UMASS AMHERST

ECE 671
Introduction to Computer Networks

Lesson 4
Transport Layer & Reliability

Rationale

- Transport layer is an important part of Internet protocol stack
- Understanding complete protocol stack is important
- Transport layer services are used by application layer
  - Even if someone does not care about networking details, understanding which transport layer protocol to use (and why) is really important
- Reliability achieved in transport layer is a really “cool” example of how protocol design can achieve surprising functionality

Objectives

- Defend the importance of content distribution networks (CDNs)
- Analyze necessity of de-/multiplexing as a transport layer feature
- Assess the impact of channel reliability on data transfer outcomes
- Differentiate UDP and TCP functionality in the transport layer

Prior Knowledge

- Application layer
  - Connects distributed applications
  - Relies on transport layer service
- Transport layer connect application processes
  - Different transport layer protocols depending on requirements

Orchestrated Discussion (Hand Raise): Lesson Reflection Feedback

- Discuss questions and comments on Lesson Reflection from prior lesson

Student Presentation: Content Distribution Networks (CDNs)

- Case Study: Content Distribution Networks (CDNs)
  - Work in 3-person teams
- Prepare 5-minute presentation
  - Why do we need (CDNs)
  - How do CDNs work
  - Where do we seen CDNs in practice
- One team will be selected at random to present at beginning of next lecture

Content Distribution Network

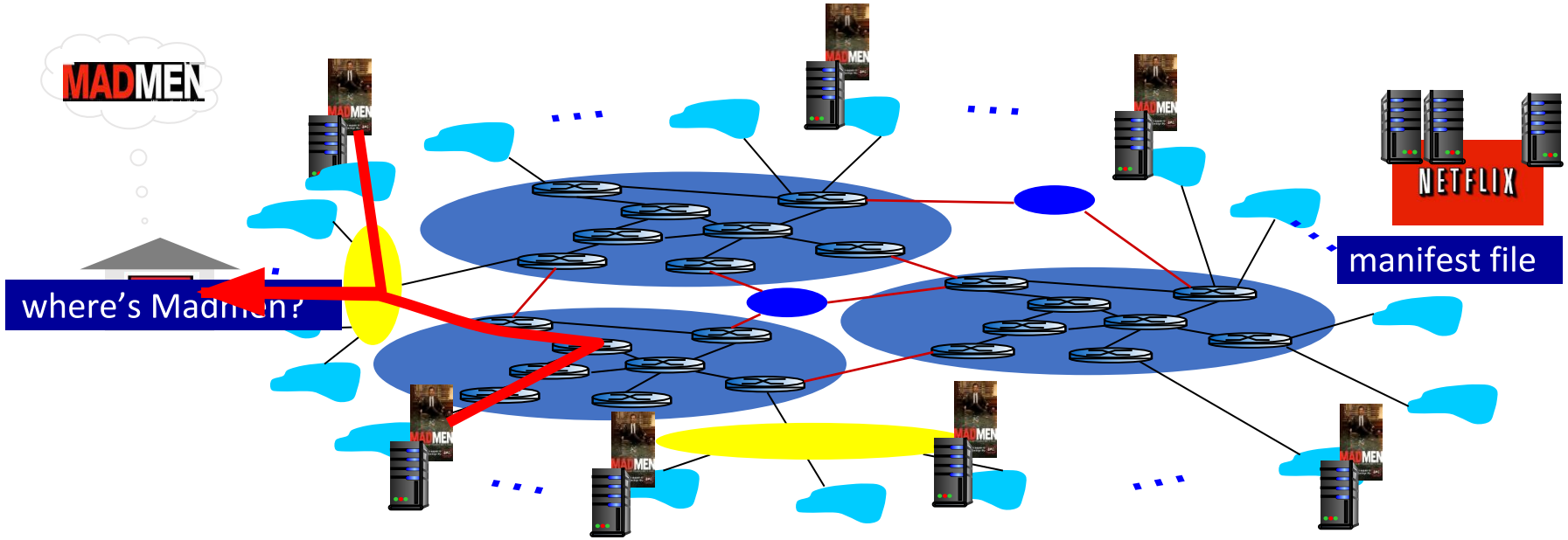
- Challenge: How to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users
- Option 1: single, large “mega-server”
  - Single point of failure
  - Point of network congestion
  - Long path to distant clients
  - Multiple copies of video sent over outgoing link
- This solution doesn’t scale

Content Distribution Network

- Challenge: How to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users
- Option 2: Store/serve multiple copies of videos at multiple geographically distributed sites (CDN)
  - Enter deep: push CDN servers deep into many access networks
    - Close to users
    - Used by Akamai, 1700 locations
  - Bring home: smaller number (10's) of larger clusters in points of presence (POPs) near (but not within) access networks
    - Used by Limelight

Content Distribution Network

- CDN: Stores copies of content at CDN nodes
  - e.g. Netflix stores copies of MadMen TV show
- Subscriber requests content from CDN
  - Directed to nearby copy, retrieves content
  - May choose different copy if network path congested



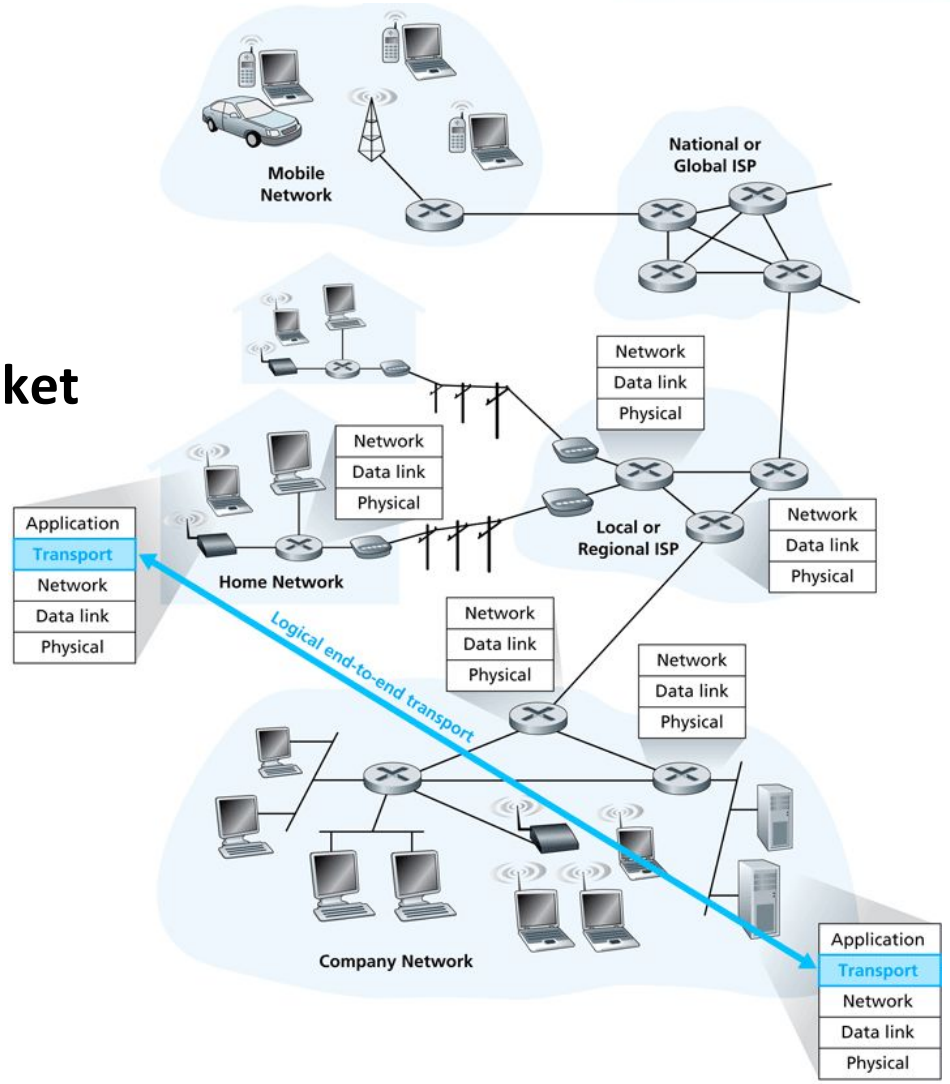
Content Distribution Network

OTT challenges: **Coping with a congested Internet**

- **From which CDN node to retrieve content?**
- **Viewer behavior in presence of congestion?**
- **What content to place in which CDN node?**

