CSC2231: TCP 101

http://www.cs.toronto.edu/~stefan/courses/csc2231/05au

Stefan Saroiu

Department of Computer Science

University of Toronto

Administrivia

- **On Thursday:**
  - Project progress reports due at noon!
  - 2 hour lecture:
    - 12-1: proper lecture
    - 1-2: “mock” PC meeting
      - If you ranked a paper high, come prepared to defend it
      - If you ranked a paper low, come prepared to reject it

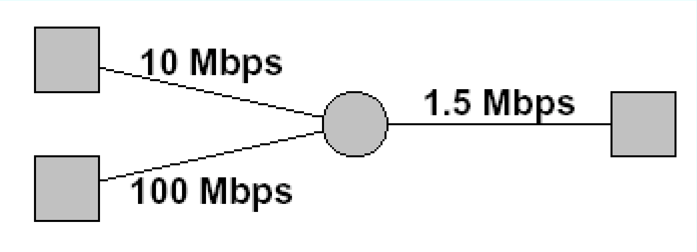
Network Congestion

- **Why does congestion occur?**
  – Routers have finite buffers

- **Buffer is empty:**
  – Small queueing delay in router
- **Buffer is filling:**
  – Longer queueing delays in router
- **Buffer is full:**
  – Packet is dropped

- **Main idea:**
  – Equate packet drops with full buffers, and therefore congestion

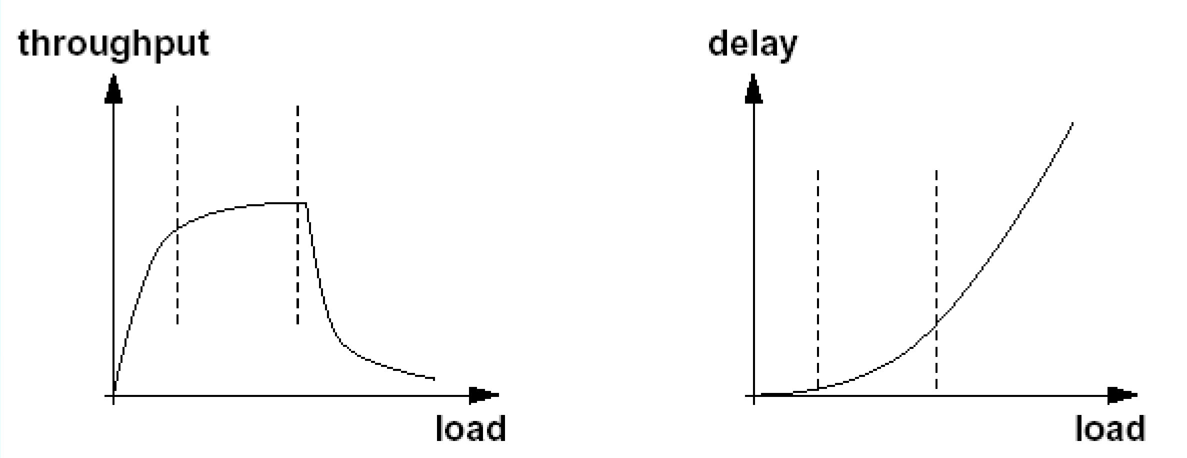
Congestion Collapse



- **Congestion collapse:**
  1. Senders lose data from congestion
  2. Retransmit data
  3. More congestion

Congestion Control and Avoidance

- **A mechanism for:**
  - Using network resources efficiently
  - Preserving fair network resource allocation
  - Preventing and avoiding collapse
- **Network collapse can occur frequently in practice**



Congestion vs. Flow Control

- **Flow control:**
  - Avoids overrunning the receiver
    - wnd
- **Congestion control:**
  - Avoids overrunning router buffers and network
    - cwnd

