Transport
Part I

Lecture 5, Computer Networks (198:552)

Fall 2019

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Network Core: Best effort packet delivery

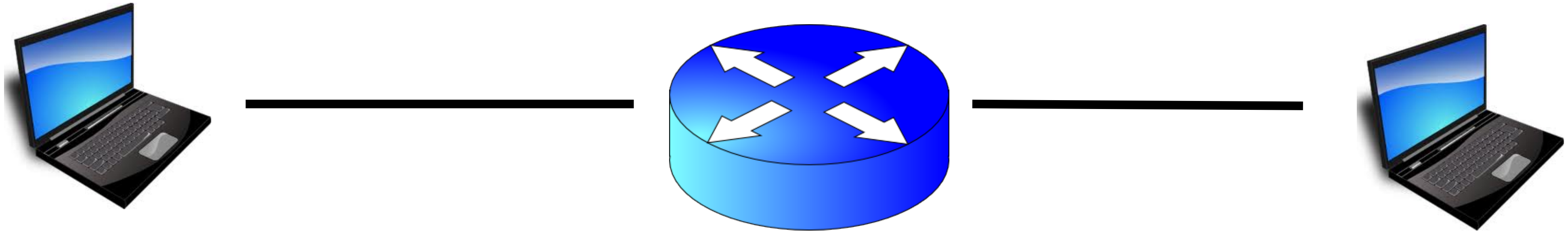
- Routers (typically) make no guarantees about

- … whether packets get delivered
- … whether packets will reach without being corrupted
- … whether packets will reach the other side in order
- … the app performance experienced by a user

- So how are we still able to get good performance over the Internet?

Network Edge: Application guarantees

- How should endpoints provide guarantees to applications?



- Transport software on the endpoint is in charge of implementing guarantees on top of an unreliable network
  - Reliability
  - Ordered delivery
  - Packet delay not exceeding 50 ms?

Modularity through layering

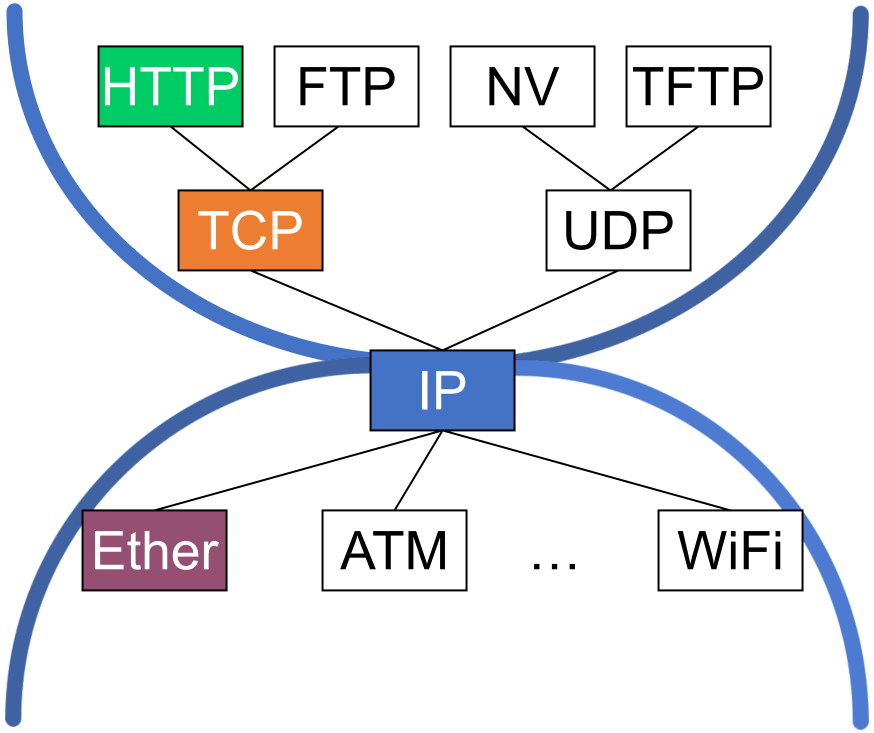
Protocols “stacked” in endpoint and router software/hardware

