HAVELLS

classmate
Date
Page

rs:

→ dividing the data to be transferred in parts.
(eg: sending email, voice/video chat, requesting a file)

sequencing of packets
and reassembling the packets at destination

routing the packets

source & destination IP address

rce and destination MAC address

CN

Computer Networks

to gaia.cs.umass.edu
within a TCP
selected in) an
in an Ethernet / Wifi
network → link)
(frame)
(datagram)

that uses a packet capture
er.
layer frames.
n packet to refer to link-layer frames,
datagrams, transport layer segments,
ayer messages.
using link layer technologies like Ethernet,
Serial (PPP), etc

t sniffer = Packet analyzer + Packet capture (pcap)
Packet capture library receives a copy of every link layer
frame that is sent from / received by the computer over a
given interface (link layer such as Ethernet/Wifi)

Layers:

i) Application → dividing the data to be transferred in parts. (eg: sending email, voice/video chat, requesting a file)

ii) transport (TCP) sequencing of packets and reassembling the packets at destination

iii) network (IP) routing the packets eg: identifying source & destination IP address

iv) link → identifying source and destination MAC address

v) physical

* HTTP GET message that is sent to gaia.cs.umass.edu (or any URL) web server is contained within a TCP segment, which is contained in (encapsulated in) an IP datagram, which is encapsulated in an Ethernet / Wifi frame.

(Application → transport → network → link)
(messages) (segments) (datagrams) (frame)

Wireshark:

→ is a packet analyzer that uses a packet capture library in your computer.

→ Its captures link layer frames.

→ uses generic term packet to refer to link-layer frames, network layer datagrams, transport layer segments, application layer messages.

→ operates using link layer technologies like Ethernet, Wifi, Serial (PPP), etc.

→ Packet sniffer = Packet analyzer + Packet capture (pcap)
Packet capture library receives a copy of every link layer frame that is sent from / received by the computer over a given interface (link layer such as Ethernet / Wifi)

INTRODUCTION

* **Circuit Switching:**

→ dedicated communicat^n path b/w two stations

→ path is connected sequence of links (physical) b/w network nodes

→ first circuit established (link by link), routing & resource alloc^n (FDM or TDM) then data transfer, then circuit is cancel

→ (Rate determined through ~~STD code~~ subscriber code, STD code ISD code)

→ network resources (eg bandwidth, switch capacity) divided into pieces

→ dedicated channel (reserved) → no data ⇒ capacity wasted.

→ delay in establishment, after establishment fixed bit rate.

→ FDM (frequency division multiplexing) and TDM used

* **Packet Switching:**

→ data sent in form of discrete packets

→ share network resources & no dedicated allocation.

→ each packet use full bandwidth

→ store and forward (packets move one hop at a time)

Node receives complete packet before transfer.

→ congestion: packets queue, wait for link use.

→ allows more user to use the network (statistical multiplexing gain)

eg: N users over 1 Mb/s link, each user active 10% time

for circuit switching, at most 10 users with 100 Kb/s.

For packet switching:

With 35 users, probability >10 active is less than 0.0004 ($N=35$, $p=0.1$)

since more user $\left(P(N>10) = 1 - P(N \le 10) = 1 - \sum_{i=0}^{10} \binom{N}{i} p^i (1-p)^{N-i}\right)$

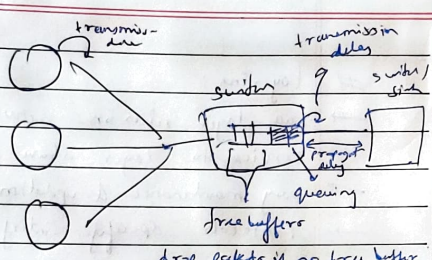
→ great for bursty data

→ bad for app. with hard resource requirements (bandwidth guarantee needed) (video streaming)

→ excessive congestion can lead to packet delay & loss.

(applications must handle eg YouTube)

∵ we don't have infinite bandwidth at switch's output, due to transmission delay there would be queuing delay



Delays:

i) Nodal processing delay:
→ check bit errors, determine output link

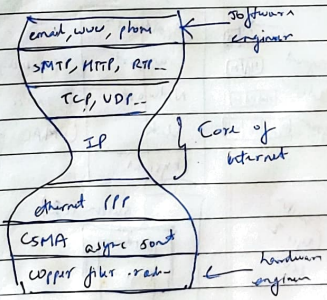
ii) Queuing delay:
waiting time at queue (output link) for transmission (depends on congestion of switch/router)

iii) Transmission delay: $= L/R$
where R = link bandwidth (bits/s), L = packet length (bits)

iv) Propagation delay: d/s
d = length of physical link, s = propagation speed in medium ($\sim 2 \times 10^8$ m/s)

→ Generally Queuing delay > transmission delay > Processing & propagation (ps to msec)

Hourglass Design:



→ Central part of hourglass is almost fixed (not much changes in function also)

→ Edges responsible for innovation (functionality) (top & bottom layer)
(applications and wires, etc)

→ Phone network : dumb edge devices, intelligent network
Internet : dumb network, intelligent edge devices

Networking Examples:
email, voice/video chat, fetch file from server, whatsapp message
P2P file sharing, multiuser game, streaming stored video

Layering:

→ Each layer relies on services from layer below & export services to layer above (hides implementation, modularity)

→ easy maintenance & updation of system

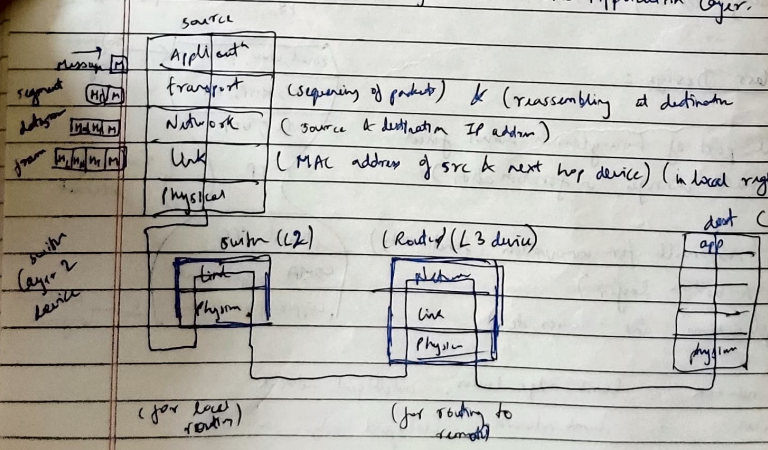
→ Protocols specify interface to higher levels (API), Application
interface to peer, interface defines interaction. ↓ Host to host connectivity
↓ Link hardware

OSI (Open system Interface) model : 7 layers

FTP, SMTP, HTTP	everything else (web browser, whatsapp, etc)	← Application
	byte ordering, formatting, security	← Presentation
	how to tie flows together (no. of channels opening & closing)	← Session
TCP, UDP	sending packets in order end2end	← Transport
IP	route packets towards destination	← Network
PPP, Ethernet	how end users connect to nearby switches (local communication)	← Data Link
	(wires & how bits will transmit through it)	← Physical

Host

→ First 3 layers can be merged into one Application Layer.



Ch 2 : Application Layer :

* Application Architectures :
i) Client server
ii) Peer to Peer (P2P)
iii) Hybrid of client server and P2P — eg: Skype → VoIP P2P app; centralized server (find address of remote system); chatting is P2P

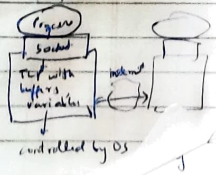
Client server Architecture :
* Server : always on host (end system)
→ permanent IP, server farms for scaling

* Clients : communicate with server (sends requests)
→ may have dynamic IP address, may be intermittently connected
→ do not directly communicate with each other.

Peer to Peer Architecture : eg: Torrent
→ no always on server
→ end systems (arbitrary) directly communicate (highly scalable)
→ peers intermittently connected & change IP address

Processes Communication :
→ Inter process communication within same host → by OS (eg semaphores)
→ Processes in diff host communicate by exchanging messages.
→ Client Process - initiates communication
→ Server Process - waits to be contacted
} P2P architectures have both

Sockets :
→ Process send / receive message to / from its socket
→ interface btw process in application layer and the lower level layers (TCP and others) (like a door)
→ API (Application Programming Interface) :
i) choice of protocol (eg TCP or UDP (unreliable))
ii) fix few parameters



Addressing Processes:

→ host device has unique 32-bit IP address (IPv4) (IPv6 → 128 bit) ↗ 8×4

→ many process can be running on same host, hence along with IP, port no. associated with the process is also required.

eg HTTP server : 80 (port no.)
Mail server : 25

eg IP: 128.119.245.12

App layer Protocols define:

→ Type of message exchanged (eg request, response)
→ Message syntax & semantic (fields in message, how delineated)
→ rules for when & how processes send & respond to message (eg close connection after each request)

→ Public domain protocols: defined in RFCs (Proprietd. protocols Skype)
→ interoperability (Windows, Linux)
→ eg HTTP, SMTP

Transport Service an App Needs:

i) Data loss : → audio / video can tolerate some loss (minor glitches)
→ file transfer should be 100% reliable data transfer (else error)

ii) Timing : → interactive games, require low delay
→ file transfer, email : delay is OK

iii) Throughput : → multimedia apps require minimum throughput to be effective
→ email can use whatever is got (elastic)

iv) Security : → Encryption, data integrity eg: credit card transactions

Internet transport protocols services :

* TCP service :	* UDP service
→ connection oriented (setup required b/w client & server)	→ no connection
→ reliable transport	→ unreliable (data loss)
→ flow control : sender won't overwhelm receiver (if sending rate is very high, receiver may need to buffer so it controls the sending rate)	→ no flow control, no congestion control, timing, ... etc.
→ congestion control : at the queue of switch/router to handle data loss.	→ Why bother? lightweight & faster
→ does not provide : timing, min. throughput, security	internet eg streaming multimedia & telephony
eg email, web, file transfer (HTTP uses TCP)	combination of TCP and UDP (security) (speed)

Web and HTTP :

→ Webpage = basic HTML file + referenced objects (images, files, audio)

→ Each object has a URL : www.xyz.com / dept/ pic.gif
hostname path name

→ HTTP is web's application layer protocol

→ client-server model (browser) (web server) (sent request, receive response as web objects)

→ Uses TCP ~~to exchange~~ (creates socket)

→ HTTP is **stateless** (no info about past client requests with the server)

* Non-persistent HTTP : At most one object sent over a TCP connection

→ Response time = (2 RTT (round trip time) + transmit time (of the file)) / object.

