TCP Congestion Control II

RTT Estimation, self-clocking

Three Questions

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

Three Improvements

- Congestion window
- Timeout estimation
- Self-clocking

Timeouts

- Round trip time estimation is critical for timeouts
  - Too short: waste capacity with restransmissions, trigger slow start
  - Too long: waste capacity with idle time
- Challenge: RTT is highly dynamic
- Challenge: RTT can vary significantly with load

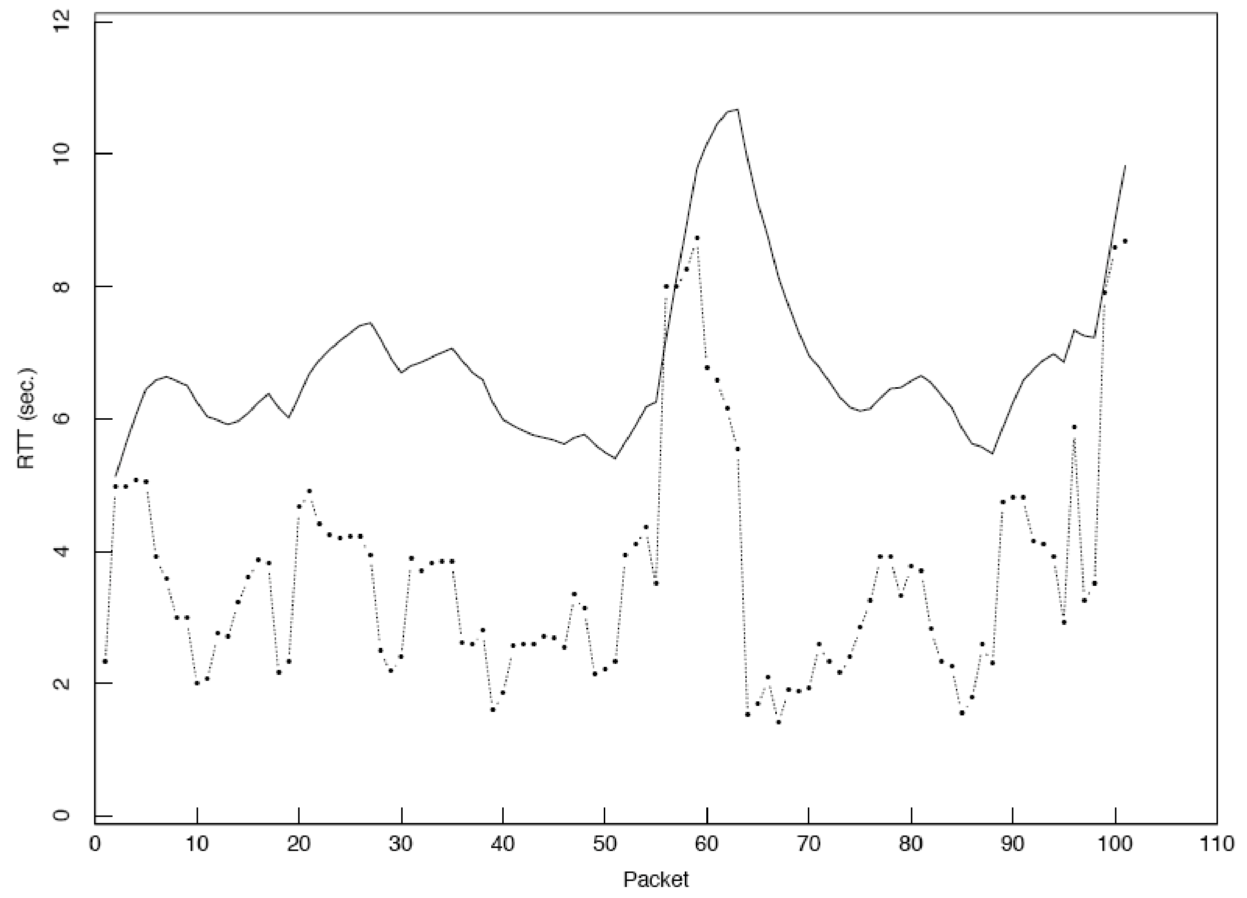
Pre-Tahoe Timeouts

- r is RTT estimate, initialize to something reasonable
- m, RTT measurement from most recently acked data packet
- Exponentially weighted moving average: $r = \alpha r + (1-\alpha)m$
- Timeout = βr, $\beta=2$
- What's the problem?

