Unit - III

Transport layer, TCP & UDP

→ **Responsibilities of transport layer:**

→ End to End delivery (port to port) or (process to process)

→ It follows two main protocols TCP & UDP

→ Error & flow control is managed

→ Multiplexing & demultiplexing. the data/ from source to destination request

→ **Main services of transport layer:**

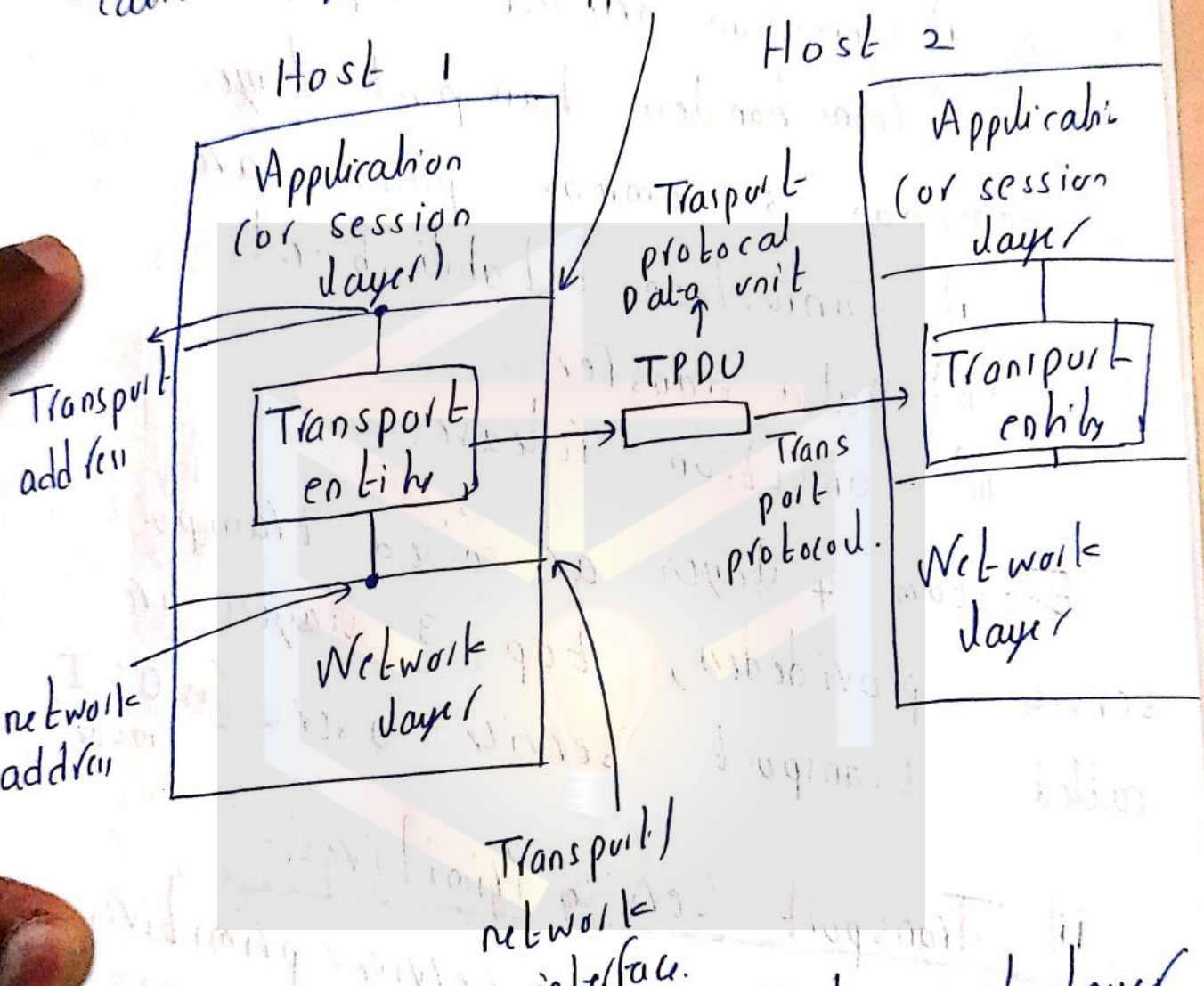
i) Service provided to the Upper layer:

→ The upper layers of transparent layer are session, presentation & application layer.

→ The ultimate goal of the transport layer is to provide efficient,

Education4Fun.com

reliable & cost effective services to its users, normally processes in the application layer.



→ component / s/w / h/w in transport layer are called Transport layer.

Education4Fun.com

There are two types of transport service similar to network services

i) Connection oriented Transport layer
ii) Connectionless transport layer.

connection 3 common phases are

i) Connection establishment
ii) Data transfer
iii) Connection release

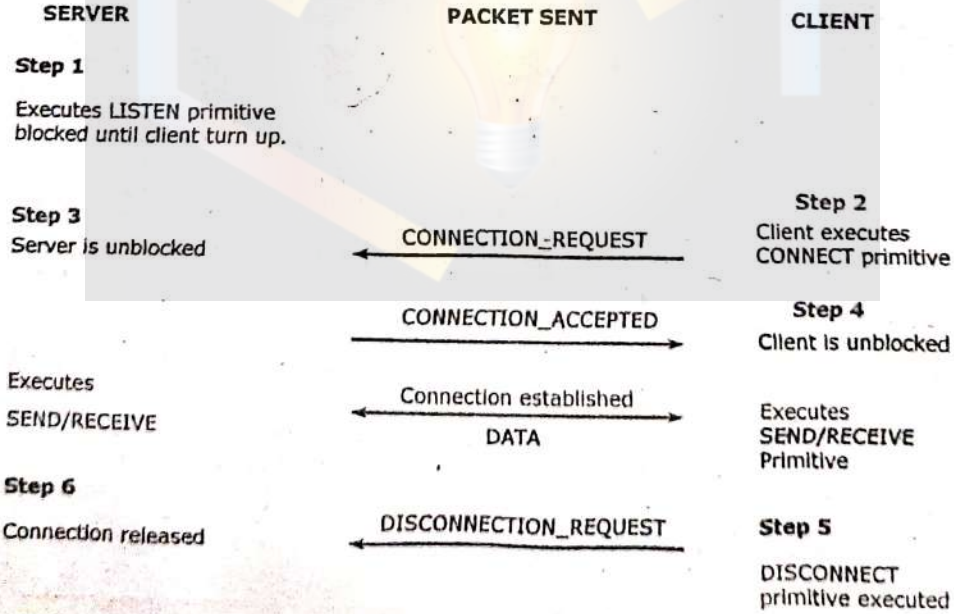
Bottom 4 layers are called transport service providers, top 3 layers are called transport service users (in OSI mode).

ii) Transport Service Primitives:-

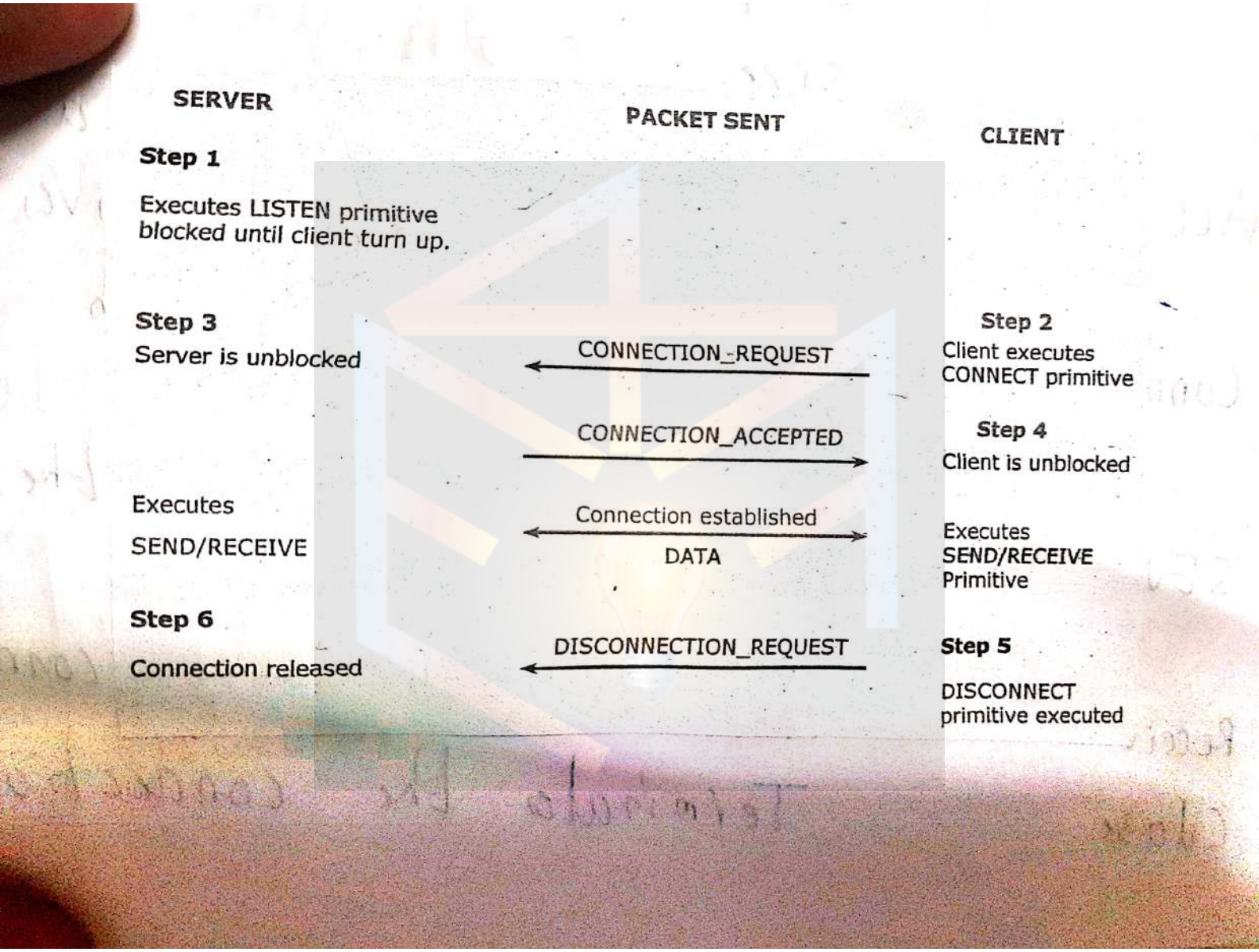
→ Transport service primitives are used by the user to access the transport service.