Persistent HTTP : multiple objects over single TCP connection b/w client-server

Response time = 1 RTT + (1 RTT + transmission time) / objects
(handshake sort of) → for initiating TCP connection

* HTTP request message : (format) (ASCII)

(GET/ POST/ HEAD) ← GET /dir/ page.html HTTP /1.1

header lines:
Host : www.xyz.com
User-agent : mozilla
Connection : close
Accept-Language : fr

(for receiving only header (Robots))

request line
Header lines
(carriage return)
Body
(empty in case of GET request)

GET → input is uploaded in URL field
POST → " " " to server in entity body (eg: form data)
PUT → upload file in entity body to path specified in URL
DELETE → delete file specified in URL

HTTP response message:

status line ← HTTP/1.1 200 OK

header lines:
- Connection close
- Date:
- Server:
- Last Modified:
- Content-Length:
- " - Type: text/HTML

→ data ------ (requested file)

Status codes:
- 200 OK
- 300 Moved Permanently
- 400 Bad Request
- 404 Not Found
- 505 HTTP Version Not Supported

Cookies:

→ User-server state eg: shopping cart, authorization (and since HTTP is stateless)

→ During initial HTTP request, site creates unique ID, entry in backend database for ID. (cookie file is kept on user's host ← managed by browser)

→ Cookie header line in HTTP request & response message

Then cookie specific action taken by ~~database~~ server on further request.

Web Caches:

→ satisfy client request without involving origin server

→ Browser sends HTTP request to cache (proxy server) if object present then return the request object from origin server, then return obj to client.

→ reduces response time, traffic on access link.

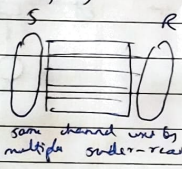
→ Conditional GET: specify date of cached copy in HTTP request, if modified since that date (at server), send response object to proxy server else don't send response object to proxy server

Physical Layer

Functionalities:

→ Cable and connectors
→ Physical topology
→ Hardware (Hubs, repeaters)
→ Transmission mode (duplex, etc)
→ Multiplexing (TDM, FDM, etc)
→ Encoding (digital, analog, etc)

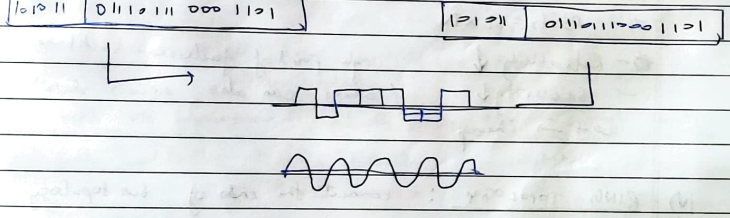
multiplexer demultiplexer
n → 1 → n
(signals)



same channel used by multiple sender-receiver

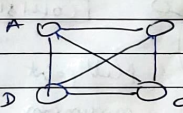
from data link layer

to data link layer



Topology (how the nodes are connected)

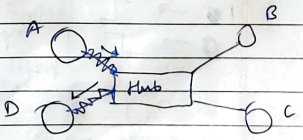
i) Mesh topology:



Each node connected to all other nodes via cables

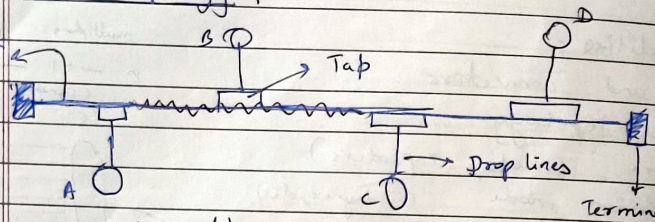
Cost = high
No. of cables = $^{N}C_{2}$
No. of ports = $(N-1)\,N$ (per node $(N-1)$ ports)
Reliability = high (can communicate even though one cable fails)
Security = high
(point to point connection / dedicated))

ii) Star topology / Hub topology:



Cables = N
Ports = 1 × N = N
→ Point to point
→ single point of failure (Reliability ↓ Security ↓) → broadcast by hub

iii) BUS Topology :



to amplify signal in case backbone is very long ← repeater

→ ~~***~~ cables = N + 1 → backbone

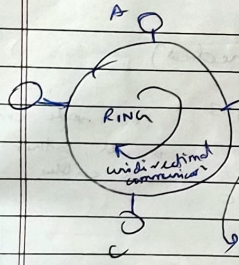
\# Ports = 1 × N

→ Reliability ↓ (single point of failure - backbone)

→ Security ↓ (others can also access data)

Cost → Cheap

iv) RING TOPOLOGY : connect the ends of bus topology



→ Same properties as bus topology

COLLISION → (signals collide from multiple devices)

Multipoint ~~Communication~~ Network (not point to point)

maximum collision = n

can be reduced via TOKENS.

v) HYBRID TOPOLOGY

\# Manchester Encoding & Differential Manchester encoding

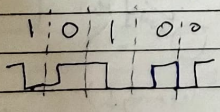
1 → ⊓ | Dr Thomas represent

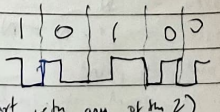
0 → ⊔ | if IEEE represent exchange

0 → ⊓, ⊔

1 → ⊐, ⊏

eg (Dr Thomas)





(start with any of the 2)

Various Devices in Computer Networks:

i) Cables
ii) Repeaters
iii) Hubs
} H/W (hardware)

iv) Bridges
v) Switches
vi) Routers
} H/W + S/W

vii) Gateway
viii) IDS
ix) Firewall
} security

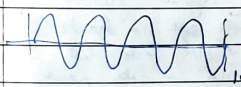
x) Modem (digital ⇔ analog)

Types of Cables:

→ Unshielded twisted pair cable → e.g. in Ethernet LAN
10 Base T : 10 Mbps, 100 m → Bandwidth
→ Base band vs broad band → in base band only one signal can travel on a cable at a time (no collision)

→ Co-axial cable → 10 Base 2 → 10 Mbps, 200 m
10 Base 5 → " , 500 m

→ Fibre Optic → 100 Base Fx : 100 Mbps, ≈ 2 km

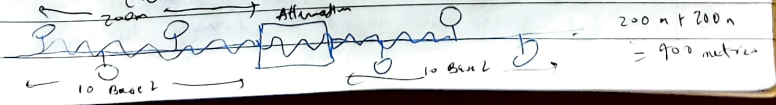


After 100 m, attenuation (low signal strength)

Repeaters: (physical layer hardware device)

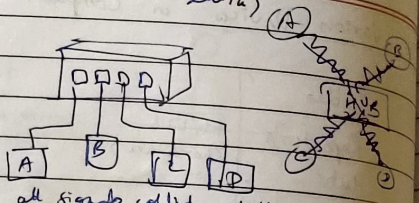
→ 2 port device
→ Forwarding (forwards the incoming signal)
→ No filtering (cannot filter a signal just transmits it)
→ Collision domain = n as there may be collision from all n nodes in the worst case.

→ Function: regenerates the strength to original value (which was lost due to attenuation)



Hub: (physical layer, pure hardware device)

→ Multiport repeater
→ Forwarding
→ No filter
→ Collision domain = n
(when all devices send together, all signals collide at the HUB)



If A sends signal to D, the hub broadcasts to all the devices not just D, hence no filtering
(Also high traffic)

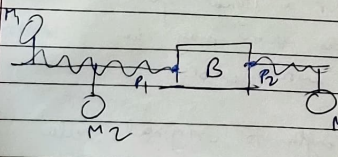
No buffer present at HUB, which can store the packets and then forward preventing collision.

Bridges (Physical & Data Link Layer):

→ To connect two different LANs (2 port device)
→ Forwarding
→ Filtering
→ Bridges can check MAC address (Data Link layer) & decide whether to forward the signal further or not
→ Collision domain: No collision among left side hosts & right side hosts but collision can take place among hosts present at the same side.
→ store & forward of packets
→ Bridges Data Unit Protocol
→ prevents loops through spanning tree so that data packets dont loop infinitely.

