26/6/18

Unit - I

COMPUTER NETWORK's.

Data Communication —

Communication is of Many types..

* Means Exchanging of data between 2 device

1) Data Communication.
2) Tele Communication (Far Kind of communication)

Data Communication:— It is the Exchange of data between two devices through some form of transformation Media such as Wired Cable Called Connection Oriented data Communication or the Cable is wireless then it is Wireless Communication. Called tele Communication.

Advantages:— delivary; time; Accuracy; Jitter.

2

Components of data Communication:—

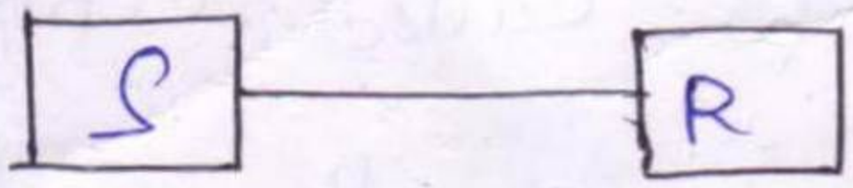
Protocols

Data Representation:— Data Representation in the form of text, Numbers, images, Audios Video files.

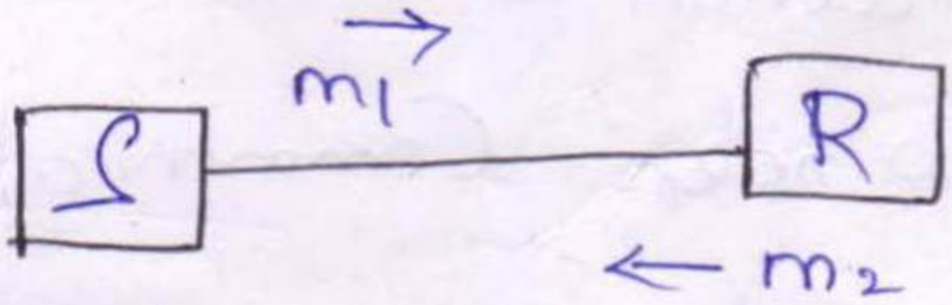
Data flow of Data Communication:—

It is of 3 types.

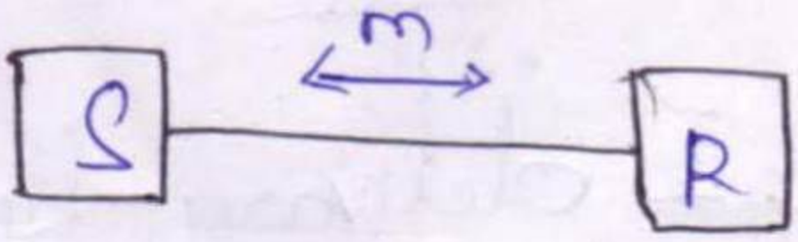
1) Simplex



2) Half duplex



3) Full duplex



Networks :—

A Network is a interconnecton between a set of devices cable of Communication.

29/6/18

Protocol:— It is defined as a set of rules to be followed by all the devices for Communication is known as protocol.

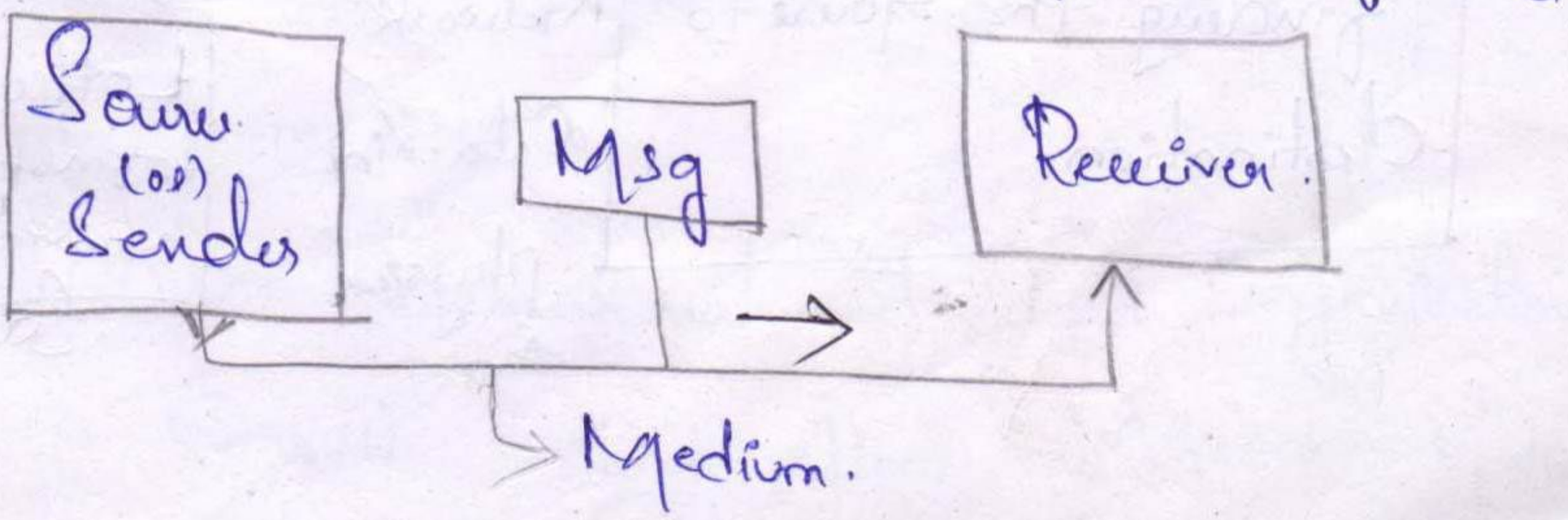
There are five Components for Communication each Component has to follow Some rules for a peacefull Communication.

The First topic is One-to-one Communication.

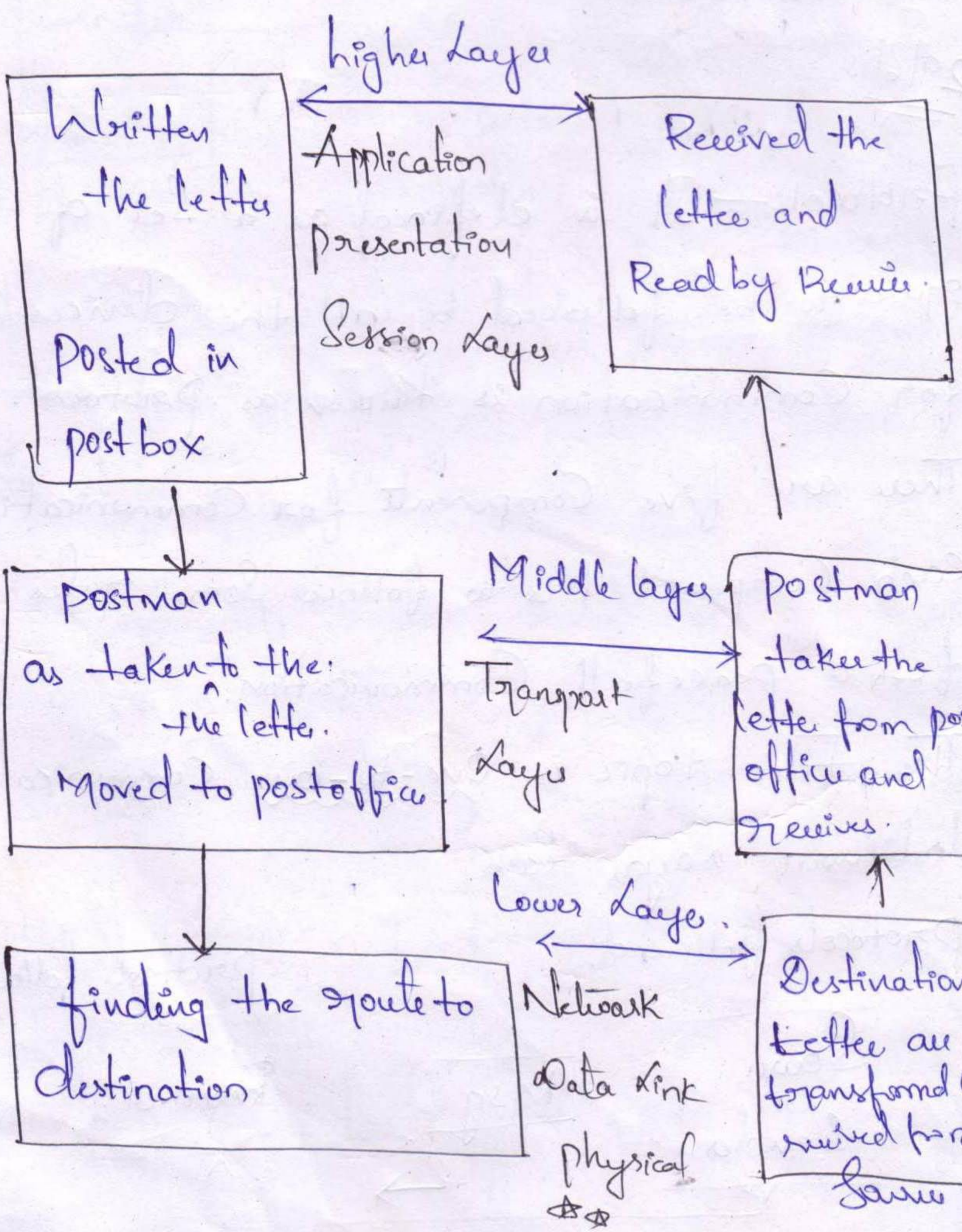
Without Layer task

protocols followed

protocols followed



The Second topic follows as the layer.



The three layers in the layer task are devided into 7 layers they are.

1) Application Layer. (APPS)
2) Presentation Layer. (Dialogue)
3) Session Layer. (Syntax)
4) Transport Layer. (Checks the path can route) — Middle layer.
5) Network Layer Packets
6) Data link Frames
7) Physical Layer. Bits

} Higher Layers.

} packets ↓ frames lower layer.

These 7 layer are also called Open Source Interconnection". (OSI).

2/7/18

Some of the Standards and Committees are Implemented to develop OSI reference Model.

Some of them are

1) "ISO" — International Standard Organisation
It is a Committe which allows Maximum of the

6 Applications and Standards. for Network based Communication. This Committe has to give permission for various National organisations to implement New technologies.

2) "IEE" - Institute of Electrical and Electronics Engineer.

→ It Mainly Focusing on the networks with Communication Standards. It is developed in the year 1963 and almost 160 Countries are Combined with IEE.

3) "ITU" - International telegraph can tele Communication union receive the This Communication gives the to Signals to the (or) providing a tele Communication based on ^permission Connection Oriented.

Very imp

OSI Reference Model.

Open Source (or) System Interconnection Model.

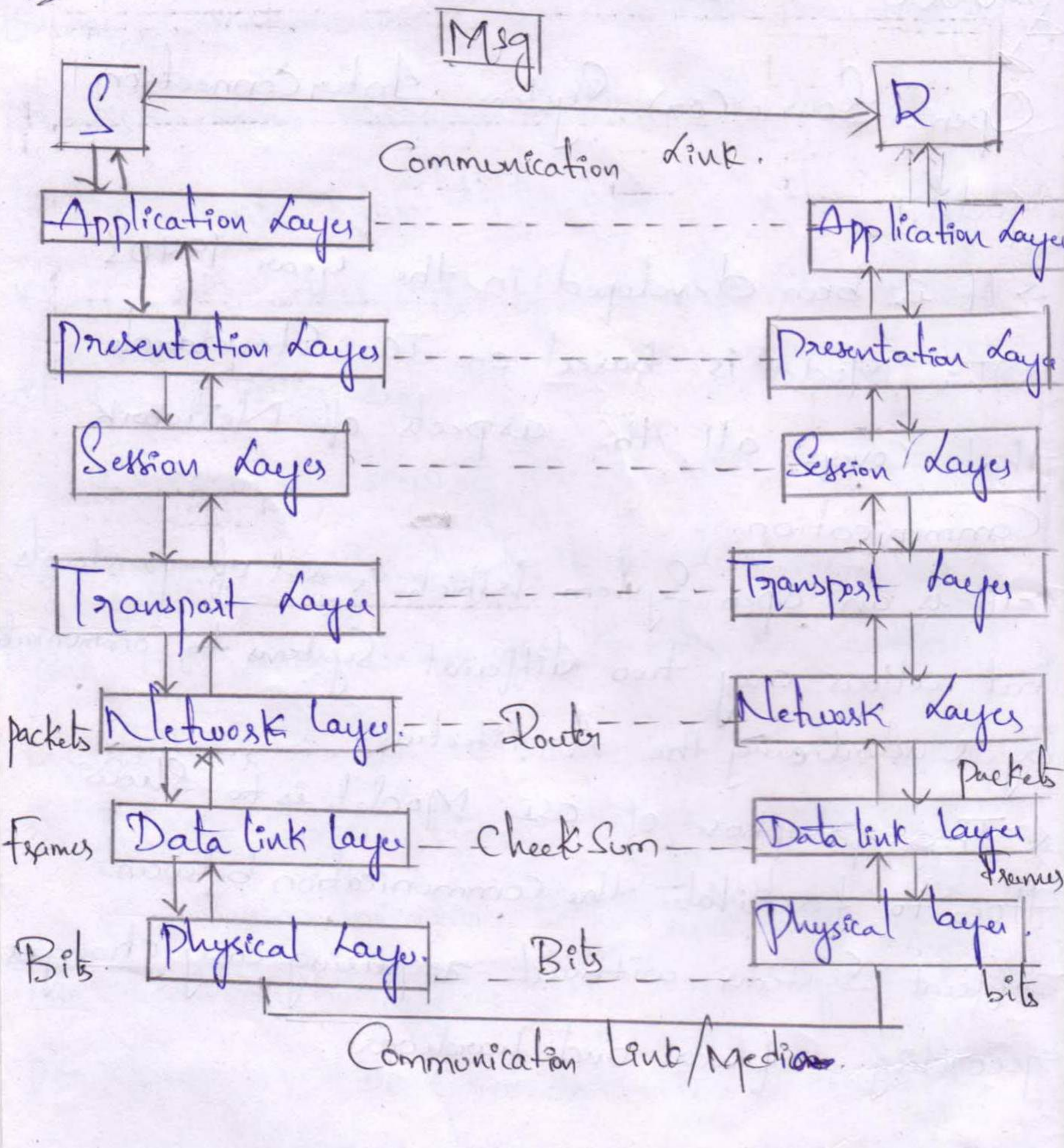
→ It is been developed in the year 1970's

→ This Model is based on Iso Standards that Covers all the aspects of Network Communication.

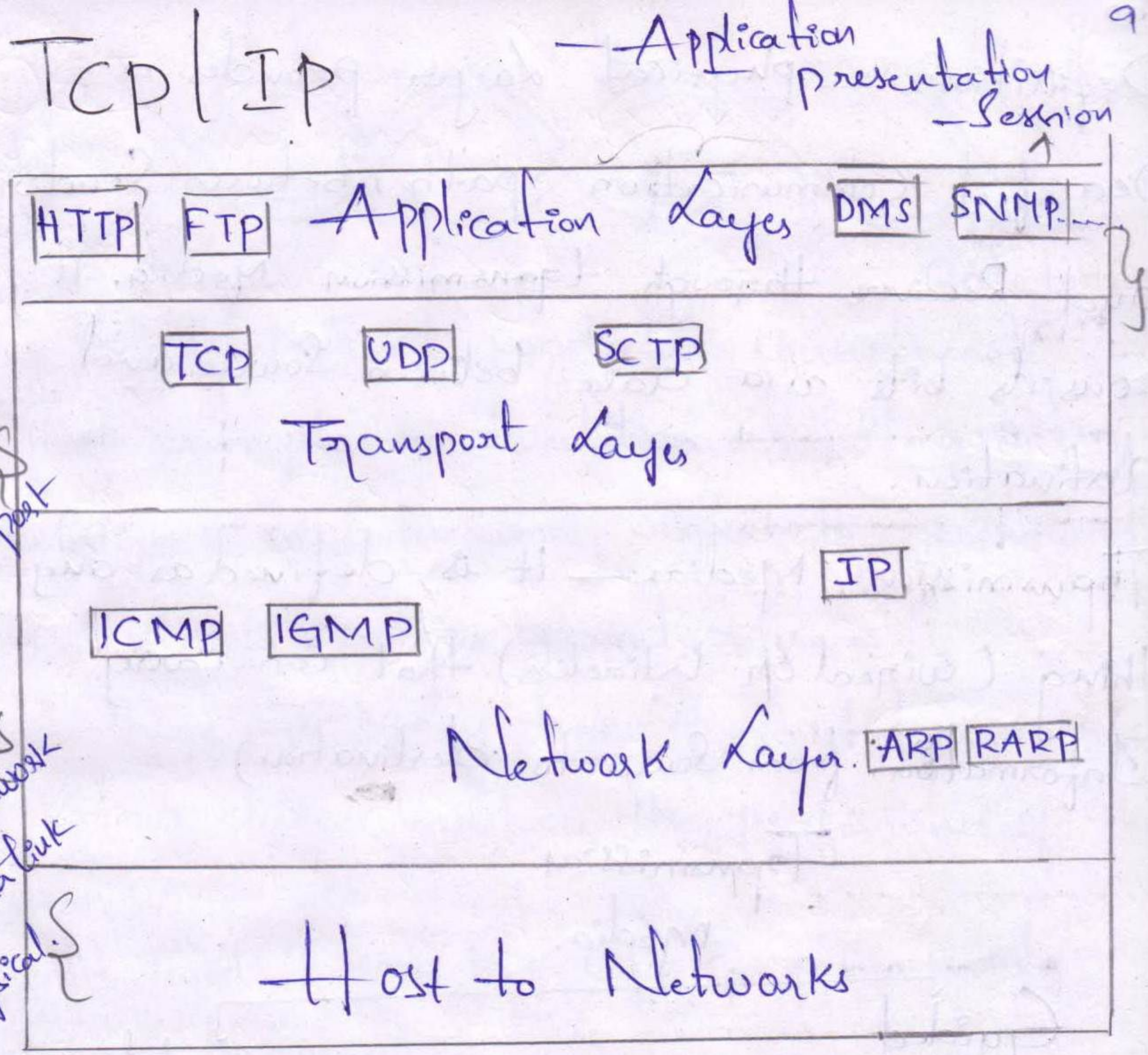
→ It is an Open System which is set of protocols that allows any two different Systems to Communicate by underlining the Architecture.

→ The purphose of OSI Model is to Show How to facilitate the Communication between different Systems without requiring any changes recording Software and hardware.

3/7/18. # OSI Reference Model.



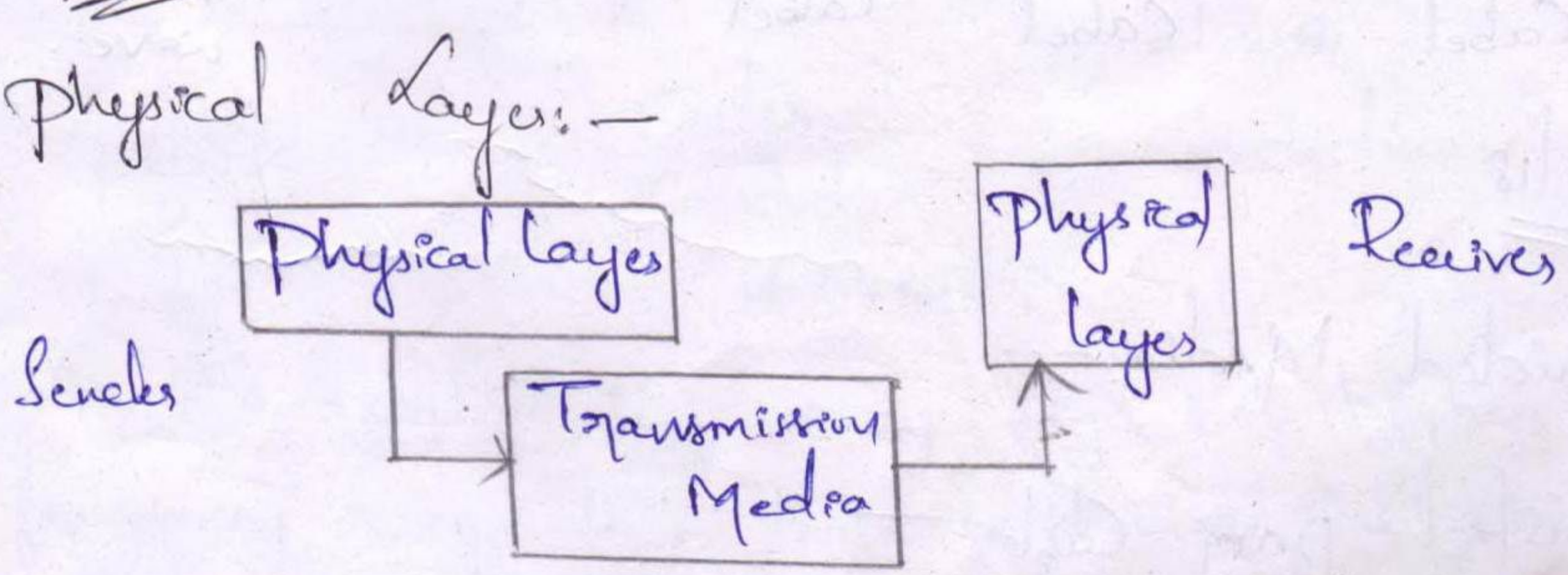
TCP/IP



Transmission Control protocol/Internet protocol.

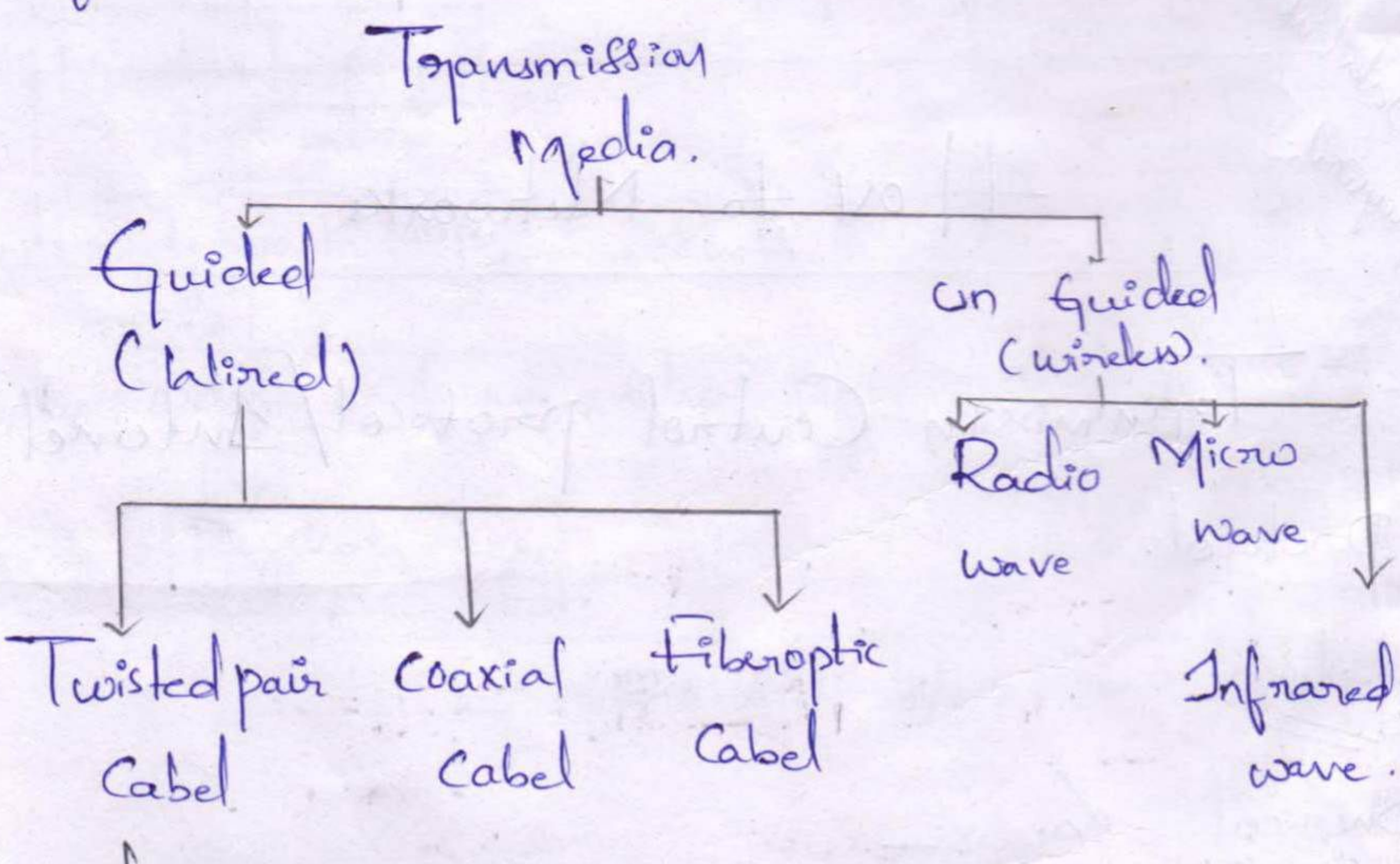
4/7/18

Physical Layes:—



Definition:— physical Layer provides a peaceful Communication path, between Sender and Receiver through transmission Media. It Consists bits as a data between Source and destination.

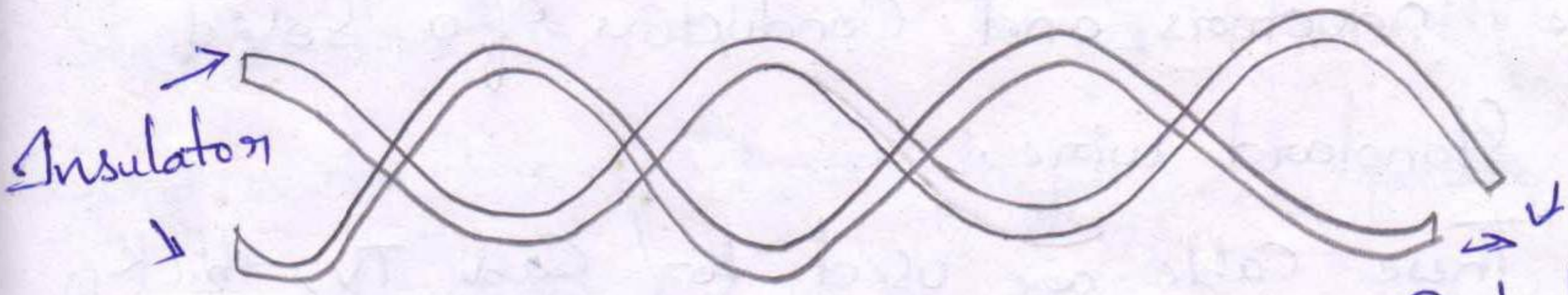
Transmission Media:— It is defined as any thing (wired or wireless) that can carry Information from Source to destination)



6/7/18

Guided Media:—

Twisted pair Cable:—

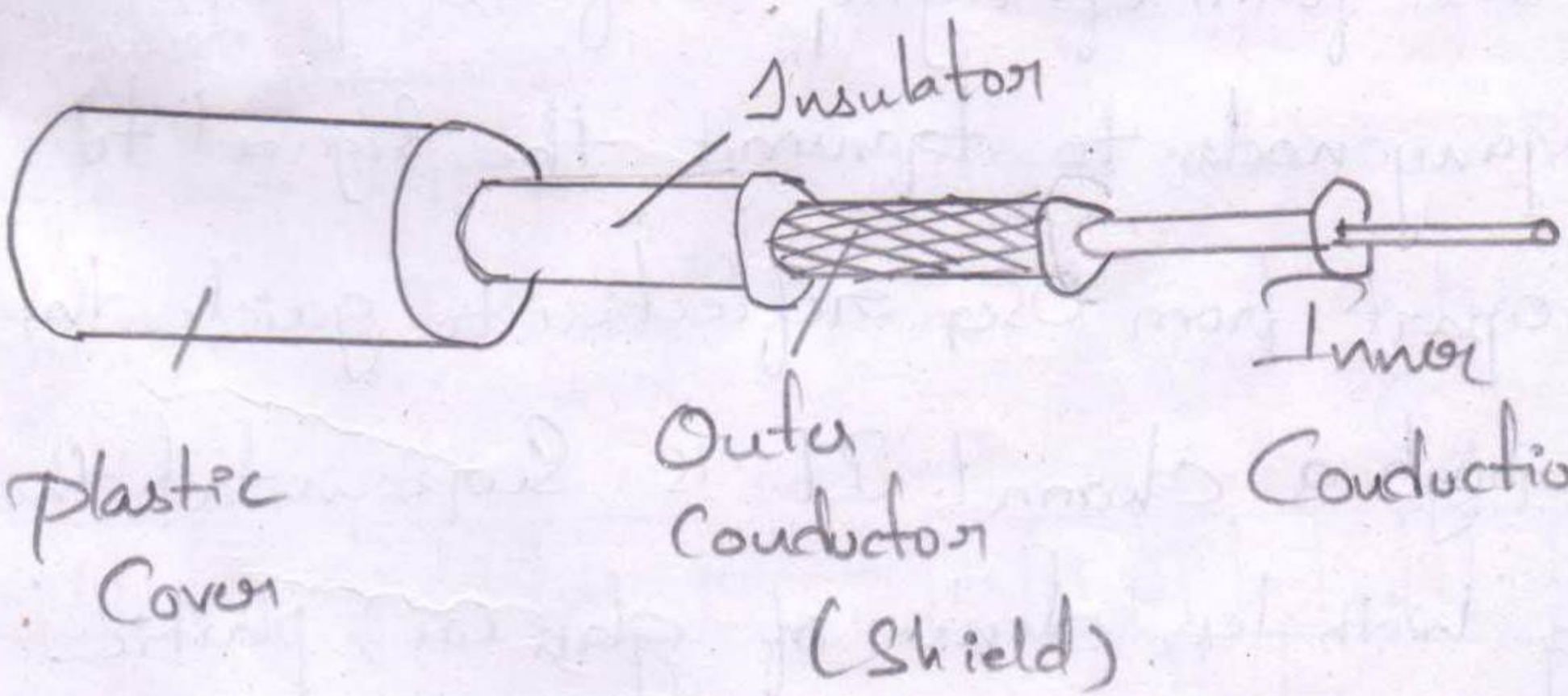


A Twisted pair of Conductor Consistreach with its own plastic Insulation as shown in fig. One wire is used to Carry Signal to receive and Other wire is used as ground refeerence.

There are Many types of twisted wires used for Various devices based on the performance of the twisted wire Cable.

A Twisted wire pair Cable Can pass a wide range of frequencis.

Co-axial Cable—



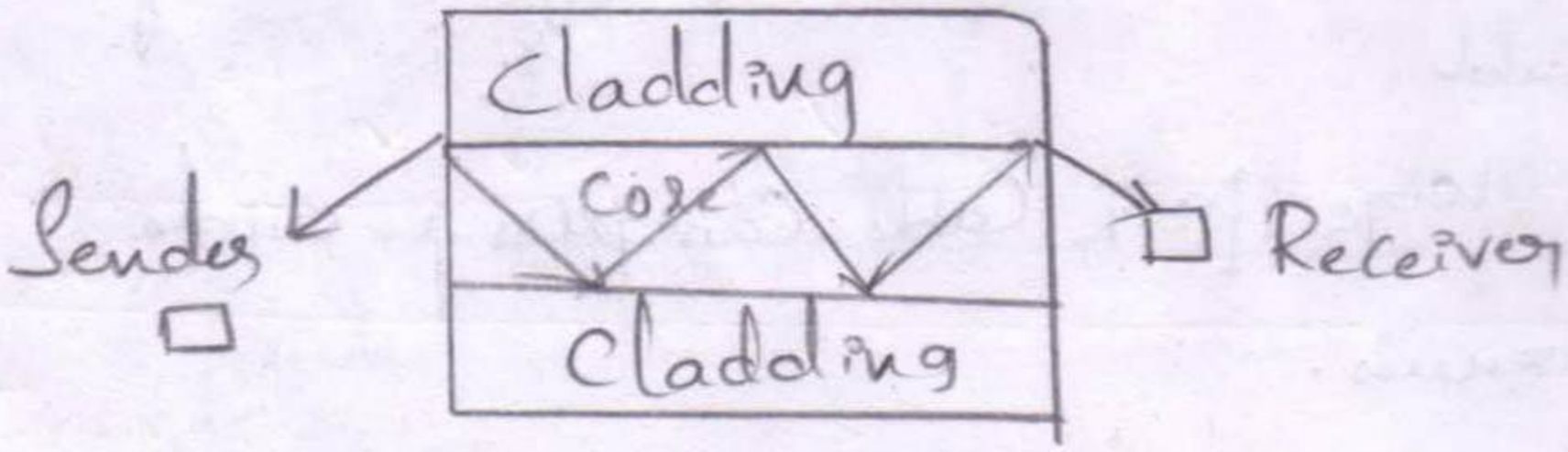
A Co-axial Cable Carry Signal of higher frequency range than those in twisted Cable, instead of having two wires Co-axial Cable has Central

Core inductors and Conductors of a Solid and Standard wire.

These Cable are used for Cable TV, thick ethernet, thin ethernet used for motors.

The performance is high compared to twisted pair Cable by Measuring in band width Signals frequency.

Fibre Optic Cable —



It is Made of glass (or) plastic and transmit Signal in the form of light. A fibre optic Cable is of So Many nodes to transmit the Signal to Receiver apart from Use of reflection to guide the light through a channel. It is Surrounded the cladding with less dense of glass (or) plastic.

Data Link Layer — It is located in Between physical and Network Layer it provides

Services to the network Layer it provides Services to the Network layer and receive Services from physical layer.

The Major designation for data link Layer are framming, flow Control, error Control.

Data Link Layers are divided into two Sub Layers (Wired).

logical link Control and Medium access control (Wireless).

