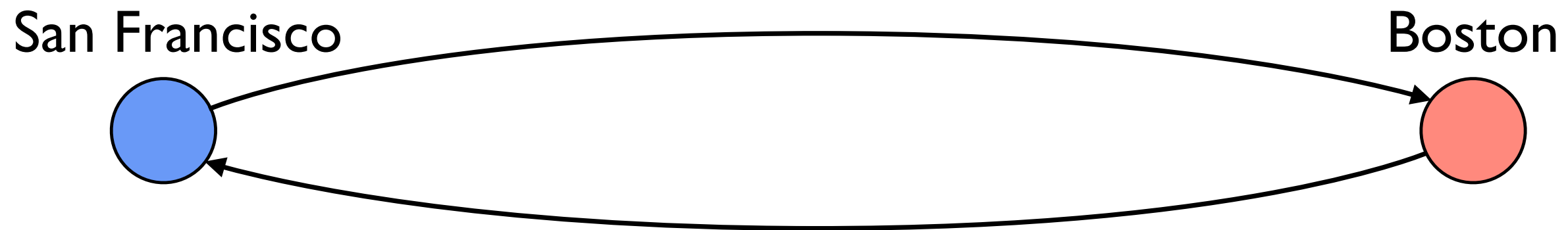


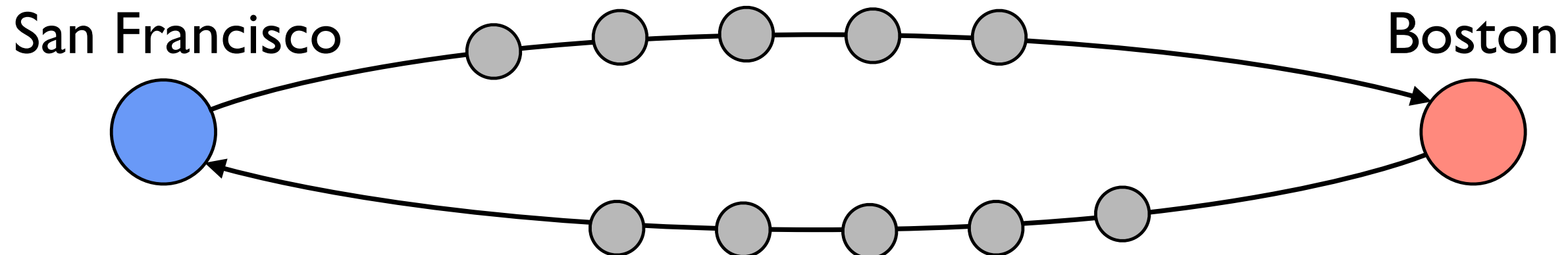
# TCP Congestion Control I

Slow start, congestion avoidance, triple duplicate acks

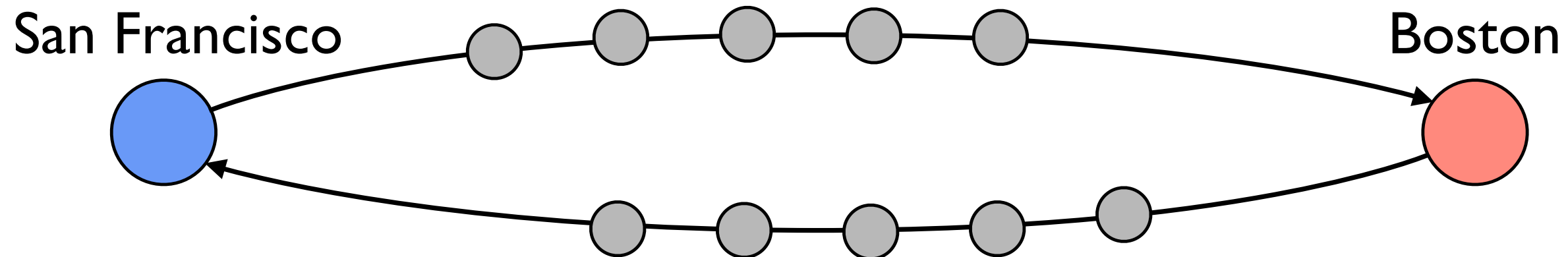
# Congestion Control Motivation



# 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 how it 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)
- 1990: Fast recovery added (Reno)

# 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?



# TCP Pre-Tahoe

- 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?