Bridges
- Static (by network administrator)
- Dynamic (or transparent)
  - → initially table empty
  - → self learn by looking at sender & reply from receiver
  - → first time broadcast next time onwards table

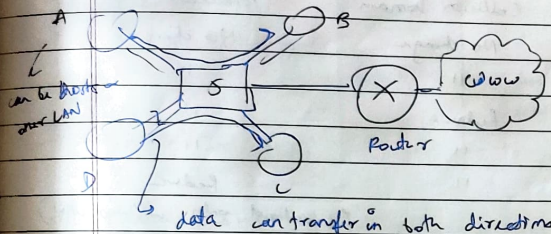
Mac	Port
M_1	P_1
M_2	P_1
M_3	P_2



If transmission from M1 to M2, both are on the same side of the bridge so no need to transmit the signal further

Switch :

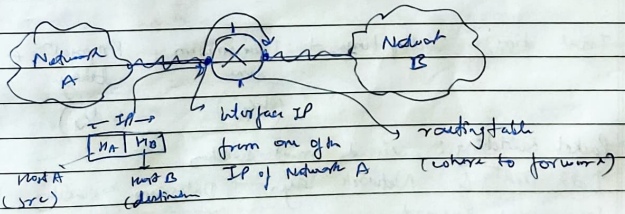
→ Multiport bridge
→ Layer 2 : data link layer device
→ Full duplex link : ~~sender~~ ~~data~~
→ Traffic is minimal (No broadcast) → transmit based on MAC address
→ Collision domain is zero (0)



↳ data can transfer in both directions simultaneously without collision

Routers : (Network layer, data link layer, Physical layer)

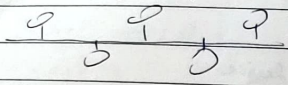
→ multiple ports
→ forwarding
→ filtering (routing table)
→ Routing
→ flooding (broadcasting if required)
→ Collision domain → 0 (store & forward)



SURGE PROTECTED

classmate
Date
Page

Collision domain V/S Broadcast domain :

→ 

Initially collision domain = n (all signals collide simultaneously)
broadcast domain = n (broadcasted signal by any host goes to all other hosts)

Device Name	Collision Domain	Broadcast domain
Repeater	No change	No change
Hub	"	"
Bridge	Reduce	"
Switch	"	"
Router	"	Reduce (broadcast only within a LAN)

(store & forward)

Circuit Switching :

→ Physical layer
→ dedicated circuit, no sharing of bandwidth
→ No headers
→ Efficiency less
→ Delay less
→ high initial set up time
→ Packets received in order

Total time = Setup time + Transmission time (L/R) + Propagation delay (d/v) + Tear down time (to free the resources)

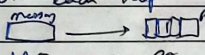
Packet Switching : → virtual circuit switching
→ Data link & Network layer → Datagram switching
→ Store and forward
→ Pipelining used
→ Efficiency ↑
→ Delay ↑

Data transferred in form of packets but works like circuit switching (global packet reserves the resources)

# Datagram Switching	Virtual Circuit
→ Connectionless	→ Connection oriented
→ No reservation	→ Reservation
→ Out of order	→ Same order
→ High overhead (eg ~~header~~ headers with each packet)	→ less overhead (only one global header packet)
→ Packet lost ↑	→ Packet lost ↓
→ Cost ↓ Efficiency ↑	→ Cost ↑ efficiency ↓ (due to reservation of resources like CPU, bandwidth, etc)
→ used in Internet	→ Used in x.25, ATM

Message switching :

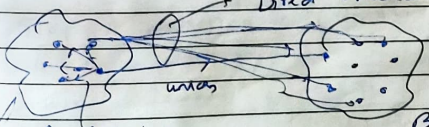
→ Predecessor of packet switching
→ store & forward of messages (not packets)
→ Hop by hop delivery
→ no reservation of resources, higher delay due to processing at each hop



CS → MS → PS

Unicast, Broadcast, Multicast:

one to one — one to all — one to a group



90.0.0.0 — Network A
255.255.255.255 — Broadcast destination address if broadcast within same network

92.0.0.0
92.255.255.255 (broadcast address) if diff network

Data Link Layer :

functions :

→ Hop to hop / Node to Node delivery (Network layer ↑ (not src to dest^n) (eg within a LAN)

→ Flow Control : next hop device such as a router or switch must not be flooded by messages : due to which messages might get lost (buffer becomes full). Hence the sending rate needs to be controlled.

↓ Stop & Wait ↓ Selective Repeat ↓ Go Back N

→ Error Control : (at each node) CRC, Checksum, Parity ↓ Data link layer ↓ Transport layer

→ Access Control (CSMA/CD, ALOHA, Token) (to prevent collision in the network)

→ Physical address (MAC address) → fixed 48 bit address (communication within a local network)
↳ network layer not required for local communicat^n

→ framing of packets (header and tail added)

Flow Control in Data Link Layer :

i) Stop & Wait ARQ : (Stop & wait + timeout timer + data & ack) ← automatic repeat request ← assumes communicat^n channel is perfect & noise free ← sequence no. for data & ack

→ only 1 frame transmit
→ Sender window = 1
→ Receiver window = 1
→ Efficiency $\eta = \frac{1}{1+2a}$ where $a = \frac{T_p}{T_t}$ (also called link utilⁿ) $= \frac{T_t}{T_t + 2T_p}$
→ Retransmission = 1
→ Available sequence No. = $W_{sender} + W_{receiver}$ (window size) minimum sequence no. req.
eg: in LAN ($T_d \ll T_t$)

Total time to send = frame size (packet) / one packet + 2 PD

Throughput = Efficiency × Bandwidth (also called bandwidth utilⁿ)

RTT = $2T_p$

Channel Capacity = Bandwidth × T_p

Go Back N:

→ Available sequence no. = N

→ Multiple frames sent at once

→ Sender window = $2^k - 1$ $(N-1)$

where K = no. of bits to represent window size $(N=8 \Rightarrow K=3)$

→ Receiver window = 1

→ $\eta = \underbrace{(2^k-1)}_{SW\ size} \frac{1}{1+2x}$, $x = \frac{T_p}{T_t}$

→ Cummulative ACK (independent ACK also possible)

→ Retransmission = $2^k - 1$ (worst case)

→ In order packets accepted

(NACK not sent, corrupted packets just discarded. (does not accept out of order frames)

Selective Repeat:

→ multiple frames

→ Sender window size = 2^{k-1}

→ Receiver window size = 2^{k-1}

→ Available sequence no. = 2^k

→ can receive out of order packets

→ $\eta = 2^{k-1} \times \frac{1}{1+2x}$ (counting at receiving side req)

→ Cummulative & Independent ack

→ can also send NACK

→ Retransmission = 1

→ according to sender side (for missing frame) → selected frame is repeated

Sliding Window:

Optimal window size = $1+2a$

Min no. of bits required to represent seq no. = $\lceil \log_2 (1+2a) \rceil$

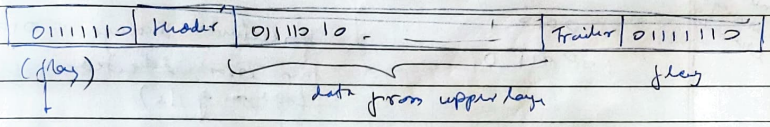
η = sender window size / optimal window size

Framing | Bit Stuffing v/s Byte Stuffing (character):

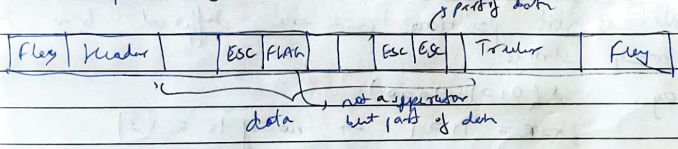
→ Byte stuffing is the process of adding 1 extra byte whenever there is a 'flag' or 'ESC' character in text.

→ Bit stuffing → adding one extra 0 whenever 5 consecutive 1s follow a 0 in data, so that receiver does not mistake the pattern 01111110 for a flag.

Bit stuffing



Byte stuffing



Error Detection and Correction:

→ single bit error : 101 → 100

→ Burst error : 101010 → 111011

length of error = 5

→ If the bandwidth of a channel is 1 Gbps, then for how much duration the error should be? (for 1 bit)

10^9 bits → 1 sec

1 bit → 10^{-9} sec = 1 ns (for single bit error)

If error occurs for 10^{-3} sec, then bits that will be corrupted = $10^{-3} \times 10^9 = 10^6$

→ Error detection → simple parity (even, odd), 2D parity, checksum, CRC (cyclic redundancy check)

→ Error correction → Hamming codes

Single Bit Parity & Hamming distance:

→ m data bits ⇒ (m+1) message bits (single parity bit)

→ Even parity : no. of 1s should be even

→ Can detect all single bit errors in code word (detection only, no correction)

→ Can detect all odd no. of errors also

	data	parity
eg:	0000	0
	0001	1
	0010	1
	0011	0

min hamming distance = 2

even parity

If received message = 01000 then error detected. (odd no. of 1s)

→ Hamming distance(d):

No. of bit positions which are different b/w two binary strings.

eg: d(0101, 1000)

XOR = 1101 → no. of 1s = 3

⇒ d(0101, 1000) = 3

If minimum hamming distance = d, (2)

then can detect (d-1) bits error (1)

Cyclic Redundancy Check : (CRC)

→ based on binary division

→ Total bits in message m+r

where m → data bits, r → redundant bits

→ Polynomial : eg: $x^4 + x^3 + 1$ = 11001

(divisor represented in form of polynomial where each bit is the coefficient of x)

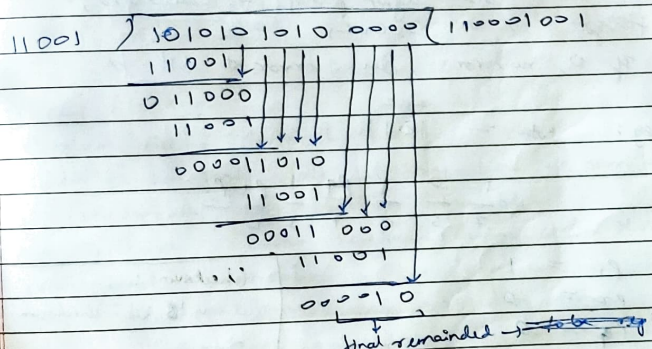
If degree = n, r = n, divisor = (n+1) bits

→ Polynomial should not be divisible by x
Also not with (x+1)

→ Can detect all odd errors, single bit, burst error of length equal to polynomial degree.

eg- ~~Dividend~~ = 1010101010 (10 bit)
Polynomial divisor = x^4+x^3+1 = 11001 (5 bits)

⇒ Dividend = 1010101010 0000
↳ to be replaced by remainder



∴ Message = 1010101010 0010

→ On receiver side, Message divided with divisor
If remainder = 0 ⇒ message correct
else error detected

Utilization = $\frac{10}{14} \times 100$ $\left(\frac{m}{m+r} \times 100\right)$

Hamming Code for error detection & correction:

→ m : data bits , r : redundant bits (parity bits)

$2^r \geq m + r + 1$

→ Redundant bits placed at 1, 2, 4, 8, ...

→ Parity bits C_i

M_1 : C_1 = parity (1, 3, 5, 7, 9, ...)

M_2 : C_2 = parity (2, 3, 6, 7, 10, ...)