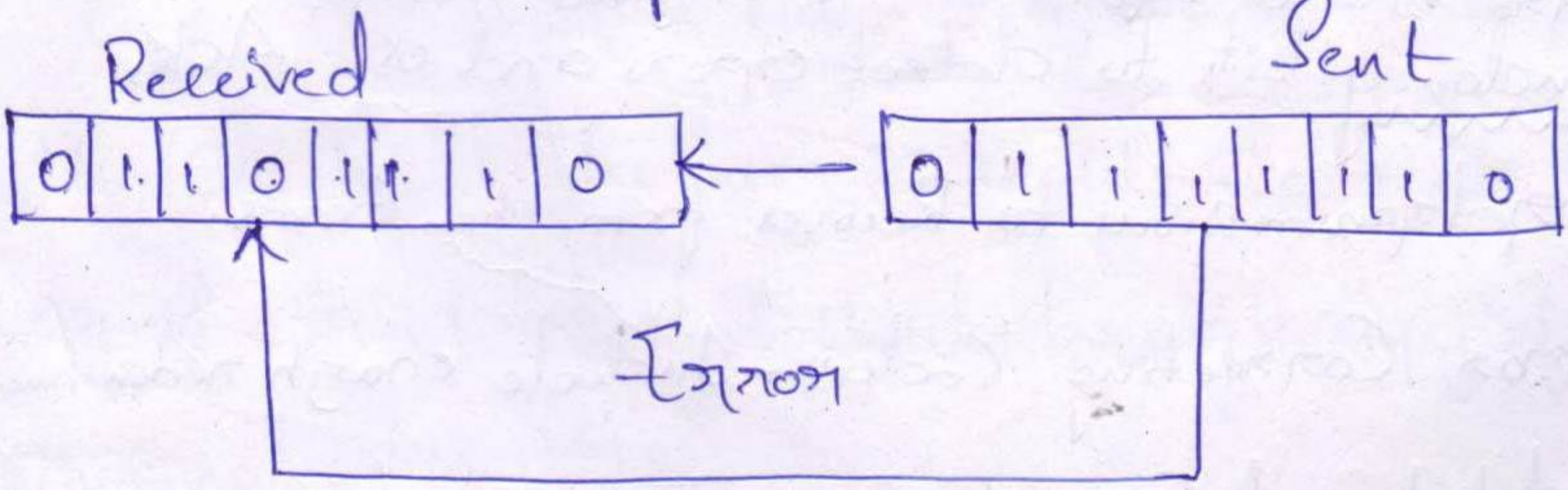
9/7/18 Error Correction and detection.

Types of errors: —

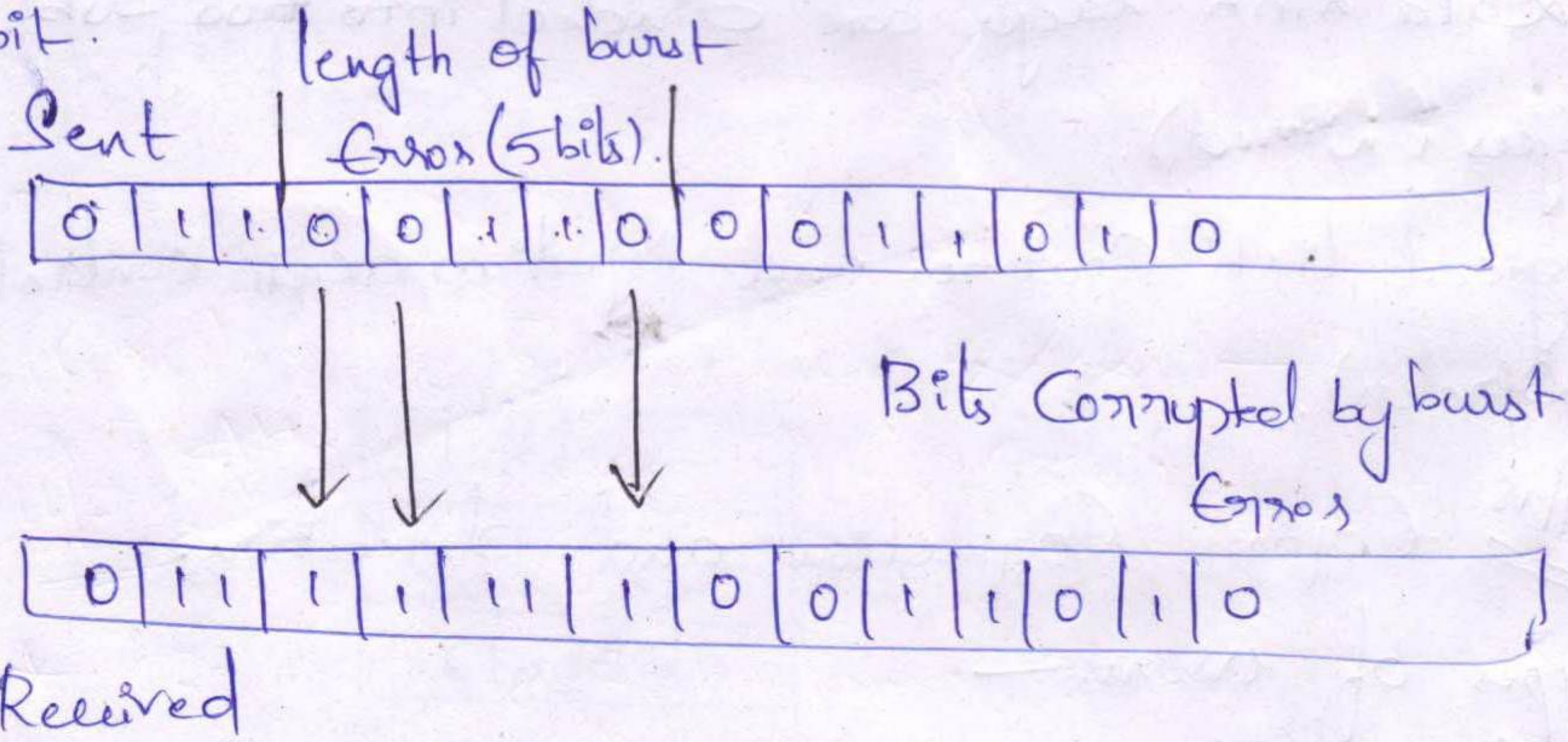
1) Single bit errors.

2) Burst errors

Single bit error: — It Means only one bit of data unit is Changed from 1 to 0 or from 0 to 1.



Burst Error:— It Means. two (or) More bits in data unit are changed from 1 to 0 from 0 to 1. In burst Error, it is not necessary that only consecutive bits are changed. The length of the burst Error is Measured from first changed bit to last changed bit.



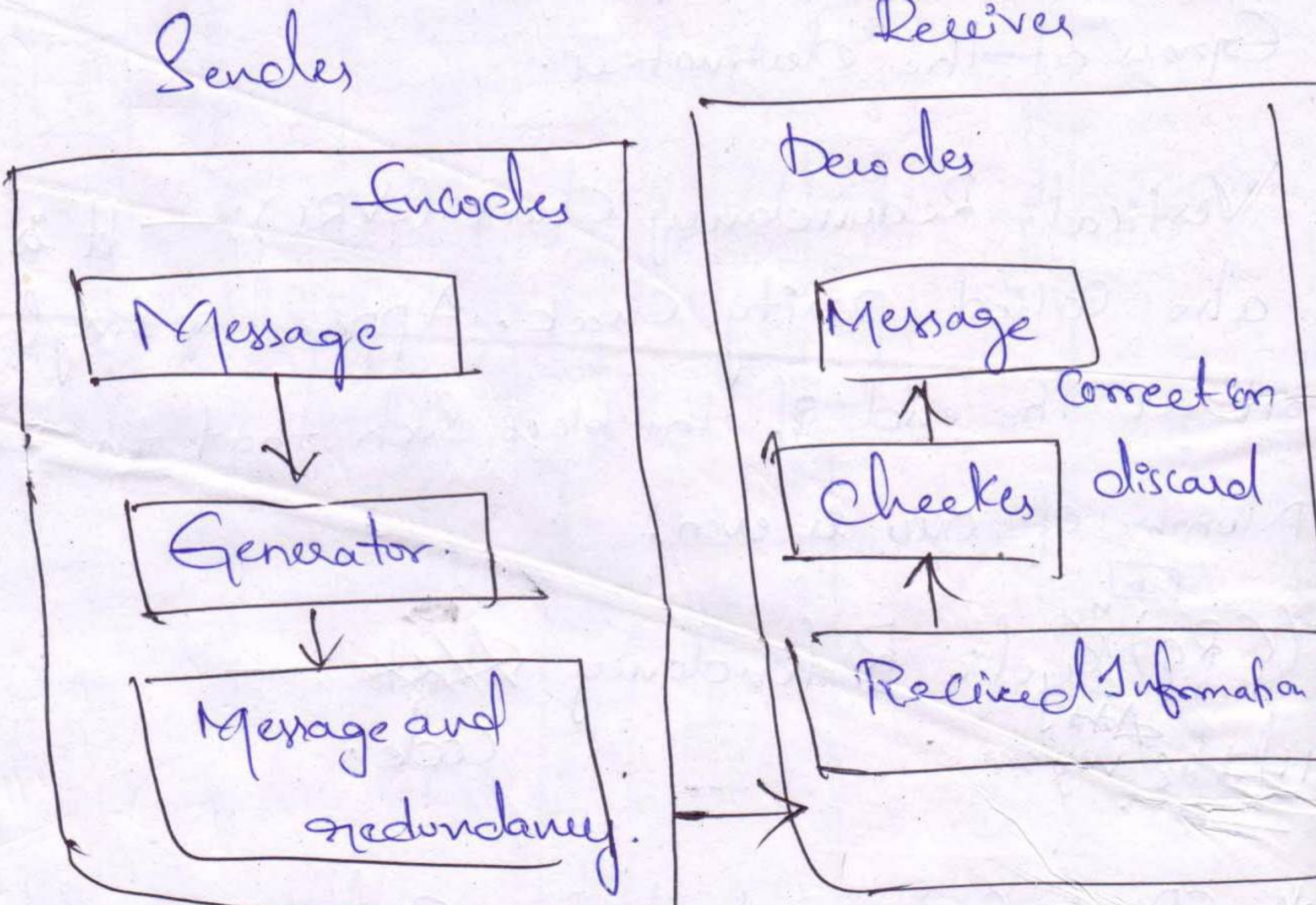
Error Correction Vs Error detection.

There are two types of attacts against Errors.

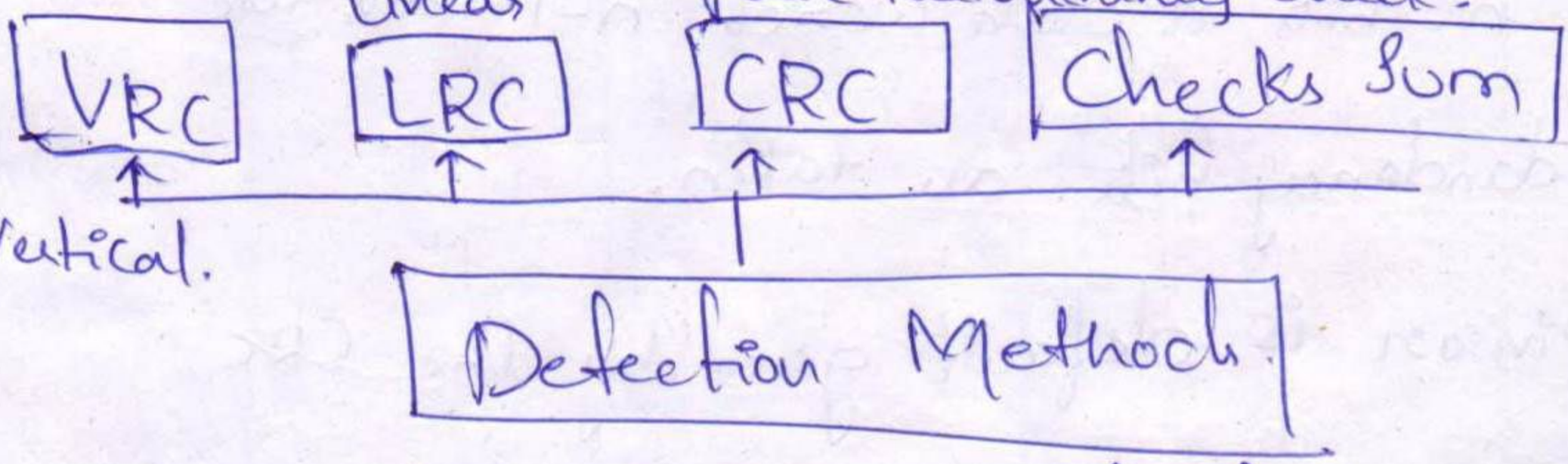
1) Error Detecting Code:— Include enough (Extra bits are added to the original data). Redundancy bits to detect errors and use Acks and retransmissions to Recoves from the Errors.

2) Error Correcting Codes:— Include enough redundancy to detect and Correct Errors. The Use of Error —

Correcting codes is often referred to as forward Error Correction. 15



Error detection:— It is of four types
Linear Cyclic Redundancy check.



It checks whether received data is Correct (or) Not. Without having a Copy of the Original Message.

Error detection use the concept of redundancy which means adding extra bits for detecting Errors at the destination.

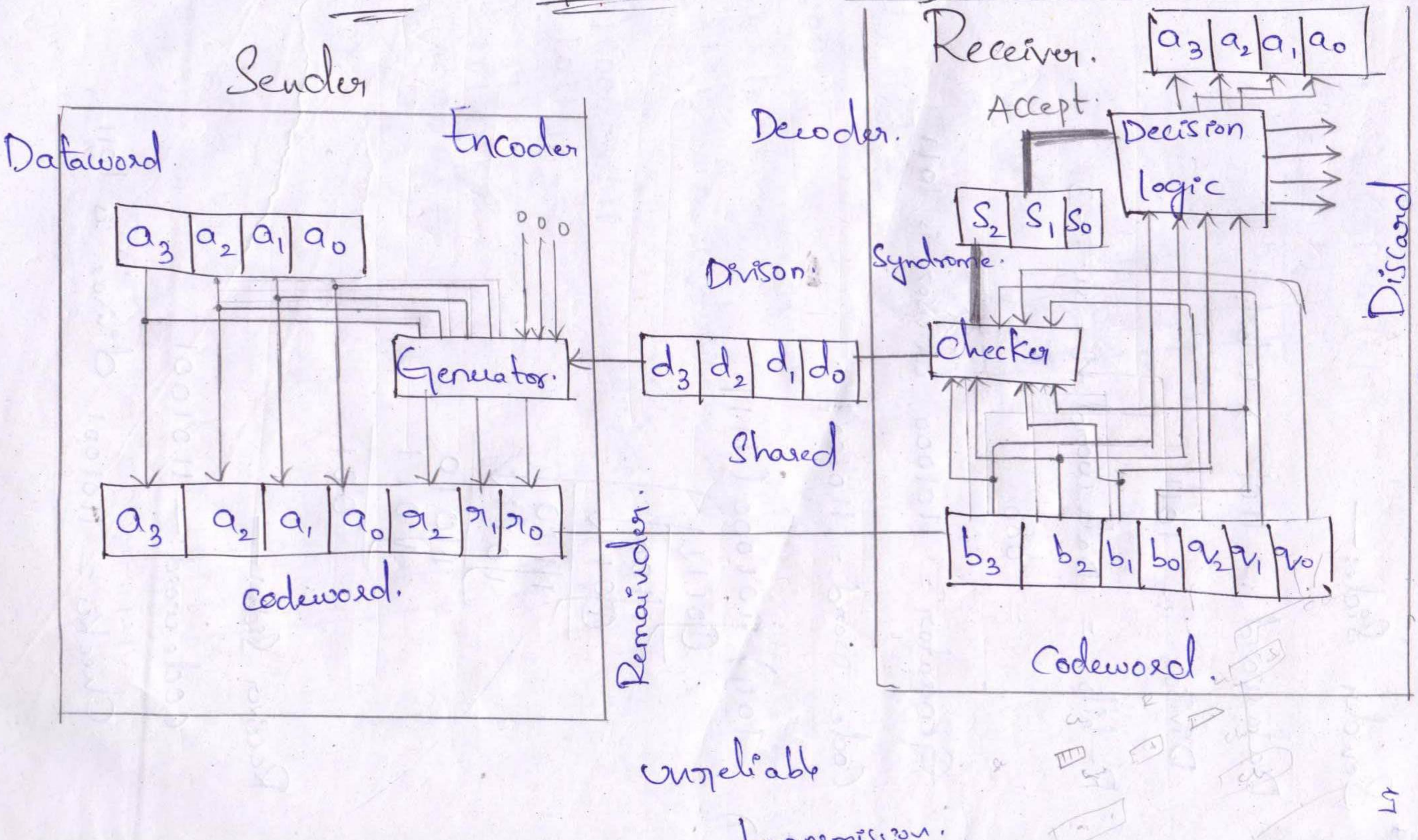
Vertical Redundancy Check (VRC):— It is also called parity check. Append a single bit at the end of the block such that the Number of one is even.

(CRC) Cyclic Refundancy ~~Check~~ :— Code.

11/7/18

* CRC is Cyclic Redundancy Code.
* Used to Identify errors.
* if n-bits is data word n-1 bits are redandancy bits are taken.
* Divisor is default given by the CRC developer
  ie.. 1011 for 4-bit data.

CRC Structure.



(B) Sender Side:—

Dataword :- 1101 → 4 bits

Divisor :- 1011

R-bits = Data word - 1 = 4 - 1 = 3

= 000

Generator = 1101000 divisor is 1011

Code word = 1101001

1) 1011) 1101000 (1111

1011↓

1100

1011↓

1110

1011↓

1010

1011↓

001

Receiver Side:—

code word:— 1101001

Checker = 1101001 divisor is 1011

```
1011) 1101001 (1111
      1011
      ----
       1100
       1011
       ----
        1110
        1011
        ----
         1011
         1011
         ----
          000
```

Dataword :-

0000

Divisor 1011

Generator = 0000000. divisor is 1011

```
1011) 0000000 (11
      1011
      ----
       1011
       1011
       ----
       00000
```

```
1011) 0001000 (1111111
      1011
      ----
      1010
      1011
      ----
       0010
       1011
       ----
       1001
       1011
       ----
       0100
       1011
       ----
       1111
       1011
       ----
       1000
       1011
       ----
        011
```

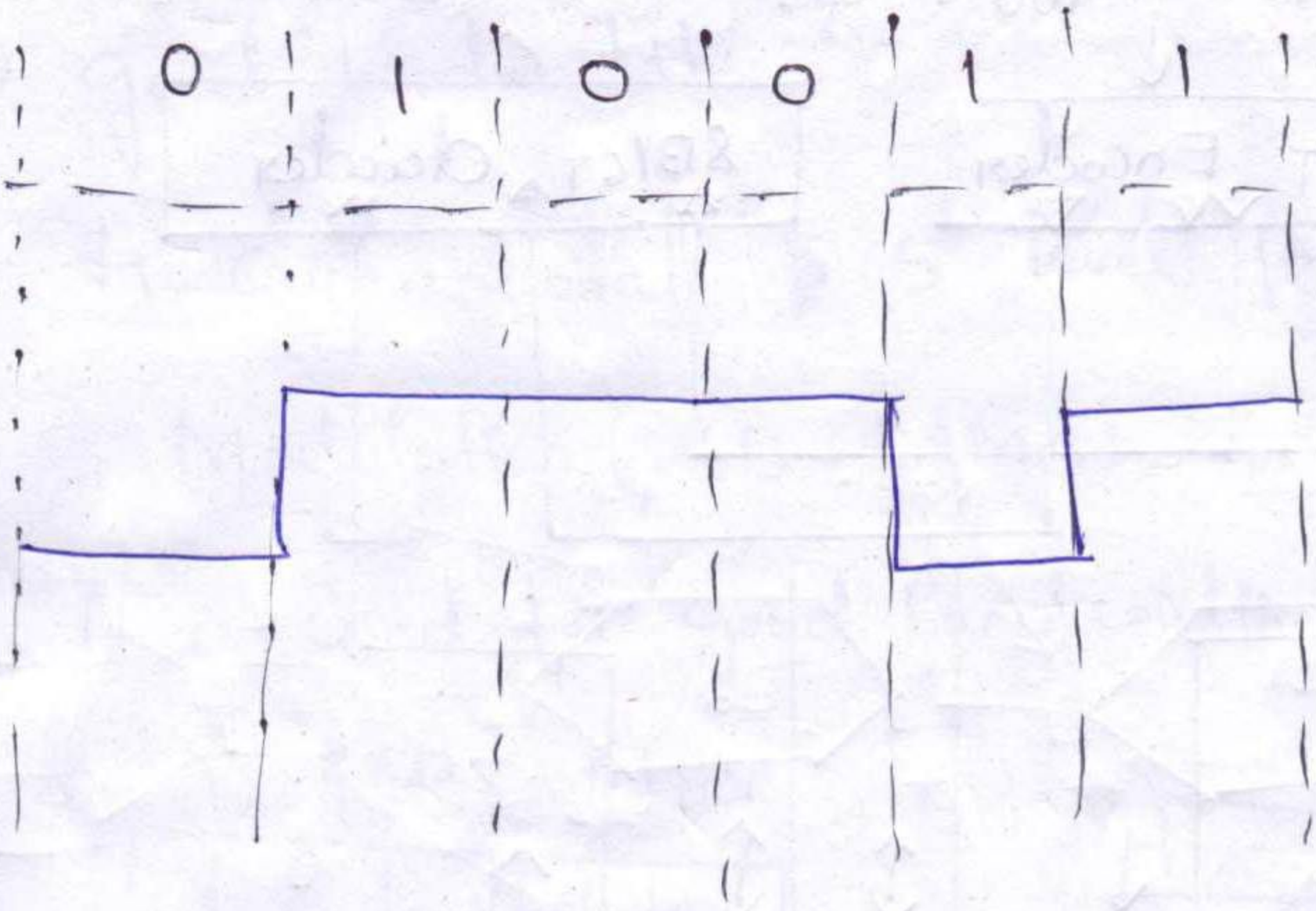
Dataword	Codeword	Dataword	Codeword
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

1011) 1111000 (1111

1011↓
1000
1011
0110
1011
1101
1011↓
1100
1011
111

1101) 8110001 (111111

1101
1011
1101↓
1100
1101↓
0010
1101
1111
1101
0101
1101
1000



4x25 mbps

4B/5B encoder

It is used for providing band width then the capacity will be high.

125 mbps

NRZ-I encoder

100 Base Fx →

4x25 Mbps

4B/5B decoder

125 mbps

NRZ-I decoder

[Non Return Zero]

Band width - High

↓

Capacity Low

4x25 mbps

4B/5B decoder

125 mbps

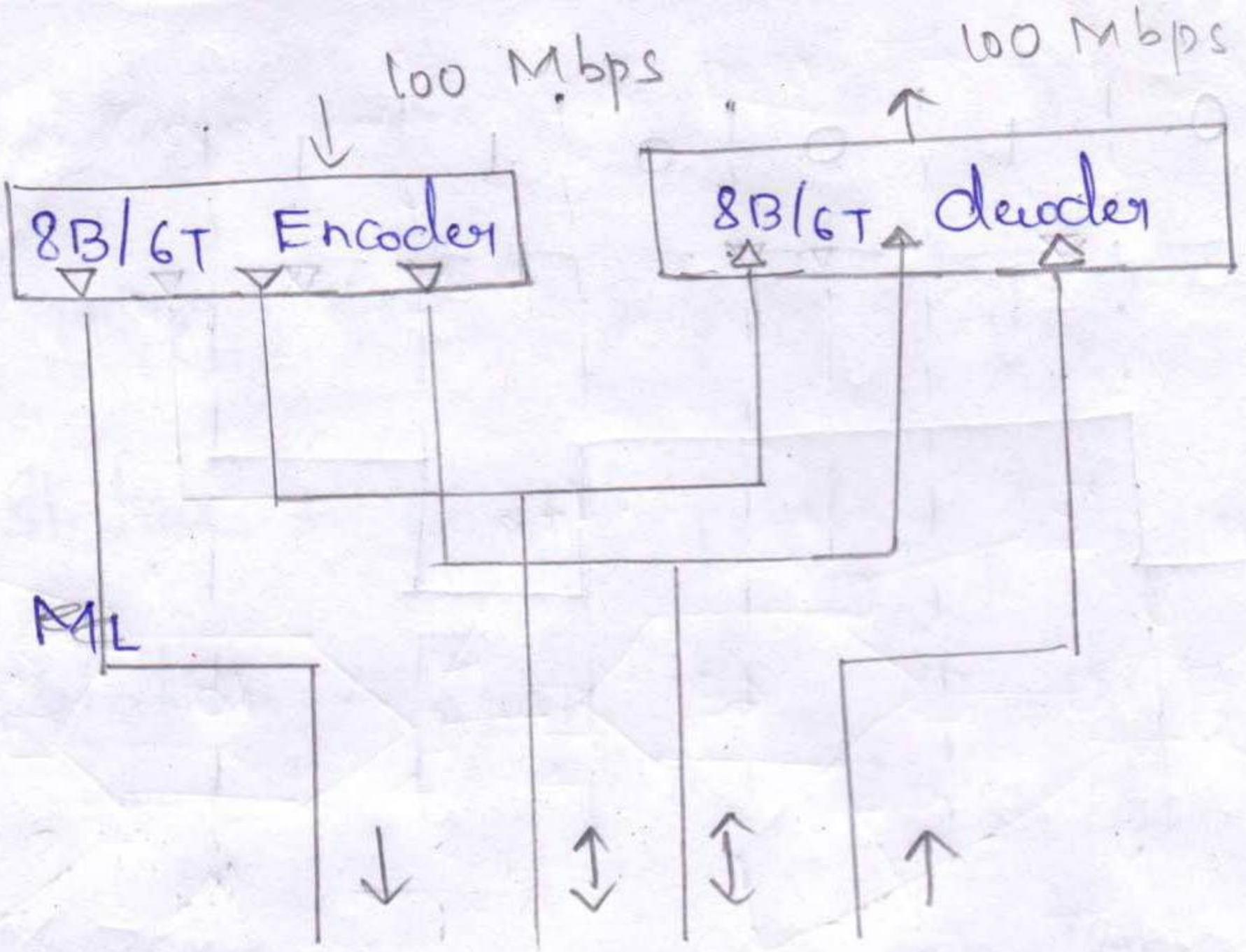
MLT-3 decoder.

It decodes the data that is sended by sender.

Band width - Low

↓

Capacity high.

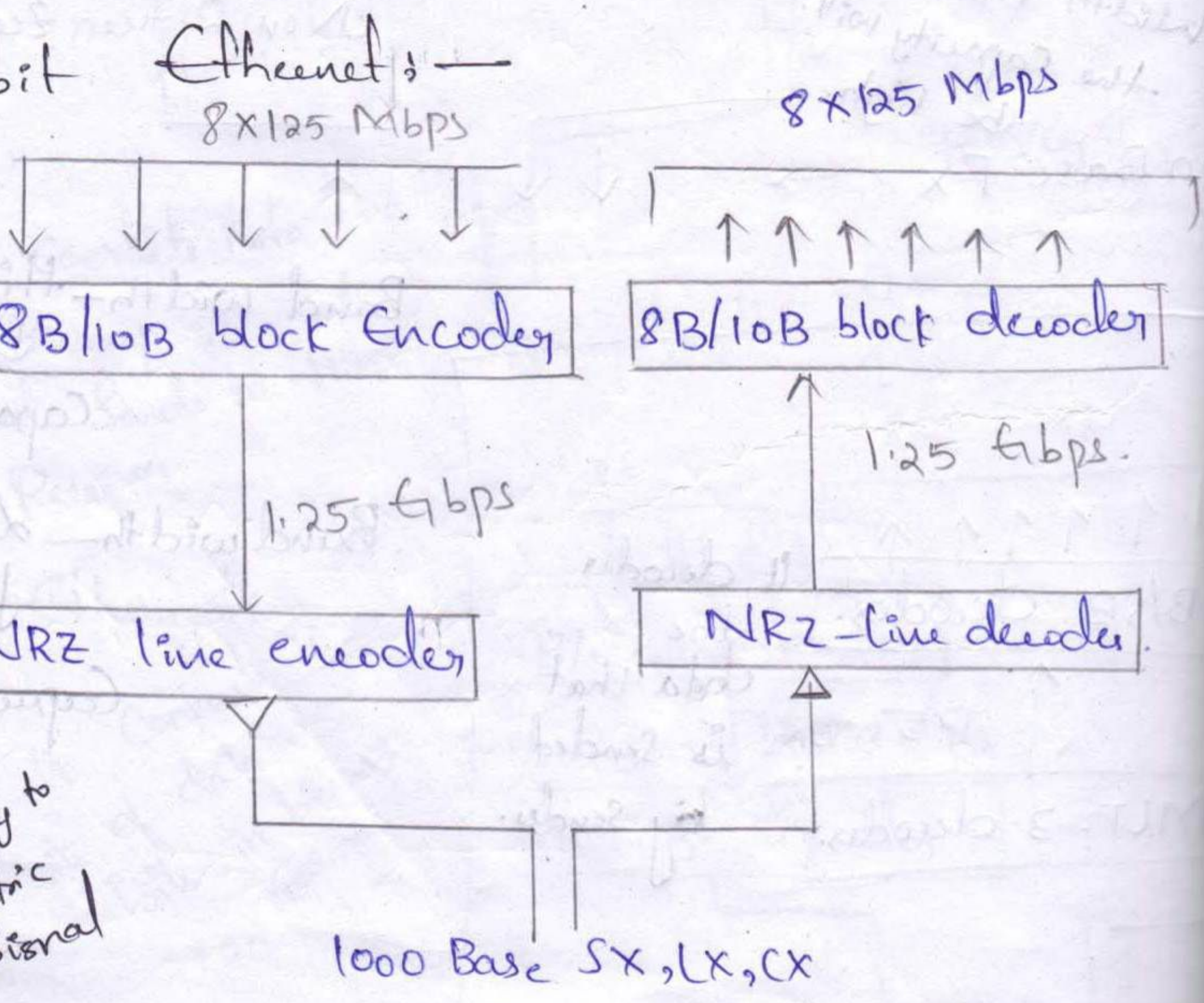


100 Base T4.

Single Wire used for error detection.

20/8/18.

Gigabit Ethernet :—

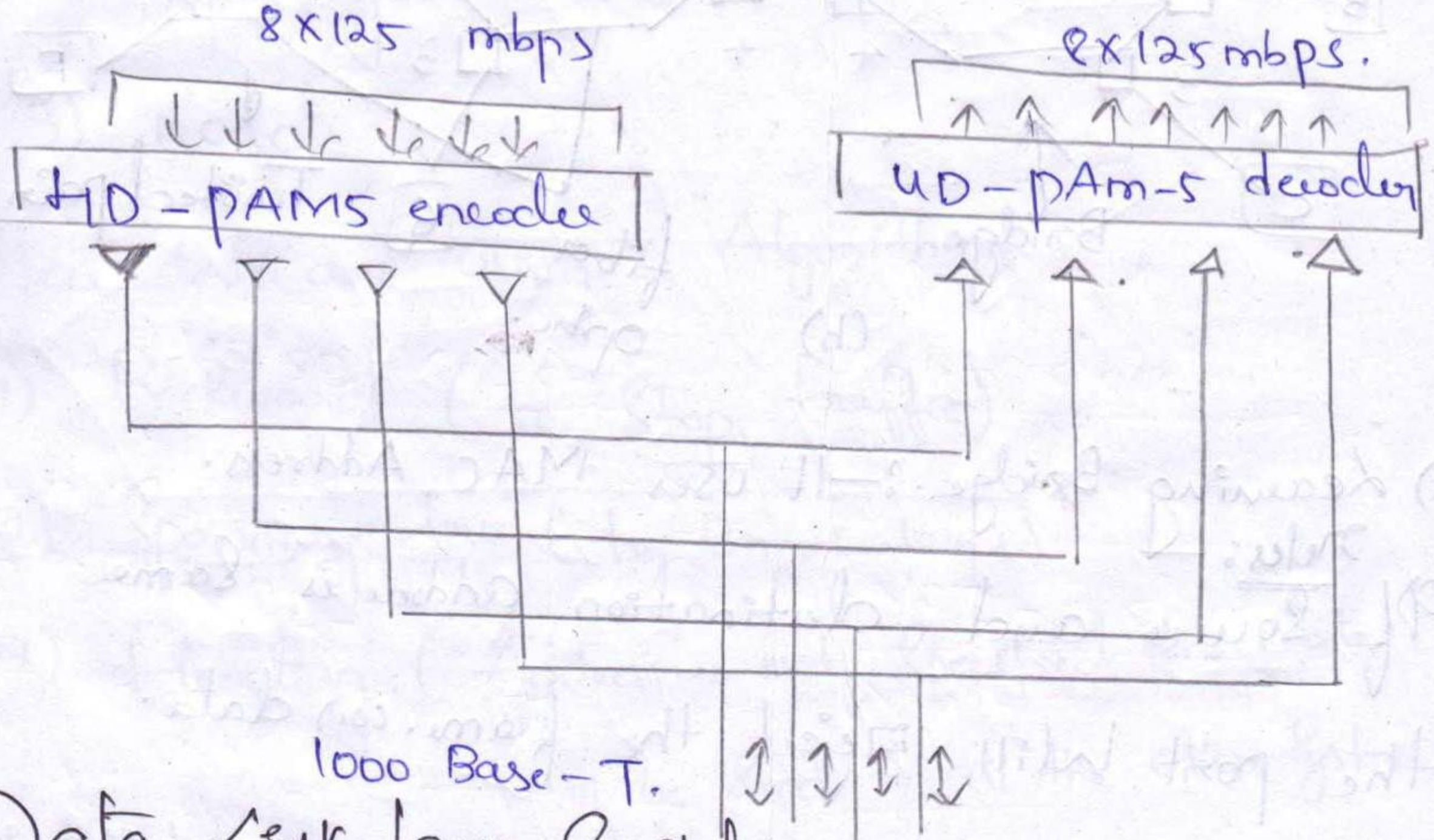


1) good bandwidth.