Education4Fun.com

Primitive	Packet Sent	Meaning
LISTEN	None	Block until some process tries to connect
CONNECT	Connection-request	Attempt to establish connection.
SEND	DATA	Send information
RECEIVE	None	Block until a data packet arrives
DISCONNECT	Disconnection-request	Release the connection



Education4Fun.com



Education4Fun.com

iii) Berkeley Sockets:-

Socket is used for establishing a TCP connection in the network

Few primitives

Primitives	Meaning
SOCKET	Creates a new connection
BIND	Assigns local address to new socket.
LISTEN	Announce willingness to accept connections, give queue size.
ACCEPT	Block the caller until a connection attempt arrives
Connect	Attempts to establish connection
SEND	Transmit data over the connection
Receive	receives data from the connection
Close	Terminates the connection

Education4Fun.com

→ Elements of transport protocols:

→ Addressing:

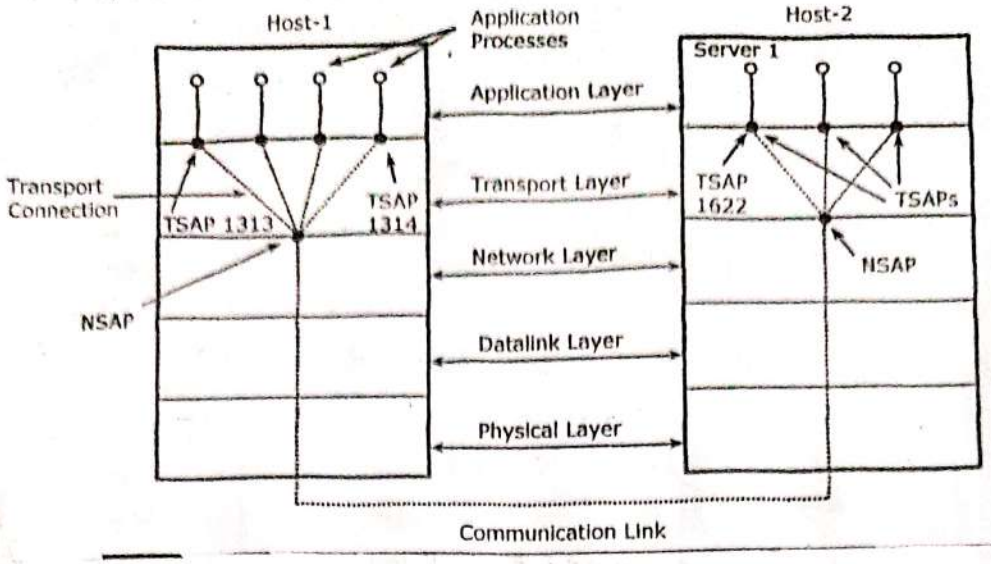
→ When an application process whishes to setup a connection to a remote application process, it must specify which one to connect to (commonly we call as address)

→ In network layer we use to give IP address to connect to a network that address is called NSAP (Network Service Access Point).

→ Hence similary here in transport layer we have port no. (that a specific application uses. it is called as TSAP Transport Service Access Point.

→ Based on that the connection is established in process.

Education4Fun.com



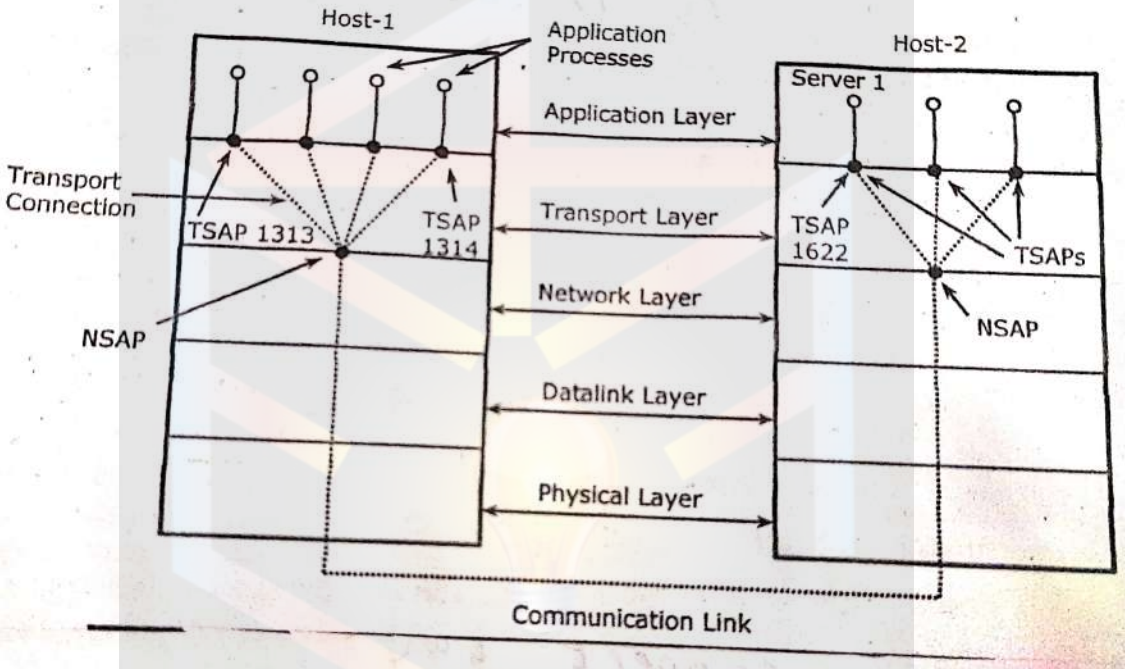
→ Connection Establishment :-

→ When we think of connection establishment it looks very simple & straight forward. But the problem occurs when the network lose or store duplicate packets.

→ When their is heavy congestion on the subnet, etc...

→ To solve these problems we introduced the three-way handshake.

Education4Fun.com

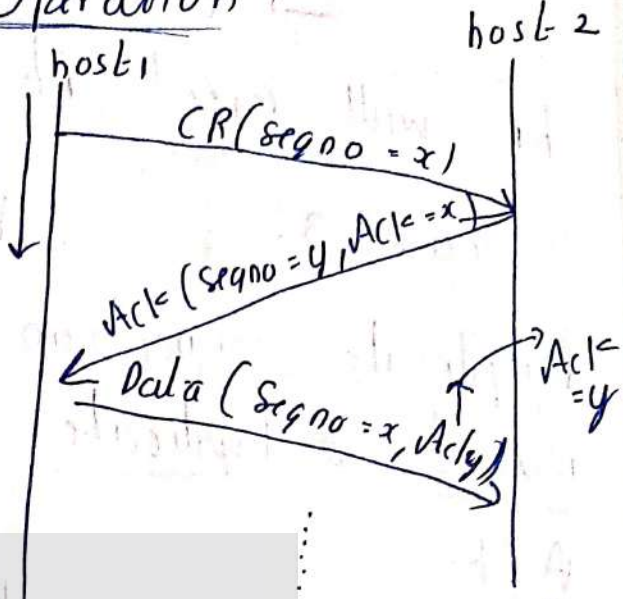


→ Connection Establishment :-

→ When we think of connecti

Education4Fun.com

case i) :- Normal Operation



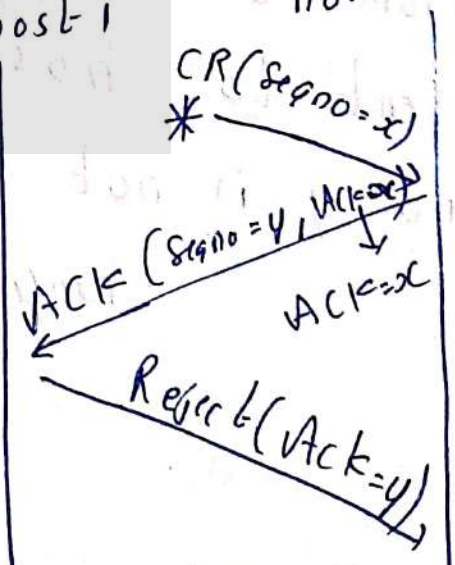
→ Host 1 choose a sequence x & send a request connection request to Host-2

→ Host 2 receives & replies with ACK x & assiging its own initial sequence number to y

CR - connection request
ACK - Acknowledgement

→ Finally host 1 ack host 2's choice of an initial sequence number & send data.

case ii) :- Delayed duplicate control :-



→ assume due to some delay host 1 request went to date from old connection.

→ but host 2 doesn't know that this hence he give (ACK (seqno = y, ack = x))

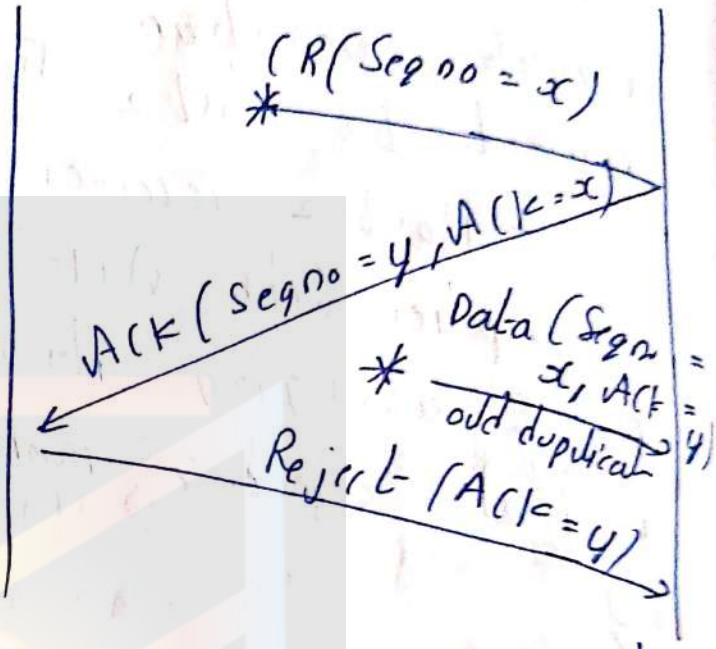
Education4Fun.com

→ host 1 got ack hence hi will reject it.

case 3:

Duplicate connection - request & Duplicate ACK

→ host 2 received a delayed connection - request & replies ACK to it.



→ in this point, second delayed duplicate from old connection with ACK (x, z) sent to host 2. But the sequence no. y is not ack from host 1.

hence both are rejected.

Education4Fun.com

→ Connection Release :-

→ Connection release is easier than establishing the connection.

i) Asymmetric Release :- In asymmetric release, either side can release connection & there may be chances of loosing the data (like a phone call)

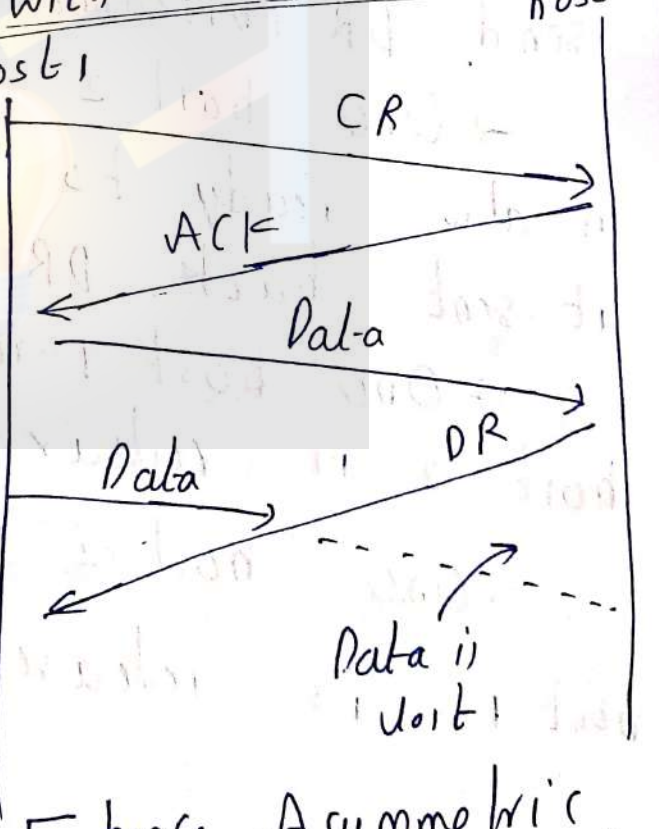
ii) Symmetric Release :- both sides they should release separately

Abrup disconnection with loss of data :-

→ once host 1 & host 2 established their connection.

→ assume host 2 disconnected request.

