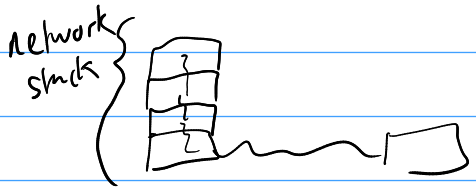
CS 3214 network protocols

Layering

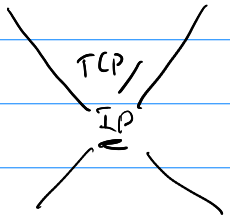


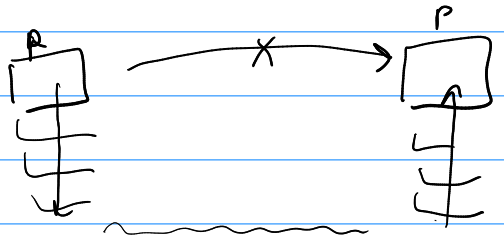
each layer can be independently maintained / upgraded

Internet Protocol stack

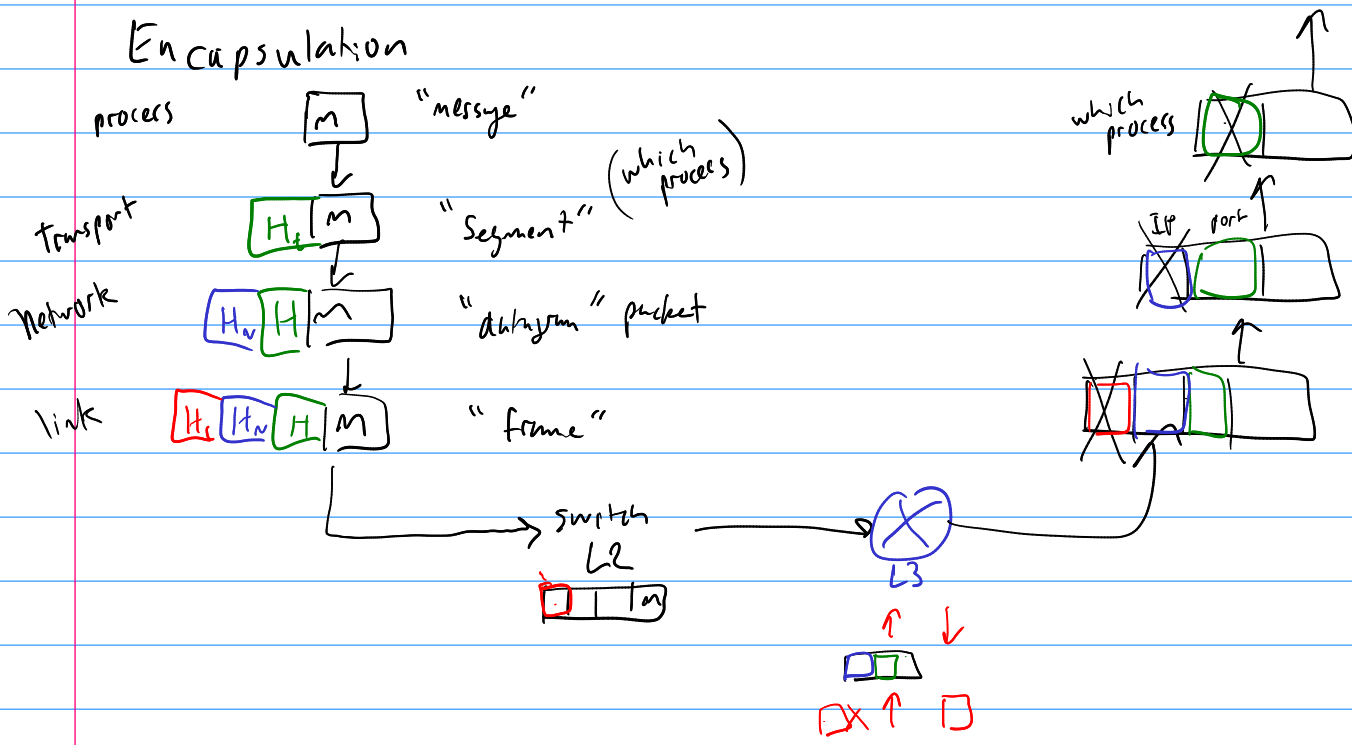
OSI/ISO 7-layer

(7) application — HTTP, IMAP, SMTP
6 presentation
5 session
(4) transport (TCP, UDP) process-to-process
(3) network IP (routing) host-to-host
(2) link Ethernet, wifi, ... host-to-switch
(1) phys bits on the wire

