Transport Layer Protocols

- inside the host.
- comm. b/w processes.

(Network Layer: Comm b/w hosts)

TCP: reliable, in order, congestion control, flow control, connection setup.

multiplexing (sender)

demultiplexing (recv)

- Conn"- Less Demux (UDP)

IP datagram with same dstn port no → same socket at recv.

- Conn"- Oriented Demux (TCP)

4 tuple: {src IP, src port, dest IP, dest port}

each socket of server associated with a diff connecting client
(same dstn port no then too into diff socket, based on tuple)

→ UDP:

best effort service
no RTT delay of handshake
Small header size.

no state maintain at snd, recv. (fast)

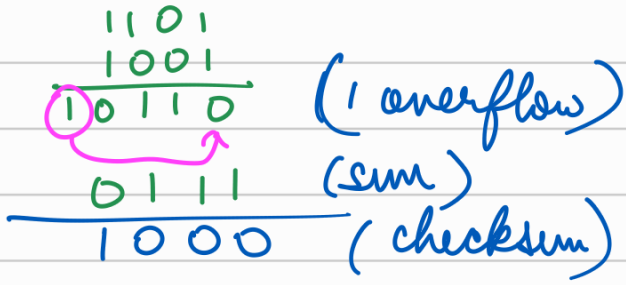
(may add reliability, congestion control at application layer — HTTP/3).

use: multimedia apps, DNS, SNMP, HTTP/3

Checksum:

treat contents of UDP segment (incl header fields & IP addr) as seq of 16 bit integers

add them, one's compl sum



→ Reliable data transfer

RDT 1.0 : perfect channel, no errors / loss

RDT 2.0 : (with bit errors)
— checksum
— ACK / NAK

RDT 2.1 : seq. # (0, 1) for corrupted ACK/NAK

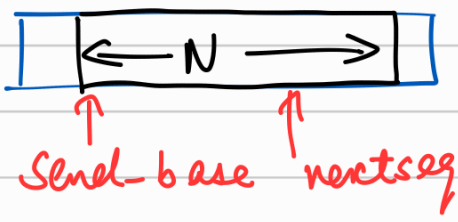
RDT 2.2 : NAK free
recv sends ACK of last pkt recvd OK.
(recv needs to incl seq no)

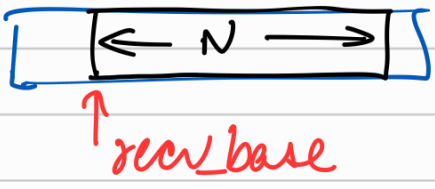
RDT 3.0 : (errors + loss) timeouts

$$U_{sender} = \frac{L/R}{L/R + RTT}$$ (do pipelining)

— Go Back N

- window size = N. = (transmitted unack. pkts)
- ACK(n): ack ∀ pkt ≤ n.
  ↳ send_base at n+1.
- timer for oldest in flight pkt.
- timeout (n): retransmit n ⟶ n+ nextseq.





recv: send ack for correctly recv highest seq no pkt.

— Selective Repeat

recv buffers out of order pkts
sndr maintains timer for each un ACK pkt
retransmits only the pkt whose timeout

problem: window size and seq # magnitude should not be close
(seq size >> window size)