Apps: useful user-level functions

Transport: provide guarantees to apps

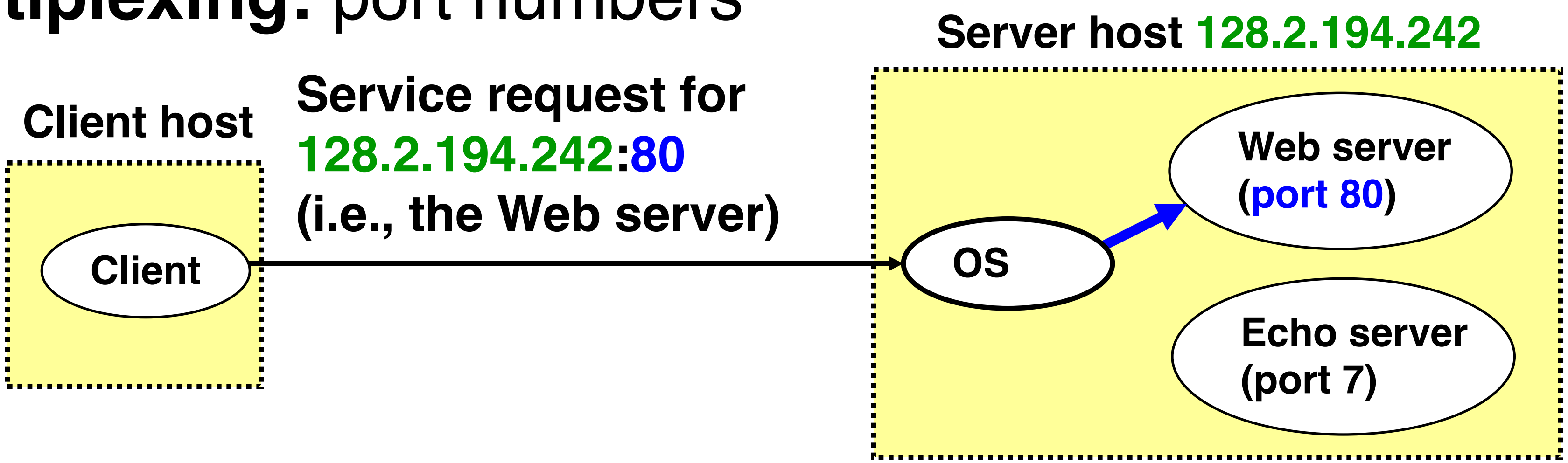
Network: best-effort global pkt delivery

Link: best-effort local pkt delivery

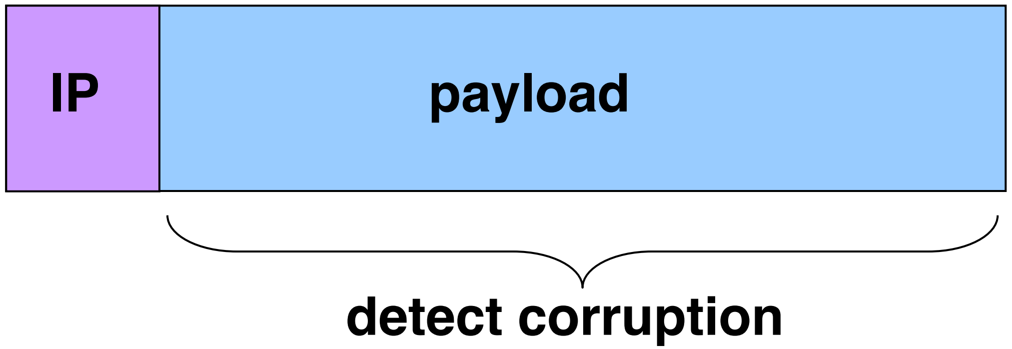


Two Basic Transport Features

- **Demultiplexing:** port numbers



- **Error detection:** checksums



Two Main Transport Layers

- User Datagram Protocol (UDP)
  - Abstraction of independent messages between endpoints
  - Just provides demultiplexing and error detection
  - Header fields: port numbers, checksum, and length
  - Low overhead, good for query/response and multimedia

