THE TRANSPORT LAYER

COMP 30650: NETWORKS AND INTERNET SYSTEMS

Dr. Gavin McArdle
Email: gavin.mcardle@ucd.ie
Office: A1.09 Computer Science

RECAP

We have been moving up through the layers

- Physical Layer
- Link Layer
- Network Layer

TODAY'S PLAN

Transport layer

- Connection Establishment
- Connection Release

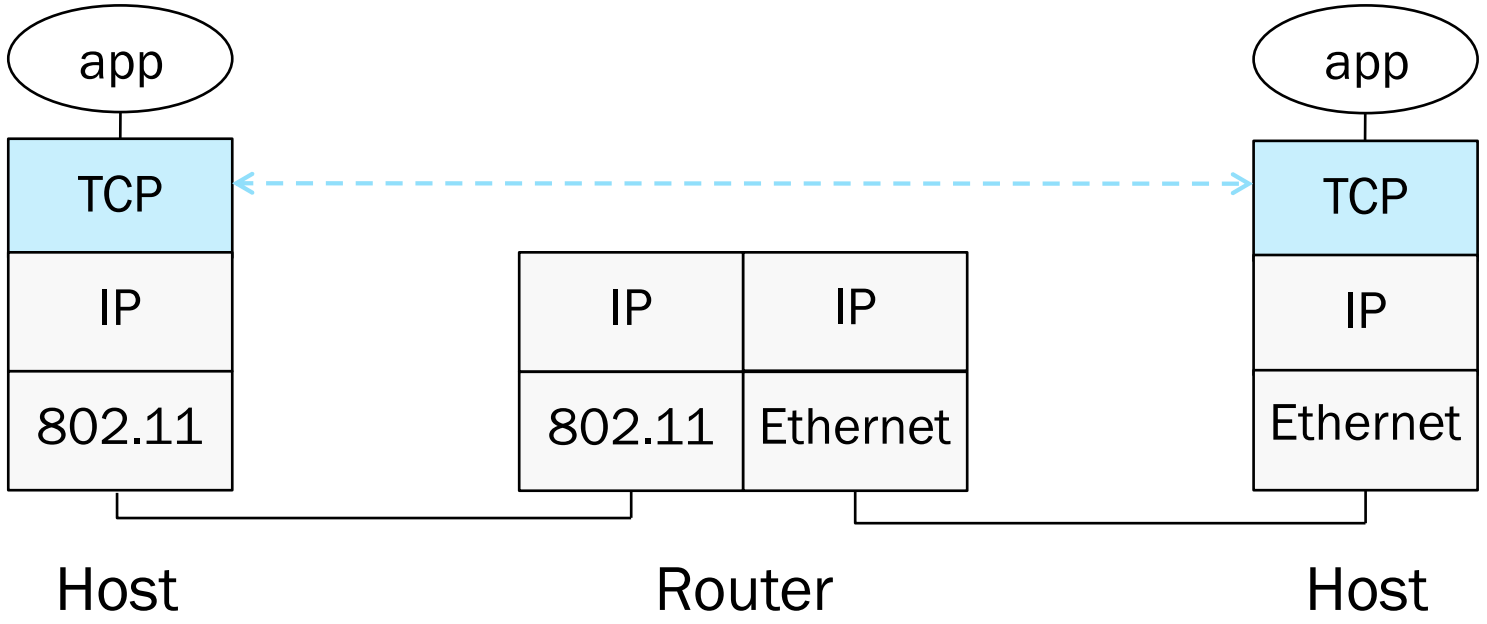
Sliding Window

TRANSPORT LAYER

- It is responsible for **end-to-end** communication over a network
- It provides logical communication between **application processes** running on different hosts within a layered architecture of protocols.
- It builds on the network layer to deliver data across networks for applications with the desired **reliability or quality**

RECALL

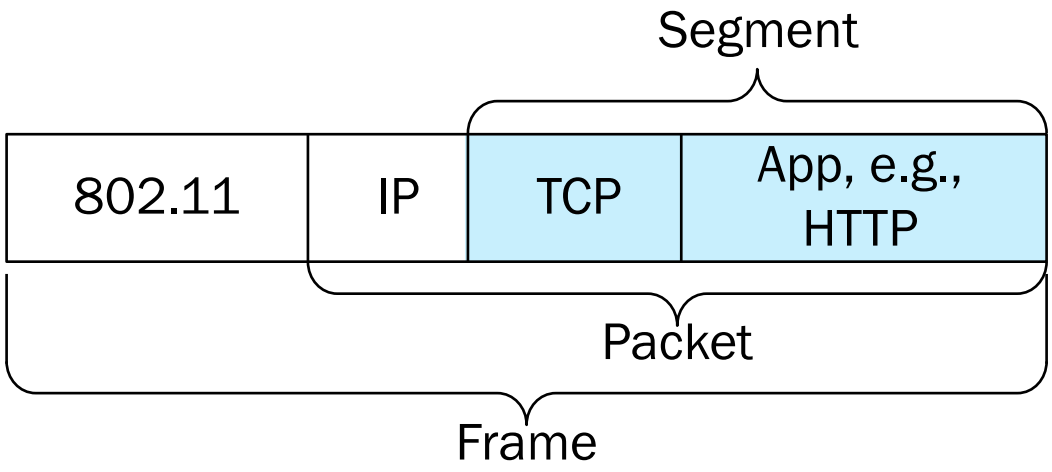
Transport layer provides end-to-end connectivity across the network



UNIT OF DATA

Segments carry application data across the network

Segments are carried within packets within frames



TRANSPORT LAYER SERVICES

Provide different kinds of data delivery across the network to applications

	Unreliable	Reliable
Messages	Datagrams (UDP)	
Bytestream		Streams (TCP)

COMPARISON OF INTERNET TRANSPORTS

TCP is full-featured, UDP is a glorified packet

TCP (Streams)	UDP (Datagrams)
Connections	Datagrams
Bytes are delivered once, reliably, and in order	Messages may be lost, reordered, duplicated
Arbitrary length content	Limited message size
Flow control matches sender to receiver	Can send regardless of receiver state
Congestion control matches sender to network	Can send regardless of network state

SOCKETS

Simple abstraction to use the network

- A **socket** is one **endpoint** of a two-way communication link between two programs running on the **network**. A **socket** is bound to a **port number** so that the TCP layer can identify the application that data is destined to be sent to. An endpoint is a combination of an IP address and a port number.

