University of Massachusetts Amherst

ECE697AA – Lecture 3

Transport Layer: TCP and UDP

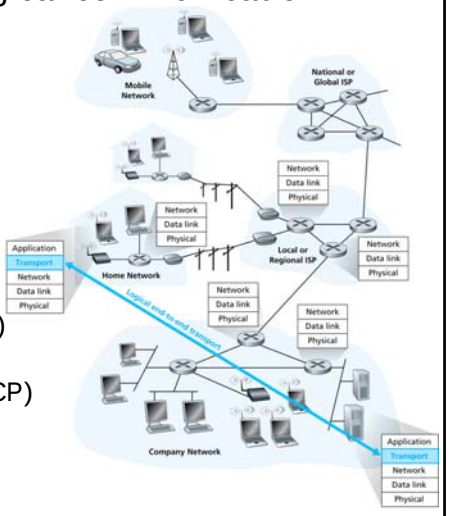
Tilman Wolf
Department of Electrical and Computer Engineering
09/09/08

Today's lecture

- Transport layer
  - User datagram protocol (UDP)
  - Reliable data transfer
  - Transport control protocol (TCP)
- We ignore for now
  - Flow control (part of TCP)
  - Congestion control (part of TCP)

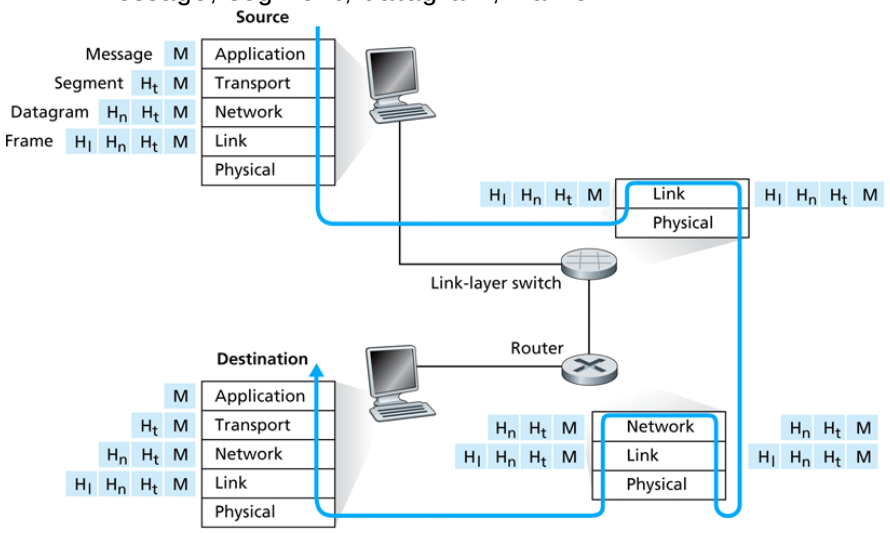
Transport Layer

- Transport layer provides logical communication
  - Service to layer above
    - » Connects applications
  - Uses network layer
    - » Data is "segmented" and encapsulated in packet
- Services
  - Transport-layer multiplexing and demultiplexing
    - » Delivery to correct process
  - User Datagram Protocol (UDP)
    - » Send and hope for the best
  - Transport Control Protocol (TCP)
    - » Reliability
    - » Flow control
    - » Congestion control



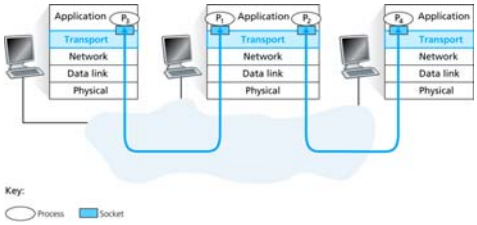
Layered protocol stack

- Note naming of data in different layers
  - Message, segment, datagram, frame



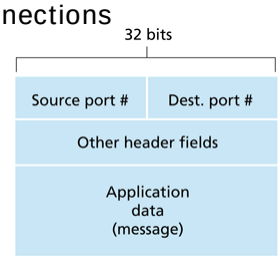
De-/multiplexing

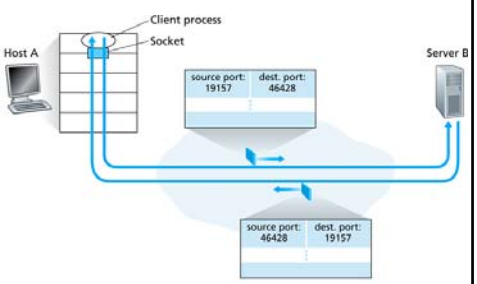
- Why do we need this feature in transport layer?