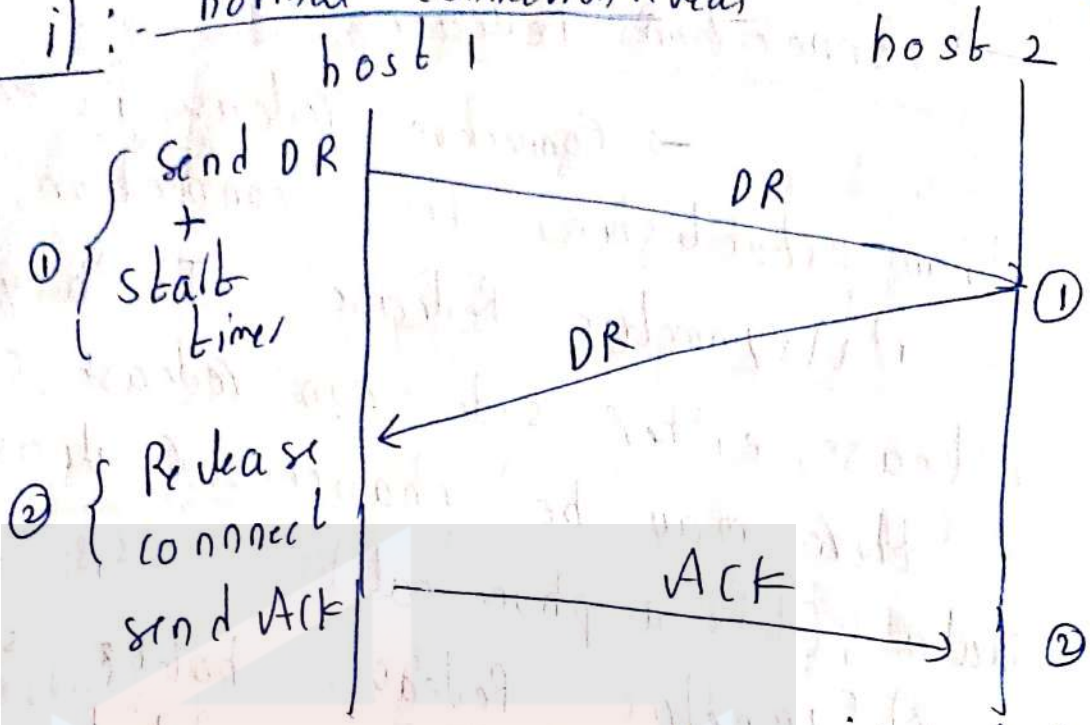
→ hence host 2 is not ready to receive & host 1 dat ii lost in middle



[hence Asymmetric release n't a good way

Education4Fun.com

Case i): normal connection release



→ Once the data transfer done host 1 send DR (Disconnect requst) to host 2

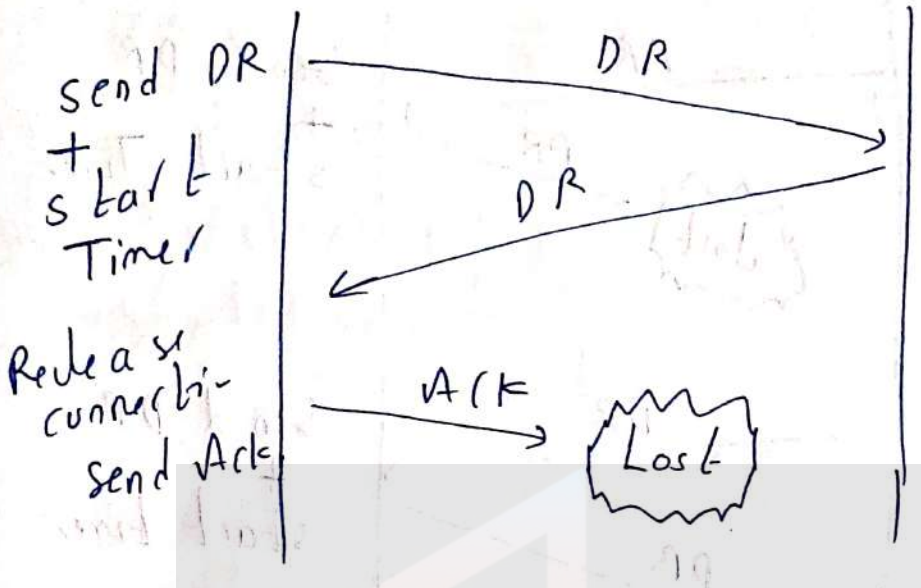
→ Once host 2 received it host 2 is also ready to Disconnect hence it sent back DR

→ Once host 1 received DR from host 2 is released & sent ACK

→ Once host 2 received ACK from host 1 is released

Education4Fun.com

case 2 :- Final ACK is lost



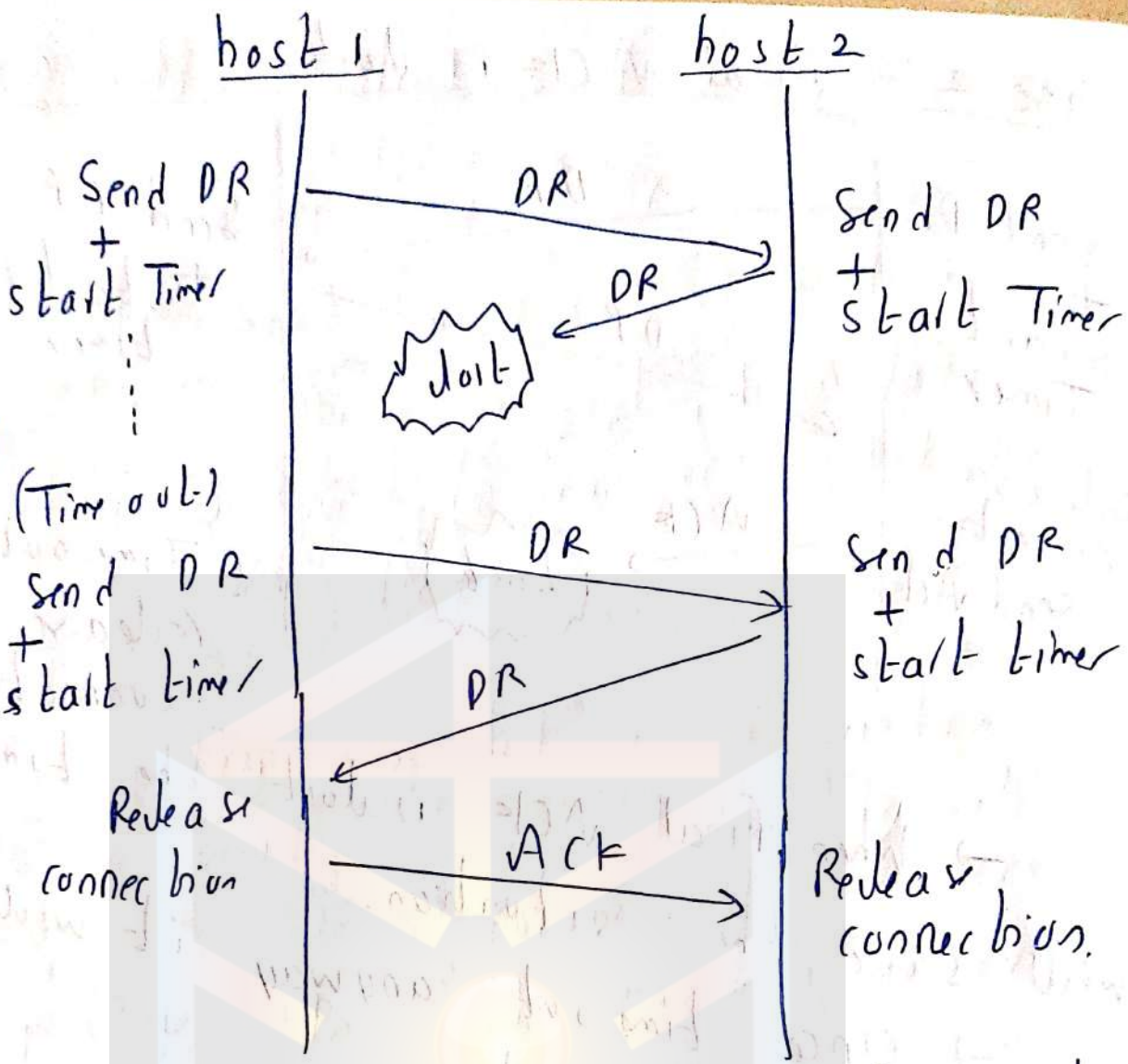
→ When final ACK is lost it will save the situation. it will

→ Once timeout anyway release the connection.

case 3 :- Response lost

→ This case, when second DR being lost. The request restarts after time. & finally released one successfull.

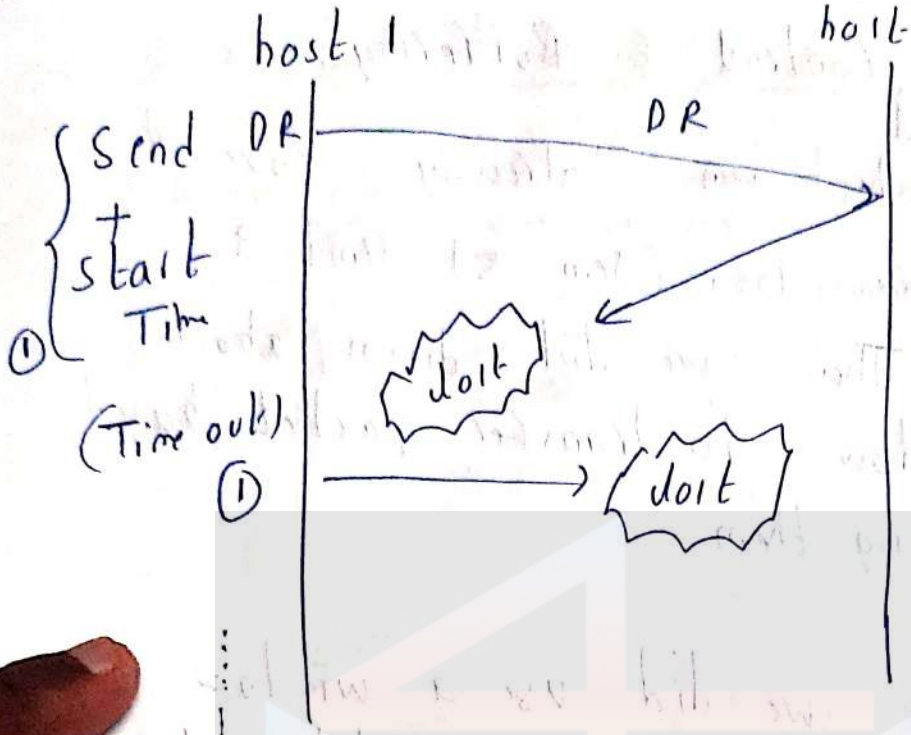
Education4Fun.com



case 4: Both response & subsequent DR's lost.

In this case, assume all the repeated attempts to retransmit the DR fails due to lost

Education4Fun.com



→ as after multiple host 1 will give up & release connection

→ after waiting for more time host 2 will also give up & release (as no info is shared.