Supports both Internet transport services (Streams and Datagrams)

PORTS

- An application process is identified by the tuple IP address, protocol, and port
  - Ports are 16-bit integers representing local “mailboxes”
- Servers often bind to “well-known ports”
- Applications are given port numbers by the OS as required

SOME WELL-KNOWN PORTS

Port	Protocol	Use
20, 21	FTP	File transfer
22	SSH	Remote login, replacement for Telnet
25	SMTP	Email
80	HTTP	World Wide Web
110	POP-3	Remote email access
143	IMAP	Remote email access
443	HTTPS	Secure Web (HTTP over SSL/TLS)
543	RTSP	Media player control
631	IPP	Printer sharing

USER DATAGRAM PROTOCOL (UDP)

Used by apps that don't want reliability or bytestreams

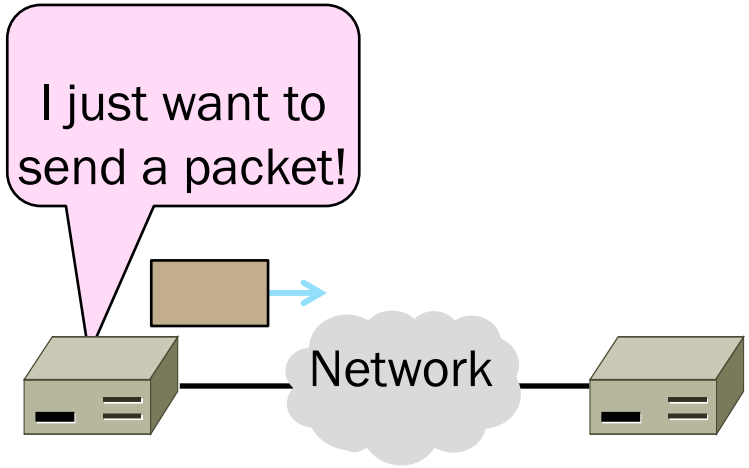
- **VOIP,** Voice-over-IP (unreliable)
- **DNS**, RPC (message-oriented)
- **DHCP** (bootstrapping)

(If application wants reliability and messages then it has work to do!)

USER DATAGRAM PROTOCOL (UDP)

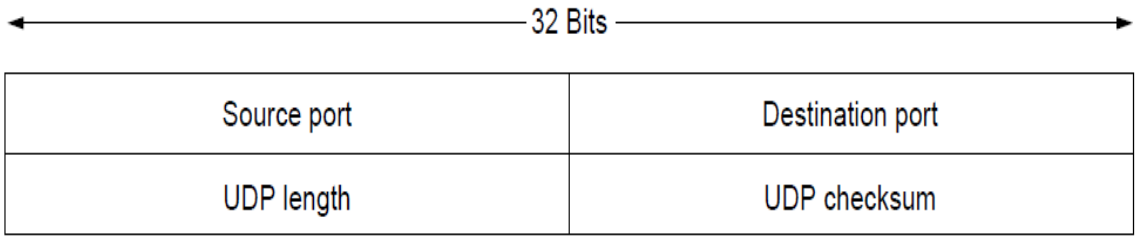
Sending messages with UDP

- A small step above a network layer and packets



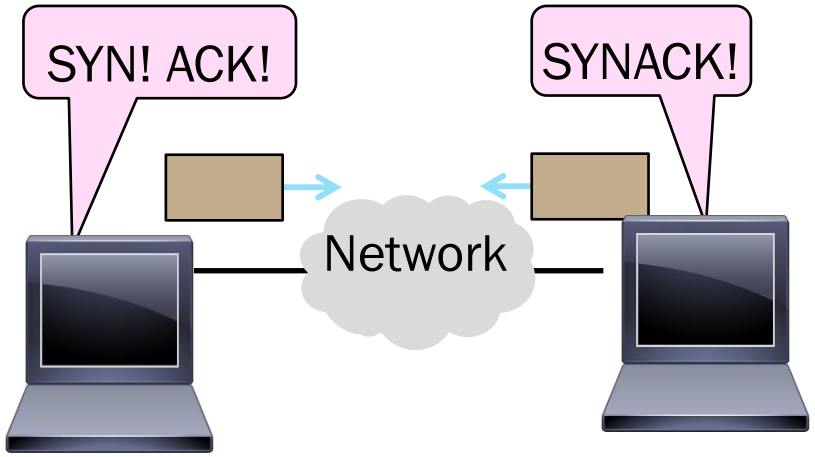
UDP HEADER

- Uses ports to identify sending and receiving application processes
- Datagram length up to 64K
- Checksum (16 bits) for reliability



ADDING RELIABILITY WITH CONNECTIONS

How to set up connections with Transport Control Protocol (TCP)



CONNECTION ESTABLISHMENT

Both sender and receiver must be ready before we start the transfer of data

- Need to agree on a set of parameters
  - The Maximum **Segment** Size (MSS)
    - The default TCP Maximum Segment Size is 536 octets
  - Sequence Numbers

