

# End-to-End Protocols



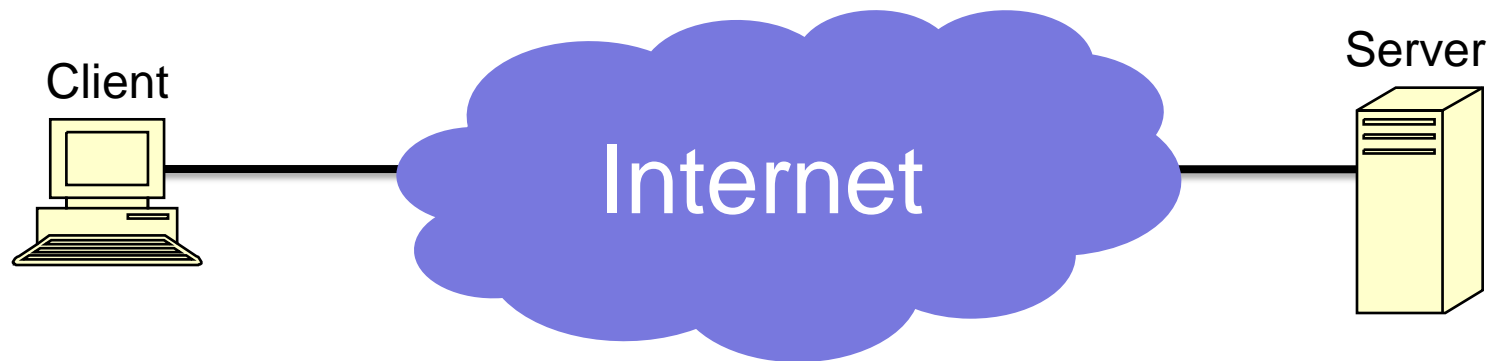
# Application Requirements

- Applications run end-to-end
  - Don't need to or want to know about intermediate networks, routers, links
  - Want end-to-end channel across the network
- Specific properties
  - Guaranteed message delivery
  - Delivers messages in the same order they are sent
  - Delivers at most one copy of each message
  - Supports arbitrarily large messages
  - Supports synchronization between the sender and the receiver
  - Allows the receiver to apply flow control to the sender
  - Supports multiple application processes on each host



# Network Properties

- Networks can...
  - Drop messages
  - Reorder messages
  - Deliver duplicate copies of messages
  - Limit messages to a finite size
  - Deliver messages after arbitrarily long delay