4 dimenstional 5 level pulse amplitude

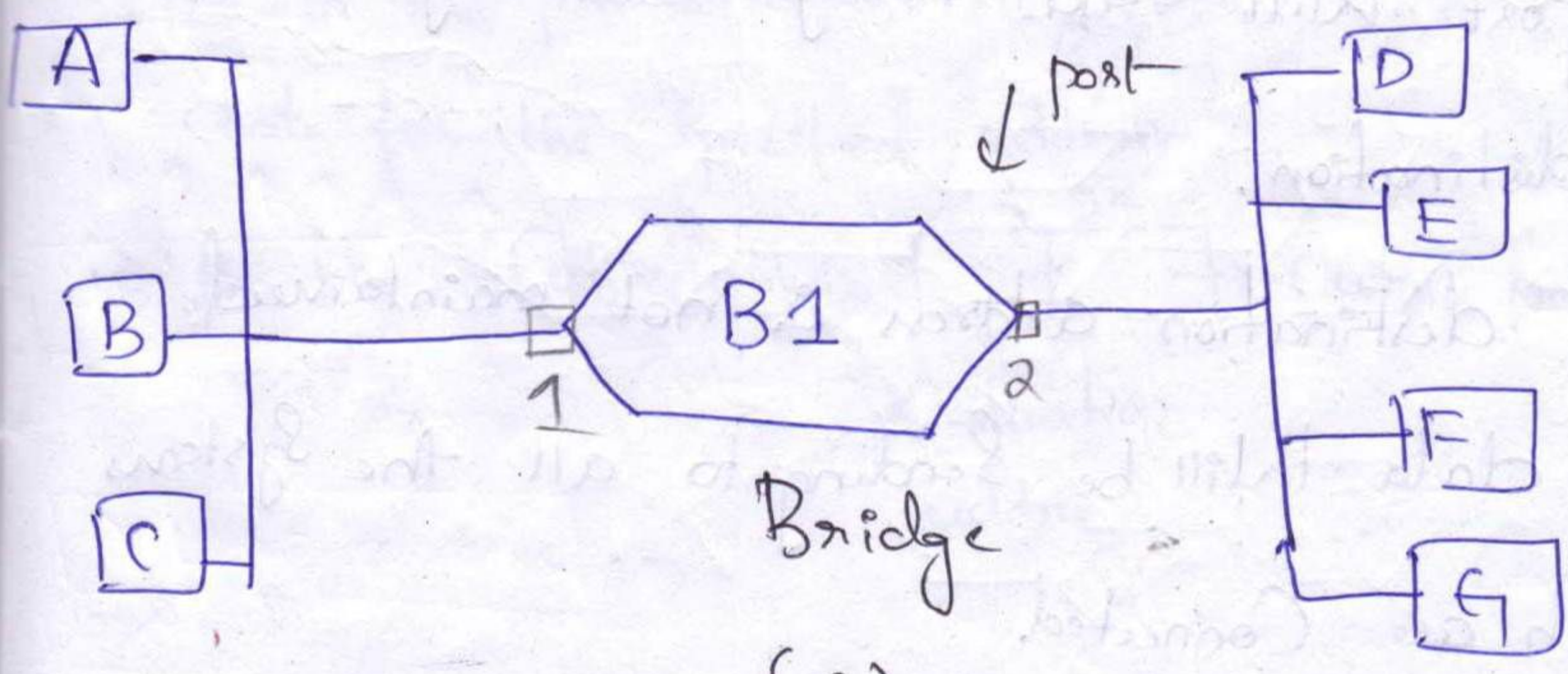
Modulation (4D-PAM5).

→ it is used for good bandwidth



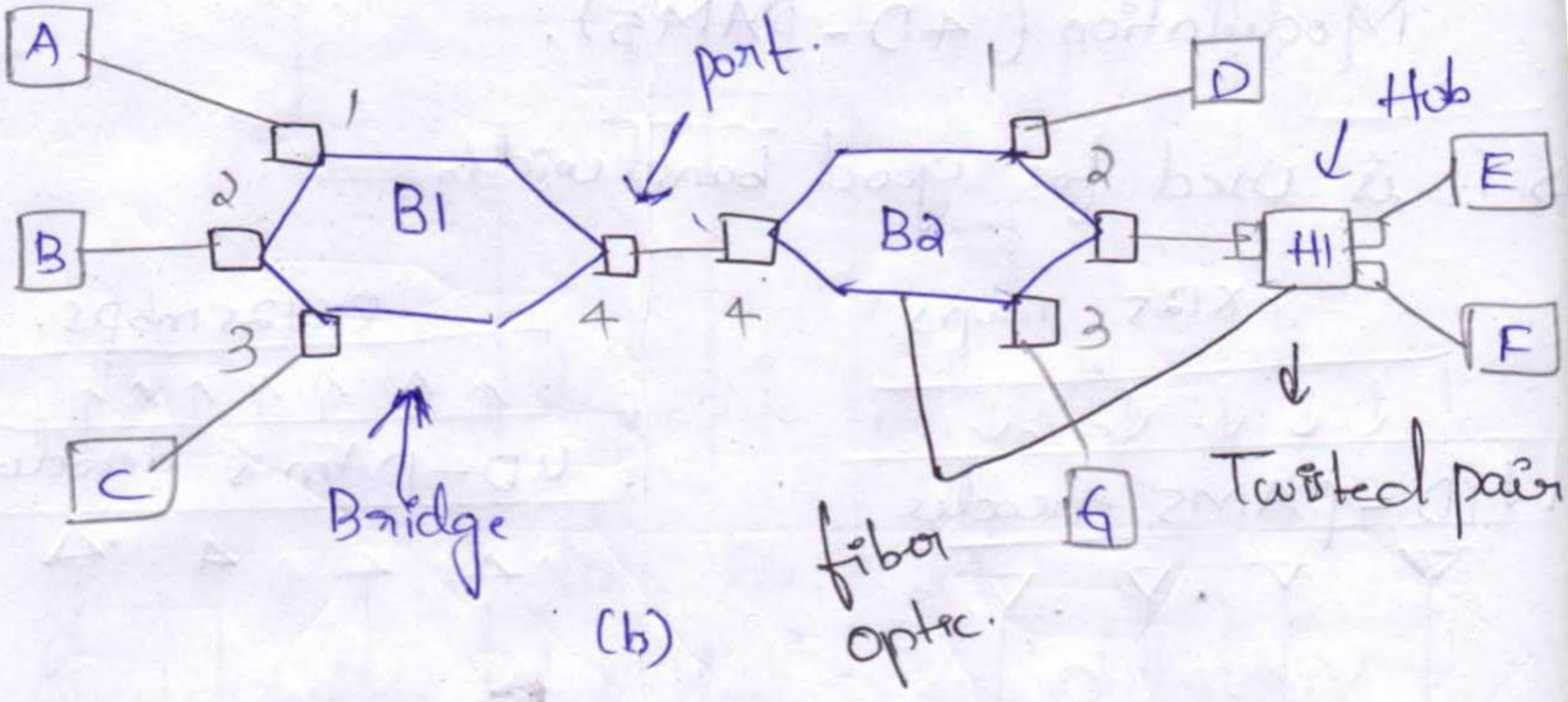
Data Link layer Switching:-

Bridge Connecting two Multidrop LAN's



(a)

Bridge Connecting seven- point -to- point



(b)

1) Learning Bridge :- It uses MAC Address.

Rules:

2) If Source and destination address is same the port will reject the frame. (or) data.

If Source and distination address is different the port will Send through the Bridge to the distination.

If the destination address is not maintained the data will be Sending to all the Systems Which are Connected.

Hub don't Need any Mac address to Send the frames.

21/8/18

1) Communication
2) Capacity.
3) Load

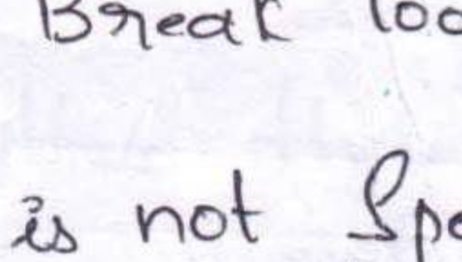
Here We are using 3 Algorithms.

1) Backward (to Stop traffic)
2) Spanning tree. (to Break loops).
3) Flooding. (Address is not Specified the data will be Send to all the other Systems)

Learning Bridge:— It uses flooding. While.

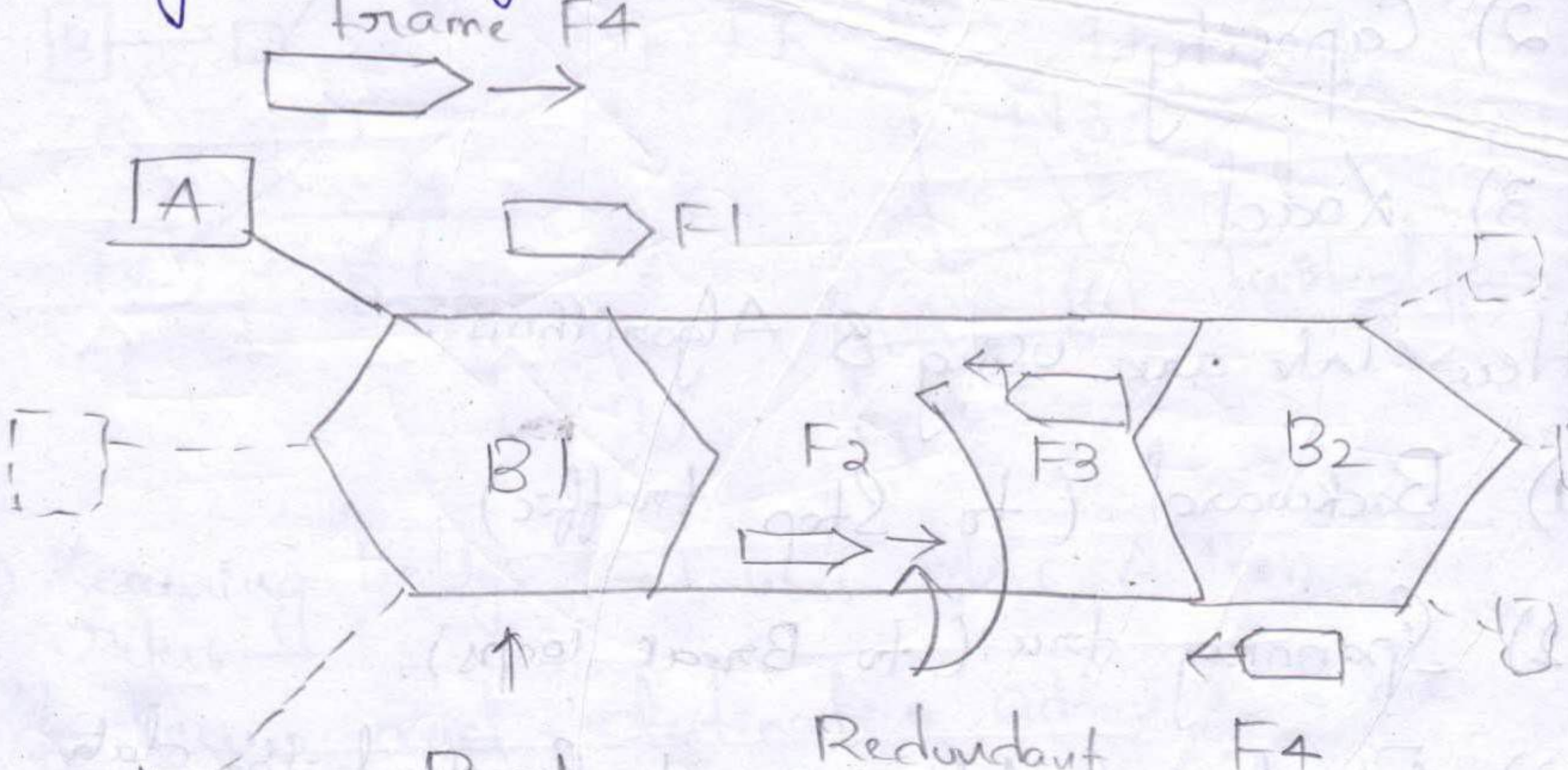
1) ~~If~~ in the process it will learn to Send the data for the Specified address System.

→ It has Hash table.

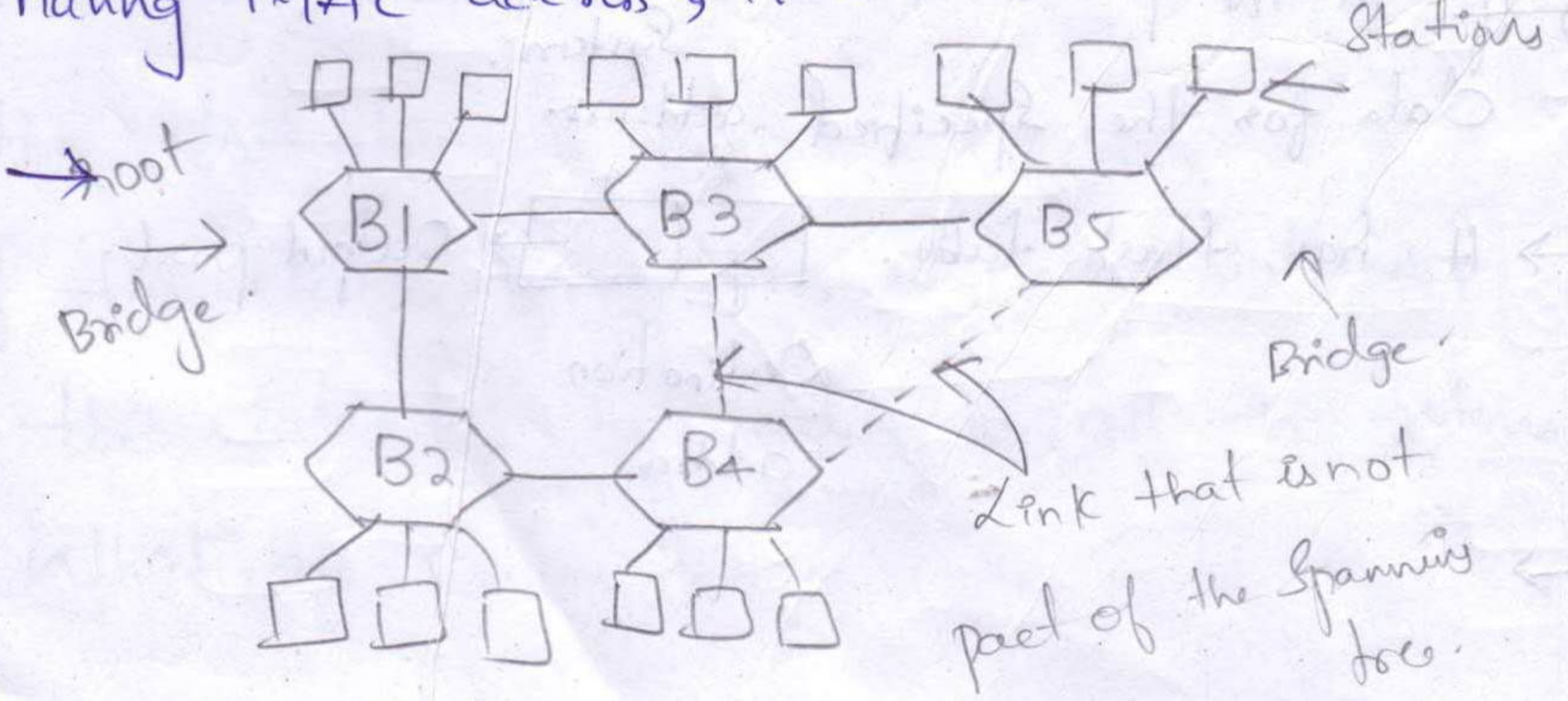


→

ub Spanning Tree Bridges:— The Duplicate Copy is Created When the Redundant Link is formed to avoid the redundant data we are using Spanning Tree.



A Spanning tree:— Each Bridge will be having MAC address, It is the Shortest path.



Repeaters:— They are used in Ring topology.

Hubs:— Hubs are used. in Star topology to Communicate between the Systems and act like. Intermediate Controller.

Bridges:— Connects the two or More LAN's. and uses the MAC addresses.

Routers:— Think logically and routing the data through Shorest path.

Switches:— Checks whether there is any error in Sowce data.

Gateway:— Connected between two different networks. Receive the data from the sender Interpret the data and send to the Receiver.

2) Dataword = 000

Divisor :- 1011

R bits = Dataword − 1

= 4 − 1

= 3 = 000.

Generator = 000 0000 divisor is 1011.

1011) 00000000 (0000
0000↓
0000
0000↓
0000
0000↓
0000
0000

4) D.W = 0010

Div = 1011

G = 0010000

C.W = 0010110.

1011) 0010000 (0010
0000↓
0100
0000↓
1000
1011
0110
0000
0110
0000
110

3) D.W 0001

Div 1011

G = 0001000.

C.D = 0001011

1011) 0001000 (0001
0000↓
0010
0000↓
0100
0000
1000
1011
011

5) Dataword = 0011

Divisor = 1011

Generator : 0011000

Codeword = 0011101

1011) 0011000 (0011
0000↓
0110
0000↓
1100
1011↓
0110
1011
101.

6) D·w = 0100

Divisor = 1011

G = 0100000

C·w = 0100111

```
1011) 0100000 (0101
      0000↓
      ------
       1000
       1011↓
       ----
        0110
        0000
        ----
         1100
         1011
         ----
          111
```

8) Data word : 0110

Divisor : 1011

Generator : 0110000

Codeword : 0110001

```
1011) 0110000 (0111
      0000↓
      ------
       1100
       1011↓
       ----
        1110
        1011
        ----
         1010
         1011
         ----
          001
```

10) Data word : 1000

Division : 1011

Generator : 1000000

codeword :- 1000101.

7) D·w = 0101

Divisor = 1011

G = 0101000

C·w = 0101100

```
1011) 0101000 (0100
      0000↓
      ------
       1010
       1011↓
       ----
        0010
        0000
        ----
         0100
         0000
         ----
          100
```

9) Data word = 0111

Division = 1011

Generator : 0111000

Codeword : 0111010

```
1011) 0111000 (0110
      0000↓
      ------
       1110
       1011↓
       ----
        1010
        1011
        ----
         0010
         0000
         ----
          010
```

```
1011) 1000000 (1011
      1011↓
      ------
       0110
       0000
       ----
        1100
        1011
        ----
         1110
         1011
         ----
          101
```

11) Datacoord: 1001

Divisor: 1011

Generator: 1001000

Codeword: 1001110

```
1011) 1001000 (1010
      1011
      ----
       0100
       0000
       ----
        1000
        1011
        ----
         0110
         0000
         ----
          110
```

12) Datacoord: 1010

Divisor: - 1011

Generator: - 1010000

Codeword: - 1010011

```
1011)1010000 (1001
     1011
     ----
      0010
      0000
      ----
       0100
       0000
       ----
        1000
        1011
        ----
         011
```

13) Datacword: - 1011

Divisor: - 1011

Generator: - 1011000

Codeword: - 1011000

```
1011)1011000 (1000
     1011
     ----
      0000
      0000
      ----
       0000
       0000
       ----
        000.
```

14) Datacword: - 1100

Divisior: - 1011

Generator: - 1100000

Codeword: - 1100010

```
1011) 1100000 (1110
      1011
      ----
       1110
       1011
       ----
        1010
        1011
        ----
         0010
         0000
         ----
          010.
```

15) Dataword: - 1110

Divisor: - 1011

Generator: - 1110000

Codeword: - 1110100

Dataword - 1111

Divisior: - 1011

Generator: - 1111000

Codeword: - 1111111

24

```
1011) 1110000 (1110.0
      1011 ↓
      ------
       1010
       1011 ↓
       ------
        0010
        0000 ↓
        ------
         0100
         0000
         ----
          100.
```

```
1011) 1111000 (1101
      1011 ↓
      ------
       1000
       1011 ↓
       ------
        0110
        0000 ↓
        ------
         1100
         1011
         ----
          111.
```

16/7/18.

Polynomial Cyclic Redundancy Code :—

Eg:— $2x^3+x+1$. — 1011.

Given polynomial Expression will be Converted into Binary Code.

3 2 1 0

1 0 1 1 — Binary Code.

2Eg:— Data word :- x^3+x^2+x+1

Divisor :- x^2+1

3 2 1 0 — 2 1 0

1 1 1 1 — 1 0 1

101) 1111000 (Quotient = 1111000.

Codeword —

```
101) 11111000 (11000
     101↓
     ----
      101
      101↓
      ----
       000
       000↓
       ----
        000
        000↓
        ----
         000.
         000
         ---
          00.
```

Syndrome :- 000.

Eg:- $f(x) = x^7 + x^5 + x^2 + x + 1$

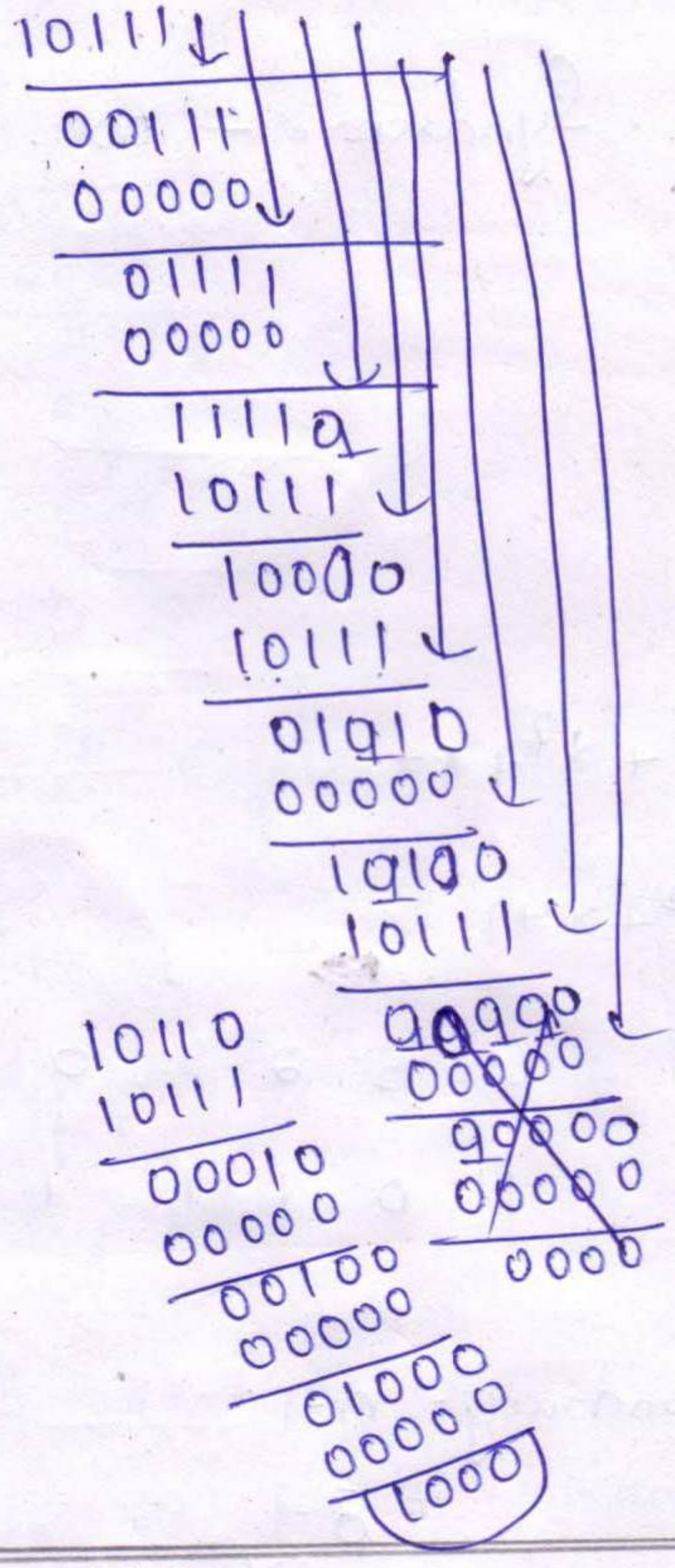
$g(x) = x^4 + x^2 + x + 1.$

7	6	5	4	3	2	1	0
1	0	1	0	0	1	1	1

4	3	2	1	0
1	0	1	1	1

```
10111) 10100111 (Redundancy = n-1
                          = 8-1
                          = 7
```

10111) 10100111000000000(1001101100



Syndrome :- 1000

Name	polynomial	used in.
CRC-8	x^8+x^2+x+1 100000111	ATM header
CRC-10	$x^{10}+\ldots$	ATM AAL
CRC-16	$x^{16}+x^{12}+x^5+1$ ..	HDLC
CRC-32	$x^{32}+x^{26}+\ldots+x+1$	LAN's

ATM - Asynchronous Transfer mode.

AAL - Asynchronous Adaption Layer.

HDLC - High level Data Link Control.

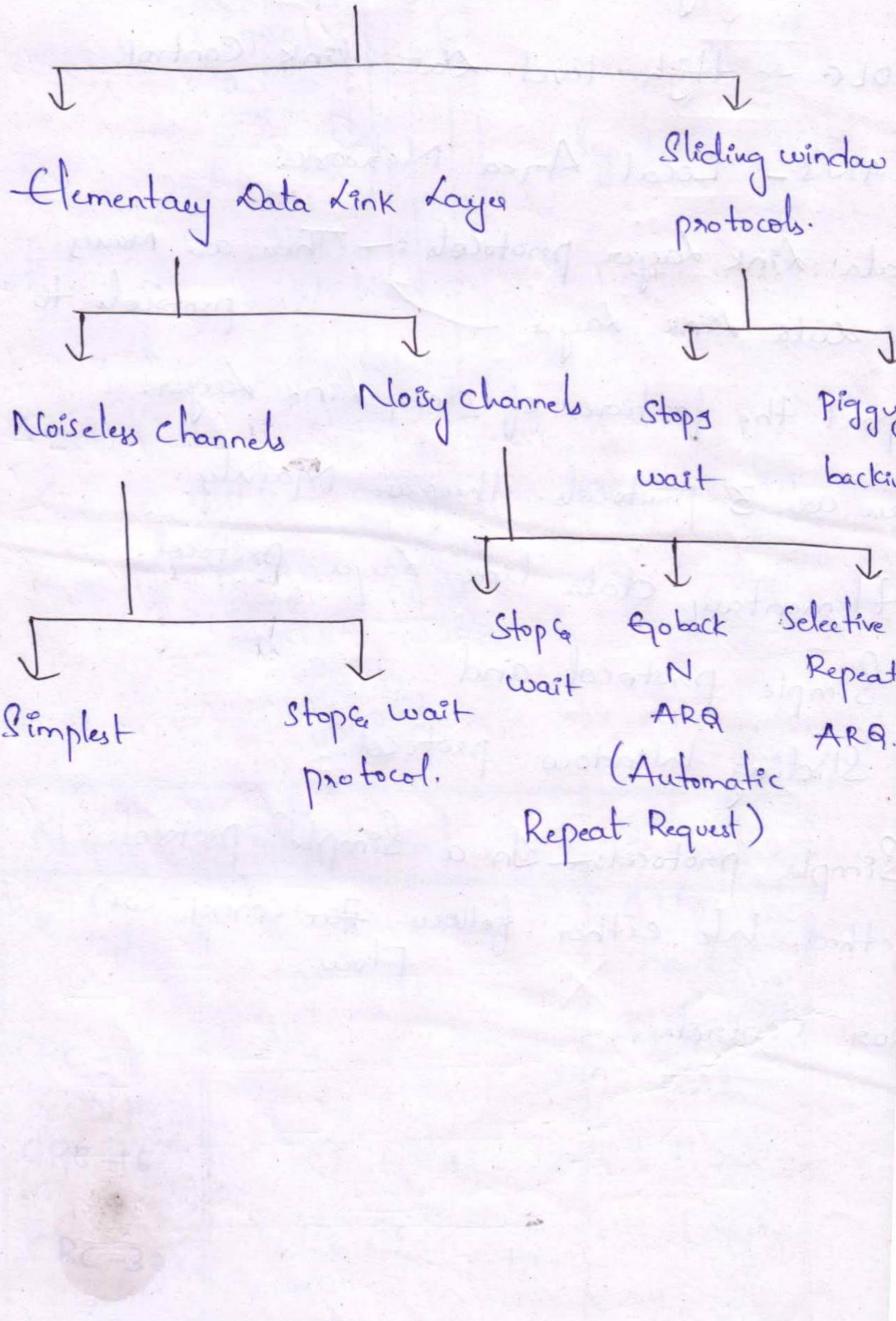
LAN's - Local Area Networks.

Data Link Layer protocols :- There are Many protocols. to In Data Link Layer. ↗ Support the features of data Link Layer. There are 3 protocols. they are Mainly.

1) Elementary data link Layer protocol.
2) Simple protocol. and
3) Sliding Window protocol.

Simple protocole - In a simple protocol. Method. We either follow ~~flow~~ control (or) Flow error Control.

17/7/18. Protocol in Data Link Layer.



18/7/18.

UNIT- II (Wireless physical & Medium access protocol. Data link Layer).

(Implemented in TCPIP)

- Random access protocol (First) (Implement wireless)
  - ALOHA protocol
    - Pure ALOHA
    - Slotted ALOHA
  - CSMA
  - CSMA/CD
  - CSMA/CA
- Controlled access protocol (Authorization purpose).
  - Reservation
  - Polling
  - Token passing
- Channelization protocol (Routers providing mobiles In)
  - FDMA
  - TDMA
  - CDMA
  - SDMA

* Carrier Sense Multiple access - CSMA
* Collision detection - CD/CSMA
* Collision avoidance - CA/CSMA
* Frequency Division Multiple access - FDMA
* Time Division Multiple access - TDMA
* Code Division Multiple acces. - CDMA

* Space division Multiple access — SDMA.

Random access protocol:— It is a Method used to transmit randomly among the Stations (or) channels. In Random access there are Many protocols which are used to Implemented a flow Control Mechanism based on Some of the protocols;

ALOHA:— It is the First wireless protocol.

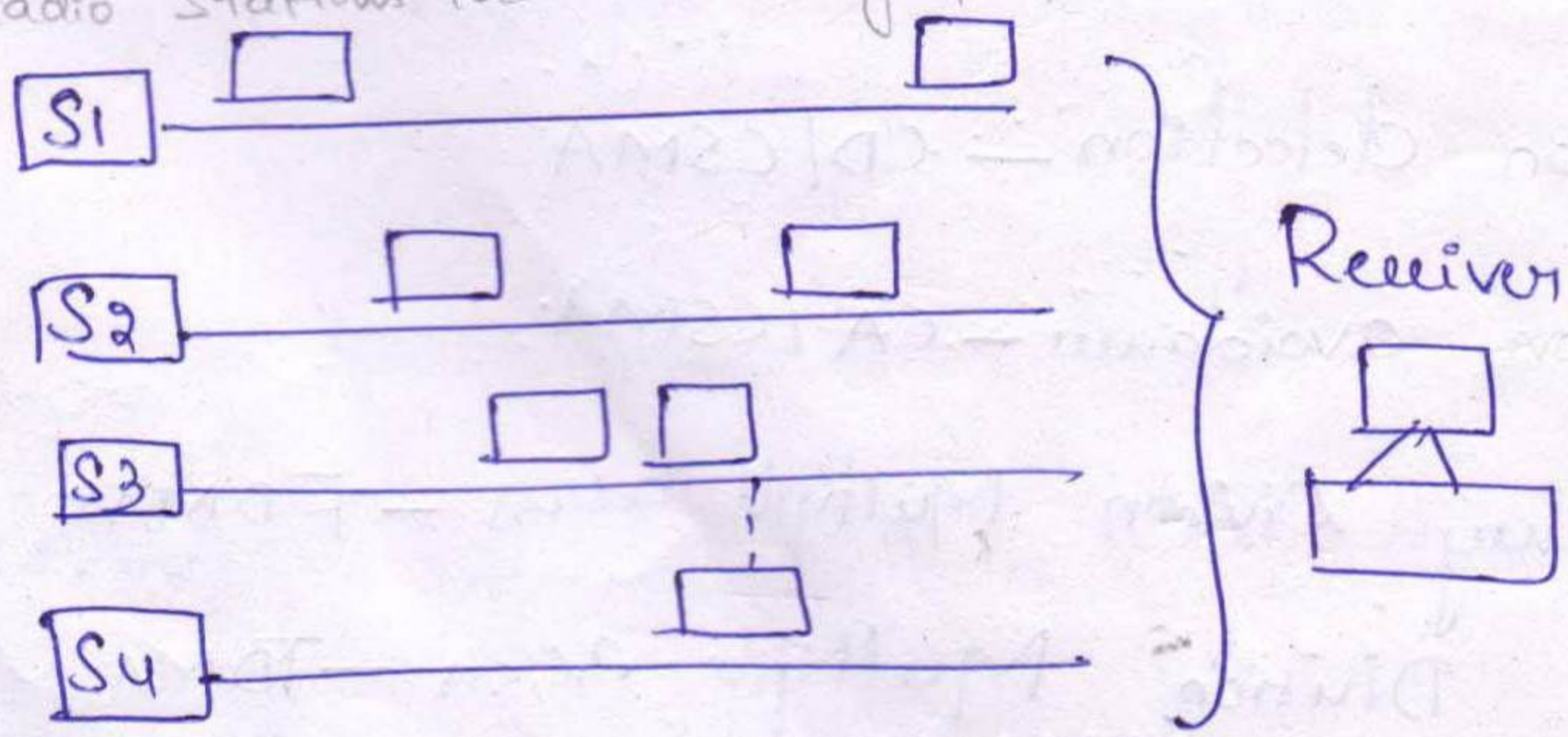
i) developed at the University of HAWAII in the Early 1970's. It is designed for radio and wireless ~~lan~~ LAN. It is divided into two types.

a) Pure ALOHA (Collision occurs).

