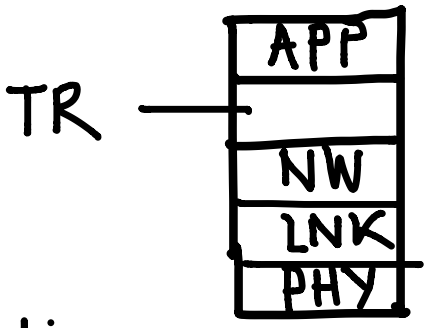
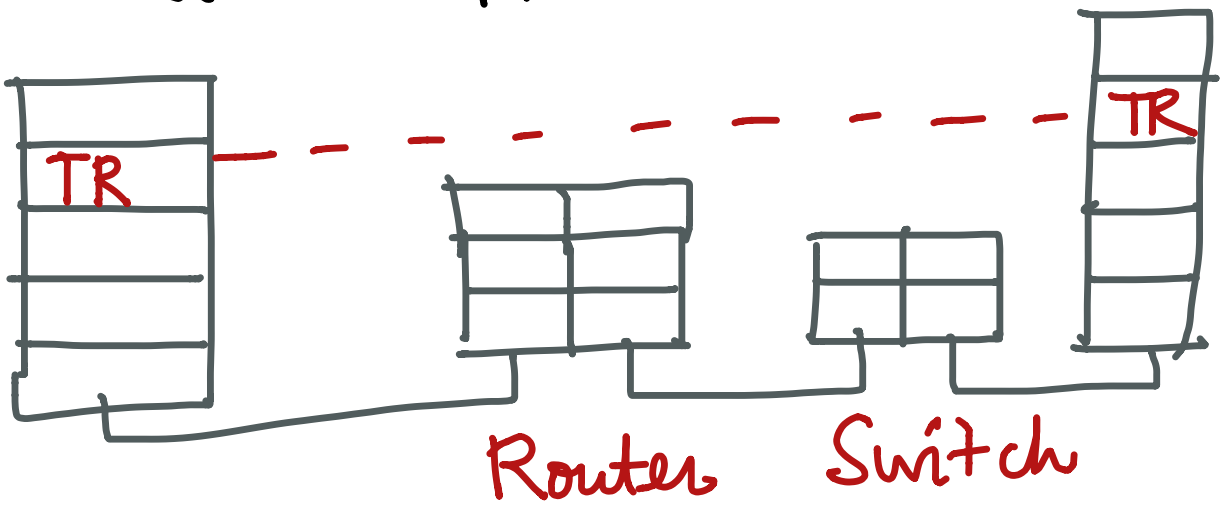
Transport Layer

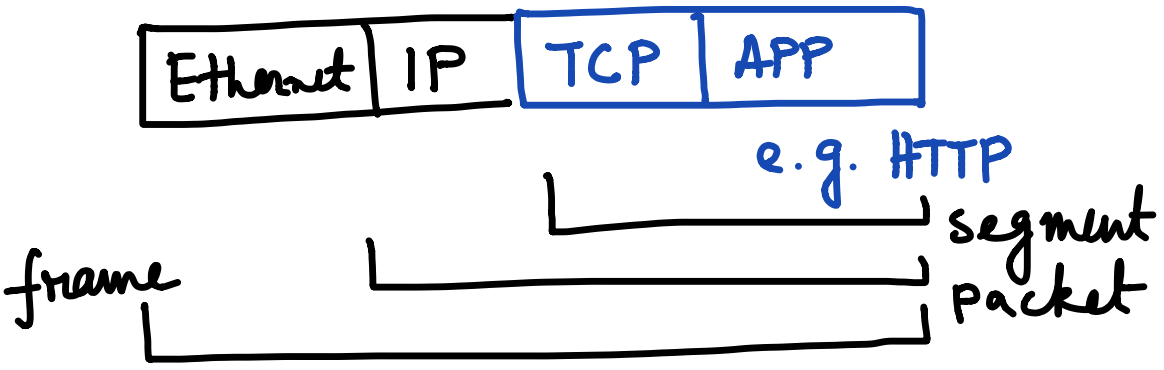


Task: deliver data end-to-end for different applications



Layer that gives end-to-end service.

