Advances in TCP Functionality

- Better Congestion control
  - Can throughput be improved over the Reno protocol we studied?
  - For low latency applications, can we reduce delays by not filling up buffers?
- Improving Loss Detection:
  - Can we distinguish between losses due to buffer overflow (congestion related) vs. wireless link losses due to poor channels
  - Can we distinguish between genuine packet losses vs. packets delayed and thus arriving out of sequence

Can we do better in congestion avoidance?

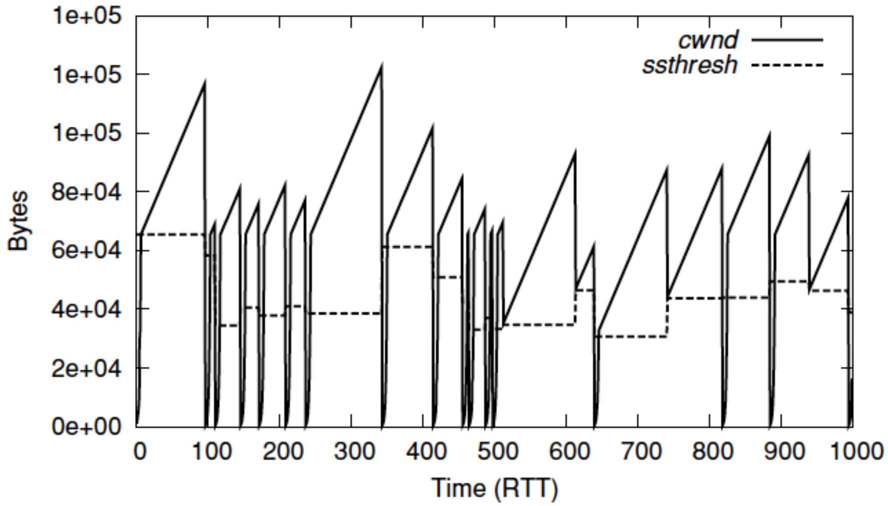


Figure 6.6. The evolution of *cwnd* and *ssthresh* for a TCP connection, including slow start, congestion avoidance, fast retransmit, and fast recovery.

Table 6.1. *The slow start and congestion avoidance algorithms*

(1) If $cwnd \le ssthresh$ then /* *Slow Start* Phase */
  Each time an ACK is received:
    $cwnd = cwnd + segsize$
  else /* *Congestion Avoidance* Phase */
  Each time an ACK is received:
    $cwnd = cwnd + segsize \times segsize/cwnd + segsize/8$
  end

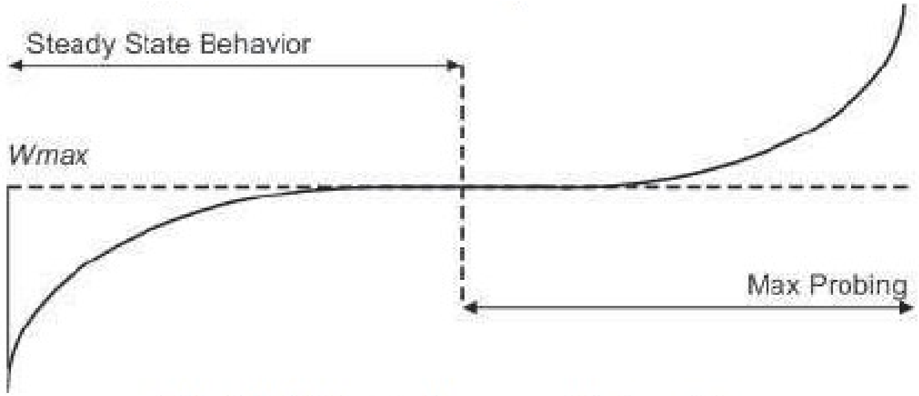
(2) When congestion occurs (indicated by retransmission timeout)
    $ssthresh = \max(2, \min(cwnd, awnd)/2)$
    $cwnd = 1\ segsize = 1\ MSS\ bytes$

(3) $Allowed\ window = \min(cwnd, awnd)$

- Key idea: can we improve the cwnd growth function in congestion avoidance?
- TCP Reno growth function for congestion avoidance increases a fixed amount in each round-trip time (RTT)
- Linear, extremely slow for networks with high BDP (bandwidth delay product)

1. S. Ha, I. Rhee, and L. Xu, "Cubic: A new TCP-friendly high-speed TCP variant," SIGOPS Oper. Syst. Rev., vol. 42, no. 5, p. 64{74, Jul. 2008. [Online]. Available: https://doi.org/10.1145/1400097.1400105

Improve network utilization – TCP CUBIC [1]



(b) CUBIC window growth function.