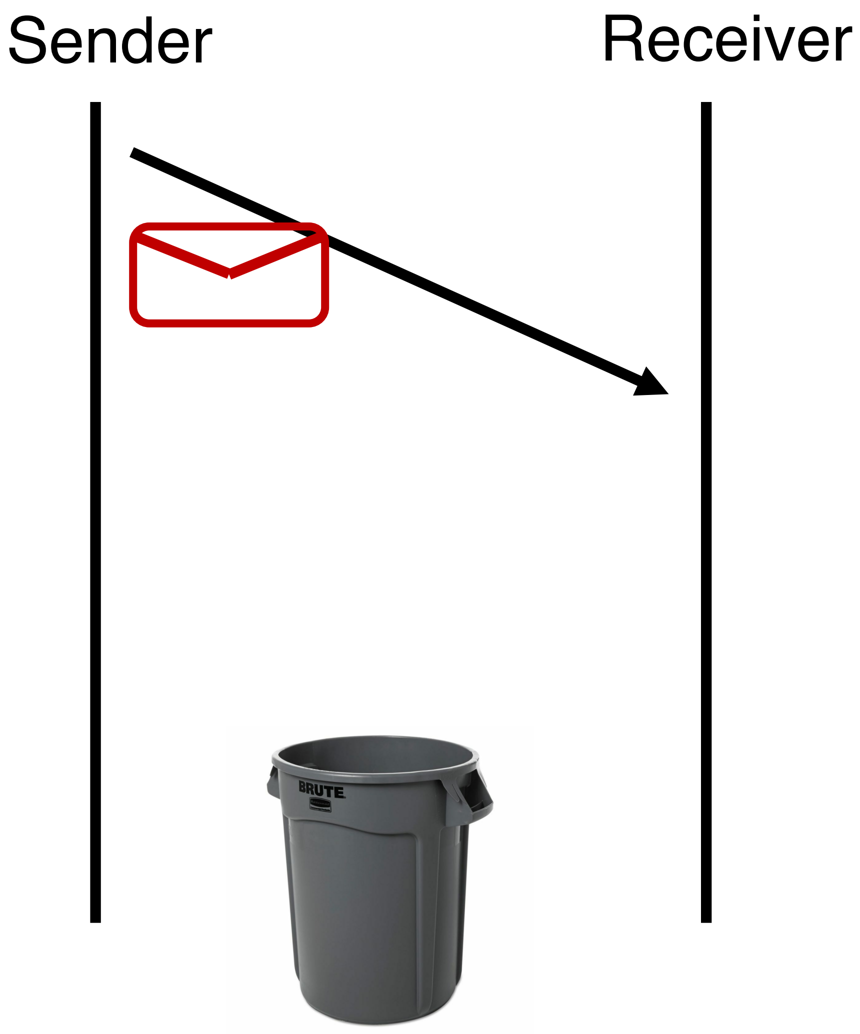
- Transmission Control Protocol (TCP)
  - Provides support for a stream of bytes abstraction

Transmission Control Protocol (TCP)

- Multiplexing/demultiplexing
  - Determine which conversation a given packet belongs to
  - All transports need to do this
- Reliability and flow control
  - Ensure that data sent is delivered to the receiver application
  - Ensure that receiver buffer doesn't overflow
- Ordered delivery
  - Ensure bits pushed by sender arrive at receiver app in order
  - Q: why would packets ever be received out of order?
- Congestion control
  - Ensure that data sent doesn't overwhelm network resources
  - Q: which network resource?

Reliable data delivery

Packet loss



- How might a sender and receiver ensure that data is delivered reliably (despite some packets being lost)?