Transport layer messages are "segments"



Transport Layer Services

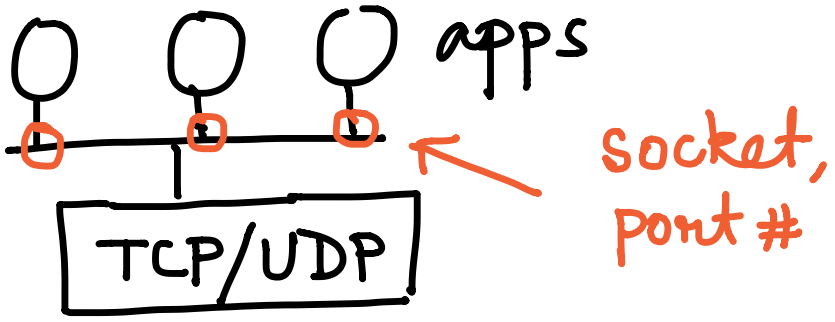
		Implemented via
Unreliable	messages	UDP
reliable	bytestreams	TCP

reliability does NOT mean it can't lose packets, but it is capable of handling such losses yet get the information across.

Comparison Telephone Verbal

TCP	UDP
Connection-oriented	connectionless
Arbitrary length	limited size
flow control used	sends regardless of receiver status
Congestion control used	sends regardless of network status

Sockets provide a simple abstraction of the network to the applications



Socket API uses object called sockets

can be used for streams and datagrams → same API

Socket API functions (Recap)

Socket – create a new communication endpoint

Bind – a port to a socket

Listen – willing to connect

Accept – establish connection (passive)

Connect – " (active)

Send (to) - Send data
Receive (from) - receive data
Close

Ports

Application identifier =
(IPaddress, protocol, port)

ports are numbered with 16 bits

Analogous to email mailboxes

Servers bind to special ports
<1024, requires admin privileges

Clients bind to "ephemeral" ports
does not need special previleges

Well known ports for server applications

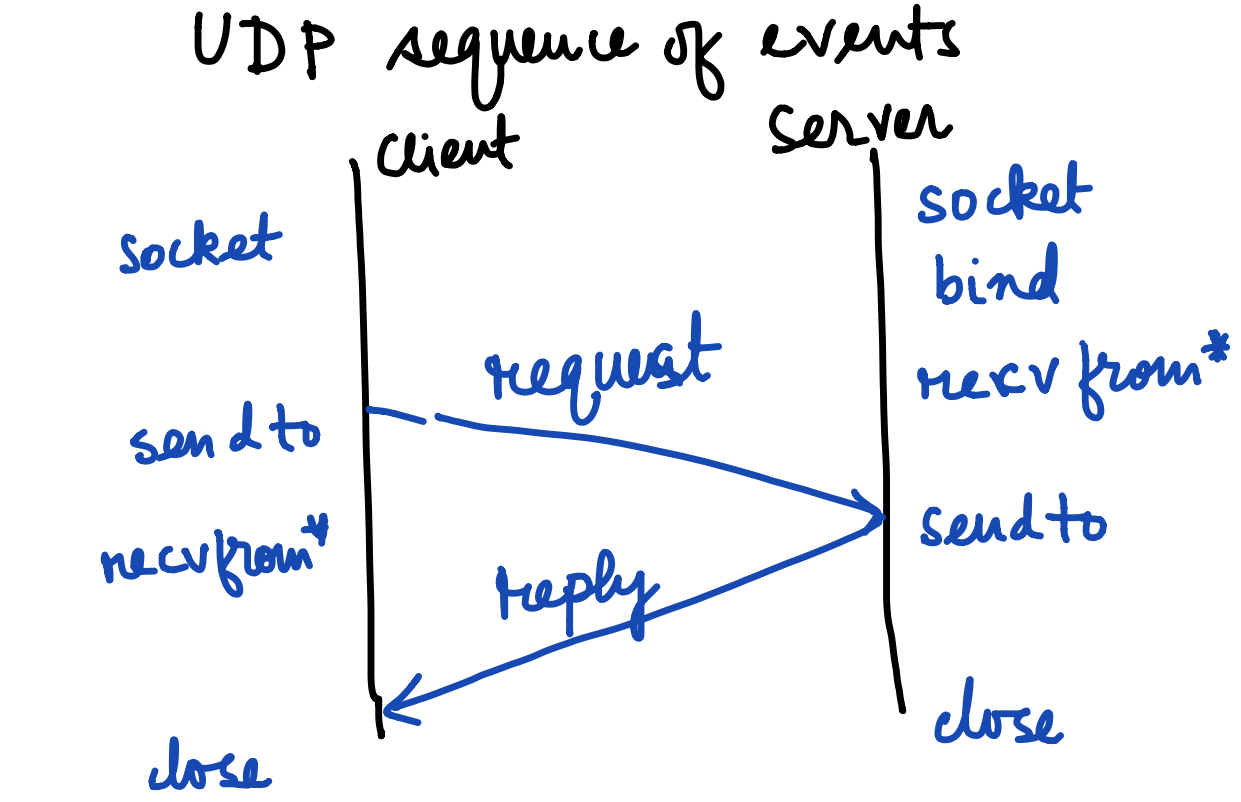
20, 21 – FTP
80 – HTTP
22 – SSH
25 – SMTP
.....
443 – HTTPS