Education4Fun.com

→ Flow Control & Buffering :-

↓

which was already covered in data communication (sem 5) Unit 3

Their we did discuss about sliding window to transfer packet based on numbering them

Their we did use a window size we wait for acknowledgement hence we use a buffer at receiver side to store the packets received

Buffering :- is commonly done in 3 ways

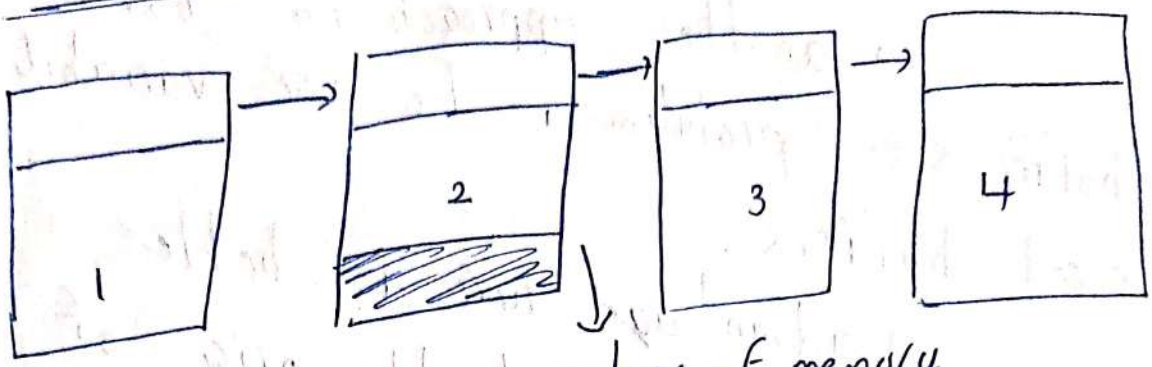
i) Fixed size buffers (chars)

ii) variable size buffers (chars)

iii) Circular Buffers

Education4Fun.com

i) Chained Fixed size buffer:



wastage of memory

or

large size memory cannot store

→ If the buffer size is chosen to be equal to the largest possible segment, space will be wasted whenever a short segment arrives

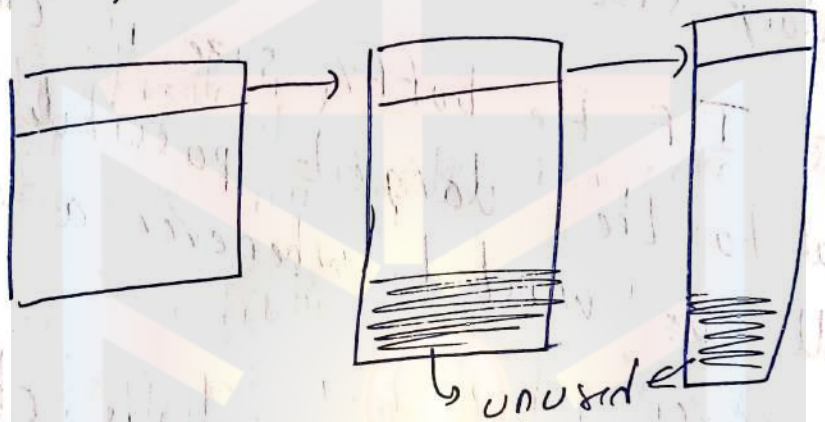
→ If the buffer size is chosen to be than, the maximum segment size, multiple buffers will be needed for long segments, with the attendant (make) complexity.

Education4Fun.com

ii) Chained variable sized Buffers

→ another approach to the buffer size problem is to use variable sized buffers.

→ advantage here is better memory utilizations, at the price of complexity.

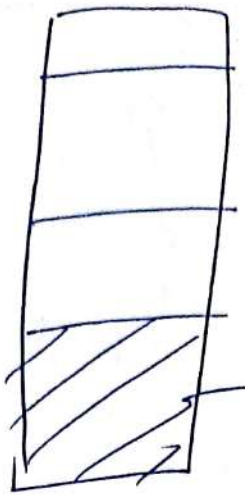


iii) One large circuit buffer per connection

→ A possibility is to dedicate a single large circular buffer per connection

→ Best, simple & donot depend on segment size, but makes good use of memory.

Education4Fun.com



hence to resolve these problems

→ Dynamic Buffer allocation:-

→ Dynamic buffer allocation means allocating buffers with variable size

→ The sender requests a certain number of buffers, based on its requirement

→ The receiver then grants as many of these as it can afford

Education4Fun.com

img -

→ Multiplexing :-

multiple conversation linked to one or distributed to many connections can be called as multiplexing with respect to transport layer

2 types

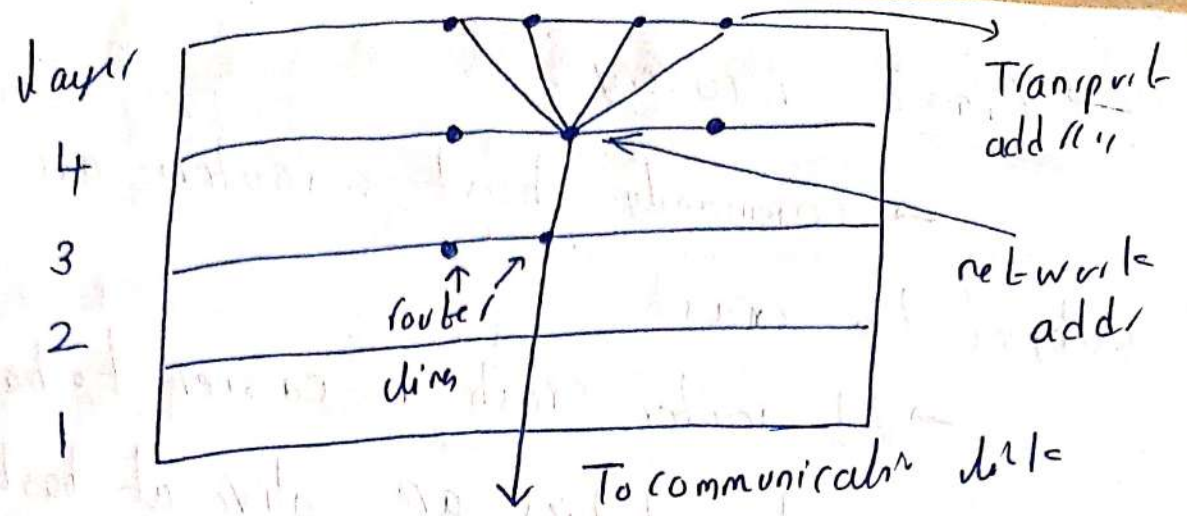
i) Upward multiplexing :-

In upward multiplexing the multiple connections are multiplexed on to a single connection.

Education4Fun.com

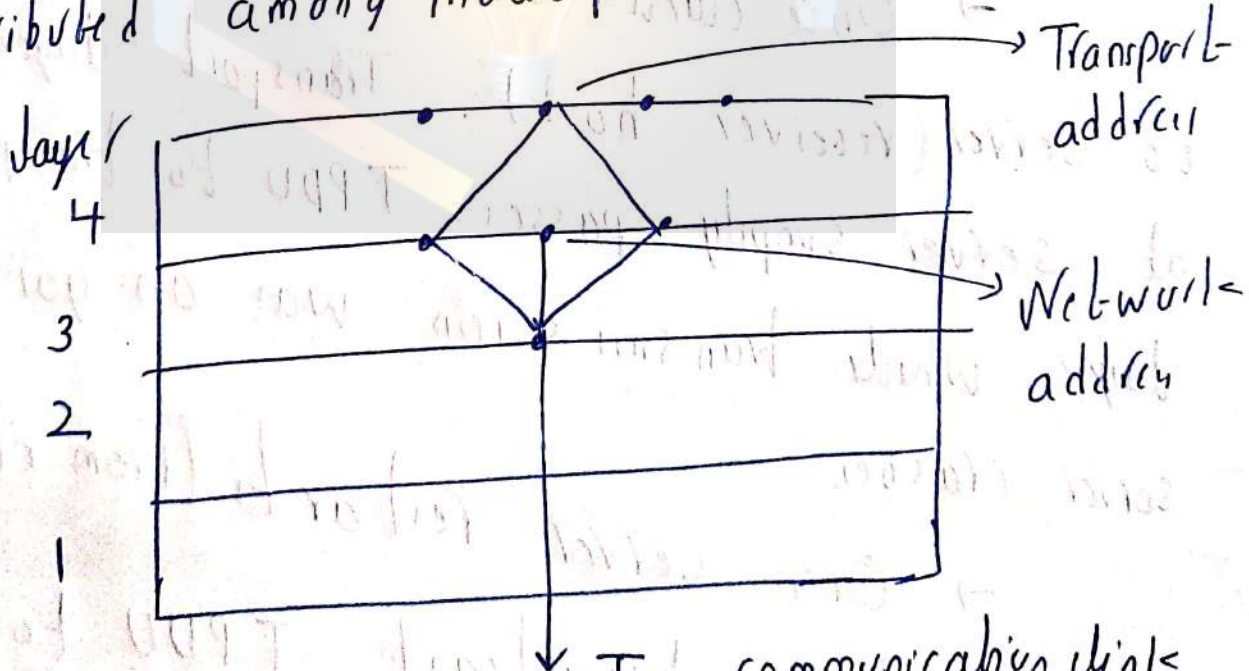
	A	Message	B	Comments
1	→	< request 8 buffers>	→	A wants 8 buffers
2	←	<ack = 15, buf = 4>	←	B grants messages 0-3 only
3	→	<seq = 0, data = m0>	→	A has 3 buffers left now
4	→	<seq = 1, data = m1>	→	A has 2 buffers left now
5	→	<seq = 2, data = m2>	• • •	Message lost but A thinks it has 1 left
6	←	<ack = 1, buf = 3>	←	B acknowledges 0 and 1, permits 2-4
7	→	<seq = 3, data = m3>	→	A has 1 buffer left
8	→	<seq = 4, data = m4>	→	A has 0 buffers left, and must stop
9	→	<seq = 2, data = m2>	→	A times out and retransmits
10	←	<ack = 4, buf = 0>	←	Everything acknowledged, but A still blocked
11	←	<ack = 4, buf = 1>	←	A may now send 5
12	←	<ack = 4, buf = 2>	←	B found a new buffer somewhere
13	→	<seq = 5, data = m5>	→	A has 1 buffer left
14	→	<seq = 6, data = m6>	→	A is now blocked again
15	←	<ack = 6, buf = 0>	←	A is still blocked
16	• • •	<ack = 6, buf = 4>	←	Potential deadlock

Education4Fun.com



→ There are 4 distinct transport connection & all of them use the same network connection

ii) Donwald multiplexing :- In downward multiplexing, a single connection is split & distributed among multiple connections.



Education4Fun.com

→ Crash recovery :-

→ commonly host & routers are subject to crash

→ At router crash is easier to handle since transport entities are alive at host, routers are only intemediate nodes which forward packets, they do not have any transport entity

→ Where as host crash can raise some problems

→ Once client (host) is sending a file to server (receiver host). Transport layer at server simply passes TPDU to transport layer. while transmission was on going server crashes.

→ Once server restarts (from crash) server sends a broadcast TPDU to all

Education4Fun.com

host announcing that it had just crashed & requesting that its client inform it about the status of all open connection

S_0 : no outstanding TPDU → sending all do nothing new to send

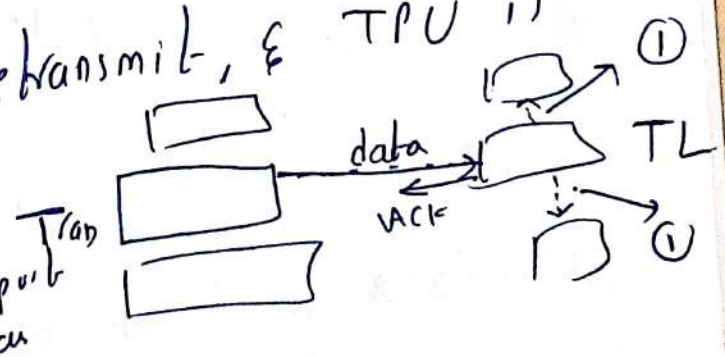
S_1 : one TPDU outstanding ↓ disconnected in middle still need to send.

S_0 → has no problem as it is done

S_1 → In outstanding case we got 2 situations

i) if server has first sent ACK & before it can send TPDU to next layer, server crashes.

→ In this case client will get ACK so it will not retransmit, & TPU is lost by server



① not transmitted to any layer (commonly upper).

Education4Fun.com

ii) if server first send packet to next layer, then it crashes before it can send ACK. In this case through server has already received a TPDU, client thinks TPDU is lost & and it will retransmit

There data could duplicate as it didn't received ACK!