De-/multiplexing

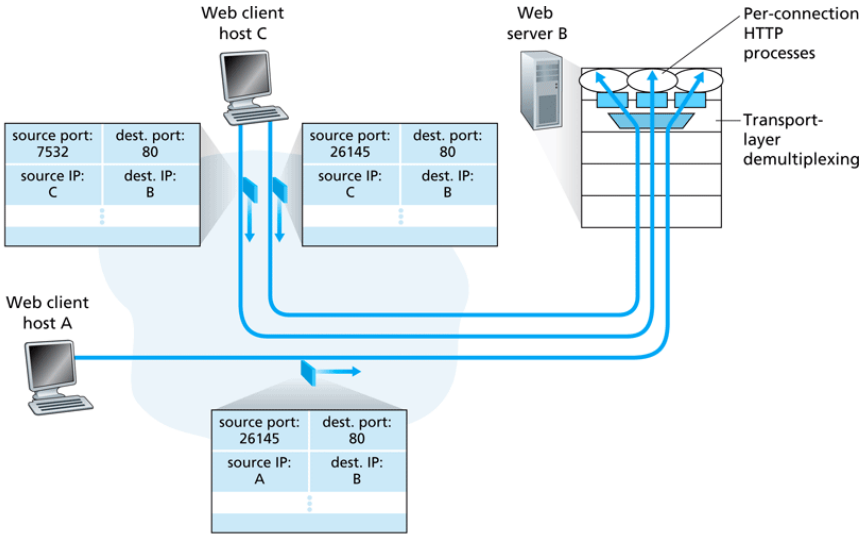
- Multiple processes on one host use network
  - Need to distinguish different active connections
- "Port number" identifies process
  - 16-bit field
  - Destination port depends on
  - application layer service
  - Source port (usually) randomly chosen
- Port-pair used for bidirectional transmission
  - Source and destination swappedon reverse path





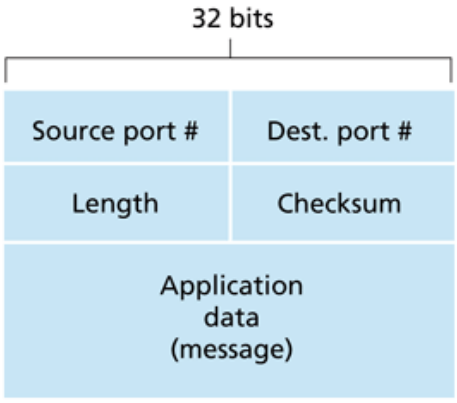
De-/multiplexing

- Works even if multiple connections go to same port



User Datagram Protocol

- Bare-bones transport protocol
  - Connectionless, state-free
  - Unreliable (just like IP)
  - Low overhead
- UDP segment
  - Port numbers
  - Length field
  - Checksum field (optional!)
- UDP couldn't do less...



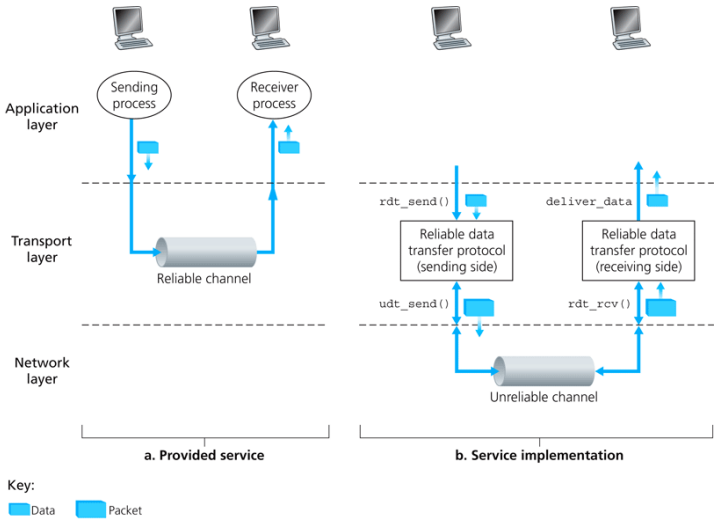
Reliable data transfer

- What can happen that causes unreliable transfer?