- CUBIC growth function: $W(t) = C\,(t-K)^3 + W_{max}$
  - Two regions: **concave** (steady state), **convex** (max probe), inflection point is $K = \sqrt[3]{\frac{W_{max}\beta}{C}}$
  - W_{max} : window size for previous loss event, β: multiplicative decrease factor, C: CUBIC parameter
  - **Concave region** : Concave profile to increase cwnd in steady state after a loss event, quick initial increase with decreasing rate, plateauing at W_{max}
  - **Convex region** : Convex profile after window crosses W_{max} designed to improve network utilization by quickly occupying network bandwidth. Essentially, this means probing for a new W_{max}

1. S. Ha, I. Rhee, and L. Xu, "Cubic: A new TCP-friendly high-speed TCP variant," SIGOPS Oper. Syst. Rev., vol. 42, no. 5, p. 64{74, Jul. 2008. [Online]. Available: https://doi.org/10.1145/1400097.1400105

Proactive instead of reactive – BBR [2,3]

- Key idea: can we proactively avoid congestion instead of reacting to events?
- Previous TCP variants (CUBIC, Reno, etc.) - loss-based congestion control
- Detect packet losses, then adjust congestion window using growth functions
- Operating point is close to buffer-limited regime – suboptimal when buffer size much larger than BDP, often causing RTTs of seconds instead of milliseconds.
- Optimal point – transition from buffer-limited to bandwidth-limited – maximizing bandwidth while minimizing delay and loss (Kleinrock)

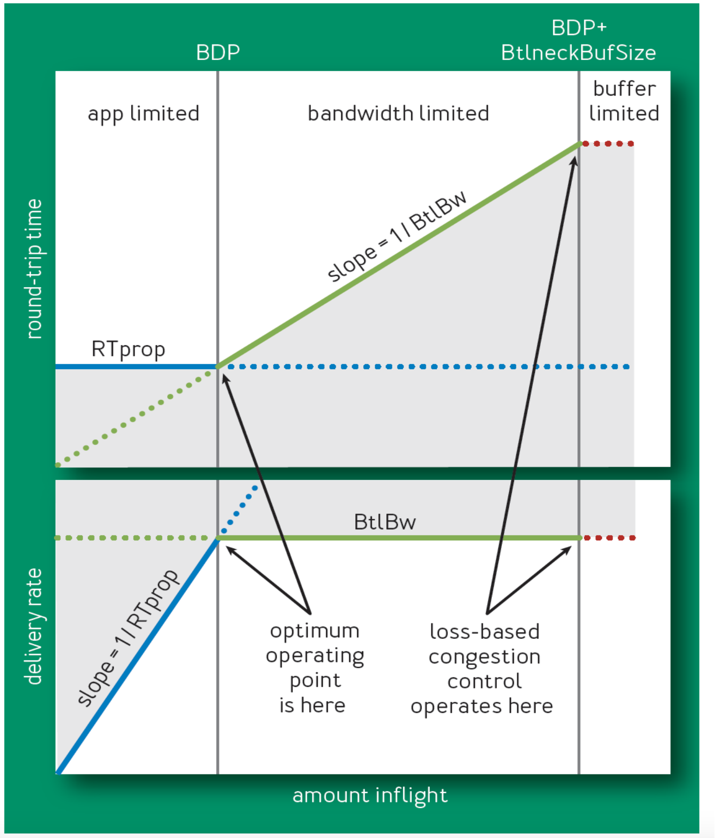
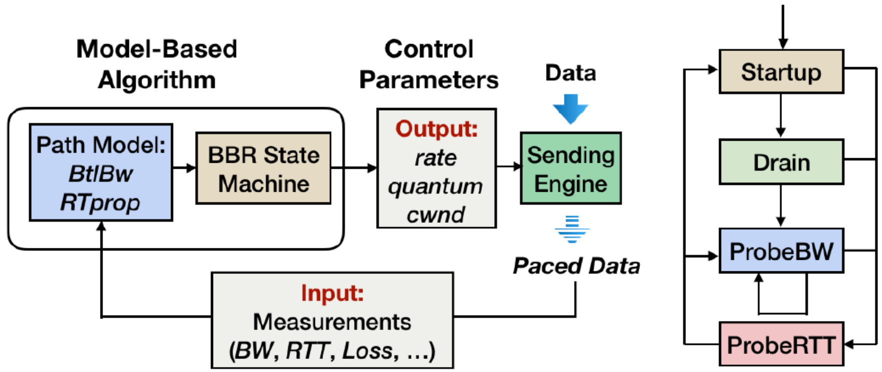