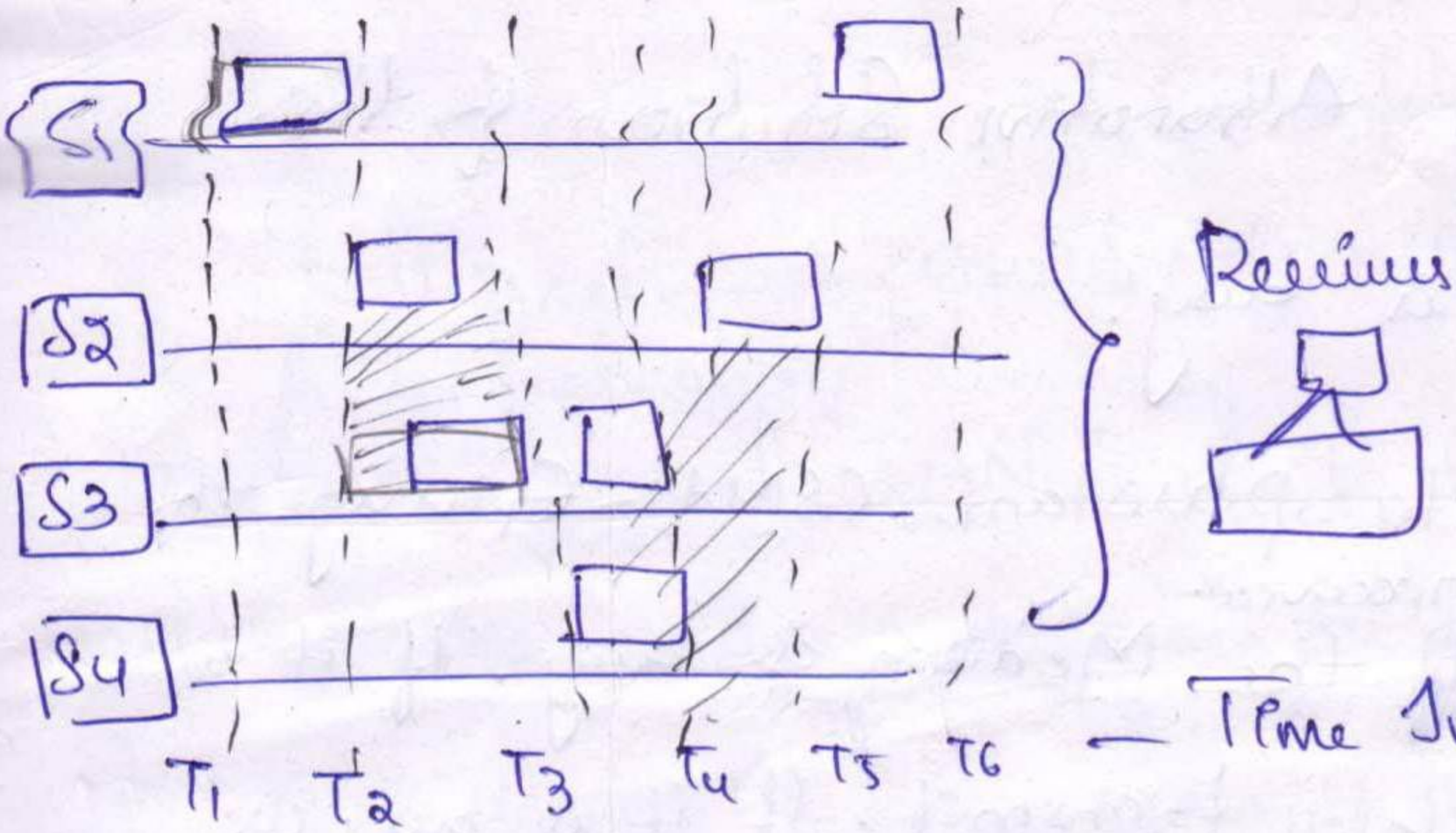
b) Slotted ALOHA (equal time Intervals are divided)

Pure ALOHA:—
(Radio Stations location choosing purpose).

Eg:-



Slotted ALOHA:—



20/8/18

Carrier Sense Multiple Access. — CSMA.

→ It is known as "Listen Before Talk".

→ It determines whether the Station is ready to listen or not.

→ 3 types of CSMA.

1) Non-persistant CSMA
2) 1-persistant CSMA
3) p-persistant CSMA.

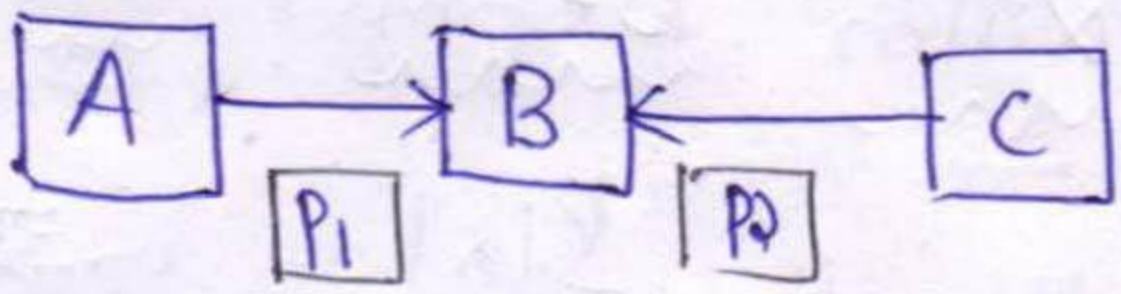
Non-persistant :— 1) Medium is idle, then transmits.
2) If Medium is busy, then try again.

1-persistant:—

Choosing Alternative Solutions If the Medium is busy.

Probability persistant CSMA:- Giving the priority to receiver. If the Medium is busy. If it is free he will transmit, If it is busy we will wait. it is based on probability.

CSMA Collision Detection:—



When two packets collide the Medium remains unstable for the duration of transmission of both damaged packets. This leads to wastage of time and bandwidth.

This Wastage is reduced by Quickly terminating transmission upon Collision

Collision Avoidance:— It is difficult to detect Collision.

→ Control of radio environment is difficult

3) Hidden Station Problem.

10/8/18

Network Interphase

Ethernet:— It Consists of NIC card.

It is a technology Commonly.

1) Used in "Wired LAN" used

2) It is the Network protocols to control the LAN System we are using Ethernet and How data is transmitted. Once LAN

3) Technical is named as "IEEE 802.3" protocol.

4) It is developed in 1973.

The Ethernet is divided into four types.

1) Standard

2) Fast

3) Giga bit

4) Ten Giga bit.

Standard Ethernet:—

MAC Sub layer:- Multiple access control.

It Controls the data. It converts the data into frame format.

Frame format:—

Preamble	SFD	Destination Address	Source Address	length or type	Data or padding	CF
7 bytes	1 byte	6 bytes	6 bytes	2 bytes	46 bytes	4 by

Preamble:— It will give the alert to the receiver. and warn to Synchronize the Data an

SFD:— Start frame Delimiter. proceed it

It Consists of 10101011 format. Immediat

'11' destination— Address.

Destination Address — It Consists of the address of the frame (or) Receives address.

Source Address. :— Sender address.

length (or) type:— Unity Simple file transfer protocol and the data length is measured

The Minimum length of the frame should be.
64 bytes (Correct operation of CSMA/CD)
The Maximum length of the frame should be 1518 bytes. (Reduce buffering).

Addressing :—

Hexa decimal format

01 : 03 : 05 : 0A . 0C . Card.

In Ethernet it Consists of Network Interface NIC generates physical address. which is in the form of Hexa decimal format.

1) Unicast - one-to-one. and having one Sender and one Receiver. (1)

2) Multicast - One Sender and Multiple Receiver. (1)

It Sends the data who wants (for a particular person).

Ethernet :- It is Standard way to Connect the Network over wired Connection.

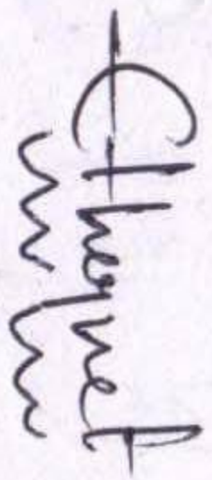
History of Ethernet :- developed by "Robert Met Calfe" in the year of 1973 at "Xerox palo alto Research Center California" used ideaof ALOHA Network.

* Develop CSMA/CD. algorithm
* He also developed more Sofastigated back of time that allow Ethernet to function More beffer.

Back of time :- The Medium is busy then the Station Should wait for a Certain amount of time.

Ethernet.

- Standard (10 Mbps).
  - → 10 Base 5
    Bus, thick Coaxial
  - → 10 Base 2
    Bus, thin Coaxial
  - → 10 Base T
    Star, Twisted
  - → 10 Base F
    Star, Fiber.
- Fast
  - → 100 Base Tx
    Two Wires
    Category 5UTP
  - → 100 Base Fx
    Two wires fiber
  - → 100 Base T4
    Four Wires Category 3 UTP.
- Gigabit
  - → 1000 Base Sx
  - → 1000 Base Lx
  - → 1000 Base CX
  - → 1000 Base T.
- Ten Gigabet



Based on Speed of transmission, cables are divided into

* Cat 3 - 10 Mbps

  Cat 5 - 100 Mbps

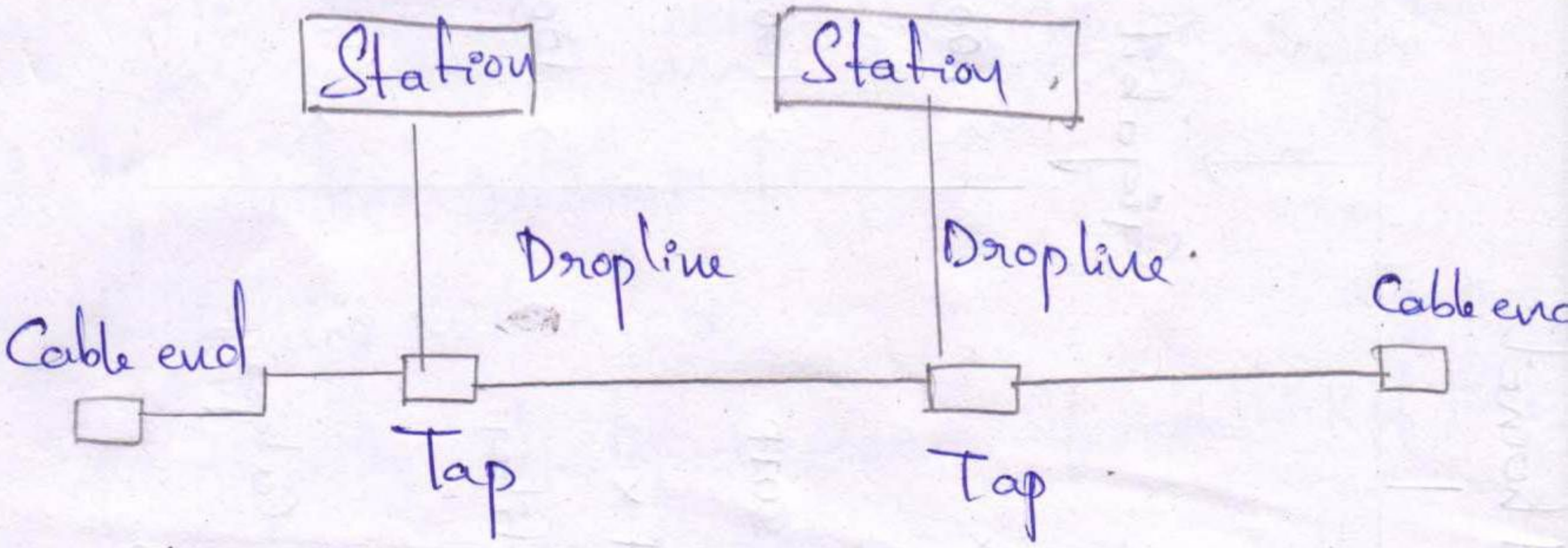
  Cat 5e - 1000 mbps

  Cat 6 - 10,000 mbps.

Cat - Category

Mbps - Mega bits per Second.

Bus topology :—



Drop line :— Connection b/w device & Main Cable.

Tap :— treated as Connector that Connects drop line with Central Cable.

10 Bw 5 :—

→ 10 : Transmission Speed of 10 Mbps

→ Base : Base band Signaling.

→ 5 : Maximum Segment length of 500 meters.

→ use bus topology with External transceiver Connected via a tap to thick coaxial Cable.

Mbps - Mega bits per Second (Speed of dowload + upload).

MBps - Mega Bytes per Second (file size, amount of data transfered).

16/8/18 Giga bit Ethernet:—

1) 1000 Base SX :- range is 550m.
2) 1000 Base CX
3) 1000 Base LX
4) 1000 Base T.

1000 Base SX:— It is used in Multi-Mode. It is paralley transmitts the data.

→ NIR

1000 Base CX:— Its range is very short 25m. D9 (or) 8P8C. Connector in 1000 Base Cx.

1000 Base LX:— Long wave length and the range is 5000m.

1000 Base T:— It is using Cat 5 ~~Shielded~~ Sheilded. Twisted pair and range is 100m.

uo

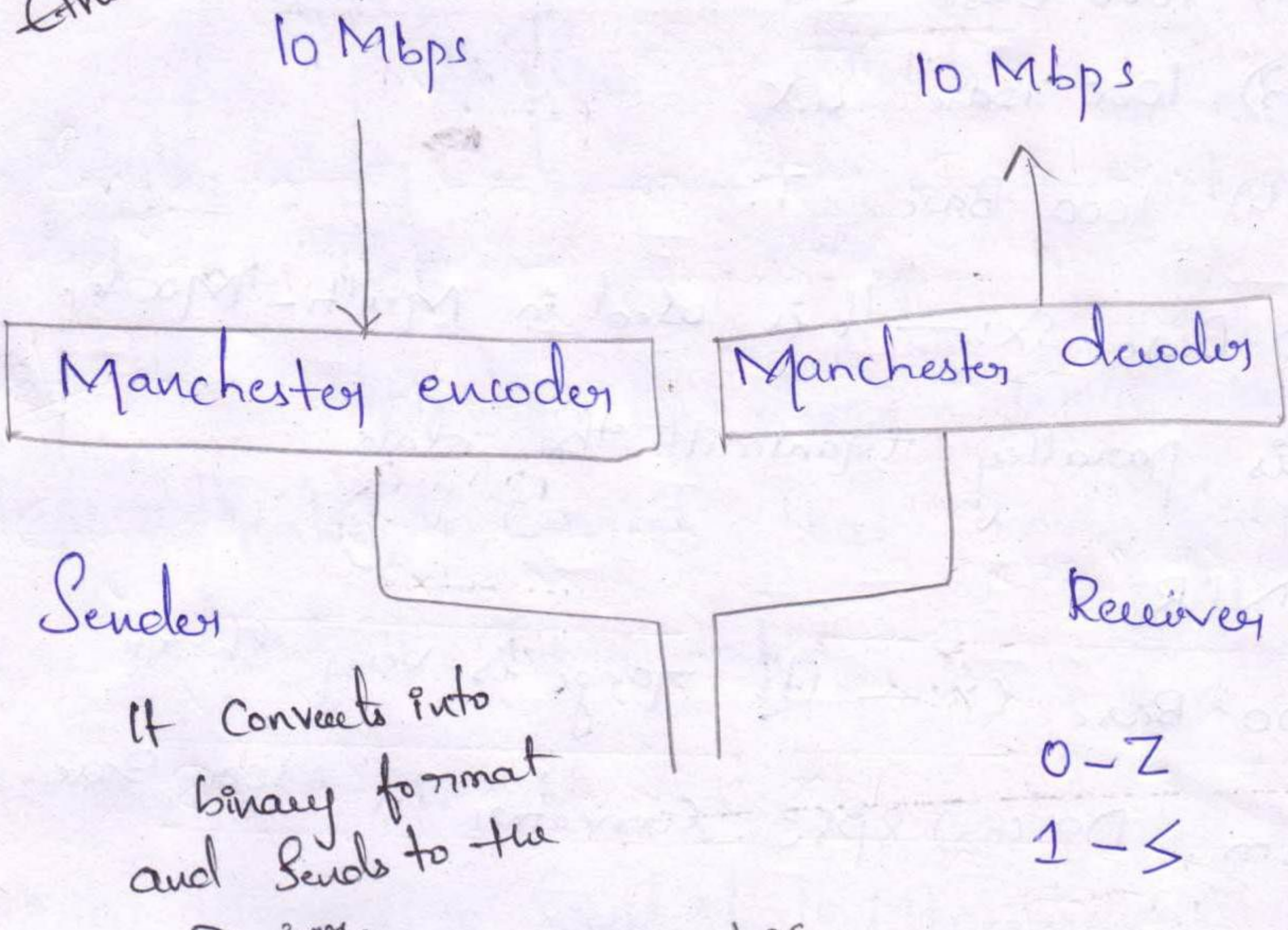
physical Layer:—

The topology used in physical Layer are.

2 Stations — point. to point.

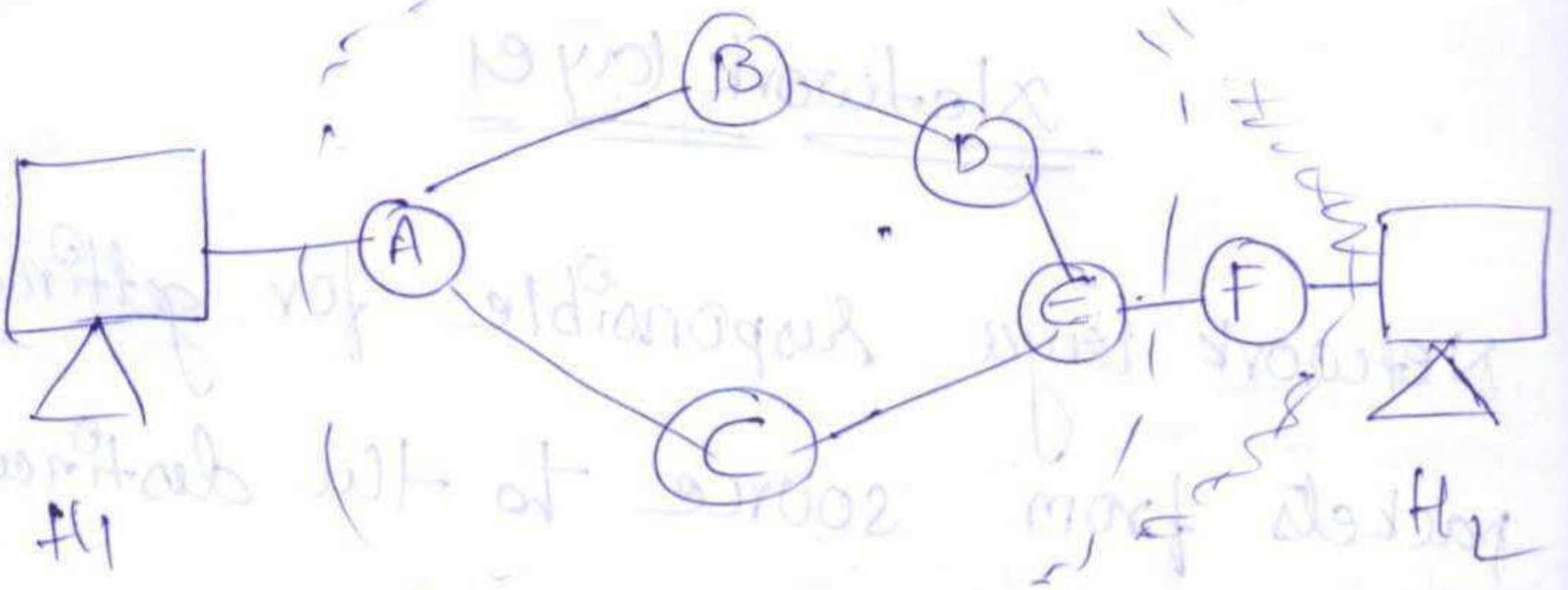
multy 3 or more Station — Star

Standard Ethernet



Fast Ethernet:—

It Converts the 4 bits of data to 5 bits in binary format.



ISP → Internet service provider

* Store and forward packet switching :-

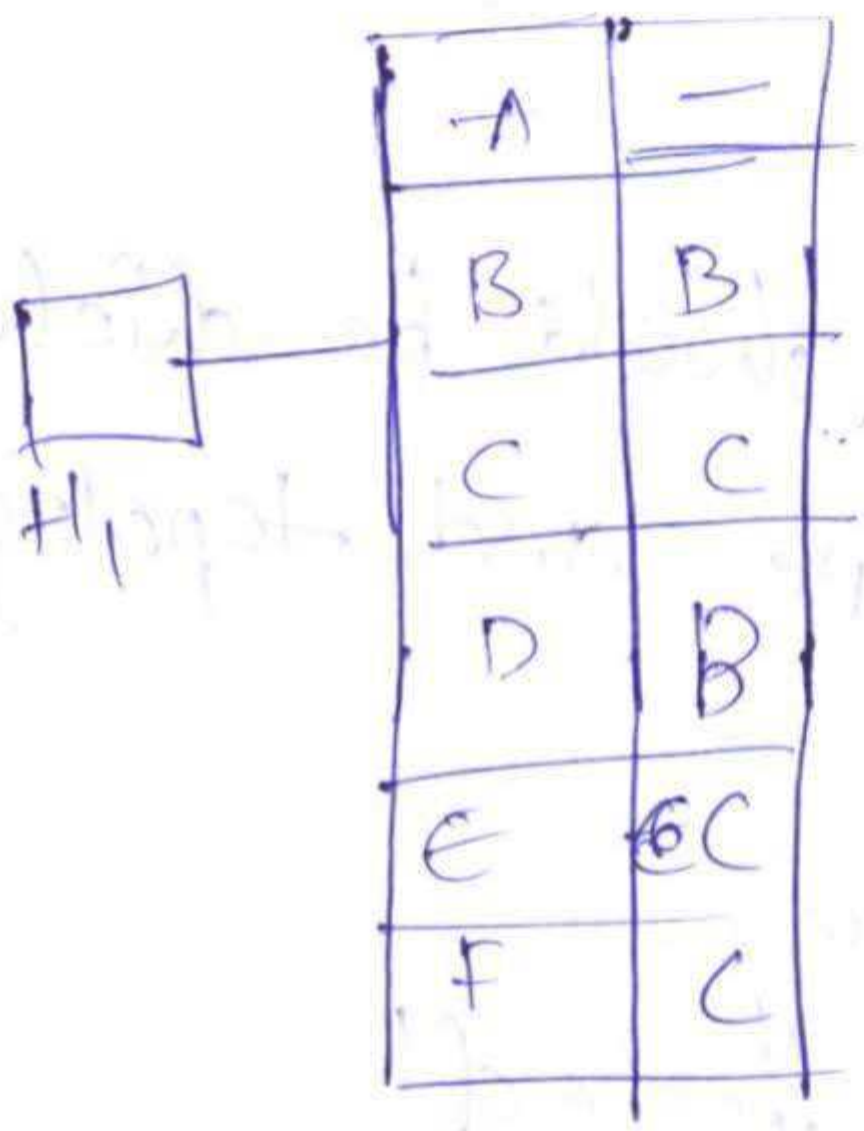
* Here H1 wants to send a packet to the nearest routers and forwards the packet to the next upcoming routers

* The packet is firstly stored and then forwarded from H1 until it reaches the destination and gets it "checksome verified"

3) Implementation of connectionless Network:-

A's table initial

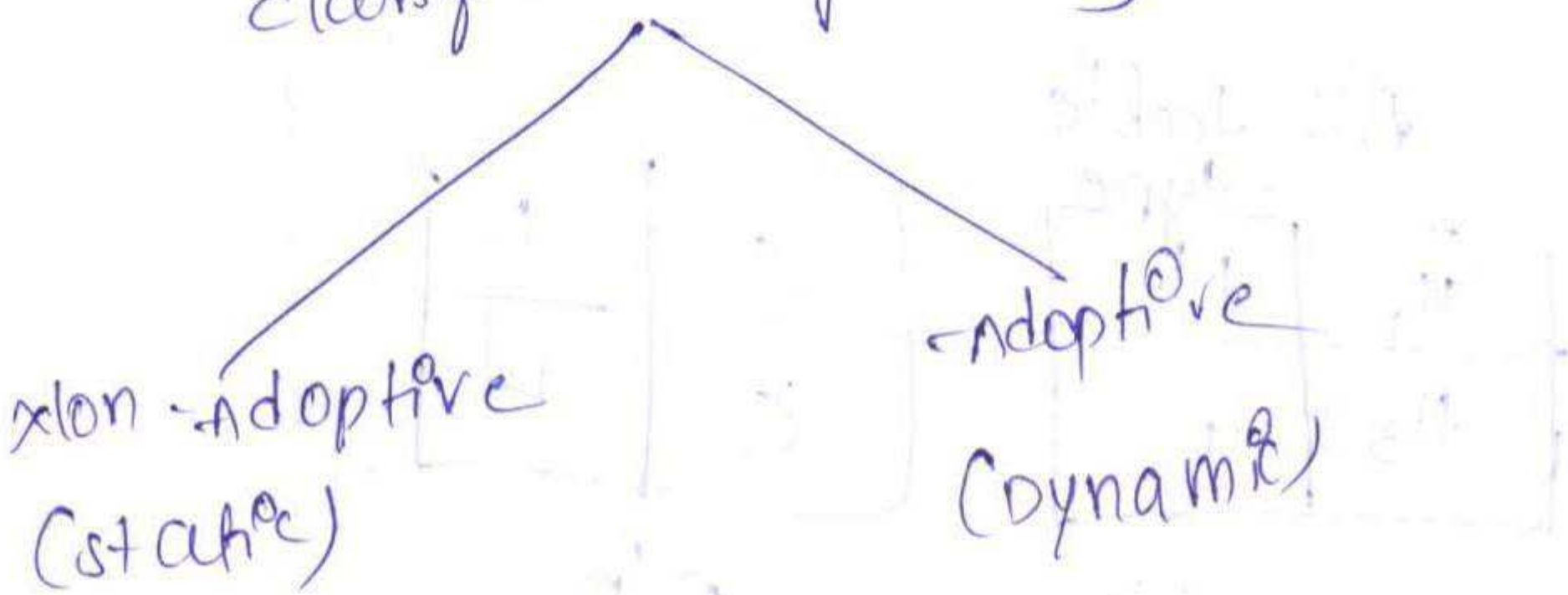
A	—
B	B
C	C
D	B
E	C
F	C



A	—
B	B
C	C
D	C
E	B
F	B

4) Implementation of Connection oriented n/w

* classification of Routing :



* Routing ~~algg~~ algorithms

* Routing algorithms make decisions based on topologies and traffic

1) Optimality pri principle (1957, Bellmeom)
2) shortest path algorithm (Dijkastras)
3) Flooding algorithm
4) Distance vector Routing
   i) coond to infinite problem
5) link state Routing
6) Hierarchal Routing

H_1 – connection 1 – H_2

H_3 – connection 1 – H_2

A's table

	type
H_1	1
H_3	1

I/p Table

C	1
C	2

O/p

* Routing algorithm :-

* Routing is a proces of selecting a path from source to the distination
* even if the source and the distination are not in the same network, it maintains the Route and transmits the packets without any traffic
* Few algorithm have been proposed such that which path is to be selected to transmitt the packets

a) service provide to Transport layer:-

* services should be independent ~~effet~~ of the router technology

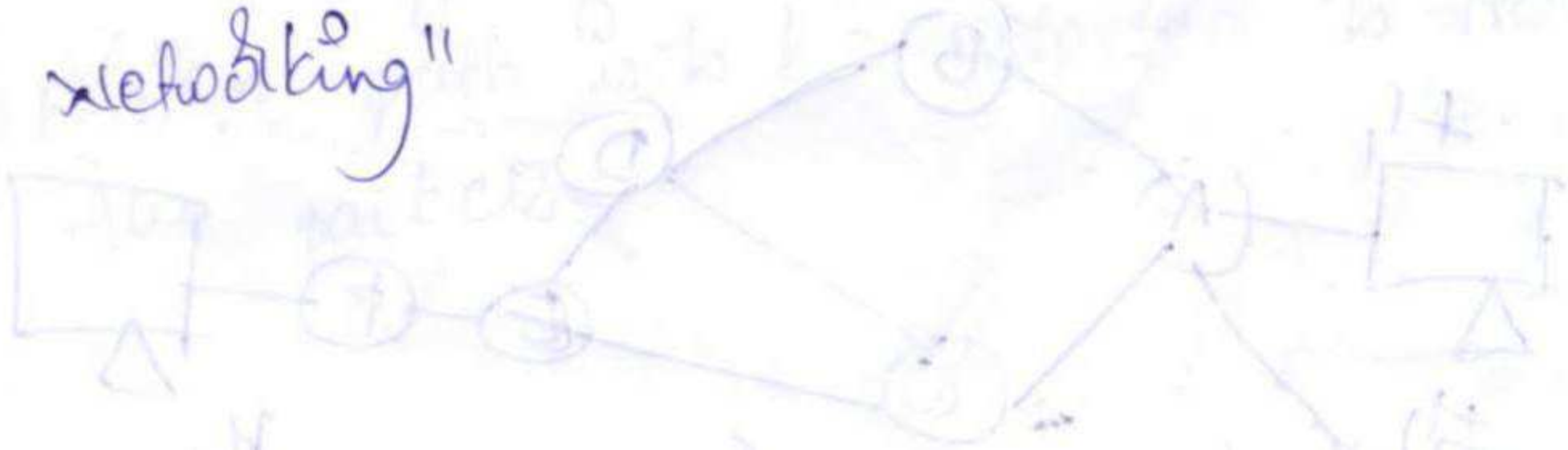
* The transport layer should be shielded from the number, type and topology of routers present

* shielded means "uneffected".

* The network addres must be made available to the transport layer

* If your using connectionless service then that is called "datagram networking"

* If your using connection oriented service then that is called "virtual circuit networking"



21/08/18

Unit-3

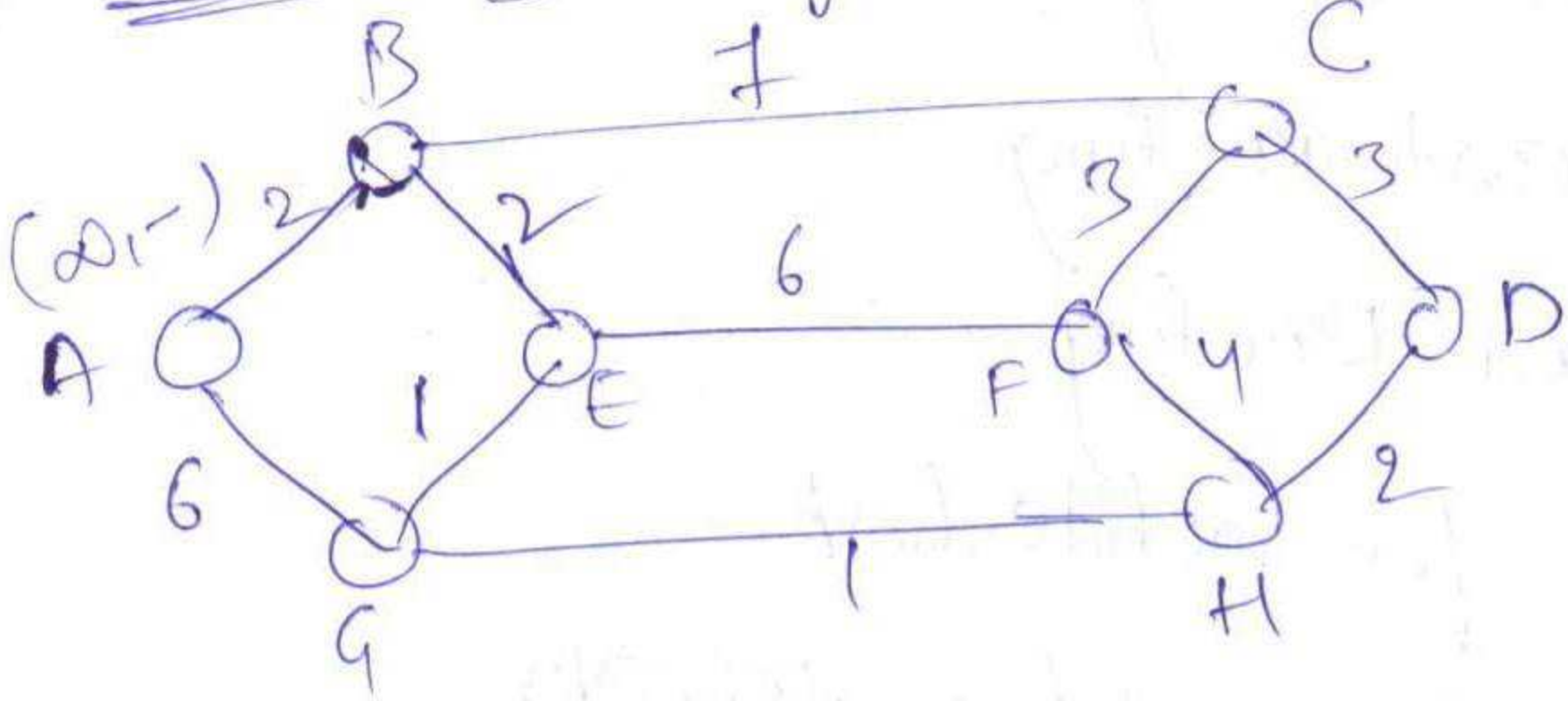
Network layer

* Network layer responsible for getting packets from source to the destination
* To acquire this the network layer must known about the topology of the network and also choose appropriate path to avoid overloading

* Network layer design issues:

* In order to design a network layer following things are to kept in mind
  1) store and forward packet switching
  2) Services provided to Transport layer
  3) Implementation of connection less Network
  4) Implemention of connection Oriented Nlw

2) shortest path algorithm



A to D

(weights, source node)

(∞, ∞) → variable

B (2, A)

3) Flooding algorithm :-

* hop counts

* Flooding is a technique in which every incoming packet is sent out to each and every node except the one @ it arrived on

* Flooding generates a duplicat packets Therefore one of the maj measure is to Layer

7) Broad cast Routing

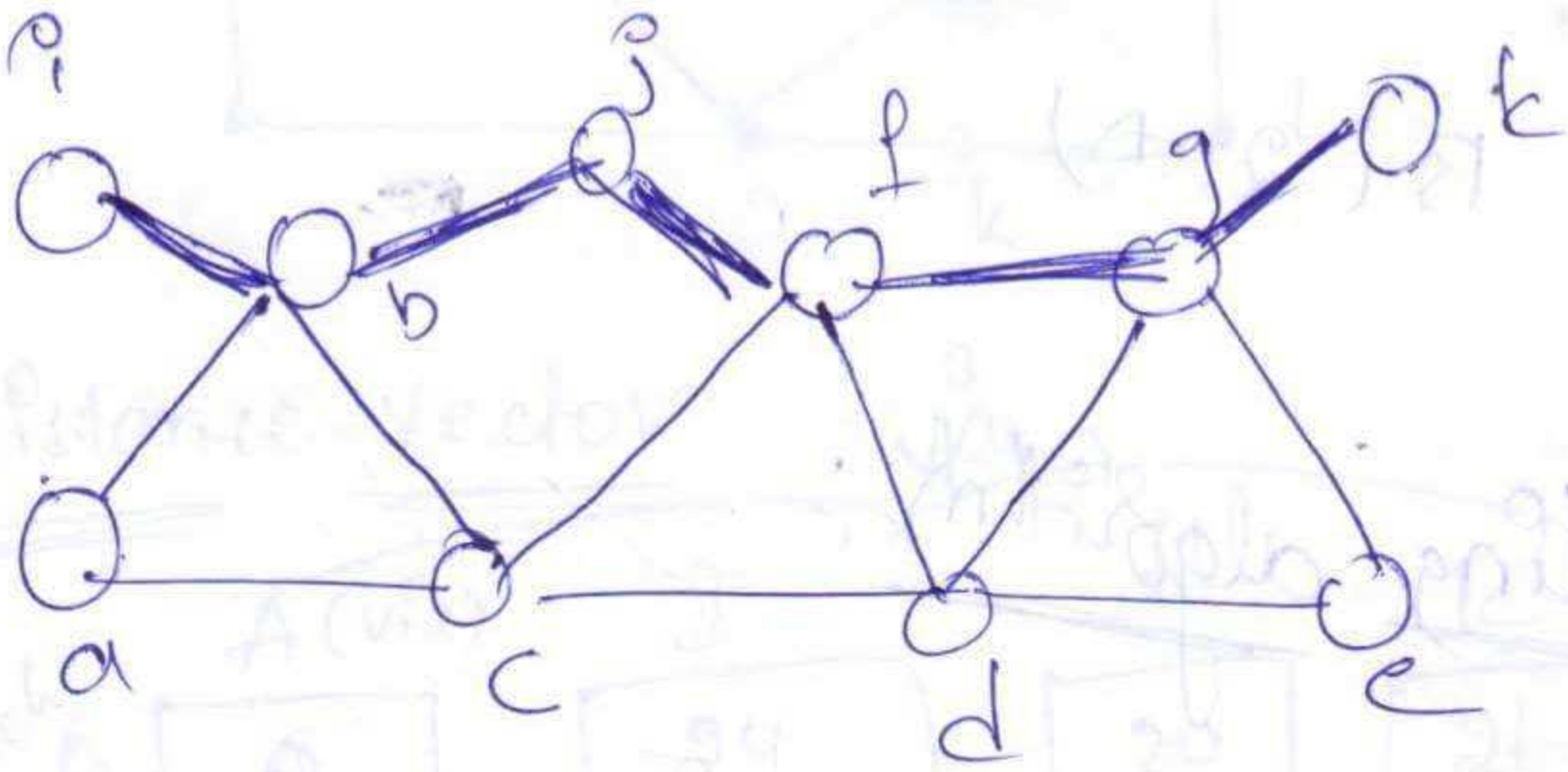
8) multi cast Routing

9) Any cast Routing

10) Routing for mobile host

11) Routing for Ad hoc networks

* optimality principle



* optimal path is best path

* If Router J is on the optimal path i to k, then the optimal path from j to k also falls along the same route

count to infinite problem

∞ . . . Initially

∞ 1 . . 1st exchange.