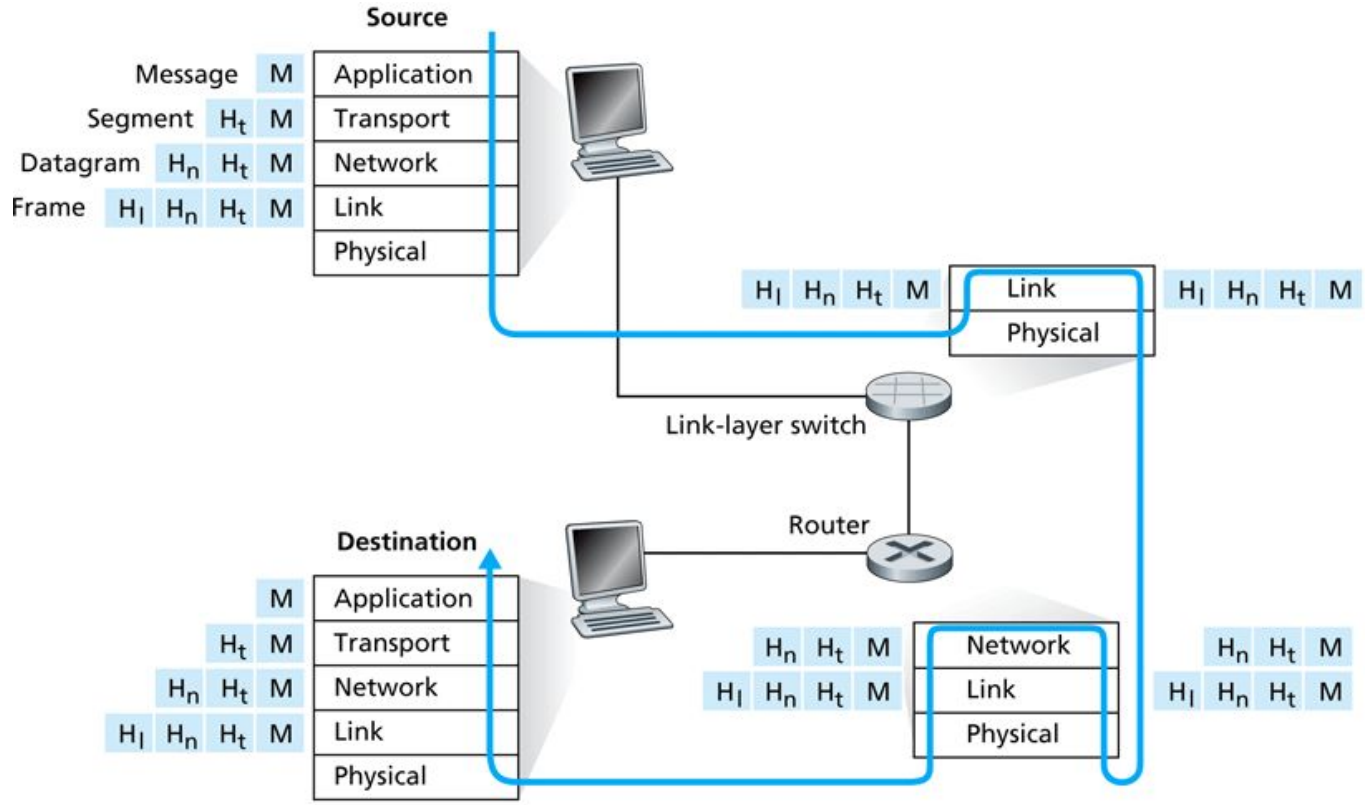
Transport Layer

- 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



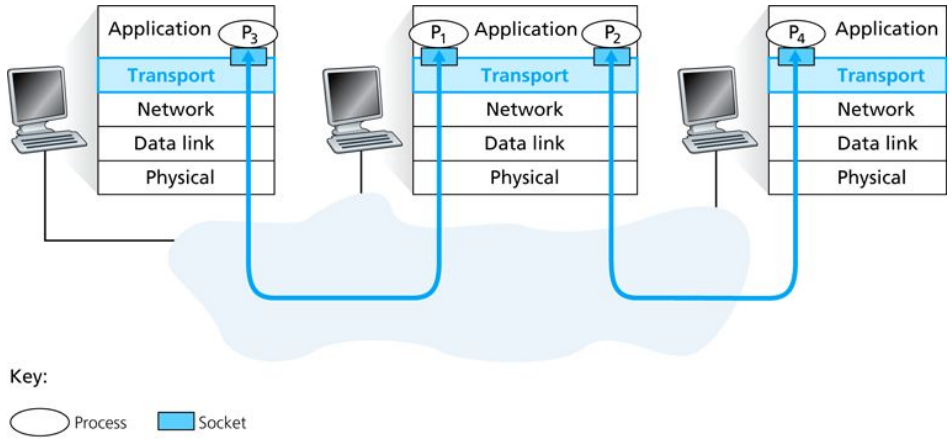
Layered Protocol Stack

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



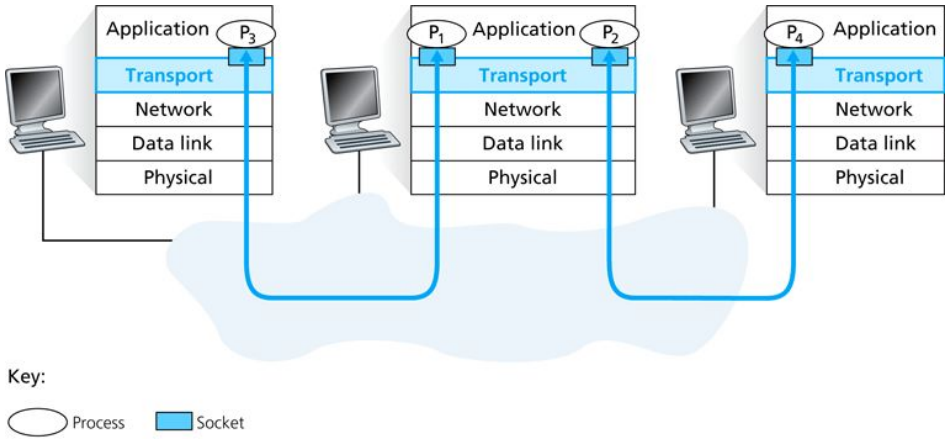
Orchestrated Discussion (Short Answer): De-/Multiplexing

- **What is de-/multiplexing?**



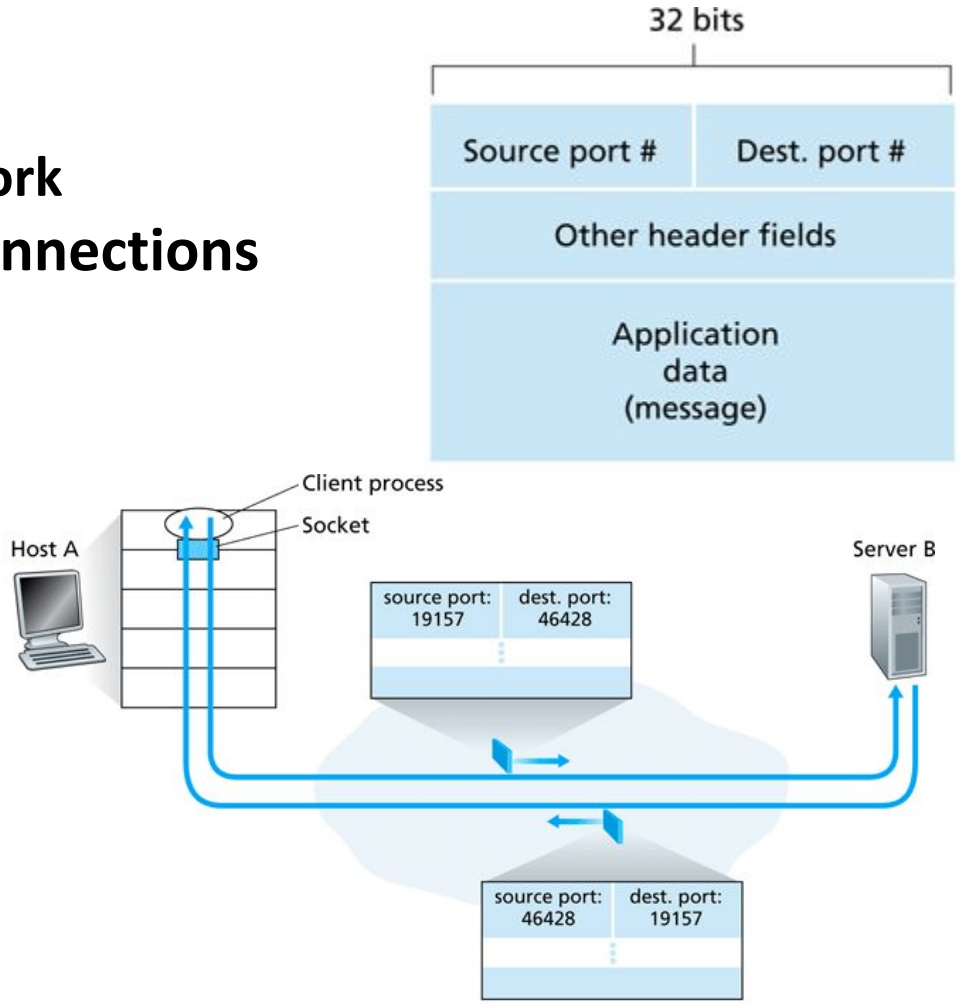
Group Discussion and Report Back (Short Answer): De-/Multiplexing

- **Why do we need this feature in the transport layer?**



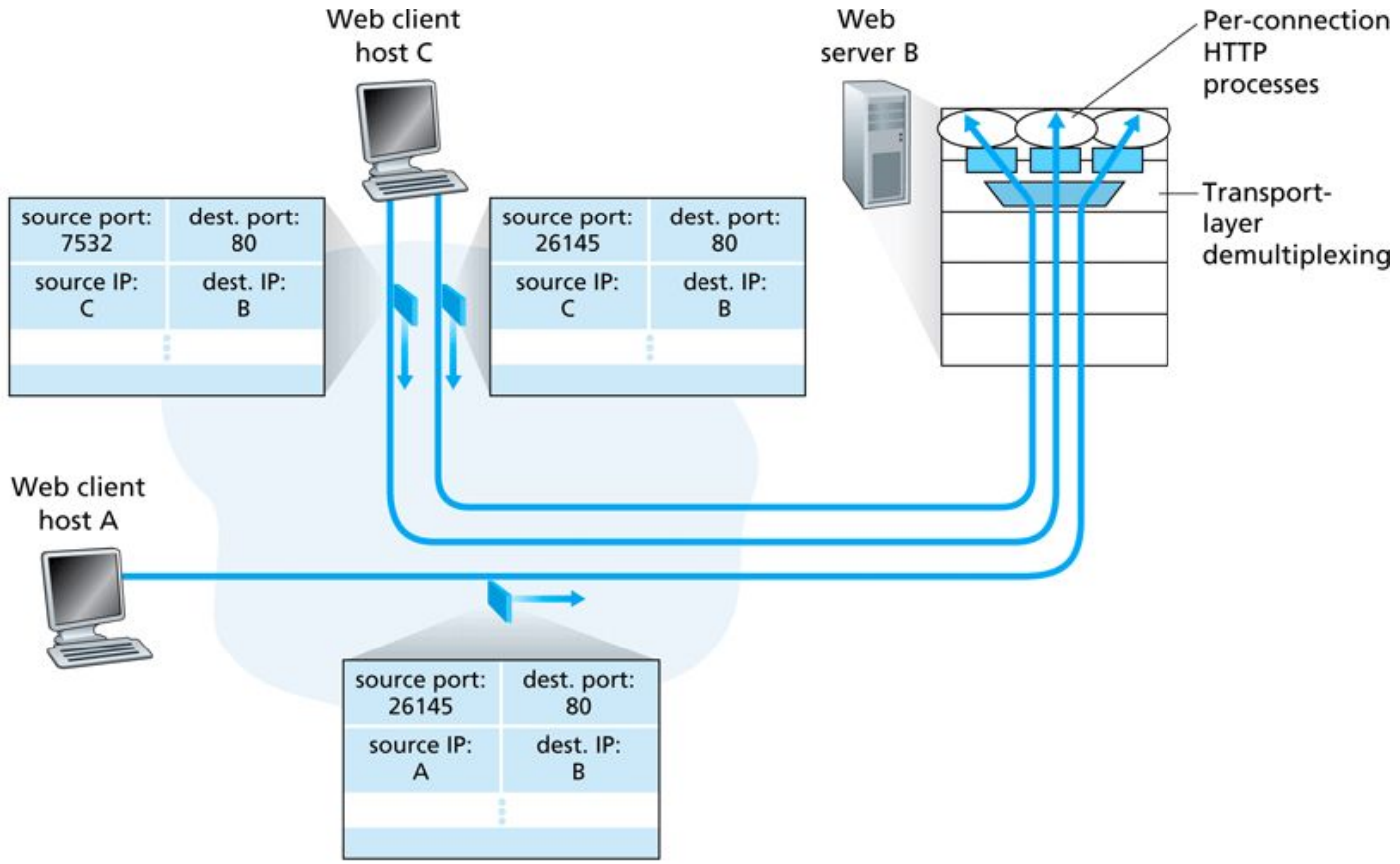
De-/Multiplexing

- Multiple processes on one host use the network
  - Need to distinguish different active connections
- "Port number" identifies the 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 swapped on reverse path



