Computer Networks

Classfull IP address Classification:

Class A: (0 ___)
* NID:8, HID:24
* $2^7 - 2$ N/w
* $2^{24} - 2$ Hosts
(0+-126)

Class B (10 —)
NID:16; HID:16
* 2^{14} N/ws
* $2^{16} - 2$ hosts
(128-191)

Class C (110 —)
NID: 24; HID: 8
* 2^{21} N/ws
* $2^8 - 2$ Hosts
(192-223)

Class D (1110 —)
No NID & HID
* used for multicasting
2^{28} addresses
(224 - ~~240~~ 239)

Class E (1111 —)
No NID & HID
* Reserved
* 2^{28} address
(240-255)

→ NID 0 & 127 of class A are not used.
→ For every N/w starting add is n/w id

→ LBA: 255.255.255.255
→ DBA: NID.(All 1's)

Subnetting

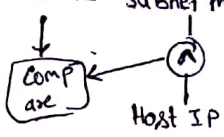
* Divide N/w by borrowing sufficient no of bits from host id part.
* Think abt confusion in interpreting certain IP add by insider N/w and outside N/w
* no of hosts = no of IP add − (2 * no of subnets)

Subnet mask: Nid + subnet id → All 1's
Hid → 0's

Subnet Mask ∧ Host IP → subnet ID

Routing table structure

(Subnet id | Subnet mask | Interface)



→ If more than 1 match then subnet id with more 1's.

→ Fixed Length Subnetmasking
Variable Length Subnetmasking (VLSM)

→ Larger the subnet, smaller the ~~size~~ value of subnet mask.

CIDR (Classless Inter Domain Routing)

* IP add representation: a.b.c.d/n
n is no of bits in NID
* CIDR block forming rules:
→ IP add are contiguous
→ Block size in power of 2.
→ Start IP add / block size.

* Subnetting in CIDR is also same.

Supernetting / Aggregation

* Rules: → Contiguous blocks
→ equal sizes can be combined at once
→ Start IP / total size

→ Using subnet mask a host determines whether a given IP add is in the same N/w or not (Think how).
If yes, then pckt will be sent directly otherwise through router.

FLOW CONTROL

Delays:

Transmission Delay: $\frac{L}{B}$; Propagation Delay = $\frac{d}{v}$

→ Queuing delay; processing delay.

Stop & wait:

→ pckt is sent and sender waits until ack is received

$$\eta = \frac{T_t}{T_t + 2T_p} = \frac{1}{1+2a}$$

(assuming queing, processing delay, ack transmission time are negligible)

$a \rightarrow \frac{T_p}{T_t}$

distance ↑ ⇒ η ↓

size of pckt ↑ ⇒ η ↑

no of pckts transmitted = $1 + np + np^2 + \dots$ → same for SR also

p → probability of error.

Sliding Window Protocol / Pipelining:

$\eta = \frac{1}{1+2a}$ ⇒ Min window size for 100% efficiency $= \lceil 1+2a \rceil = \lceil \frac{1}{\eta} \rceil$

no of bits of seq num = $\lceil \log_2 (1+2a) \rceil$

→ But this not posible practically.

→ GBN, SR are implementations of sliding window.

Go Back N ($N \geq 1$)

Sender window size = N ; Receiver window size = 1;

→ out of order pckts not accepted. So for retransmission we need to go back N and retransmit

$$\eta = \frac{N}{1+2a}$$

→ GBN uses cumulative ack.

→ Buff = N+1; Seq num = N+1; Bandwidth req = high

CPU req = low; Implementation difficulty = easier than SR.

Selective Repeat:

Sender window size = Receiver window size = N

→ out of order pckts accepted.

$$\eta = \frac{N}{1+2a}$$

→ uses independent ack.

→ Buff = 2N; seq = 2N; Bandwidth req = moderate

CPU req = high; Implementation is complex.

Note:

→ seq num ≥ sender win + receiver win

→ wi sender window = $\min(1+2a, 2^n)$ → n = ack bits in seq. num. → This is for sliding window (not for GBN, SR)

→ Ack
- Cumulative: ack timer is started (when a pckt is received) and ack is sent for group
- Independent: 1 per pckt. SR sends negative ack and allows early retransmission.