This is signaling

- It sets up state at the endpoints

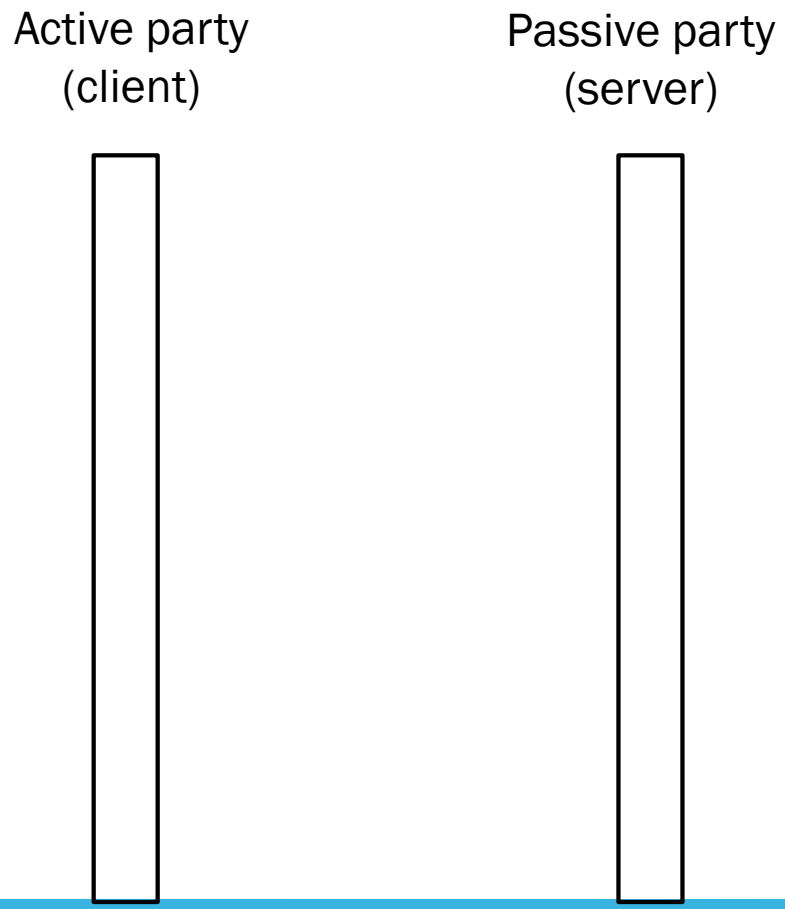
THREE-WAY HANDSHAKE

Used in TCP; opens connection for data in both directions

Each side queries the other with a fresh Initial Sequence Number (ISN)

- Sends on a SYNchronize segment
- Echo on an ACKnowledge segment

Chosen to be robust even against delayed duplicates

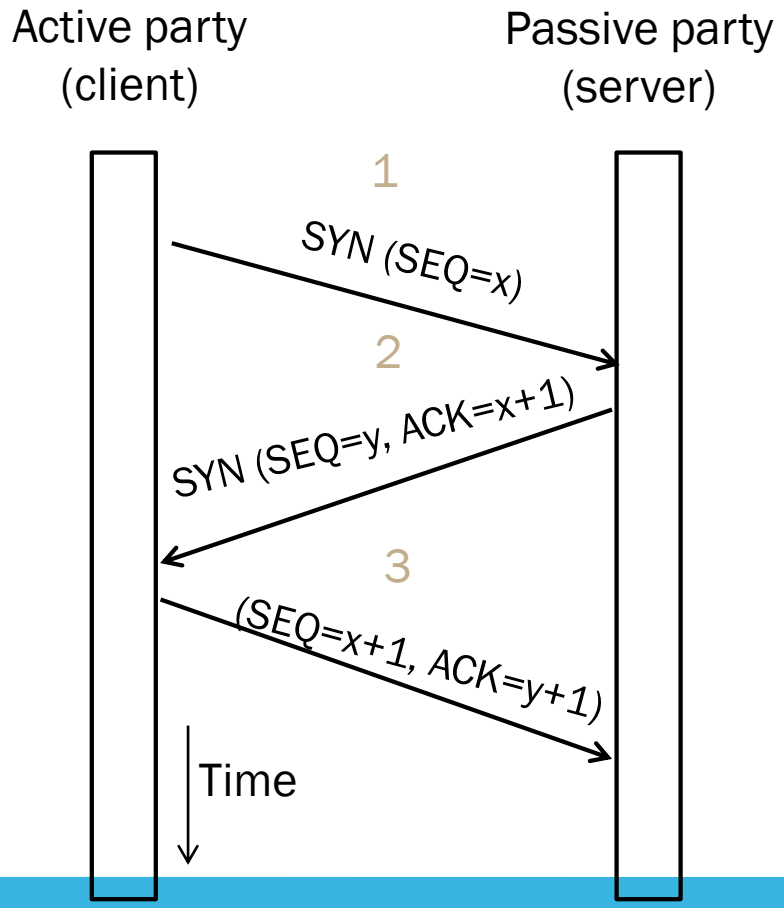


THREE-WAY HANDSHAKE

Three steps:

