Congestion Control III

Performance improvements: fast retransmit and fast recovery

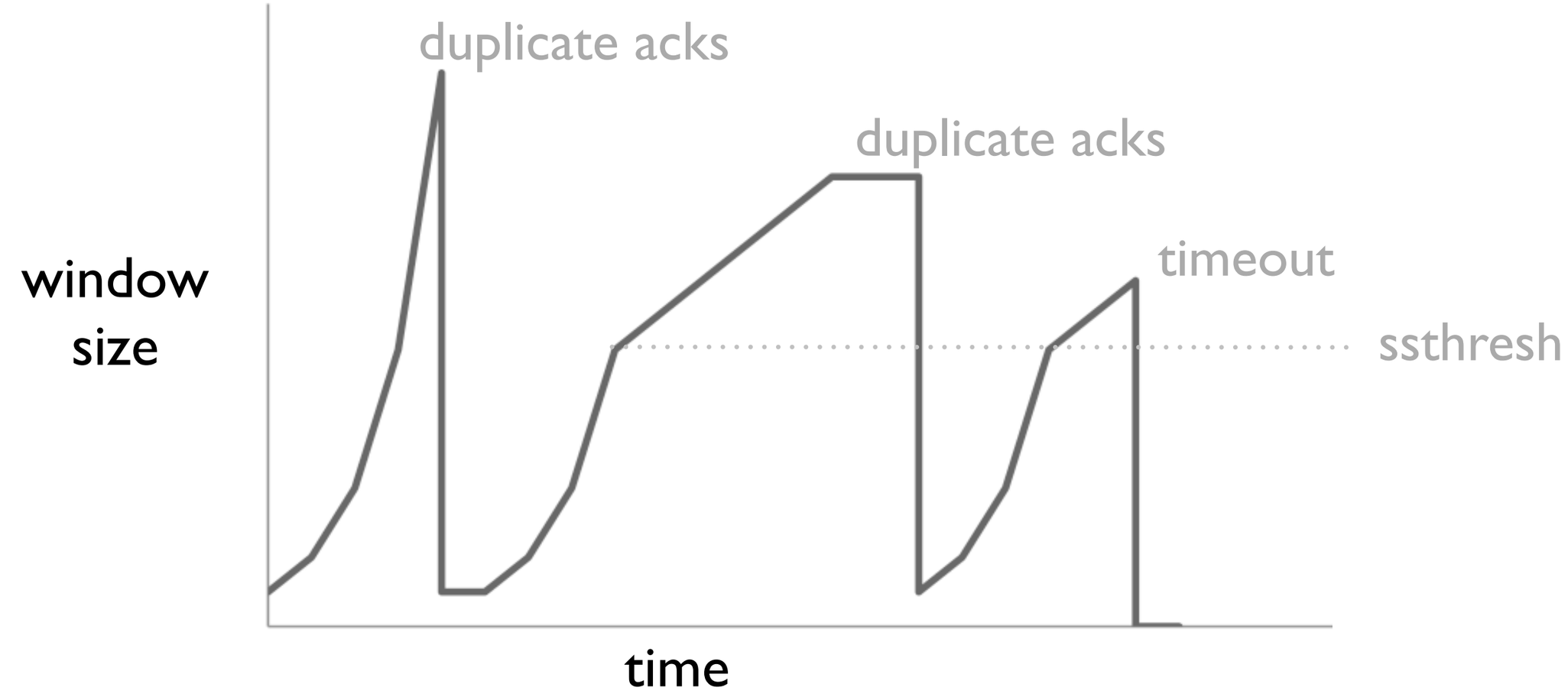
Three Mechanisms

- Fast retransmit (TCP Tahoe): don't wait for a timeout to retransmit a missing segment if you receive a triple duplicate acknowledgement
  - Only drop back to slow start state on a timeout
- Fast recovery (TCP Reno): halve the congestion window (don't set it to 1) on triple duplicate acknowledgements
- Fast recovery (TCP Reno): while in fast recovery state, inflate the congestion window as acknowledgements arrive, to keep data flowing
  - Each duplicate ack increases congestion window by 1
  - If the old window is c, then the new window is c/2
  - Receiving c acks will increase window size to 3c/2 -- can send c/2 new segments