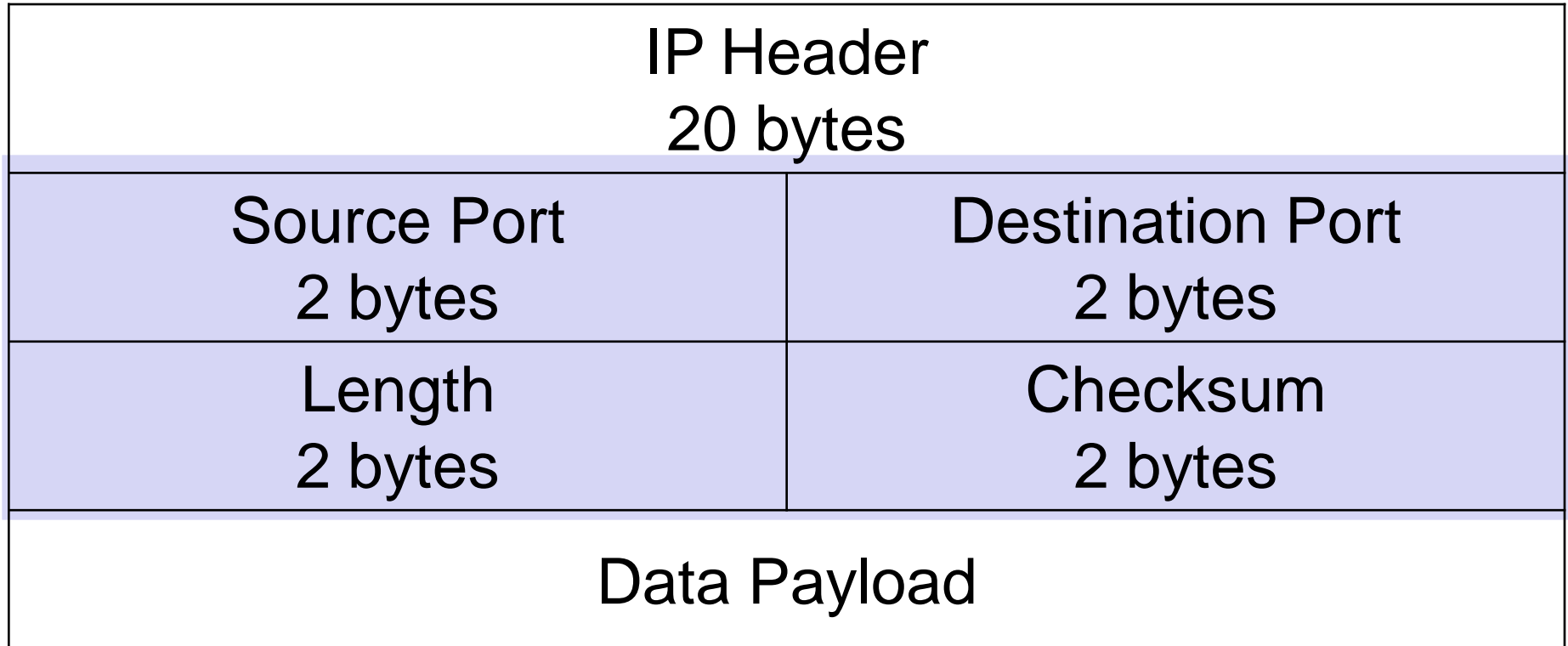


# Multiple Application Processes on Each Host

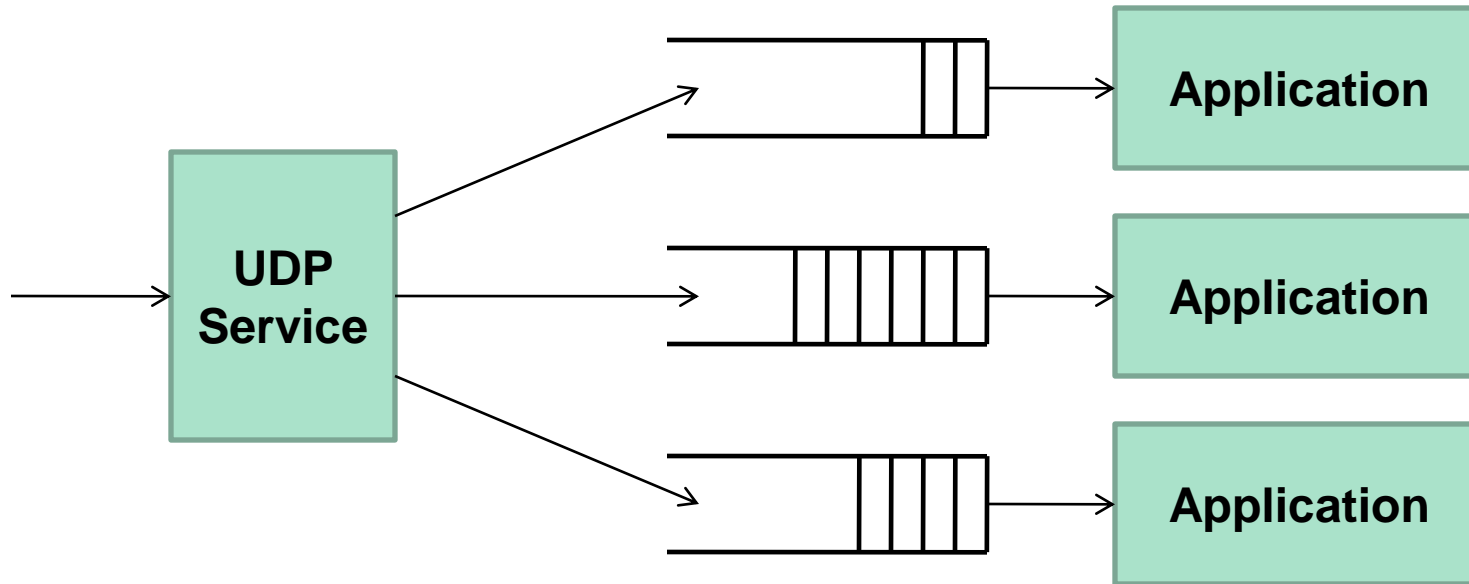
- User Datagram Protocol (UDP)
- Simple Demultiplexer
  - Port numbers
    - Source and destination
    - Analogy
      - IP = post office
      - UDP port = mailbox
      - Allows multiple users to share a post office, like multiple applications sharing a single Internet host



# UDP Format



# UDP Demultiplexing



One queue per port



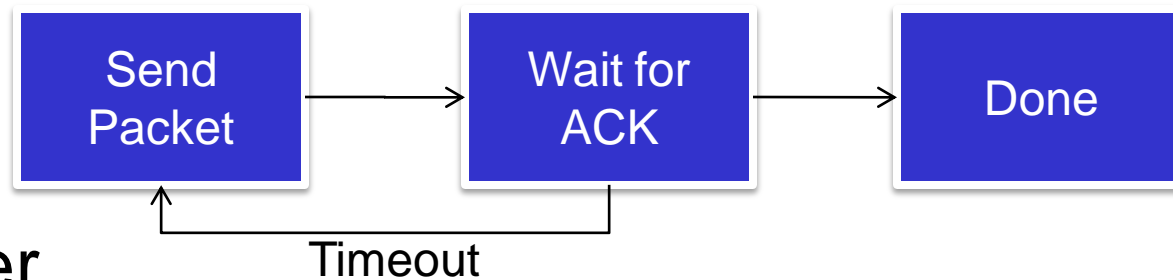
# Transmission Control Protocol (TCP)