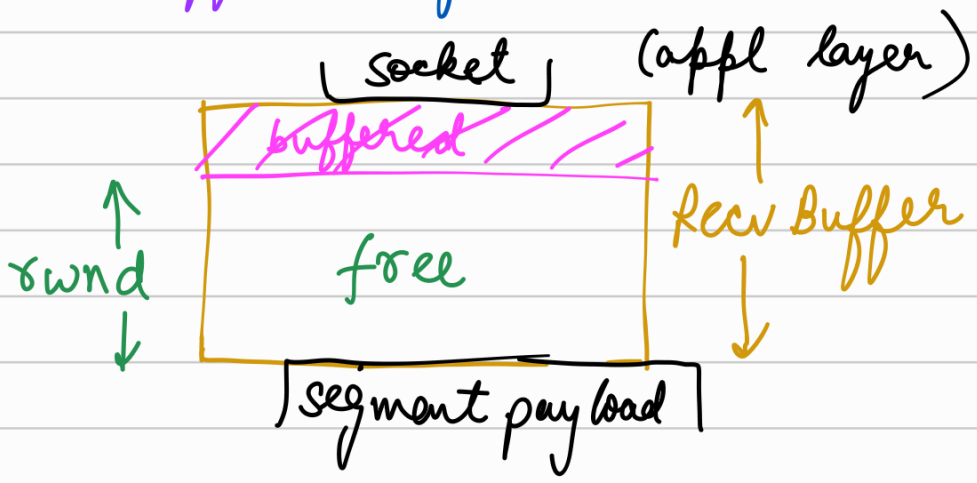
→ TCP overview

* point to point (one sender one recv)
* reliable in order byte stream.
* bidirec^n data flow in same dir

* MSS : max seg size
* cumul ack, window size fix.
* TCP flow control + congestion control
* handshaking initialises sndr, recv

→ **Flow Control**

Recv tells free buffer space in rwnd field in TCP header.
RcvBuffer : (def = 4096B) - set using socket options



sndr : amt of unACK in flight data ≤ rwnd

→ **2 way handshake**

retransmit then

req-conn or data

half open conn

dup data accepted

→ End-End Cong Control: (TCP does this)
no explicit feedback from network
cong. inferred from observed loss / delay

→ Network Assisted Cong. Control
router sends direct feedback to sending /

recv hosts. Tells cong' level/sets sending rate
TCP ECN, ATM, DEC

→ AIMD (probing for bandwidth)

inc send rate by 1 until pkt loss occurs (congestion), then dec send rate expo^n at each loss

TCP Reno – Cut in half on loss detected by triple duplicate ACK.

TCP Tahoe – Cut to 1 MSS when loss detected by timeout

- cwnd : Sndr window size.

Last Byte Send – Last Byte ACK ≤ cwnd.

TCP rate ≈ $\frac{\text{cwnd}}{\text{RTT}}$ B/sec

- TCP slow start "ssthresh"

initially cwnd = 1 MSS.
double cwnd every RTT, until first loss event

new ssthresh = $\frac{1}{2}$ (cwnd before loss)

- TCP Cubic

W_{max} – sending rate at which congestion was detected

after cutting rate in half on loss initially ramps to W_{max} faster, and approach W_{max} more slowly.

→ Delay based TCP Cong. Control

RTT_{min} = min obs RTT (uncongested)

best throughput = $\frac{cwnd}{RTT_{min}}$

measured " = $\frac{\#Bytes\ sent\ in\ last\ RTT}{RTT_{measured}}$

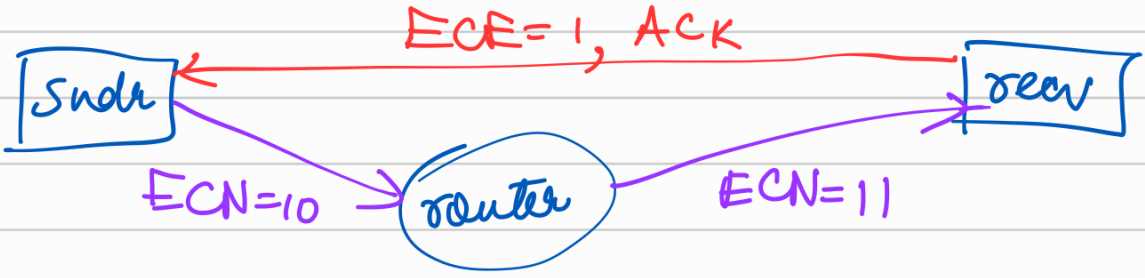
if meas. close to Best ⇒ inc cwnd linearly
if meas. far below Best ⇒ dec cwnd linearly

→ ECN (Explicit Congestion Notification)

(Network Assisted Cong. Control)

- 2 bits in IP datagram header, marked by router to indicate congestion.
- dest^n sets ECE bit on ACK segment to notify sndr of congestion.

both IP (ECN bit) & TCP (ECE bit)