∞ 1 2 . .

∞ 1 2 3 .

∞ 1 2 3 4

∞ 1 2 3 4

3 2 3 4

3 4 3 4

5 4 5 4

5 6 5 6

5 6

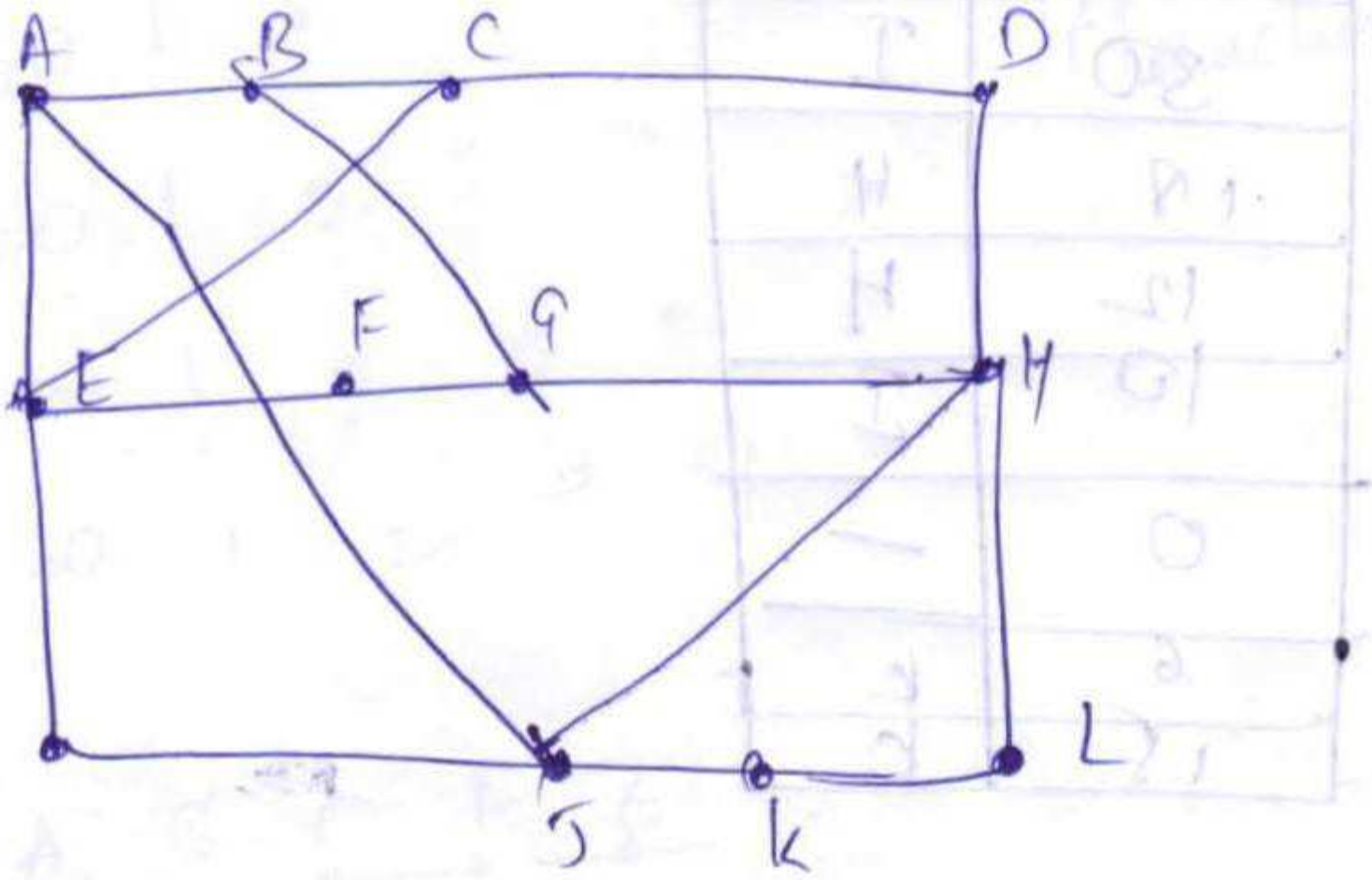
*Distance

2ul08/18

* Distance vector Routing :-

* Distance vector Routing was developed by "Belman ford", "Fulkerson" in the year 1962

hop Counter, It is contained in the head of each packet. It decrements its value when it move on to the next hop

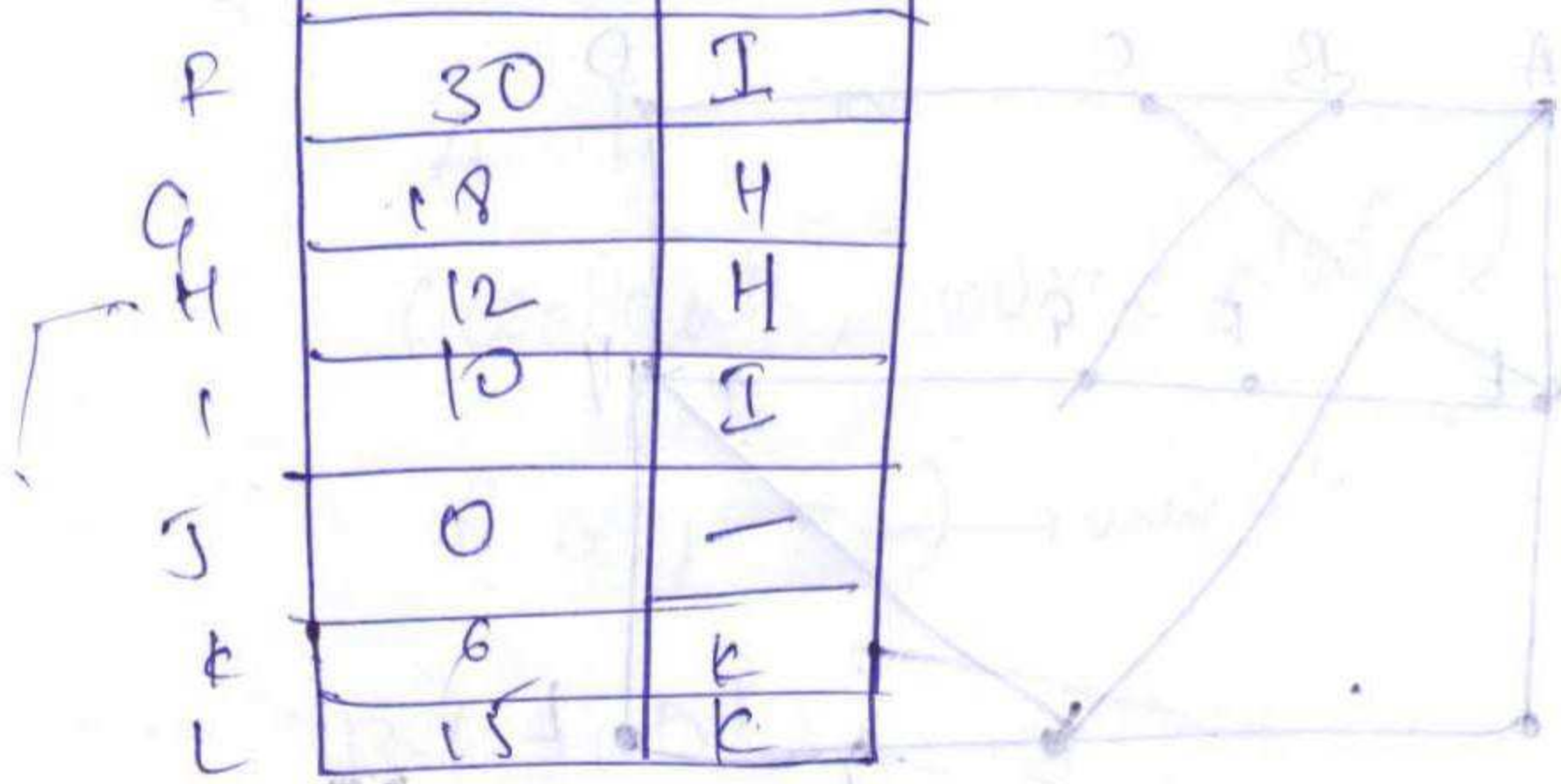


* Distance vector

To Reach	A (via)	B	H	K	Delay Given
A	0	24	20	21	J to A — 8 milli sec
B	12	36	31	28	J to I — 10 "
C	25	18	19	36	J to H — 12 "
D	40	27	8	24	J to K — 6 "
E	14	7	30	22	
F	23	20	19	40	
G	18	31	6	31	
H	17	20	0	19	
I	21	0	14	22	
J	9	11	7	10	
K	24	22	22	0	
L	29	33	9	9	

	Node estimated delay	Line via
A	8	A
B	20	A
C	28	I
D	20	H
E	17	I
F	30	I
G	18	H
H	12	H
I	10	I
J	0	—
K	6	K
L	15	K

5 — other nodes



Estimation from J to G → table

via A $8+18=26$

B $= 10+31=41$

adjacent node for J

H $= 12+6=18$ — least

K $= 6+31=37$

J to G via H is 18 (min)

so in table at place G we have to write (H, 18)

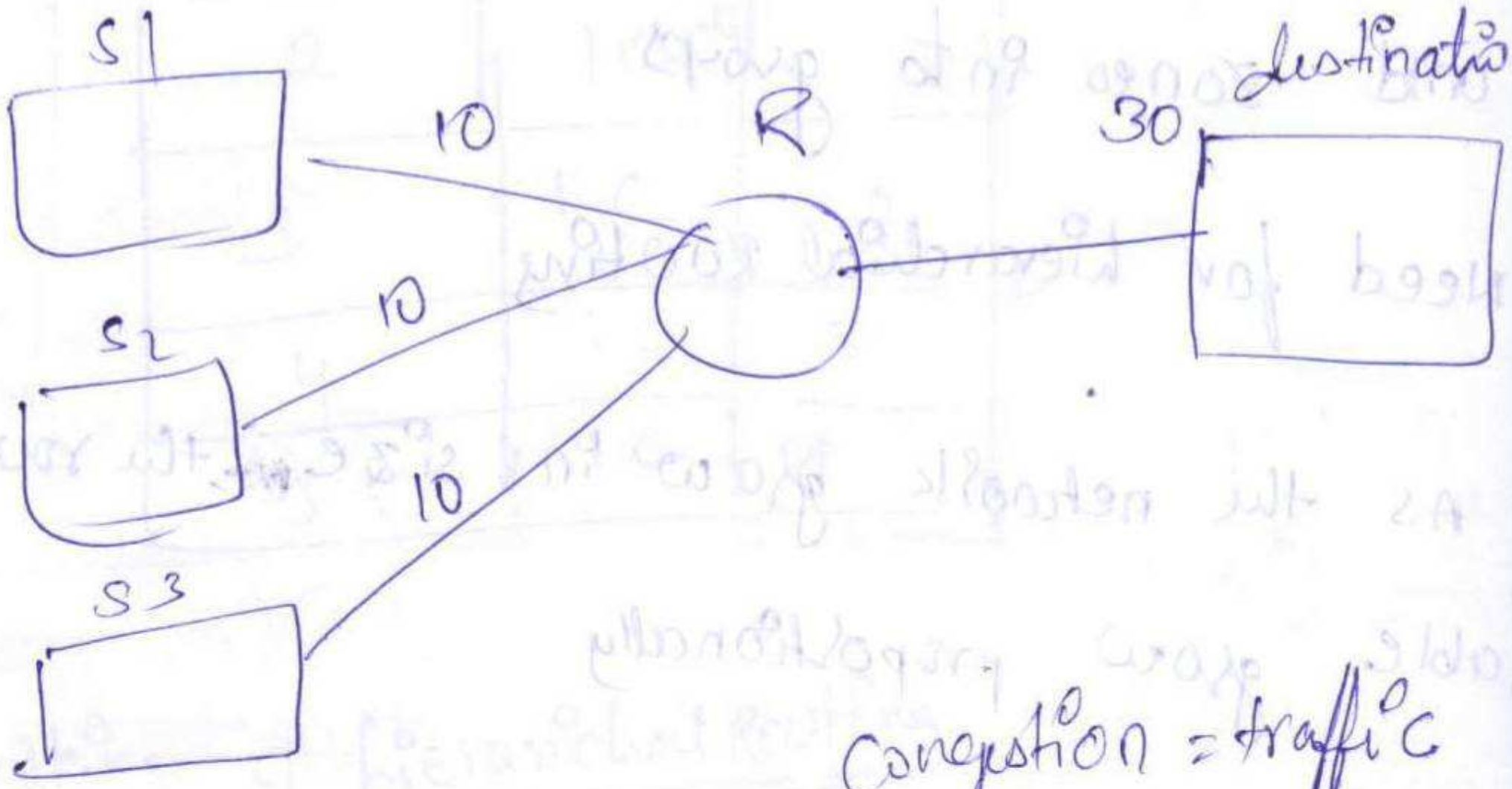
* It is an algorithm that operator by having each Router maintainning a table which gives the best path to ~~a known~~ each of its distination

* Routing tables are updated by exchange information with the neighboors

* Eventually, every ~~Router~~ Router knoons the best link to reach each distination

* The Router table entry has 2 parts
  *) ~~that~~ the prefered outgoing line to use for that distination and estimate of the distance to that distination

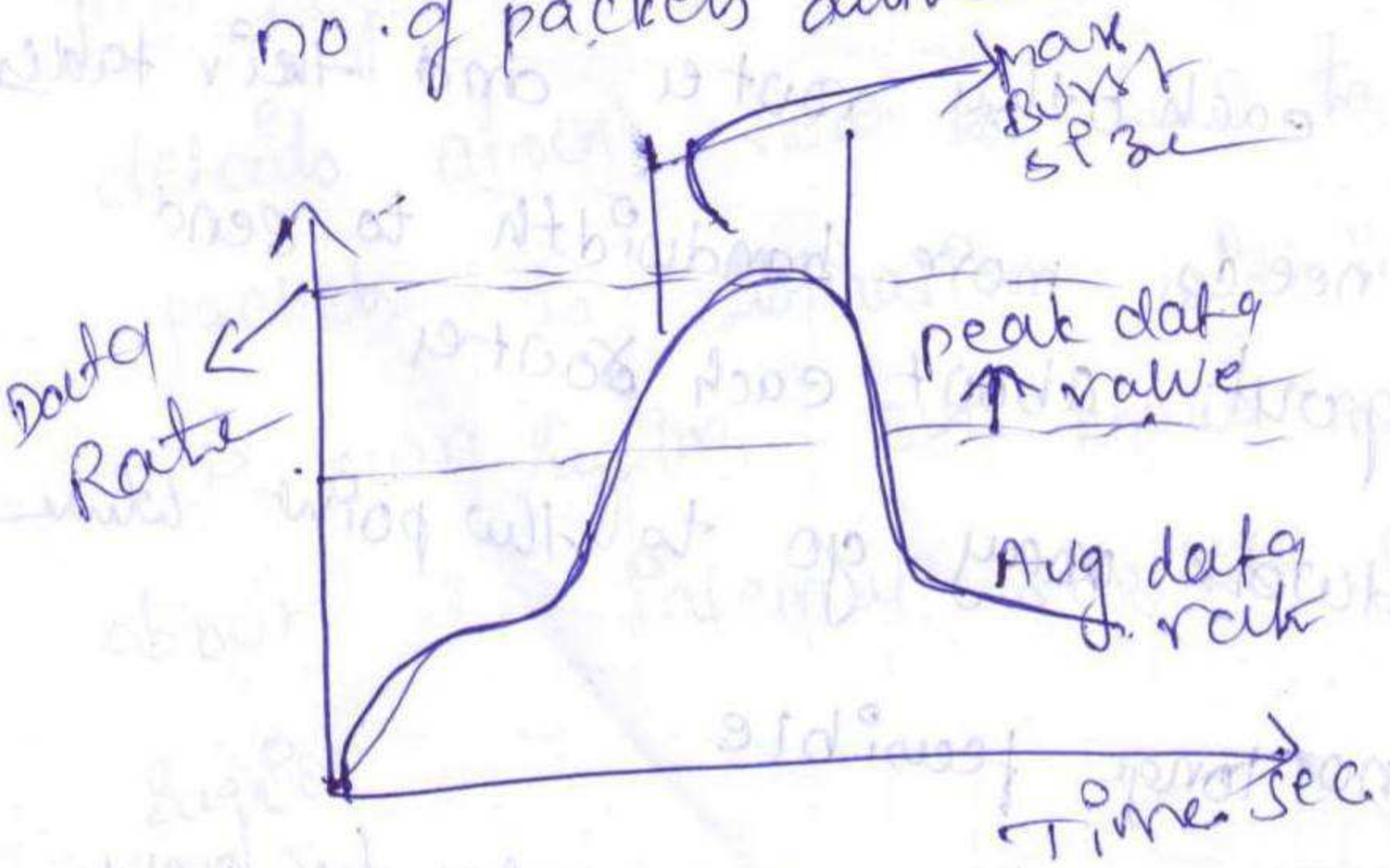
Congestion control :-



Congestion = traffic

Congestion

no. of packets delivered ∝ no of packets sent



Heirarchal table ⟹ Region 1 table

Regions		
1A	—	—
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

*Working of hierarchal Routing

* In a network as routers are divided into region, each router knowns all the details about, how to Route packets to distination with in the its own region, But knows nothing about the Internal structure of other region

* peak data value

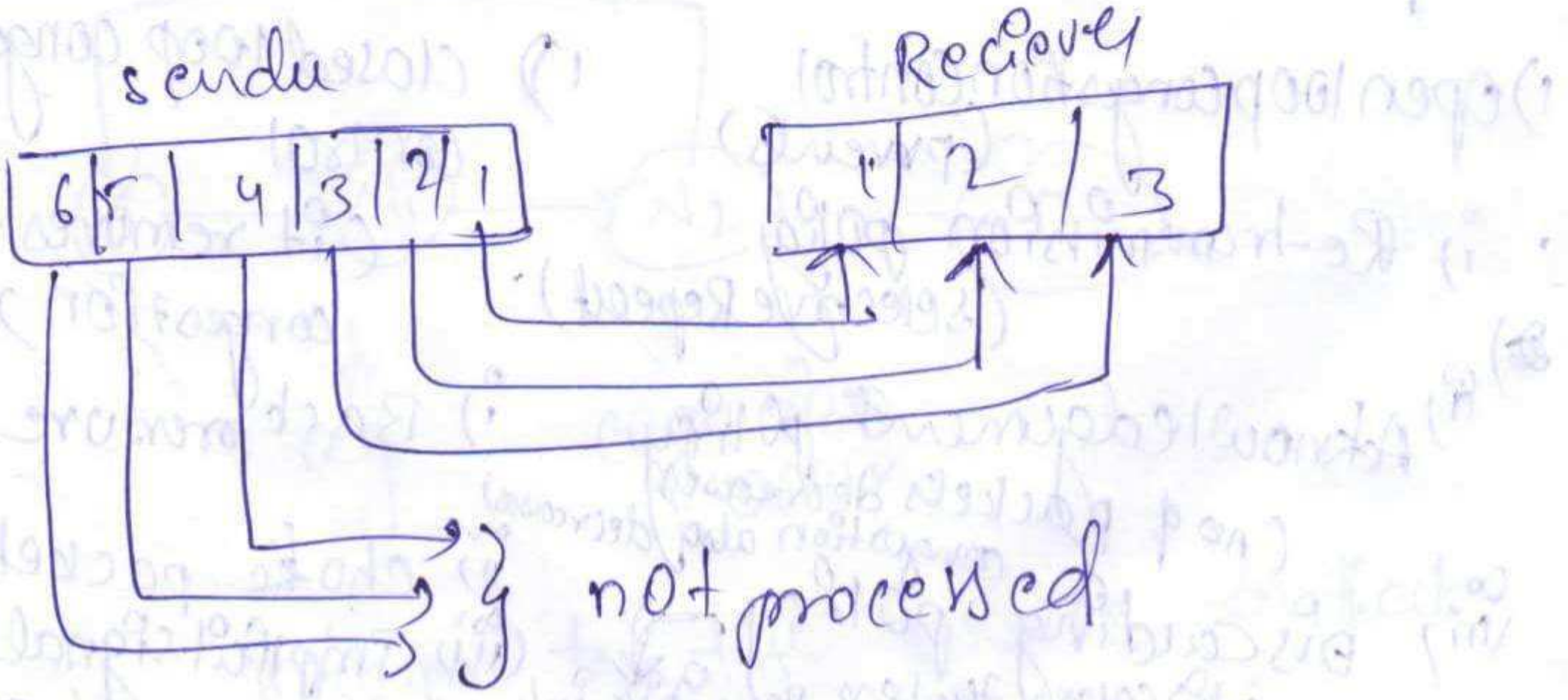
* It is the maximum data rate of the traffic

* Average data rate

* Amount of data transfered per time

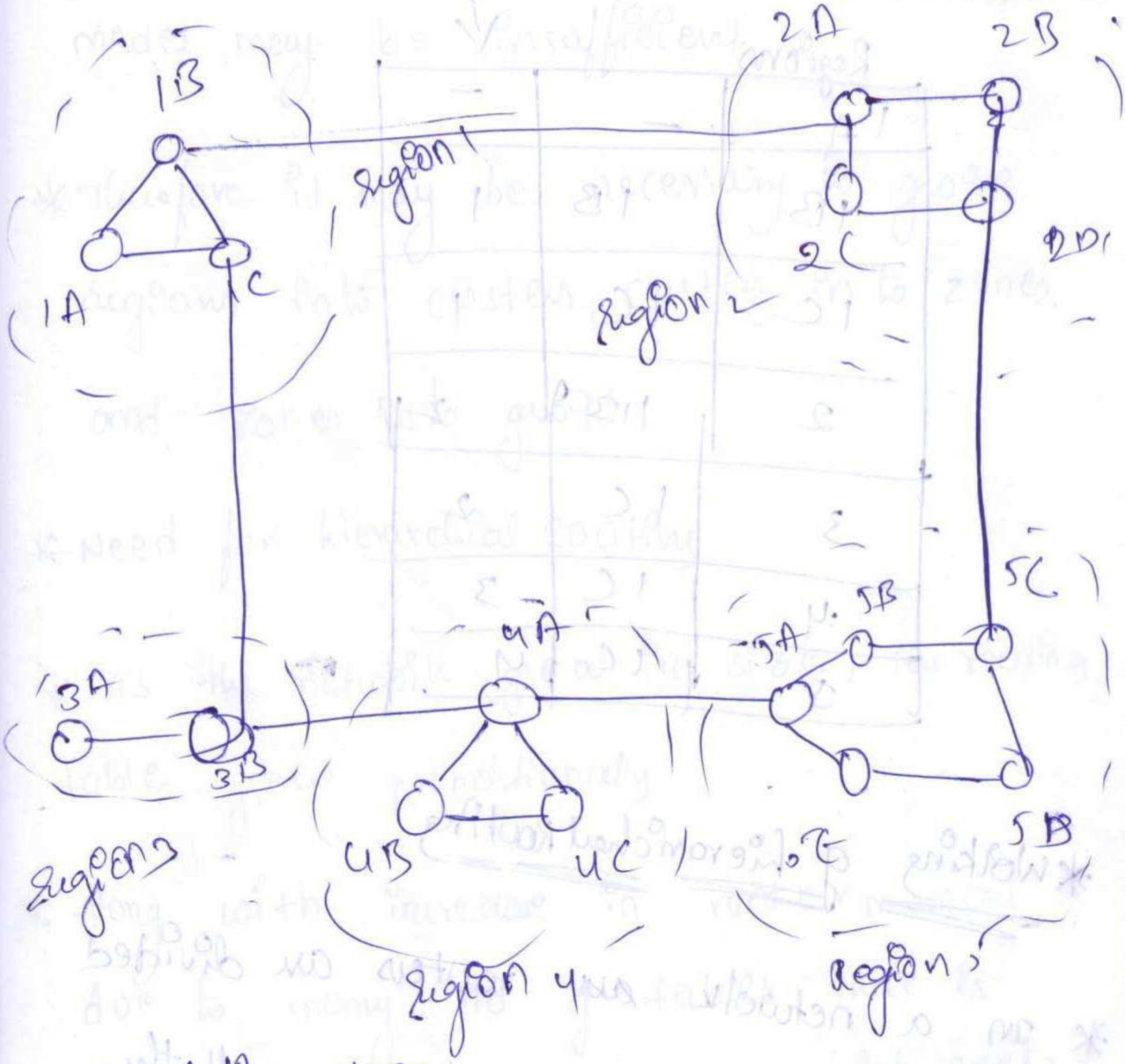
* maximum burst size

* max length of time above which the traffic is generated at peak rate



04/09/18

Hierarchical Routing



- Dest line Hops

1A	-	-
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2

4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

* for huge networks, a 2-level hierarchal model may be insufficient.

* Therefore it may be necessary to group regions into clusters, clusters into zones, and zones into groups

* Need for hierarchical ROUTing

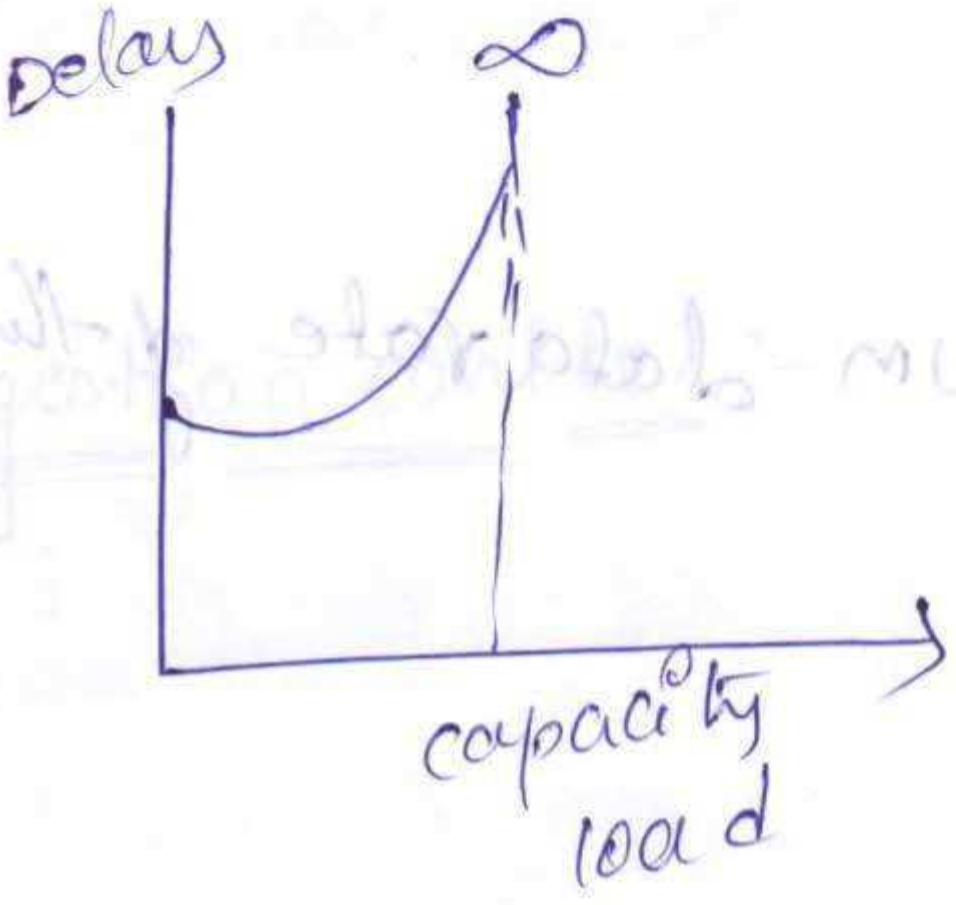
* As the network grow in size, the routing table grow proportionally

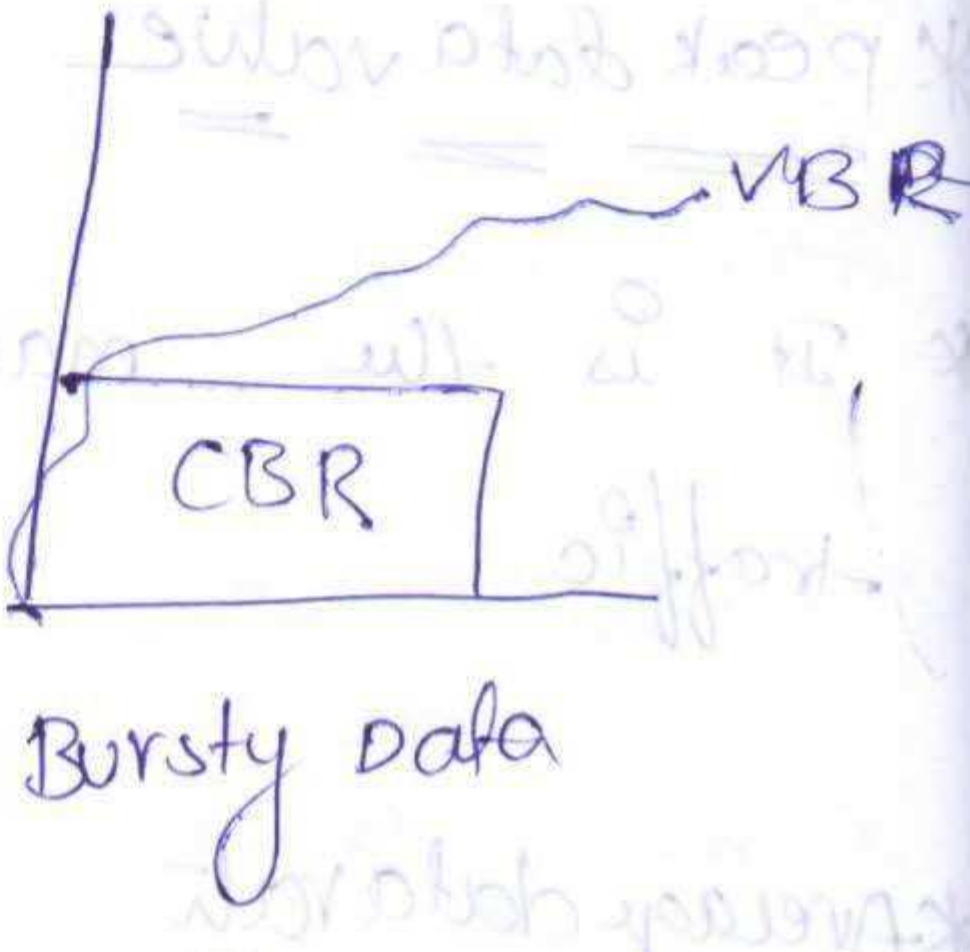
* along with increase in router memory due to many no of tables there is also increase in cpu time. which needs to scan each other router and their tables

* It also needs more bandwidth to send status report about each router

* the network may go to the point where it is no longe feasible

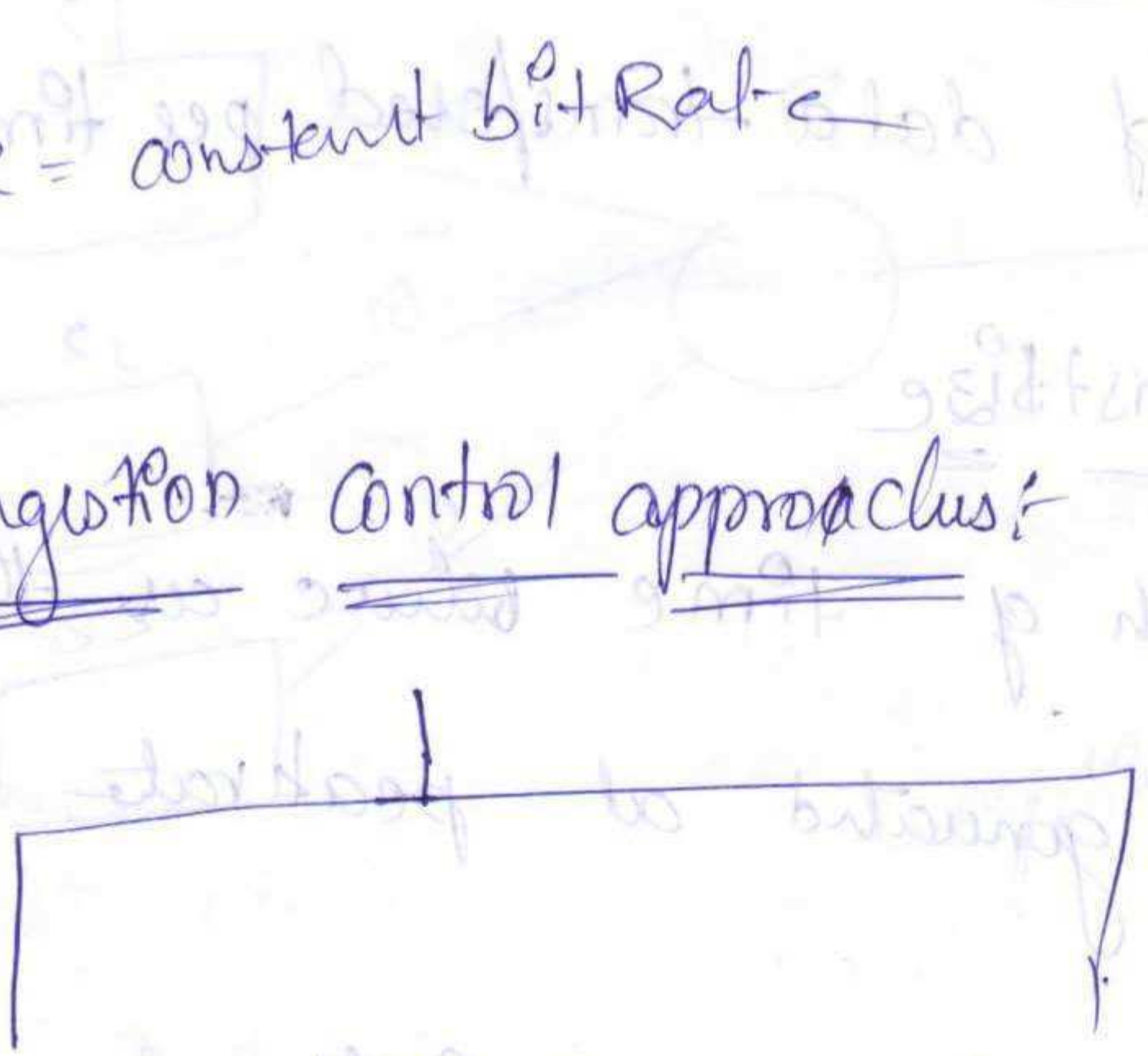
* for every router to have an entry for every other routes.





