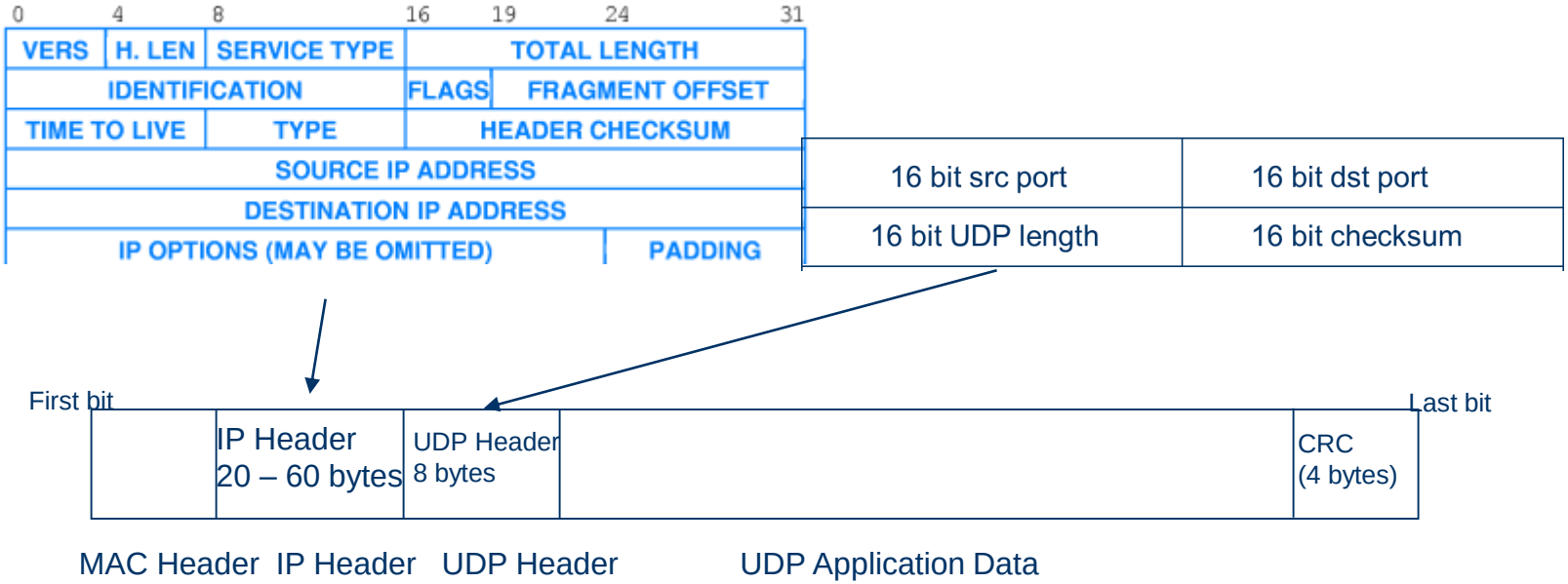
Transport Protocols - functions

- Error detection and error recovery
- Socket abstraction to network convergence layer
  - UDP: sendto maps to a single transport message
  - TCP: socket is a pipe, the application ‘fills the pipe’ with a stream of packets
- End-to-end flow control
  - Handles fast sender and a slow receiver
- Network congestion control
  - Handles reducing the send rate when the network sends congestion indications
  - Reacts to implicit signals of network congestion (packet loss events) or explicit signals of network congestion (Explicit Congestion Notification)