→ TCP Fairness

K TCP connections share same bottleneck router of capacity R ⇒ each get avg of R

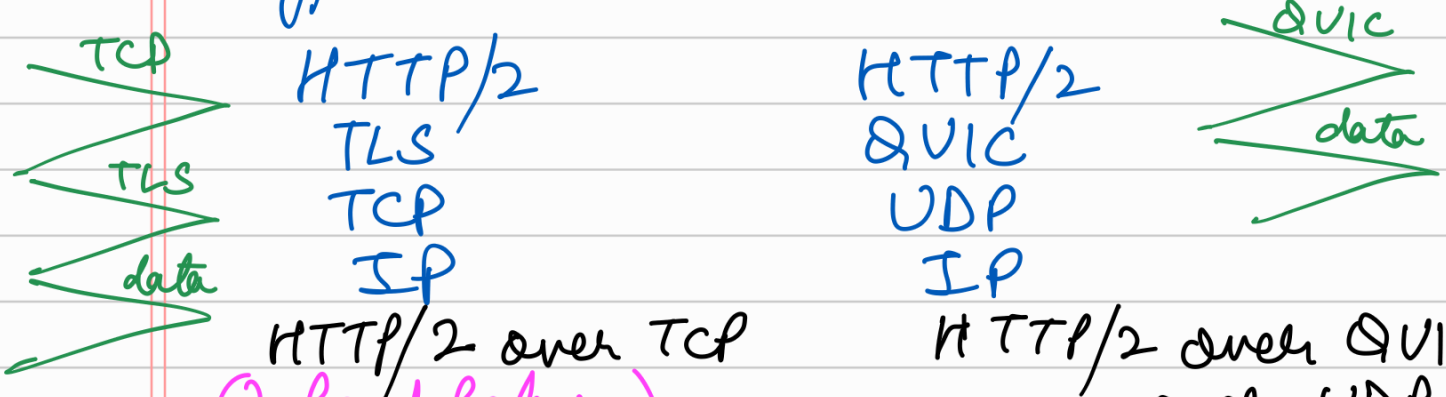
start at any points, hosts reach equal bw share in AIMD.

- Multimedia apps: use UDP, tolerate loss

Long fat pipes: large data transfer.
Wireless: loss due to noise, mobility
Long delay link: Exto. long RTT
Data centre network: Latency sensitive
Background traffic flows: Low priority (in backg)

HTTP/3: QUIC — moving transport layer func to appl layer on top of UDP
(Quick UDP Internet Connections)

QUIC gives:
reliability, cong. control, authentication, encryption, state estab, loss/error control.



QUIC — parallelism, no HOL blocking

→ TCP Throughput

W — window size (B) when loss occurs.
avg window size = $\frac{3}{4}$ W

avg throughput = $\frac{3}{4}\frac{W}{RTT}$ bytes/sec

TCP thruput = $\frac{1.22\ MSS}{RTT\sqrt{L}}$ (L = segment loss prob.)

Ch-4

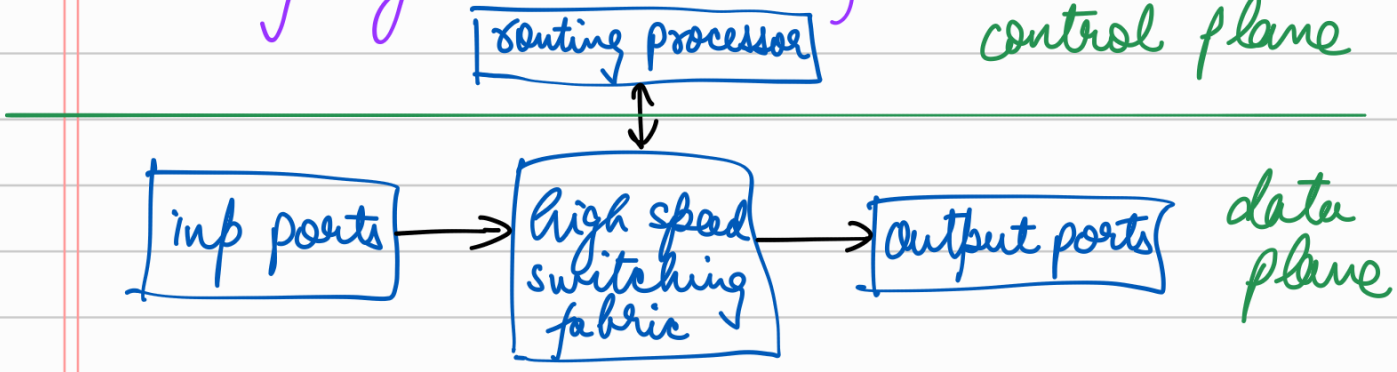
Data Plane:

local, per router function. inp port → out port

Control Plane:

end-end path routing
- Trad^n routing algos
- Software Def^n Network (SDN)

Routing algo → local fwd table.