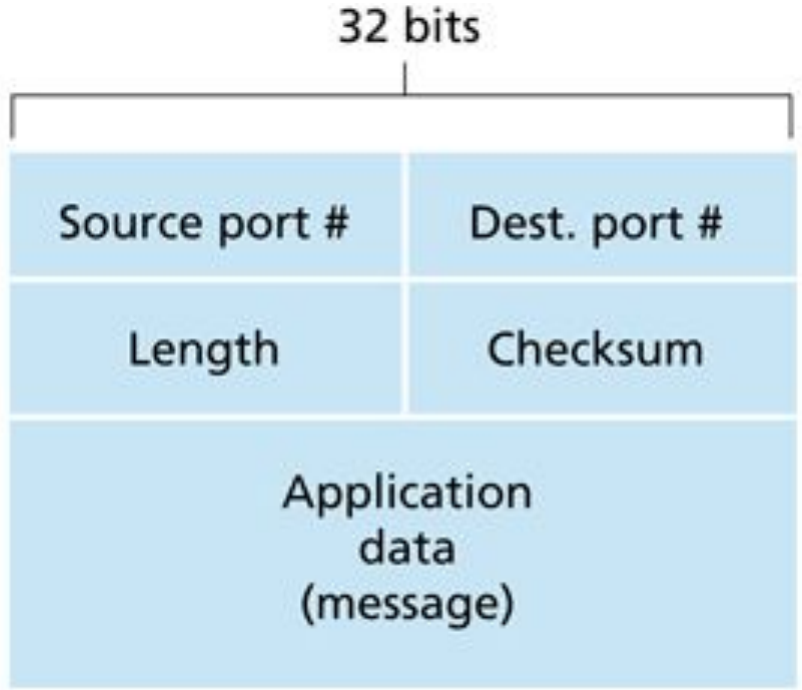
Document Cam: De-/Multiplexing

- **Works even if multiple connections go to the same port**



User Datagram Protocol

- Bare-bones transport protocol
  - Connectionless, state-free
  - Unreliable (because network layer is unreliable)
  - Low overhead
- UDP segment
  - Port numbers
  - Length field
  - Checksum field (optional!)
- UDP couldn't do less…



Orchestrated Discussion (Short Answer): Reliable Data Transfer

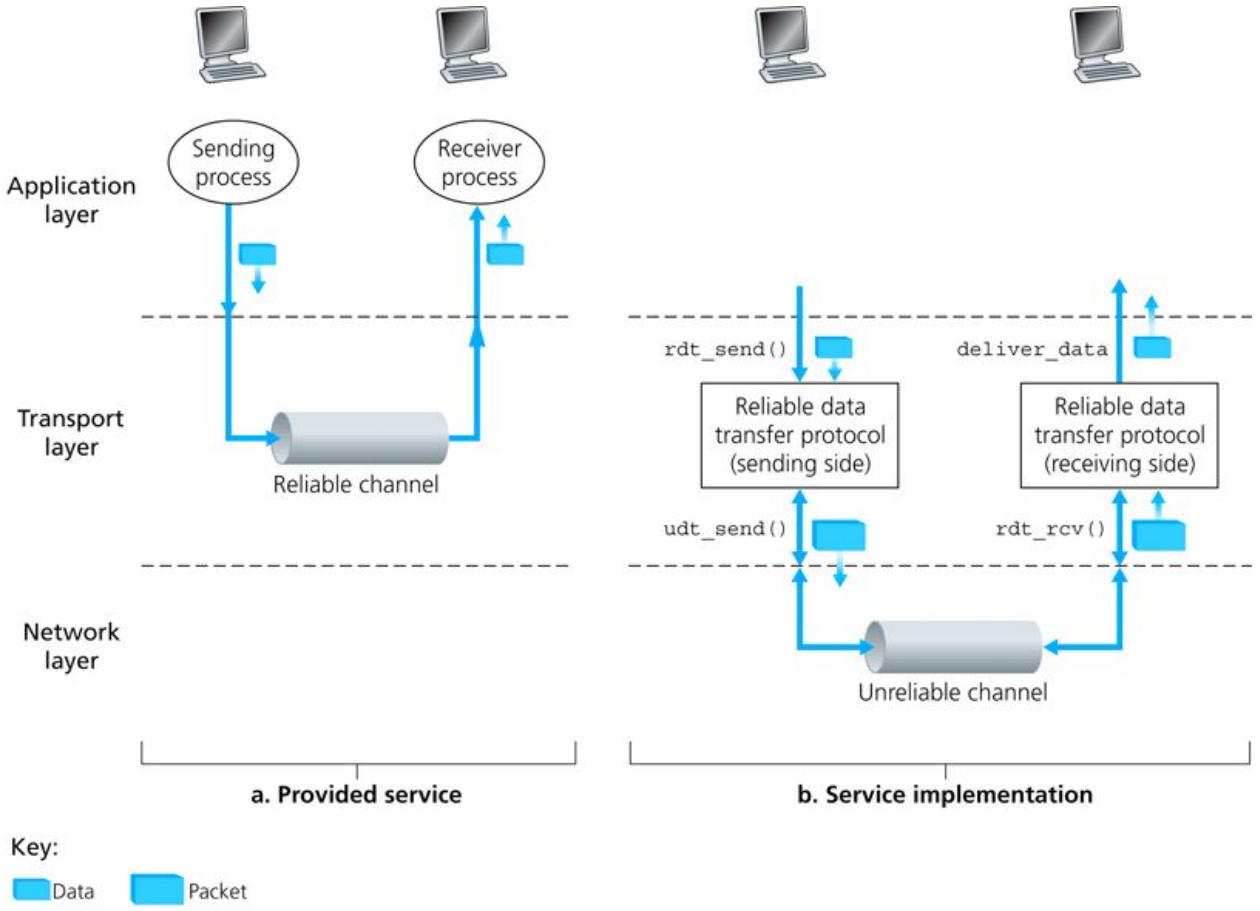
- What can happen that causes unreliable transfer?

Orchestrated Discussion (Short Answer): Reliable Data Transfer

- How can we ensure reliability?

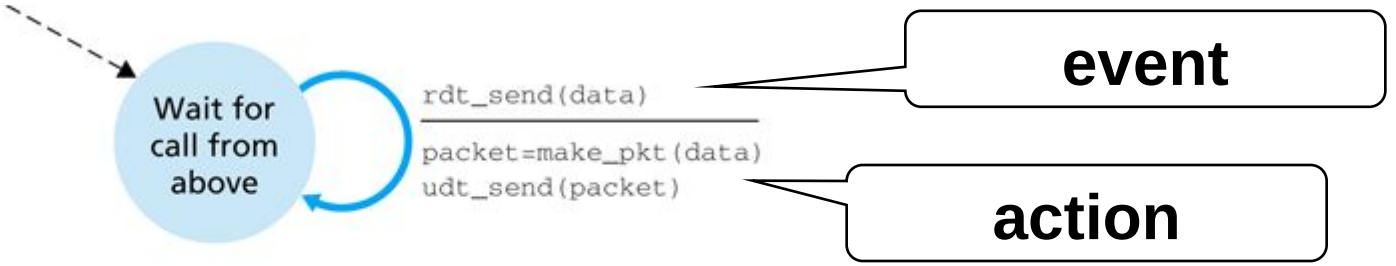
Reliable Data Transfer

- **Reliability is provided on top of unreliable channel**

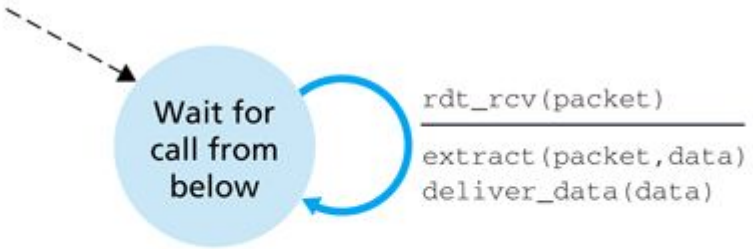


Whiteboard: Reliable Data Transfer

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



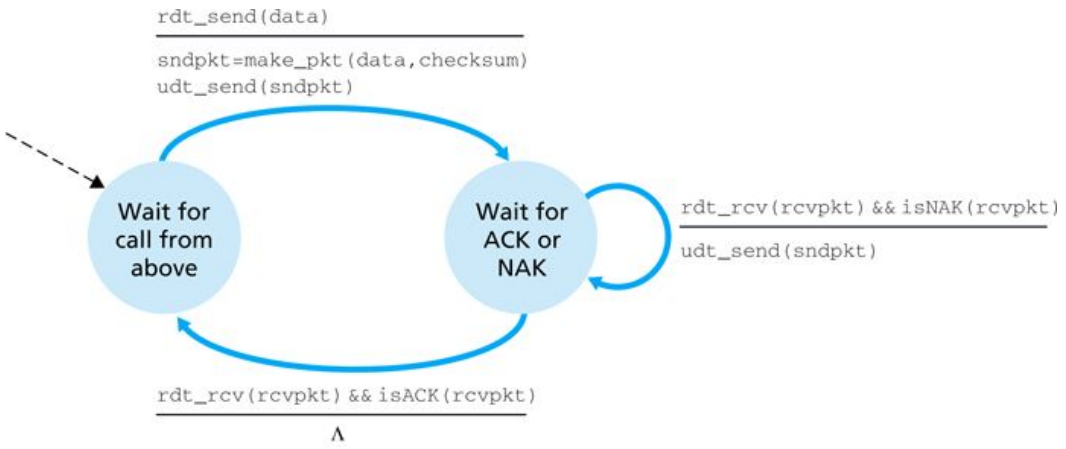
a. rdt1.0: sending side



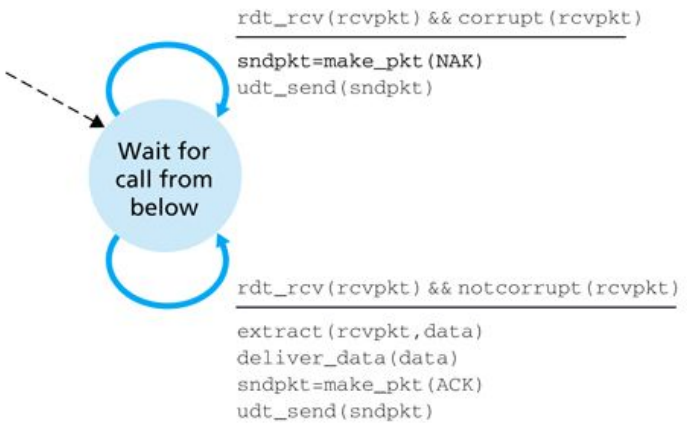
b. rdt1.0: receiving side

Whiteboard: 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?!