- **Window to send is min(wnd,cwnd)**

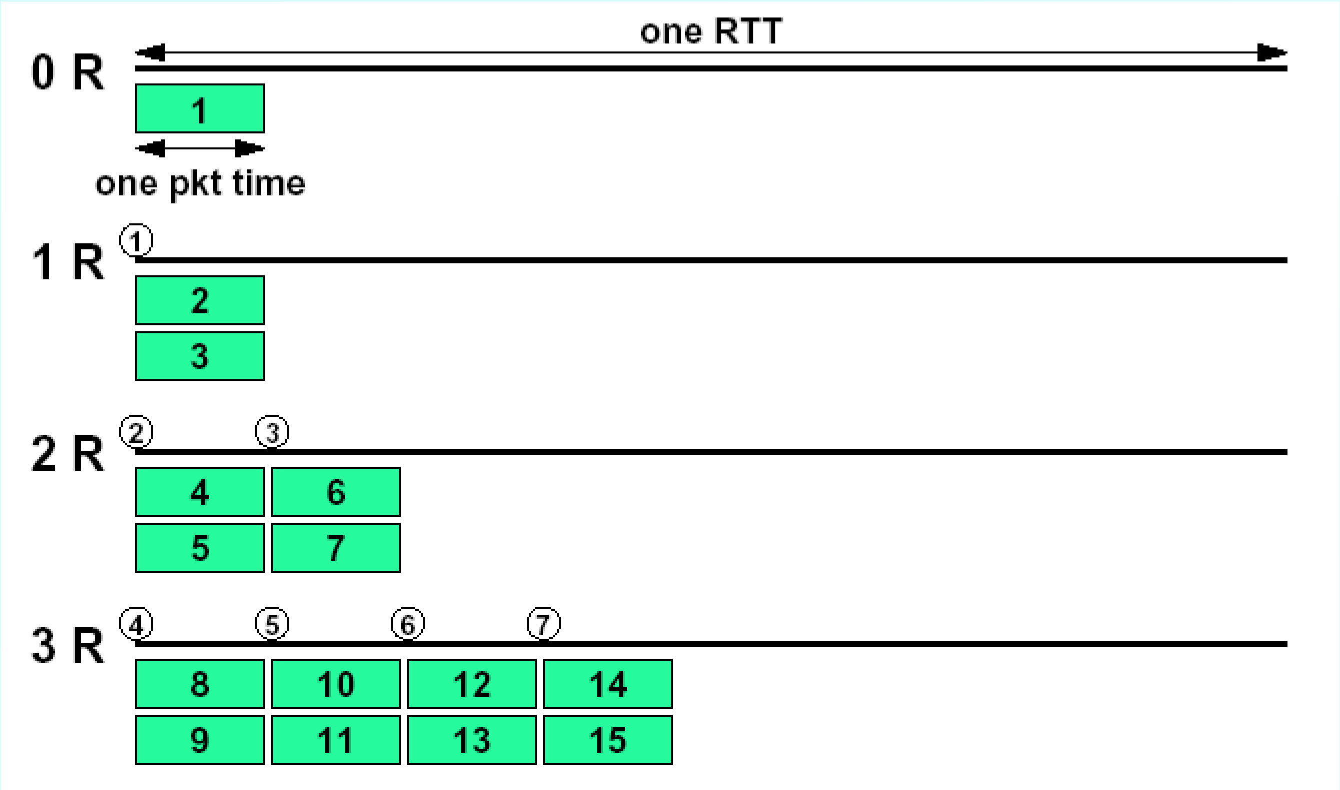
Feedback Control Model

- **Two steps:**
  - Reduce window when congestion is perceived
  - Increase window otherwise
- **Keep a congestion window, cwnd**
- **Sender's maximum window:**
  - Min(advertised_window, cwnd)
- **Sender's actual window:**
  - Max_window - unacknowledged segments