- How can we ensure reliability?

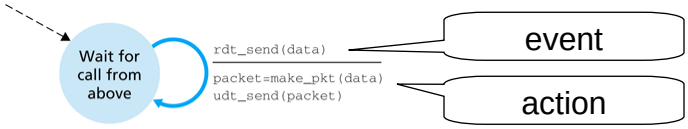
Reliable data transfer

- Reliability is provided on top of unreliable channel

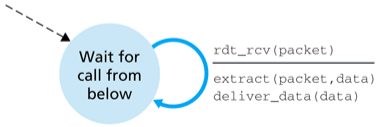


Reliable data transfer

- Build protocol for incrementally worse system
- Scenario 1: completely reliable channel



a. rdt1.0: sending side

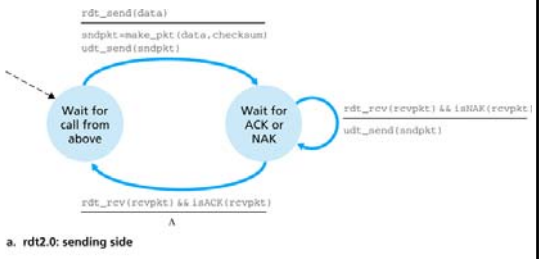


b. rdt1.0: receiving side

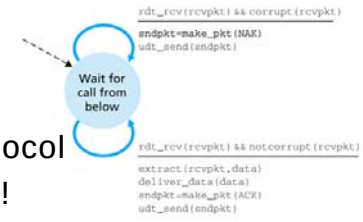
  - Next step: bit errors

Reliable data transfer

- Scenario 2: channel with bit errors
  - Packets show up, but might have errors
- What extra functionality do we need?
  - Error detection
  - Receiver feedback
  - Retransmission
- Sender reacts to ACK/NAK
- "Stop-and-wait" protocol
- What's the problem?!



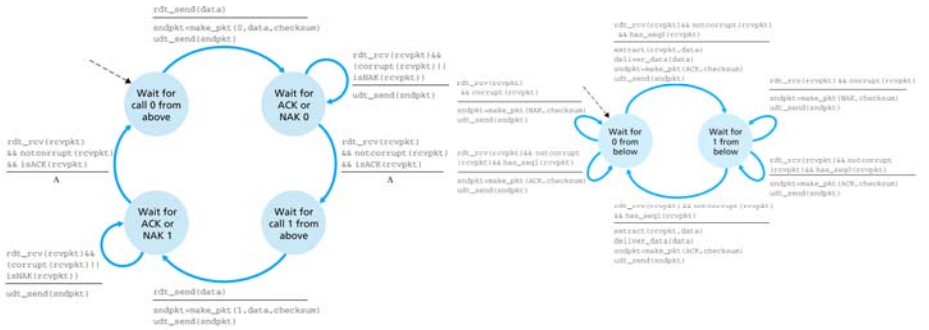
a. rdt2.0: sending side



b. rdt2.0: receiving side

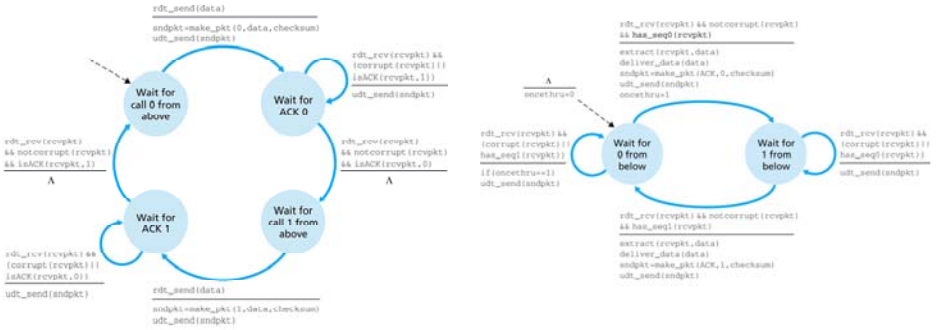
Reliable data transfer

- What if data is retransmitted on garbled ACK/NAK?
  - Potentially duplicate packets!
- Add sequence number to packet
  - How many sequence numbers do we need?