CBR = constant bit Rate

* Congestion control approaches:



i) open loop congestion control (prevents)

- i) Re-transmission policy (selective Repeat)
- ii) Acknowledgement policy (no of packets decreases congestion also decreases)
- iii) Discarding policy → (discards fewer sensitive information)
- iv) Admision policy (Resources are checked before transmitting packets)

checks Resources available at the source node

i) closed loop congestion control (it removes the congestion)

- i) Back pressure
- ii) choke packet
- iii) Implicit signalling
- iv) Explicit signalling

* Here one choke packet is send directly from congested node to source node

* Implicit signalling

* If no communication b/w congestion node with other node / source node. Then source assumes that congestion has be occured

Therefore the source slows down itself and send packet again after sometime

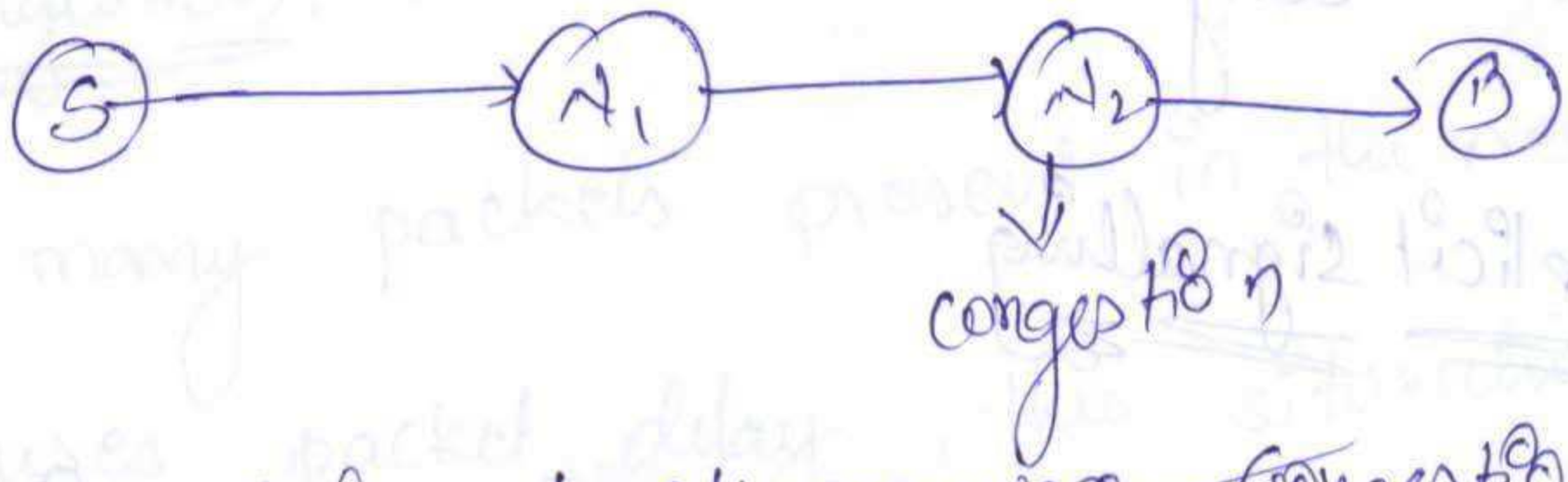
* Explicit signalling :

* Congested node sends a signal to source / destination, signal included in the packets

* Therefore signal can be Backward or forward.

* Backpressure

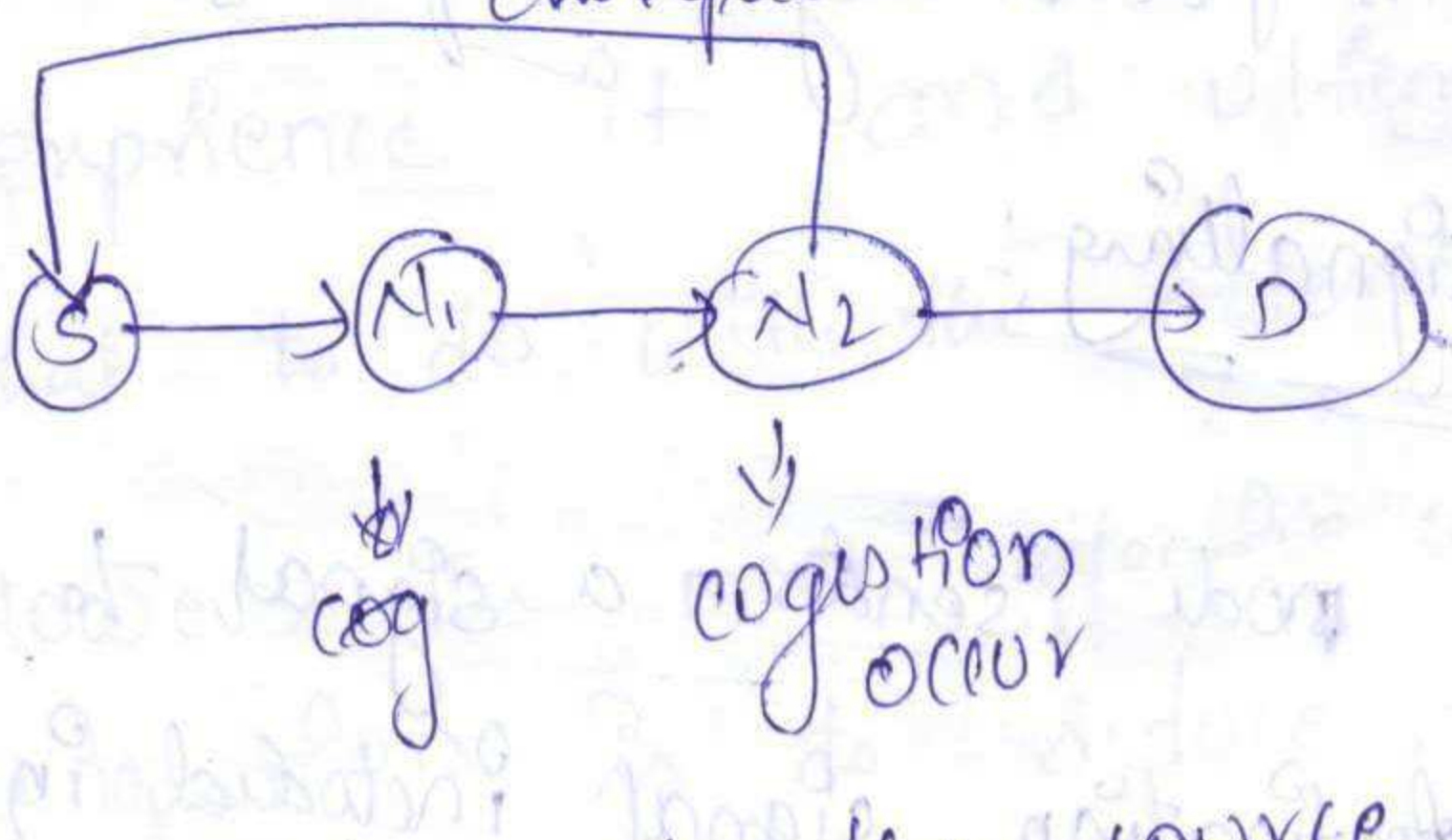
*



* By applying backpressure congestion removed
* Backpressure is an congestion control technique (at2) that starts with node(N_2) and propagate up to the source node

* chokepacket:

*



* It informs to the source that congestion occur
* It is a packet send by a node to the source to inform its congestion

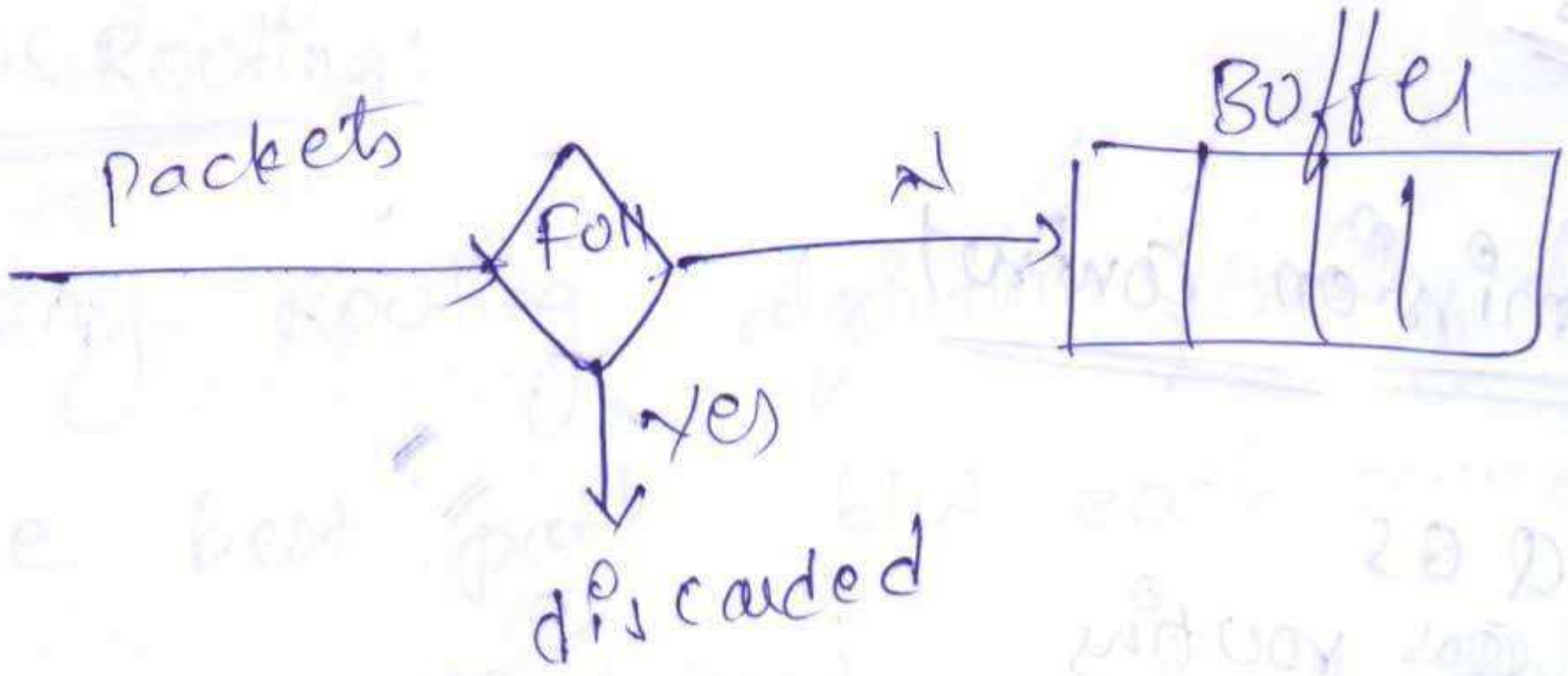
* Congestion :-

* To many packets present in the network causes packet delay, this situation is called congestion

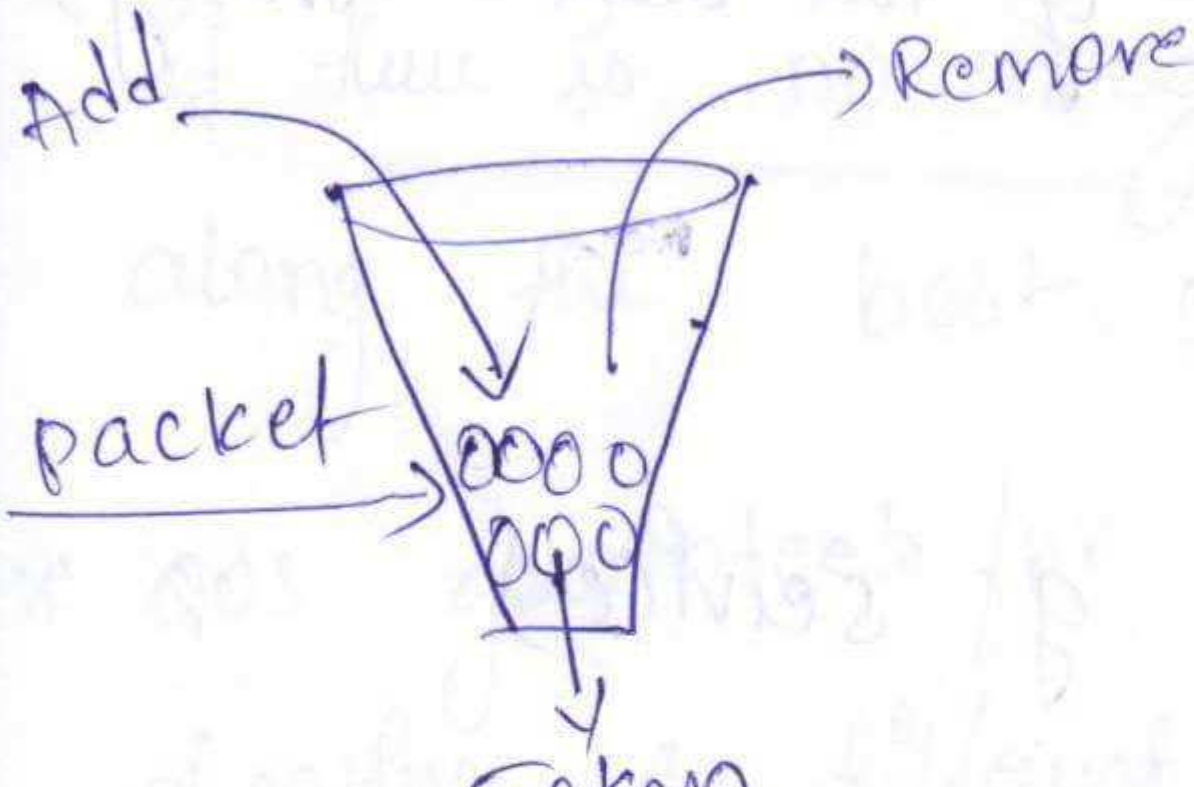
* The network layer and the transport layer shares the responsibility for handling congestion

* since the congestion occurs within the network, it is the network layer that experience it and ultimately determines what to do with the excess packet

* However the most effective way to control congestion is to reduce the load, that the transport layer is placing on the network layer.



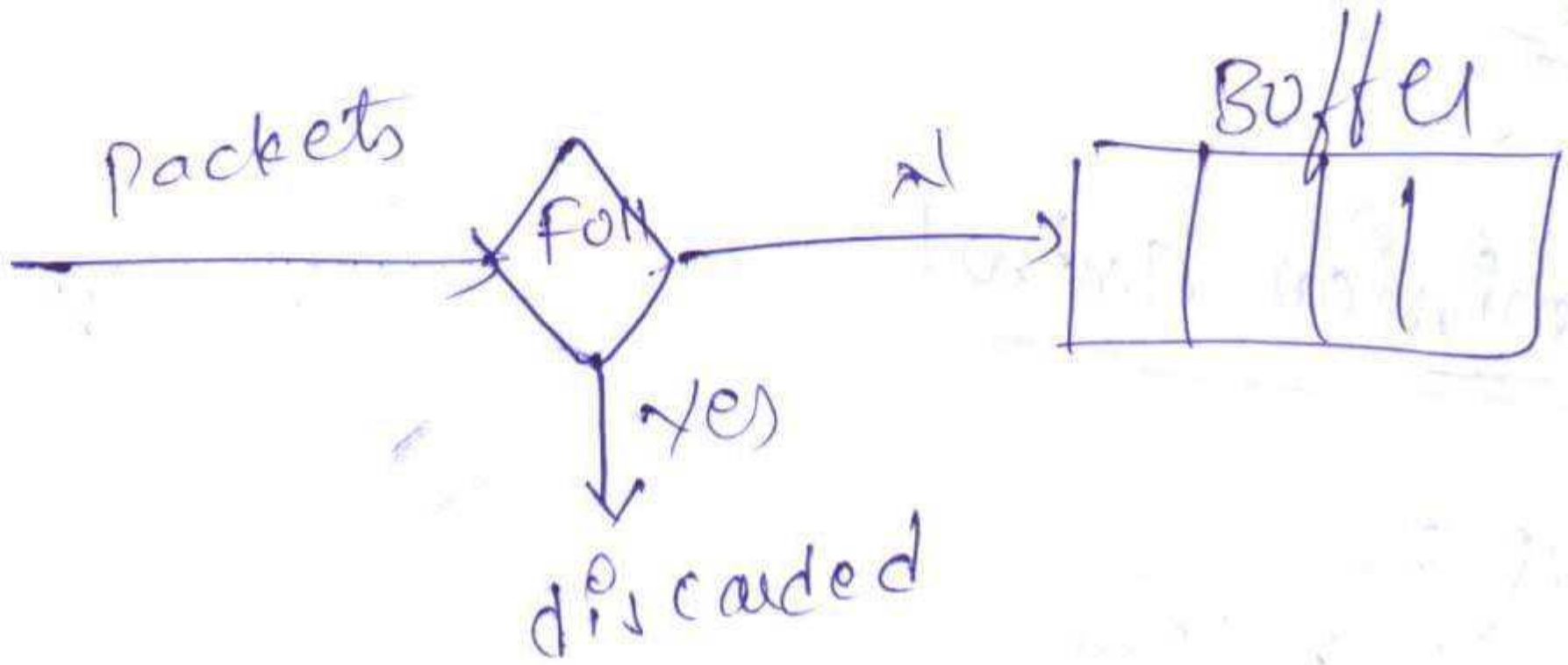
*Token bucket



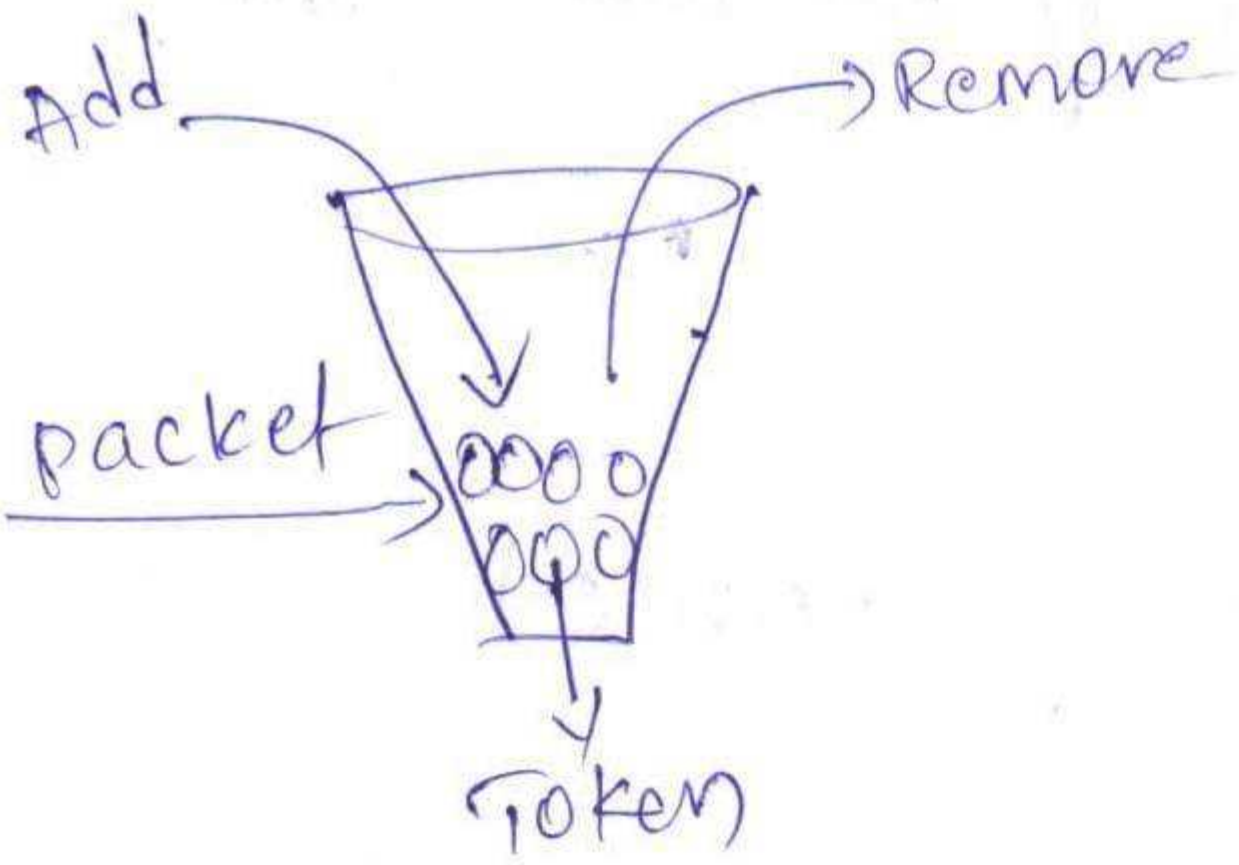
counter is used

when token is add counter is incremented
when token is removed counter is decremented

* token add, according to the capacity of the bucket
* Remove means packet is sending to destination
* Speed of Adding of token is equal to speed of Removing of token



*Token bucket



counter is used

when token is add counter is incremented.
when token is removed counter is decremented

* token add, according to the capacity of the bucket
* Remove means packet is sending to destination
* Speed of Adding of token is equal to speed of Removing of token

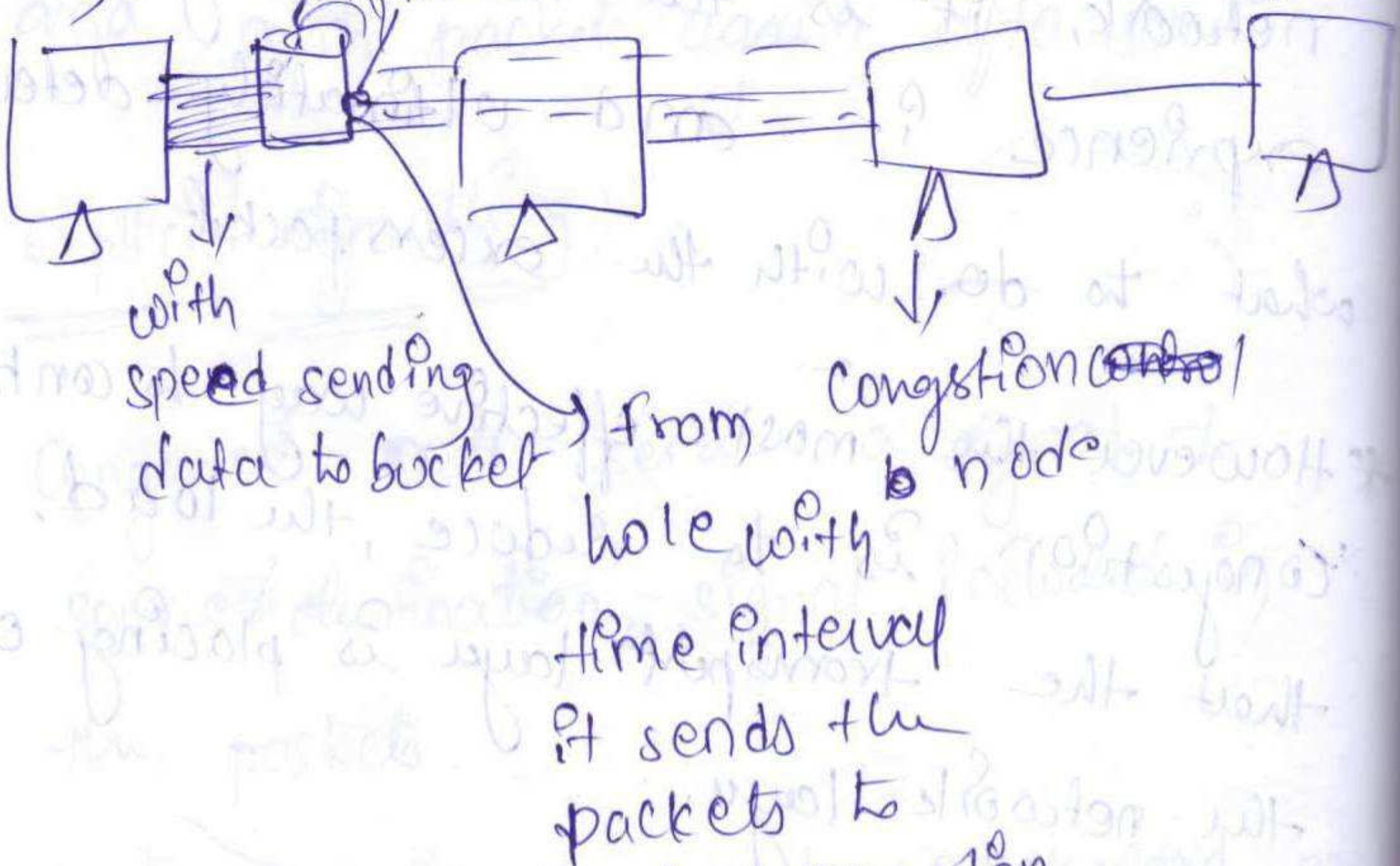
Congestion control algorithm:

* This are the mechanism used to control traffic

* 1) Leaky Bucket Algorithm
  2) Token ~~Bucket~~ Bucket

* ## Leaky bucket:

* leaky Bucket is used to control Congest if packets from source is more than capacity of Bucket (if will discard (spillout)



hole with time interval it sends the packets to control congestion

* Admission control

1) QOS
2) Qos routing
* 3) flow specification

* Resources are to be check before the transmission of the data through the network

* QOS :- (qualitive of service)

* * QOS refers to a network ~~data~~ ability to achieve maximum bandwidth and deal with good network peformance
* qos also involves controlling and managing network resources
* ~~no~~ rck

a flow specification proposing the parameters it would like to use. As this specification propagate along the route, each router examines it and modifies the parameters as needed to be

parameter	units
Token Bucket Rate	Bytes/sec
Token Bucket size	Bytes
peak data rate	Bytes/sec
minimum packet size	Bytes
maximum packet size	Bytes

Congestion control Algorithms:-

1) Leaky Bucket Algorithm
2) Token Bucket Algorithm

* QOS Routing:-

* Many Routing algorithm finding the single best path blw each source and each destination and send all traffic over the best path.

* This may cause some flows to be rejected if there is no enough spare capacity along the best path.

* QOS guarantees for a new flow by choosing a different route for the flow that has excess capacity

* Flow specification:-

* many parties may be involved in the flow negoth negotiation, flow must be described accurately in terms of specific parameters.