# TCP in 1986

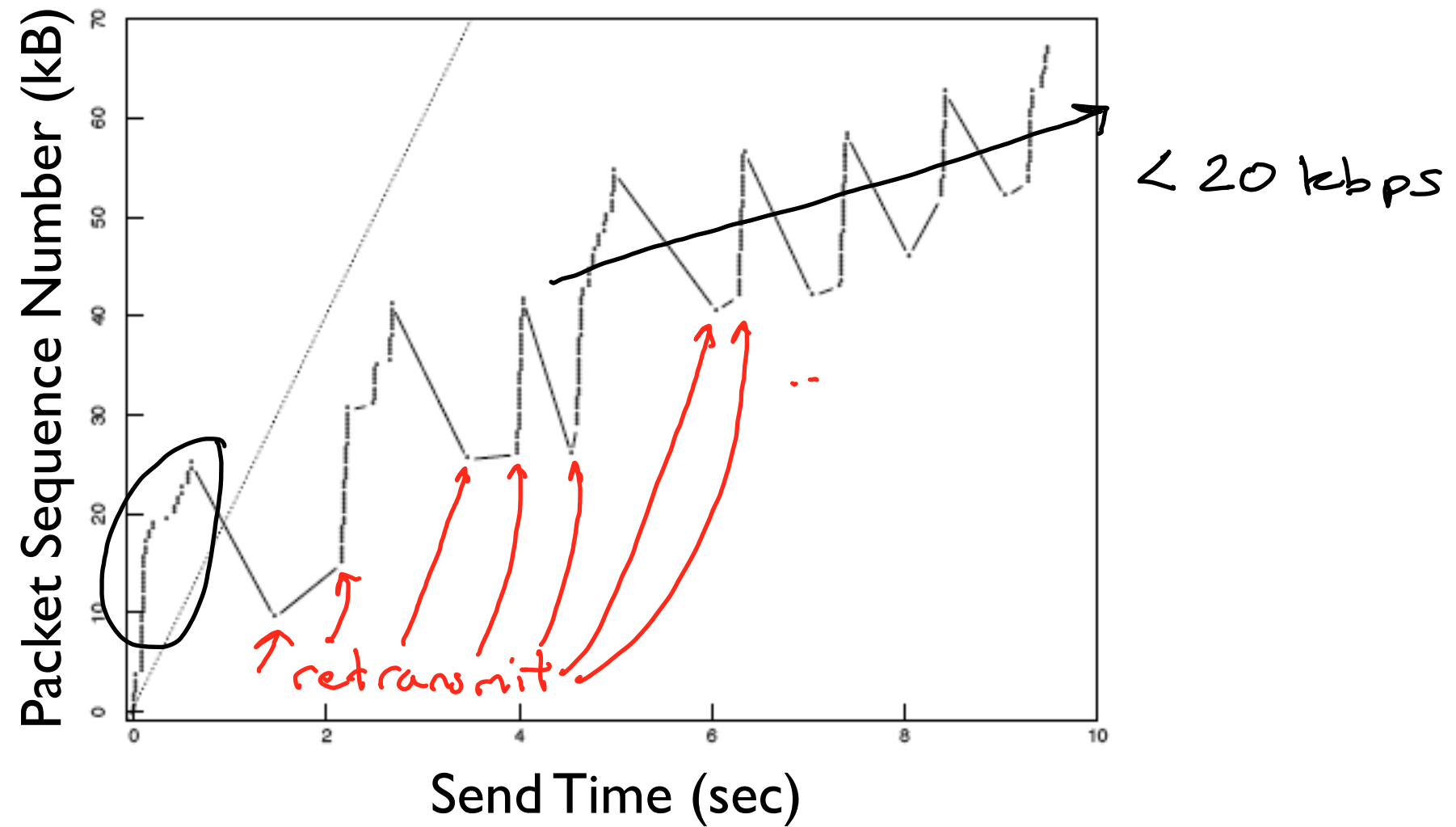


Figure from "Congestion Avoidance and Control", Van Jacobson and Karels. Used with permission.

