CH #5

Intro:

Node, link, ~~link~~ network datagram → link layer frame.

Link layer Services:

Framing:

frame contains datagram & headers

link Access:

- MAC → tells rules.
- Multiple access problem.

reliable Delivery:

- without error.
- mostly used in wireless link.

Error Correction:

~~- error~~

CRC:

- error detection $R = \text{remainder } \frac{D \cdot 2^r}{G}$
- Can detect:
  - odd number of errors,
  - single bit
  - burst equal to length of polynomial degree.

Example:

Notes:

- Divisor polynomial might be given instead of binary string.

  ↓

  Degree = r

link layer Protocols:

Multiple access Protocols:-

1- Channel Partioning
2- Rand. access
3- taking turns.

Channel Partitioning:

- FDMA
- TDM
- CDMA

Random Access:

- Slotted ALOHA:
  - Frames have L-bits
  - Size of slot = $\frac{L}{R}$ → time to transmite one frame
  - Efficiency = $Np(1-p)^{N-1}$ ∴ N = no. of nodes.

    ↓

    finally $1/e = 0.37$

CSMA & CSMA/CD

$$\text{Efficiency} = \frac{1}{1 + 5\, t_{prop}/t_{trans}}$$

Taking Turns:

- Pollings
- Token Passing

link layer Adressing:

- ARP : Address resolution:-

Ethernet:

- Popular link layer Protocol.
- Simple, fast, cheap, first

- Preamble:
  - 6x10101010
  - 10101011

- Type:
  - Bonds link layer and Network layer.

CH #4:

Routing Algorithms:

Link State
- Complete Topology is known.
- Tell the world about your neighbours
- Global

Dist. Vector
- know your neighbours
- Tell ~~to~~ the neighbours about the world.
- Decentralized

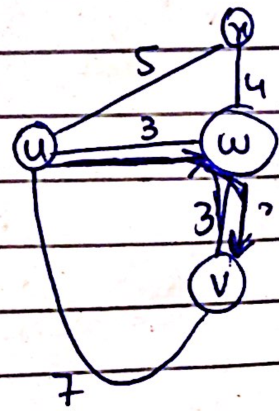
Static
- Slow Changes

Dynamic
- Periodic Update

Link State:

Dijkstra's Algorithm:

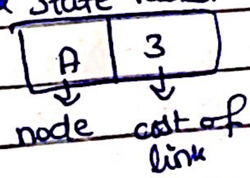
Step	N'	D(v) P(v)	D(w) P(w)	D(x) P(x)
0	u	7,u	3,u	5,u
1	uw	6,w		5,u
2	uwx	6,w		
3	uwxv			



Link State DB:

	u	v	w	x
u	0	7	3	5
v	7	0	3	∞
w	3	3	0	4
x	5	∞	4	0

Link State Packet (LSP):



Distance-Vector Algo:

Heirarchical routing:

- **Hot Potato Routing**: Choose between the two closest

Routing Protocols:

- Intra-AS:
  - ~~RIP~~
  - RIP & OSPF or (EIGRP) → CISCO ka hai

- RIP:
  - Distance vector Algo
  - Distance Metric
    - # of hops
    - max = 15 hops
  - Share distance vectors every 30 seconds
    - upto 25-subnets
  - If no advertisement for 180 seconds — link down/dead
  - Uses UDP

- OSPF:
  - Direct IP datagrams
  - Flood advertisements, carried ~~one~~ atmost one per neighbour
    - ↓
    - entire AS

BGP:
- Peers
- ~ eBGP & iBGP
- route = Prefix + Attributes
  - ↓
  - AS-PATH
  - NEXT PATH

BGP messages:
uses TCP.
OPEN, UPDATE, KEEPALIVE, NOTIFICATION

IP datagrams

← 32bits →

ver	h.l	tos	length
16-bit identifier		flgs	frag-off
TTL	Upper lay		Checksum
32bit S-Add			
〃 D-add			
options			
data			

Subneting:

- Efficiency of broadcasting
- Security
- Reduce size of broadcast Domain

CIDR :-

- Arbitrary length
- fixed-bits are host bits

DHCP:

- Plug & Play
- DORA

Discover Offer Request ACK

- Also gives:
  - Default Gateway
  - Subnet Mask
  - DNS Server.