* A set of such parameters is called a flow specification. The sender produces

* Leaky Bucket Algorithm:

let us consider an example to understand. imagine a bucket with a small hole in the bottom, no matter at what rate water enters the bucket, the outflow is at constant rate. when the bucket is full with water additional water entering spills over the sides and is lost

* steps involved in algorithm

1) when the host wants to send packet it is thrown into the bucket

2) The bucket leaks at a constant rate, meaning the n/w interface transmits packets at a constant rate

3) Bursty traffic is converted to a uniform traffic by the leaky bucket

4) In practice the bucket is a finite that outputs at a finite rate

2) Token Bucket Algorithm:

* Need of token bucket algorithm:

The leaky bucket algorithm enforces output pattern at the average rate, no matter how bursty the traffic is. so inorder to deal with the bursty traffic. we need a flexible algorithm so that the data is not lost. one such algorithm is token bucket algorithm

Steps:-

1) In regular intervals tokens are thrown into the bucket
2) The bucket has a maximum capacity
3) If there is a ready packet, a token is removed from the bucket, and the packet is send
4) If there is no token in the bucket the packet cannot be send

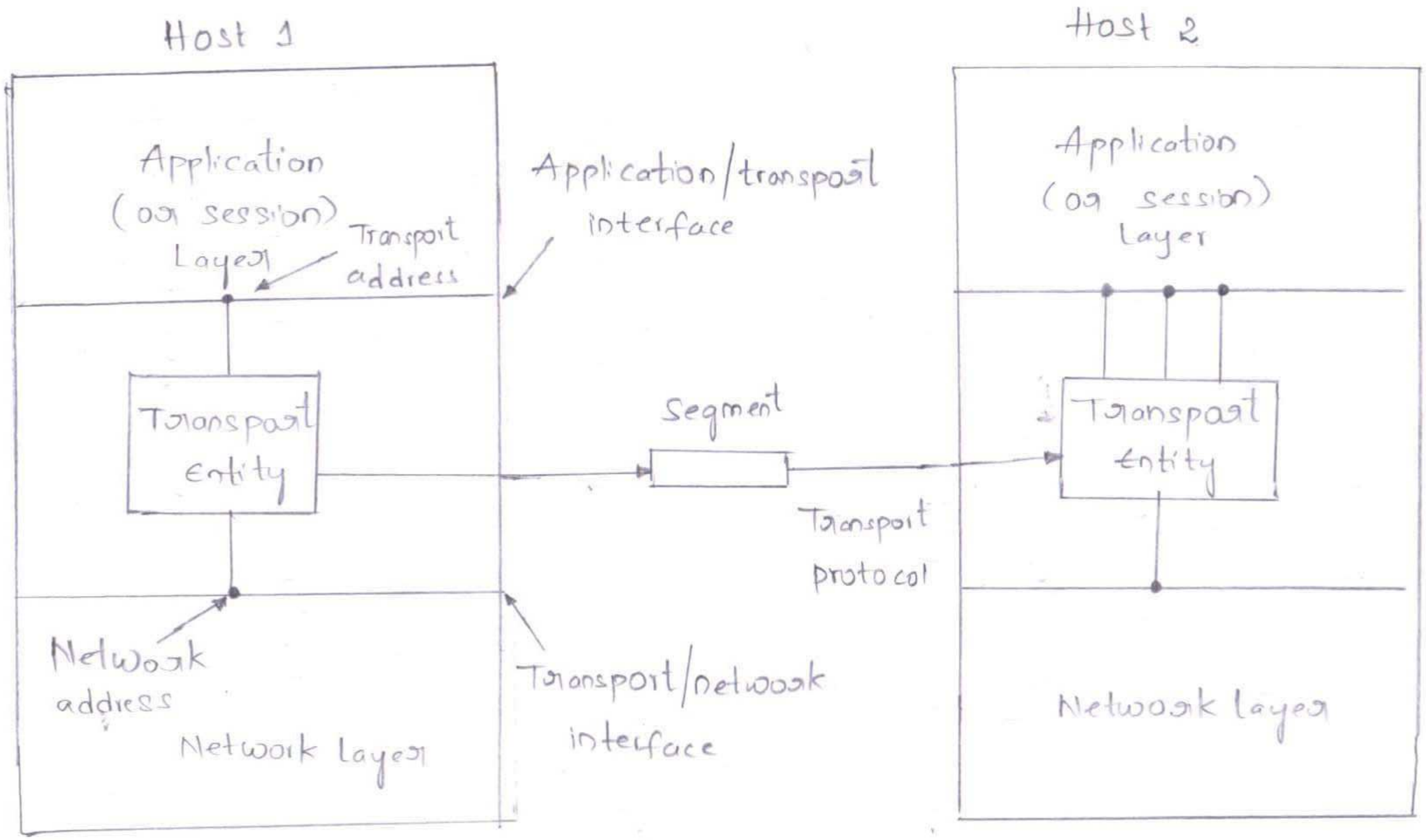
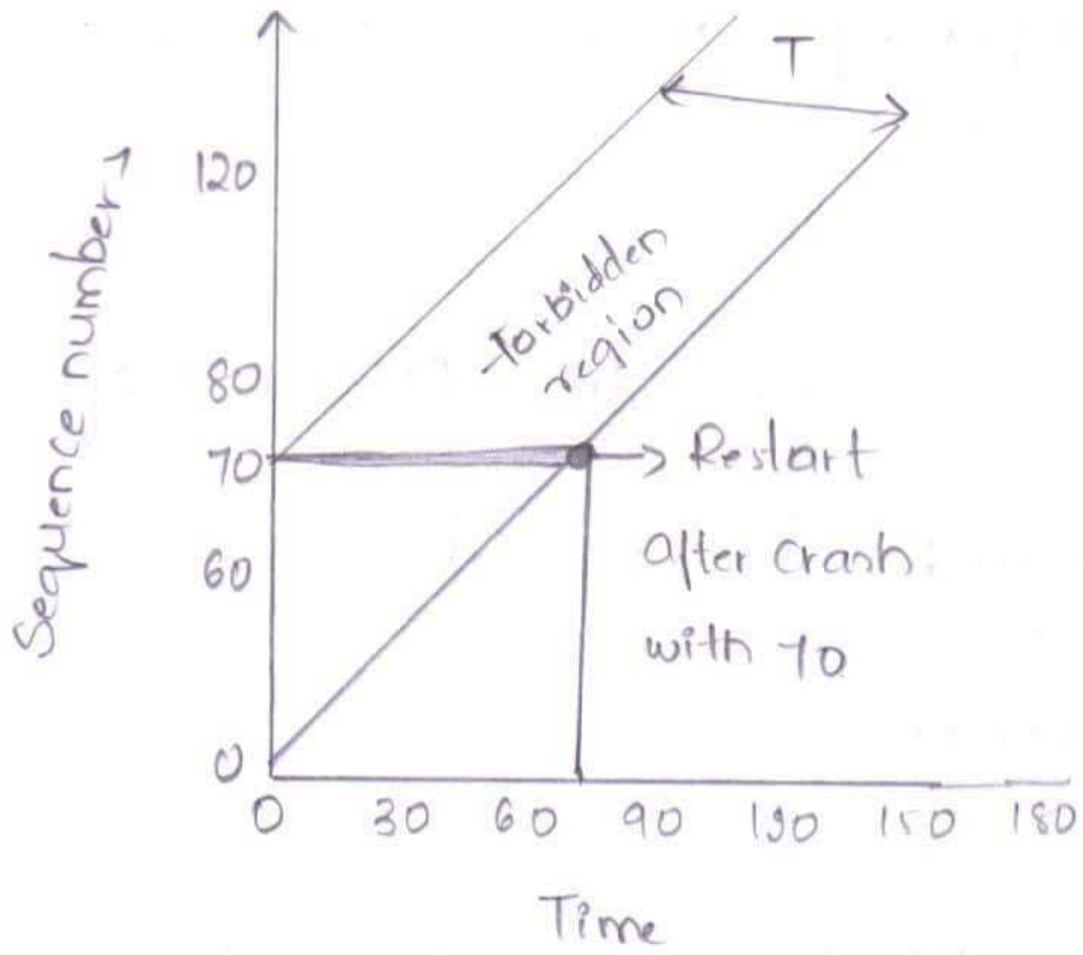
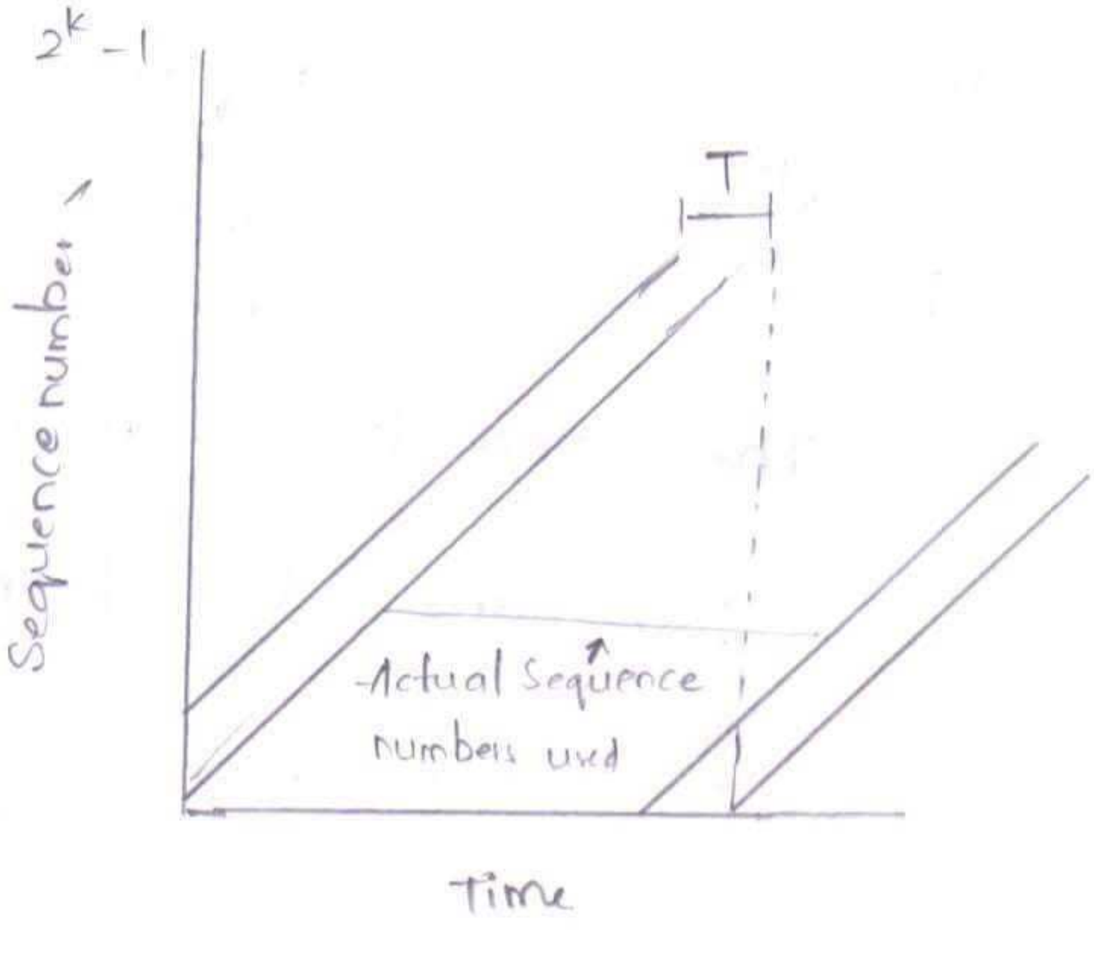


Fig: 6-1 The network transport and application layer



(a)



(b)

(a) Segments may not enter forbidden region

(b) the resynchronization Problem.

To keep the packets sequence numbers out of forbidden region, we get into trouble in two distinct ways. If a host sends too much data too fast on a newly opened connection, the actual sequence number versus time curve, causing the sequence number to enter into forbidden region To prevent this from happening maximum data rate on any connection is one segment per clock tick.

UNIT-4

Service Provided to the Upper Layers.

The ultimate goal of the transport layer is to provide efficient, reliable and cost-effective data transmission service to its users, normally processes in the application layer. To achieve this, the transport layer makes use of the services provided by the Network layer. This software and/or hardware within the transport layer that doesnot work is called Transport entity. The Transport entity can be located in the operating system kernel in a library package bound into Network applications. In a Seperate used process (or) even on the Network interface. The first of two options are most common on the internet. The (Logical) Relationship of the Network, transport and application layers is illustrated in Fig 6.1.

Just as there are two types of Network service, connection oriented and connection-less. There are also two types of transport service. The connection oriented transport service is similar to the connection-Oriented Network service in many ways. In both cases, connections have three phases: establishment, data transfer and release. Addressing and flow control are also similar in both layers. Furthermore, the connectionless transport service is also very similar to the connectionless network service. However it is difficult to provide a connectionless transport service on top of a connection-Oriented network service. Since it is inefficient to set up a connection to send a single packet and then tear it down immediately afterwards

The obvious question is this: if the transport layer service is so similar to the network layer service, why are there two distinct layers?

the heart of the method is for the source to label segments with sequence numbers that will not be reused within T secs. the period 'T', rate of. Packets per second determine size of sequence numbers.

To get around the problem of machine losing all memory of where it was after a crash, one possibility is to require transport entities to be idle for T secs after a recovery this may be unattractive.

Tomlinson proposed equipping each host with a time of day clock. The clocks at different hosts need not to be Synchronized. Each clock takes the form of binary counter that increments itself at Uniform intervals.

As such when a connection is setup, the lower-Order k bits of the clock are used as k-bit initial sequence number. So each connection starts numbering its segments with a different initial sequence number

Why is one layer not adequate? The answer is subtle, but crucial. The transport code runs entirly on the users machines, but the network layer mostly runs on the routers.

problems occurs, That's what. The users have no real control over the network Layer. So, they cannot solve the problem of poor service by using better routers (or) putting more error handling in the data link layers because they dont own the routers. The only possibility is to put on top of the Network Layer another layer that improves the quality of services. If in a connection-oriented network a transport entity is informed halfway through a long transmission that its network connection has been abruply teminated, with no indication of what has happend to the data currently in transmit, it can set up a

network connection to the remote transport entity. using the network connection, it can be send a query to its peer asking which data arrived and which did not and knowing where it was, pick up from where it left off.

The existence of the transport layer makes it possible for the transport service to be more reliable than the underlying network. Futhermore, the transport primitives can be implemented as calls to library procedures to make them independent of the network primitives. The network service calls may vary considerably from one network to another. Hiding the network service behind a set of transport service primitives ensures that changing the network merely requires replacing one set of library procedures with another one that same thing with different underlying services.

Thanks to the transport layer, application programmers can write code according to a standard set of primitives and have these programs work on a wide variety of networks, without worrying about delaying with different network interfaces and levels of reliability. If all real networks were flawless and all had same services primitives and were guaranted never, ever to change, the transport layer might not be needed. However, in the real world it is fulfills the key function of isolating the upper layer from the technology-design of the Network. The bottom four layers are the transport service provider. The distinction of provider versus user has a considerable impact on the design of the Layers and put transport layer in a key position. Since it forms the major boundary between the provider and user reliable data transmission service.

It also means transport entity should wait until the clock ticks before opening a new connection after a crash restart.

The clock-based method solves the problem of not being able to distinguish delayed duplicate Segments from new segments.

Tomlinson introduced the THREE-WAY HANDSHAKE this involves one peer checking with the other that connection request is indeed current. Host 1 chooses sequence number x and sends a CONNECTION REQUEST segment containing it to host 2. Host 2 replies with an Ack segment Acknowledging x and announcing it own initial Sequence number 'y'. Finally host 1 acknowledges host 2's choice of an initial sequence number in the first data segment that it sends.

Connection Establishment:

Establishing a connection sounds easy, but it is tricky. It would be sufficient for one transport entity to just send a Connection request segment to the destination and wait for a connection Accepted reply. the problem occurs when the network can lose, delay, corrupt and duplicate packets. this behaviour causes serious complications.

Imagine a network that is so congested that acknowlegments hardly ever get back in time and each packet times out and is retransmitted two or three times. Suppose that the network User datagrams inside and that every packet follows a different route. Some of the packets might get stuck in a traffic jam inside the network.

connection and transfer again.

This scenario may sound unlikely, but the point is protocols must be designed to be correct in all cases. and common cases to be implemented efficiently to obtain good network performance. Protocols must be able to cope with uncommon cases without breaking. If not a fair-network is to be built.

for this we study problem of delayed duplicates, with emphasis for establishing connections in a reliable way. Delayed duplicates are thought to be new packets. We cannot prevent those from being duplicated and delayed. But if done they must be rejected as duplicates and not Proceeded as fresh packets.

Oneway is to use throwaway transport addres

and take a long time to arrive. That is, they may be delayed in the network and pop out much later, when the sender thought that they had been lost.

If a user establishes a connection with a bank, sends menages telling the bank to transfer a large-amount of money to the account of a not entirely-trustworthy person. Unfortunately, the packets take a scenic route to the destination and go off exploring a remote corner of the network the sender then times out and sends them all again. this time the packets take the shortest route and are delivered Quickly So the Sender releases the Connection.

Eventually the initial batch of packets finally come out of hiding and arrive at destination in order asking the bank to establish a new

Connection Release.

* Releasing a connection is easier than establishing one. Nevertheless, there are more pitfalls than one might expect here.

* There are two types or styles of terminating connection. They are 1) Asymmetric Release
2) Symmetric Release

* Asymmetric Release :

It is the way the telephone system works, when one party hangs up, the connection is broken. It is abrupt and may ~~rel~~ase result in data loss.

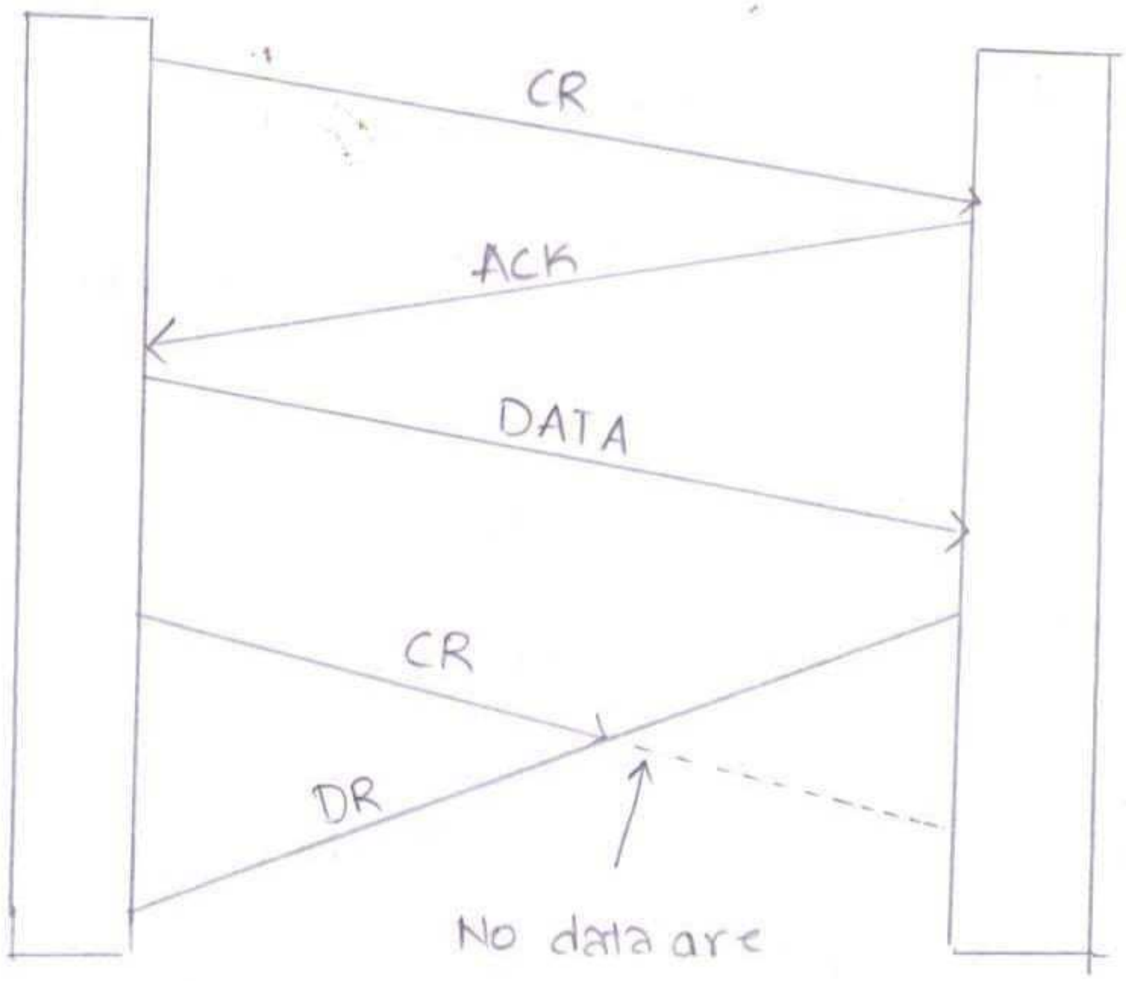


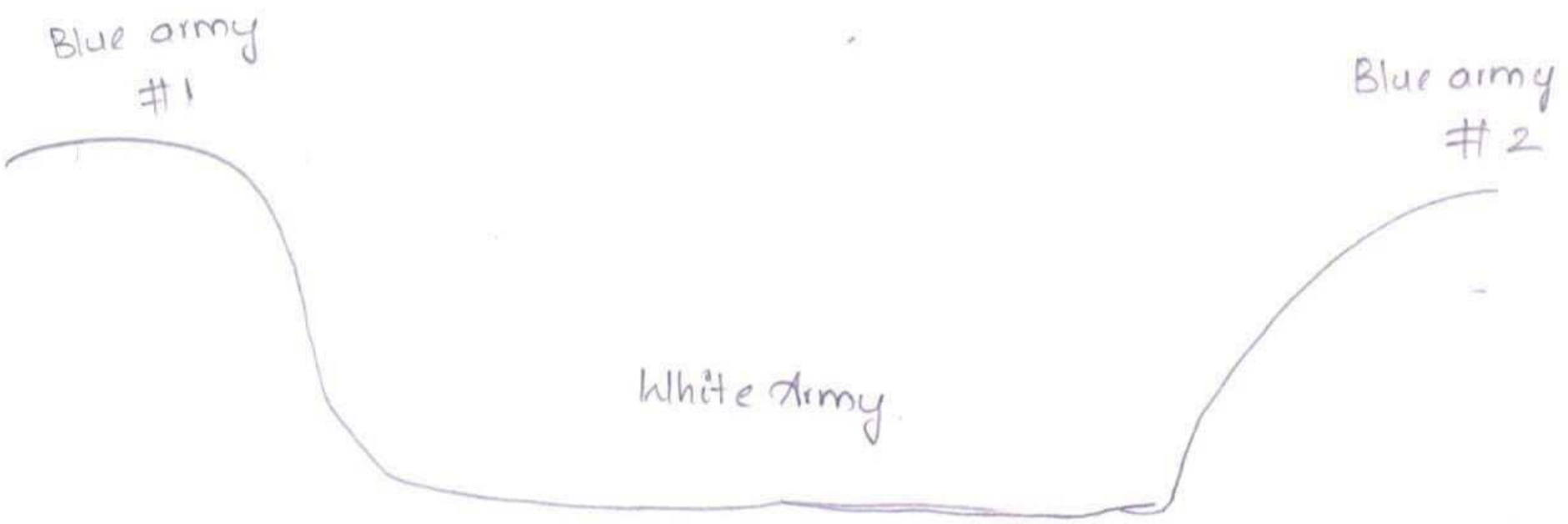
fig: Abrupt disconnection with loss of data

* There is a famous problem that illustrate this issue

Two- Army problem:-

Imagine that a white army is encamped in valley. On both sides (surroundings) are blue armies.

* The White army is larger than either of the blue armies alone, but together the blue armies are larger than the white army.
* If either blue army attacks by itself, it will be defeated.
* The blue armies want to synchronize their attacks, their only communication medium is to send messengers on foot down into valley, where they might be captured and message lost.



Each time a transport addren is needed, a new One is generated. When Connection is releard the addren is discarded and never used agaus. This approach is more difficult to Connect with a procen in first place.

Another possibility is to give a unique identifier chosen by the initiating party and put in each segment, including the one requesting the Connection. Whenever a Connection request comes in, it can be checked against the table to see if it belongs to previously releard Connection.

Packet lifetime can be restricted to a known maximum using following techniques

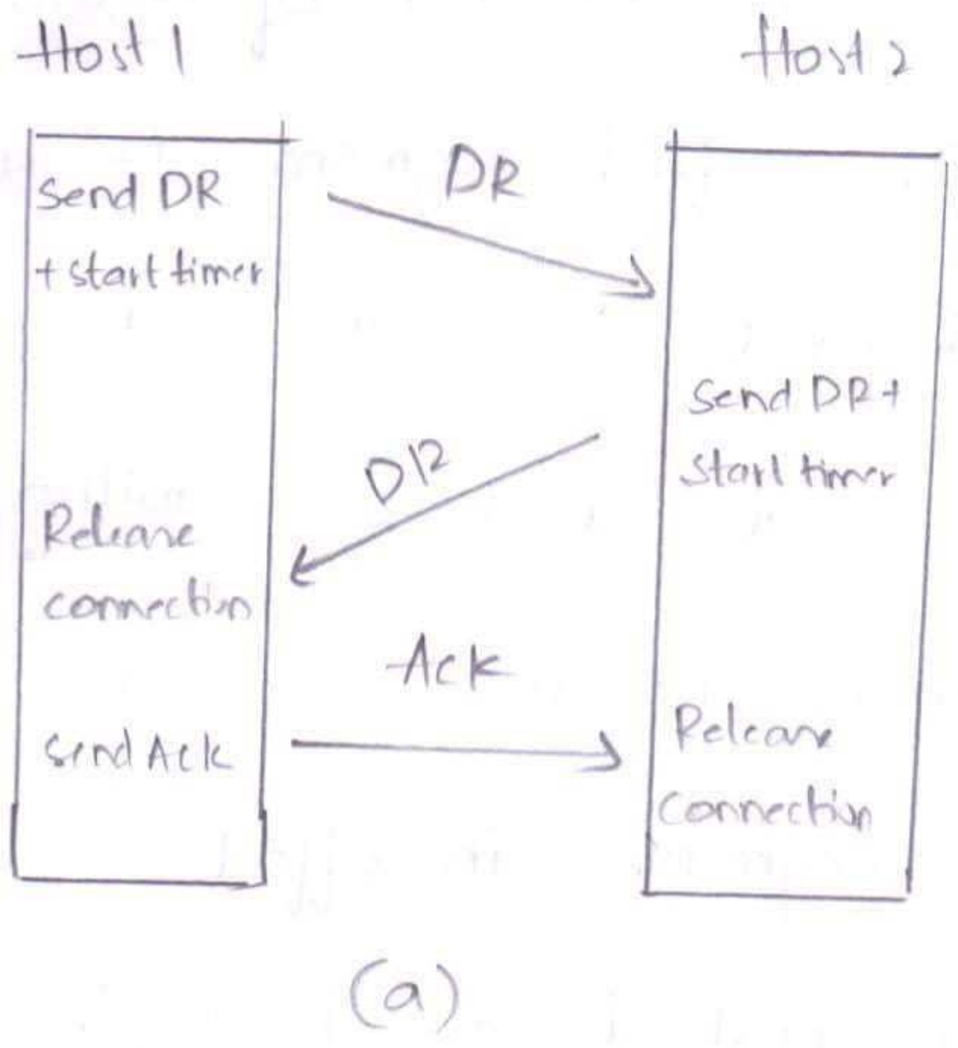
1. Restricted Network design
2. putting a hope counter in each packet
3. Timestamping each packet

* Suppose the commander of blue army #1 sends a message to blue army #2 commander. then he receives he agree and send his reply. but the attack will not happen because commander of blue army ‡2 does not know if reply got through.

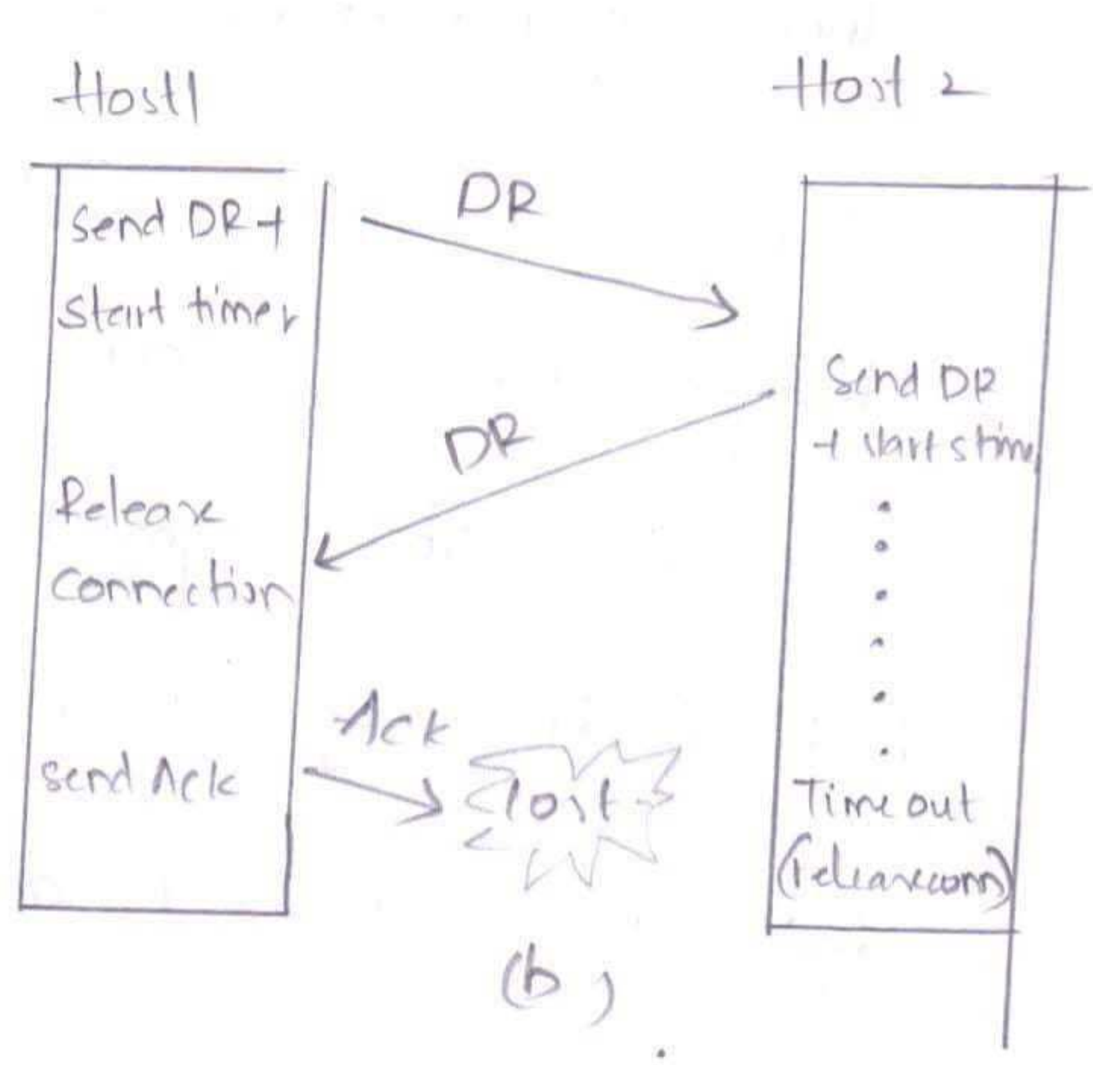
* Now, let us improve the protocol by making it a three-way handshake. The initator of the original proposal must acknowledge the response. Assume that the #Message are lost, #2 will get the Acknowledgement. After that, he does not know if his Acknowledgment got through, and if it did not, he knows that #2 will not attack.

* In fact, it can be proven that no protocol exists that works. Suppose their exists, the last message of the protocol is essential. or it is not, we can remove it untill we are left with a protocol in which every message is essential.

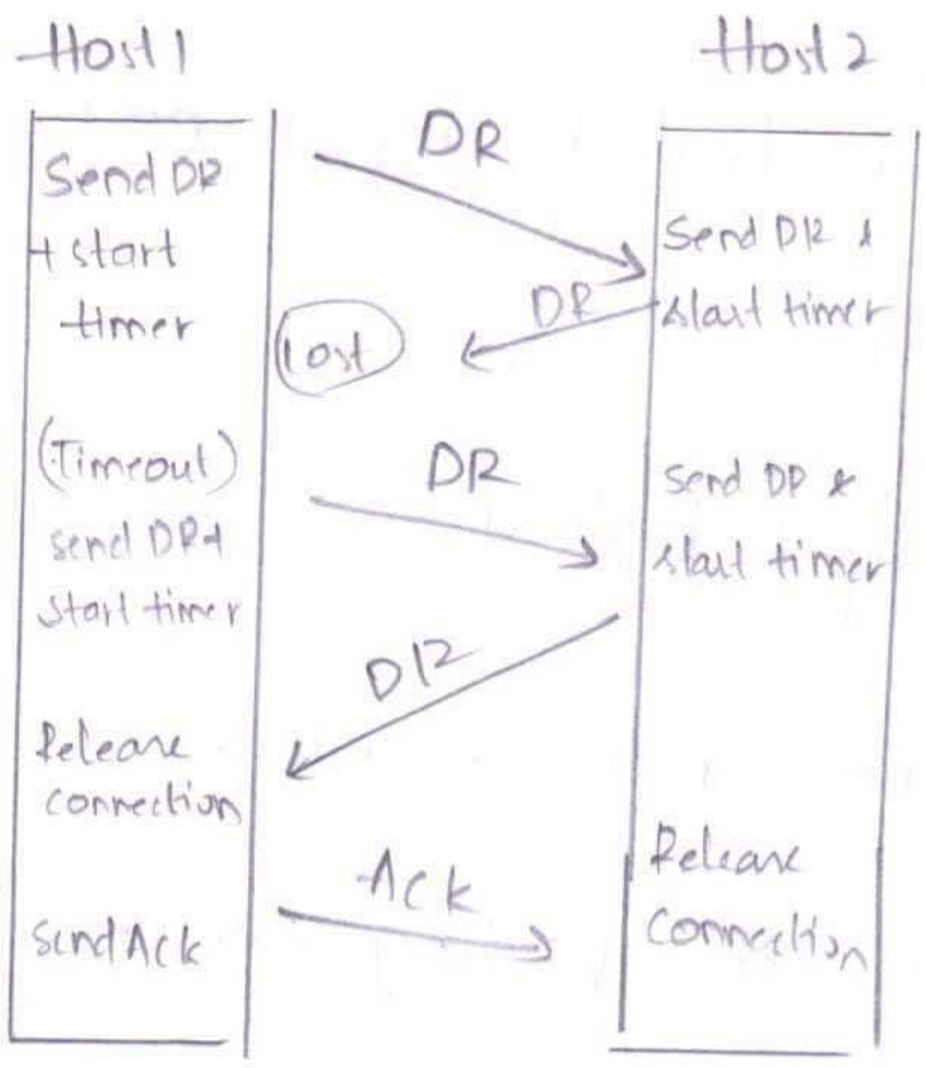
* To see the relevance of two-army problem to release connection. just substitute "disconnect" for "attack". If neither side prepared to disconnect.