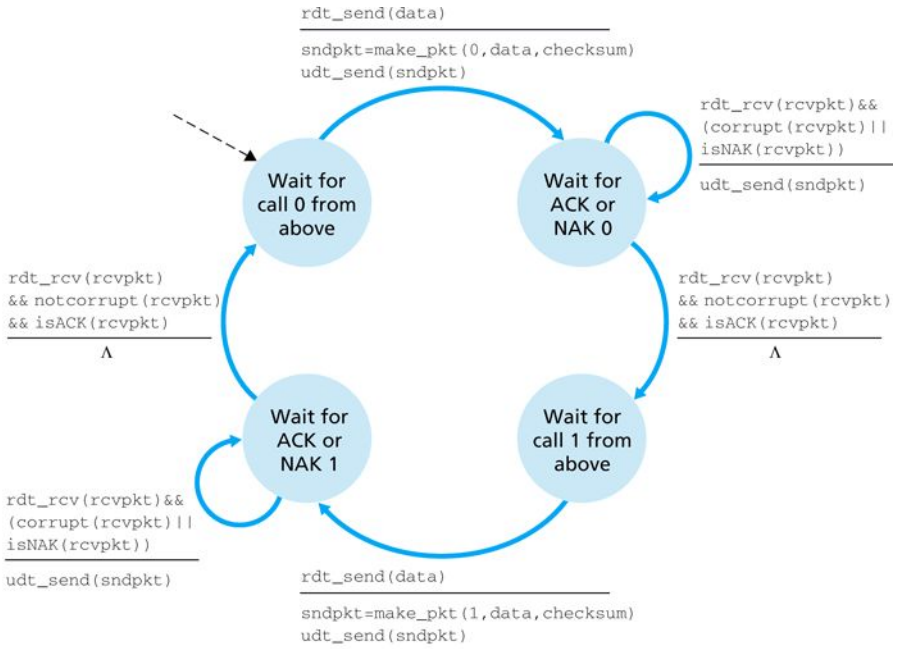
a. rdt2.0: sending side

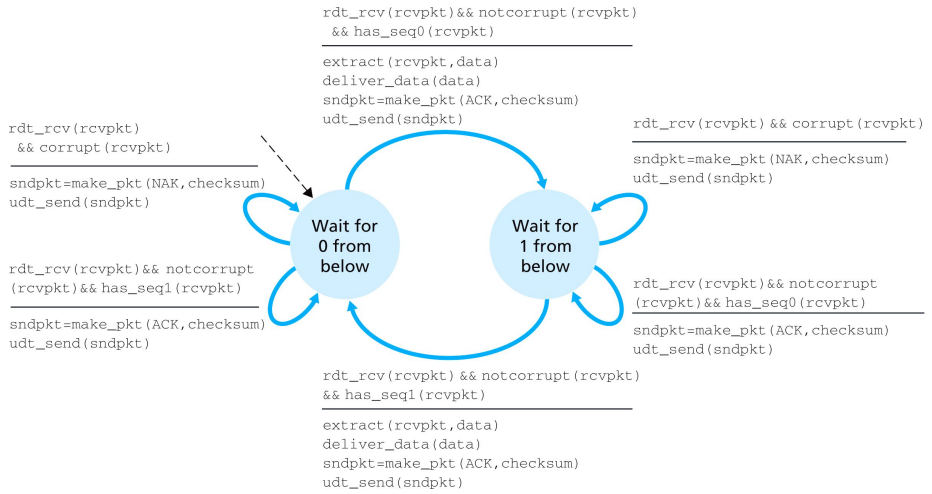


b. rdt2.0: receiving side

Reliable Data Transfer

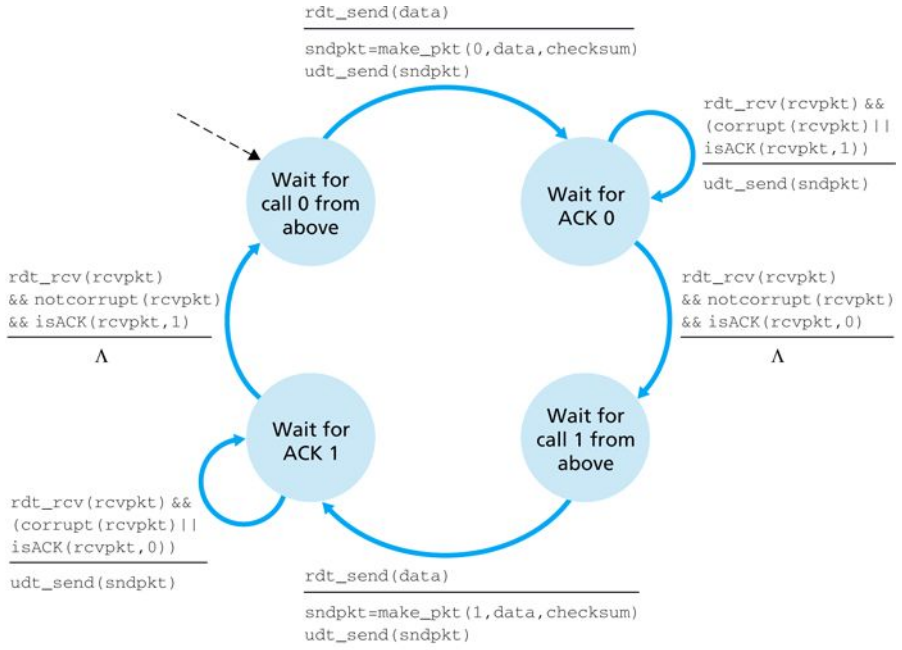
- What if data is retransmitted on garbled ACK/NAK?
  - Potentially lost or duplicated packets!
- Add sequence number to packet
  - How many sequence numbers do we need?

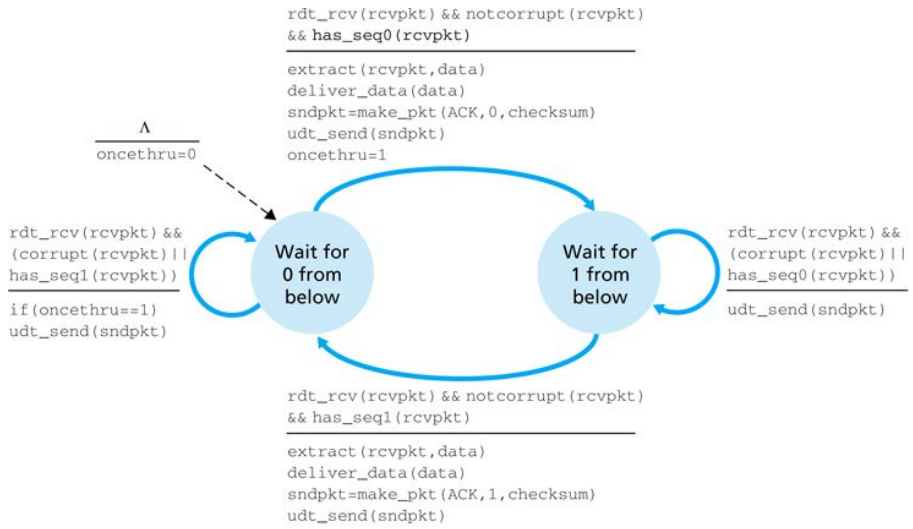




Reliable Data Transfer

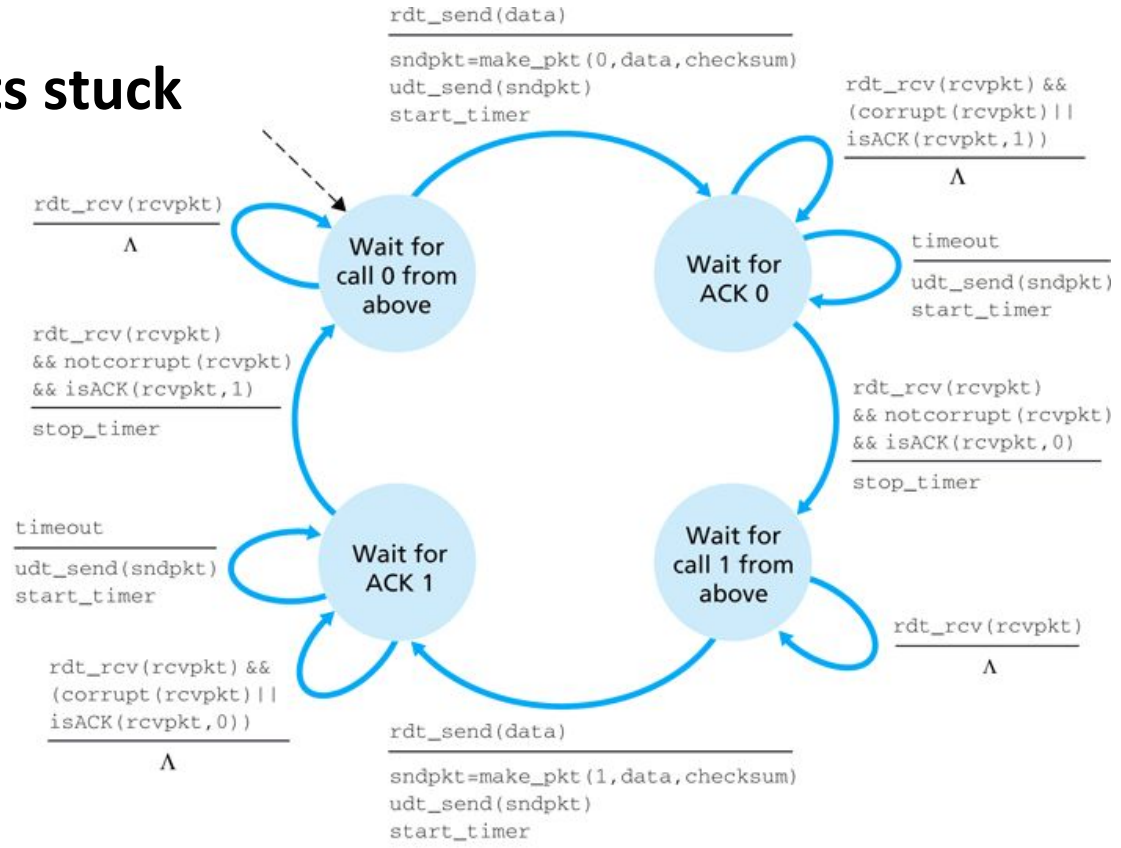
- Use only ACK instead of ACK/NAK
  - ACK requires sequence number