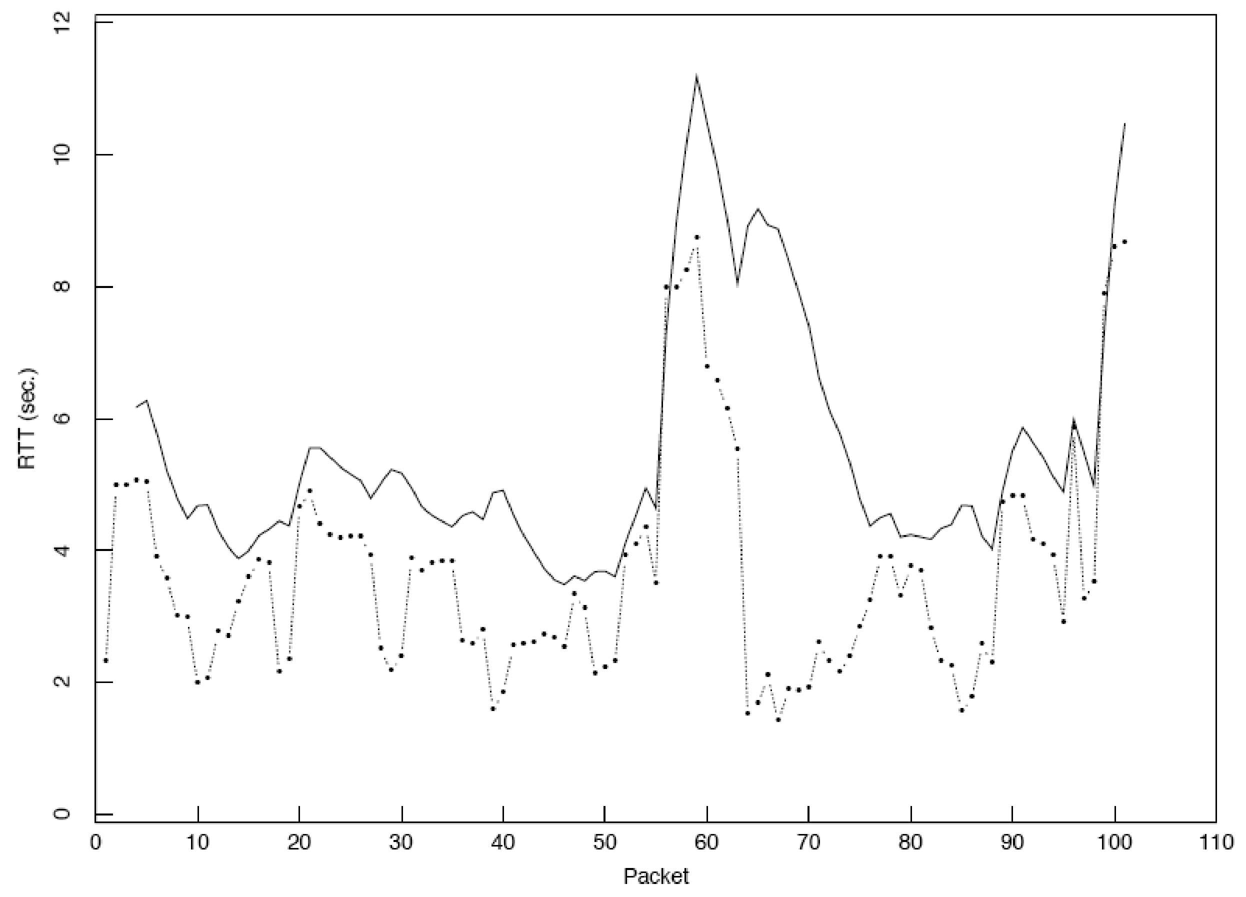
TCP Tahoe Timeouts

- r is RTT estimate, initialize to something reasonable
- g is the EWMA gain (e.g., 0.25)
- m is the RTT measurement from most recently acked data packet
- Error in the estimate $e = m-r$
- $r = r + g \cdot e$
- Measure variance $v = v + g(|e| - v)$
- Timeout = $r + \beta v$ (β=4)
- Exponentially increase timeout in case of tremendous congestion

RTT Estimation Improvement



Pre-Tahoe



Tahoe

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

Three Questions