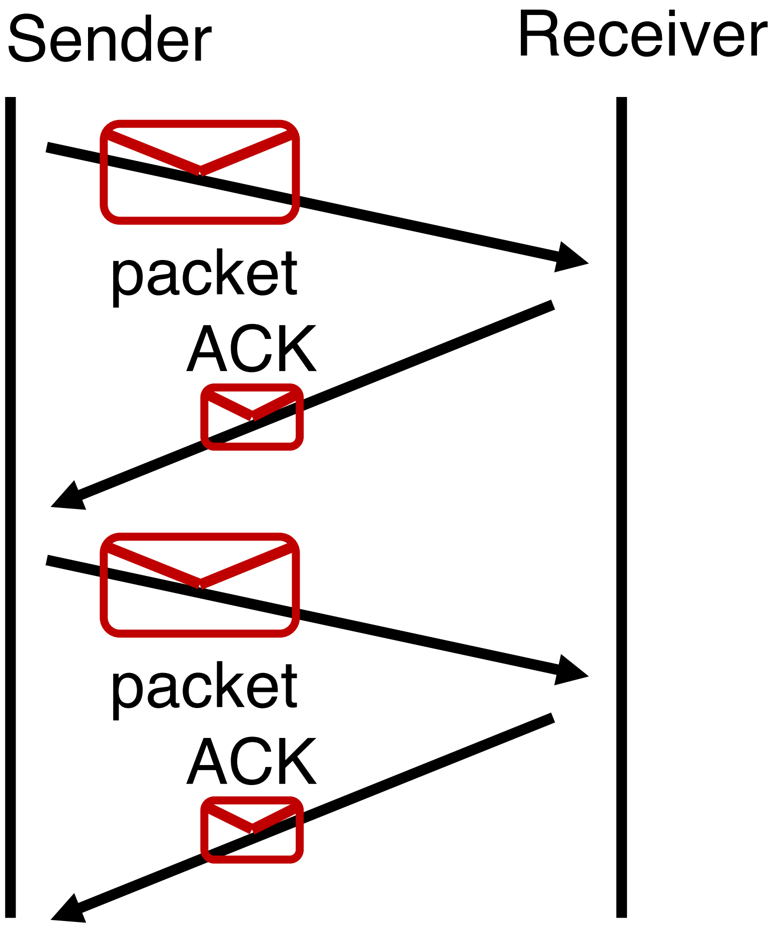
- TCP uses two mechanisms

Coping with packet loss: (1) ACK

- Key idea: Receiver returns an acknowledgment (ACK) per packet sent

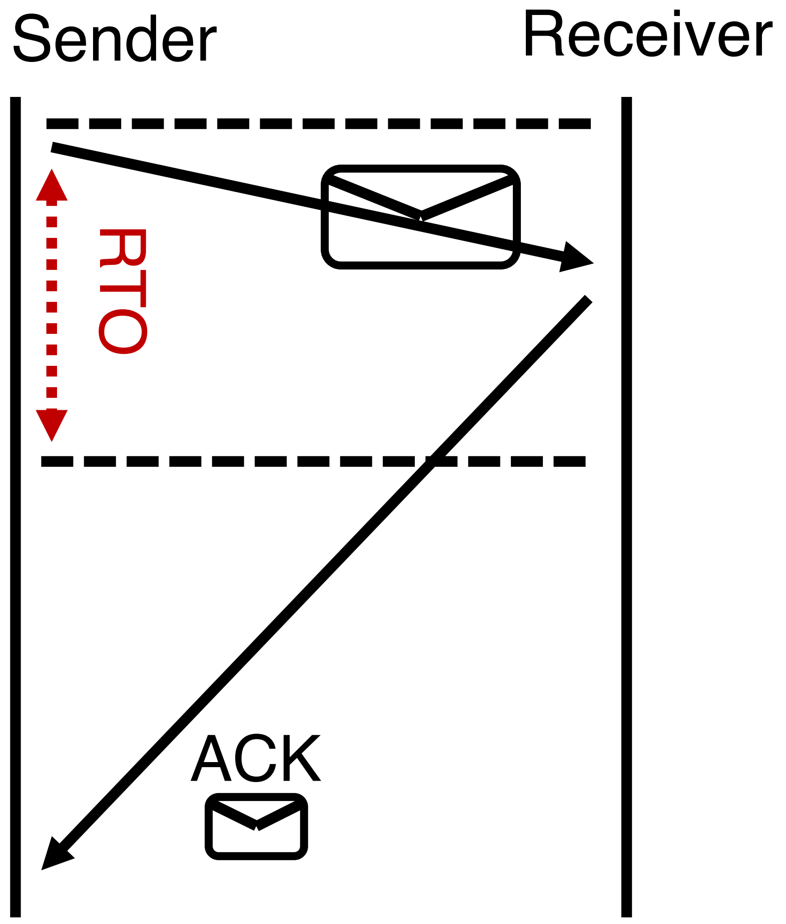
- If sender receives an ACK, it knows that the receiver got the packet.