Education4Fun.com

Strategy used by sending host	Strategy used by receiving host: First ACK, then write			Strategy used by receiving host: First write, then ACK		
	AC(W)	AWC	C(AW)	C(WA)	W AC	WC(A)
Always retransmit	OK	DUP	OK	OK	DUP	DUP
Never retransmit	LOST	OK	LOST	LOST	OK	OK
Retransmit in S0	OK	DUP	LOST	LOST	DUP	OK
Retransmit in S1	LOST	OK	OK	OK	OK	DUP

OK = Protocol functions correctly
DUP = Protocol generates a duplicate message
LOST = Protocol loses a message

Education4Fun.com

→ The internet Transport protocol :-

TCP :-

will be discussed later → UDP is a simple protocol & has many uses but it cannot provide reliable & sequenced delivery which is needed in most internet application hence TCP is used).

→ TCP is a connection oriented, reliable & most commonly used protocol in transport layer

→ TCP was specifically designed to provide a reliable end to end byte stream over an unreliable internetwork (topologies, bandwidth, delays, packet size etc...).

Characteristics of TCP:-

i) Connection - Oriented :- Before actual data transfer, two communication processes

Education4Fun.com

must exchange control segments to establish a connection. This means processes must first hand shake each other (more reliable)

ii) Full Duplex :- The connection establish is bidirection. So data transfer will take place in both direction.

iii) Point to point :- connection established b/n the single sender & single receiver

iv) Byte streeming (continuous flow of data) (may be segments)

v) Piggy backing (Ack + Data).

vi) error flow, flow control & congestion control.

Education4Fun.com

→ TCP service model:-

→ TCP service is obtained by both sender & receiver creating end points called socket.

→ each socket has a socket number (address) consisting of the IP address (32 bit) of the host and a 16 bit number local to that host called port (16 bit)

→ Port is the TCP name of TSAP (Transport service access point).

→ Port number can be assigned from abov 1024 (below are reserved).

few are

Port	Protocol	Application
21	FTP	File transfer
23	Telnet	Remote login
25	SMTP	email
69	TFTP	Trivial FTP
79	Finger	look up
80	HTTP	WWW
110	POP 3	Remote email acces
119	NNTP	USENET news

Education4Fun.com

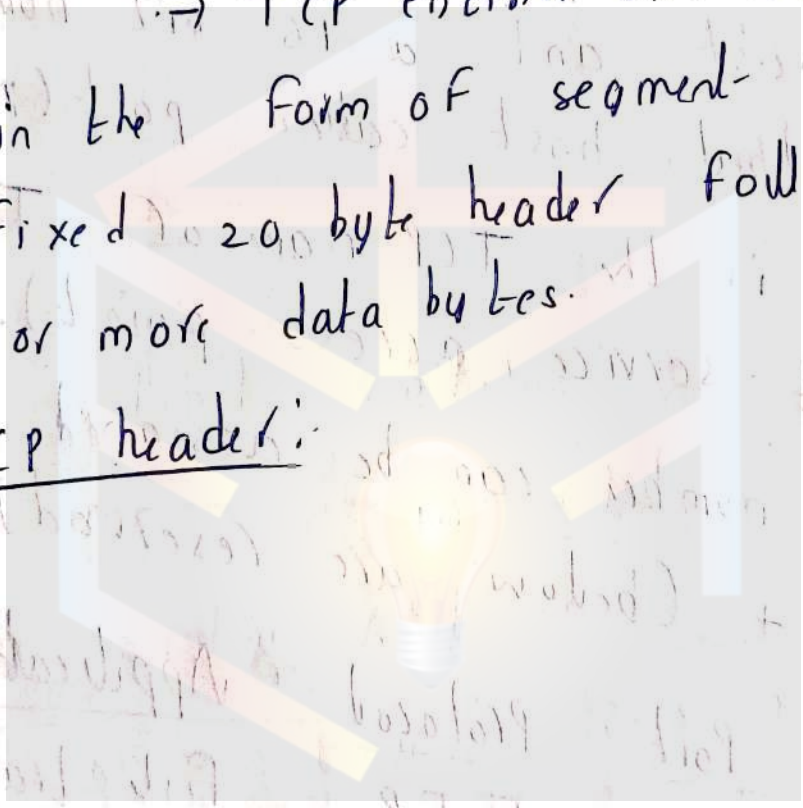
→ The TCP Protocol:- (unit intro)

→ Every byte in a TCP connection has its own 32 bit sequence number.

→ The sequence number used for ACKs & for window mechanism.

→ TCP entities exchange the data in the form of segment consisting of fixed 20 byte header followed by 0 or more data bytes.

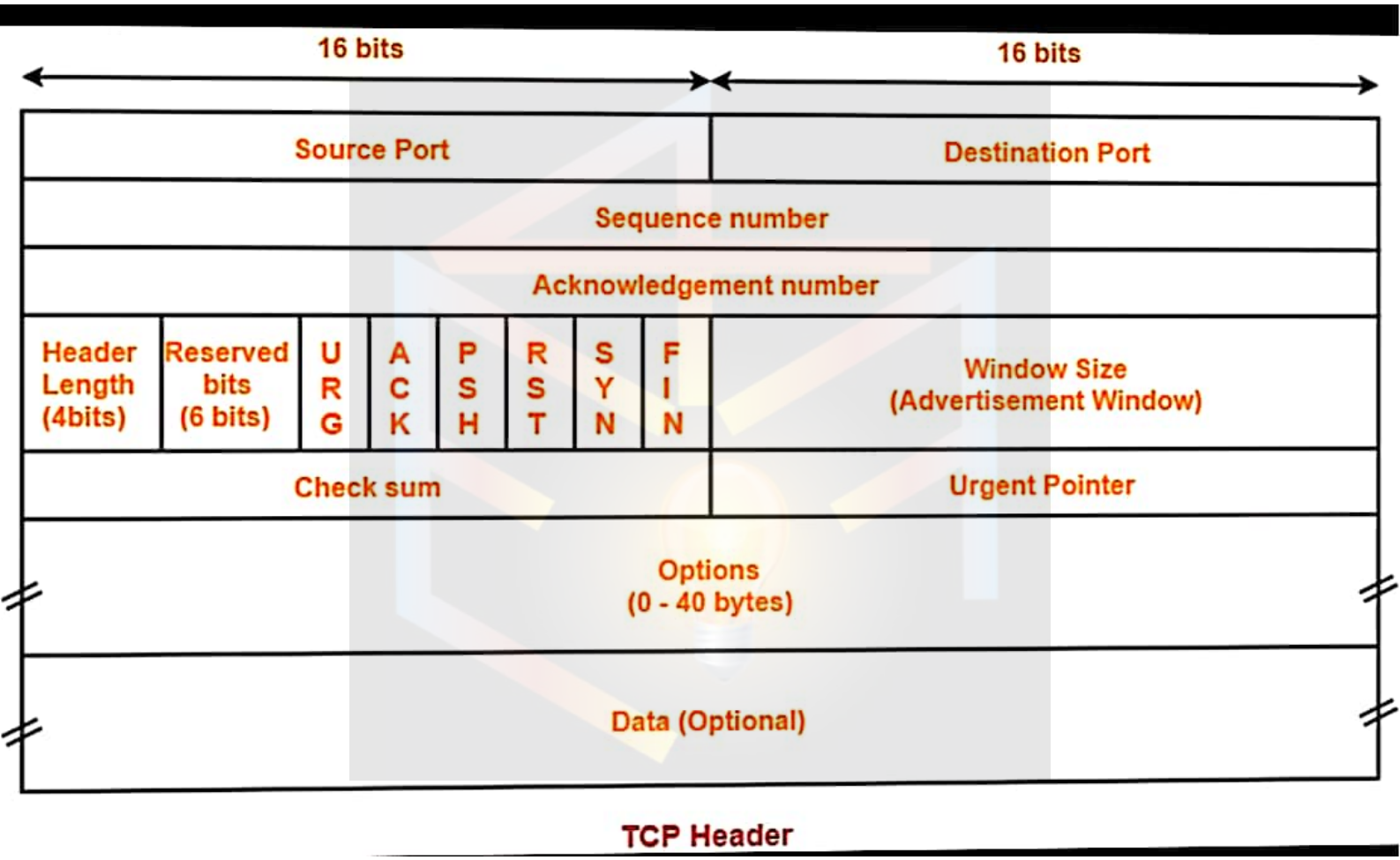
→ TCP header:-



img.

(20-60 bytes)

Education4Fun.com



TCP Header

Education4Fun.com

→ Source Port & Destination Port :- (16 bit)

↳ It identifies source port of the application process on the sending device

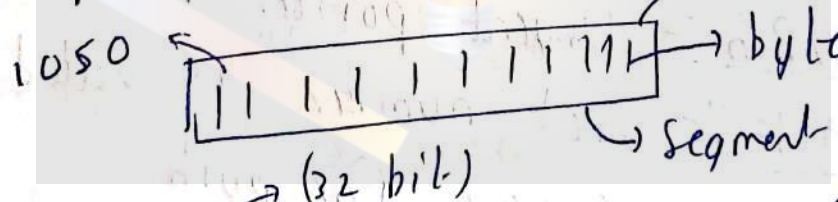
→ destination Port :- It identifies destination port of the application process on the receiving device (16 bit).

(0 - 65535) → Port

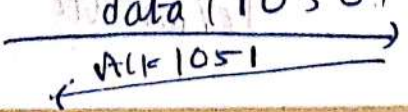
0 - 1023 — well defined

1024 - 65535

→ Sequence number :- (32 bit) Sequence number of data bytes of a segment in a session.



→ Acknowledgement number :- (32 bit) This field that holds the ACK number, i.e, the byte number that the receiver expects next.



//.

Education4Fun.com

→ Hlen (header length) (4 bit) it indicate the length of the header.

4 bit → 0000 scale of 4

1111

0.10 ⟹ 20 (5×4)

:

1111 ⟹ 60 (15×4)

are valid

→ Reserved bit (6 bits) Reserved for further use & all are set zero by default.

→ Flags & Flags (6 bit) :-

urgent = 1 — URG :- Urgent pointer is valid

has ack = 1 — ACK :- Ack number is valid

Push it = 1 — PSH :- request for push

reset connect = 1 — RST :- reset the connection

Sync = 1 — SYN :- Synchronize sequence number

Terminate = 1 — FIN :- Terminate the connection.

Education4Fun.com

→ Window size: (16 bit)

It is used for maintaining flow control. It will show how many receiver is willing to accept.

→ Check sum:- (16 bit)

It is used for error control (an error control mechanism)

→ Urger Pointer Field: (16 bit)

It points to the urgent-data byte if urgent flag is set to 1.

(1000 - 1015)

(1015)

→ Optional any extra facility commonly not provided in regular header.