Slow Start

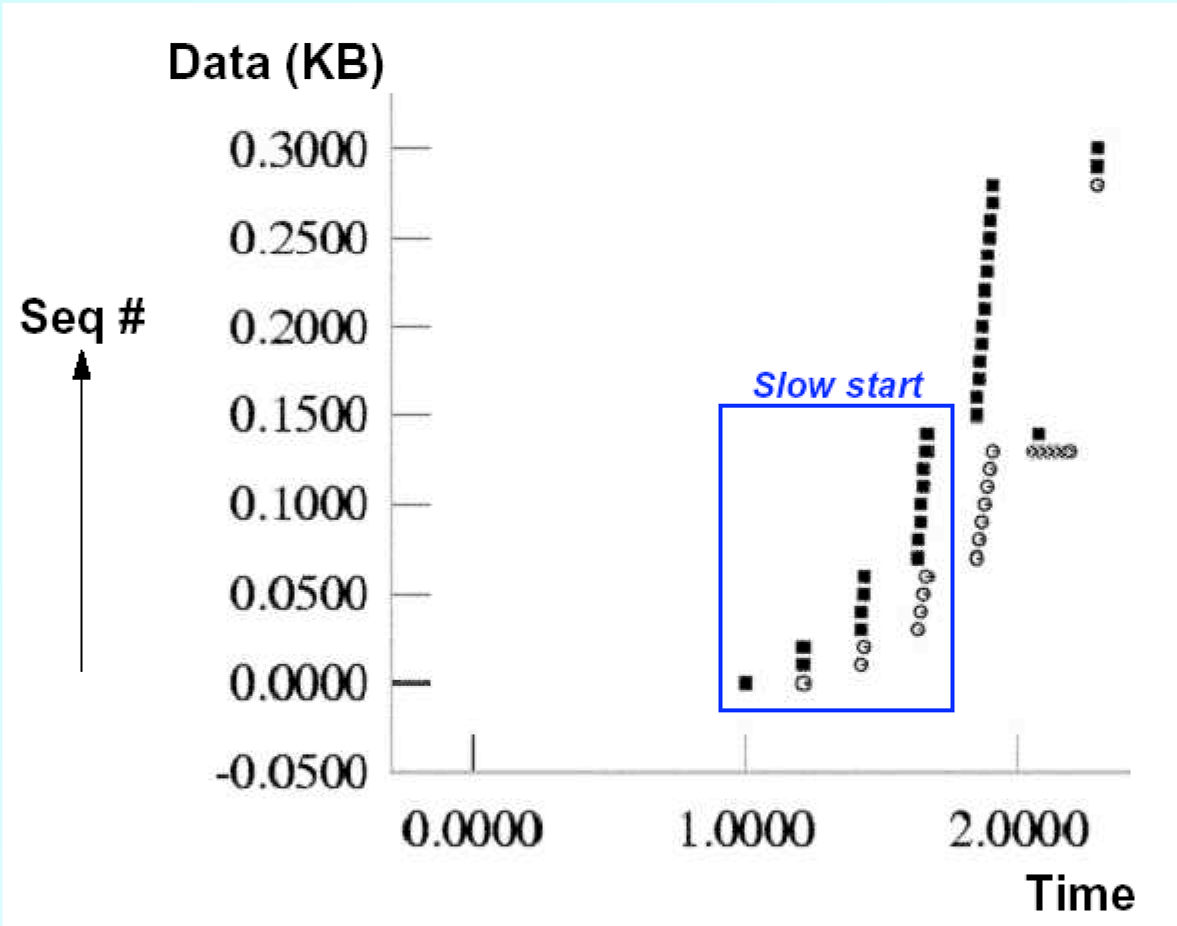
- **Confusing name:**
  - Initialize cwnd=1
  - Upon receipt of every ack, cwnd += 1
- **Implications:**
  - Window size doubles in every RTT
  - Can overshoot window and cause packet loss

Example



As each ACK arrives, 2 packets are generated

Slow Start Sequence Plot



Ending Slow-Start

- **End when the pipe is full**
  - cwnd > ssthresh
  - Start with large ssthresh and then refine it

- **On packet loss:**
  - cwnd=1 and go back to slow-start
  - Ssthresh = cwnd/2
    - Pipe size between last good window (cwnd/2) and current window (cwnd)

Congestion Avoidance

- **If loss occur when cwnd=W**
  - Set cwnd=0.5W (multiplicative decrease)
- **Upon receiving ACK**
  - Increase cwnd by 1/cwnd (additive increase)