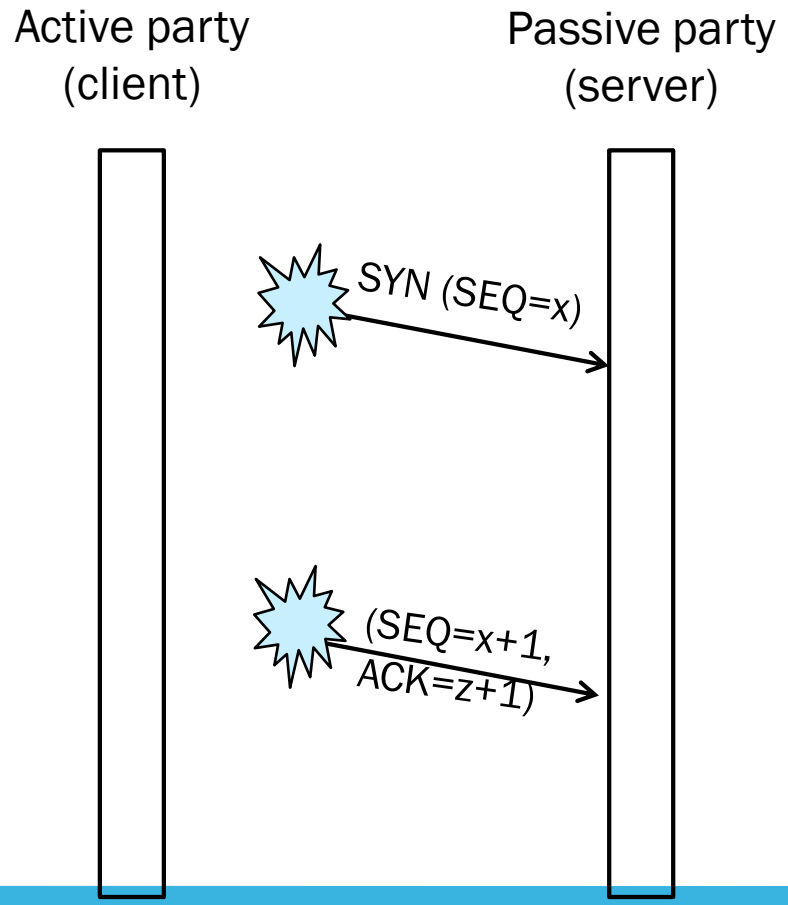
- Client sends SYN(x)
- Server replies with SYN(y)ACK(x+1)
- Client replies with ACK(y+1)
- SYNs are retransmitted if lost

Sequence and Acknowledgment numbers carried on <u>all</u> further segments



THREE-WAY HANDSHAKE

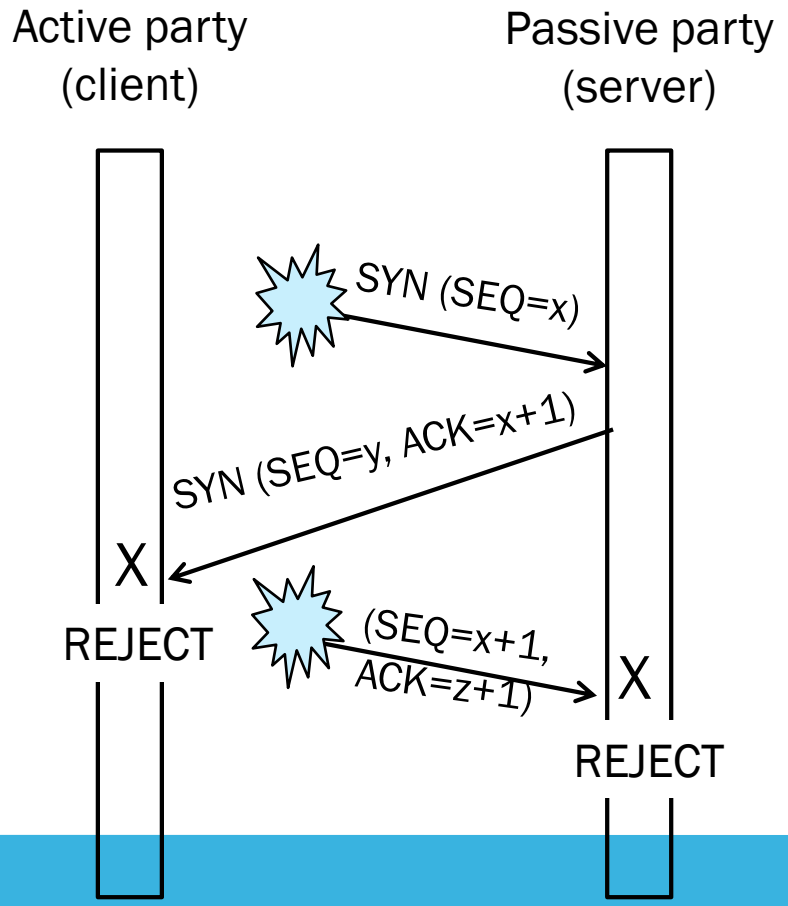
Suppose delayed, duplicate copies of the SYN and ACK arrive at the server



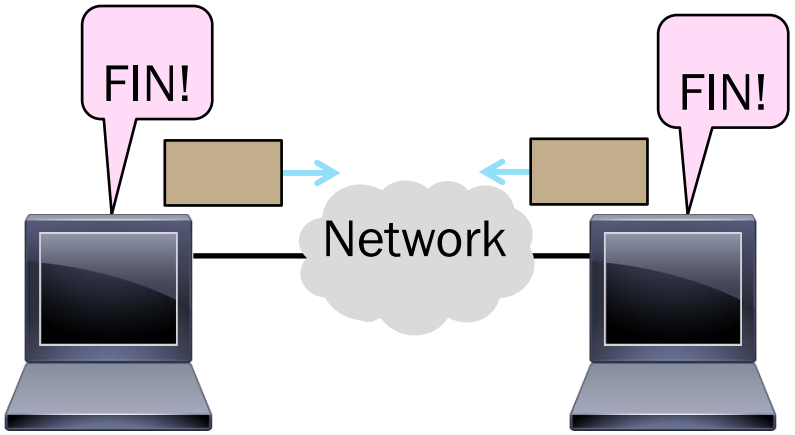
THREE-WAY HANDSHAKE

Suppose delayed, duplicate copies of the SYN and ACK arrive at the server!