~~Po~~ Message:

Src:
Dest:
yiaddr:
transaction:
lifetime:

in the payload → yiaddr

in ~~D~~ORA ← lifetime

NAT:

Uses Single IP address

ICMP:

- error reporting
- Slightly above IP, but not transport layer.

ICMP msg has:

- Type,
- Code, of
- first 8-byte Datagram.

(Type, code)

TTL: (11, 0)

Host unreachable: (3, 3)

, echo: (8, 0)

, timestamp: (13, 14)

, Congestion Cntrl: (4, 0)

IPV6:

- 128-bits
- Header: 40 bytes
- No fragmentation
- No Checksum
- Options moved to payload
-

- ICMPV6 for IPV6

ver	pri	flow label
payload len	next hdr	hop limit
src addr (128-bit)		
Dest addr (" ")		
Payload		

my name is adeel, What is your name?

CH# 01:

abcdefghijklmno
pqrstuvwxyz

RFC: request for comments
IETF: Internet Engineering Task force

Internet provides : Web, VOIP, email, games, social
Protocols govern all internet communication

Things in a network:
- edge: hosts, Servers
- Core : network of networks, routers
- Access Medium : Wired, wireless communication links.

Access Networks:-

1- DSL:
- external telephone line ⟷ DSLAM
  - Data goes → Internet
  - Voice goes → Telephone
- 2.5 upload , 24 Mbps downstream

2- Cable network:
- frequency Division
- Hybrid fiber COAX
  - 2 Mbps Up, 30 Mbps down

network of cables upto ISP, DSL has direct access

3- Home Network:
ethernet, wireless access point

4- Enterprise Access Network:
ethernet Switches

5- Wireless Access Networks:
wirless lans: (100 ft), 802.11 b/g
wirless wan: Telecom, 10 km, 3G, 4G: LTE

Physical Mediums,
Guided Media: Solid Media
Unguided media: signals propagate freely : radio

1- Guided Media:

- Twisted Pair:
  - Two copper wires
  - Category 5 : 100Mbps, 1Gbps
  - // 6 : 10 Gbps
- Coaxial cable:
  - two copper conductors
  - Bidirectional : HFC
- Fiber Optic:
  - light pulses → 1 pulse = 1 bit
  - High Speed
  - Low Error
  - Birdirectional
  - effected by:

    reflection, interference, obstructs by obj's.

2- Radio link Types:

- Terrestial microwave
- Lan
- Wan
- Satellite

Network Core:

Packet Switching:

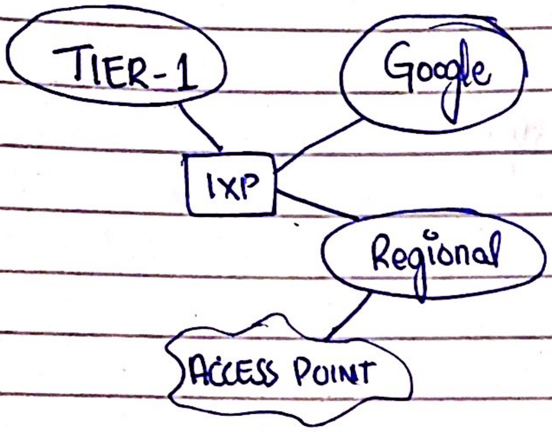
- Full link Capacity.

· Transmission Delay = $\frac{L}{R}$ L → length of packet (bits), R → transmission Rate

- for store and forward,

  $\frac{2L}{R}$ (assume no propagation delay).