Figure: Delivery rate and round-trip time vs number of packets inflight[2]

2. N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, “BBR: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time," Queue, vol. 14, no. 5, p. 20{53, Oct. 2016. [Online]. Available: https://doi.org/10.1145/3012426.3022184

Proactive instead of reactive – BBR [2,3]



(a) BBR high level design. (b) State machine.

Figure: BBR congestion control algorithm design [3]

- BBR - state machine, periodic input measurements (Bandwidth, RTT, loss rate)
- BtlBw – models bandwidth with max filter, Rtprop – models RTT with min filter
- Uses current state (BtBlw, RTT) to determine control factors *pacing_gain* (scale BtlBw), *cwnd_gain* (scale cwnd). Inter-packet spacing determined by *pacing-rate* (regulated between 0.75-1.25 BtlBw) – primary controller.
- Does not *explicitly* respond to losses, *feedback-driven* – avoids congestion

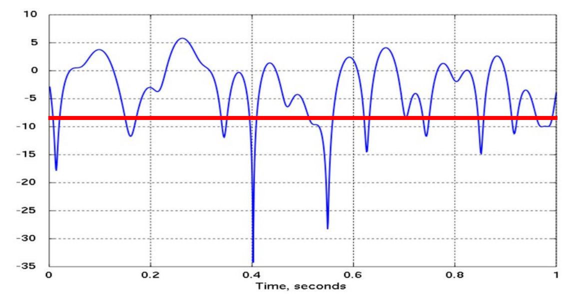
3. Y. Cao, A. Jain, K. Sharma, A. Balasubramanian, and A. Gandhi, "When to use and when not to use BBR: An empirical analysis and evaluation study," in Proceedings of the Internet Measurement Conference, ser. IMC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 130–136. [Online]. Available: https://doi.org/10.1145/3355369.3355579

Detecting Packet Loss

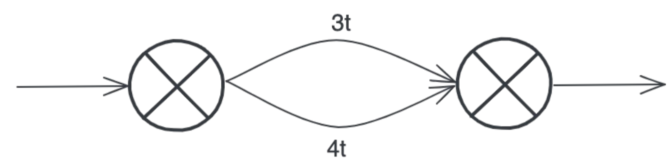
Potential Problems: Network Packet Reordering

What might cause reordering?