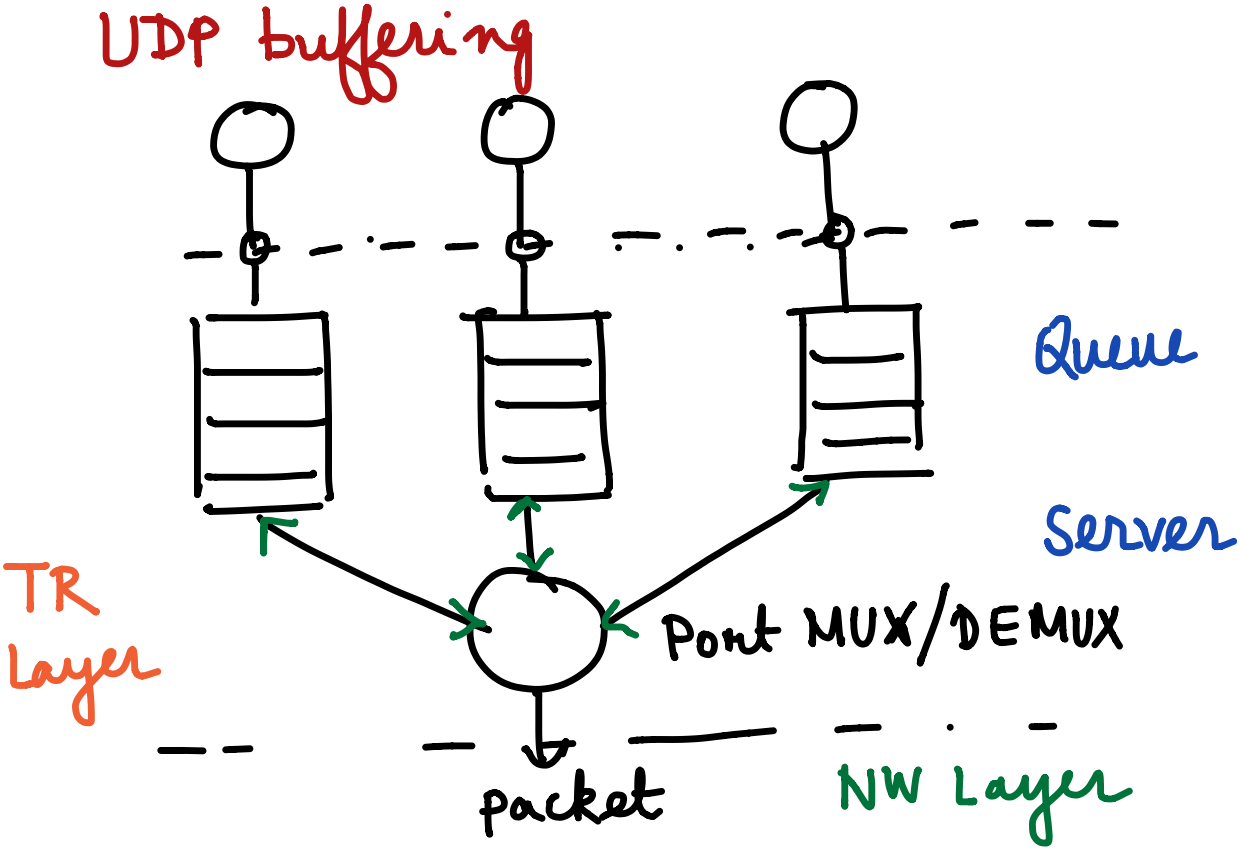
User Datagram Protocol (UDP)