How ISP's are connected? :-



Packet Delay:

$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$

$d_{trans} = \frac{L}{R}$, $d_{prop} = d/s$ $d \to$ distance, $s \to$ speed

Traffic Intensity:

$\frac{La}{R}$ $a \to$ avg packet arrival rate.

< 0 small queue delay

$<= 1$ some queue // added

> 1 mork wore coming.

CH#2:-

Network Apps:

- Communicate Over network;

App Architecture:

- Client Server:
  FTP, Web, TelNeT, Email
- P2P:
  - Dynamic IP
  - No server
  - Complex

Sockets: (Door of ~~house~~ Process)

- Share data — through Socket

Port Numbers:

HTTP: 80
mail: 25

Apps require:	TCP
reliable Data Transfer	reliable
Throughput	Flow Control
Security	Congestion Ctrl
Timing	- To encrypt TCP, use SSL

HTTP:

- Client/server Model
- Stateless
  - State saving protocols are complex

Non-Persistent

- multiple obj's mean
  " connections
- 2RTT + file trans time
- with parrallel connections.

Persistent

- multiple obj's sent over single TCP connection.

RTT:

- Time for client to server and back

RTT

HTTP Message:

method URL ver

header-feild name value

Body

Input Methods: POST, URL

Methods:

1.0/: GET, POST, HEAD

/1.1: ", ", ", PUT, DELETE

Responce Codes: 200, 301, 400, 404, 505

Cookies.

Server sets cookie,
Give cookie header line in request after.

Web Caching:- Example.

Bandwidth = 15 Mbps
avg obj size = 1Mbits
" req rate = 15/sec
Size of HTTP request is small
& Internet Delay = 2 sec

Total response time = ~~Sum of~~ LAN delay + access Delay + Internet Delay.
avg delay

access Delay ↓ delay b/w two routers

Traffic Intensity: 0.15 (LAN), 1 (access link)
↓
This is very high, avg Response time can go to mins.

Now, install a web cache:
Hit rates = requests satisfied by web cache
typically 0.2 to 0.7

assume, hit rate = 0.4
response time of cache = 10 ms

So, avg delay = delay from respone of cache + of internet
0.4(10ms) + 0.6(2s) ≈> 1.2s

HTTP 1.1	HTTP 2.0
- Pipelining, multiple objects over a single TCP.	- Same header, etc feilds
- FCFS	- not FCFS, priority set by server
- Small obj's stall behind large "	- push unrequested obj's to client not possible in 1.0 → We could only send the requested one's.
- Retransmition of lost obj's stalls	- Framing of obj's. we send frames instead of whole obj's. Then frames are combined together at the client side.

FTP:

One control Connection, One TCP connection for every file

methods:

USER username
PASS password
LIST
RETR filename
STOR "

DNS: Local name server, Root name Server, Authoritative name servers.

UDP, Port 53, Complexity at networks edge

Services:

- Host name to IP address translation.:

Browser extracts Hostname → DNS client → DNS Server $\xrightarrow{IP}$ DNS client → Browser.

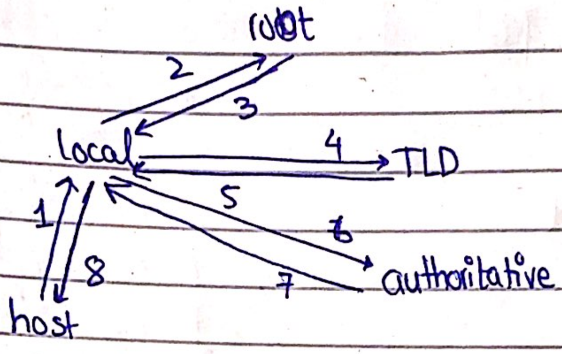
- ~~Host Aliasing~~
- ~~Canonical~~

- Types of servers:

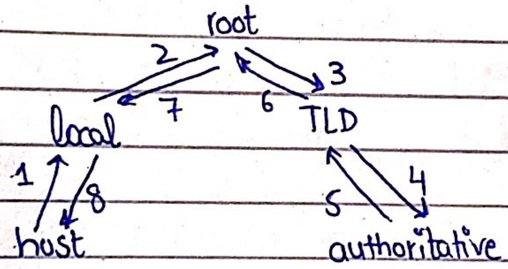
~~TLD~~: - Top level Domain (TLD) : com, org, net, edu,