Input Port func

lookup, forwarding, queuing (if datagrams arrive faster than fwd rate into switching fabric).

Match + Action:

Dstn based fwd — based only on dest IP addr.

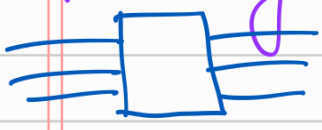
Generalised fwd — based on any set of header field values

→ Longest Prefix Match for dstn IP addr.

- **Switching Fabric**

Inp port to output port switching rate → NR ideally
(N ports, R pkts per port)

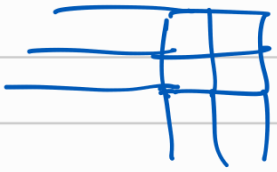
Memory



(2 bus crossings per datagram)

Bus

Interconn. Network



(multistage switch)

Inp port queuing — HOL blocking, if sw. fabric slower than NR.

Output port queuing —
Buffering : datagrams arrive from fabric faster than link transmission rate.
(Network neutrality : Drop policy, priority scheduling)

Buffer size:

① RTT × C (C = link capacity)
C = 10Gbps, RTT = 250 ms ⇒ 2.5 Gbit buffer

② N flows ⇒ $\frac{RTT \times C}{\sqrt{N}}$

Packet Drop: < tail drop
priority drop.

Marking (ECN bits, RED).

→ Pkt Scheduling

1. FCFS / FIFO.
2. Priority Sch: FCFS within priority class.
3. Round Robin (based on any header field)
4. Weighted Fair Queuing: $\frac{w_i}{\sum w_i}$ (min bw guarantee)

each class gets weighted amt of service in each cycle.

→ IP addressing

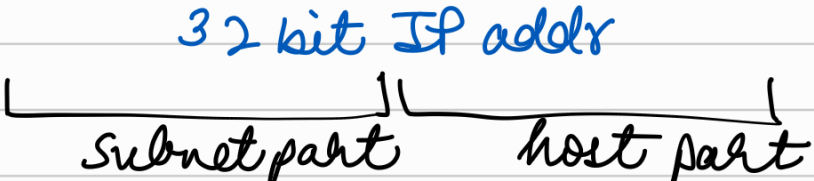
IP address: 32 bit identifier associated with each host or router interface

interface: conn. b/w host/router and physical link

routers: multiple interfaces
hosts: generally one / two (wireless + ethernet)

→ Subnet

device interfaces that can physically reach each other, without passing thru a router. Have common high order bits.



223.1.3.0/24 ⇒ 24 = subnet mask.
high order 24 bits – subnet part of IP addr, 8 bits diff for host.

CIDR (classless InterDomain Routing)

subnet portion of addr. of arbitrary len.

a.b.c.d/x ⇒ x bits in subnet

Host IP addr

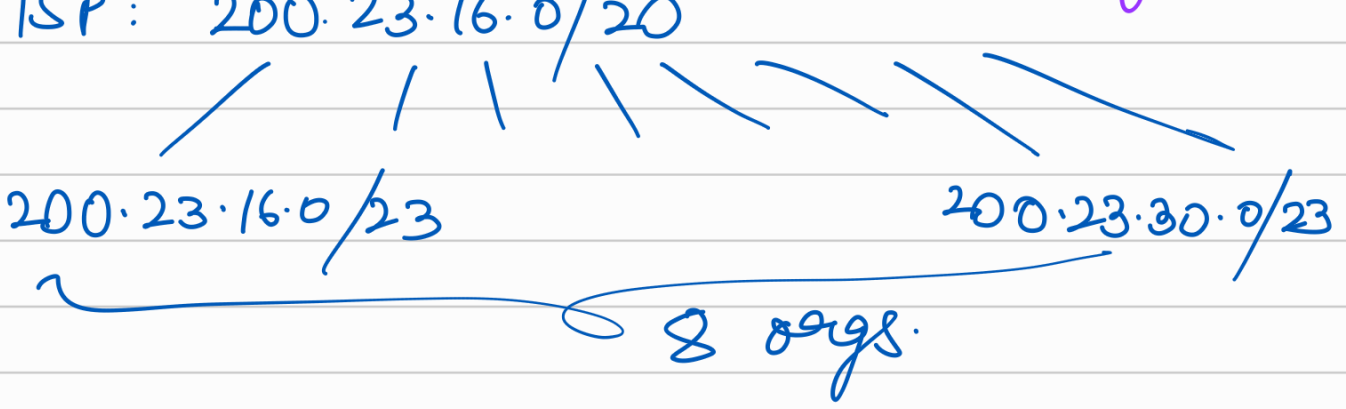
- hard-coded by sysadmin in config file
- DHCP: dynamic host config^n protocol (plug and play)