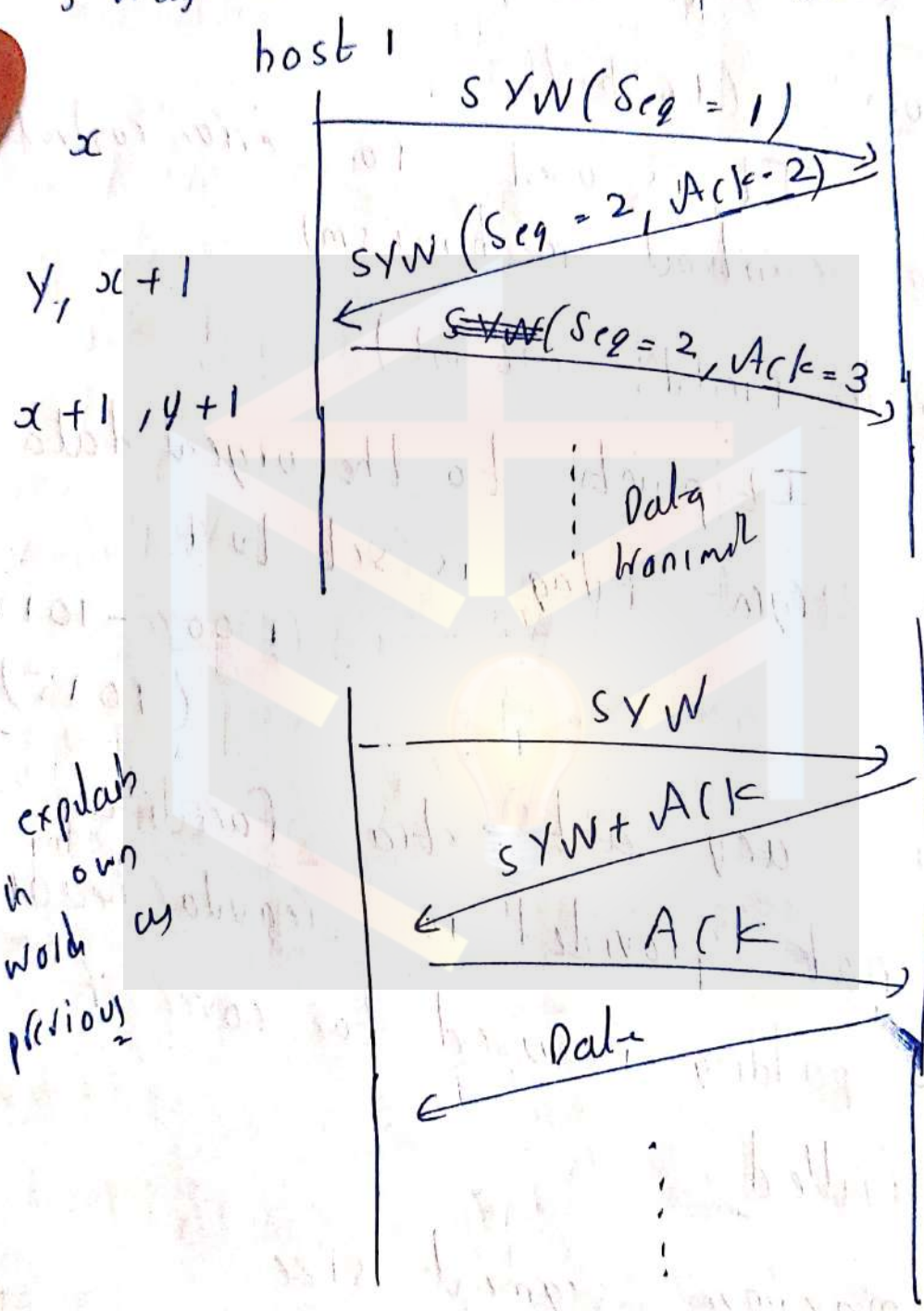
→ padding is used to cover if it's not filled.

mss maximum segment size.

Education4Fun.com

→ TCP connection Establishment

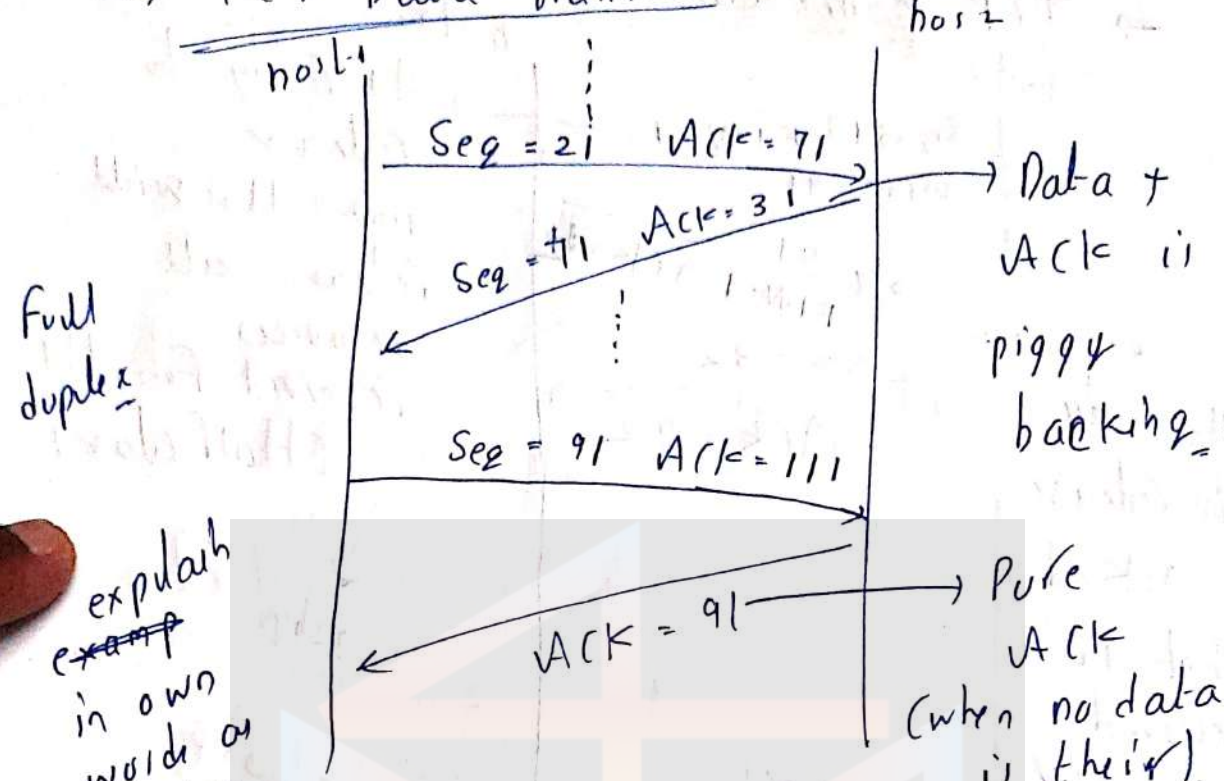
→ To establish connection we use 3 way handshake.



explain in own words as previous

Education4Fun.com

→ TCP Data transfer



Full duplex

explain ~~example~~ in own words as previous

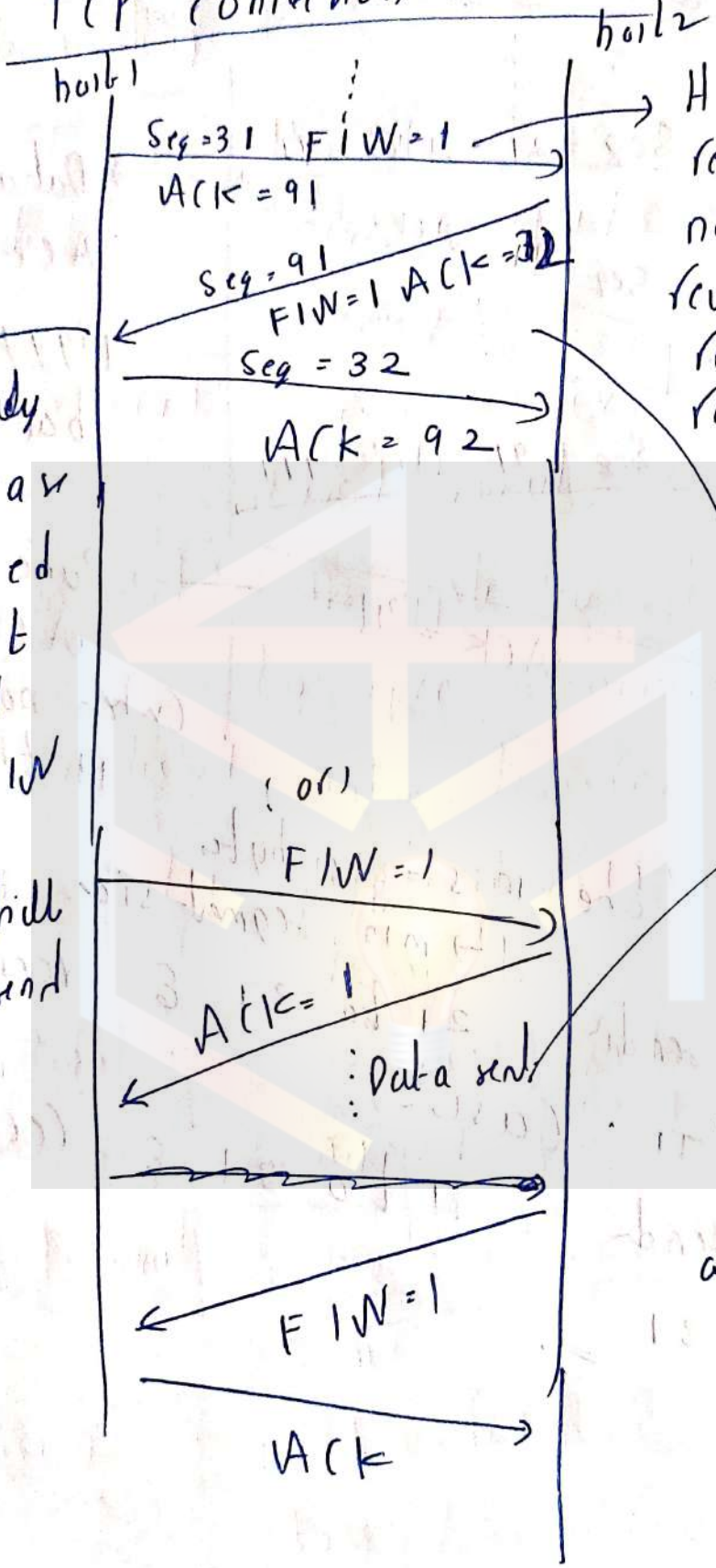
→ assume the mss = 10 byte
↳ min segment size.

→ H1 send 21 to 31 & received till 71 (assu-)

→ H2 send 71 to 81 & received till 31 =.

Education4Fun.com

→ TCP connection termination



H1 ready to release now H2 will release all resources reserved for H1 (Half close)

3 steps

4 steps

(or)

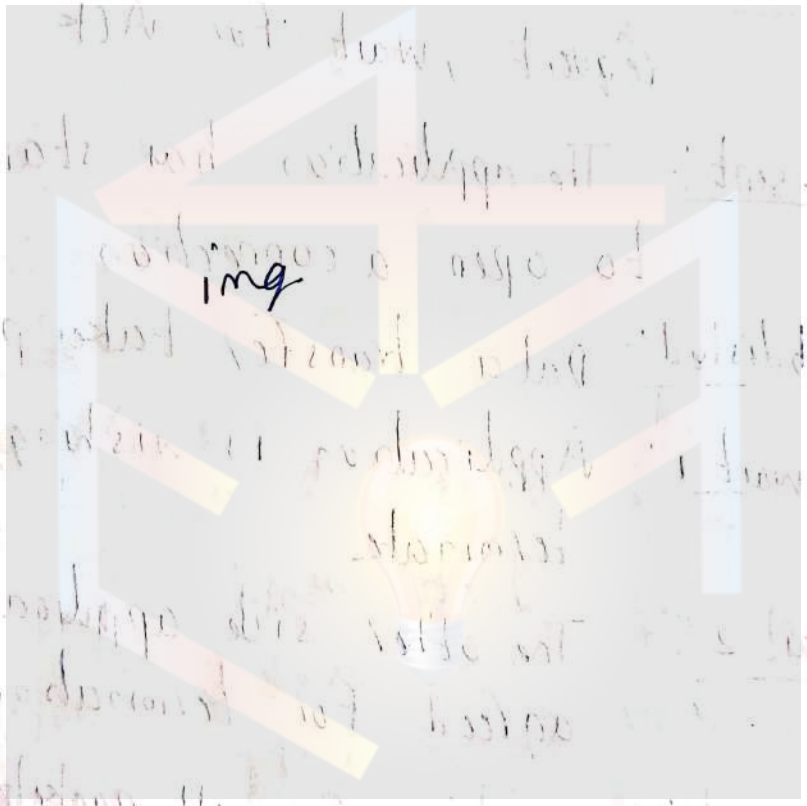
H2 ready to release & acked that it received previous FIN