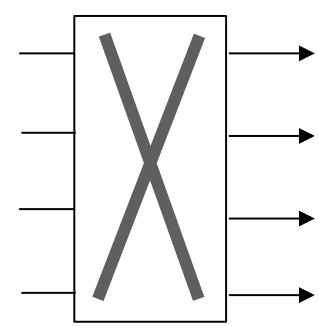
•Wireless Link Layer Retransmissions



•Multipath Routing / Load-Balancing

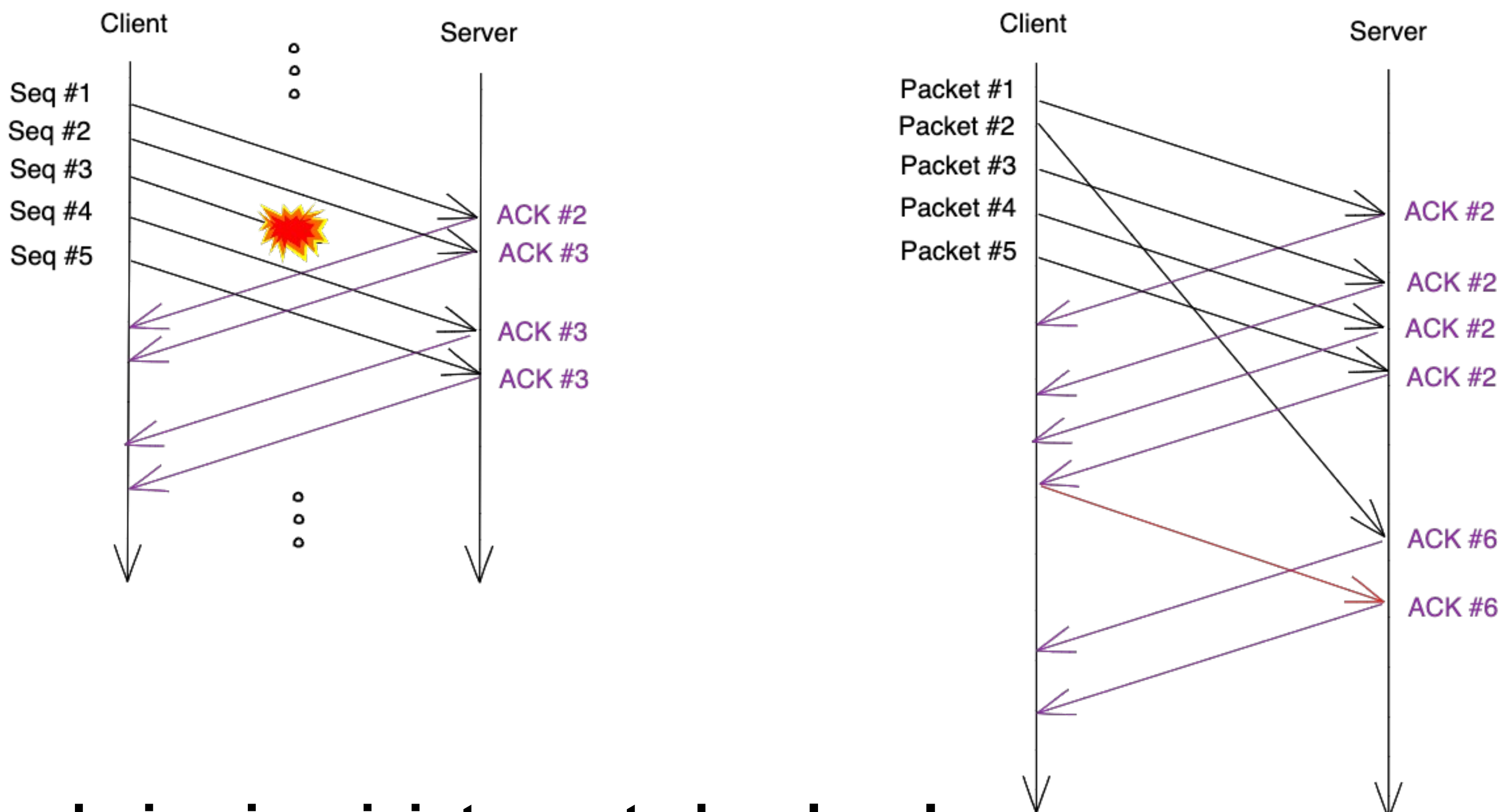


•Switch Architectures (some)



Detecting Packet Loss

How does triple duplicate ACK rule do under reordering?