- Authoritative:
  - Organizations DNS
    ↓
    maintained → org or service provider:

Iterative:



Recursive:



Resource Records: (RR)
- (name, value, Type, TTL)

Types:
- A, NS, CNAME, MX
- A: name = host
  value = IP
- NS (name server):
  - name = Domain e.g. FSU.edu
  - value = host name of authorative server for that Domain
- CNAME:
  name = Alias
  value = real name (canonical Name)
- MX: same as CNAME, but for mail server.
  - Used to direct email a server for domain

SMTP:

TCP, 25 port,

3 phases:

Handshaking, transfer of message, closure

- Uses persistent HTTP
- Multipart messages
- CRLF: carriage return line feed.

~~POP:~~ MAP:

- mail server to user agent.
- 3:

POP, IMAP, HTTP

POP3:

username & password

then:

list

retr

dele

quit.

- It is stateless.

IMAP:

- Allows: Servers to have folders.
- It is not stateless.

CH#03:

3.1: Transport layer:-

- logical communication — processes — Diff hosts.
- End systems
- App msg

↓

Break in segments

↓ Pass on

Network layer

- TCP & UDP is in Transport layer.

3.2 MUX & DEMUX:

- IP address & port number:
- ~~Data~~ Segment format:

Source port #	Dest port #
other header feilds	
Body	

TCP/UDP segment

- Connectionless DeMux:

IP & port # of dest

- Connection Oriented DeMux:

IP & port # of both src & dest

3.3: UDP:

- Segments may be lost, out of order
- No handshaking
- Reliability added app layer

src port #	dest port #
length	checksum
Payload	

- Checksum: error detection.

Sender:
- One's complement sum of segment contents - deal as 16-bit int's.

Reciever:
- Compute checksum of received segment

3.4: Principles of reliable Data Transfer:

- reliable Data Transfer Protocol (rdt):

method: (in order)

called by App ← rdt_send() } sending side
unreliable ← udt_send() }
channel transfer
packet arrives ← rdt_recv() } receiving
deliver_data() }
called by rdt ←

FSM: $\frac{\text{event}}{\text{action}}$

rdt 1.0:
- no error
- no loss

simple send and rcv of data

rdt 2.0:
- error checking
- recovery using ACK's & NAK's

- wait for ACK or NAK on sender side.

problem: what if ACK or NAK corrupt?

rdt 2.1:

Introduce Seq numbers.

rdt 2.2:

No NAK's.

rdt 3.0:

Timer Introduced.

- Performance of rdt 3.0:
  - performance stinks

Example:

link = 1Gbps , 15 ms dprop , RTT = 30 msec

$$\text{Sender Utilization} = \frac{\overbrace{d_{trans}}^{\text{transmission time}}}{\underbrace{RTT + d_{trans}}_{\text{total time}}}$$

Pipelining:

- Increase range of sequence numbers.
- Buffering at sender or receiver.

Two generic protocols:

1- Go-back-N
2- Selective Repeat

$$\text{Sender Utilization} = \frac{n(L/R)}{RTT + L/R}$$

- Go-Back-N:
  - upto N-unacked Packets
  - Send Cumulative Ack for all these packets.
  - Timer for the oldest pakt.

    If (expired) {
        send all pakt's again.
    }

  - K-bit Seq. number in pakt. header

~~Selective Repeat:~~

- Selective Repeat:
  - Only thing different from Go-Back-N is that we send ACK's for individual Pakt's.

Note: Window size must be less than or equal to half the size of sq number space. $WS <= \frac{1}{2}$ (sq.no. space)

3.5: TCP: - In TCP, sequence number is bytes of data, not segments.

