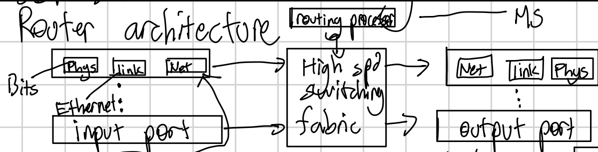
Application: Supporting Network apps (HTTPS, SMTP, DNS) - Message
Transport: Process-Process Data Transfer (TCP, UDP) - Segment
Network: Routing Datagrams from src to Dst (IP & Routing Protocols - Datagrams
Link: Data transfer between neigboring network elements (Ethernet, 802.11) - Frame
Physical: Bits on a wire

4.) Forwarding: move packets from router input to output links (intersection) - Data plane
Routing: Determine route taken by packets from src to dst (GPS) - Control plane
Per-router control plane: individual component in each router
SDN (software defined Networking: Remote server installs forwarding table

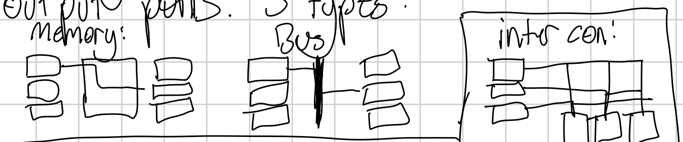
Router architecture



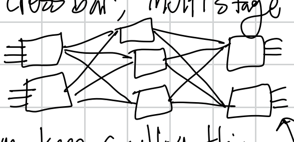
Decentralized switching: use forward table
Dst-based - using only IP address
Generalized - Based on header field values

Longest prefix: forwards to longest matching prefix

Switching fabrics: Transfer packets from input to output ports. 3 types:



↳ cross bar, multistage:



Parallelism: fragment datagram and switch cells thru
can keep scaling this

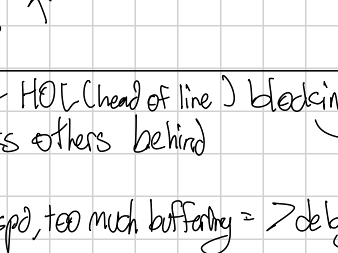
Input Port Queuing: HOL (head of line) blocking: Queued datagram blocks others behind
Output Port Queuing:
Buffering: arrival rate > output spd, too much buffering = > delays
drop: Tail drop or Priority, mark for congestion
Packet scheduling: FCFS, Priority, Round Robin, Weighted fair Queuing
lines, airport gold, categorize

32 bits	diff serve ccn		
IP ver	head len	Type of service	Total length
16-bit ID		flgs	fragment offset
Time to live	Upper layer	header checksum	
remaining hops	TCP or UDP	Source IP	
Dest IP			
Options			
Payload			

← fragment data
20 bytes

CIDR: Classless InterDomain Routing
ISP just like subnet,
Nat: Network address translation
Private IP's & only 1 pub. IP

DHCP:



ver	priority	flow label	
payload len		Next HDR	hop lim.
src addr			
dst addr			

Tunneling: IPv6 carried as payload in IPv4 datagram in "holes" of IPv6

Match + Action: match bits in packet, take action, can do firewall/NAT/forwarding
Middle boxes: between src & dst

"link state": Routers have complete knowledge
"distance Vector": Routers only know neighbors

Dijkstra's link state routing alg.
Init: $N' = \{u\}$
For all nodes v
if v adj to u, $D(v) = c_{u,v}$
else, $D(v) = \infty$
loop: find w not in N' such that $D(w)$ is a min
add w to N'
update $D(v)$ for all v adj to w and not in N':
$D(v) = \min(D(v), D(w) + c_{w,v})$

Algo complex: n nodes
$n(n+1)/2 = O(n^2)$
possible $O(n \log n)$ efficiency
Message complex:
Crossing $O(n)$: $O(n^2)$

Oscillations: Swapping CW & CCW

Distance Vector Algorithm (Bellman-Ford equation)
$D_x(y) = \min_v \{c_{x,v} + D_v(y)\}$
Iterative, asynchronous, distributed, self-stopping
bad news travels slow

LS	DV
n routers, $O(n^2)$ msgs	convergence times vary
$O(n^2)$ alg & msg, oscillations	varies, loops, count to ∞
can have incorrect links	black-holing
calculates own table	error propagate

To make routing scalable, split routers into AS or domain

intra-AS: within AS
RIP: classic DV
OSPF: Open Shortest path first: Link state
Enhanced Interior Gateway Route Prot. DV based, cisco
OSPF: Routers flood OSPF ads, has full topology & Dijkstra
Hierarchal: splits into more.

inter-AS: between AS's
BGP, Border Gateway Protocol
eBGP, out of AS, iBGP = within
OPEN: opens BGP connection
UPDATE: advertises new path
KEEPALIVE: keeps con open
NOTIFICATION: errs & close
hot potato routing: Not eff.