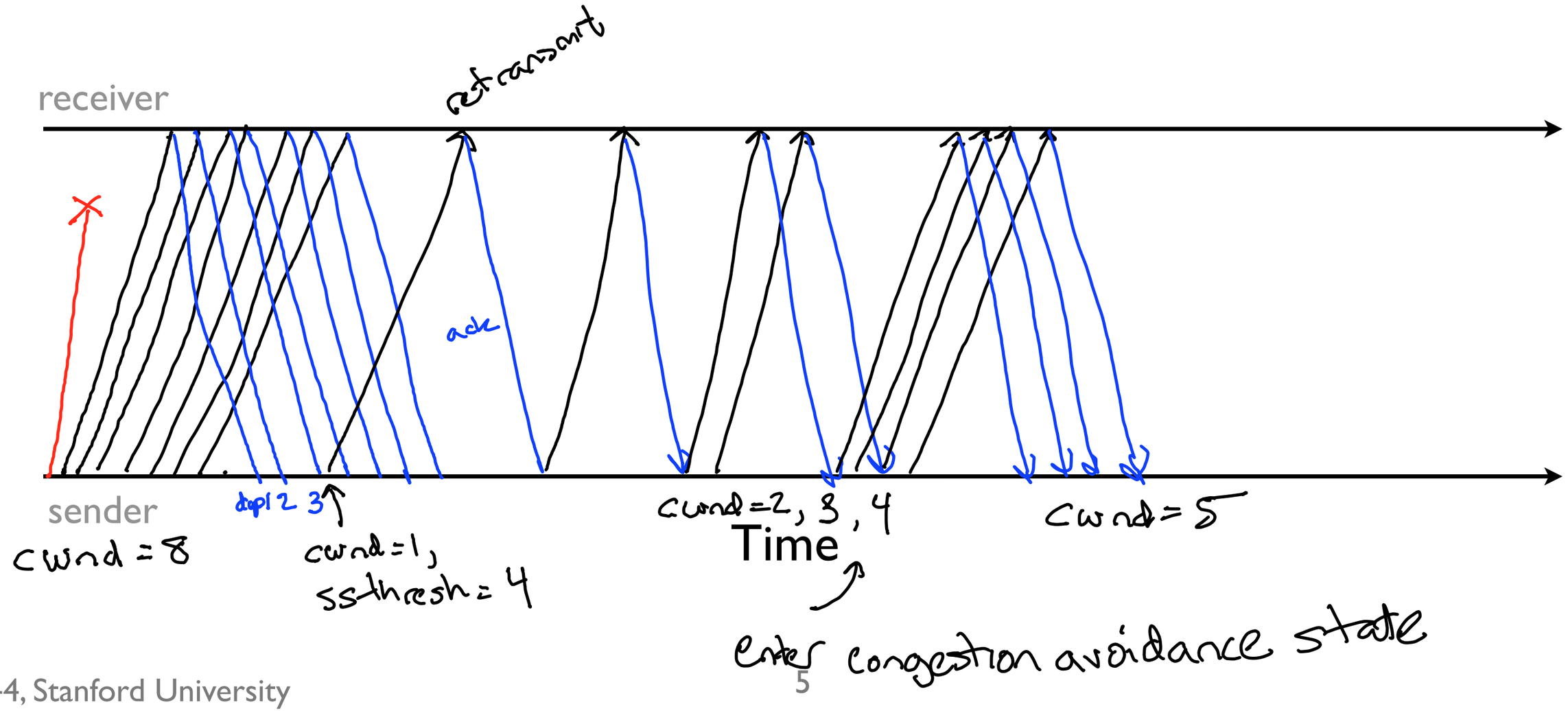
TCP Tahoe

- On timeout or triple duplicate ack (implies lost packet)
  - Set threshold to congestion window/2
  - Set congestion window to 1
  - Retransmit missing segment (fast retransmit for triple duplicate ack)
  - Enter slow start state