- What if a packet was lost and ACK never arrives?



Coping with packet loss: (2) RTO

- Key idea: Wait for a duration of time (called retransmission timeout or RTO) before re-sending the packet
- In TCP, the onus is on the sender to retransmit lost data when ACKs are not received
- Retransmission works also if ACKs are lost or delayed

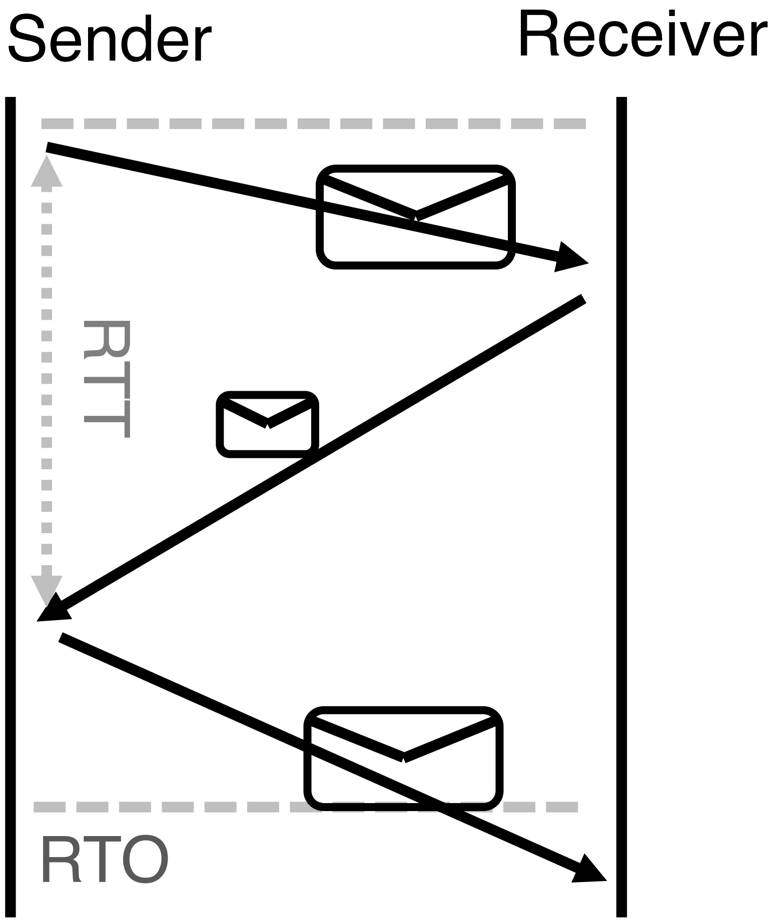


Sending one packet per ACK enough?

- Should sender wait for an ACK before sending another packet?

- Consider:
    - Round-trip-time: 100 milliseconds
    - Packet size: 12,000 bits
    - Link rate: 12 Mega bits/s
    - Suppose no packets are dropped

- At what rate is the sender getting data across to the receiver?