M_4 : C_3 = parity (4, 5, 6, 7, 12, 15, ...)

→ At receiver side again parity is calculated,

to get $x = C_3 C_2 C_1$

If 0, no error , else error at bit x.

eg : data = 1011 (bit 4 3 2 1)

1	0	1	P_4	1	P_2	P_1
7	6	5	4	3	2	1

$P_1 = 1$

$P_2 = 0$

$P_4 = 0$

	7	6	5	4	3	2	1
Message	1	0	1	0	1	0	1
Received message	1	0	1	0	1	0	1

$C_1 = 0$

$C_2 = 1$

$C_3 = 1$

error at bit (110) i.e 6

Checksum :

bit say 16 bit checksum

Data :

1001 0011 0010 0011

1001 1000 1001 1101

Data is of 32 bits :

divided into segments of 16 bits

Now added

sum = 1 0010 1111 0000 0

0010 1011 1111 0000 +

= 0010 1010 1111 0001

1's complement

checksum value

1101 0101 0000 1110

1's complement of (At receiver, sum of all segments + checksum) = 0 then no error

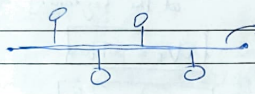
Access control : controls
→ access of stations to the transmission link
(broadcast link requires access control)
— to prevent / handle collision

Various Medium Access Control Protocols :

Data link → LLC (logical link control) → error, flow control, etc
→ MAC (medium access control)

Multiple Access Protocols (MAC)

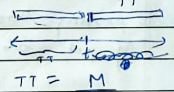
- Random Access Protocols
  - Aloha
    - → Pure
    - → Slotted
  - CSMA
  - CSMA/CD
  - CSMA/CA
- Control Access
  - Polling
  - Token Passing (token ring)
- Channelization Protocol
  - FDMA
  - TDMA



MAC : To control collision with shared medium

Pure Aloha :

→ Random Access Protocol
(sender can send data at any time as soon as data ready) ⇒ collision possible
→ ACK is there (when receiver receives data it sends Ack)
→ LAN based protocol
→ Only transmission time, no propagation time



→ **Vulnerable Time = $2 T_t$**

$T_t = \frac{M}{B/b}$

→ **Efficiency $\eta = G e^{-2G}$**
where G = no. of stations who want to transfer data at a particular time slot

if any other sender sends the data in this duration ($2T_t$), collision will occur

Max efficiency : $\frac{\partial \eta}{\partial G} = 0$

$\Rightarrow G = 1/2$ (when half of the stations transmit at a time)

$G e^{-2G} = \frac{1}{2} e^{-1} = 0.184 = 18.4\%$

Pure Aloha

→ At any time transmission can start by any sender.

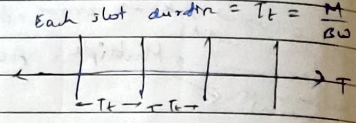
→ $VT = 2 \times T_t$

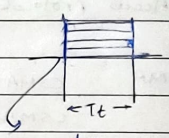
→ $\eta = G e^{-2G}$

→ Max = 18.4%

Slotted Aloha

Each slot duration = $T_t = \frac{M}{BW}$





any sender can start transmission at the beginning of the slot only

$$V_T = T_t$$

max when $G = 1$, $\eta = 36.8\%$. ← $\eta = G e^{-G}$

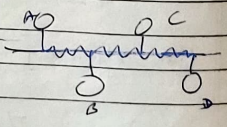
Carrier - Sensor Multiple Access (CSMA):

1 Persistent | 0 persistent | p persistent

→ ~~Keep on checking~~ sense the carrier before sending data. (transmitter attached to every node checks any signal passing through that point or not)
If not busy (no signal passing), then can send data.

→ 1 persistent :
Continuously sense until their is no signal passing and then immediately send data.

→ As soon as ~~sig~~ data is transmitted completely from A to D, B & C will immediately send data if available hence high collision



ii) 0-persistent :

→ If medium idle send immediately, if busy → wait for a fixed time before checking again

→ lesser collision compared to 1-persistent.

→ cannot send before the time even if medium becomes idle before.

iii) p-persistent :

→ checks continuously and as soon as found idle, will send with probability p. (to reduce collision)

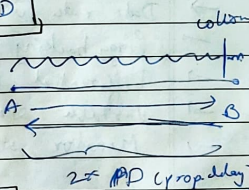
→ Wifi uses this (0<p<1)

CSMA/CD (Carrier Sense Multiple Access / Collision Detection):

⇒ No ACK (previously required to check if collision occurred)

→ sender / station can detect collision if it senses the collided signal while it is transmitting the message. (sender must be sending message when it senses the collided signal)

→ ∴ To detect collision, $TT \geq 2 \times PD$



→ CSMA/CD used : LAN

mud still be transmitting when receives collided signal (after 2xPD in worst case)

$$\eta = \frac{1}{1 + 6.44a} \quad \text{where} \quad a = \frac{PD}{TT}$$

d) Consider CSMA/CD network, transmit data @ 100 Mbps over 1 km cable with no repeaters. If minm. frame size required for the network is 1250 bytes, what is the signal speed?

$TT \geq 2PD \Rightarrow TT = 2PD \Rightarrow \frac{L}{BW} = \frac{2d}{v} \Rightarrow v = \frac{2 \times D \times BW}{L} = \frac{2 \times 1 \times 10^3}{1250 \times 8}$ [km/s]

Ethernet Frame Format:

min: 46B, max: 1500B

PREAMBLE	SFD	DA	SA	Length	Data	CRC
7B	1B	6B	6B	2B	46B - 1500B	4B

Added by Physical Layer (to alert the receiver) — destination & src MAC address (48 bit) — error control

- IEEE 802.3
- 10 Base 2 – Thin
- 10 Base 5 – Thick
- 10 Base T
- 10 Base FX – Fast
- 10G Base T – Gigabit
- Topology: Bus, star
- 1 Mbps – 400 Gbps

Token Ring (IEEE 802.5):

- Ring topology used
- Access control method used is token passing
- Token ring is unidirectional
- Data rate used is 4 Mbps & 16 Mbps
- Piggy backing acknowledgement is used
- Differential Manchester encoding is used
- Variable size framing
- Monitor station is used

Efficiency = $\frac{N \times T_t}{T_r + N \times (THT)}$

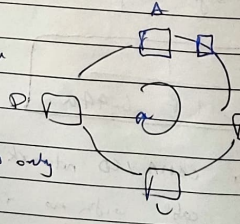
THT → total holding time

(Sender takes the token while it is transmitting & releases the token when done)

(one transmission at a time as only one can hold the token

- for delayed token reinsertion, THT = $T_t + T_r$

Early token reinsertion / THT = T_t



Classmate
Date
Page

Q) Channel has bit rate 1 Mbps. PD = 270 ms
Frame size = 125 bits
ACK is always piggybacked into data frames
4 bit sequence no. is used (ignore header size)
What is the maximum available channel utilization of
GO BACK N?

If Window size = $2^4 - 1 = 15$

F = 125 bytes = $125 \times 8 = 1000$ bits

$T_t = \frac{1000 \text{ bits}}{1 \text{ Mbps}}$

$= 10^{-3}$ s = 1 ms

$$U = \frac{W}{1 + \frac{2T_p}{T_t}} = \frac{15}{1 + \frac{2 \times 270 \text{ ms}}{1 \text{ ms}}}$$

Q) A channel has a bit rate = 4 Kbps and a long PD = 20 ms.
STOP & WAIT protocol is used,
transmission time of ack is negligible.
To get channel efficiency of at least 50%,
the minm. frame size = ?

U = 1/2, BW = 4 Kbps
T_p = 20 ms, frame size = L bits

$$U = \frac{W}{1 + 2a} \Rightarrow \frac{1}{1 + \frac{2T_p}{T_t}} \geq \frac{1}{2}$$

$$\frac{1}{1 + \frac{2 \times L \text{ bits}}{4 \text{ Kbps}}} \geq \frac{1}{2} \Rightarrow 1 + \frac{2T_p}{T_t} \leq 2$$

$$\frac{2T_p}{T_t} \leq 1$$

$2T_p \leq T_t$

$2 \times 20 \times 10^{-3} \leq \frac{L}{4 \times 10^3}$

$L \geq 160$ bits

2) Determine the max length of cable (in Km) for transmitting data at rate of 500 Mbps. in an Ethernet LAN with frame size = 10 Kb
Assume signal speed in cable is 200000 Km/s

$T_t \geq 2T_p$

$\frac{10 \times 10^3 \text{ bits}}{500 \times 10^6 \text{ bits/s}} \geq 2 \frac{d \text{ Km}}{2 \times 10^5 \text{ Km/s}}$

$\frac{10^4}{5 \times 10^6} \geq d$

$2 \geq d$

max cable length = 2

TDM :

→ Time slots to each station in Round Robin
Size of time slot = $T_t + T_p$
Efficiency = $\frac{T_t}{T_t + T_p} = \frac{1}{1+a}$, $a = \frac{T_p}{T_t}$

Polling :

→ polling created : all stations willing to send data
→ polling algo chooses one station and station sends data
→ Efficiency (η) = $\frac{T_t}{T_{poll} + T_t + T_p}$ assume T_{poll} = time taken for polling
→ Effective bandwidth = $\eta \times$ bandwidth
→ Max available Effective Bandwidth = (No. of stations) × (bandwidth req. of 1 station)

Network Layer

Functions :

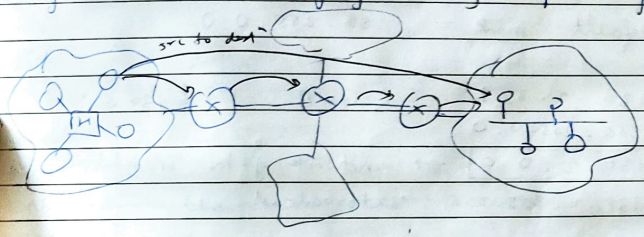
→ Host to host (source to destination delivery)

→ Logical addressing (IP) → Network, Host

→ Routing : select the best / shortest path for the packet to reach the destination (done by router)
RIP, OSPF

→ Fragmentation (in case message size becomes large)

→ Congestion control (majorly done by transport layer)

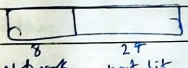


Classful Addressing : A, B, C, D, E :

→ Dotted decimal notation

→ 32 bits (4 octets separated by .)