TCP Tahoe Behavior



TCP Tahoe Walkthrough



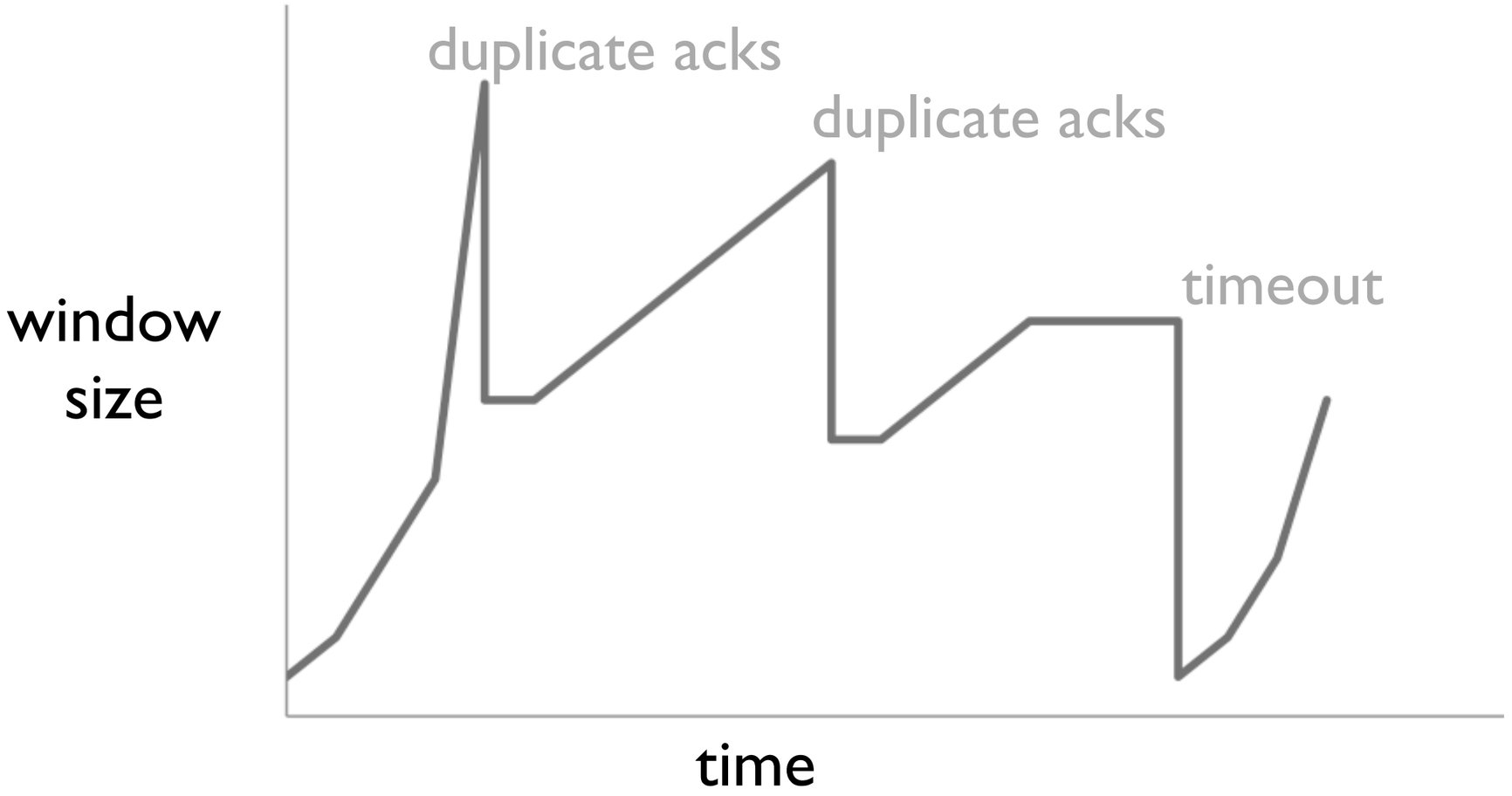
TCP Reno

- Same as Tahoe on timeout
- On triple duplicate ack
  - Set threshold to congestion window/2
  - Set congestion window to congestion window/2 (fast recovery)
  - Inflate congestion window size while in fast recovery (fast recovery)
  - Retransmit missing segment (fast retransmit)
  - Stay in congestion avoidance state

TCP Reno

- Same as Tahoe on timeout
- On triple duplicate ack
  - Set threshold to congestion window/2
  - Set congestion window to congestion window/2 (fast recovery)
  - Inflate congestion window size while in fast recovery (fast recovery)
  - Retransmit missing segment (fast retransmit)
  - Stay in congestion avoidance state