- **AIMD: additive increase, multiplicative decrease**

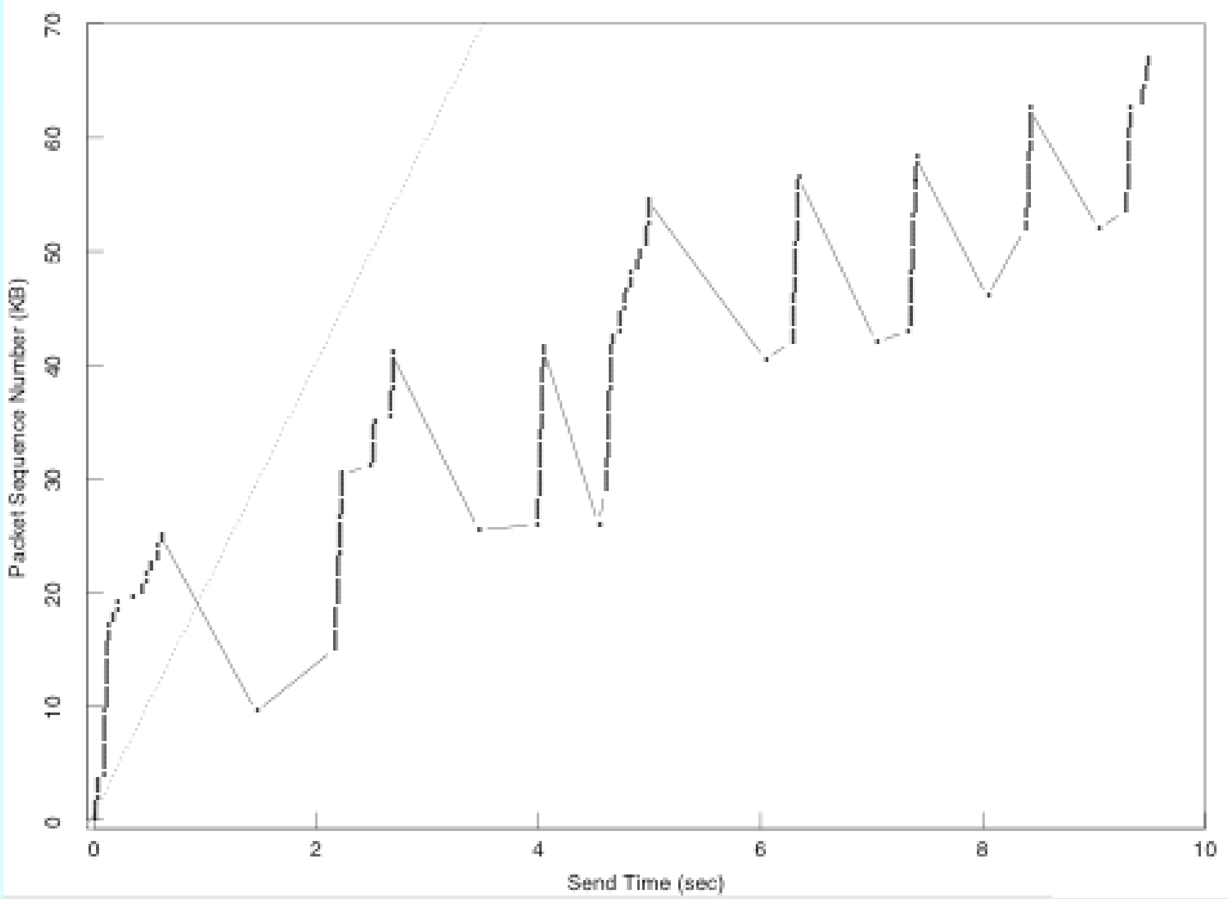
- **Why not multiplicative increase?**

Putting everything together