- Provides additional services
  - Guaranteed message delivery
  - Delivers messages in the same order they are sent
  - Delivers at most one copy of each message
  - Supports arbitrarily large messages
  - Supports synchronization between the sender and the receiver
  - Allows the receiver to apply flow control to the sender

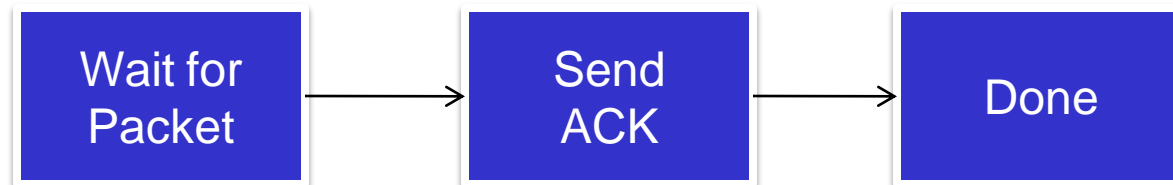


# Early approaches

- Automatic Repeat-Query (ARQ)
- “Stop and Wait” protocol
- Transmitter



- Receiver



- Transmitter ensures delivery of packet N before sending packet N+1



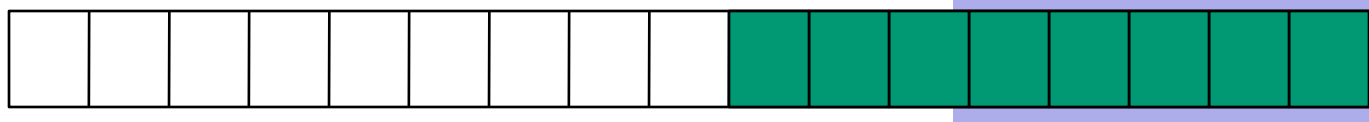
# Performance of ARQ

- Error-free environment:
  - Total time to transmit a packet
    - $\text{tx-time}(\text{packet}) + \text{tx-time}(\text{ack}) + 2 * \text{prop-time}$
  - Bits of data delivered
    - $\text{size}(\text{packet})$
  - Rate
    - $\text{size}(\text{packet}) / (\text{tx-time}(\text{packet}) + \text{tx-time}(\text{ack}) + 2 * \text{prop-time})$
  - Raw link rate
    - $\text{size}(\text{packet}) / \text{tx-time}(\text{packet})$
  - ARQ Efficiency
    - $\text{tx-time}(\text{packet}) / (\text{tx-time}(\text{packet}) + \text{tx-time}(\text{ack}) + 2 * \text{prop-time})$
- Environment with errors: much worse



# Improvement: Sliding Window Protocol

- Allow multiple unacknowledged packets in transit
- As packets are acknowledged, transmit additional ones
- “Window Size” defines how many unacknowledged packets can be in transit



# Sliding Window

- Both transmitter and receiver maintain sliding window
  - Transmitter uses it to decide which packets to send, as ACKs arrive, the window slides
  - Receiver uses it to keep track of reconstructed data streams, and only delivers data to the application when it is complete, in-order



# ACK Strategies

- Per-Packet ACK
  - Send ACK with sequence number of each received packet
- Cumulative ACK
  - Send ACK with sequence number of last packet delivered to application
- Selected ACK
  - Send bit mask of packets in current window received
- NACK
  - Send NACK for packets NOT received



# Tradeoffs

- Cumulative ACKs

- Pros

- Fewer ACKs can be sent for the same number of packets, so there can be a saving in reverse channel bandwidth.
    - Sender window can move faster since It does not have to wait for all ACKs

- Cons

- Does not help in reducing retransmissions. Although the receiver might have later packets, it would ACK an earlier one if an intermediate packet is lost.
    - Can cause bursty transmission, since the window can increase suddenly due to a cumulative ACK.



# Tradeoffs

- Selective ACKs

- Pros

- Indicates the packets that need to be retransmitted explicitly and avoids unnecessary retransmission.
    - Fewer ACKs sent

- Cons

- Sequence number problems – if sequence numbers wrap there is no way to find out if the ACK is for an older session.



# Tradeoffs

- NAKs

- Pros

- Indicates only the packets that were not received or corrupted and hence prevents unnecessary retransmissions.
    - Fewer “ACK”s, no packets sent to confirm a proper receipt, hence bandwidth saved.

- Cons:

- If a NAK for a packet is lost, that packet might never be recovered.



# TCP Window Size

- The larger the TCP Window size, the more packets will be in transit
- If number of packets exceeds bandwidth-delay product for the end-to-end link packets will be dropped
- $WS * MTU * 8 = \text{bits in transit}$
- $\text{Rate} * \text{Latency} = \text{BW-Delay}$
- $WS = \text{Rate} * \text{Latency} / (MTU * 8)$



# TCP Window Size

- $WS = \text{Rate} * \text{Latency} / (\text{MTU} * 8)$
- If Rate, Latency unknown, how do you compute WS?
- Determine it experimentally
- Increase WS until packet loss occurs, then decrease it
- After each complete window
  - If no drops,  $WS = WS + N$
  - If drop,  $WS = WS / 2$
- Additive Increase, Multiplicative Decrease