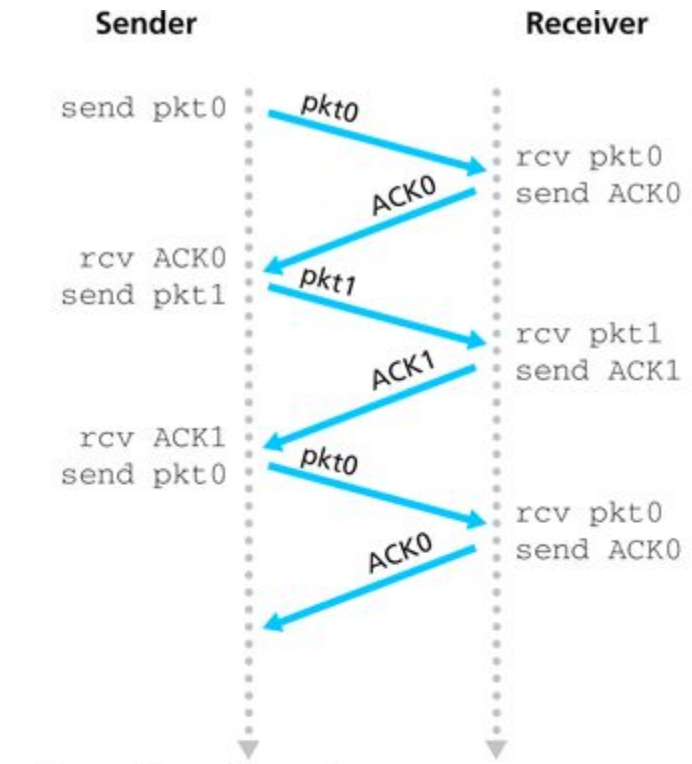
Whiteboard: Reliable Data Transfer

- 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"

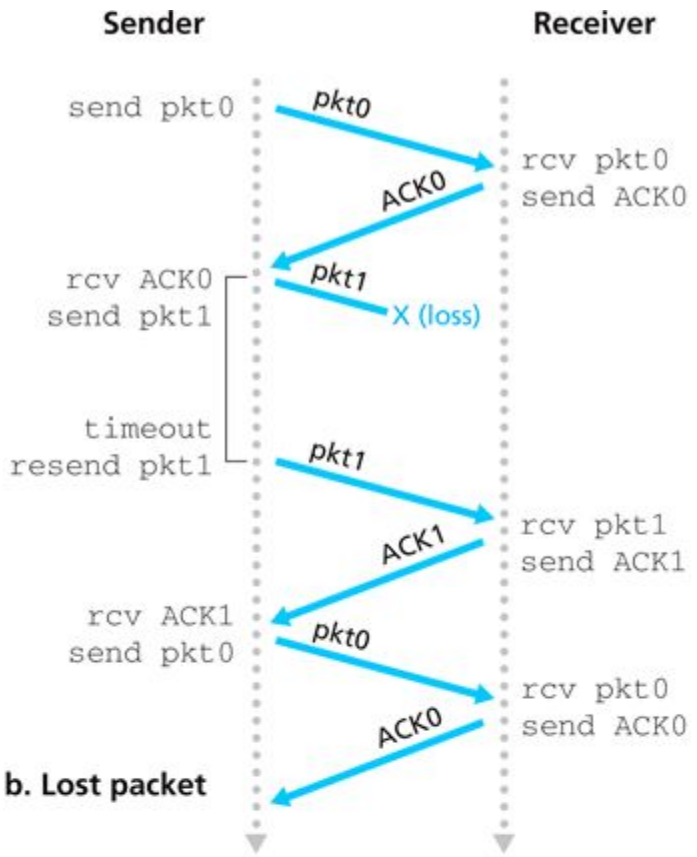


Alternating-Bit protocol

- Error scenarios are handled correctly



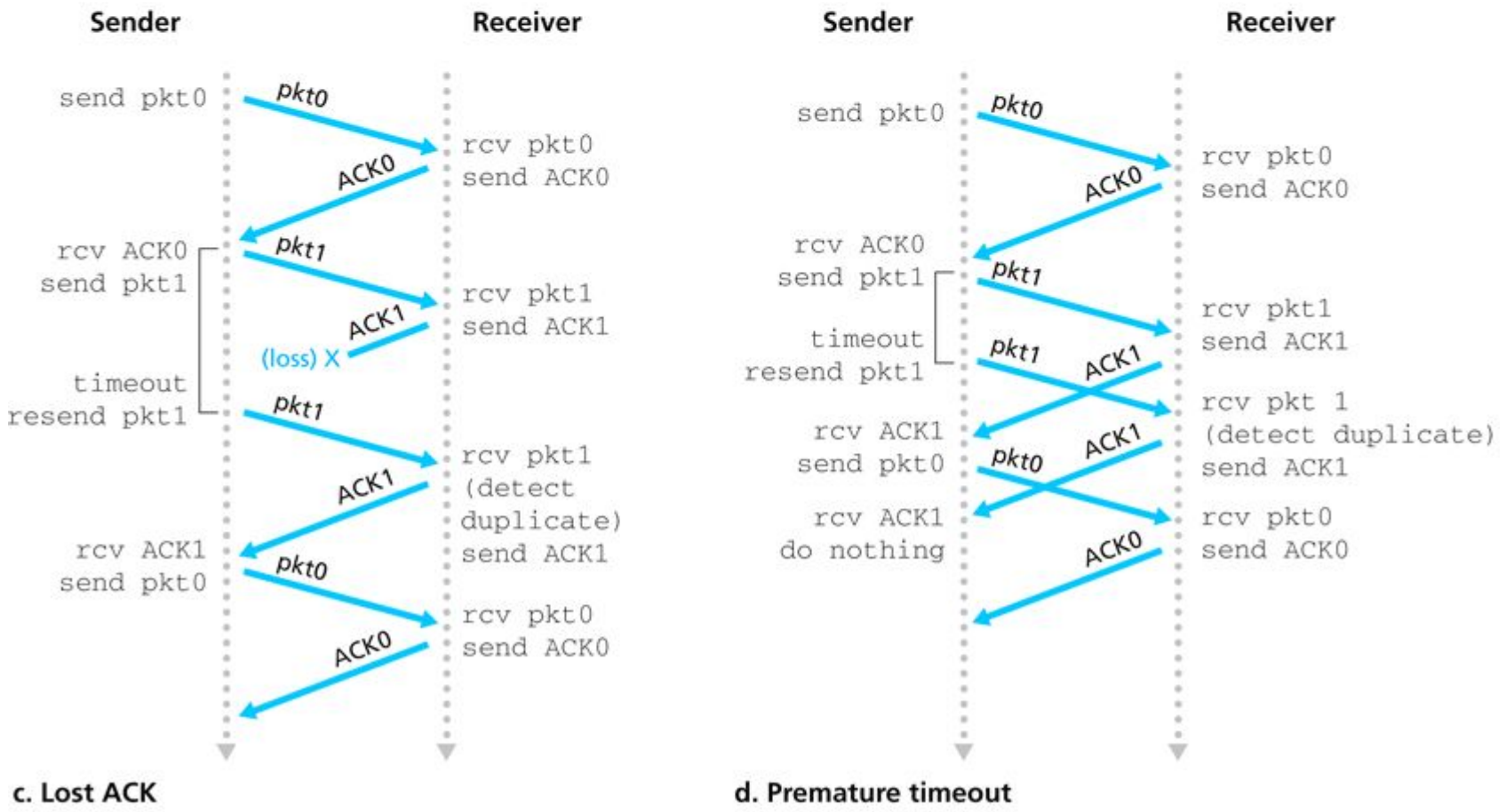
a. Operation with no loss



b. Lost packet

Alternating-Bit Protocol

- Error scenarios are handled correctly



Poll: Protocol Performance

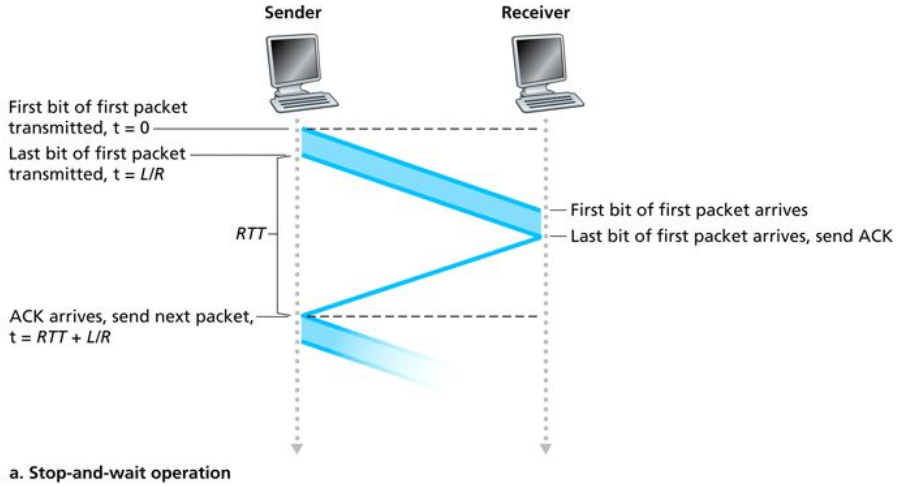
- What is the maximum throughput that can be achieved with alternating-bit protocol?
  - One packet per round-trip time
  - Two packets per round-trip time
  - Three packets per round-trip time
  - Four packets per round-trip time

Orchestrated Discussion (Short Answer): Protocol Performance

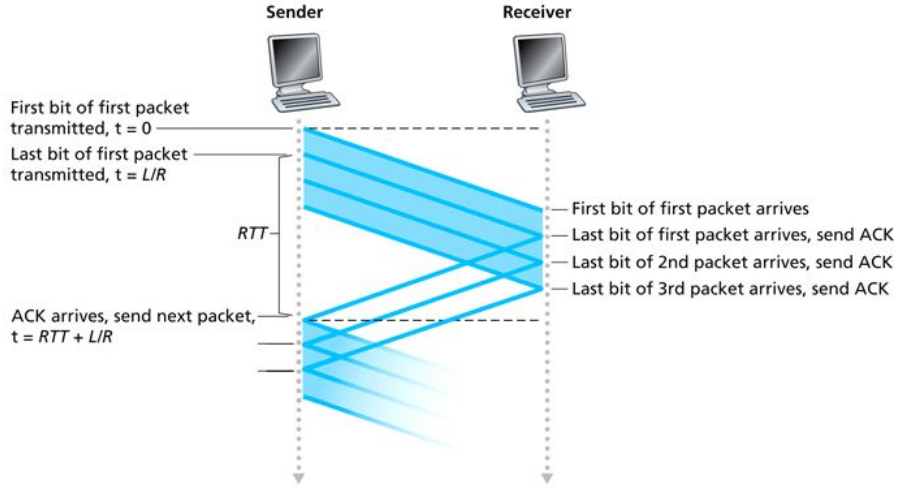
- How can the protocol throughput be improved?