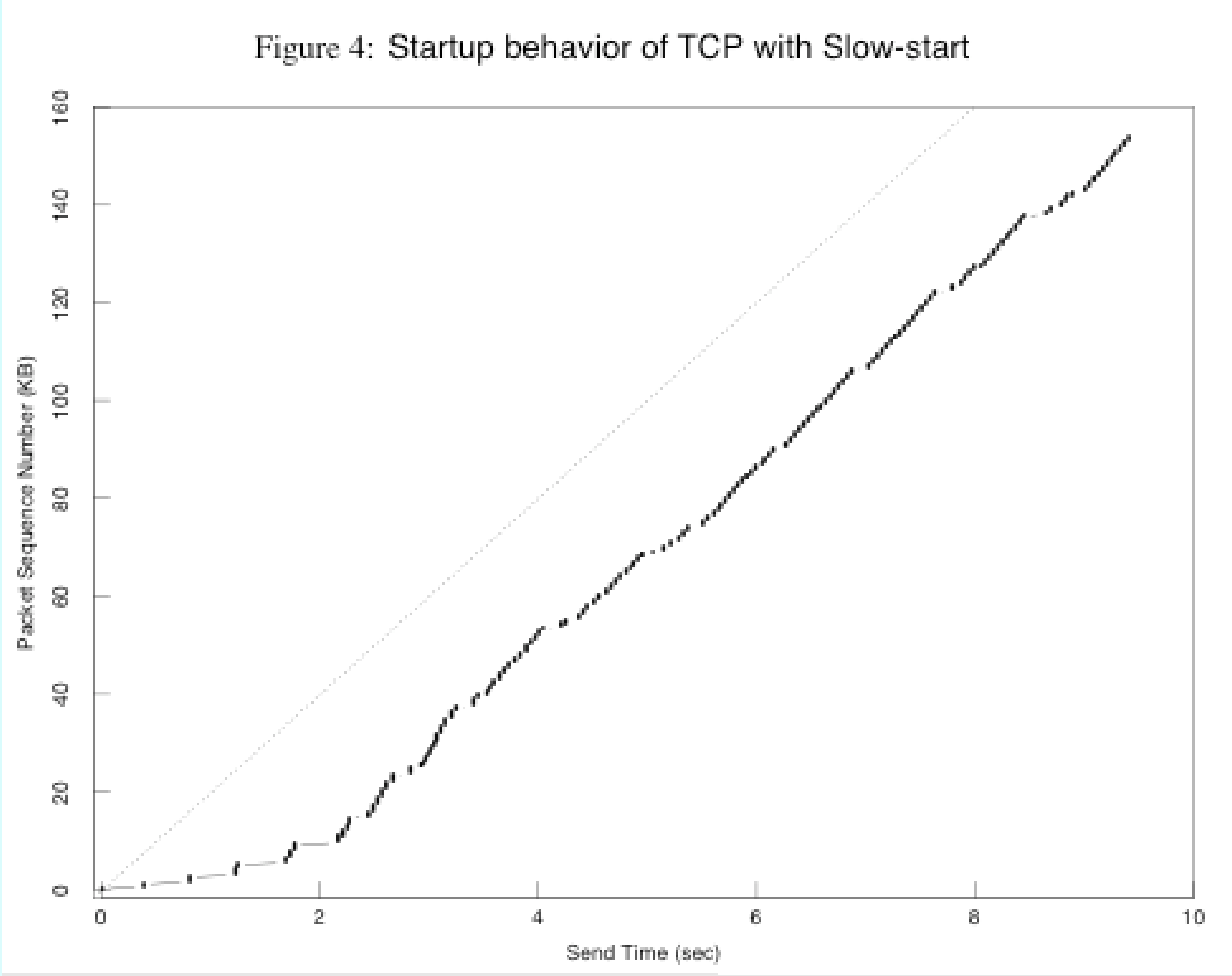
- **When timeout occurs set ssthresh to 0.5w**
  - Set ssthresh to cwnd/2
  - Set cwnd to 1
  - If cwnd < ssthresh, use slow start
  - Else use congestion avoidance