# Three Improvements

- Congestion window
- Timeout estimation
- Self-clocking

# Three Improvements

- Congestion window
- Timeout estimation
- Self-clocking

# Congestion Window (TCP Tahoe)

- Flow control window is only about endpoint
- Have TCP estimate a *congestion window* for the network
- Sender window =  $\min(\text{flow window}, \text{congestion window})$
- 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

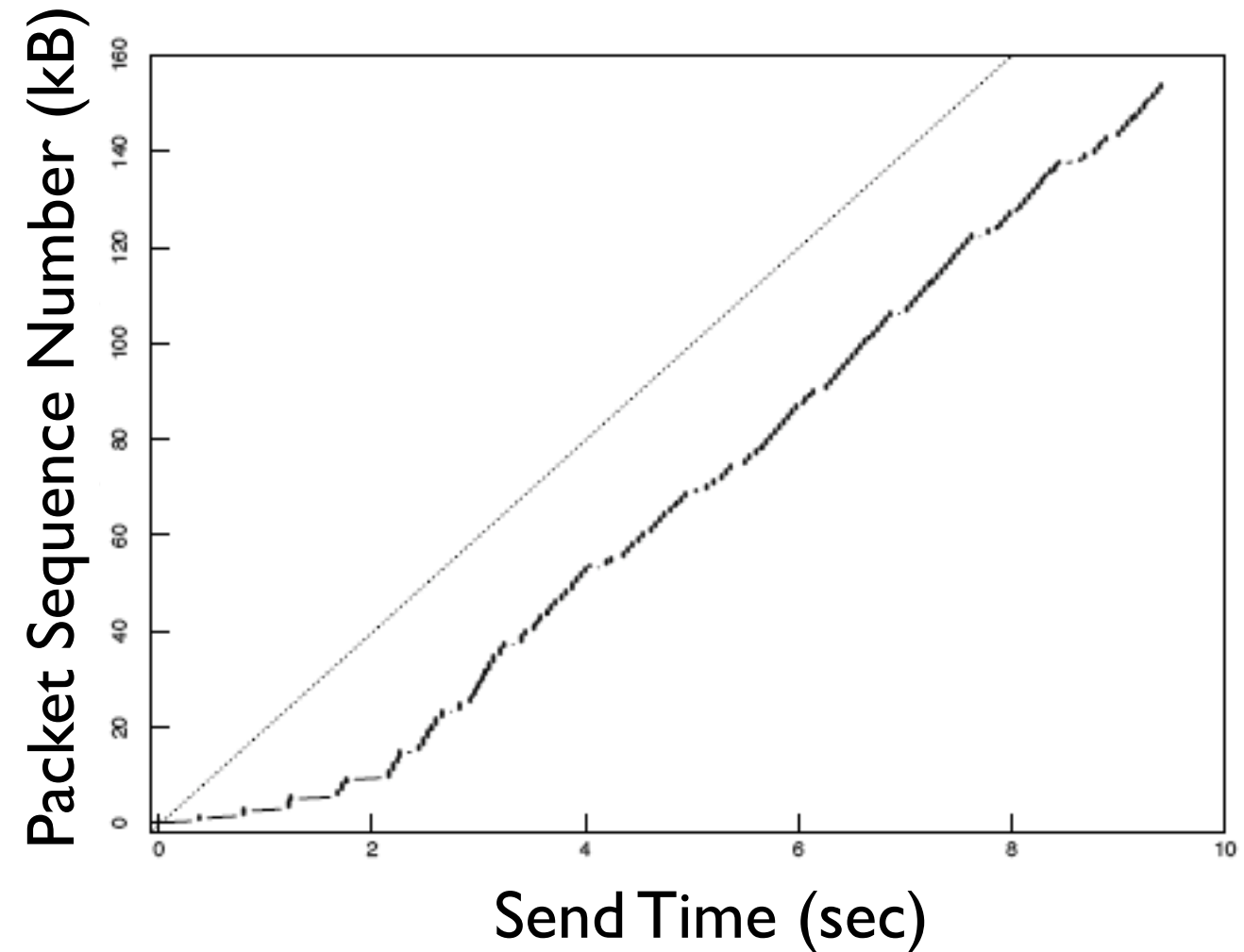