host 2 still wants to send data

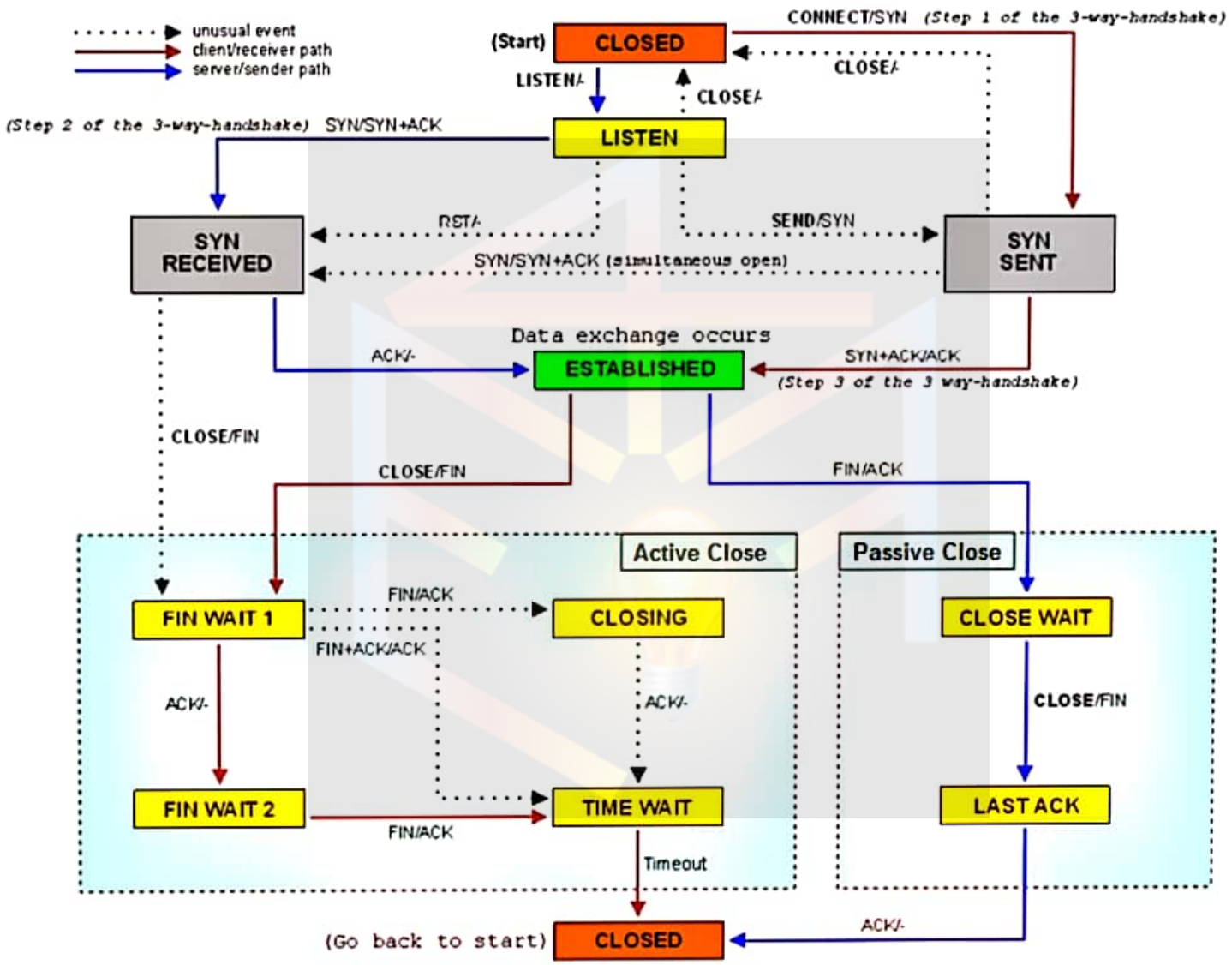
: Data sent

explain in own words as previous

→ TCP Connection management model:



Education4Fun.com



Education4Fun.com

States in diagram:

→ Closed :- Their is no connection

→ listen :- Server is waiting for an incomming call.

→ Syn-rcvd :- Server received a connection request, wait for ACK

→ Syn-sent :- The application has started to open a connection

→ established :- Data transfer takes place

→ Fin wait 1 :- Application is wishing to terminate

→ Fin wait 2 :- The other side application agreed for termination.

→ Timed wait :- waiting for all packets to die.

→ Closing :- Both sides have tried to close simultaneously

→ Close wait :- The other side has initiated a release.

Education4Fun.com

→ Last ACK Waiting for last ACK =

observe diagram

in diagram ~~after~~ blue lines server path
red lines client path
above establish is establishment phase
below establish connection termination.

step/path

→ initial server keeps on listening.
→ client connects to server calling SYN
→ server is ready to receive & moved to SYN Received. with a ACK sent
→ client sent seq with ack
→ Now connection established
→ using sliding window data is transferred
→ once all done
→ if client want to close connection (FIN)
→ & for ack & FIN from server
→ once recieved it will close

use the diagram names =

Education4Fun.com

→ TCP Transmission Policy :-

img

→ Window management in TCP

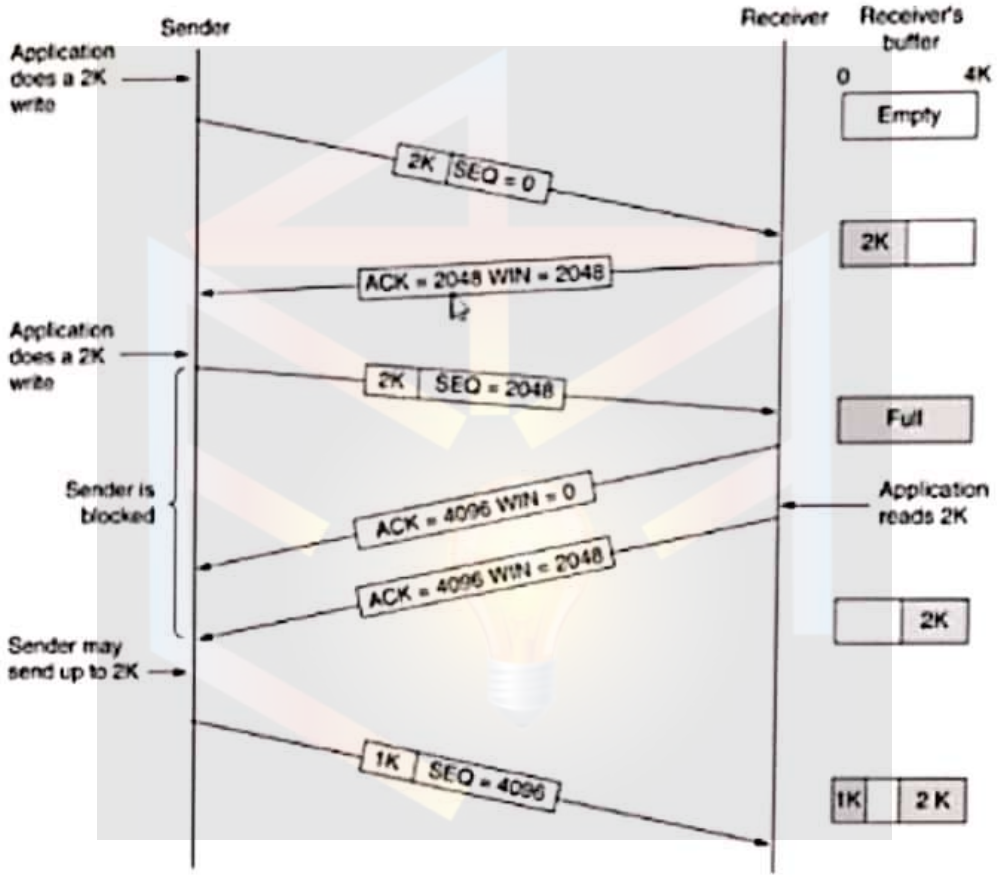
→ In simple words we can say as per the buffer space we store data & once full ship transmit

→ Once back empty we continue sending back

(explain diagram / flow in own words)

Education4Fun.com

TCP Transmission Policy

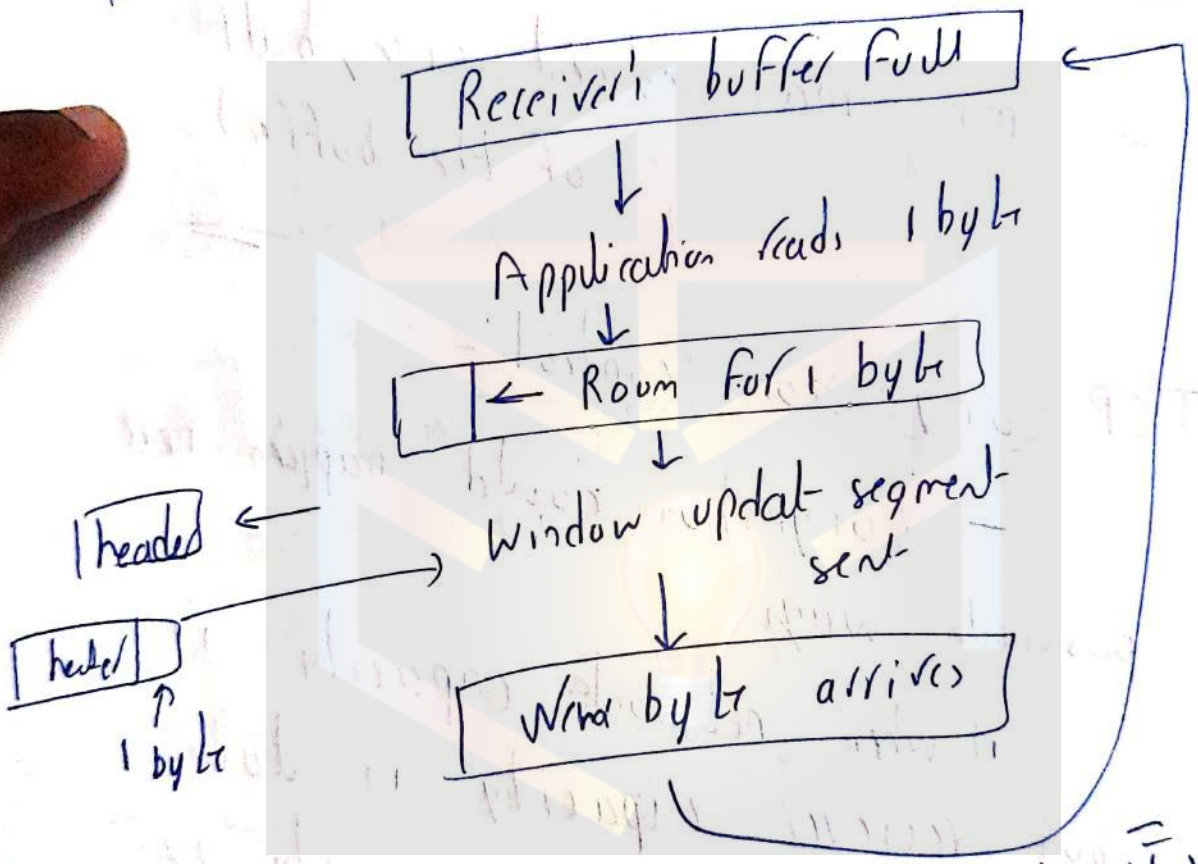


Window management in TCP.

Education4Fun.com

→ Silly window Syndrome (a problem):

When buffer is full, application usually reads byte by byte if sender with the empty space it will transmit & the buffer will always be full.



Solution for this problem → (sender side)

i) Nagle solution: when data comes into the sender in small pieces send once & start collecting into buffer until you get ACK &

Education4Fun.com

the buffer once there is enough space

ii) Clark's Solution: Receiver should not send a window update if it can handle the maximum segment size

→ min (max segment size, half of the buffer)

→ TCP Congestion control:-

→ Congestion could happen here in 2 possible ways

i) when network capacity is high but receiver capacity is low.

ii) when network capacity is low but receiver capacity is high congestion could occur.