TCP without Slow-Start

Figure 3: Startup behavior of TCP without Slow-start

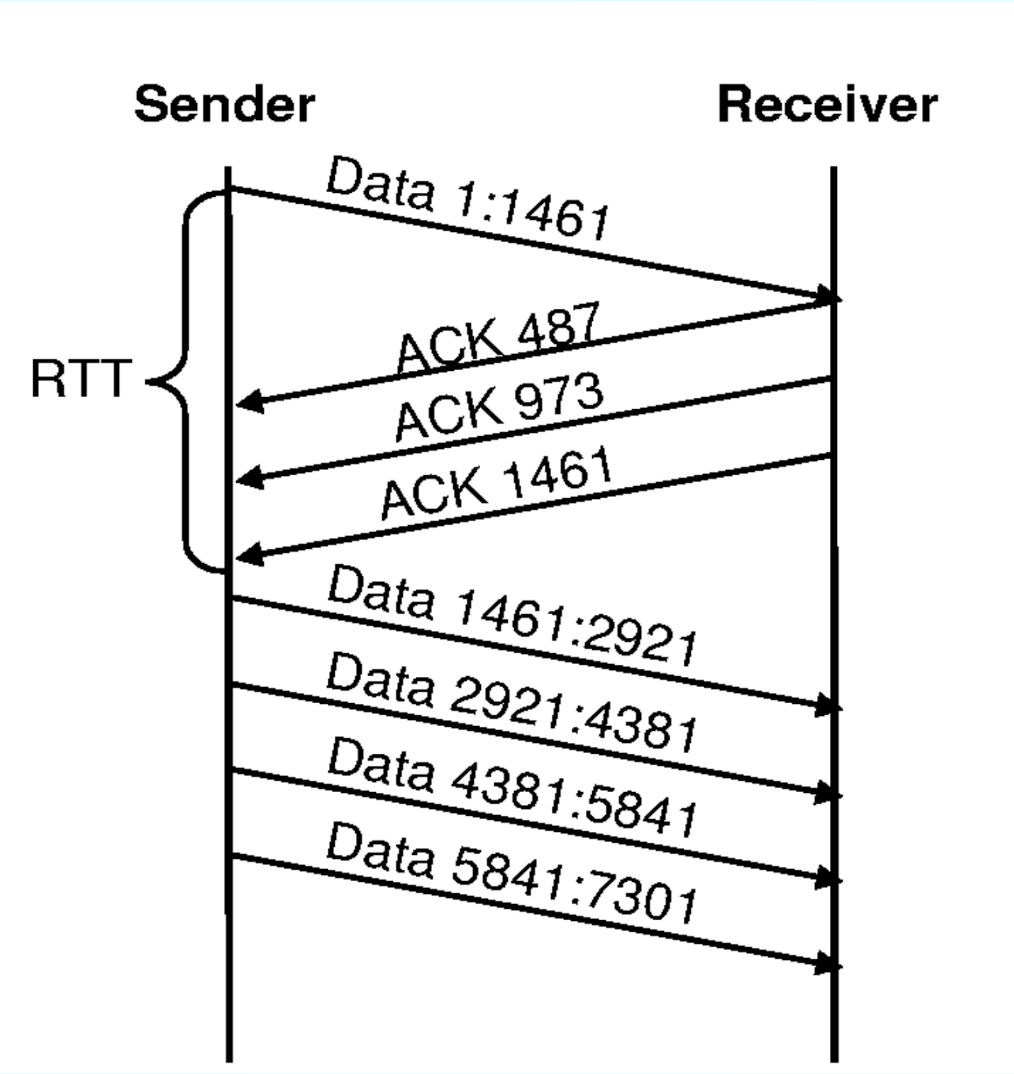


TCP with Slow-Start



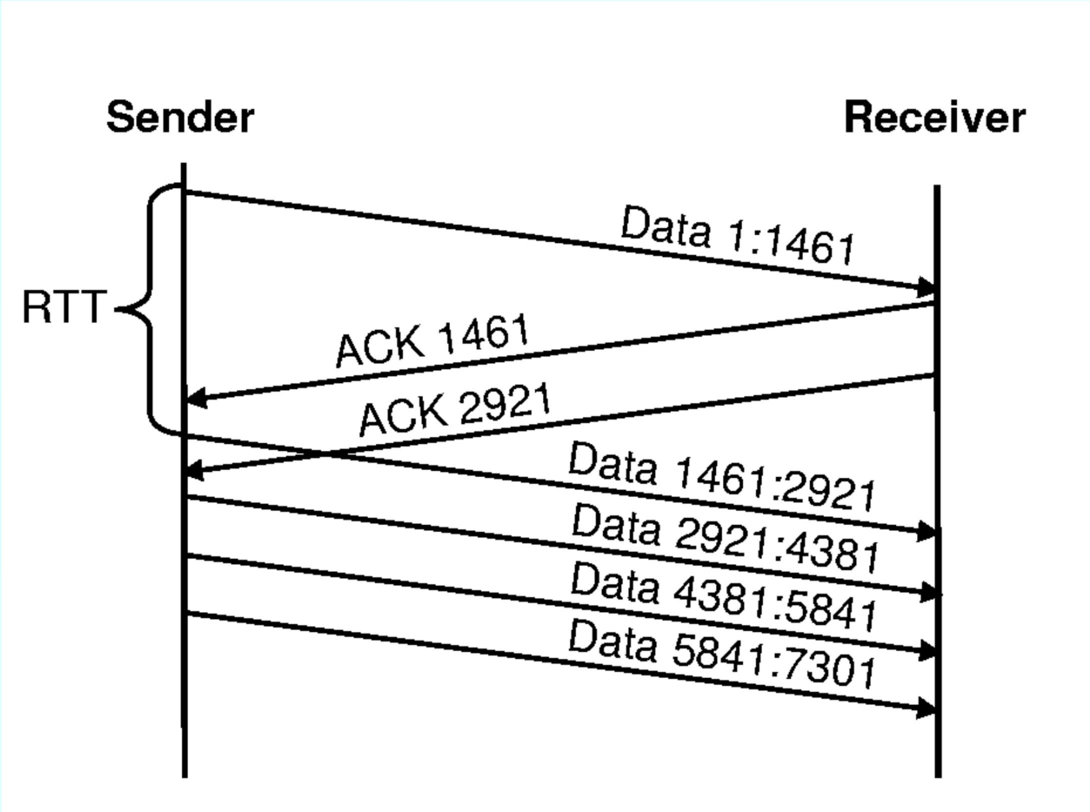
Ack Division

- **Receiver sends multiple, distinct acks for the same data**
- **Max: one for each byte in payload**
- **Smart sender can determine this is wrong**



