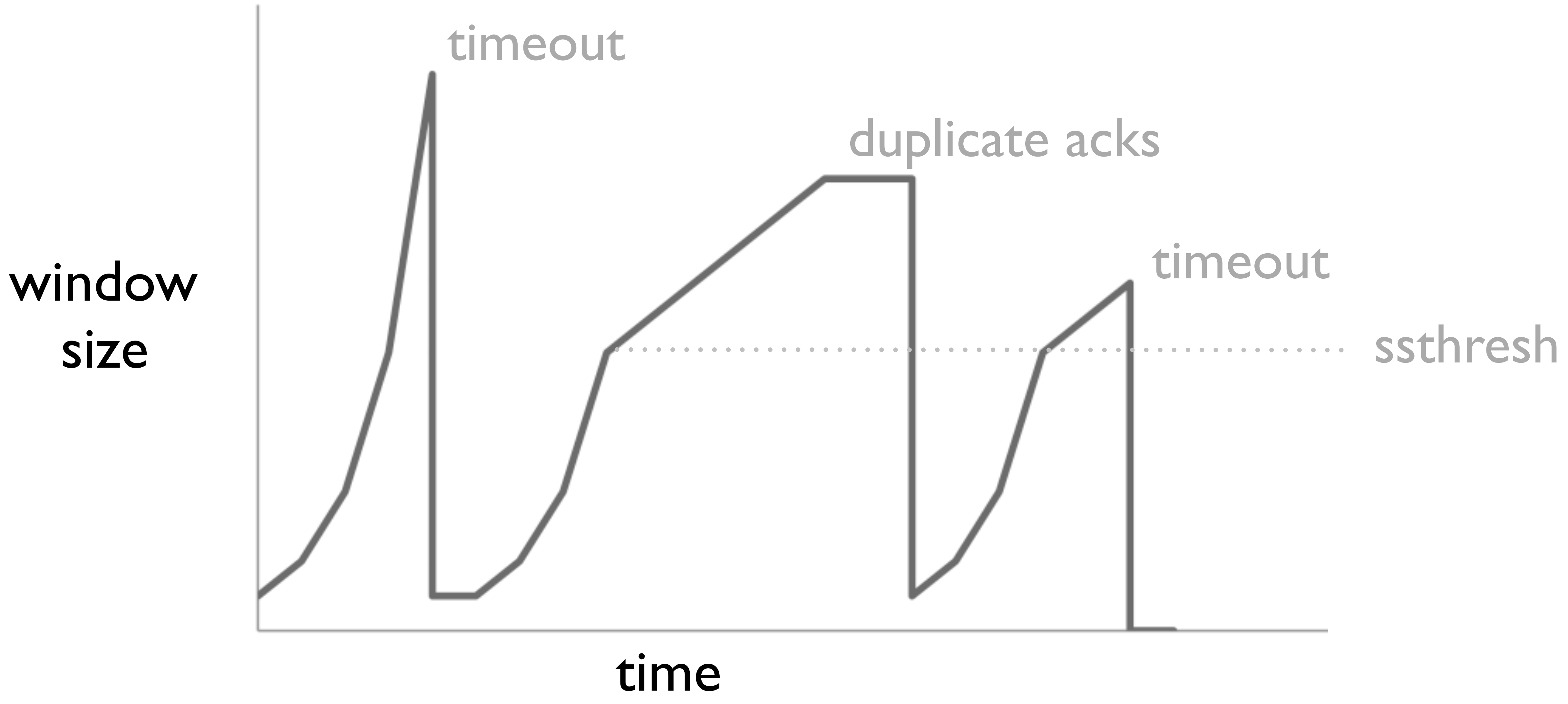
State Transitions

- Two goals
  - Use slow start to quickly find network capacity
  - When close to capacity, use congestion avoidance to very carefully probe
- Three signals
  - Increasing acknowledgments: transfer is going well
  - Duplicate acknowledgments: something was lost/delayed
  - Timeout: something is very wrong

TCP Tahoe FSM



TCP Tahoe Behavior



`

- Fast retransmit (Tahoe+Reno), fast recovery (Reno)
  - NB: Videos incorrect attribute mechanisms to TCP versions; these factoids are not important, it is important you understand the mechanisms and why they help
- Multiple losses in one window only decrease once (NewReno)