Features
- datagram/message (contrast to bytestream)

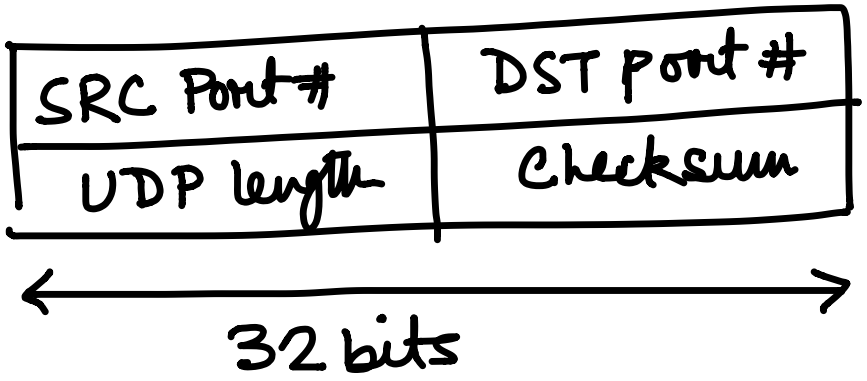
used for applications that don't need reliability of stream

VoIP – no time to recover
DNS – can request multiple times
DHCP – bootstrapping


UDP sequence of events
Client
Server
socket
socket
bind
request
recv from*
send to
send to
recvfrom*
reply
close
close


UDP buffering
Queue
Server
TR
Layer
Port MUX/DEMUX
packet
NW Layer

UDP header



Connection oriented service

- Connection Establishment
- Connection Release

] Telephone call.

process to setup the infrastructure to send and receive data (end to end)