Connection will be cleanly rejected on both sides



CONNECTION RELEASE
How TCP releases connections



CONNECTION RELEASE

Orderly release by both parties when done

- Delivers all pending data and “hangs up”
- Cleans up state in sender and receiver

Key problem is to provide reliability while releasing

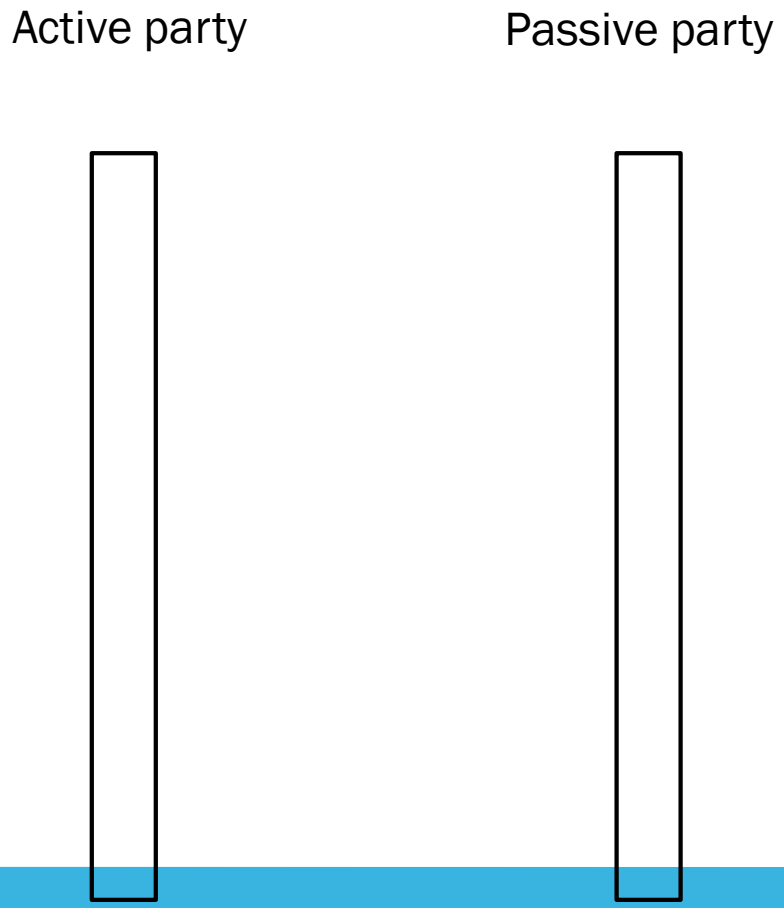
- TCP uses a “symmetric” close in which both sides shutdown independently

TCP CONNECTION RELEASE

Two steps:

- Active sends FIN(x), passive ACKs
- Passive sends FIN(y), active ACKs
- FINs are retransmitted if lost

Each FIN/ACK closes one direction of data transfer

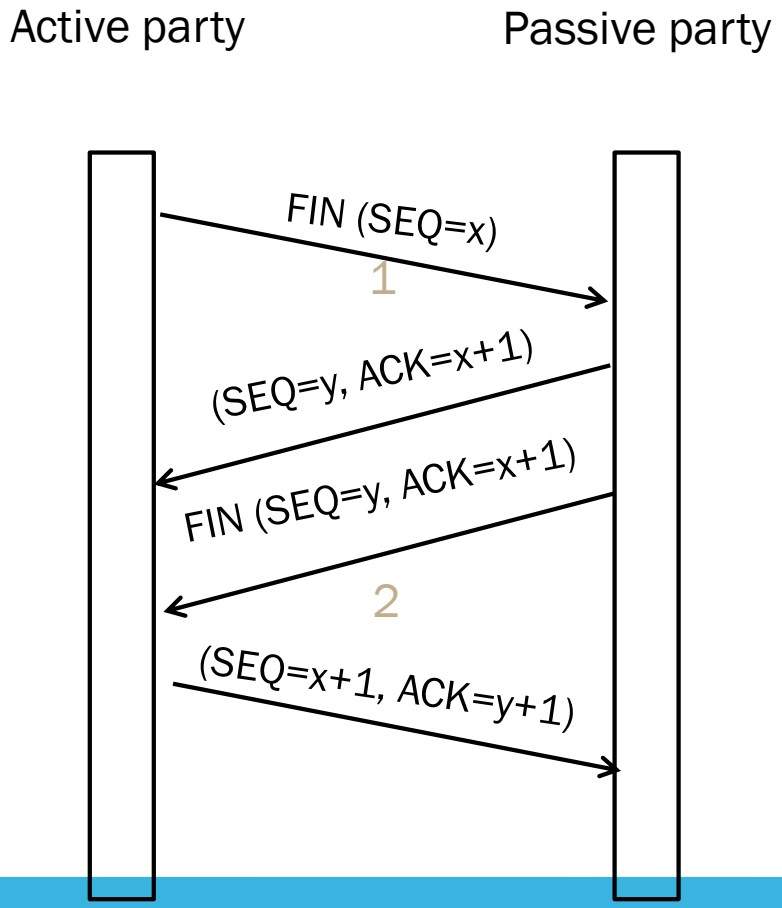


TCP CONNECTION RELEASE

Two steps:

- Active sends FIN(x), passive ACKs
- Passive sends FIN(y), active ACKs
- FINs are retransmitted if lost