0.0.0.0 to 255.255.255.255



* **Class A** : leftmost bit is 0 : 0.0.0.0 to 127.255.255.255

→ No. of IP address = 2^{31}

→ No. of networks in class A = $2^7 = 128$

Usable networks = 128 − 2 = 126 (0.0.0.0 → Network address, 127.0.0.0 → loopback)

→ No. of ~~networks~~ hosts possible in every network = 2^{24} reserved

Usable host ids = $2^{24} - 2$ ((— . 0.0.0) → for the entire network, (— . 255.255.255) → for broadcast address (if other network wants to send to all the hosts in the current network))

→ Default mask = 255.0.0.0 (first & last IP address)

eg: 64.0.0.0 → for network

64.0.0.1 → first host in network

→ to get the network of the IP address, (IP & mask) = (64.0.0.1) & (255.0.0.0) = 64.0.0.0

Class B : prefix = 10 (first 2 bits reserved)

Network ← 10 _ _ _ _ _ _ . ◯ | ◯ . ◯ → Hosts

→ Range = 128.0.0.0 to 191.255.255.255

→ No. of addresses = 2^{30} (25%)

→ No. of networks = $2^{6+8} = 2^{14} = 16384$

→ No. of hosts in every network = $2^{16} = 65536$

Usable = $2^{16} - 2$

→ Default mask = 255.255.0.0

eg: 130.2.3.4 (class B 128 ≤ 130 ≤ 191)

& 255.255.0.0

130.2.0.0 → network of the IP address

130.2.255.255 → directbroadcast address

Class C : prefix = 110

Network ← 110 _ _ _ _ _ . ◯ . ◯ | ◯ → host

→ Range : 192 to 223

→ No. of addresses = 2^{29} (12.5%)

→ No. of networks = $2^{5+8+8} = 2^{21} \approx 20$ Lakhs

→ No. of hosts in every network = $2^8 = 256$

Usable = $2^8 - 2 = 254$

→ Default subnet mask = 255.255.255.0

eg: 192.2.3.4 → class C

255.255.255.0

192.2.3.0 → Network ID

192.2.3.255 → directbroadcast address

Class D : prefix : 1110

1110 ---- . . . .

→ Range : 224 to 239

→ No. of IP addresses = $2^{4+24} = 2^{28}$ (6.25%)

→ No networks / hosts (all IP are reserved for multicasting, group email / broadcast.

Eg: 239.1.2.3

Class E : prefix 1111

1111 ---- . . . .

→ Range = 240 to 255

→ No. of IP addresses = $2^{4+24} = 2^{28}$

→ No networks / hosts (all IP reserved for military purpose)

Eg: 245.0.1.2

Q) IP address = 201.20.30.40 ∵ 192 ≤ 201 ≤ 223 ⇒ class C

Calculate Mask for class C = 255.255.255.0

i) Network ID = 201.20.30.0

ii) 4th host ID = 201.20.30.4

iii) Last host ID = 201.20.30.254

iv) Broadcast address → direct broadcast addr → 201.20.30.255

→ limited broadcast addr → 255.255.255.255

Limited → when a host wants to broadcast within his own network

Direct → some other network's host want to broadcast to current network then uses current network's direct broadcast address (201.20.30.255)

Disadvantages of classful addressing :

→ wastage of addresses (no flexibility) → eg Class D and Class E (if org. wants 1024 IPs, Class B will provide 65K (wastage), Class C → 256 (not sufficient))

→ maintenance is time consuming

→ more prone to errors

Sol^n : Classless IP addressing, subnetting, etc.

Classless Addressing (1993) (CIDR):

→ Addresses should be contiguous (.0, .1, .2, .3, ...)

→ No. of addresses in a block must be a power of 2. (here 2^4 in the following example)

→ First address of every block must be evenly divisible with size of block. (200.10.20.32 / 28)

→ No classes, only blocks

eg: ← 32 → (IPv4)

28	4
Block ID	Host ID

200.10.20.0010000

size of block = 2^4 = 16

last 4 bits 0

→ Notation: x.y.z.w/(n) (n) → mask; no. of bits used for block/network

eg: 200.10.20.40 / 28

→ Mask = 11111111.11111111.11111111.11110000

= 255.255.255.240

→ Network ID = 200.10.20.00101000

255.255.255.11110000

200.10.20.00100000

= 200.10.20.32 / 28

→ No. of hosts = 2^4 = 16

Subnetting in Classful Addressing:

eg: Let's take a class C address 200.10.20.0 to 200.10.20.255

200.10.20.0 ---------- Subnet mask = 255.255.255.128 for both S1 & S2

200.10.20.1 ----------

S1 (126) S2 (126)

subnet ID for S1 ← 200.10.20.0 to same → 200.10.20.0 default mask 255.255.255.0

broadcast address for S1 ← 200.10.20.127

No. of hosts = 128

Usable = 126

200.10.20.128 → subnet ID for S2

to

200.10.20.255 → broadcast ID of S2

Hosts = 128

Usable = 126

WWW

* Initially 256 − 2 = 254 hosts possible

Now 126 + 126 = 252 hosts possible

* Don't change the network ID bits for subnetting otherwise the entire network would change.

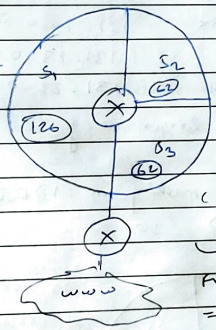
→ Network ID of entire network & subnet s1 is same but no issue as

eg: If packet is to be sent to 200.10.20.15, external router uses external mask (default) to find network ID = 200.10.20.0. Now packet goes to internal router which uses the mask 255.255.255.128 to get subnet ID = 200.10.20.0 which is S1. Now packet goes to 15th host of S1.

eg: If to be divided into 4 subnets, reserve 2 bits from the host ID part (00------, 01------, 10------, 11------)
Then subnet mask = 255.255.255.110----- = 255.255.255.192
(subnet mask is same for all 4 subnets as divided equally)

Variable Length Subnet Masking :

eg: 200.10.20.0 to 200.10.20.255 (class C)



200.10.20.0 ------- (subnet ID) to 200.10.20.127 (broadcast)

Subnet mask = 255.255.255.128

200.10.20.1 0 ------- (200.10.20.128 to 200.10.20.191) subnet ID, subnet broadcast

200.10.20.1 1 ------- (200.10.20.192 to 200.10.20.255) subnet ID, subnet broadcast

For S2 and S3, subnet mask = 255.255.255.192

Usable host IDs = 126 + 62 + 62 = 256 - 6 = 250

Subnetting in Classless Inter-domain Routing (CIDR) :

eg: 195.10.20.128/26

195.10.20.100------

195.10.20.128/27 to 195.10.20.159/27

195.10.20.101------

195.10.20.160/27 to 195.10.20.191/27

Now 27 network ID bits / Block ID bits

Q) CIDR receive a packet with address 131.23.151.76
The router's routing table has following entries :

Prefix	Output interface	
131.16.0.0/12	3	✓
131.28.0.0/14	5	✗
131.19.00/16	2	✗
131.22.0.0/15	1	✓

Packet will be forwarded to which interface ? 1

131.23.151.76

131. 00010111 .151.76

masking with first 12 bits : 131.16.0.0 ✓ (correct network ID)
14 bits : 131.20.0.0 ✗
16 bits : 131.23.0.0 ✗
15 bits : 131.22.0.0 ✓

Choose the one with larger no. of 1s in the prefix (network ID)

Variable length subnet masking in CIDR :

Eg 245.248.128.0/20

245.248.10000000.00000000

245.248.10000---- S1

245.248.128.0/21
to
245.248.135.255/21

S2 245.248.136.0/22 to 245.248.139.255/22

S3 245.248.10001---- 245.248.140.0/22 to 245.248.143.255/22

IPV4 header format:

⇒ Connectionless service → 0100 (IPV4)

⇒ Datagram service

VER 4	HLEN 4	Type of service (DSCP) 8	Total Length 16
Identification bits 16		Flag 3	Fragment bits 13
Time to live TTL 8	Protocol 8	Header Checksum 16	
Src IP Address 32			
Destination IP Address 32			
Options and Padding (optional)			

HLEN: multiplied by 4 to get header length (0, 1, 2, 3, 4 → not possible); 1111 → 56, 1000 → 60

Total Length: of the datagram

Flag 3 + Fragment bits 13: used for fragmentation

Time to live TTL 8: Protocol 8: Header Checksum 16: error check 20 bytes

Differentiated Service Code Point (DSCP)

P	P	P	D	T	R	C	0

0 reserved (not used)

P: precedence (priority)

D: delay, T: throughput, R: reliability, C: cost

DSCP | ECN

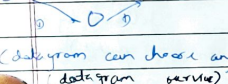
DSCP: 0–5

ECN: 6–7 → Explicit congestion notification

Datagram → Header (metadata) size: 20–60 bytes + Payload (actual data) → comes from upper layer

size = 0 – 65515 bytes

Max datagram size = 65515 + 20 = 65535 ($2^{16}-1$)



(datagram can choose any path)

(datagram service)

Fragmentation of IPV4 datagram:

→ packets that become large in size need to be fragmented into smaller fragments and later combined to form original packet. Hence identification bits are used,

Fragmentation:

Identification bits 16	Flag 3	Fragment offset 13

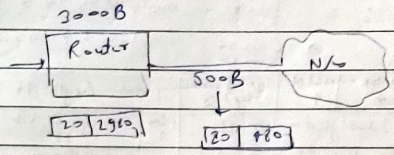
Reserved 0	DF	MF

DF: don't fragment (if 1, not fragmented)

MF: more fragment (if 1 more fragment pending)

DF	MF	
0	0	
0	1	more fragment
1	0	
1	1	

Q) A datagram of 3000 B (20 B of IP header, 2980 B IP payload) reached at router and must be forwarded to link with MTU of 500 B. How many fragments will be generated? Also write MF, offset, total length value for all.