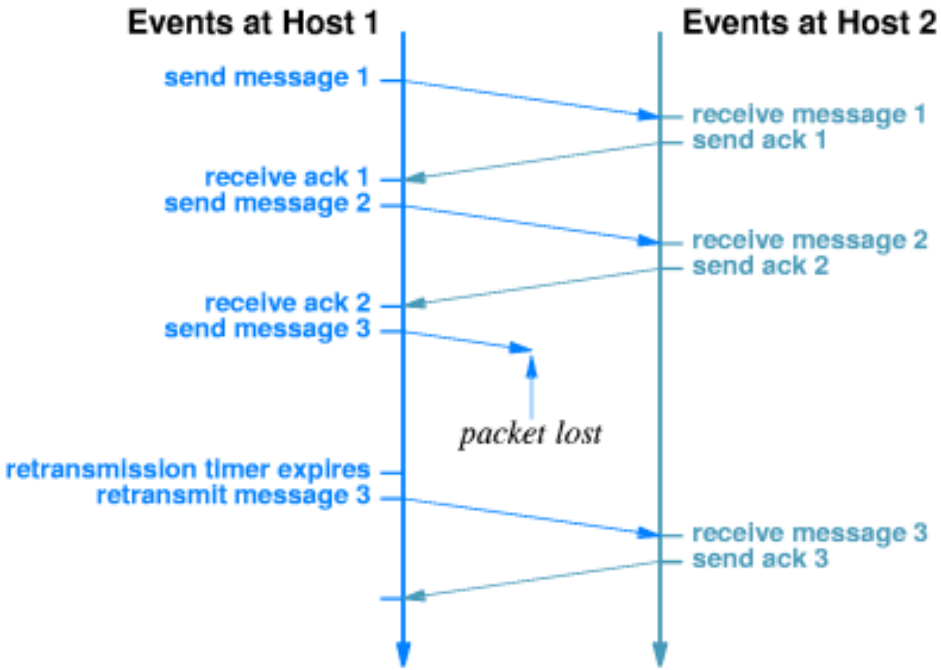
Error Detection: Example UDP



Bit or byte error detection typically performed multiple times:

- Link layer- a MAC level Cyclic redundancy check
- Network layer – At each router, IP performs a checksum of JUST the header bytes
- End-to-end: TCP and optionally UDP performs the same checksum on header and application data.

Packet Loss …. Retransmission

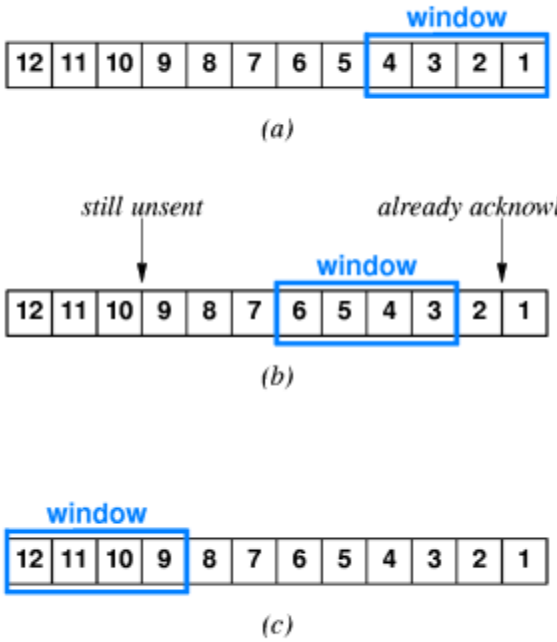


Example of retransmission. Items on the left correspond to events in a computer sending data, items on the right correspond to events in a computer receiving data, and time goes down the figure. The sender retransmits lost data.

Error Recovery

- Two methods
  - Error correcting codes –
    - Redundant information is sent such that is a bit, byte, or entire frame is lost, the receiving can potentially recover the original message
  - Retransmissions -
    - ARQ: Automatic Repeat ReQuest protocols use acknowledgement packets from the receiver to instruct the sender what data has been correctly received (or not received)
    - Combined client sending behavior with a window – sliding window ARQ
      - A sender has a stream of data, sends portions of the stream in segments, assigns each segment a sequence number.
      - The sender has a 'Window' that controls how many segments can be 'in the pipe' before the sender must stop sending and wait for the next ACK packet.
      - Once an ACK arrives, the sender's window is incremented allowing it to send new data
      - Stop and wait - assumes the Window size is 1 segment

Sliding Window Protocol



1. A 4-segment window sliding through an outgoing stream of data
2. In the figure, ACKs for segments 1,2 arrive and slide the window forward
3. Usually the data retransmission method is intertwined with the decision of how and when to send new data- E.g., TCP ACKs that arrive at a sender clock out new segment transmissions