Step 1

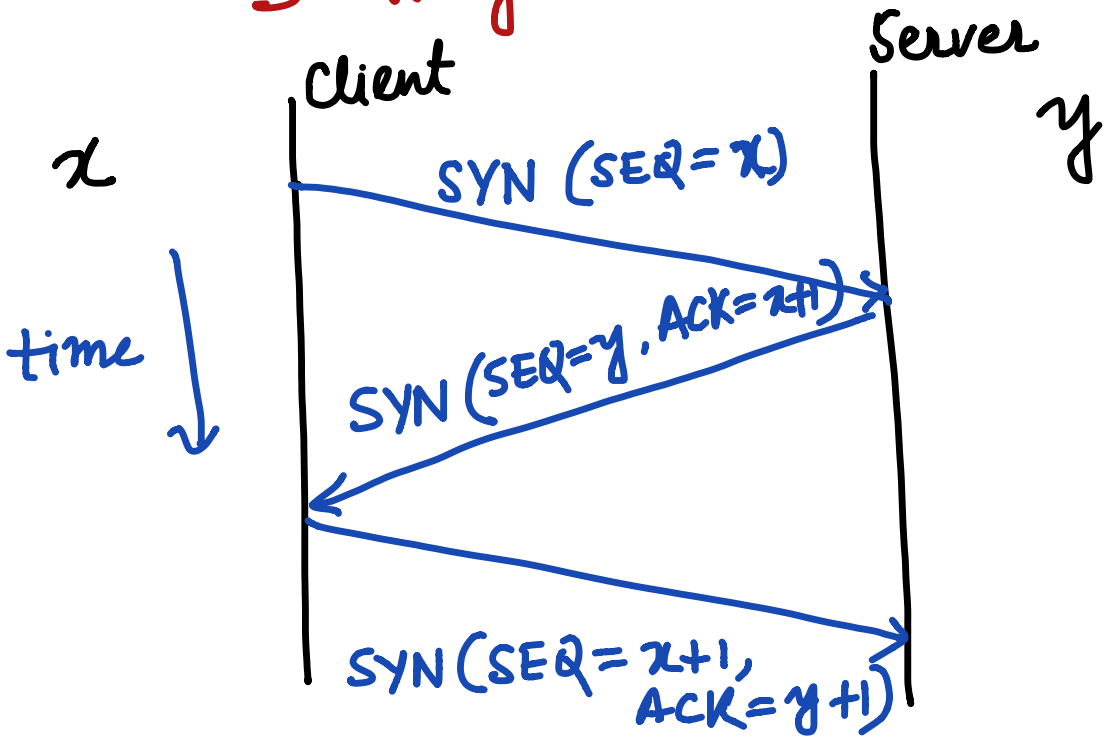
Agreement on the parameters

e.g., max segment size

— SIGNALING

TRANSMISSION CONTROL PROTOCOL

3-way handshake



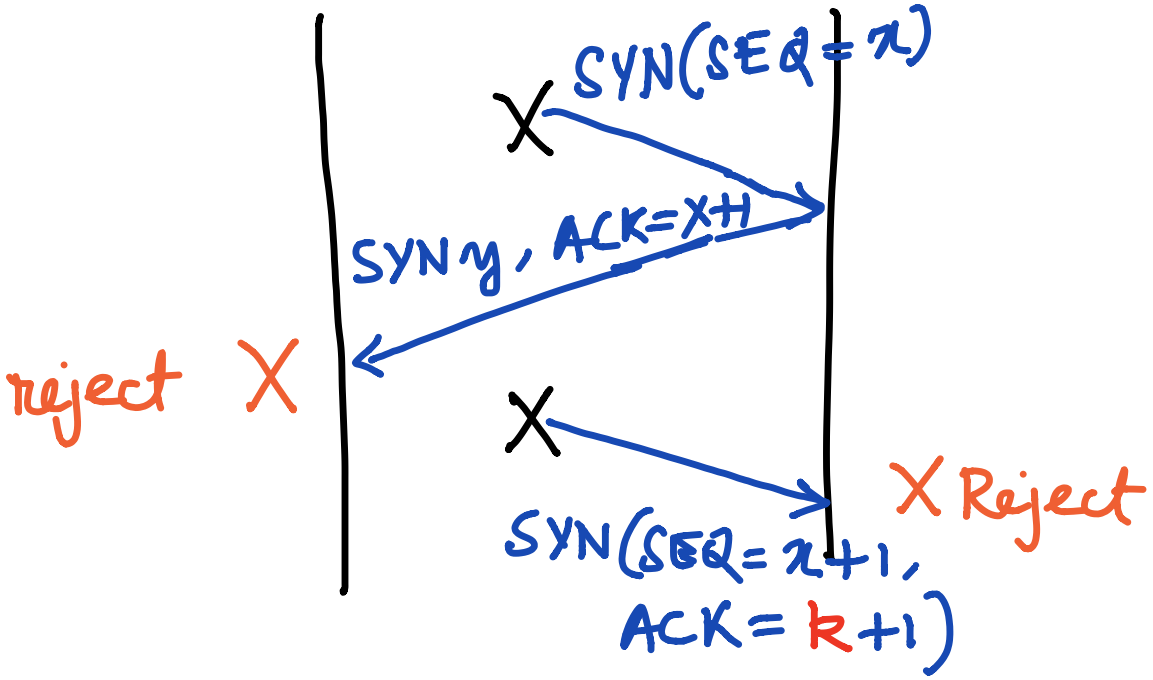
Both the parties pick an arbitrary sequence number before setting up a connection

- SEQ, ACK are fields of TCP header
- SEQ numbers can be arbitrary

Robustness

Easy case – any of the packets lost in transit
after a timeout, duplicate packet is sent
– lost packet doesn't reappear

Harder case – lost packets reappear



Also, both sides can open a TCP connection

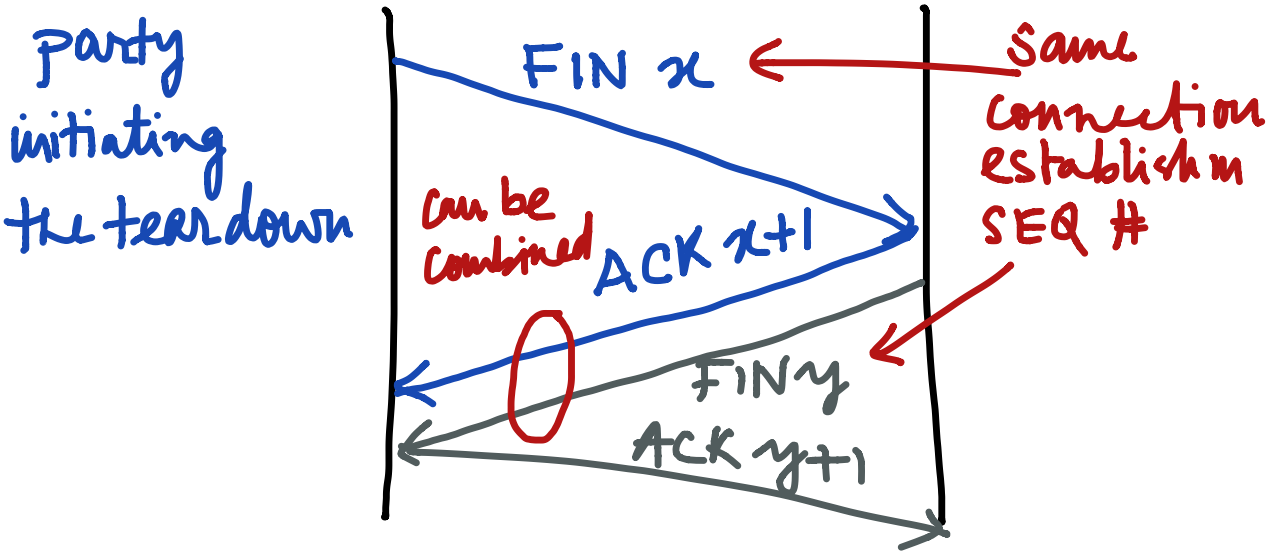
need to keep track of the sequence and ack numbers
- finally after 3-way handshake both connections with diff SEQ & ACK #'s are established.