Problem	Soln
Data pckt lost	Timeout timer
Duplicate pckt problem (ack lost)	Seq number
Delayed Ack (Missing pckt problem)	Seq num for Ack

Capacity of channel: Max no of bits present on the channel.

∴ capacity of channel = BW * T_p

High capacity channel is called thick pipe and low capacity channel is called thin pipe.

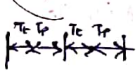
Access Control Methods

If T_p is given in bits then divide with bandwidth " " " " meters " " " velocity.

→ when channel is broadcast channel we need access control methods.

Time Division Multiplexing:

$$\eta = \frac{T_t}{T_t + T_p}$$



If a station doesn't transmit, then the slot is wasted.

Polling:

$$\eta = \frac{T_t}{T_{poll} + T_t + T_p}$$

Stations that are interested participate in polling.

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

→ Every host, before sending, sense carrier. Carrier sensing is done as long as data is transmitted

⇒ $T_t \geq 2T_p$ ⇒ $L \geq 2T_p * BW$

→ No acks are used.

If no sufficient data, extra bits are padded.

$$\eta = \frac{T_t}{C*(2T_p) + T_t + T_p} = \frac{T_t}{T_t + (2C+1)T_p} = \frac{1}{1+6.44a}$$

Probability of successfull transmission is $= {}^nC_1\, p(1-p)^{n-1}$

(p is probability to transmit)

This maximum when $p = 1/n$

⇒ Max value = $(1-1/n)^{n-1} = e$ (for large values)

Backoff Algo / Binary Exponential back off algorithm:

→ Every pckt has a collision num initialized to 0.

→ If collision occurs, then collision num is incremented.

Now system choose a num blw $[0, 2^{n-1}]$ when n is collision num. Now system waits for this amount of time and retransmits.

→ As collision num increases, Probability of collision ↓ exponentially.

→ Drawback: Capturing Effect.

Token Passing:

→ Stations are connected in ring topology
→ Every system transmits whenever it gets a token.

Ring latency = $\frac{d}{V} + N*t$

$t \rightarrow$ delay at every station

Efficiency, $\eta = \frac{N * T_t}{T_p + N * THT}$

Delayed token reinsertion	Early token reinsertion
$THT = T_t + \text{ring latency}$	$THT = T_t$
$= T_t + T_p + 0$ (assuming t=0)	
$= T_t + T_p$	
$\eta = \frac{N * T_t}{T_p + N(T_t + T_p)}$	$\eta = \frac{N * T_t}{T_p + T_t}$

ALOHA:

→ any station can transmit any time. So collision is possible.
→ Acks are used; No collision detection.

Pure aloha	Slotted Aloha
• vulnerable time = $2T_t$	• vulnerable time = T_t
$\eta = G * e^{-2G}$	$\eta = G * e^{-G}$
G → no of stations who wants to transmit in T_t	
$\eta_{max} = \frac{1}{2e} = 0.184$ (at $G = 1/2$) i.e., 18.4%	$\eta_{max} = \frac{1}{e} = 0.368$ (at $G = 1$) i.e., 36.8%

ERROR CONTROL METHODS

Error Detection: Detect and req for retransmission
- → Send data twice (D+D)
- → Parity check
- → CRC
- → Checksum

CRC: Both sender and receiver will have CRC generator.
→ An n-bit CRC generator is represented using a polynomial of degree n-1.

For data we append 'n-1' bits and perform XOR division
→ The obtained remainder is called CRC and it appended to original data.
→ Now receiver gets remainder 0 is data is received correctly.

Checksum:

→ To compute n-bit checksum, divide data into groups of n-bits each.
→ Now add all these. (If carray occurs add to the LSB)
→ If checksum field is present in data, consider it a
→ Now negate the result and store it in checksum.
→ Now receiver must get 0 when added all these.

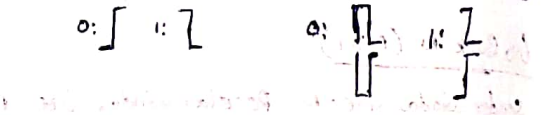
Note:
Meaningful errors are possible.