Total fragments = $\left\lceil \frac{2980}{480} \right\rceil = 7$

Header added to all fragments

	P_7	P_6	P_5	P_4	P_3	P_2	P_1
	(100+20)	(480+20)	. . .	. . .	. . .	(480+20)	(480+20)
	= 120	500	- - -	- - -	- - -	500	500
MF	0	1	1	1	1	1	1
Offset	360	300	240	180	120	60	0

$2980 - 6 \times 480$
$= 2980 - 2880$
$= 100$ B (remaining)

(last packet is not followed by any packet)

eg: For P_3, no. of bytes of data to the right of P_3 = 480 + 480 = 960
Now result is divided by 8 to get the offset = $\frac{960}{8} = 120$

Options and Padding: 0 - 40 Bytes

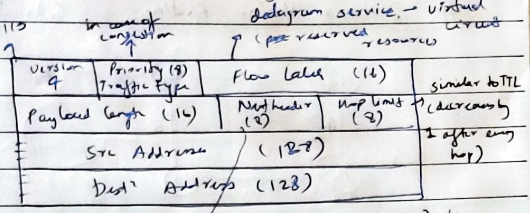
Used for:

i) record route (record the IP address of all the routers on the path to the destination & store in the header)

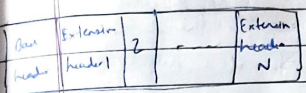
ii) Source routing (path through routers (fixed) predetermined)
→ strict source routing (strict path Src → R_1 → R_2 → R_3 → Dest)
→ loose source routing (Src → ... → R_1 ... → R_3 → R_6 → ... → Dest)

iii) Padding : if header size is not a multiple of 4 additional zeros appended to make the size multiple of 4
(20 - 60 B)

IPV6 header :



Base header = 40 bytes (320 bits), fixed



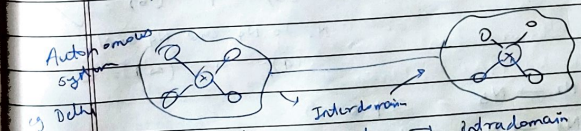
Extension headers :

i) Routing header (43)
ii) Hop by hop option (0)
iii) Fragment header (44)
iv) Authentication header (51)
v) Destination options (60)
vi) Encapsulation Security payload (50)

Routing Protocols : (forwarding the packet to the right path) (optimal)

- Intra domain
  - Distance Vector (RIP) (Routing Info. Protocol)
  - Link State (OSPF) (Open shortest path first)
- Inter domain
  - Path Vector (BGP) (border gateway protocol)

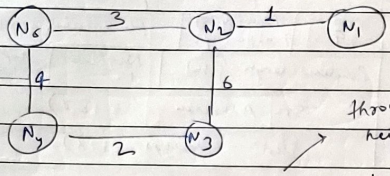
→ Routing table used : static or dynamic (auto updation)
→ updated time to time (values to transfer the packets)



Within a autonomous system → Intradomain routing
→ These protocols are for unicasting (one to one communication)

classmate
Date
Page

Distance Vector Routing: (Intradomain)



through 'Hello' message to the neighbours

Initial routing table for N5: (distance vector)

i) Only share with neighbour
ii) Only distance vector is share (not the entire bandwidth)

i)

Destination	Distance	Next
N_1	∞	-
N_2	3	N_2
N_3	∞	-
N_4	4	N_4
N_5	0	N_5

N5 on receives the distance vectors of its neighbours N_2 and N_4

N_2	N_4
1	∞
0	∞
6	2
∞	0
3	4

ii) Updated table in the next iteration

Dest^n	Dist	Next
N_1	4	N_2
N_2	3	N_2
N_3	6	N_4
N_4	4	N_4
N_5	0	N_5

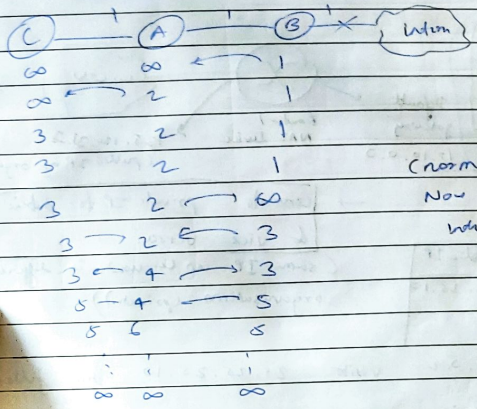
N_5 to N_1 = N_5 to N_2 (3) and N_2 to N_1 (1) : 4 ✓
or N_5 to N_4 (4) and N_4 to N_1 (∞) = ∞

N_5 to N_2 = N_5 to N_2 (3) and N_2 to N_2 (0) = 3 ✓
or N_5 to N_4 (4) and N_4 to N_2 (∞) = ∞

N_5 to N_3
= $N_5 \rightarrow N_2$, $N_2 \rightarrow N_3$ = 3 + 6 = 9
or $N_5 \rightarrow N_4$, $N_4 \rightarrow N_3$ = 4 + 2 = 6 ✓✓

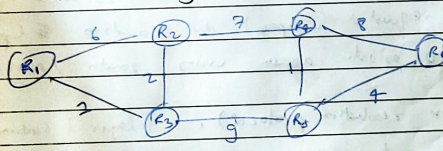
→ Algorithm repeated for some iterations for each of the nodes (similar to Bellman ford)

Count to Infinity problem: (in distance vector routing)



(normal case)
Now if link from B to Internet fails

Link State Routing:



Link state Table (through Hello message) :

R_3
R_1 3
R_2 2
R_5 9

} other info also added

→ **Flooding** is done by each host (~~broadcasts~~ it every host broadcast (sends to all) its link state table to every other host (not just the neighbours).

Dest	Dist	Via
R_1	0	R_1
R_2	6	R_1
R_3	3	R_1
R_4	12	$R_3 R_2$
R_5	12	R_3
R_6	16	$R_3 R_5$

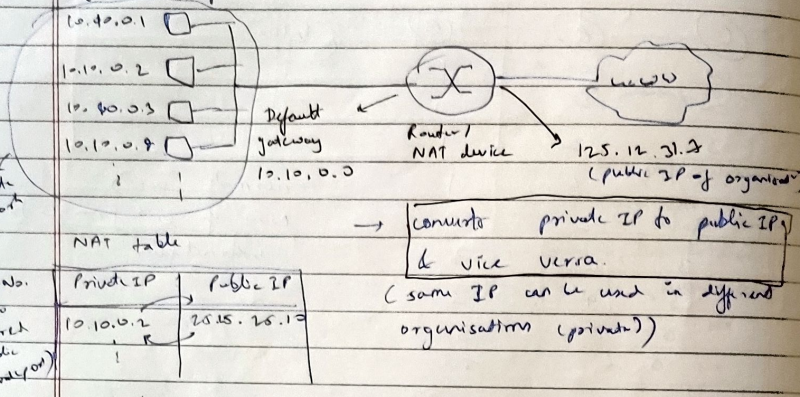
→ Now R1 has info of all tables, thus it forms a graph (through global knowledge)

Routing table for R1

→ Single source shortest path (Dijkstra) used to compute the routing table

Network Address Translation (NAT):

eg range of IP = 10.0.0.0 to 10.255.255.255



→ converts private IP to public IP & vice versa.

(same IP can be used in different organisations (private))

if host 10.10.0.2 visits 25.25.25.10 this filled in NAT table.

The packet src address changes from private to public through the NAT device while sending to 25.25.25.10 & the reply (requested webpage) dest address is (public) translated changed to private again using router / NAT device.

ARP (address resolution protocol); (layer 3 (internet layer))

→ function : convert IP to MAC address

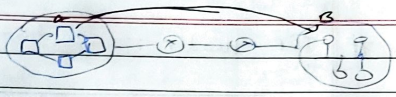
(logical → physical)

→ eg host A broadcasts to entire network for a specific IP and then whichever host has that IP, it responds (unicast) with its MAC address.

Transport Layer

classmate
Date
Page

Functions :



i) End to end delivery (port to port) | packets to segments
(process to process)
(multiple processes / applications may be running at sender & receiver end) eg: HTTP : port 80
SMTP : port 25

ii) Reliability (done by TCP)
→ in order delivery ↳ connection oriented
→ no data loss

iii) Error control : using checksum
(done at the destination (not node to node))

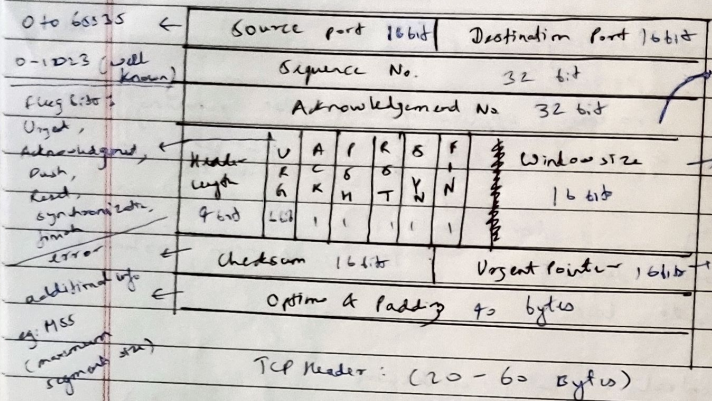
iv) congestion control (overflow of buffer of routers)
eg: AIMD (additive increase multiplicative decrease)

iv) flow control : stop & wait, Go Back N, Selective Repeat
(receiver buffer full)

v) Multiplexing and demultiplexing :
— Multiple process from sender side go through the same channel (many to one) ⇝ multiplexing.
→ At receiver end, multiple messages coming from the common channel are sent to the correct process (one to many) ⇝ demultiplexing.