Connection Release

Why? need to successfully deliver all data before tear down
- also clean up the states.

orderly manner

TCP symmetric release

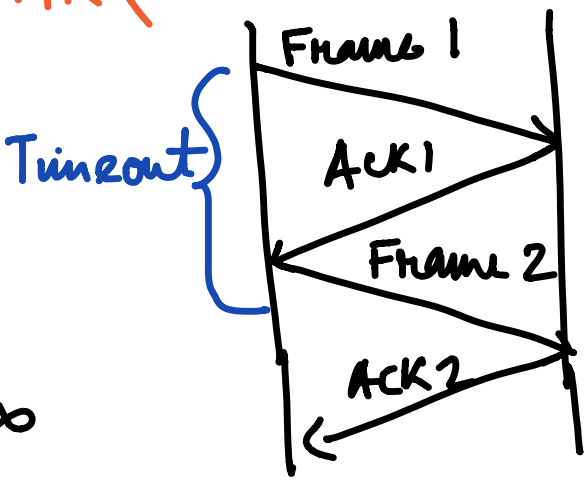


TCP sliding window algorithm

Widely known for guaranteeing reliability.

- modified stop-and-wait ARQ

Recap: Stop-Wait ARQ



Good for small network delay is negligible but the wastage is too high for larger NW.

Recall the numerical example from last time.

Sliding window

- allows W packets to be outstanding (or frames/segment)
- Can send W packets in one RTT
- Improves BW usage
- needs $W = 2BD$ to fill network path

Ex. $R = 1$ Mbps, $D = 50$ ms

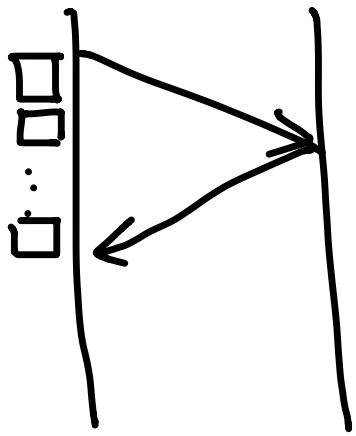
$W = 100$ Kbits ~ 10 packets

if a packet is of 10 Kbits

if $R = 10$ Mbps

100 packets

Changing the window.



More detail:

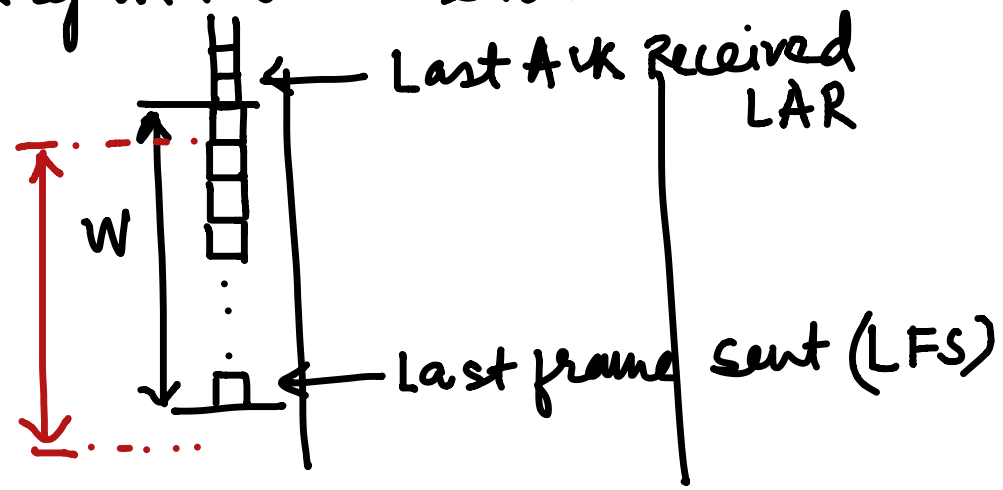
many variations of sliding window available depending on