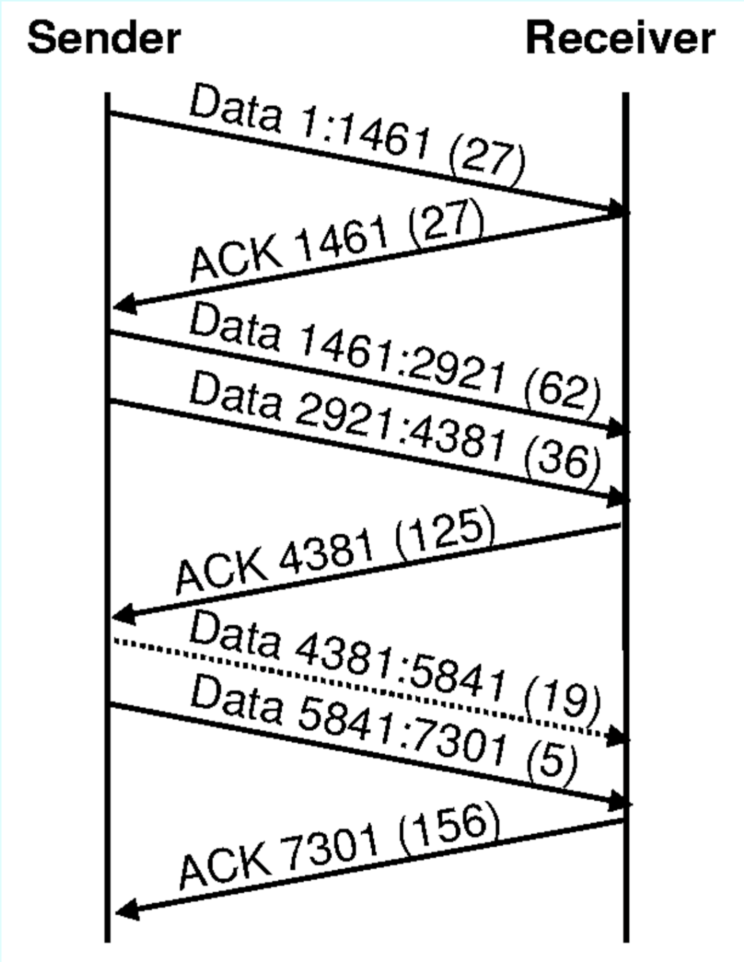
Optimistic Acking

- **Receiver acks data it hasn't received yet**
- **No robust way for sender to detect this on its own**



Solution: Cumulative Nonce

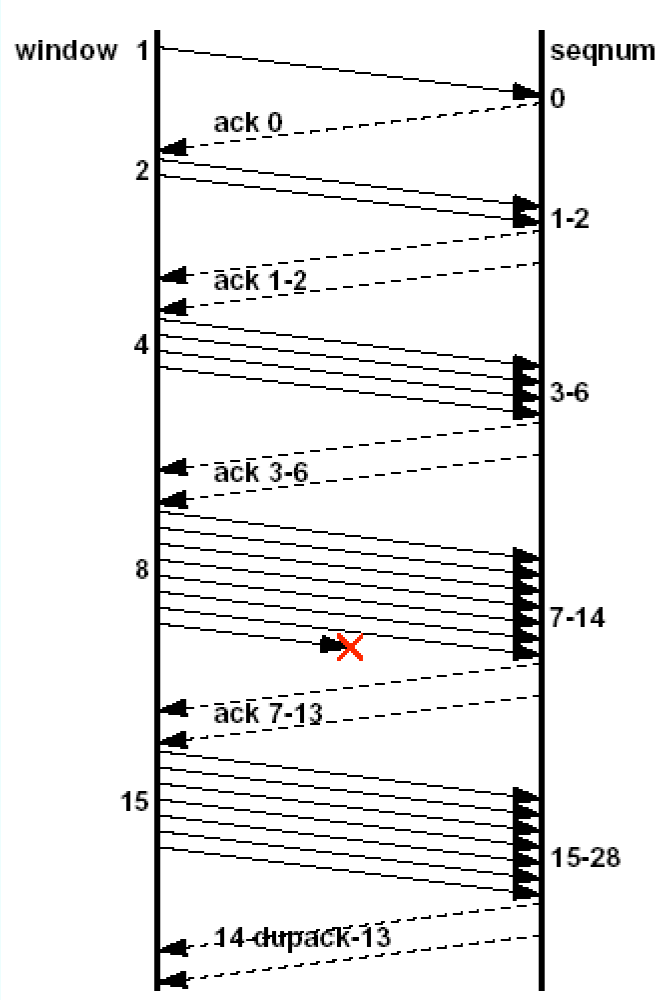
- **Sender sends random number (nonce) with each packet**
- **Receiver sends cumulative sum of nonces**
- **if receiver detects loss, it sends back the last nonce it received**