TCP :
→ Byte streaming (bytes to segments)
→ Connection oriented : 3 WAY handshake, reliability
→ Full duplex (sender & receiver can send simultaneously)
→ Piggy backing (acknowledgement & data simultaneously)
→ Error control (checksum), flow control, congestion control

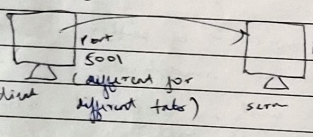
TCP Header :



TCP Header : (20 - 60 Bytes)

Minimum : 20 × 8 = 160 bits

eg: HTTP server



→ Header length : multiplied by 4 to get length

eg: 0101 to 1111 (20 to 60 bytes)

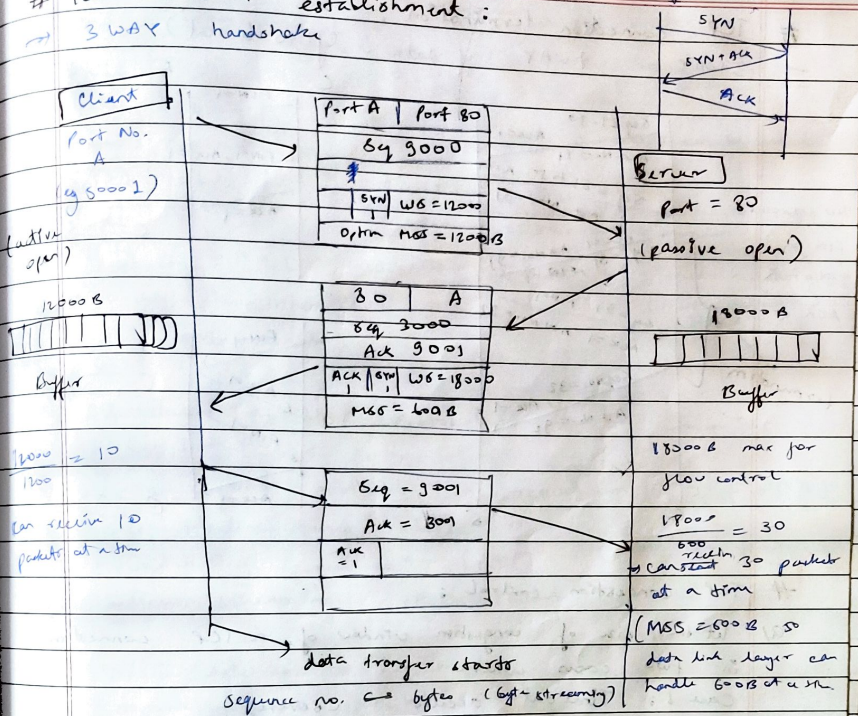
↓ 5 × 4 = 20 ↓ 15 × 4 = 60

→ Push : send received message immediately to the application (don't wait for buffer to be full)

→ SYN, FIN : for establishing connection

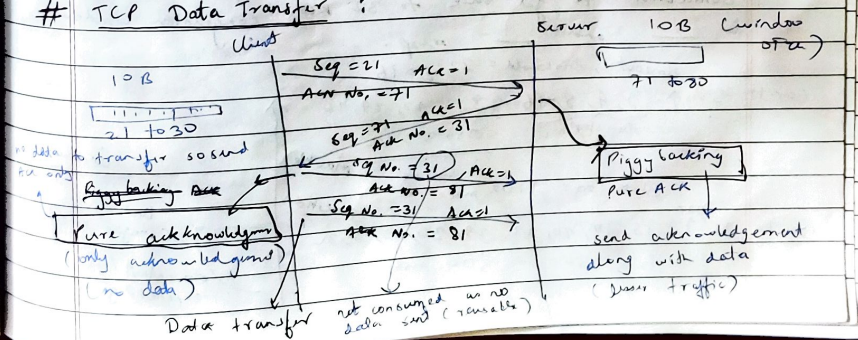
TCP connection establishment :

→ 3 WAY handshake



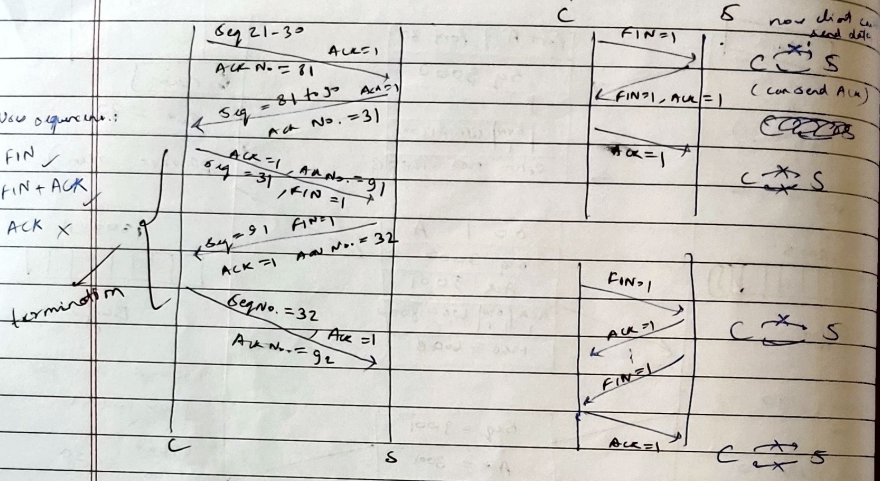
→ After connection establishment resources like CPU, buffer, bandwidth are reserved.

TCP Data Transfer :



TCP connection termination :

→ 3 WAY or 4 WAY (if data is left to be sent)



TCP congestion control :

Q/ Let the size of congestion window of a TCP connection in two cases where

Case 1: Time out occurs (severe)

Case 2: 3 ACK received (light)

is 32 KB. The RTT of connection is 100 msec. MSS = 2KB. Time taken (msec) by TCP connection to get back to 32 KB congestion window is ____ and ____ resp.

Timeout, threshold = 32/2 = 16

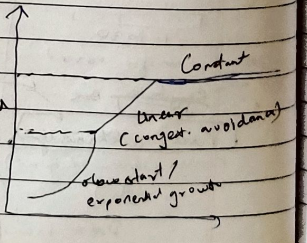
Case 1: 32 ↓ 2, 4, 8, 16, 18, 20, 22, 24, 26, 28, 30, 32

slow start | linear

11 × 100 = 1100 msec

Case 2: 32 ↑ (32/2) 16, 18, 20, 22, 24, 26, 28, 30, 32

8 × 100 = 800 msec



- Start with MSS. Slow start upto max window size/2 then upto max receiver window size → congestion avoidance

classmate Date Page

UDP header : → connectionless, No order
User datagram protocol

Source port 16 bits	Dest port 16 bits
Total length 16 bits	Checksum 16 bits

0 to 65535

0–1023 well known. rest for registered others can be used

65535 (max)

Header	Payload
8 Byte	65527 B → maximum

Checksum = UDP header + UDP data + Pseudo header of IP (more values that are fixed)
↳ optional for IPv4
(unreliable)

Advantages of UDP Protocol: faster than TCP
DNS, DHCP

i) Query response protocol (one request one reply) [DNS] DHCP
ii) Speed (online games, VoIP)
iii) Broadcasting / multicast [RIP] (routing info. protocol)
iv) Continuous streaming (Skype, Youtube)

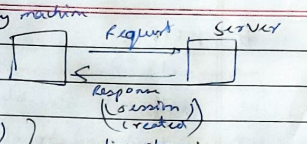
→ UDP is stateless ⇒ less overhead
(no user information, past info stored)

#	TCP	UDP
→	Connection oriented	→ connectionless
→	reliable (ordering)	→ less reliable (No order)
→	error control mandatory (checksum)	→ error control is optional
→	slow transmission	→ faster transmission
→	more overhead (larger header) (20–60 B)	→ less overhead (8 B)
→	flow control, congestion control	→ No FC, CC
	eg: HTTP, FTP, SMTP uses TCP.	→ eg: DNS, BOOTP, RIP, DHCP uses UDP

Application Layer

classmate
Date
Page

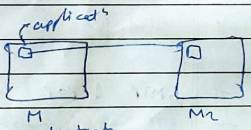
Session layer :



* Functions :
  - i) Authentication (identity verified) } eg online shopping : cart, payment, etc
  - ii) Authorization (what all access) }
  - iii) Session restoration (checkpoints) (eg network error, download resumes from previous checkpoint)
    (eg restoring sessions after power cut)
  - iv) Webinar (flow control and synchronization)
    (eg audio & video synchronized)

→ Session layer is not implemented in the Operating system rather needs to be implemented in the applicⁿ.
→ Same for presentation layer.

Presentation layer :



* Functions :
  - → Code conversion (formatting)
  - → Encryption / decryption (security)
  - → Compression (eg zip files)

plain text —encrypt→ ciphertext —decrypt→ text
using keys encrypted

Applicⁿ layer protocols & port No.