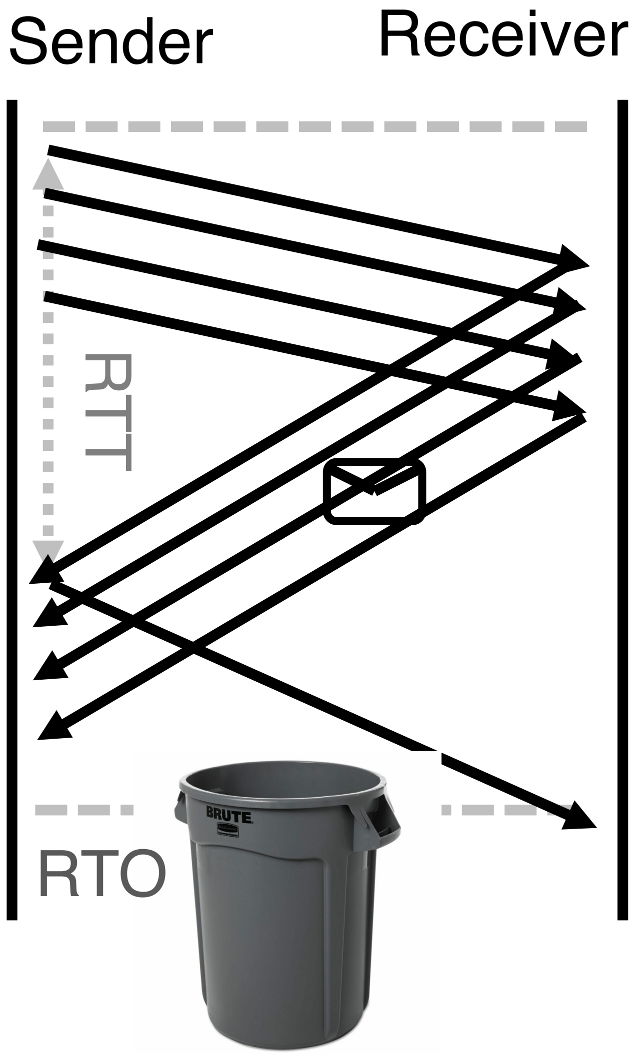
120 Kilo bit/s (1% of link rate)

Amount of “in-flight” data

- We term the amount of unACKed data as data “in flight”
- With just one packet in flight, the data rate is limited by the packet delay (RTT) rather than available bandwidth (link rate)
- Idea: Keep many packets in flight!
- More packets in flight improves throughput

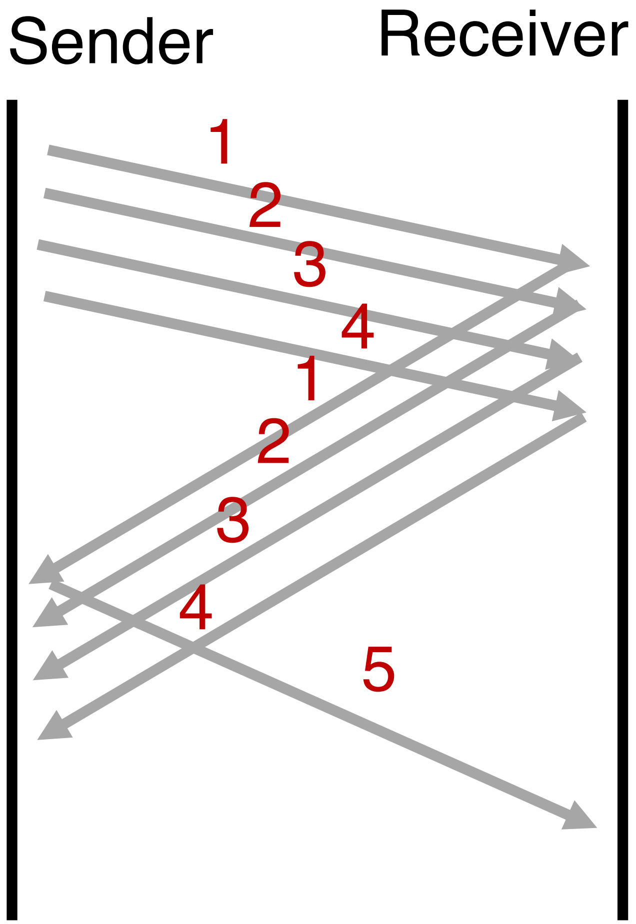
Keeping many packets in flight

- In our example before, if there are, say 4 packets in flight, throughput is 480 Kbits/s!
- We just improved the throughput 4 times by keeping 4 packets in flight
- Trouble: what if some packets (or ACKs) are dropped?
- How should the sender retransmit?



Keeping track of packets (and ACKs)

- Every packet contains a sequence number
  - (In reality, every byte has a sequence number)
- ACK echoes the sequence number of the packet that is acknowledged
- If a packet is dropped, should the receiver ACK subsequent packets?
  - If so, with what sequence number?