Each FIN/ACK closes one direction of data transfer



TIME_WAIT STATE

We wait a long time at the end (two times the maximum segment lifetime of 60 seconds) after sending all segments and before completing the close

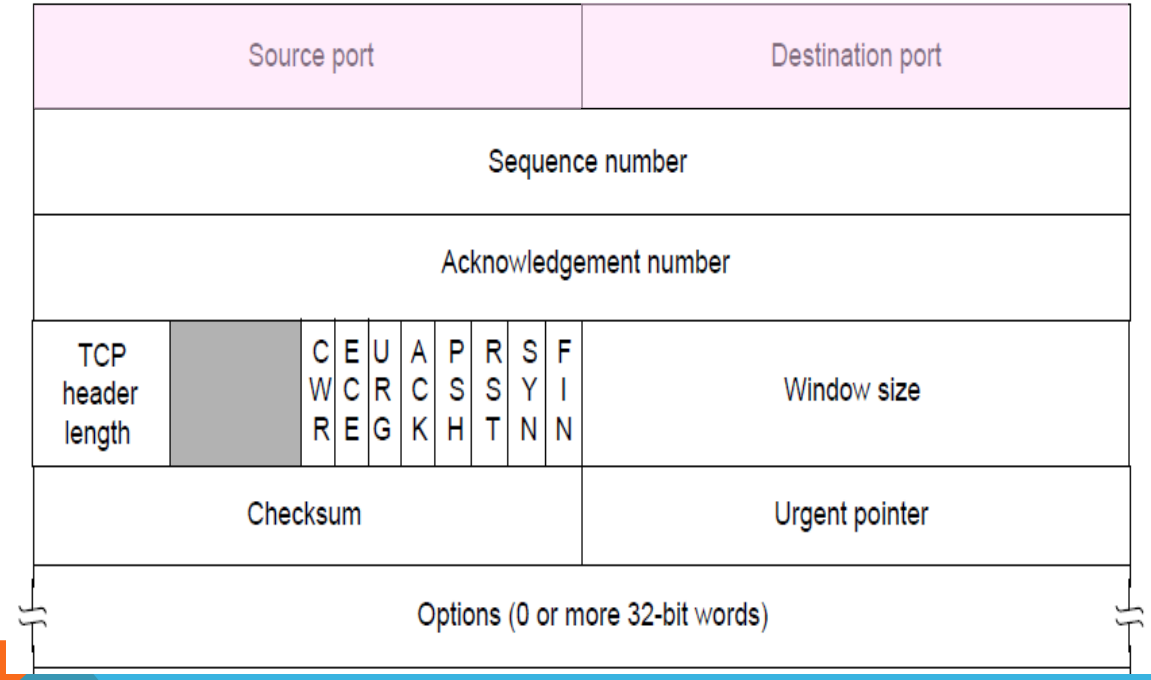
Why?

- ACK might have been lost, in which case FIN will be resent for an orderly close
- Could otherwise interfere with a subsequent connection

TCP HEADER

Ports identify apps (socket API)