- Buffer size, ACK, Retransmission

Go back N

Selective repeat

Sliding window - Sender side



IF LFS - LAR = W

WAIT until next ACK

ELSE IF LFS - LAR < W

SEND one packet

WHEN ACK RECEIVED

SLIDE WINDOW

Receiver Side

Go-back- N

Last ACK sent (LAS)

packet seq

Waiting for the next

one buffer

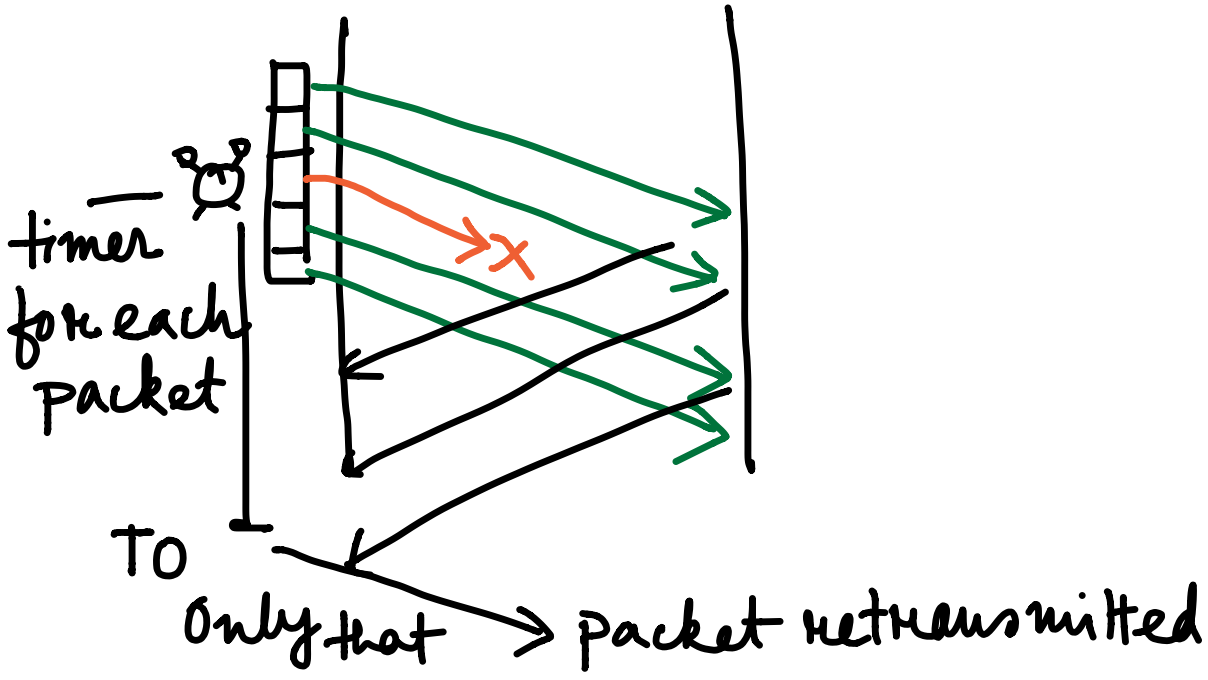
This protocol discards any out of order packets

Q: If one packet was lost, but later ones went successfully?

Go-back-n will repeat sending them all – hence the name (from LAR+1)

Selective repeat

- larger buffer – W (same as window)
- keeps track of all packets received
- sends ACKs accordingly



Seq. numbers needed

Stop and Wait – only 1 bit

Selective Repeat – W for current window packets and W for past packets' ACKs

= 2W

Go-back-N = W + 1

← only for one outstanding ACK