*	Protocol	Port No.	Transport protocol
✓	FTP	20/21 (data/control)	TCP
	Secure shell (SSH)	22	TCP
✓	Telnet	23	TCP
✓	SMTP (push)	25	TCP
✓	DNS	53	UDP ✓
	DHCP	67/68	UDP ✓
✓	HTTP	80	TCP
	POP (pull)	110	TCP
	HTTPS	443	TCP
	RIP (distance vector)	520	UDP ✓

HTTP :

→ port 80
→ stdy not reliable (uses TCP for reliability)
→ Inband protocol (commands & data sent over same port (80))
→ Stateless (no metadata stored)
→ Non persistent & persistent
→ Commands (Head, Get, Post, Put, Delete, Connect)

FTP:

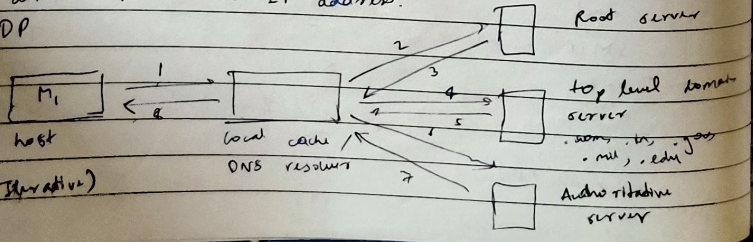
→ Port 20 for data and 21 for control commands (out inband)
→ Data connection is not persistent
→ Control connection is persistent
→ reliable,
→ Stateful

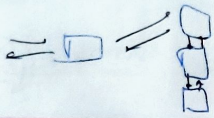
SMTP & POP :

→ FTP is synchronous (both active)
SMTP & POP both synchronous & non-synchronous
→ Port 25 for pushing the mail (SMTP)
→ POP3 by default works on 2 ports:
110 → default, non encrypted
995 → secured
→ Multipurpose Internet Mail Extensions (MIME) (protocol)

Domain Name Server :

→ domain name to IP address.
→ UDP





(recursion)

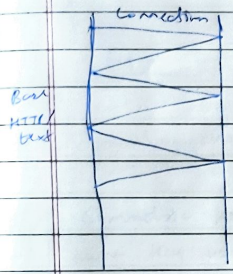
classmate
Date
Page

Persistent HTTP

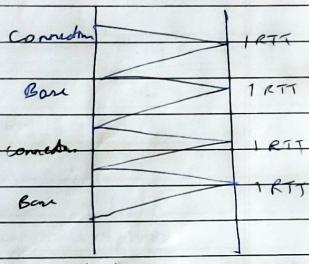
→ server leaves connection open after sending response (1 RTT for all referenced object)
→ less overhead

Non-persistent HTTP

→ requires 2 RTTs per object
→ More overhead
→ connection established everytime



eg: chrome, gmail



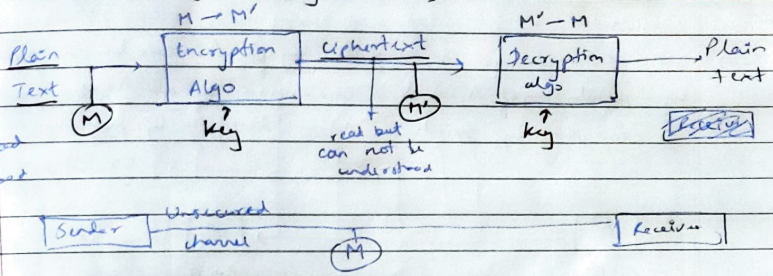
eg: online transaction

Network Security

Classmate
Date
Page

Cryptography :

→ CIA : confidentiality , Integrity , Availability

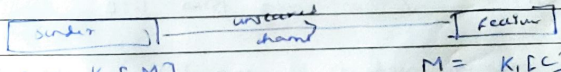


→ Symmetric Key & Assymetric Key (eg RSA)

Symmetric Key :

→ same key used for encryption and decryption

→ (DES , 3DES , AES) ; symmetric Key Algos
Data encryption standard , advanced encryption algo
56 bits 192 bits (128 bits, 192, 256 bits)



$C = K_1[M]$ (hash function)

$M = K_1[C]$

→ Challange : Key exchange : how to send the key to the receiver as channel is unsecured.

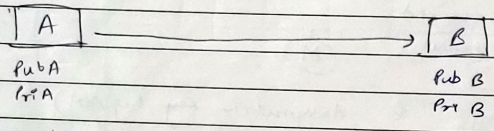
→ If n devices are connected with each other, the maximum no. of keys (symmetric) required = $^nC_2$
(no. of edges in complete graph)

Asymmetric key or Public

→ Every node/host has a public and private key.

→ anyone can know the public key but private key is confidential.

→ Encrypted message by public key of A can be decrypted by private key of A. & vice versa is also true.
(encryption by private key (A) → decryption by public key (B)



4 cases : Encryption by:

i) Pub A [M] → cannot be decrypted as no one has private key of A except A

ii) Pri A [M] → can be decrypted by anyone as everyone has the public key of A

iii) Pri B [M] → not possible as A does not have Pri B

iv) Pub B [M] → ✓ as only B has Pri B

→ Hence message needs to be encrypted by the receiver's public key.

→ If n nodes need to communicate with each other
Keys required = $2n$ (each node → 2 keys)

RSA Algorithm: (asymmetric keys)

Q) In RSA, node A uses 2 prime no. p = 13 and q = 17 to generate his public & private key. If public key of A = 35, then private key of A = (A) 11 (B) 13 (C) 15 (D) 17

Algorithm:

i) Choose 2 diff. large prime no. $p = 13$ $q = 17$

ii) Calculate $n = p \times q$ $n = 13 \times 17 = 221$

iii) Calculate $\phi(n) = (p-1) * (q-1)$ $\phi(n) = 12 \times 16 = 192$

iv) Choose e : $1 < e < \phi(n)$ $e = 35$ (given)

e is coprime to $\phi(n)$ i.e. $\gcd(e, \phi(n)) = 1$

v) Calculate d: $de \equiv 1 \pmod{\phi(n)}$ $de = 1 + K\phi(n)$

vi) Publickey = 'e', private key = 'd' $d = \frac{1 + K\phi(n)}{e}$

$K = 0, 1, 2, 3 \ldots$

$d = \frac{1 + 2 \times 192}{35} = \frac{385}{35} = 11$

∴ $d = 11$

Hence private key = 11

Firewalls: (Network security)

→ monitors and controls incoming and outgoing traffic based on predefined rules.

→ acts like a barrier

→ Host based and Network based firewall (Types)

(software — eg in windows laptop; dedicated hardware — (eg large org.)

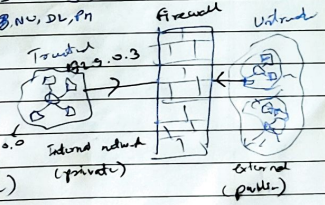
→ Packet filtering firewall: (layer 4)

→ Check IP header, TCP header

→ works on Network & Transport layer

→ can block IP, full network

→ can block a service (http, ftp, etc)

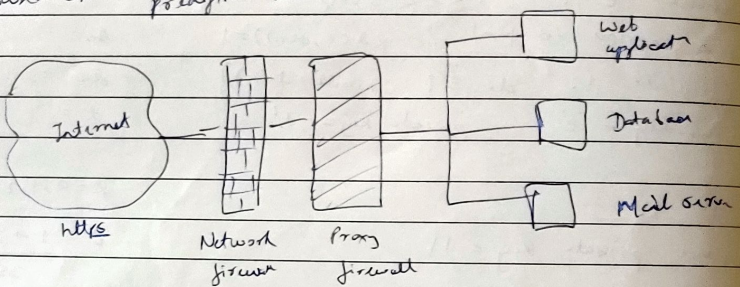


Rule	Source IP	Src Port	Dest IP	Dest Port	
1	172.2.4.80	any	any	any	→ block IP
2	192.32.0.0	any	any	any	→ block full network
3	any	80	any	any	→ block http service
4	any	any	any	21	→ block ftp "
5	any	any	192.9.0.3	any	

Application (Proxy) Firewall : (layer 5)

→ includes app layer as well (checks data as well along with headers)

→ monitors and control incoming and outgoing traffic based on predefined rules.



eg: gmail spam : deep scanning to check data like keywords, out of sequence packets

→ denial of service attacks, authorization, etc.

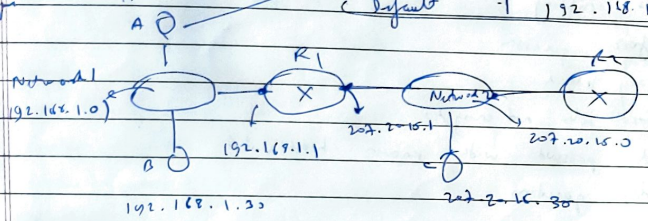
→ Static / dynamic IP — at a given point all IP unique

→ public / private IP → can be same (among diff. networks w.h diff public IP)

→ Routing table : (DHCP server)

Destination	Next hop
192.161.1.0/24	Direct
Default	192.168.1.1

always unique



→ DNS

→ Web server / P2P

Client —send request to server IP→ Server → stores information (eg webpages)

←send response to client's IP—

→ strong password : long, diff. characters, alteration
- → unique for important accounts & change after 6 month
- → can use same passwords for non-important accounts
- → store in paper / password management software / some other in intelligent manner

→ Malware : Virus, Worm, Trojan horse, Bots

→ ~~spread~~ spread via removable storage, visiting malicious links, downloading from untrusted sources, email-attachments

Sol → don't click / download from ~~malicious~~ unknown sources
- → firewall
- → antivirus
- → backing up data
- → installing software patches

Password threats :

→ social engineering
→ phishing
→ key logging
→ wireless sniffing
→ brute force guessing
→ dictionary attack
→ password files not encrypted
→ exposed password with known hash value
→ security question

Email threats

→ Eaves dropping
→ spamming & phishing
→ spoofing
→ malicious email attachments
→ CC and BCC issue

Sources of Malware :

→ removable media
→ documents & executables (.exe)
→ Internet downloads
→ network connections
→ email attachments
→ drive by downloads

Defense
→ backing up data
→ use firewall
→ install software patches
→ using antivirus

Web safety issues :

→ Handling files (always ask, always scan before save file)
→ accepting cookies (advertisements)
→ HTTPS (encrypted transfer of info. from server to client)
→ browsing history (use incognito, VPN)

classmate
Date
Page

Wireless security threats : (WIFI (not mobile networks))

→ sniffing (eg. getting info transferred b/w a user & coffee shop router)
→ rogue routers (router set up by attacker (malicious router))
→ evil twin routers (two routers with same name (connect auto))
→ unauthorized connections

Defense :

→ use strong router password
→ enable wireless signal encryption
→ enable router firewalls
→ disable SSID Broadcast (wifi name not visible in available wifi)
→ filter MAC addresses
→ turn off router

Web browsing :

→ check URL's domain name https://www.onlineweb.ittp.edu/img/...
(domain name: ittp.edu — from previous first slash)
→ Incognito
→ VPN
→ no business / private use at public places
→ point at URL to check actual URL