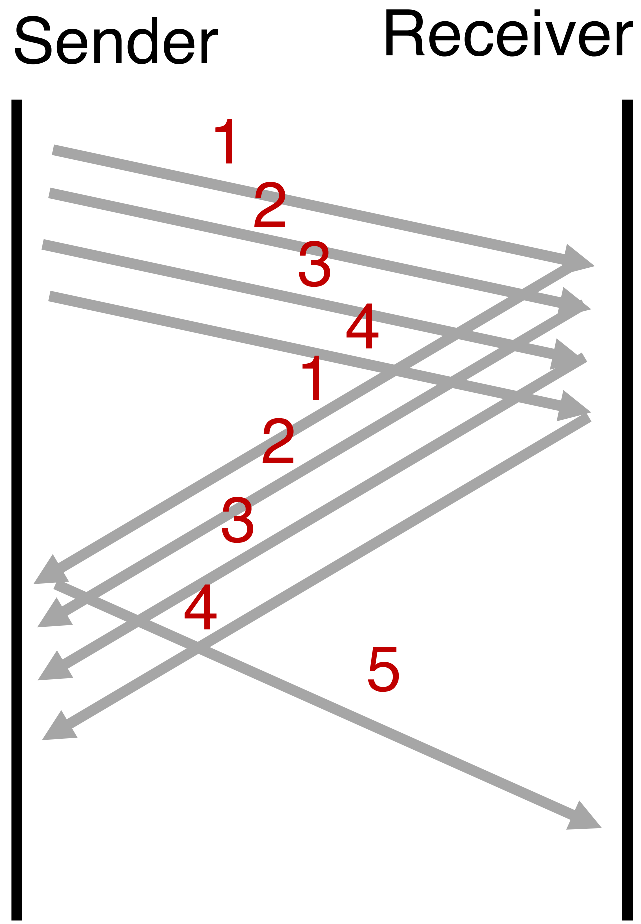


Keeping track of packets (and ACKs)

- Cumulative ACKs: ACK the latest seq# up to which all packets received
- Selective ACKs: return one cumulative seq# and ranges of other seq# received

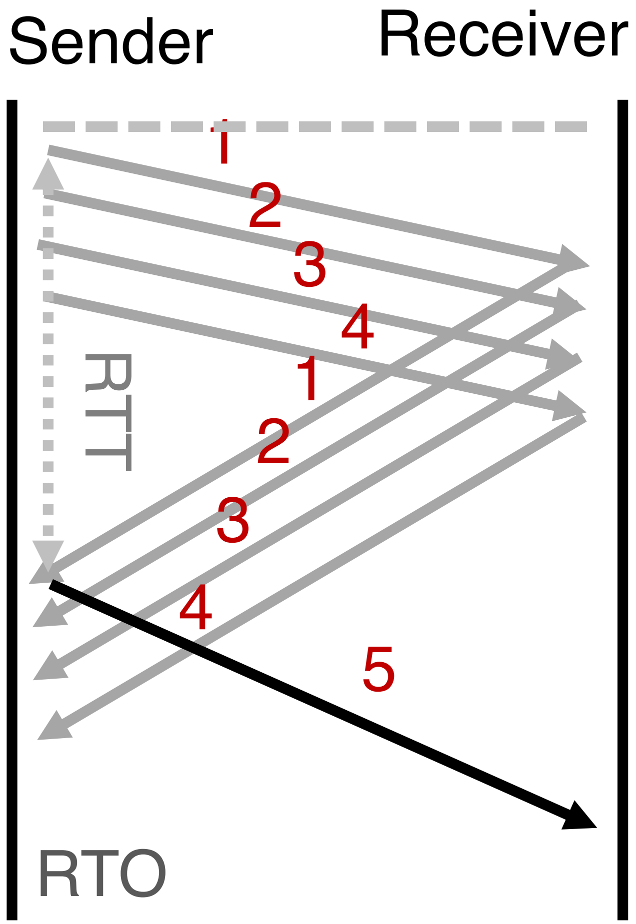
- Sender retransmits those packets whose sequence numbers haven't been ACKed

- What are the implications of selective vs. cumulative ACKs here?



How should the RTO be set?