SDN (Software Defined Networking) allows more control on paths taken & make more reliable

SDN: controller
network graph, RESTful API, Intent
stats, flow table, Link-state info etc
Open flow, SNMP

Control-switch: feature, configure
Modify state
packet-out
Ex: Open Daylight
ONOS

Switch -> controller: packet-in, flow-removed, port status
ICMP: Internet Control Protocol. #1
· Error report, echo request/reply.

Network operator approaches to managing:
CLI, SNMP/MIB, NETCONF

SNMP: req/resp or trap

Get req	get data
Set req	set MIB
response	val, response
trap	inform

MIB = Management INFO Base

Netconf:

Get-Config	get config
get	get config & op. state
edit-config	
lock/unlock	
notif	

Link layer services:
· Framing
· Reliable delivery
· Checksum
· flow control
· error detection
· error correction
· half duplex/full
· half - both ends can transmit

Error detection:
- single bit parity: detect single bit errors
- 2D bit parity: detect & correct

Cyclic Redundancy Check:
$\langle D,R\rangle = D*2^r \text{ XOR } R$
if $\langle D,R\rangle \bmod G \neq 0$, error

Multiple Access Links:
Point-to-point: PPP, ethernet switch-host
Broadcast: Shared wire or medium

Channel partitioning:
- TDMA: Time slots
- FDMA: Split frequency bands

Random Access:
- Slotted Aloha: if collision retransmit with prob. p, max eff = 0.37 (pure aloha max eff = 0.18)
- CSMA: if idle, send, else, wait
  - CD: if collide, abort ← Eth 802.11
  - Ethernet: sends jam,
    - Binary backoff, after m^{th} coll. choose from $\{0,1,2,2^m-1\}$ bit times

Taking turns: Bluetooth
- Polling, controller polls & gets data
- Token, gets passed & token holder sends data

- 32-bit ip, 48-bit MAC
- ARP Query: Uses B's IP, dst all F's
- ARP response: gives MAC addr
- Router as middleman to another subnet

- Ethernet, bus thru mid 90's, switch now

preamble	Dst addr	src addr		data	crc
7x (10101010) + 10101011			Type (IP)		

- switch forwarding table, if unknown, flood all interfaces to find
- Can use virtual LANs by seperating ports or MAC
  - adds sections to frame before type

- MPLS (Multiprotocol Label Switching
  - adds a label after eth header
Label	exp	S	TTL
- IP routes with dst, MPLS uses dst & src

- Data center networks:
  - Border router: Out of datacenter
  - Tier 1 switch: connects to ~16 T2
  - Tier 2 switch: connects to ~16 TOR
  - Top of Rack (TOR): 1 per rack
  - Server Rack: 20-40 server blades (host)

SNR: Signal to Noise Ratio — Bit error rate when BER gets high, switch

Problems:
- Hidden Terminal (blocking signal A-C)
- Signal Attenuation (distance A-C)

Code Division Multiple Access (CDMA)
- Each user has a "Chipping Sequence" to encode data. (data * chip)

802.11: Passive/active scanning

passive:	active
① AP beacon frame	① probe req. from H1
② H1 sends association req. frame	② probe resp. from AP
	③
③ AP association response	④

- 802.11 uses collision Avoidance due to not accurately detecting (hidden terminal, fading)
- Idea:
  - sender sends request to send (RTS) use CSMA
  - reciever broadcasts Clear to send (CTS)
- 802.11 frame header:

2	2	6	6	6	2	6
frame Ctrl	duration	addr 1	addr 2	addr 3	seq.	addr 4
		rcv MAC	transmit MAC	Router MAC		ad hoc

4G Elements
- Device w/ 64 bit International Mobile Sub. ID
- Base station like Wifi "AP" (Access Point)
- Home Subscriber Service: Stores info ab. devices w/ home
- Serving GateWay (SGW), PDN GateWay (PGW) ← outside
  - between mobile & internet
  - does NAT stuff (Network address trans.)
    - IP into other IP
- Mobile Management Entity
  - Auth, device handover, tracking, SDN
- LTE Link layer:
  - Packet data Convergence: header compress, encrypt
  - Radio Link control: frag/reassemble, reliable data trans.
  - Medium Access, req., use of radio trans slots
- LTE radio access Network: FDM, TDM
- Associating with a Base Station:
  - BS broadcast primary synch every 5 ms
  - Mobile finds primary, locates 2nd & get info
  - Selects BS, auth, establish
- LTE Sleep modes
  - light after 100's ms, wake up for down stream
  - Deep after 5-10 s,
- 5G: 10x peak bitrate, 10x less latency, 100x traffic capacity
- On moving, mobile associates w/ MME that tells HSS

- TCP
  - FTP, SMTP, HTTP, SIP, RIP, DASH
  - 3 way handshake
  - Segment (32 bits)

src port			dst port
seq # (byte #)			
ack # (nextSeq)			
head len			rcv window
checksum			
options			

  - TCP fast retransmit: if 3 same ACK, resend smallest
- TCP AIMD: increase by 1 MSS max seg size every RTT until
  - TCP Reno (triple ACK) (halve)
  - TCP Tahoe (Timeout) (cut to 1)
- UDP
  - SIP, RTP, games, HTTP3
  - Unreliable Segment:

src port	dst port
length + header	Checksum