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



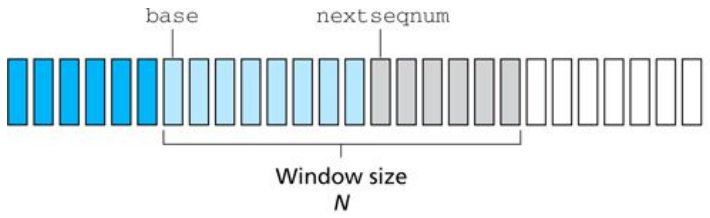
a. Stop-and-wait operation

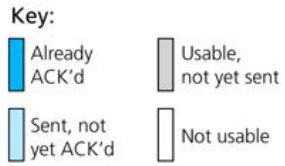


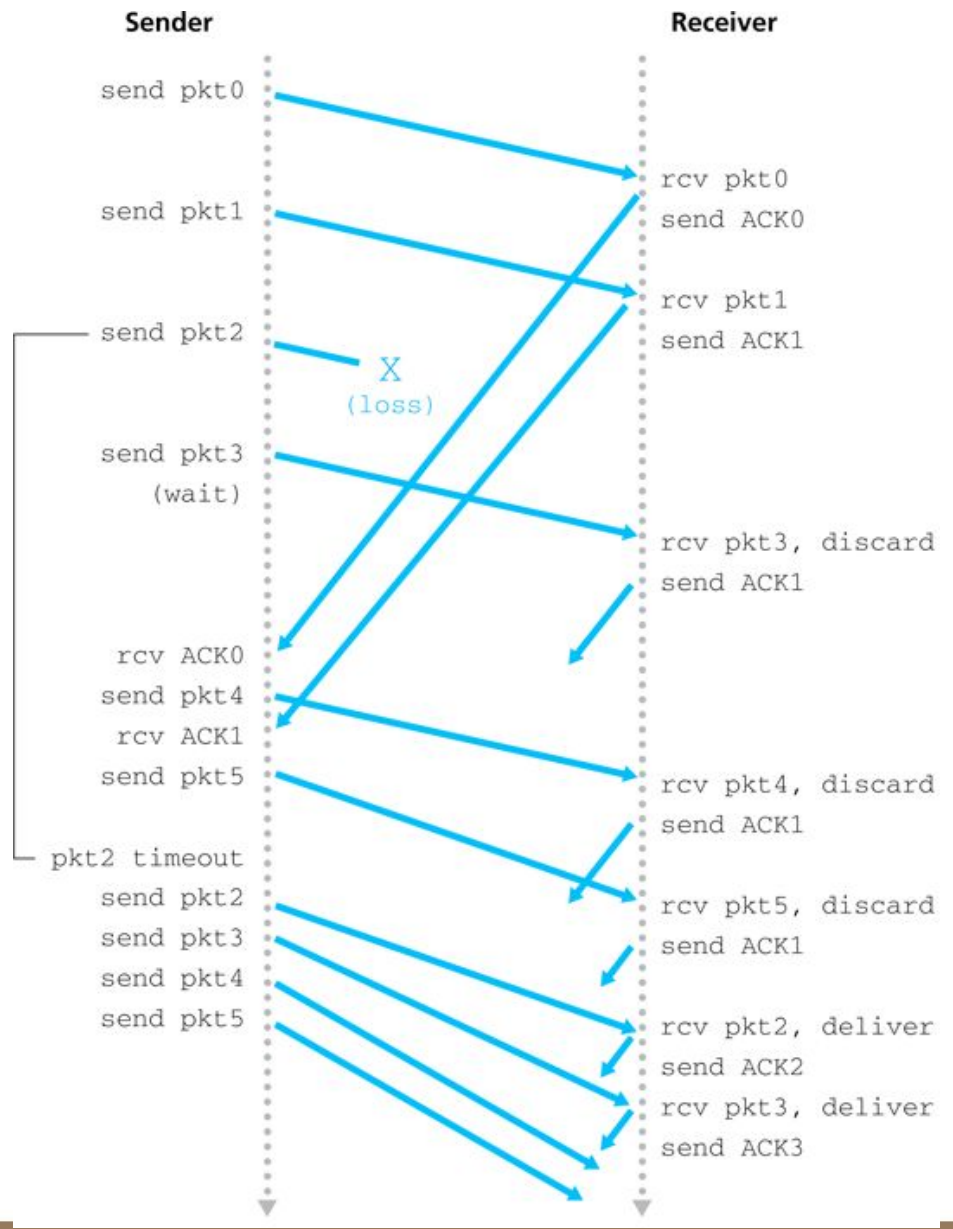
b. Pipelined operation

Go-Back-N

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

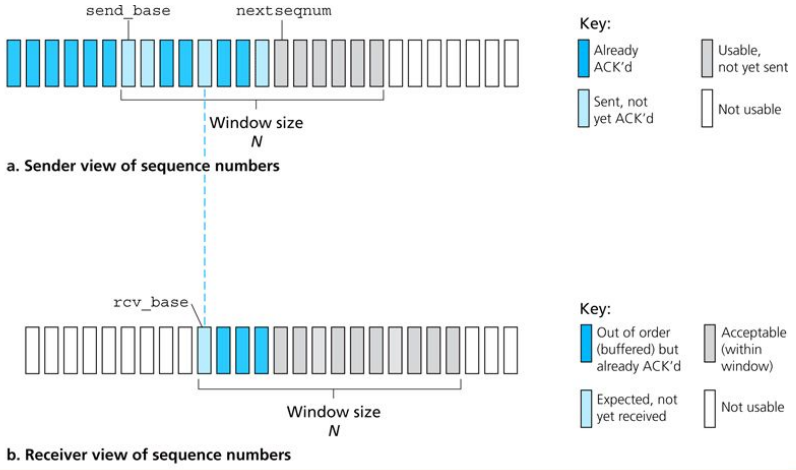


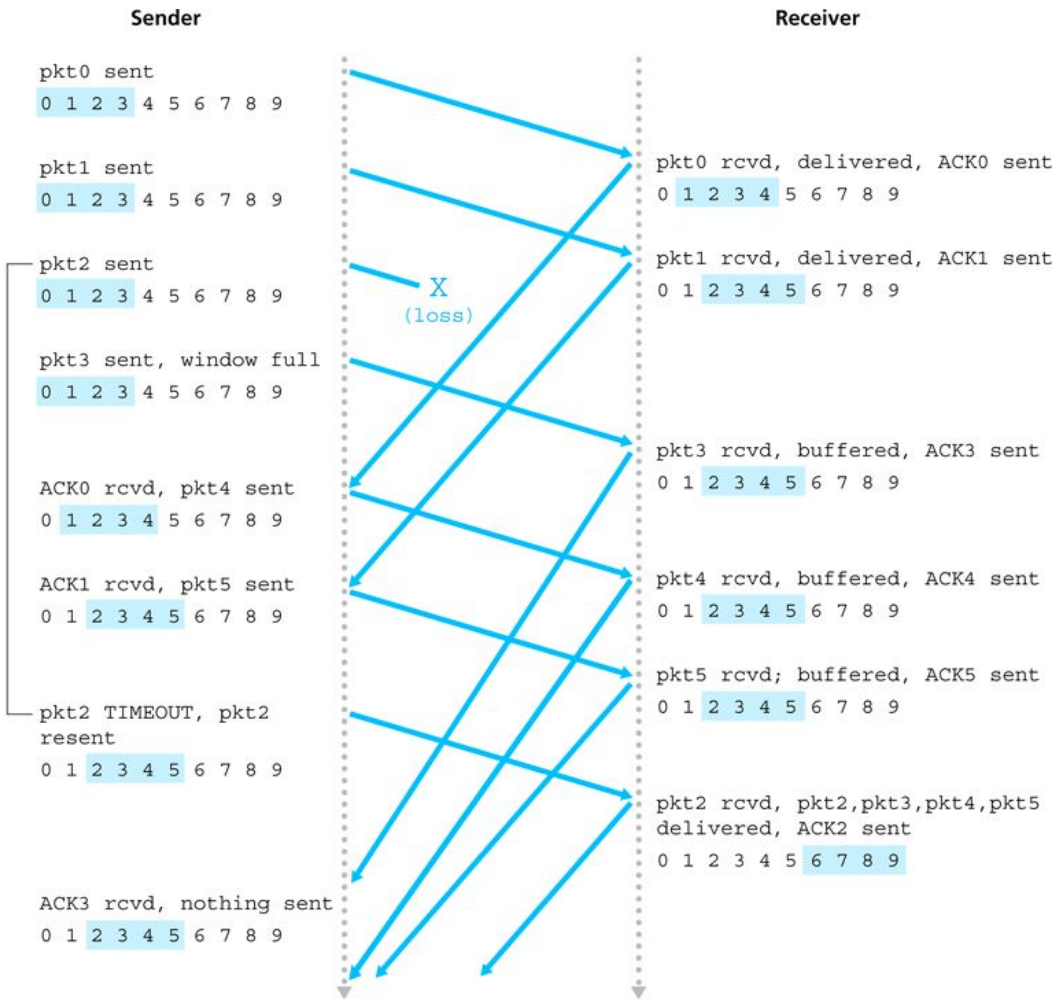




Selective Repeat

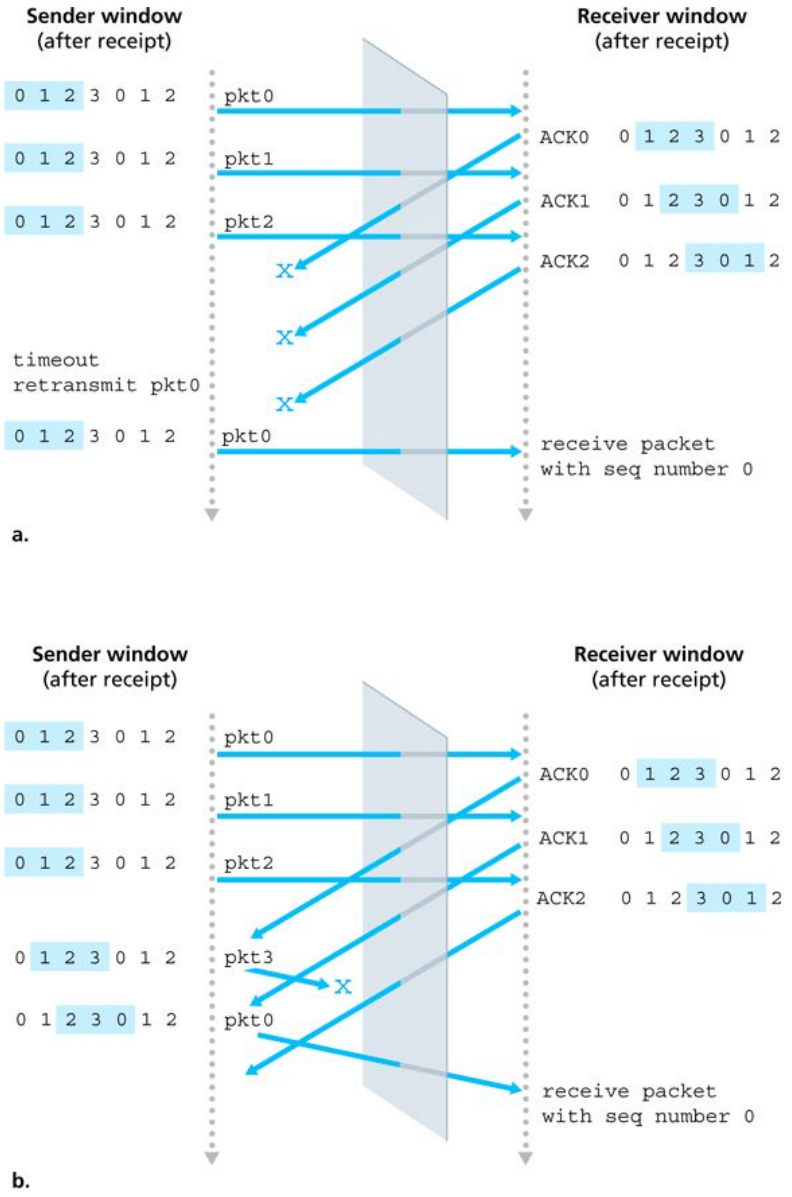
- Characteristics