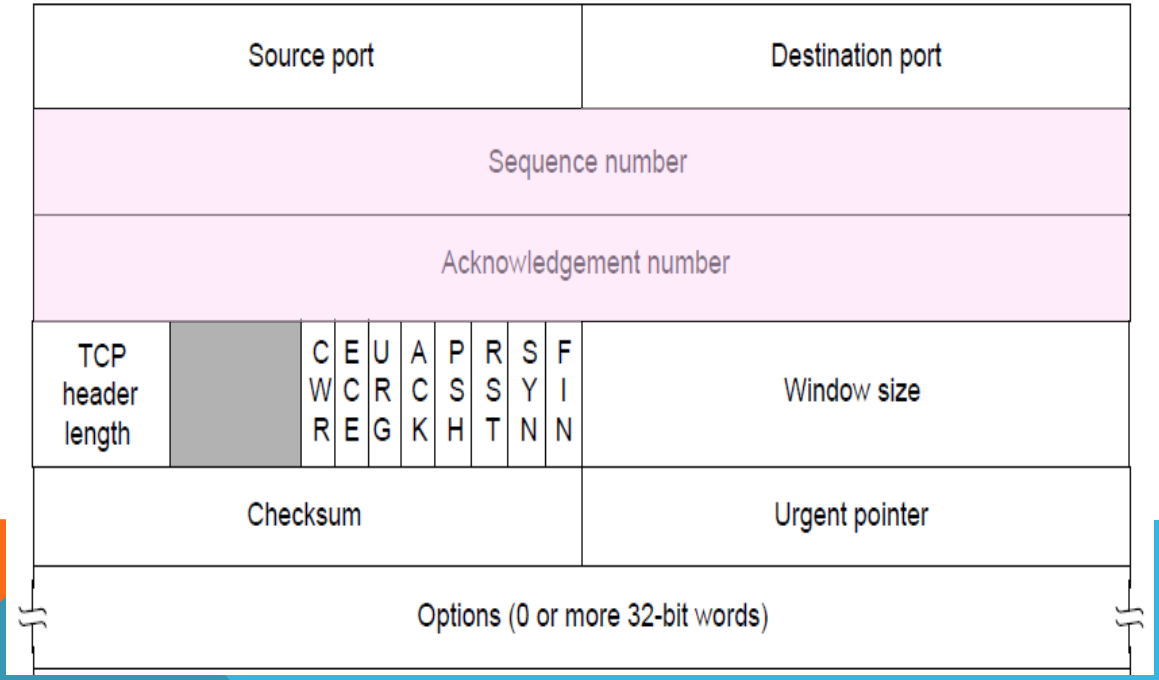
- 16-bit identifiers



TCP HEADER

SEQ/ACK used for sliding window

- Selective Repeat, with byte positions



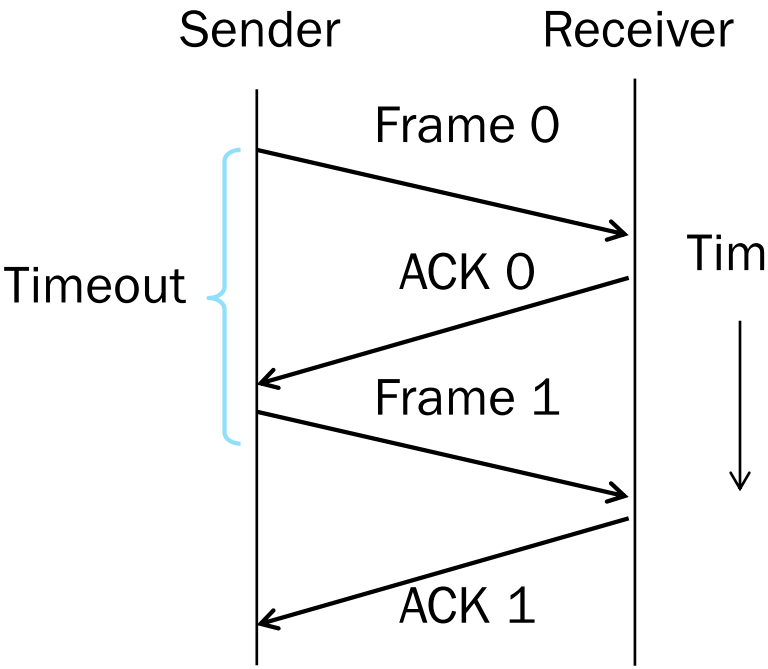
RECALL

ARQ with one message at a time is Stop-and-Wait (normal case below)

This is not very efficient

- Fine for LAN (only one frame will fit on Network)
- Not efficient for network paths with BD >> 1 packet

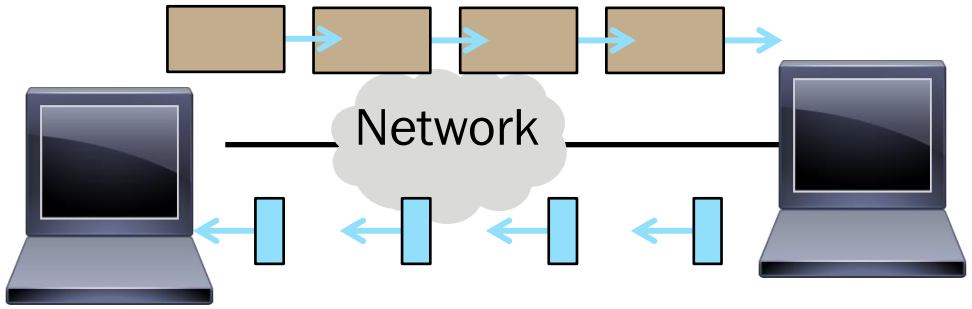
Need more than one message to be outstanding at a time.



SLIDING WIDOW

The sliding window algorithm

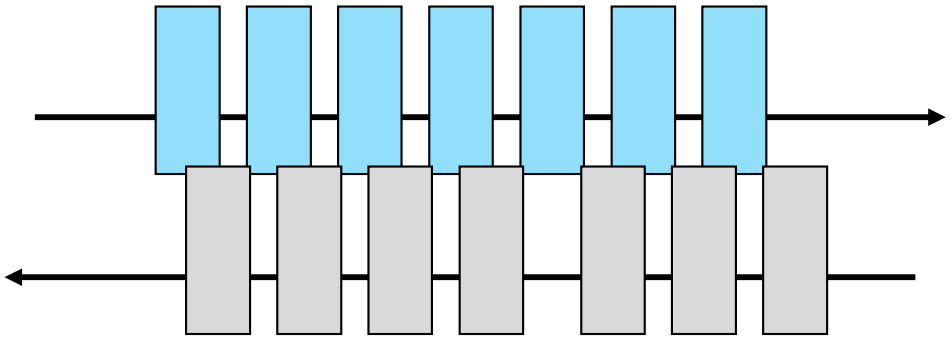
- Pipelining and reliability
- Builds on top Stop-and-Wait



SLIDING WINDOW

Generalization of stop-and-wait

- Allows W packets to be outstanding
- Can send W packets per Round Trip Time (RTT)
- Pipelining improves performance



SLIDING WINDOW PROTOCOL

Many variations, depending on how buffers, acknowledgements, and retransmissions are handled