i) host broadcasts DHCP discover msg (optional)
ii) DHCP server gives DHCP offer msg (optional)
iii) host req. IP addr: DHCP req msg
iv) DHCP server sends addr: DHCP ack msg

DHCP colocated in router.
can give: addr. of first hop router
name + IP addr of DNS server
network mask (network v/s host portion of addr info)

Network IP addr

network gets allocated portion of its provider ISP's addr space. (ICANN gives)



→ NAT

all devices in local network share one IPv4 addr as far as outside world is concerned.

all devices have 32 bit private IP addr: (10/8, 172.16/12, 192.168/16) that can only be used in local network

router maintains mapping of local {IP x local port no} → NAT IP x Port no (diff)

srcIP, port # → NAT IP, port#
NAT IP, port # → src IP, port#

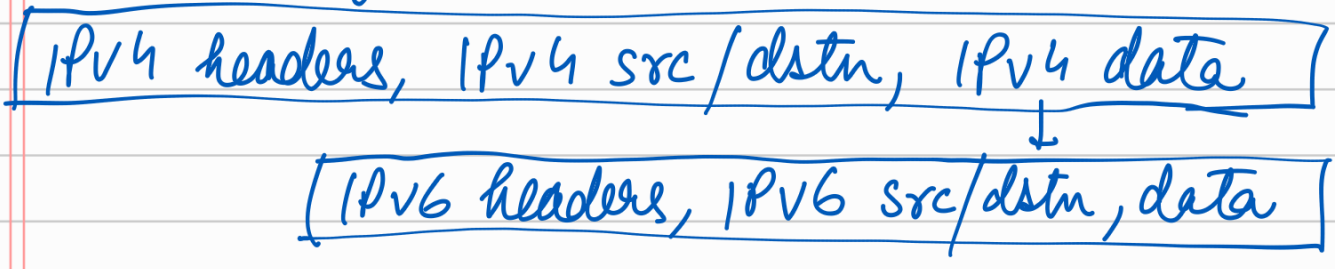
- only one IP addr needed from provider ISP for all devices
- can change addr of a host without telling outside world
- can change ISP without changing addr of devices in local network
- security

→ IPv6

"Classic" 40B fixed length header

flows — 40 bytes fixed length header.
↳ pkts of same flow: send directly, no need to intersect deeper.

Tunneling –
IPv6 datagram carried as payload in IPv4 datagram in IPv4 routers



→ Generalised Fwd / Match + Action

flow: defined by header field values (in link, network, transport layer)

① Match – pattern in table to header fields
② Action – drop / fwd / modify / send to ctol.
③ Priority – disambiguate overlapping patterns
④ Counters – # Bytes, # packets

— Middle Boxes

NAT, Firewalls / IDS, Appl Specific, Load Balancers, Caches

— Fragmentation

MTU: max transfer size (largest possible link level frame).

large datagram fragmented, reassembled only at dstn. IP header bits used to identify, order related fragments

offset in each datagram

TCP Reno

- Triple Dup
  - $ssthresh = \frac{cwnd}{2}$
  - $cwnd = \frac{cwnd}{2} + 3MSS$
- Timeout
  - $ssthresh = \frac{cwnd}{2}$
  - $cwnd = 1$

TCP Tahoe

- TD
- Time

$SSthresh = \frac{cwnd}{2}$

$cwnd = 1$