- reliable, in-order
- Segment Structure:

Src port #									dest port #
seq no									
Ack no									
Head len	not used	U	A	P	R	S	F		receive window
Checksum									Urg data pointer
options									
Payload									

sequence number: first byte in segment Data

- RTT: TimeOut = EstimatedRTT + $\underbrace{\text{"Safety Margin"}}^{4(\text{DevRTT})}$

  estimated RTT = $(1-\alpha)^*$ EstimatedRTT + α^* SampleRTT

  DevRTT = $(1-\beta)^*$ DevRTT + β^*|SampleRTT - EstimatedRTT|

  Typically, $\alpha = 0.125$, $\beta = 0.25$

- Reliable Data Transfer:
  - TCP is over IP → unreliable.

  Simple TCP:
  - Take Msg → Assign Seq# → start timer $\xrightarrow{\text{expire}}$ TimeoutInterval

    Timeout:

    retransmit & s

  - Cummulative ACK's
  - ACK Generation.

- Fast Retransmit Algorithm:
  - If three duplicate ACK

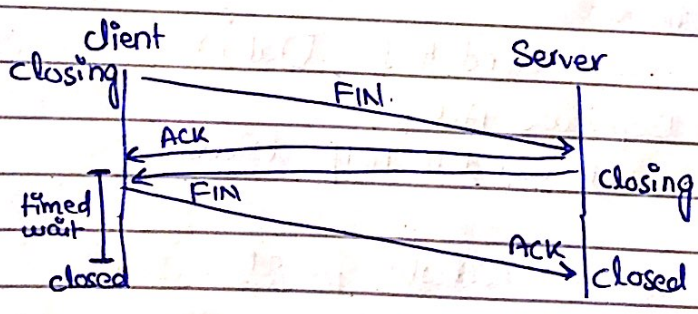
- Flow Control: Prevent Overwhelming the receiver.

rwnd = RcvBuffer - [lastByteRcvd - lastByteRead]

- every ACK contain rwnd.

Congestion Management:

- Decide these variables:
  - Seq #'s
  - Buffers
  - rwnd
- 3-Way HandShake: (Creating a Connection)
  - SYN: Initial Seq # (seq=x, Synbit=1)
  - SYN-ACK: No Data, Buffer Allocation (Seq=y, Acknum= x+1)
  - ACK: May Contain Data
- Closing a Connection:
  - FIN: Asks to close connection (client/server)
  - ACK: Response of ACK



3.6:

- Principle of Congestion Control:
  - Congestion Control is → saving the network from overflowing
  - Cost of Congestion: Large queuing Delay, Retransmission of Data, wastage of bandwidth
  - Goodput = λ_{out}
  - λ_{in} is the input
  - λ'_{in} is " " + retransmitted Input
  - Offered Load
  - End-to-End Control: Observed using loss & Delay at end-systems → Used in TCP/IP
  - Network-Assisted: Routers tell about congestion.

3.7: Congestion Control:

- End systems listen for delay & loss to perceive congestions.

- How to limit send rate?

cwnd ≥ LastByteSent - LastByteAcked

Sender limited by min (cwnd, rwnd)

$$\text{rate of the link} = \frac{cwnd}{RTT}$$

- Things to do after receiving ACK's:
  1- Slow Start:- cwnd
     - Increase ↑ Exponentially until threshold reached
       Default value
  2- Congestion Avoidance:
     - Increase cwnd linearly

- Things to do after loss:
  1- Timeout:
     cut cwnd to 1 (Tahoe)
  2- 3 Duplicate ACK's:
     cut cwnd in half (Reno)

- Threshold Reached: (ssthresh) for Slow Start
  - Switch from Slow start to congestion Avoidance.
  - Timeout:
    ssthresh = cwnd/2

- Congestion ~~Control~~ Avoidance:
  cwnd += MSS/cwnd

for packet loss
TCP Tahoe: Cut Cwnd ~~in half~~ to 1
// Reno: // // in half, use AIMD
↓
3-duplicate halfs