Error Correction:
↳ Hamming Code (Refer Notes)

ISO-OSI Layers

→ A host need not contain Nlw layer but TL is present on every host and NL is present on every router.

Physical layer:
→ signal conversion blw different transmission media.
→ transmission modes: simplex, half duplex, full duplex.
→ Topologies: bus, star, ring, mesh, hybrid.
→ Encoding: Manchester, Differential Manchester Encoding



→ It provides bit synchronization, bit rate control.
baud rate = 2x bit rate

Datalink layer:
- → Logical link Control: Flow control
- → Medium Access Control: Framing, Access Control, Physical addressing

Error Control (CRC)

Transport layer:
→ End to End connection: Service point addressing.
→ Flow control
→ Error control
→ segmentation & reassembly
→ Multiplexing & Demultiplexing
→ Congestion control

Network Layer:
→ Host to Host connectivity
→ Logical addressing.
→ Switching
→ Routing
→ Congestion control
→ Fragmentation.

Session Layer

→ Authentication & Authorization
→ check pointing.
→ Synchronization.
→ Dialog control, logical grouping
→ Connection establishment, maintainence, termination.

Presentation Layer:

→ character translation
→ Encryption & Decryption.
→ Compression.

this all layers work together (Pg: 70)

Framing:

→ SFD is used at the beginning of every frame
  ↳ (10)*11
→ we also need to detect end of frames

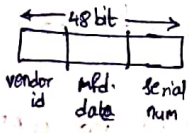
Framing
- fixed length (End is implicit)
- variable length
  - using length field
  - End delimiter

End delimiter may match with data pattern.

Character stuffing	bit stuffing
• Add null (16) as prefix to every null and ED appeared in string	• If ED is 01111, then add '0' after every 0111.

Physical Addressing:

Logical add: unique globally.
physical add: unique with N/w.
→ MAC is unique globally and used as physical address.



Unicast MAC: LSB bit of 1st byte is 0
Multicast MAC: LSB bit of 1st byte is 1
Broadcast MAC: all bits are 1's.

LAN Technologies

Ethernet (IEEE 802.3)

• Bus topology; CSMA/CD; No acks; manchester Encoding
• Bw: 10Mbps, 100Mbps (fast ethernet), 1Gbps (gigabit ethernet)

Ethernet Frame Format:

6	6	2		4 (bytes)
SA	DA	length	Data	CRC

→ PL adds 1 byte SFD (10101011) and 7 bytes preamble 10101010...10
→ Preamble & SFD provide synchronization.
→ Since CSMA/CD is used $T_t \geq 2T_p$
  ⇒ $L \geq 2T_p * Bw$. or $L \geq 64B$
→ To avoid monopolization, max data = 1500 B

	Min	Max
Data	46	1500
Frame	~~1500~~ 64	1518

Disadv:

→ not suitable for real time applications
→ not applicable for interactive application
  ∵ min data size = 46B
→ not good for client server applications (∵ no priorities)

Token Ring (IEEE 802.5)

• Ring topology; token passing; piggy backed acks; Differential manchester.

Sender side problems

- orphan pckt problem: sender goes down
- stray pckt problem: pckts get corrupted and cannot be identified by sender.

Soln: Monitor & Monitor bit (Think)

Destination side Problems:

	A	C	
Destination Down	0	0	A - available
Destination Busy	1	0 & E=0	C - copied
Packet corrupted	1	0 & E=1	E - Error
Copied	1	1	
	0	1 → invalid.	

→ while retransmitting sender has to reset monitor bit

Token Problems:

(i) Captured token: a station may hold token for indefinitely long.

soln: impose restriction on max THT

ETR ⇒ T_t = THT

DTR ⇒ T_t = THT - RL

(ii) Token Lost: Monitor regenerates token after waiting for maximum token return time.

max token return time = ring latency + N * THT

(iii) Token corrupted: monitor identifies this and regenerates a new token after waiting for max token return time.

Monitor Problems:

(i) Monitor goes down: monitor sends AMP frames as long as it is active. Every station expects AMP frames in regular intervals.