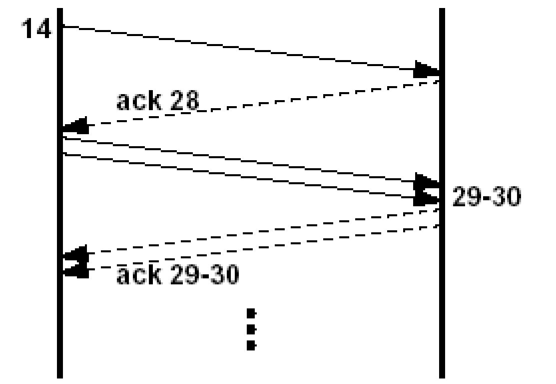


Fast Retransmit

- **When duplicate acks occurs**
  - Loss
  - Packet re-ordering
- **Assume packet re-ordering is infrequent**
  - Use receipt of 3+ dup ACKs are indication of loss
  - Retransmit that segment before timeout
  - Go into slow start when retransmit
  - Resume after

Example





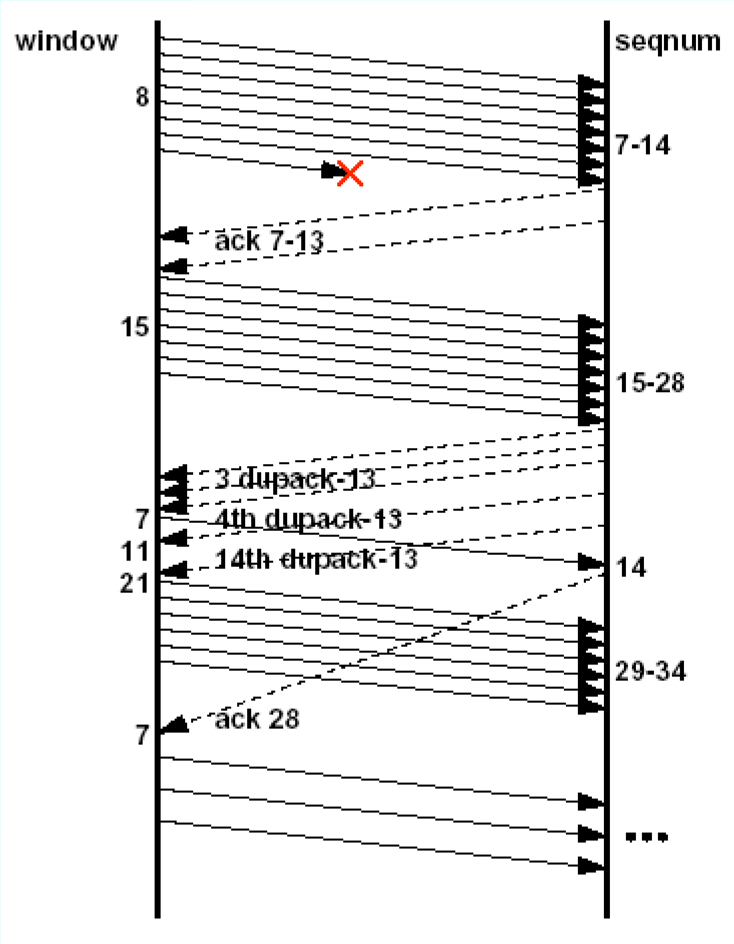
Actions after dupacks for pkt 13:

1. On 3rd dupack 13 enter fast rtx
2. Set ssthresh = 15/2 = 7
3. Set cwnd = 1, retransmit 14
4. Receiver cached 15-28, acks 28
5. cwnd++ continue with slow start
6. At pkt 35 enter congestion avoidance

Fast Recovery

- **In congestion avoidance mode, if duplicate ACKs received, reduce cwnd to half**
- **If n successive duplicate ACKs are received, we know receiver got n segments after lost segment**
  - Advance cwnd by that number

Example



- Action after dupacks for pkt 13:

1. On 3rd dupack 13 enter fast recovery
2. Set ssthresh = cwnd = 15/2 =7
3. Retransmit 14
4. Receipt of 4th dupack set W = 11
5. By 14th dupack, W=21, send 29-34
6. After ack 28, exit fast recovery
7. Set cwnd =7, continue with congestion avoidance

Sting Demo