  - ACK-based
  - Selective acknowledgements
  - On timeout, only unacknowledged packets are resent





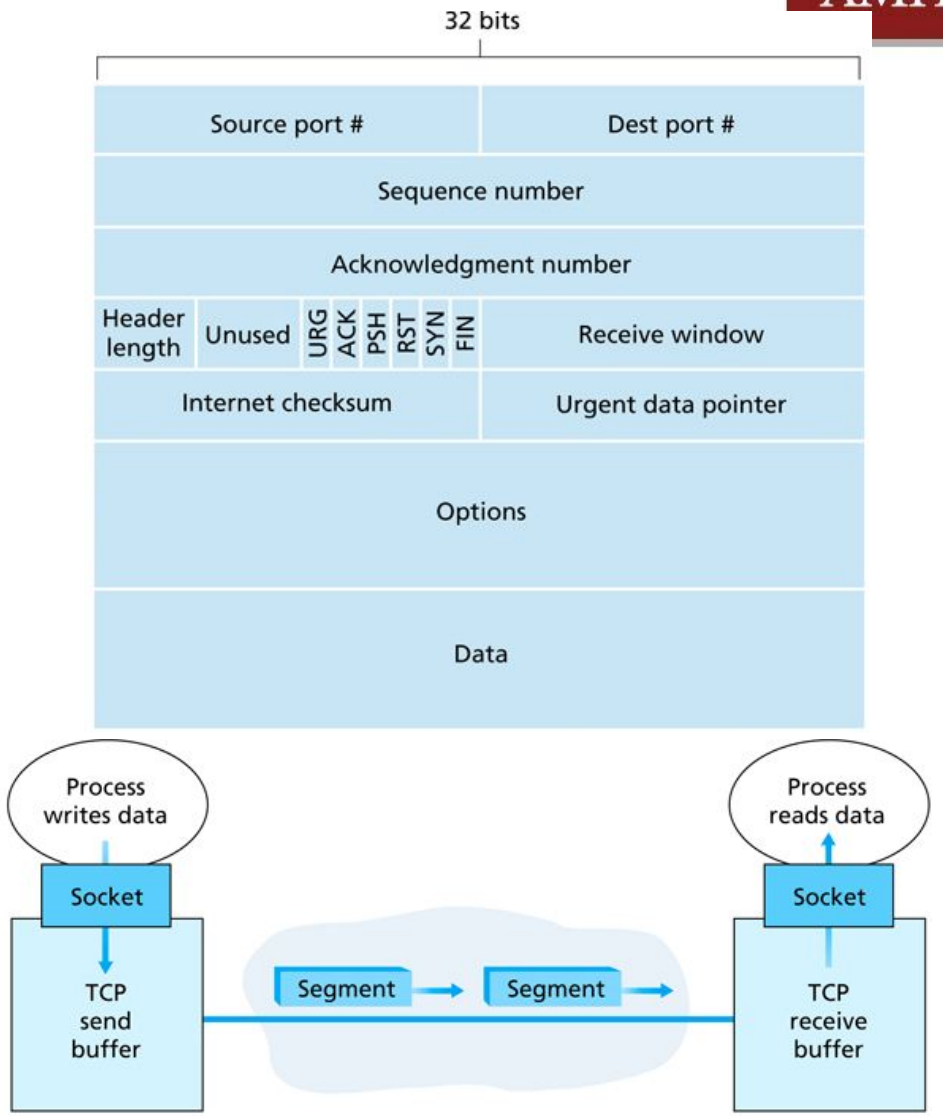
Sequence Numbers

- 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

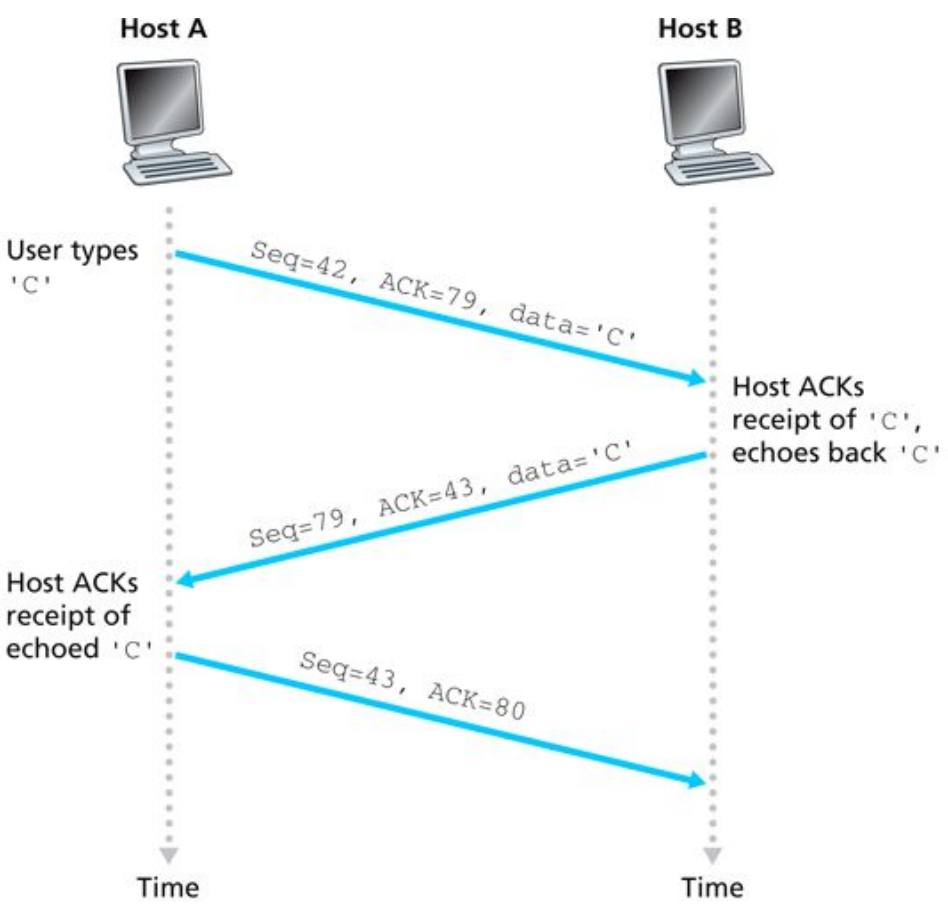


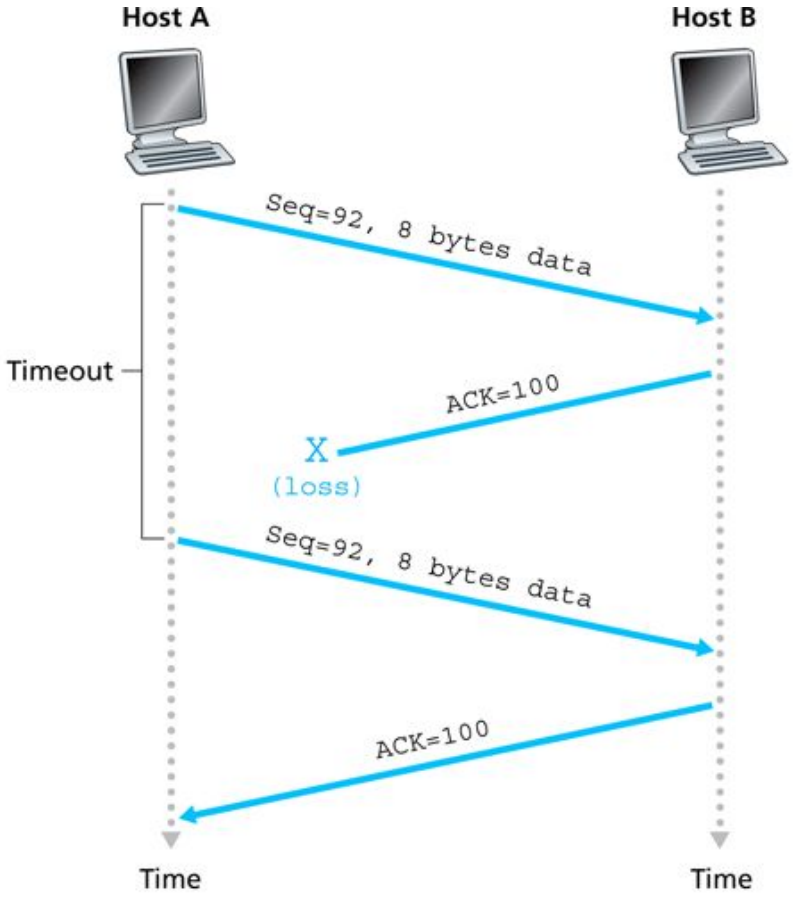
TCP Sequence Numbers

- Sequence numbers are position in byte-stream
  - 16bits = 64kByte window
- Acknowledgement number is next expected byte