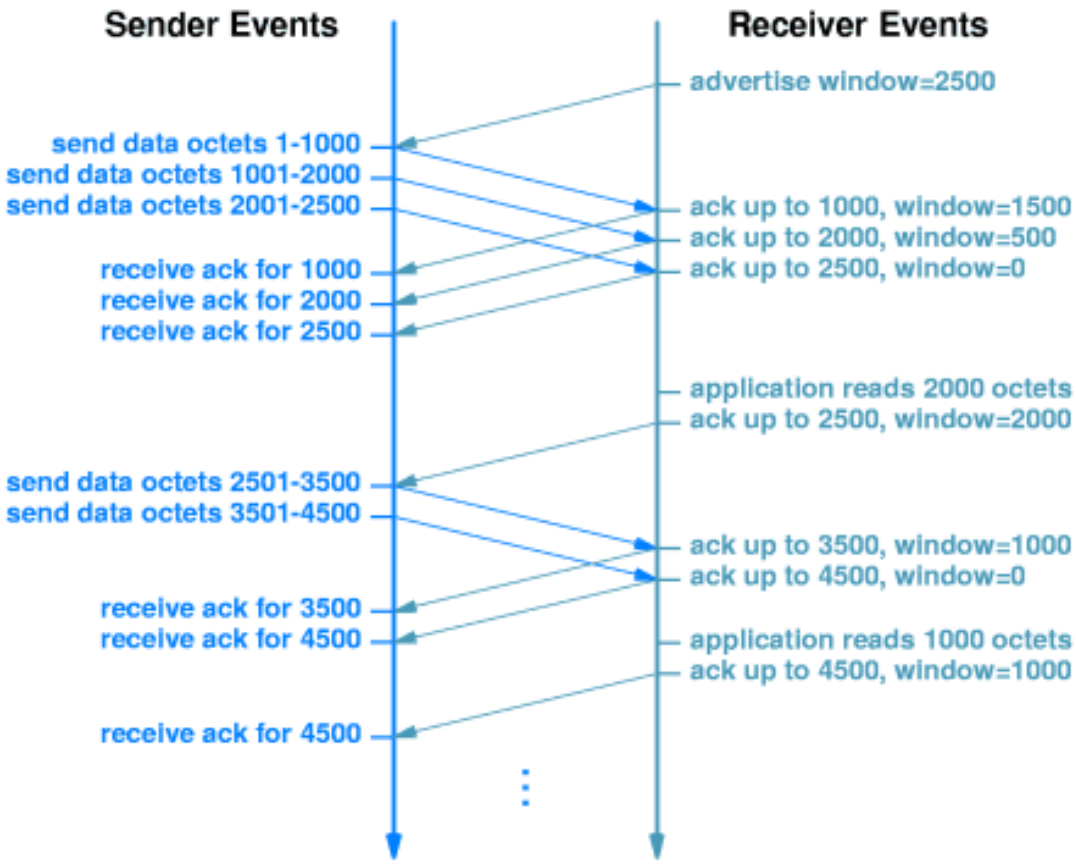
- Packet loss details: Client Retransmission strategies
  - Go-back-N – looking at figure b, if any packet in the window is dropped, the sender retransmits the entire window. Simple but not efficient!
  - Selective retransmit – the ACKs tell what has been received and what has not been received. The sender retransmits only what was dropped

- Packet loss details: Server strategy when detects an out-of0order segment:
  - Just ACK the segment that arrives?
  - Just ACK the segment that arrives AND indicate the previous segments missing?
  - Just ACK the segment next expected for in order arrivals?
    - Duplicate ACKS!!!