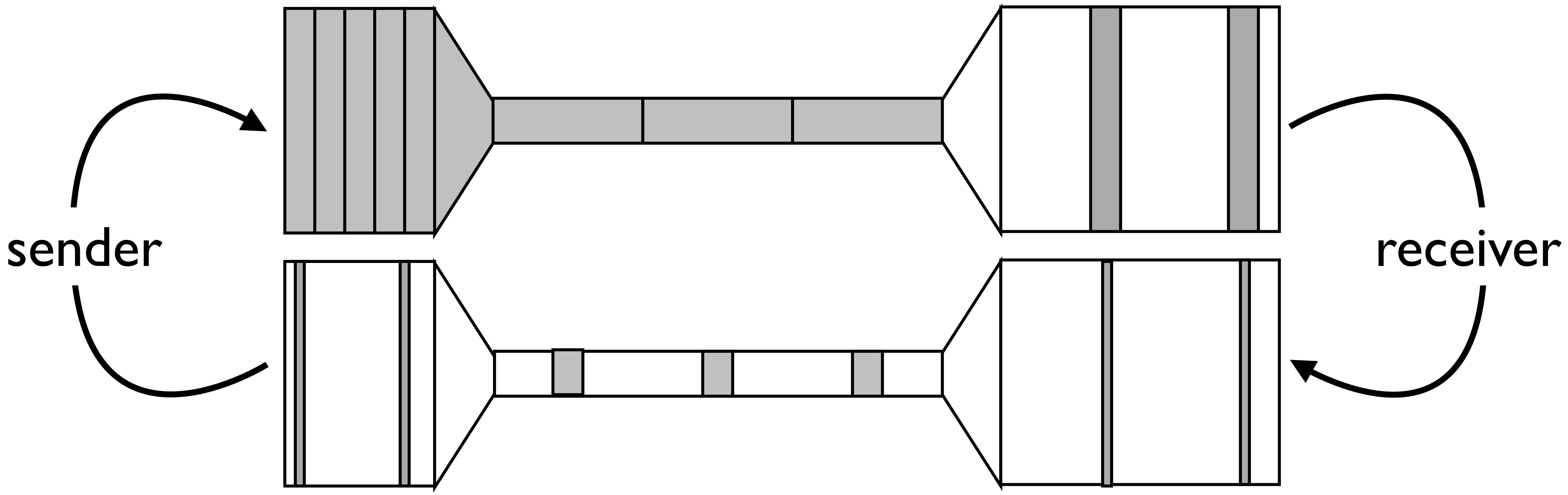
- When should you send new data?
- When should you send data retransmissions?
- When should you send acknowledgments?

Three Improvements

- Congestion window
- Timeout estimation
- Self-clocking

Self-Clocking

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



Self-Clocking Principle

- Only put data in when data has left
  - Want to prevent congestion -- too much data in network
- Send new data in response to acknowledgments
- Send acknowledgments aggressively -- important signal