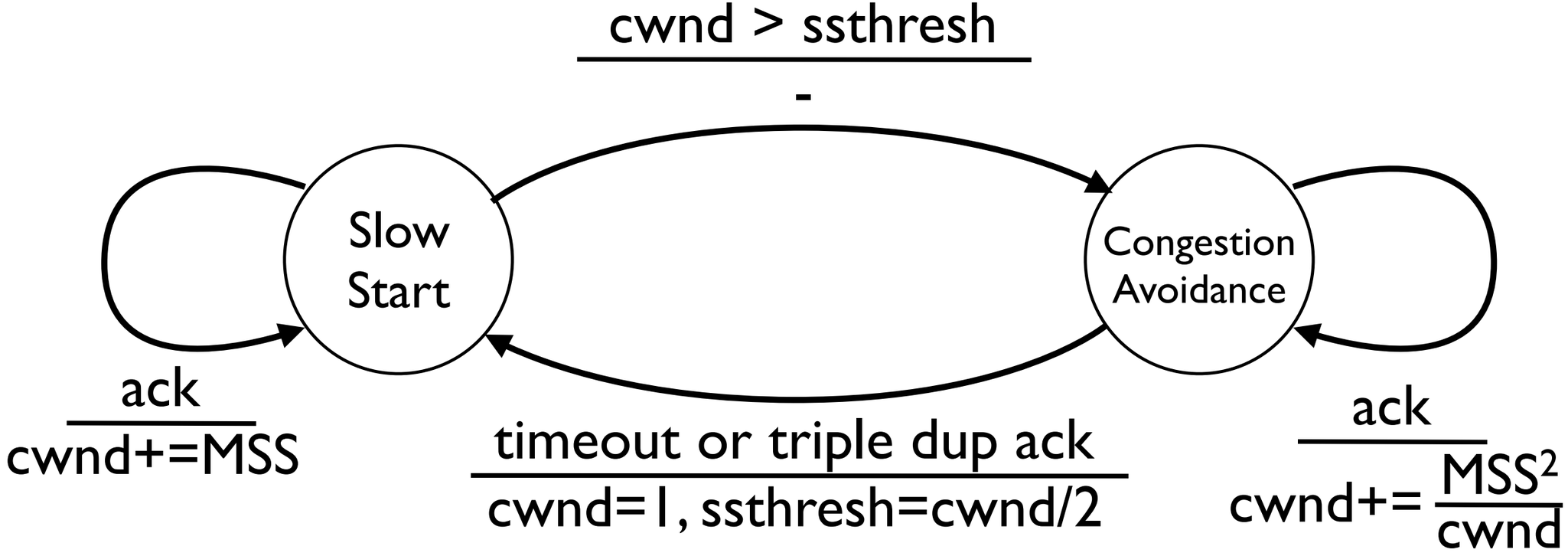
TCP Reno



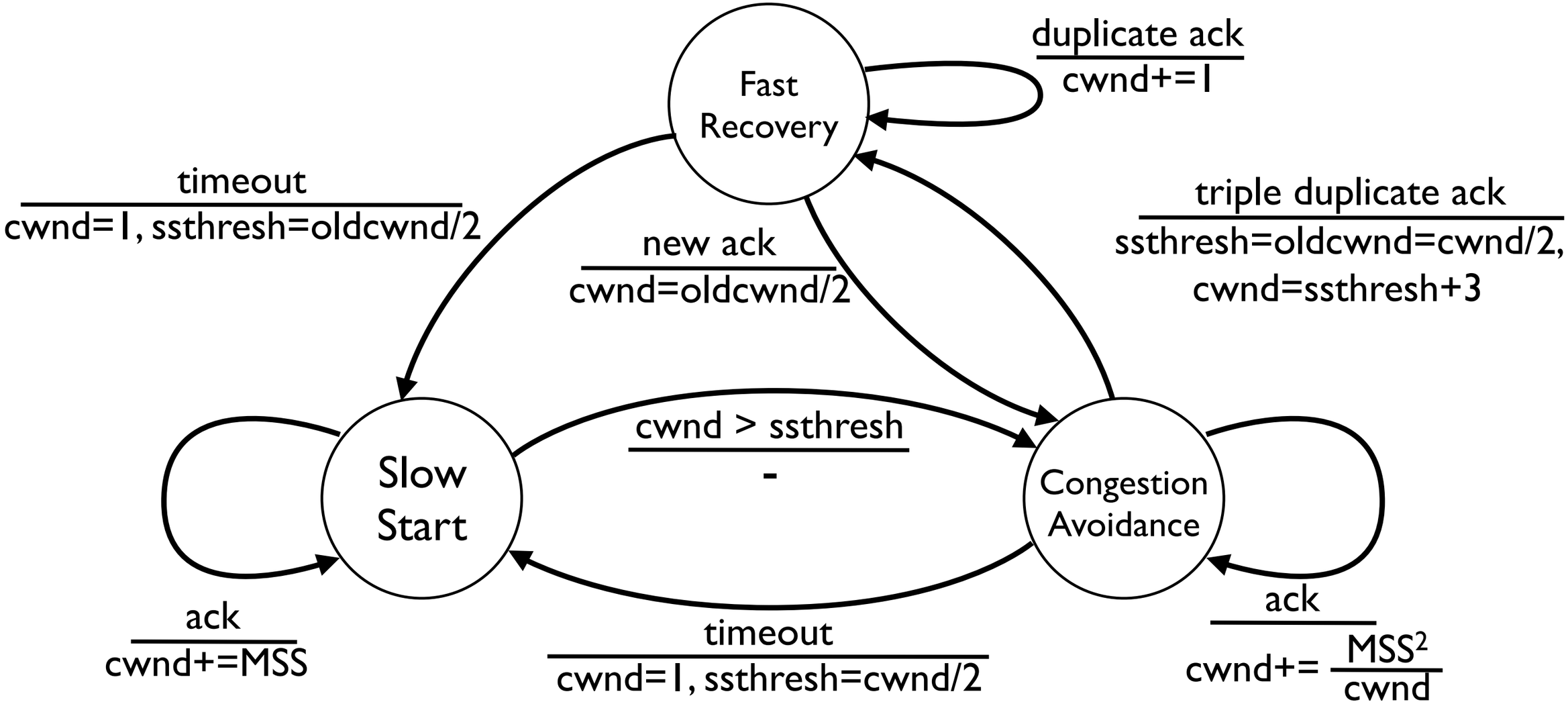
Congestion Window Inflation

- Problem: it takes a full RTT for a fast retransmitted packet to advance the congestion window
- Could put more packets into network
- Solution: congestion window inflation
  - While in the fast recovery state (haven't received new acknowledgements), increase congestion window size by 1 for each duplicate acknowledgement, including the initial 3
  - This happens after halving congestion window size ($cwnd_{new} = cwnd_{old}/2$)
  - End result: after one RTT, $cwnd_{new}$ is $3*cwnd_{old}/2$ but since no new acks yet, this results in sending $cwnd_{old}/2$ new packets