Figure from “Congestion Avoidance and Control”, Van Jacobson and Karels. Used with permission.

# Congestion Avoidance

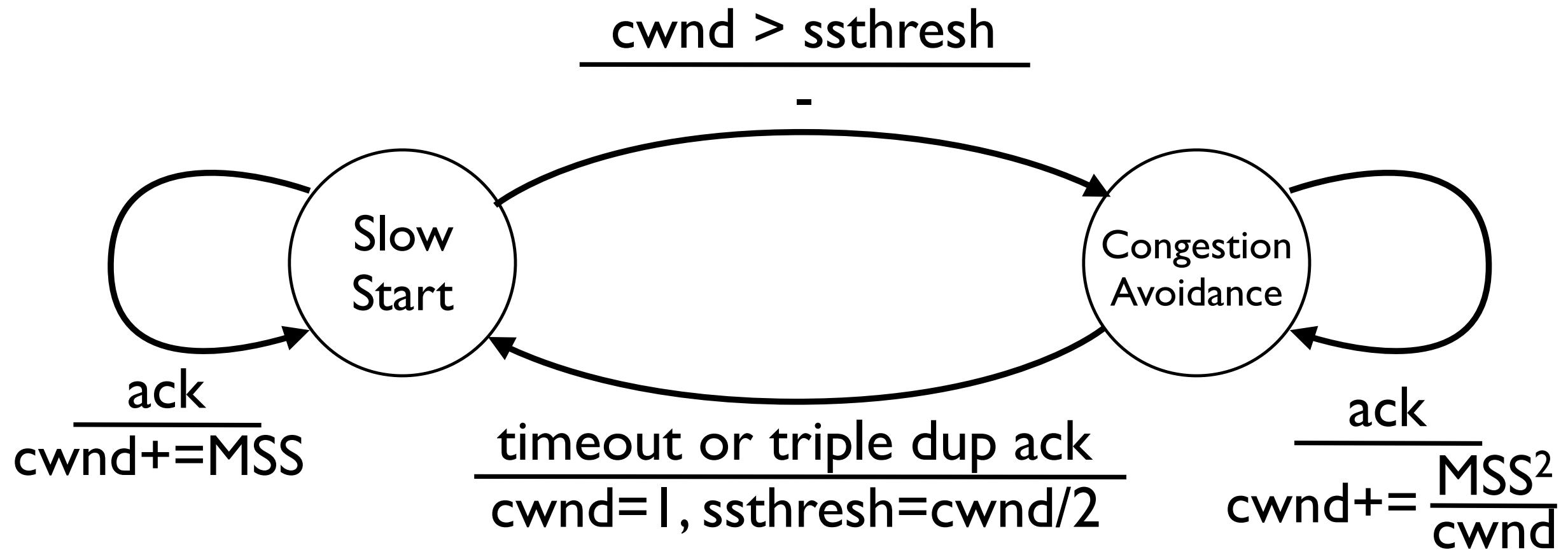
- Slow start
  - ▶ Increase congestion window by MSS for each acknowledgment
  - ▶ Exponential increase
- Congestion avoidance
  - ▶ Increase by  $MSS^2 / \text{congestion window}$  for each acknowledgment
  - ▶ Behavior: increase by MSS each round trip time
  - ▶ Linear (additive) increase

# 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