Reliable data transfer

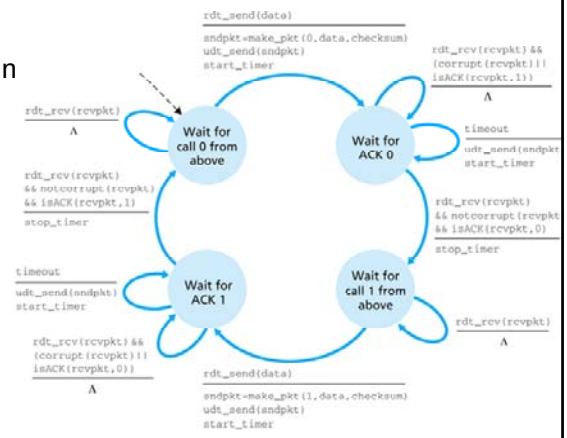
- Use only ACK instead of ACK/NAK
  - ACK requires sequence number



- Next step: packet loss

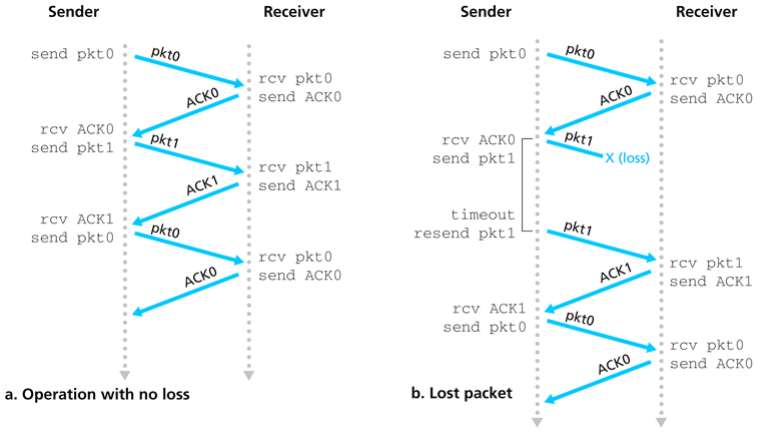
Reliable data transfer

- Scenario 3: channel with packet loss and bit errors
- What is the problem?
  - If packets get lost, transfer gets stuck
- Timer necessary
  - Starts on transmission
  - Retransmission on expiration
  - Sequence bit maintains order
- "Alternating-bit protocol"



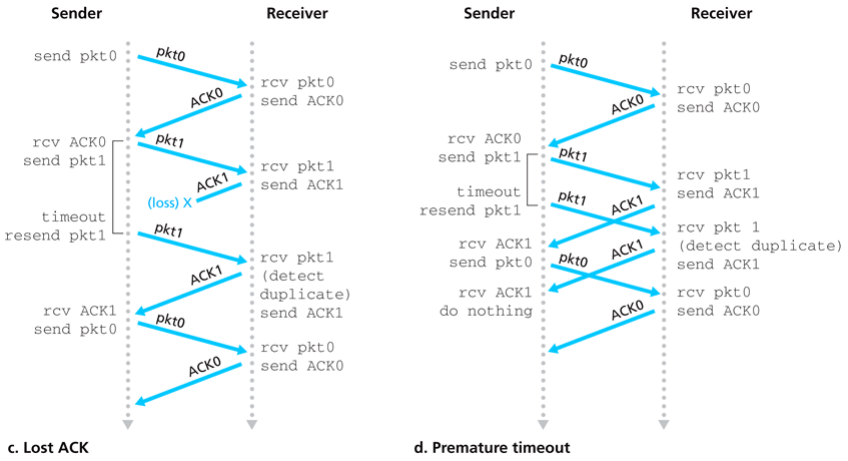
Alternating-bit protocol

- Error scenarios are handled correctly



Alternating-bit protocol

- Error scenarios are handled correctly



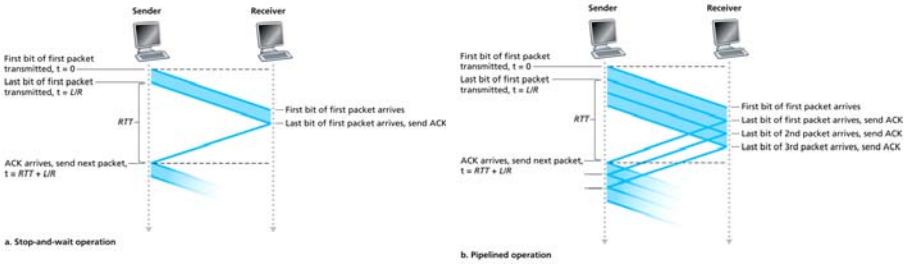
Protocol performance

- What is the maximum throughput that can be achieved with this protocol?