→ If monitor goes now, new monitor is choosen using polling

(ii) Monitor Hacked

Token ring frame format:

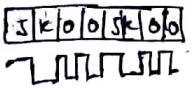
Data frame:

1	1	1	6	6	4	*	1
SD	AC	FC	DA	SA	data	CRC	ED	FS

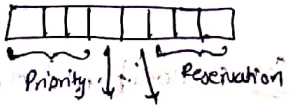
Token frame: (3 bytes)

SD	AC	ED

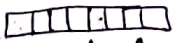
SD:



Access Control:

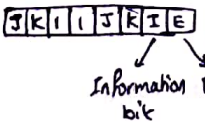


Frame Control:

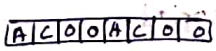


0 0 - data frame (all 8 bits are 0's)
1 1 - control frame (Eg: AMP)

End Delimiter:



Frame status:



Minimum length of token ring:

If all the systems are down, then there is a chance that sent token comes back even before it is fully transmitted.

To prevent this

propagation delay ≥ transmission delay of token

(or)

Capacity ≥ size of token.

Switching

Circuit Switching

- For every connection, reservations are made & path is set. Time required for this is called setup time.
- No header is req as path has already been setup
- Data is always sent in order.
- Circuit switching is applied at physical layer.
- Circuit Switching is implemented in 2 ways:

Freq. Division Multiplexing

- freq is divided among the connections

Time Division Multiplexing

- time is divided into slots and each connection is given a slot.

Packet Switching:

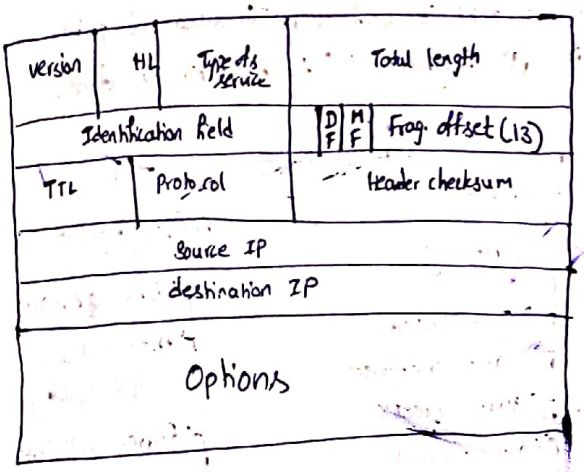
→ No reservation; store & forward is used;
→ packetization is done.
→ Queing delay is present and pckt may be discarded if there is no space in the queue.
→ No setup time is required.
⇒ 2 types of packet switching:

virtual circuit	Datagram
• Connection oriented	• Connection less
• pkts follow same path	• Diff path.
• reservations are made by 1st pckt	• No reservations
• pkts arrive in order	• out of order.
• 1st pckt has global header	• all need global header
• reliable, costly	• not reliable, cheap

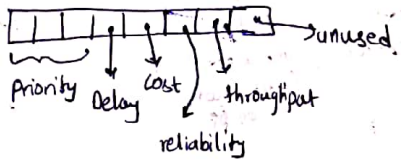
Note:

- Circuit switching may result in wastage of resources if the connection is idle.
- Circuit switching is costlier than pckt switching
- Circuit switching is more suitable for real time applications

Internet Protocol



- Header length = 4 * (HL field)
- TOS:



- Total length: length including header.
- Identification field: numbering to datagrams helps in ordering received datagrams
- Protocol: to used to discard less imp pckts when there is space.

  ICMP < IGMP < UDP < TCP
- Header checker: divide into 2 byte slots, add and negate.

Options:

(i) Record route: each router put its address

(ii) Source routing < Strict source routing / Loose source routing

(iii) Padding: to make size a multiple of 4

(iv) Timestamp: Every router in path adds in-time and out-time.

Fragmentation:

→ Max Transmitable Unit: Max size that can be transmitted (including IP header)

→ Segmentation is done at TL such that frag. will not be need at host. So fragmentation is done only at routers.

→ After fragmentation pckts are routed independently.

→ Max payload at TL = MTU − IP header − TL header

→ original fragment offset = 8 * (fragment offset field)

So payload size at IP must be a multiple of 8 in all the pckts except the last.

→ Reassembling is done only at destination.