Education4Fun.com

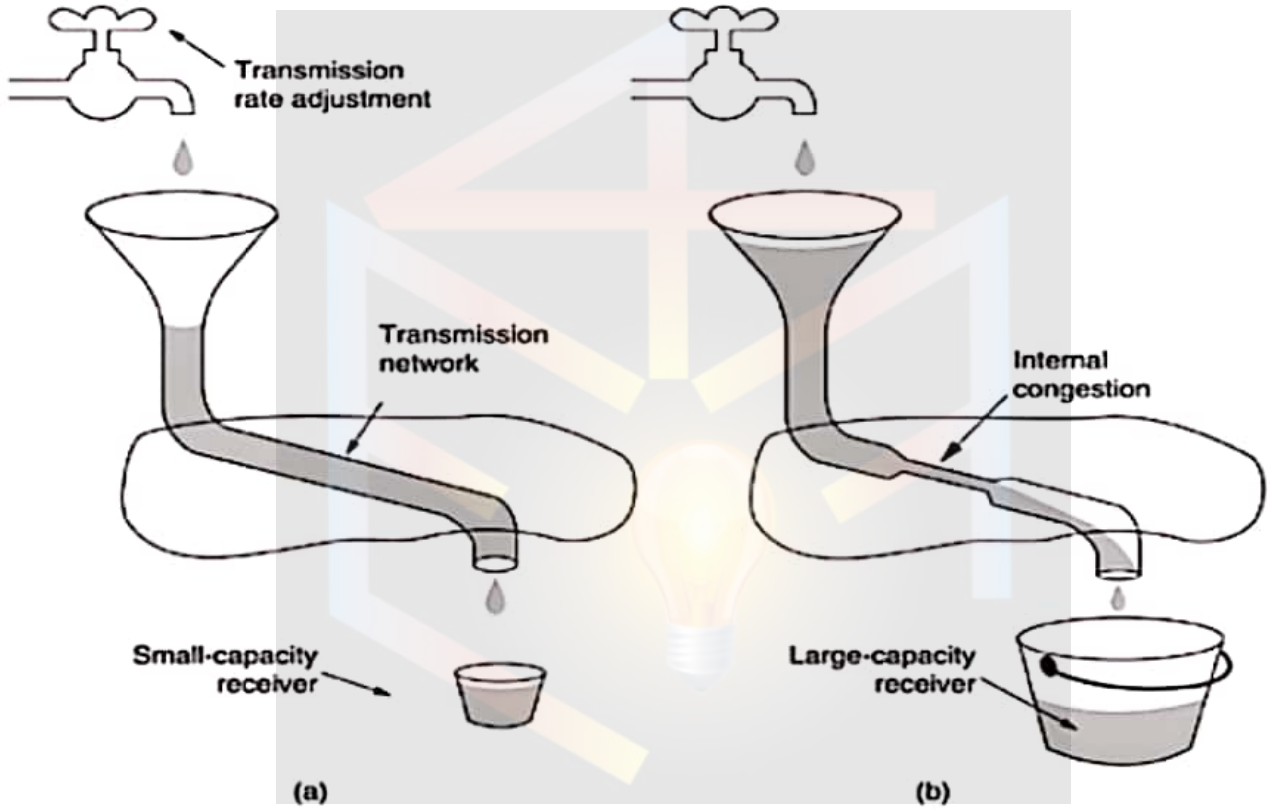
ima

exact theory explanation not found

→ fig (a) we see a thick pipe leading to a small capacity receiver. As long as sender doesnot send more water than the bucket can contain, no-water will be lost

→ fig (b) The limitation factor is not the bucket capacity, but the internal carrying capacity of the network. If too much water comes in too fast, it will no backup & some will be lost

Education4Fun.com



Education4Fun.com

→ Internet congestion control algo:

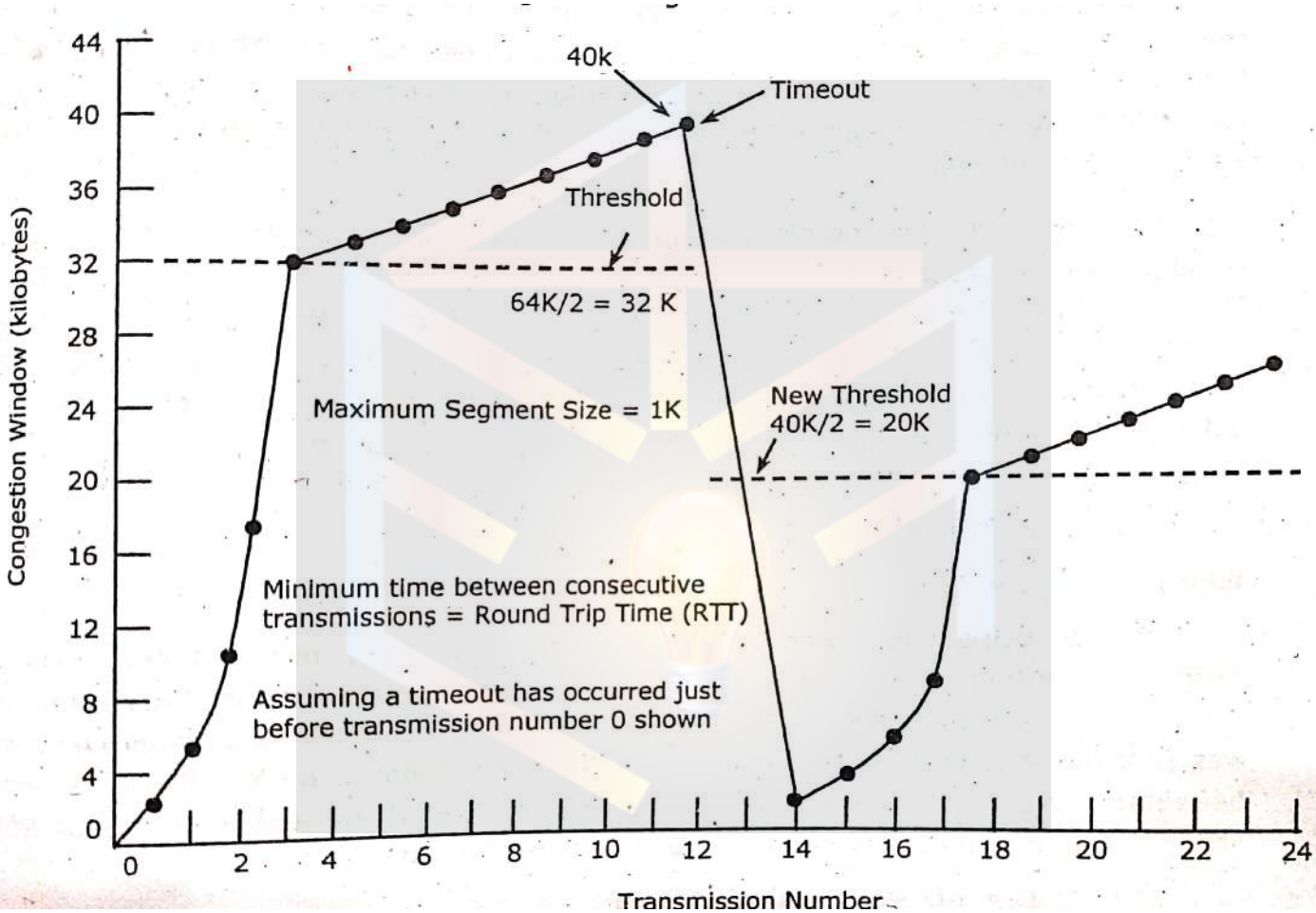
→ This algorithm uses three parameter threshold, receiver & congestion window

→ when congestion / timeout occurs the threshold is set to half of the current congestion window & now congestion window is reset to one maximum segment.

→ initially grow exponentially after threshold grow slowly

img:

Education4Fun.com



Education4Fun.com

→ TCP Timer management

Time management is very important concept that is used while transmitting data

4 types

i) Retransmission timer
ii) Persistent timer
iii) Keep Alive timer
iv) Time wait timer

i) Retransmission timer :-

→ From the name it's clear retransmission of data (when to do)

→ retransmision is done when we don't receive ACK from reciever due to (lost data, lost ACK, corrupted data).

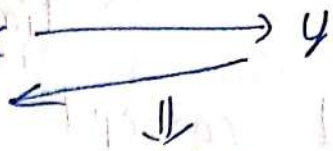
→ Hence the question is when to retransmit.

→ So we use a time to consider when to retransmit

Education4Fun.com

Smoot round trip tim (SRTT)

x to y & y to x



the total time is called SRTT

$$\Rightarrow SRTT = \alpha \, SRTT + (1-\alpha) R$$

α → 7/8, SRTT → previous measured value, R → time of ACK

$\therefore$ retransmission timeout = 2 * SRTT (RTO)

Jacobson proposed

RTTVAR (round trip time variation)

$$RTTVAR = \beta \, RTTVAR + (1-\beta) | SRTT - R |$$

β → 1/4, RTTVAR → previous

$\therefore$ final RTO = SRTT + 4 × RTTVAR

Education4Fun.com

ii) Persistence Timer: This timer is set when the receiver is sending an update packet to the sender. [when buffer Full]
→ But lost, but of them will wait for each other.
→ Once this timer expires the sender transmits a packet to receiver which replies with the window size.
→ if window size is non zero then data is transferred or else timer is set again.

iii) Keep alive timer: This timer is used to checking if the connection is still alive or not, if the connection is idle for a long time
→ Once timer expires the other terminal is checked to see if the connected (still) if not connection is terminated.

Education4Fun.com

iv) Timed wait time: It is used to ensure that the connection on both terminals have been closed and all the packet have died.

→ UDP (User Datagram protocol):

→ UDP supports a connectionless transport service. (like postal service).

Characteristics

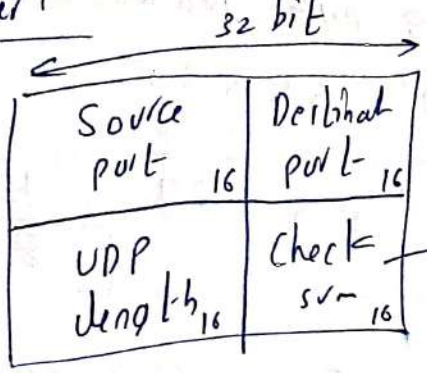
→ No connection establishment
→ No connection states
→ Small packet header overload

disadvantage

→ not reliable (loss of packet can occur)
→ not in order (received packet could be not in order)

Education4Fun.com

→ UDP header:



Source port (16 bit)

It identifies source port of the application process on the sending device.

destination port :- It identifies destination port of the application process on the receiver side 16 bit (0 - 65535) port

0 - 1023

well defined

1024 - 65535

UDP length :- length including header & dat

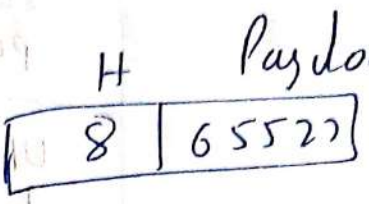
16

(0 - 65535)

Education4Fun.com

8 byte header

→ 65535
− 8
65527



Checksum for error control.

checksum = UDP header + UDP Data + Pseudo header of IP.

Applications

→ Query response protocol [DNS]
→ speed (online games)
→ Broadcasting / multicast
→ streaming (youtube, skype)

TCP	vs	UDP
→ Connection oriented		→ Connectionless (not in order)
→ Reliable, order		→ less reliable
→ error control is mandatory.		→ error control is optional

Education4Fun.com

→ slow transmission → Fast transmi
→ high overhead → less overhead
→ Flow control, → No FC, CC
congestion control
→ DNS
→ HTTP, FTP

Education4Fun.com