TCP Operation

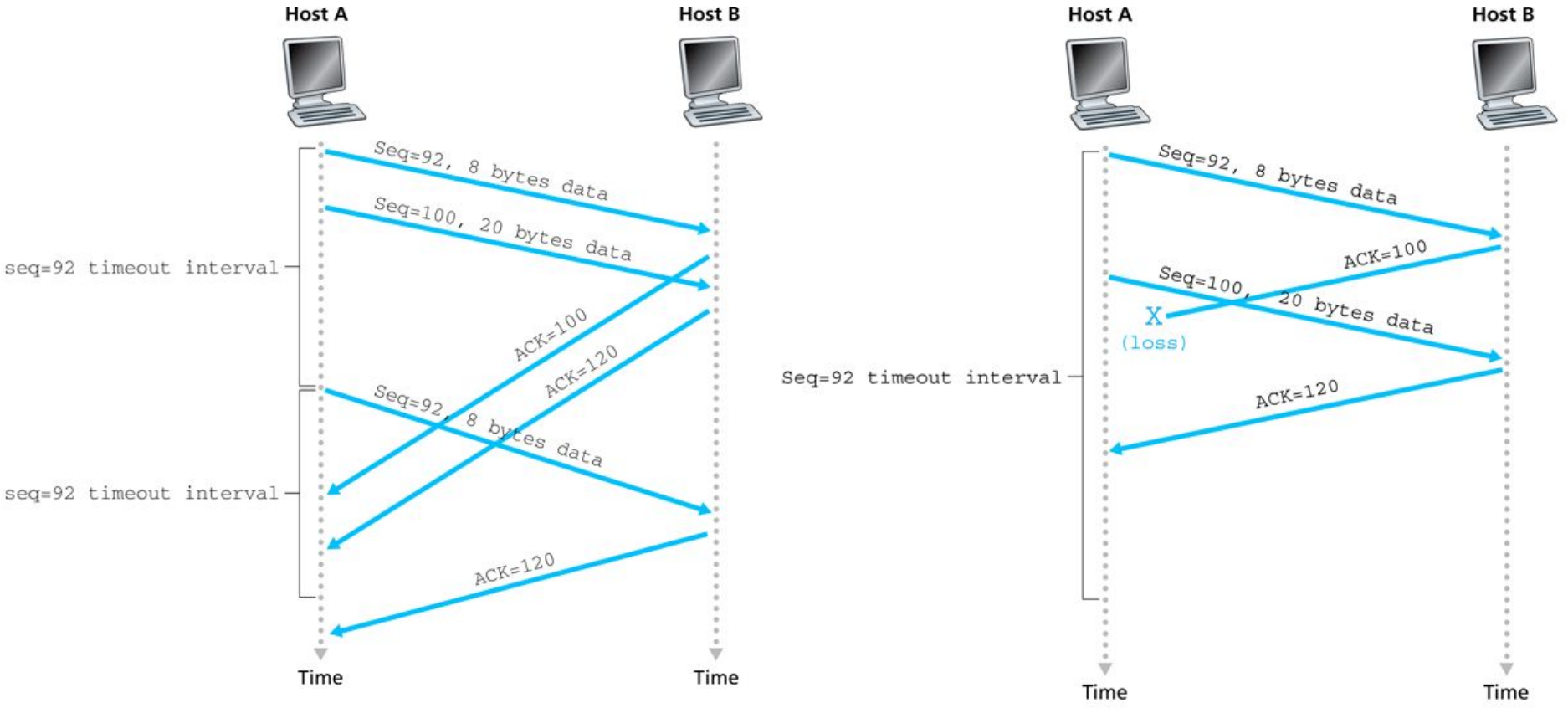
- **Acknowledgement loss and timeout**





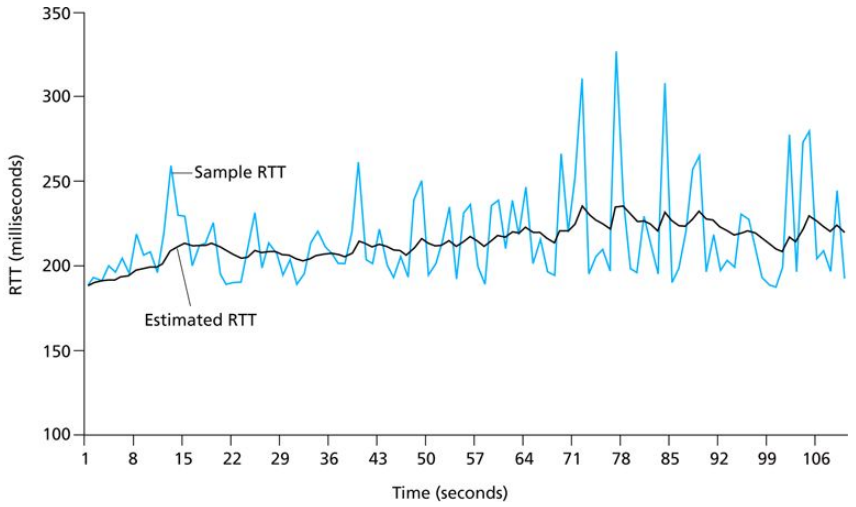
TCP Operation

- Timeout before acknowledgement received



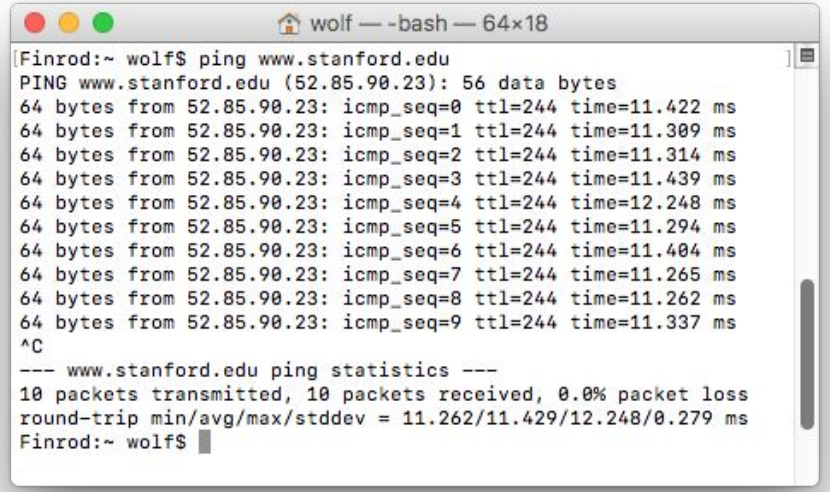
Round-Trip Time Estimation

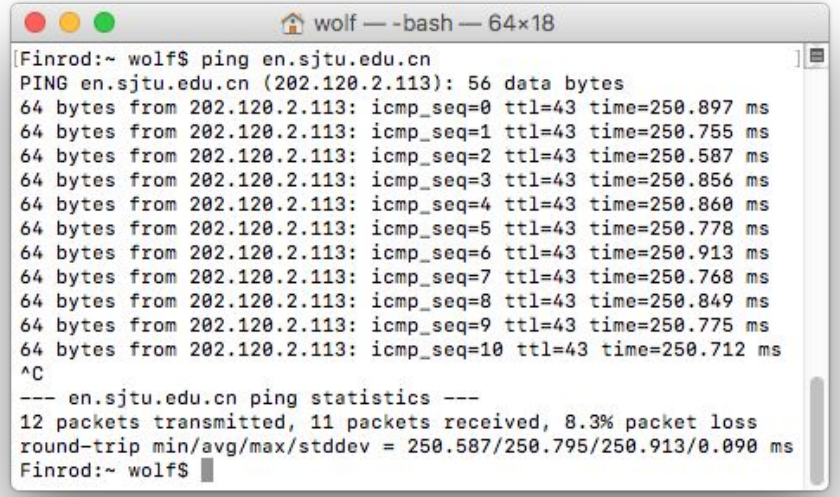
- RTT estimate important for efficient operation
  - Too long: long delay before retransmission
  - Too short: unnecessary retransmission
- TCP RTT estimation
  - TCP keeps exponential weighted moving average of RTT
  - Timeout is set to estimation+4×deviation
- Example:



Connected Device: Round-Trip Time Estimation

- Round-trip time is different for different connections
  - Can vary for the same connection due to queuing
- Round-trip time impacts performance





Summary of Lesson

- Transport layer functionality requires multiplexing/demultiplexing
- UDP is unreliable
- Reliability is achieved by error detection
- ACK/NAK
- Sequence number
- Performance is improved by pipelining
- TCP implements reliable data transfer
- TCP sequence and acknowledgement numbers
- Round-trip estimation

Post-work for Lesson 4

Lesson Reflection

- **After the Live Lecture, you will reflect on what you learned. Then, you will answer questions and share your observations. Go to the online classroom to view the questions and submit your responses.**

To Prepare for the Next Lesson

- Complete and submit the Post-work for Lesson 4.
- Read the Required Readings for Lesson 5.
- Complete the Pre-work for Lesson 5.

Go to the online classroom for details.