Reassembly algo:

- Starting from 1st fragment calculate fragment offset of next pckt and identify next pckt using this calculated value. Continue this until last fragment is met.

MF	offset	
1	0	→ 1st fragment
1	≠0	→ intermediate fragment
0	≠0	→ last fragment
0	0	→ no fragmentation

$$\text{offset of next fragment} = \left(\text{offset of current fragment} + \frac{TL - HL*4}{8}\right)$$

Other Concepts at Network Layer:

Implementation of Broadcasting

Limited Broadcasting:

S.IP = source IP, S.MAC = source MAC → D.IP = All 1's, D.MAC = All 1's

Directed Broadcasting:

S.IP = Source IP, S.MAC = source MAC → D.IP = NID + (All 1's), D.MAC = MAC of gateway router.

- once this reaches gateway router, then it replace DIP with all 1's and D.MAC with all 1's.

Address Resolution Protocol (used to find MAC given IP)

Receiver on N/w:

ARP req pckt is broadcasted with

S.IP = source IP, S.MAC = MAC of sender → D.IP = dest. IP, D.MAC = All 1's

Reciever on diff Nlw:

→ In this case sender can never know the MAC of receiver.

→ Here sender just finds ∃ MAC of default router and forwards pckt to the router.

Note: ARP req is broadcast message
ARP reply is unicast message.

Special Address 127 (Loopback address)

→ If we need to check if NIC and other layers of system we used this address.

→ we can use every address except 127.0.0.0 and 127.255.255.255

→ This pckt will go through all layers except physical layer and comes back to app. layer.

→ This is also used to test working of ~~client~~ server by using different ports.

RARP (Reverse ARP):

→ used to know IP, given its MAC.

→ RARP server is maintained at every Nlw. It contains mapping blw IP add & MAC.

→ RARP req:

S.IP = All 0's D.IP = 255.255.255.255
S.MAC = source MAC D.MAC = All 1's

→ RARP server sends reply.

Disadv: → MAPPING is static.
→ Data is not centralized.

BootP (Bootstrap protocol)

→ same as RARP except that it may not have bootp server at every Nlw. ~~Relay~~
In this case relay agents are used.

- Adv: Data is centralized
  Disadv: Mapping is static.

DHCP:

→ Here we maintain 2 mappings < static / dynamic

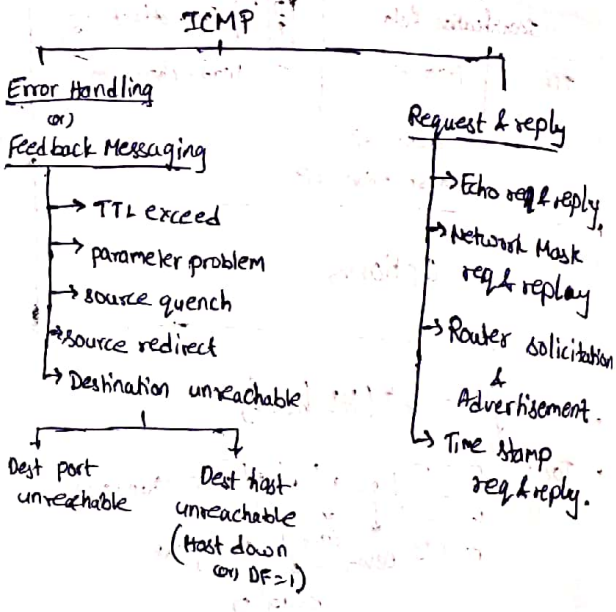
→ IP addresses are dynamically assigned from a pool of address and lease time is given after which renewal request has to be done.

Adv: • Data is centralized
• Mapping is dynamic

Note: to make DHCP compatible with bootp, port of BootP = port of DHCP.