- How can it be improved?

Pipelined transfer

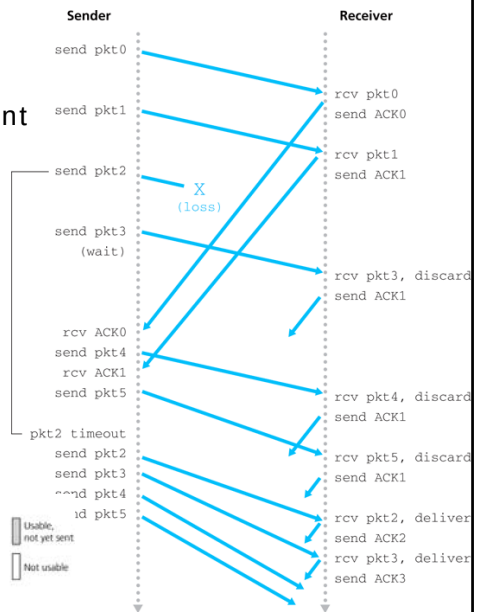
- Multiple packets can be in-flight
  - Sliding window protocol

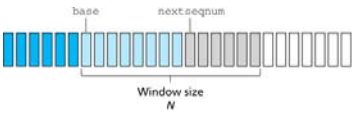


- What needs to change?
  - More sequence numbers
  - Larger buffers (retransmission or reassembly)
- Two versions:
  - Go-Back-n
  - Selective Repeat

Go-back-N

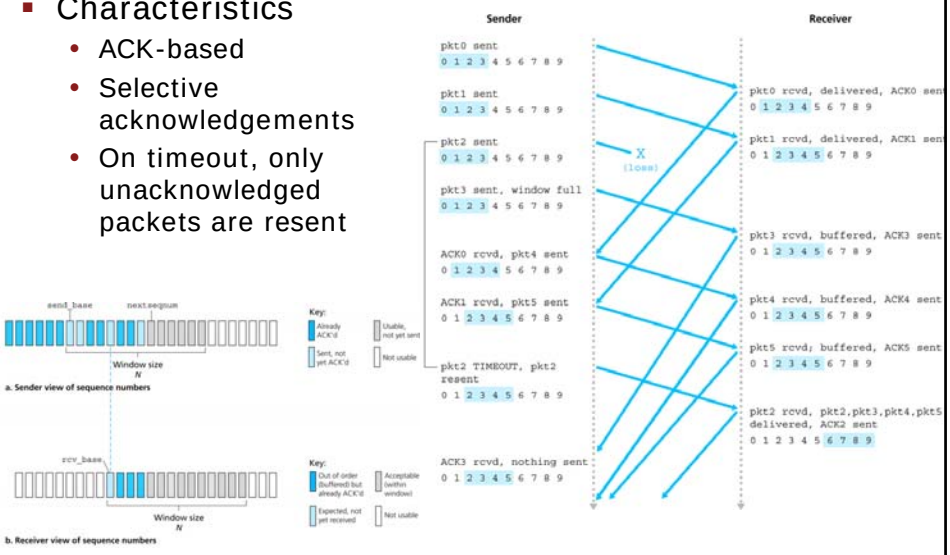
- Characteristics
  - ACK-based
  - Cumulative acknowledgement
  - On timeout, all outstanding packets are resent





Selective repeat

- Characteristics
  - ACK-based
  - Selective acknowledgements
  - On timeout, only unacknowledged packets are resent