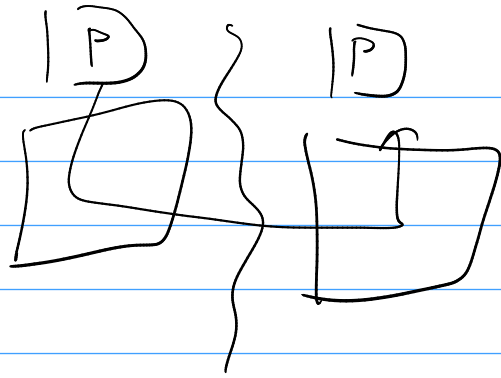




Encapsulation



transport layer



UDP: best effort

TCP: reliable data transmission

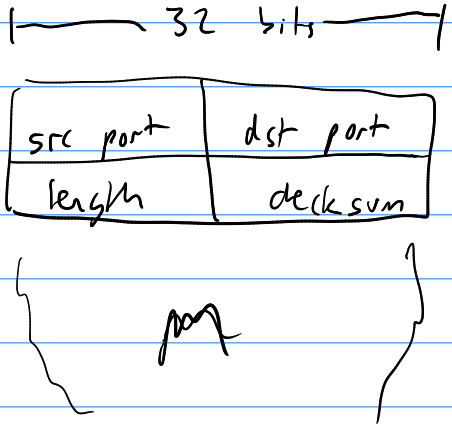
how to specify endpoint?

5-tuple

⟨ proto, src IP address, src port, dst IPaddress, dst port ⟩

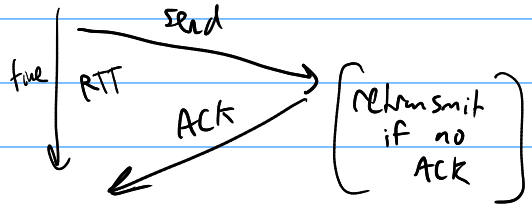
UDP - RFC 768 (1980)

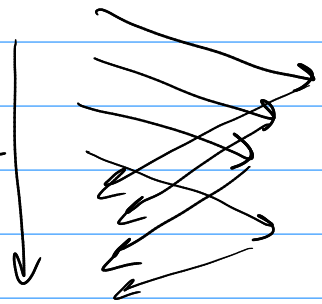
- "datagram" oriented (64K msg)
- connectionless
- unreliable



TCP - RFC 793 (1981), 1122 (1989, ... 7414

- point-to-point
- reliable, byte stream in-order
- connection-oriented
- pipelined

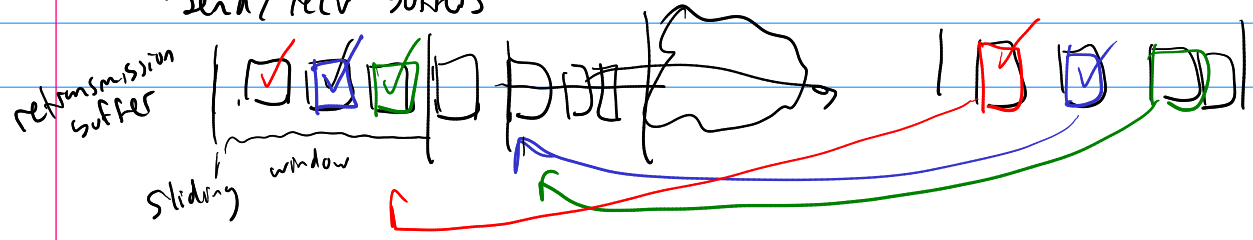




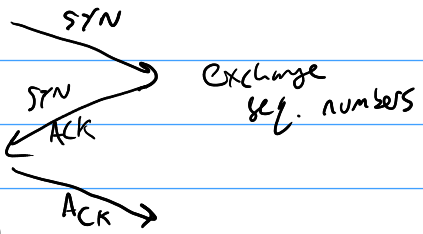
"window size"

sequence numbers

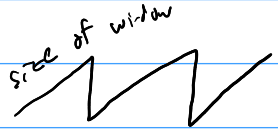
- send/recv buffers



- full duplex bi-directional
- connection-oriented → set up: TCP handshake



- flow controlled (sender will not overrun receiver)
- congestion controlled
  - protect network
  "saved the internet"



TCP sawtooth

header

src-port	dst-port
seq num	
ACK #	
flags (SYN/FIN/etc)	window size
checksum	urgent
options	

variants:
NAcks
⋮

Socket API
BSD UNIX 1981
unified: network: UDP/TCP (or)
local: UNIX sockets IPC ("inter process communication")

Sockets are fds
read/write/close

	server	client
UDP	fd = socket()	fd ← socket()
	bind()	sendto(fd,
	recvfrom()	recvfrom(fd
	sendto()	

80, 22, 443, ... 35 123

socket (int domain, int type, int proto);

- domain: PF_INET, PF_UNIX, PF_INET6
- type: SOCK_DGRAM, SOCK_STREAM
- proto: IPPROTO_UDP, IPPROTO_TCP

bind (int sockfd, struct sockaddr *, socklen_t);

sockaddr: IP, port

TCP

server	client
fd = socket()	socket
bind()	connect()
listen()	read()
	write()
OS queues connection	close()
accept() → new fd for that connection	