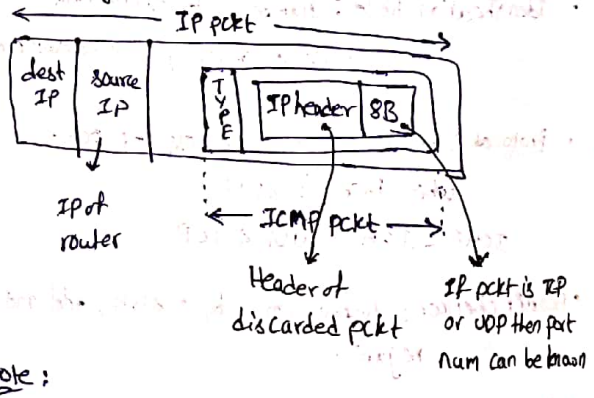
ICMP (Internet Control Message Protocol)

- ICMP is sent only if the discarded pckt is TCP or UDP.
- ICMP pckt is sent only if discarded pckt is 1st fragment (if fragmentation is done.



- ICMP pckt never meets app. layer and TL.

ICMP message format:



Note:
Before any communication starts, host must do below steps:
(i) Getting IP address (RARP, BootP, DHCP)
(ii) Knowing default router (router adv./solicitation)
(iii) Knowing Nlw mask (Nlw mask req & reply)

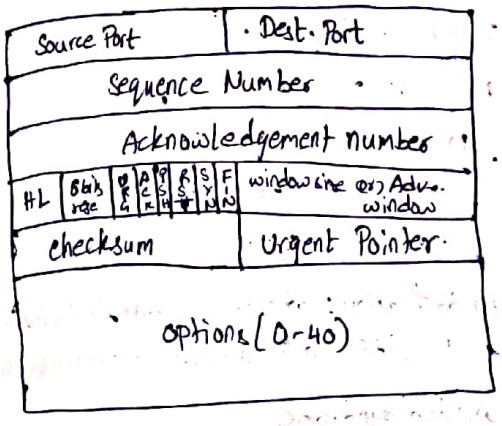
Applications of ICMP:

(i) Trace route
(ii) PMTUD (Path MTU Discovery)

Refer nodes for Routing

TCP

- TCP ensures end to end communication



Port number:

0-1023: well known

1024-49151: reserved

49152-65,535: public usage.

Seq number: Contains number given to 1st byte in the segment.

- TCP is a byte stream protocol.

Ack number: contains byte num of next expected byte.

no of bytes in segment (n) = $TL_{IP} - HL_{IP} - HL_{TCP}$

⇒ Ack num = n+1

Note:

- seq num doesn't always start with 0. Instead we start with some random number (think why)

wrap around time: It is time taken to usecup all seq number.

life time: It is max amount of time for which pckt can be in network.

- to avoid conflicts

wrap around time > life time.

- For given ~~no of bits~~ bandwidth, if no of bits in seq num field are not sufficient to satisfy above condition, then we take extra bits from options field. this option is called timestamp.

Header length: scaled down by 4

Flags:

PSH: pushes data as soon as it is received without buffering. Useful for interactive applications

~~URG~~

RST: Used when there is something wrong in connection.

Ex: receiver receives unexpected seq num.

URG: when this flag is set, the data is sent faster and it will reach dest even before the pckts that are sent earlier.

For this purpose we set priority to 7 in type of service field. (∵ routers can't look into TCP header).

Urg pointer: Used when only some part of data is urgent.

last byte of urgent data = seq num + urgent pointer.

SYN: used to synchronize sequence numbers.

Ack: If this is set then only the ack number field is considered.

If pure ack is to be sent then, only header will be sent.

FIN: used when we need to terminate the connection

Note: Refer connection establishment, ~~data~~ connection termination in notes. Also refer state transition diagram.

- TCP is full duplex connection.
- At the time of connection establishment information like initial seq num, MSS, window size are exchanged.
  - MSS: payload in TL
- Window sizes are set based on MSS shared.

Note:

- Pure ack doesn't use seq num.
- FIN pckt uses a seq num
- If no data is received, still we send ack for old data

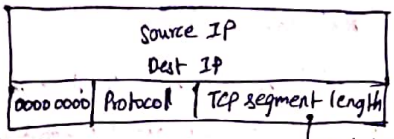
SYN	ACK	
1	0	→ 1st pckt
1	1	→ 2nd pckt
0	1	→ Ack present in header
0	0	→ Invalid

Window size / Adv window:

- used for flow control.
- receiver sends available window size.
- If window size received by sender 0 ; then sender starts persistant timer and waits until receiver notifies or the timer finishes. If timer finishes sender sends 1 byte of data. If sender gets reply then communication is continue otherwise persistant time is started again.
- window size choosen by sender = min{receiver window, congestion window}
- window size field has only 16 bits, if more than that ~~is~~ 2^{16} is available then window ~~extension~~ size extension option is used.