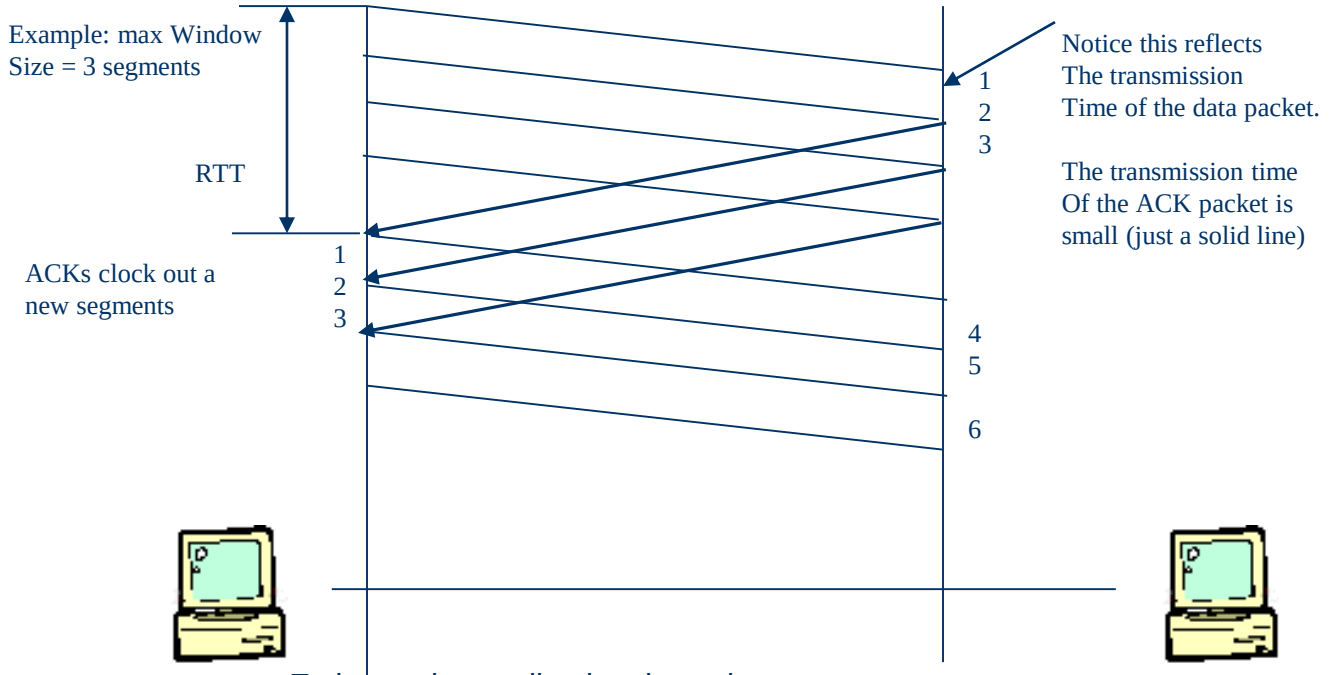
Flow Control

- Concept is to help manage buffer space in a network
- Can operate at many locations/levels –
  - Link level – for example an Ethernet Link and Phy layer can issue flow control feedback to a sender to limit transmissions
  - Network level - can limit new connections if the network does not have sufficient resources for a new flow
  - End-to-end : A socket receiver might run out of buffer space and needs to limit the sending side (think: fast send application and a slow receive side application)
- As an example, TCP's flow control is:
  - Based on the available receive buffer space at a receiving side of a socket
  - As data arrives (and fills the buffer), the receiver sends ACKs that also specify the remaining buffer size.
  - Details: TCP includes in every segment the receiver's 'RxAdvertisedWindow' - this simply is the amount of space left in the receiver's Rx buffer.

Flow Control - Example



Sliding Window Protocol - Example



Estimate the application throughput
=amount of data sent / cycle

- Throughput = Window (in bits) / RTT (seconds)
- Max Window = 3 segments, Segment Size: 1000 bytes, RTT of 60ms
- Throughput=3*1000* 8 / 0.060 = 400Kbps

Network Performance Characteristics

- Bandwidth-delay product (BDP)
  - The product of the bw * delay measures the volume of data that can be in the network.
    - POINT: The optimal Window size is based on the BDP
- Example: Two hosts connected by a link that has a data rate of 1.5 Mbps. The uncongested RTT is 60 ms, packets are 1500 bytes.
  - The pipe size in packets:
    - Pipe= (1,500,000 * 0.060) / (1500*8) packets = 7.5 packets
  - This is the best window size!!!!
    - What is the sender sends with a max window of 10 packets?
    - Assuming the RTT remains at 60ms, the sending rate is : 10*1500*8/0.060 = 2,000, 0000 bps.
    - What will happen?
      - Packets from H1 arrive at R1 at a rate of 2 Mbps
      - Packets depart from R1 at a rate of 1.5 Mbps
      - Packets will be queued at R1's output interface buffer …. And eventually the buffer will overflow causing packet loss

H1 ----- R1 ----- ---- R2 ----- H2
1 Gbps 1.5 Mbps 1 Gbps

Network Congestion Control

•When the network gets congested …..
- •Router 1 might be connected to > 2 networks…..
- •The level of aggregate traffic over a particular router's link is very likely to exceed the link data rate
- •Network congestion is when the average queue level at a router's interface is non zero for long time scales (perhaps minutes or hours)
- •Symptoms: An application might experience random packet drops and inflated RTTs – both caused by the queue buildup

H1 ----- R1 ----- ---- R2 ----- H2
1 Gbps 1.5 Mbps 1 Gbps

•Network congestion control (focusing on Internet) sends congestion notifications to endpoints in hope the senders are 'well behaved' and reduce their sending rate.

•TCP's base congestion control algorithm (Slow-Start) based on a dynamic window (slow start)
•It integrates flow control but we will ignore that…. So this is done at the send side whenever an ACK arrives
- •Init:
  - •cwnd = 1; //init the congestion window to 1 segment
  - •maxWindow = MAX; //set the max allowed to a static number of segments
- •wnd = min(cwnd, Maximum Allowed Window) //The actual window used…
- •On a timeout, cwnd = 1
- •When a new ACK arrives
  - •cwnd += 1;