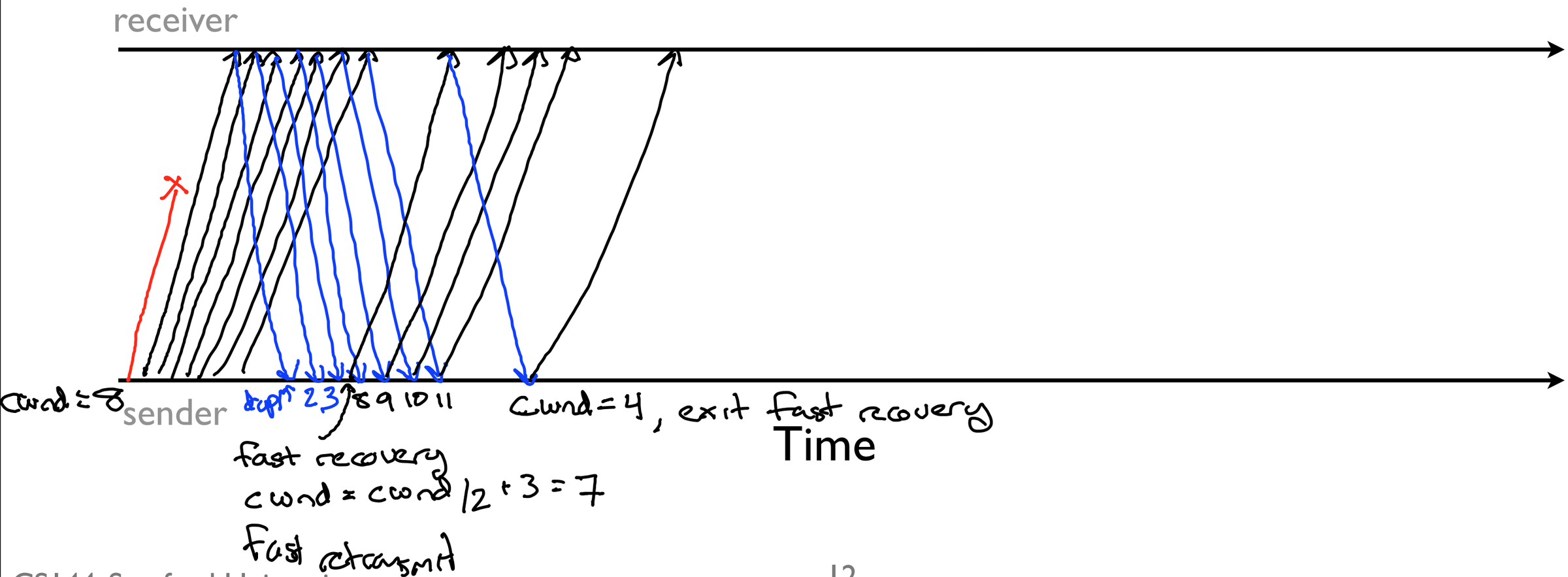
TCP Tahoe FSM



TCP Reno FSM


Fast Recovery
duplicate ack
cwnd+=1
timeout
cwnd=1, ssthresh=oldcwnd/2
new ack
cwnd=oldcwnd/2
triple duplicate ack
ssthresh=oldcwnd=cwnd/2,
cwnd=ssthresh+3
Slow Start
cwnd > ssthresh
-
Congestion Avoidance
ack
cwnd+=MSS
timeout
cwnd=1, ssthresh=cwnd/2
ack
cwnd+= MSS²/cwnd

TCP Reno Walkthrough



Congestion Control

- One of the hardest problems in robust networked systems
- Basic approach: additive increase, multiplicative decrease
- Tricks to keep pipe full, improve throughput
  - Fast retransmit (don't wait for timeout to send lost data)
  - Congestion window inflation (don't wait an RTT before sending more data)