Go-Back-N

- Simplest version, can be inefficient

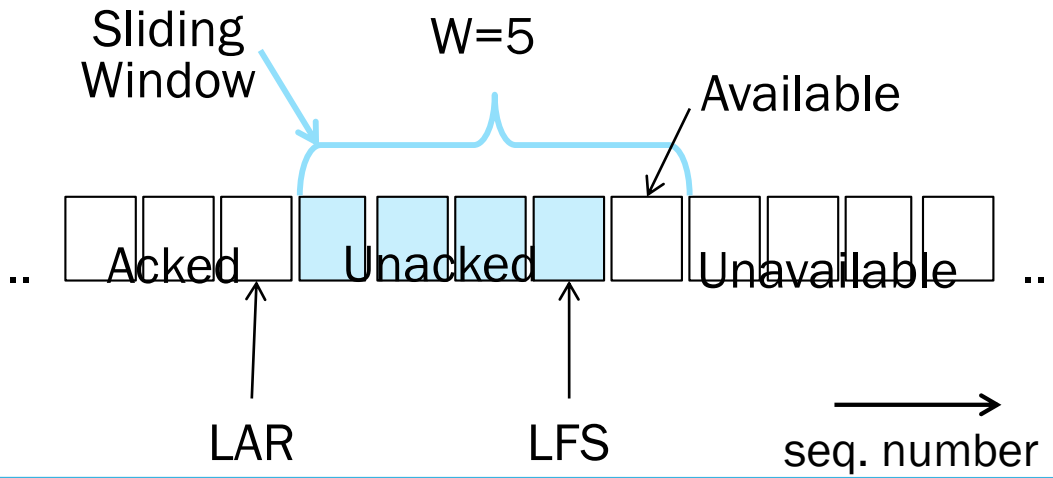
Selective Repeat

- More complex, better performance

SLIDING WINDOW – SENDER

Sender buffers up to W segments until they are acknowledged

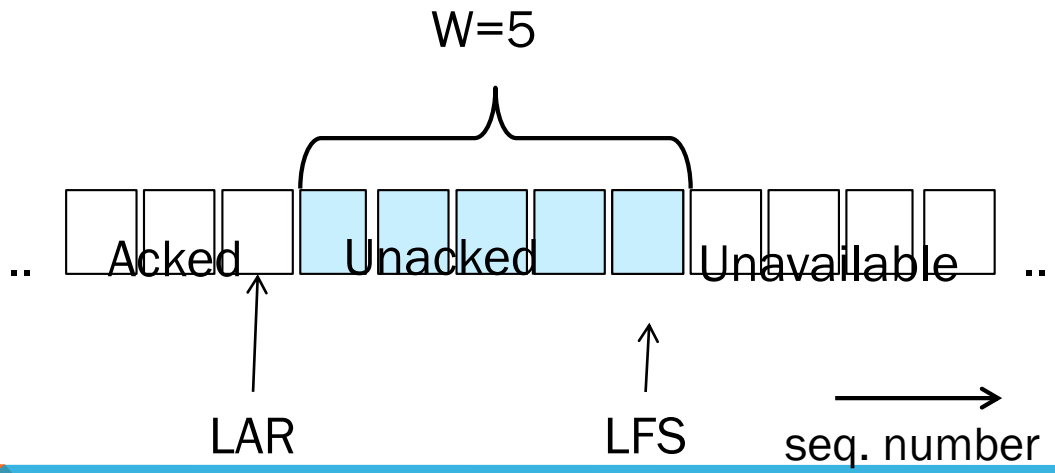
- LFS=LAST FRAME SENT, LAR=LAST ACK REC'D
- Sends while LFS – LAR ≤ W



SLIDING WINDOW – SENDER

Transport accepts another segment of data from the Application …

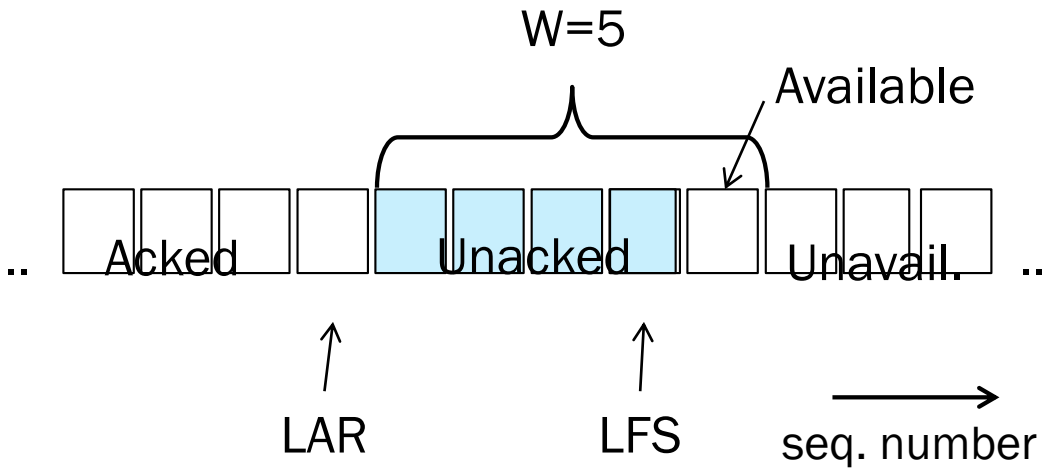
- Transport sends it (as LFS–LAR → 5)



SLIDING WINDOW – SENDER

Next higher ACK arrives from peer…

- Window advances, buffer is freed
- LFS–LAR → 4 (can send one more)



SLIDING WINDOW – GO-BACK-N

Receiver keeps only a single packet buffer for the next segment

- State variable, LAS = LAST ACK SENT

On receive:

- If sequence number is LAS+1, accept and pass it to app, update LAS, send ACK
- Otherwise discard (as out of order)

Go-Back-N sender uses a single timer to detect losses

- On timeout, resends buffered packets starting at LAR+1

SLIDING WINDOW – SELECTIVE REPEAT

Receiver passes data to app in order, and buffers out-of-order segments to reduce retransmissions

ACK conveys highest in-order segment, plus hints about out-of-order segments

TCP uses a selective repeat design;

SLIDING WINDOW – SELECTIVE REPEAT

Buffers W segments, keeps state variable,

LAS = LAST ACK SENT

On receive:

- Buffer segments [LAS+1, LAS+W]
- Pass up to app in-order segments from LAS+1, and update LAS
- Send ACK for LAS regardless
  - ACK the one passed to app

SELECTIVE REPEAT - RETRANSMISSIONS

The size of the sending and receiving windows must be equal

Selective Repeat sender uses a timer per *unacked* segment to detect losses

- On timeout for segment, resend it
- Hope to resend fewer segments