Checksum:

- checksum is computed on TCP header & pseudo IP header



→ total length − HL_{IP} − HL_{IP}

Options:

- timestamp
- window size extension
- Parameter Negotiation Eg: MSS
- padding

Retransmissions in TCP:

- In TCP, if ack num 'x' is received then sender assume all the data before x is correctly received.
- Retransmission is done in 2 cases
  - (i) receiving 3 dup acks (excluding original ack) within TO timer. ---- low congestion. This is known as early retransmission
  - (ii) Time out timer. ---- high congestion.

Congestion Control:

sender window = $\min\{w_c, w_R\}$

w_c - congestion window w_R - Receiver window.

working:

- slow start phase until threshold is reached. Here congestion window size is doubled.
- Congestion avoidance phase after threshold. Here congestion window is incremented by 1.
  initial threshod ← $w_R/2$.
- Congestion detected due to TO times ⇒ threshold = $\lceil w_c/2 \rceil$ & enter slow start phase.
- Congestion detected due to 3 dup acks ⇒ threshold ← $\lceil w_c/2 \rceil$ & enter congestion avoidance phase.

TCP time Management:

(i) time wait timer: (think of its purpose)

time wait timer = 2 * life time

(ii) keep alive timer.

(iii) persistent timer

(iv) Ack time

(v) Time out timer:

- static to TO timer leads to problems.

Dynamic TO timer is given by below algorithms

(i) Basic algorithm:

$$NRTT = \alpha(IRTT) + (1-\alpha)ARTT$$

$$TO = 2 * RTT$$

(ii) Jackobson's Algorithm:

$$NRTT = \alpha(IRTT) + (1-\alpha)(ARTT)$$

$$ND = \alpha(ID) + (1-\alpha)(AD)$$

where $AD = |IRTT - ARTT|$

$$TO = RTT + (4 * D)$$

Karn's modification: whenever retransmission is req set TO to double of previous TO.

Silly window Syndrome:

- occurs in 3 cases:

(i) Receiver advertises 0 window size. But the problem is tempory. If this continues for long time RST flag is used.

(ii) occurs when sender transmits less amount of data.

soln: Nagle's algo:

- transmits once for each RTT or transmit in advance if data sufficient for 1 MSS is available.

(iii) receiver accepts only 1 byte.

Clarck's soln: don't advertise until half of window is available (or) 1 MSS is available.

Traffic Shaping:

- Another congestion control that controls rate at which pckts enter nlw.
- During connection establishment sender and receiver negotiates traffic pattern.

(i) Leaky Bucket:

- Data comes into Q at any rate but comes out at const. rate.
- If Q is full data is discarded. (Think disadv)

(ii) Token Bucket:

- Each pckts gets a token before it goes on the nlw.
- o/p rate = token filling rate.
- If capacity is full then no more token will be added.
- If capacity is c and token filling rate is r then max no of tokens that can be sent in time 't' is

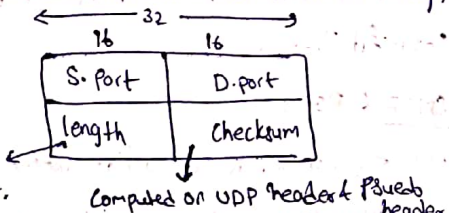
$$\frac{c + rt}{t}$$

- In leaky bucket pckts are discarded when bucket is full but here tokens are discarded.

UDP (Null Protocol)

- Used in application that need single req & reply.
  Eg: DNS, BootP, DHCP, Dist. vector routing etc.
- broadcasting & multicasting application.
- used in applications that req speed over reliability
  Eg: multimedia, gaming etc.
- If AL needs to set any priorities then UDP has to communicate this to NL. TCP should also do this.
- ICMP pckts received are informed to AL through UDP.