- **Reordering is misinterpreted as loss!**

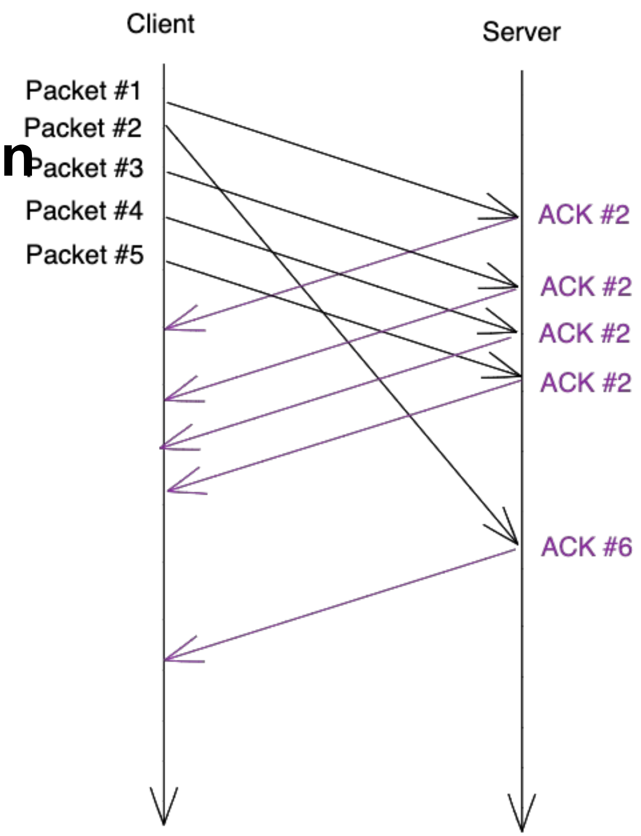
Detecting Packet Loss

How to avoid this?

•**Heuristics for adapting the threshold to trigger a retransmission**

Wait for more than 3 duplicate ACKs.

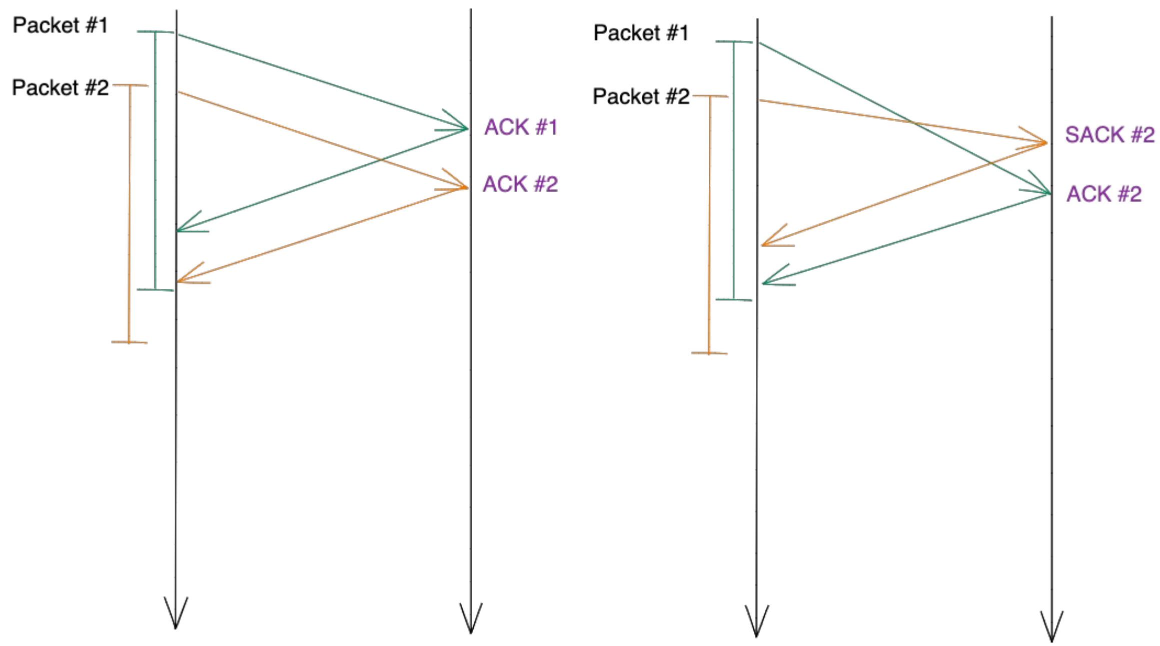
•Slows down error detection!



Detecting Packet Loss

Time based methods for loss detection – e.g. RFC 8985 RACK

- **Keep a timer for each transmission.**



The order of arriving segments does not matter as long as they arrive on time; can be reordered if necessary before being sent to the higher layer.