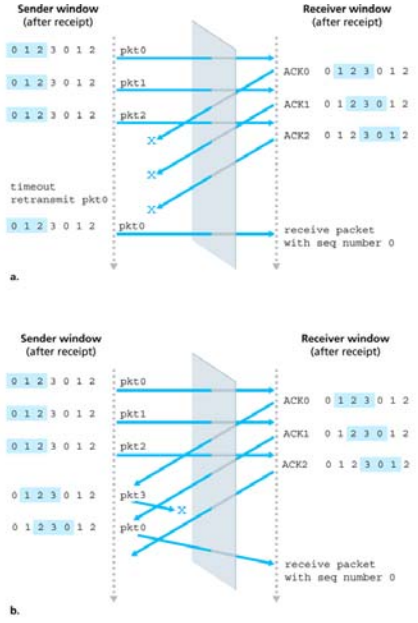


Applets

- Example of go-back-N operation:
  - http://media.pearsoncmg.com/aw/aw_kurose_network_4/applets/go-back-n/index.html
- Example of selective repeat operation:
  - http://media.pearsoncmg.com/aw/aw_kurose_network_4/applets/SR/index.html

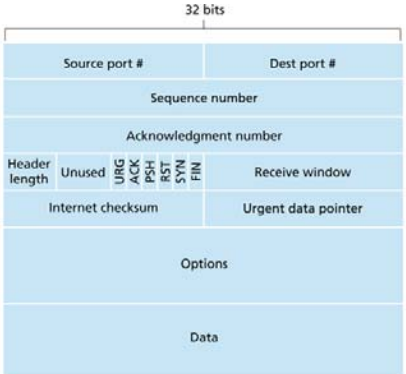
Sequence numbers

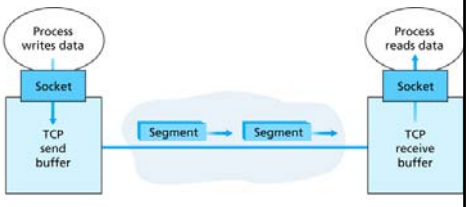
- Sequence numbers are finite
  - Need to be carried in each packet
- Problem if window is too large
  - Packets with reused sequence numbers cannot be distinguished from originals
- Problem if indefinite packet lifetime is allowed
  - New packet cannot be distinguished from retransmission
  - Internet: ~3 minutes max



Transmission Control Protocol

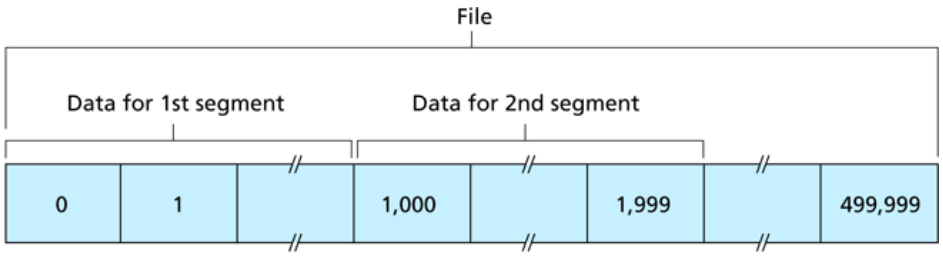
- Implementation of sliding window protocol
  - Cumulative ACKs
  - ACKs are piggy-backed on data packets in other direction
  - If out-of-order segment
    - » Option 1: discard
    - » Option 2: wait and see if other segments show up