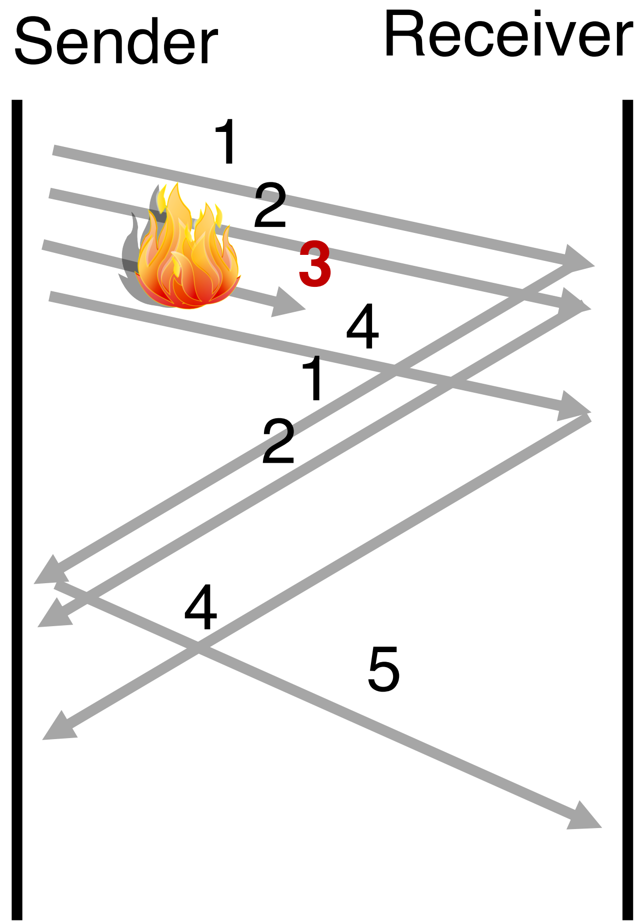
- Clearly, RTO must be related to RTT
  - But how exactly?



Ordered Delivery

Reordering packets at the receiver side

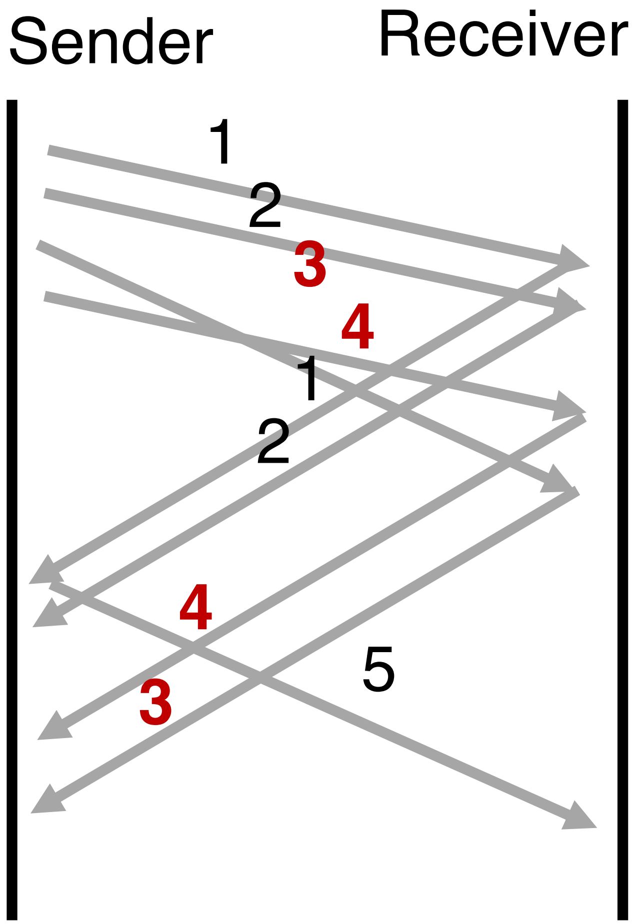
- Let's suppose receiver gets packets 1, 2, and 4, but not 3 (dropped)
- Suppose you're trying to download a Word document containing a report
- What would happen if transport at the receiver directly presents packets 1, 2, and 4 to the Word application?



Reordering at the receiver side

- Reordering can also happen due to packets taking different paths through a network

- Receiver needs a general strategy to ensure that data is presented to the application in the same order of sender side bytes pushed



Buffering at the receiver side