UDP Header:



- Since UDP is not used reliable, checksum is not used in general. In that case it is set to 0.
  Checksum is stored in 1's comp form
  000...00 ⇒ checksum not used
  111...11 ⇒ checksum used.

Hardware in Networking:

- 2 types of channels < baseband (no multiplexing)
  broadband (multiplexing)
- 10 base T (10MBPS, no multiplexing, 100m)
  10 base 2 (" " 200m)
  10 base 5 (" " 500m)

Attenuation is possible here.

Repeater:

- used to increase length of LAN by connecting 2 lan segments of same type.
- Repeater takes dead signal & regenerates it.
- Runs at PL and collision domain remains same.

Hub:

- Hub is a multiport repeater. An n-port hub connects n stations.
- A pckt to be sent is transmitted to every station. ∴ traffic is high.
- Runs at PL and collision domain remains same.

Adv: cheap

Bridge:

- Used to connects 2 lan segments. A lan segment can even be different types but MTU must be same since bridge cannot do fragmentation
- operates at PL & DLL (can see MAC)
- Collision domain is reduced.
- Every bridge has forwarding table. Based on how this table is built we have 2 types of bridges (i) static (ii) Dynamic.
- Dynamic bridge never knows MAC of a station until it transmits.
- Bridge is capable of filtering, forwarding, store & forward, flooding.

Problems:

- If connection forms cycle then pckts may go into infinite loop.

Spanning tree algo:

- Choose bridge with least id as root bridge.
- For each bridge, choose root port.
- For every LAN, choose designated bridge and make corresponding port as designated port.
- Mark root port and designate ports as forwarding ports and block remaining.
- Bridge Data unit Protocol (BDUP) is implemented to sort out this tree.

Switch:

- A n-port switch can connect n station.
- operates at PL & DLL.
- Switch ~~is designe~~ allows more than one communication at a time.
  So collision within switch = 0.
  traffic is less.
- Collision domain is reduced.

Disadv: costly.

Router:

- Connects 2 networks of same type.
- operates at PL, DLL, NL.
- filters broadcasted pckts.
- collision domain & broadcasted domain are reduced.

- Every interface of router has an IP address. The address is taken from the N/w the interface is connected to. However if 2 routers are connected then we need a ~~CIDR~~ CIDR block of size 4.

GATEWAYS:

- capable of connecting N/ws of 2 diff types.
- operates at: PL, DLL, NL, TL, AL
- ~~appli~~ reduces collision domain & broadcasting domain

Uses:

- protocol convertor
- proxy.
- NAT server.
- Firewall
- Deep packet inspection.
- Buffers management (think of its use)

Application Layer Protocols

DNS (Domain Name Service):

- port: 53
- given a domain, it gives IP address.
- DNS also does load balancing
- Data of DNS is distributed.
- There are 13 root DNS servers.
- Local DNS server contacts root DNS server in 2 ways (i) iterative (ii) Recursive.
- DNS uses UDP.

HTTP (Hyper Text Transfer Protocol)

- port: 80
- used to get web page.
- Uses TCP for reliability. No any inbuilt mechanism for reliability.
- Inband protocol
- stateless protocol. It rather uses cookies which are stored at the user end.

HTTP has no any inbuilt security mechanism

- HTTP 1.0 use non-persistent connection
  no of connections = no of objects + 1
- HTTP 1.1 uses persistent connection.
  Since connection is held for long time, congestion window grows and high B/w is available.
- Suitable for when objects are small and many in number.

Methods: Head, Get, Post, put, delete, trace, options, connect (think of their functionalities)

FTP (File Transfer Protocol)

- port: 21 (commands), 20 (data); Uses TCP
- outband protocol
- Control connection (port 21) is persistent
- Data connection (port 20) is non-persistent.
- stateful Protocol

SMTP & POP:

- FTP req stations to be online where as SMTP doesn't.
- Mail Client (MC) pushes mail into Mail Transfer Agent (MTA) using SMTP.
- Mail client (MC) at the other ends takes mail from MTA using POP.
- SMTP & POP uses TCP
- To send data other than non-text we need to convert it text format and send it. This will be converted back to non-text at receivers end.
- inband protocol.