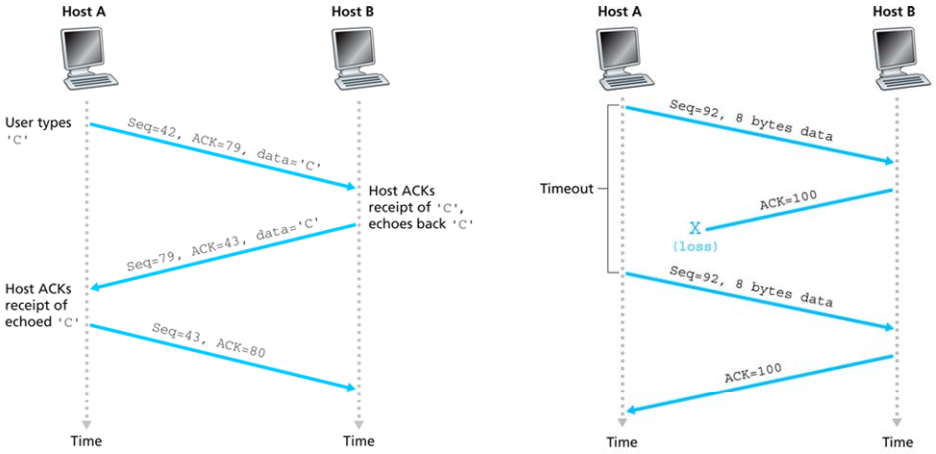


TCP sequence numbers

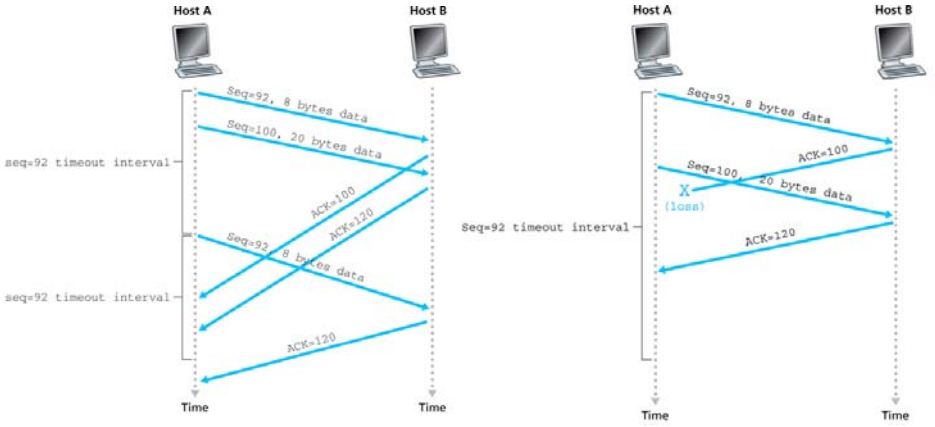
- Sequence numbers are position in byte-stream
  - 16bits = 64kByte window



TCP operation

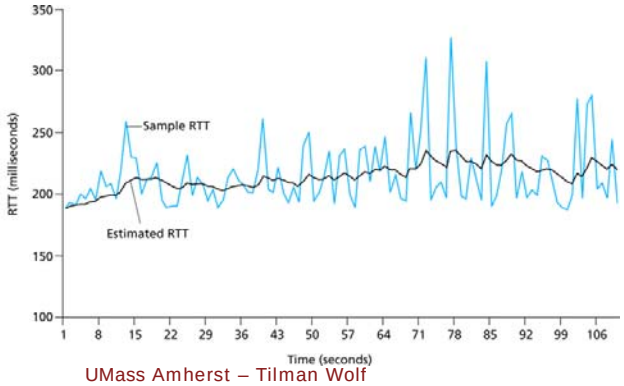


TCP operation



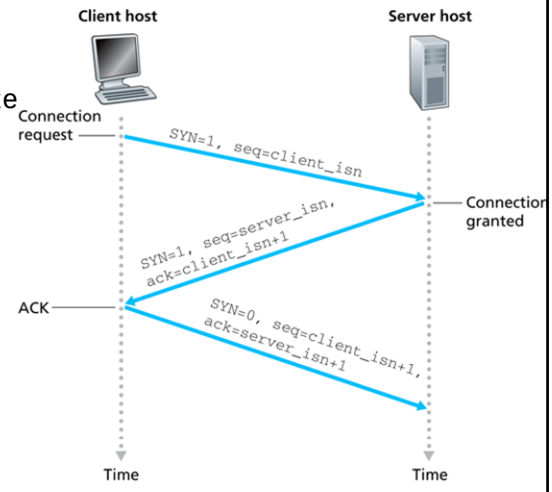
Round trip time

- RTT estimate important for efficient operation
  - Too long: long delay before retransmission
  - Too short: unnecessary retransmission
- TCP RTT estimation
  - TCP keeps exponential weighted moving average of RTT
  - Timeout is set to estimation+4×deviation
- Example:

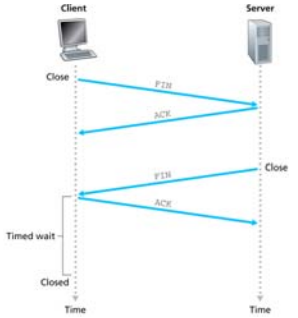


TCP connection management

- Connection setup
  - Three-way handshake
  - SYN counts as one byte



- Connection teardown



Assignments

- Read
  - Kurose & Ross: Chapter 4
    - » 4.3, 4.6, and 4.7 not necessary – will be discussed later
- SPARK
  - Assessment quiz