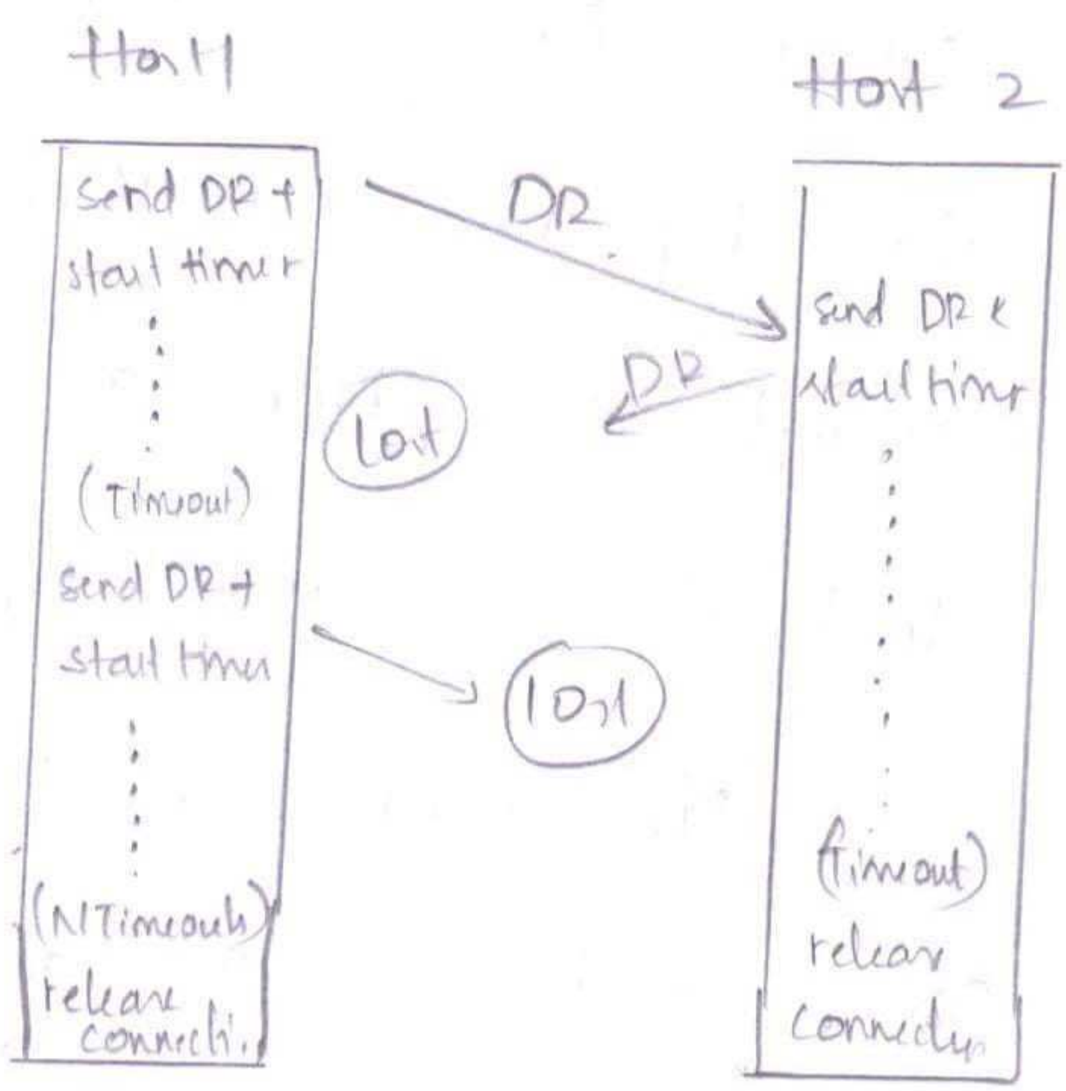
(a)



(b)



(c)



(d)

-Four Protocol Scenarios for releasing Connection

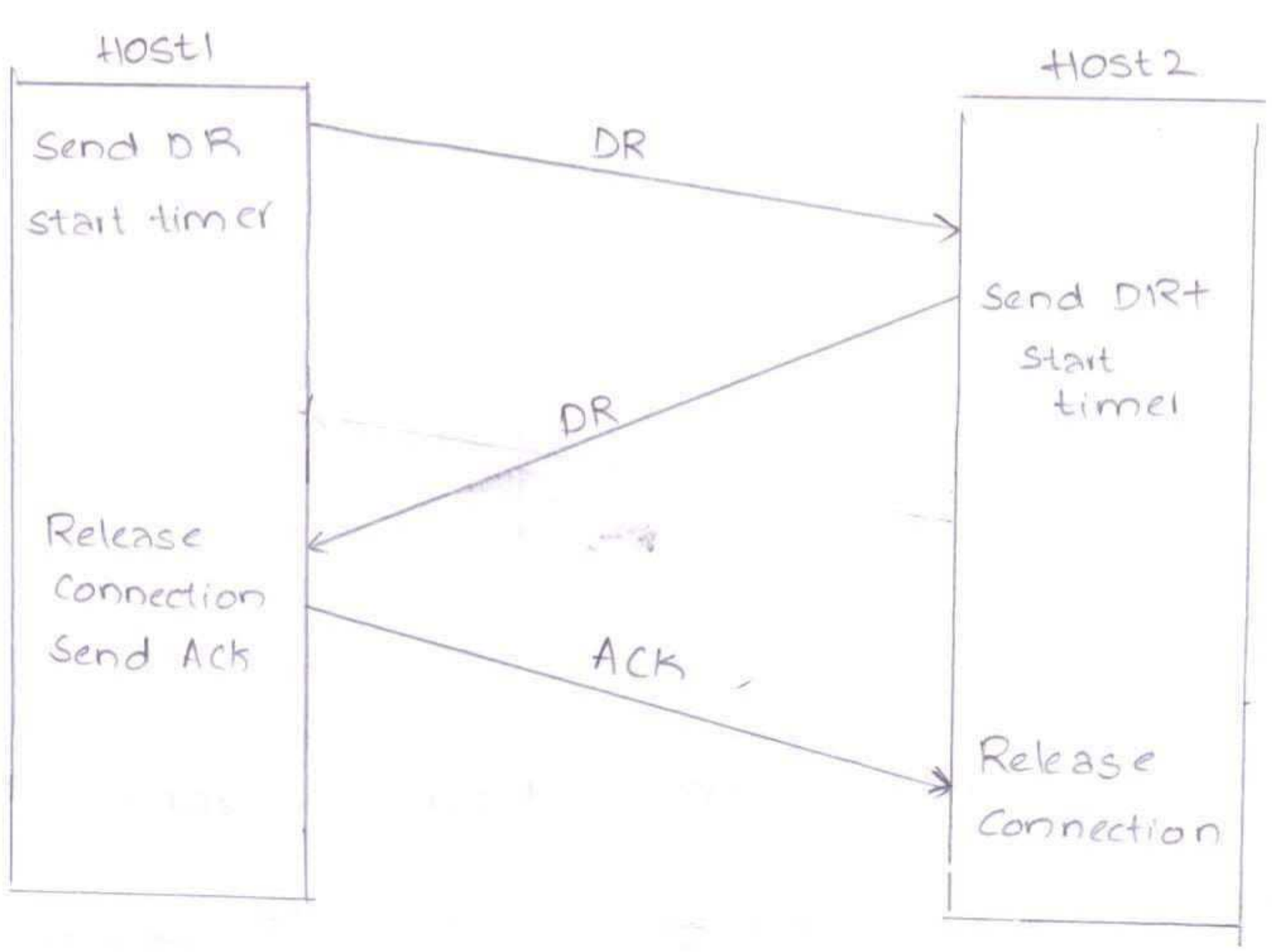
(a) Normal case of three way handshake

(b) -Final Ack lost

(c) Response lost

(d) Response lost & Subsequent DPi lost

Now, we see the four scenarios. of releasing using a three-way handshake, while this protocal is not infallible, it is usually Adequate



* We see the Normal case in which one of the users sends a DR (DISCONNECTION REQUEST) segment to intiate the connection release. When it arrives, the recipient sends back a DR segment and starts a timer, just in case DR is lost, When this DR arrives, the Original sendes back ACKs and releases the connection.

* Releasing Connection means that the transport removes the information about the connection from its table of currently open connections and signals

the first technique includes any method that prevents Packets from looping, combined with somaway of bounding delay including congestion over the largest possible path.

The second method consists of having the ~~loop count~~ hop count initialized to appropriate value and decremented each time the packet is forwarded. If count becomes zero, packet is discarded.

the third method requires each packet to bear the time it was created, with the routers agreeing to discard any packet older than sum agreed-upon time. this requires router clock,

We need to guarantee not only packet is dead but all the acknowledgements to it are also dead. The maximum packet lifetime is a conservative constant for a network which takes 120 seconds.

three-way handshake works in the presence of delayed duplicate control segments. The first segment is a delayed duplicate Connection Request from an old connection. This segment arrives at host 2 without host 1's knowledge. Host 2 reacts to this segment by sending host 1 an Ack segment, in effect asking for Verification that host 1 indeed trying to setup new Connection. When host 1 reject host 2's attempt to establish a connection, host 2 realizes that it was tricked by delayed duplicate.

The worst case is when both a delayed Connection Request and an Ack are floating around in the subnet, host 2 gets a delayed Connection Request and replies to it. At this point it is crucial to realize that host 2 has proposed ~~Used~~ using 'y' as initial Sequence number.

side will detect the lack of activity and also disconnect. If this rule is introduced, it is necessary for each transport entity to have a timer that it is stopped and then restarted whenever a segment is sent. If this timer expires a disconnect- the problem cannot be cleanly solved by the transport entities themselves.

* To see the importance of the application, consider that while TCP normally does asymmetric close. Many Web servers send the client a RST packet that causes an abrupt close of the connection that is more like an Assymetric close.

* When the Web service server is finished with its response, all the data has been sent in either direction. the server can send the client a warning and abruptly, shut the connection. If the client get warning, it will release its connection state then and there. If the client does not get the warning, it will eventually realize that the server is no longer talking to it and release the connection state. The data has been successfully transfereed in either case

user, one by one. Partway through the transmission the server crashes. When it comes back up, its tables are reinitialized, so it no longer knows precisely where it was.

* In an attempt to recover its previous status, the server might send a broadcast segment to all other hosts, announcing that it has just crashed and requesting that its clients inform it of the status of all open connections.

* Each client can be in one of two states: one segment outstanding, S_1 or no segments outstanding, S_0.

Based on only this state information, the client must decide whether to transmit the most recent segment.

* At first glance, it would seem obvious: the client should retransmit if and only if it has an acknowledged outstanding (i.e., is in state S_1) when it learns

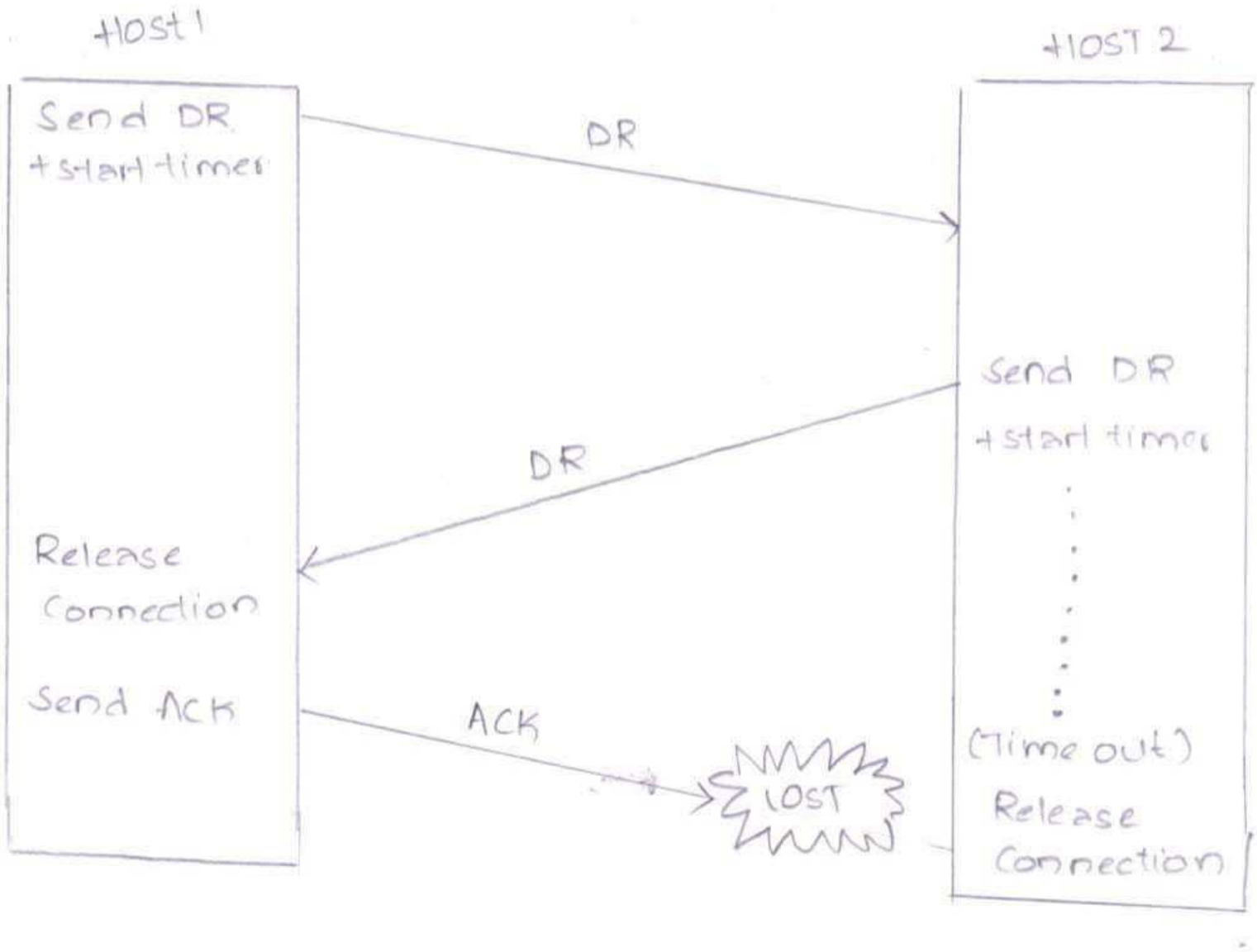
strategy used by receiving host

Strategy used by sending host	← First ACK then write →			← First write ACK then ACK →		
	A C (W)	A W C	C (A W)	C (W A)	W A C	W C (A)
Always retransmit	OK	DUP	OK	OK	DUP	DUP
Never retransmit	LOST	OK	LOST	LOST	OK	OK
Retransmit in S_0	OK	DUP	LOST	LOST	DUP	OK
Retransmit in S_1	LOST	OK	OK	OK	OK	DUP

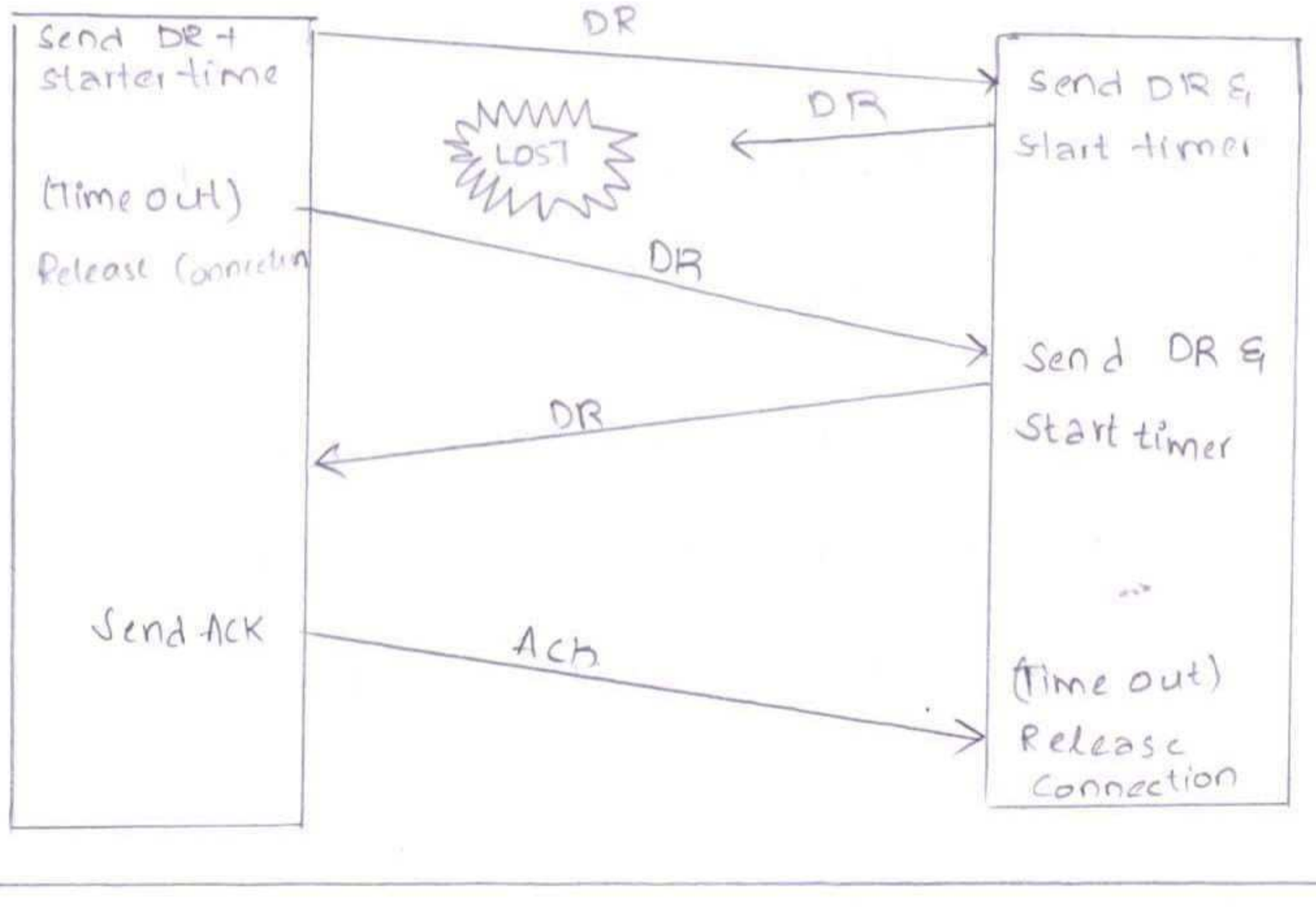
OK = Protocol functions correctly

DUP = Protocol generates a duplicate message

LOST = Protocol loses a message.



the situation is saved by timer. When it expries, the connection is released anyway. Consider the case of the second DR being lost. The user initiating the disconnection will not receive the expected response, will time out and will start all over again.



Consider this scenario. After the connection is established host 1 sends a segment that arrives properly at host 2. Then host 2 sends another segment. Unfortunately, host 2 issues a DISCONNECT before segment arrives. The result is that the connection is released and data are lost.

* Clearly, a more sophisticated release protocol is needed to avoid data loss i.e Symmetric Release.

Symmetric Release :-

* It treats the connection as two separate Undirectional connections and (release) requires each one to be released separately.
* In the same scenario, in which each direction is released independently of other one. Here, a host can continue to receive data even after it has sent a DISCONNECT SEGMENT.
* It does the job when each process has fixed amount of data to send and clearly knows when it has sent it.
* In other situations, determining that all the work has been done and connections should be terminated.

* Making the protocol more elaborate does not help. Even if the client and sever exchange several segments before the server attempts to write, so that the clients knows exactly what is about to happen, the client has no way of knowing whether a crash occured just before or just after the write.

* this conclusion is unescapable: under our ground rules of no simultaneous events - that is, separate events happen one after another not at the same time - host crash and recovery cannot be made transparent to higher layers.

* Put in more general terms, this result can be restated as "recovery from a layer N crash can only be done by layer N+1", and then only if the higher layer retains enough status information to reconstruct where it was before the problem occured.

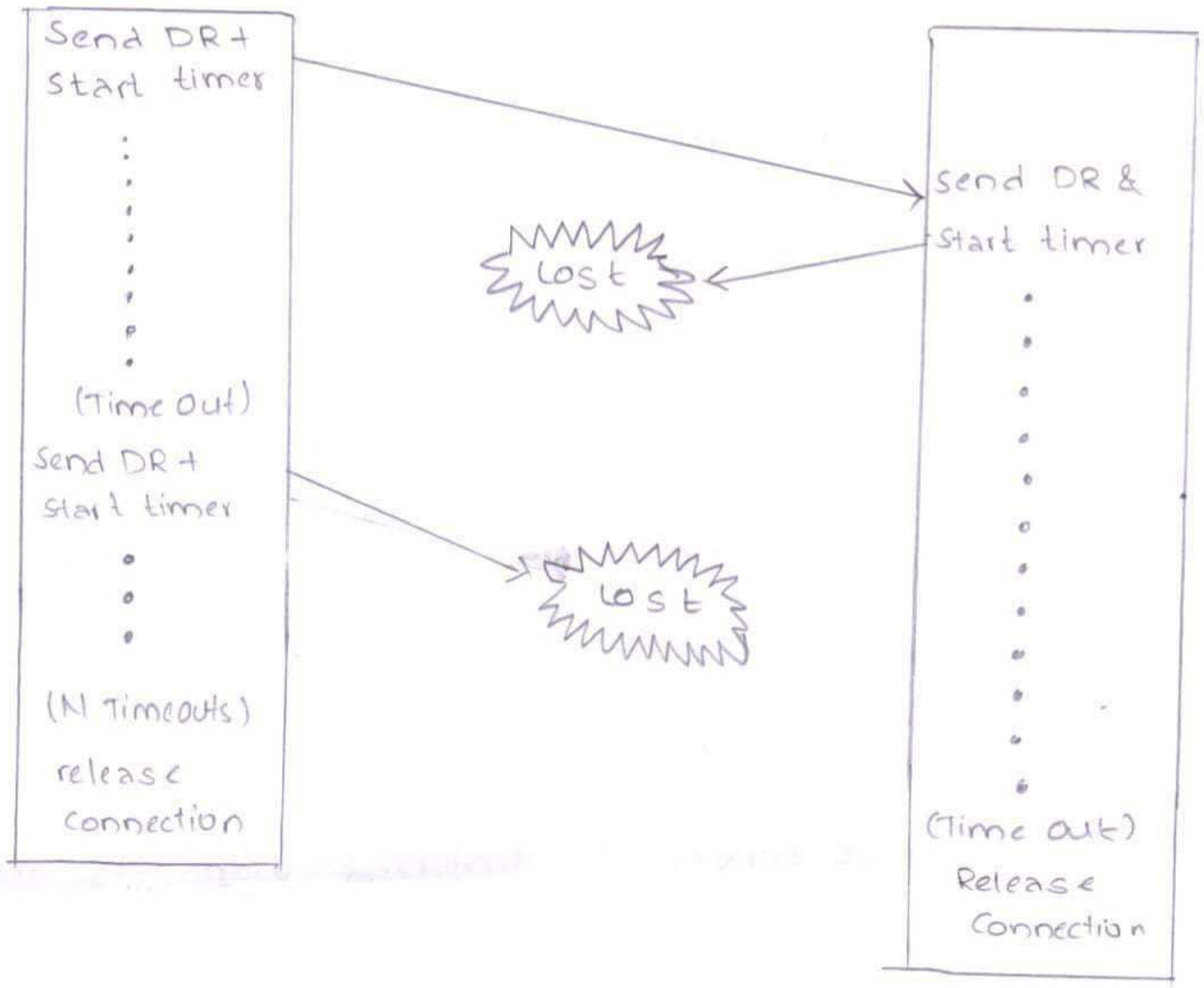
* This problem get us into the issue of what a so-called end-to-end acknowledgment

* At this point you may be thinking: "That problem can be solved easily. All you have to do is reprogram the transport entity to first do the write and then send the acknowledgement."

* Imagine that the write has been done but the crash occurs before the acknowledgement ~~has~~ can be sent. the client will be in state S_1 and thus retransmit, leading to an undetected duplicate segment in the output stream to the server application process.

* No matter how the client and server are programmed, there are always situations where the protocol fails to recover properly. The server can be programmed in one of two ways: acknowledge first or write first. The client can be programmed in one of four ways always retransmit the last segment, never retransmit the last segment,

Assuming that the second time no segments are lost and all segments are delivered correctly and on time.



It is same as the above seeoario except that now we assume all the repeated attempts to retransmit the DR also fail due to lost segment. After N retries, the sender just gives up and releases the connection mean while, the receiver times out and also exists.

* One way to kill off half open connections is to have rule segmen saying that if no segments have arrived for a certain number of seconds, the connection is automatically disconnected. That way, if one side over disconnects, the other

really means.

* In principle, the transport protocol is end-to end and not chained like the lower layers.

Crash Recovery

If hosts and routers are subject to crashes or connections are long-lived (ex: large software or media downloads), recovery from these crash-es becomes an issue. If the transport entity is entirely within the hosts, recovery from netwo-rk and router crashes is straightforward.

* The transport entities expect lost segments all the time and know how to cope with them by using retransmissions.
* A more troublesome is how to recover from host crashes.
* In particular, it may be desirable for clients to be able to continue working when servers crash and quickly reboot.
* To illustrate the difficulty, let us assume that one host, the client, is sending a long file to another host, the file server, using a simple stop-and-wait protocol. This transport layer on the server just passes the incoming segments to the transport

retransmit only in state s_0, or retransmit only in state s_1. This gives 8 combinations but as we shall see, for each combination there is some set of events that make the protocol fail

* Three events are possible at the server:

  an acknowledgement (A),
  writing to the output process (W),
  and crashing (C).

* The three events can occur in six different orderings

  AC(W), AWC, C(AW), C(WA), WAC, and WC(A).

  Where the parentheses are used to indicate that neither A nor W can follow C (i.e., once it has crashed, it has crashed).

* Eigh combinations of client and server strategies and the valid event sequences for each one are given. Notice that for each strategy there is some sequence of events that causes the protocol to fail.

of the crash.

* However, a closer inspection reveals diffi-culties with this naive approach. Consider, for example, the situation in which the server's transport entity first sends an acknowledgment and then, when then acknowledgement has been sent, writes to the application process.

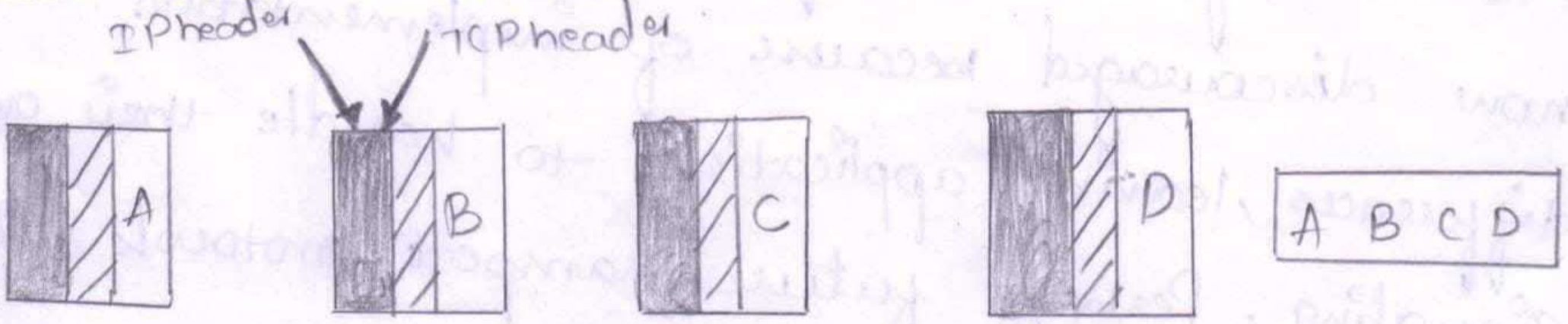
* Writing a segment onto the output stream and sending an acknowledgement are two distinct events that cannot be done simultaneously.

* If the crash occurs after the acknowledgement has been sent but before the write has been fully completed, the client will receive the acknowledgement and thus be in state so when the crash recovery announcement arrives. The client will therefore not retransmit, thinking that the segment has arrived. This decision by the client leads to a missing segment.

Port	Protocol	Use.
20,21	FTP	file transfer
25	SMTP	Email
80	HTTP	World wide Web
443	HTTPS	Secure Web
543	RTSP	Media Player Control.
631	IPP	Printer sharing.

All TCP connections are full duplex and point-to-point. full duplex means that traffic can go in both directions at same time. TCP does not support multicasting or broadcasting.

A TCP connection is a byte stream, not a message stream. Message boundaries are not preserved end-to-end. There is no way for the receiver to detect the unit(s) in which the data were written, no matter how hard it tries.



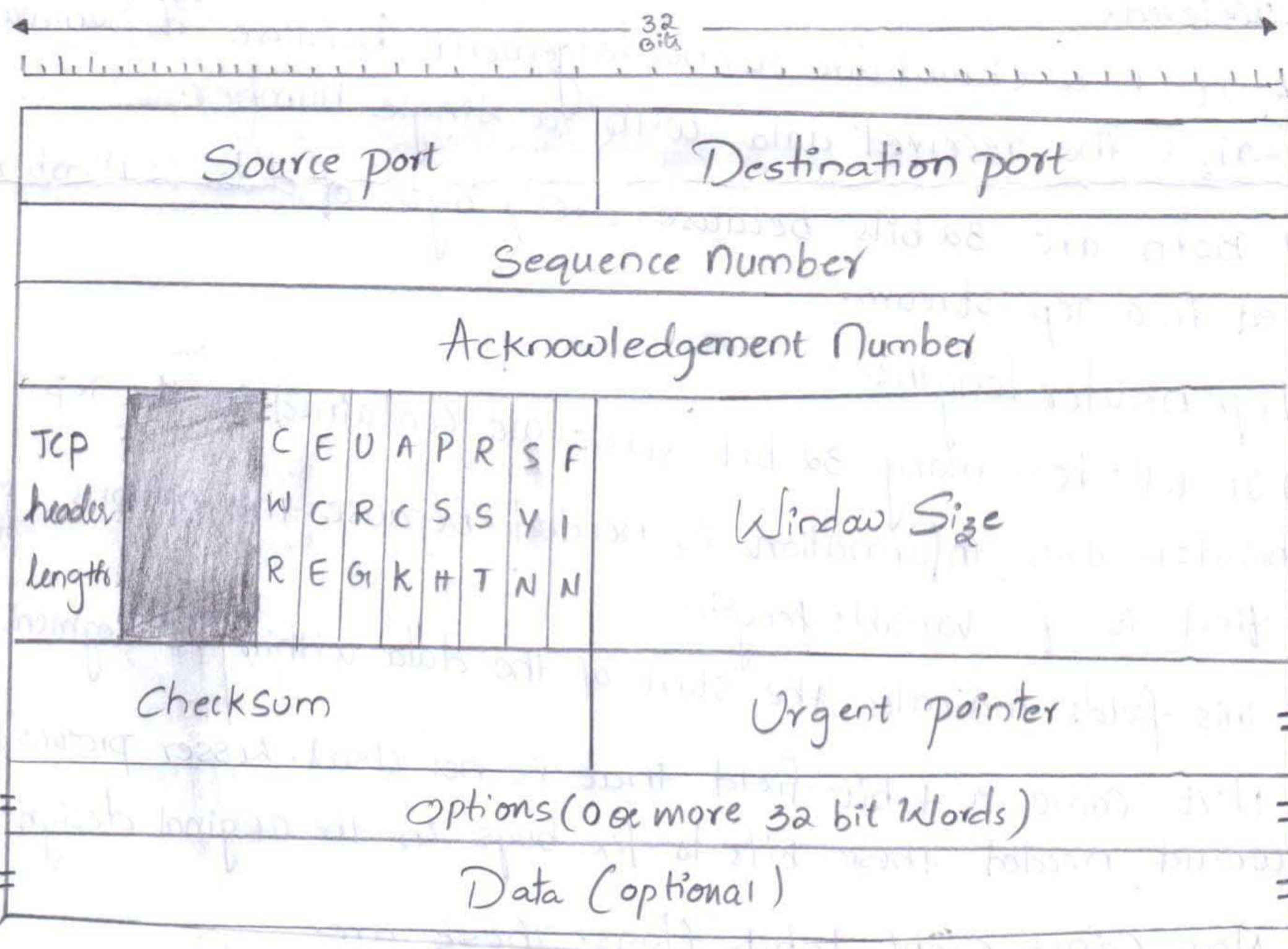
(a)

(a). four 512-byte segments sent a separate IP datagram.

Tcp Segment Header:-

* The below figure shows the layout of a TCP segment. Every Segment begins with a fixed format, 20-byte header. The fixed header may be followed by header options. If any, up to 65,535-20-20=65,495 data bytes may follow, where the first 20 refer to the Ip header and Second to the Tcp header.
* Segment without any data are legel and are Commonly used for acknowledgements and control Messages.

Tcp header:



Source port & destination port:- These identify the local end points of the connection.

* A Tcp port plus its host's Ip address forms a 48-bit unique end point.

The Internet Transport Protocols (UDP).

With RPC, Passing Pointers is impossible. because the client and server are in different address spaces. In some cases tricks can be used to make it possible to Pass Pointers. The server stub then creates a Pointer to K and Passes it to the server Procedure, just as it expects. When the server Procedure returns control to the Server stub, the latter Sends K back to the client, where the new K is Copied over the old one, just in case the server changed it. in effect, the Standard Calling Sequence of Call-by-reference has been replaced by Call-by-Copy-restore. unfortunately, this trick does not always work, for example, if the Pointer Points to a graph or other Complex data structure.

A Second Problem is that in Weakly type Languages, like C, it is Perfectly Legal to write a Procedure that Computes the inner Product of two vectors (arrays). Without specifying how Large either one is. Each could be terminated by a Special Value known only to the Calling and called Procedures. Under these circumstances it is essentially impossible for the client Stub to marshal the Parameters: it has no way of determining how Large they are.

* The Source & destination end points together identify the connection.
* This Connection identify is called a 5 tuples. because it consists of five pieces of information.
* The protocol (TCP), Source Ip and Source port, destination Ip and destination port.

Sequence Number & Acknowledgement Number:-

* These fields performs there Usual functions. later specifies the next in-order byte expected, not the last byte correctly recieved.
* It is a Cumulative acknowledgement because it Summarizes the received data with a single number.
* Both are 32 bits because every byte of data is numbered in a Tcp stream.

Tcp header length:-

⇒ It tells how many 32-bit words are contained in the Tcp header. This information is needed because the options field is of Variable length.

* this fields indicates the start of the data within the segment.
* Next come a 4-bit field that is not used. Lesser protocols would needed these bits to fix bugs in the original design.
* Now comes eight 1-bit flags. Those are.

CWR & ECE:-

* These are Used to signal Congestion, when ECN (Explicit Congestion Notification) is Used, as specified in RFC 3168.

* ECE is set to Signal an ECN-Echo to a TCP Sender to tell it to slow down when the TCP receiver gets a Congestion indication from the network.
* CWR is set to signal Congestion Window Reduced from the TCP sender to the TCP Reciever, so that it knows the sender has slowed down and can stop sending the ECN-Echo.

URG:-

* It is set to 1 if the Urgent pointer is in Use.
* The Urgent pointer is used to indicate a byte offset from the current sequence number at which urgent data are to be found.
* This facility is in lieu of interrupt Messages.
* this facility is a bare-bones way of allowing the sender to singal the reciever without getting Tcp itself involved, in this interrupt.

ACK:-

* Ack bit is set to 1 to indicate the Akhowledgement Number is valid.
* If Ack is 0, the segment does not Contain acknowledge -ment, so the Ack number field is ignored.

PSH:-

* Psh indicates pushed data.
* Here the reciever is requested to deliver the data to the applications upon arrival and not buffer it until a full buffer has been recieved.

RST:-

* RST bit is Used to reset a Connection that has become Confused due to a host Crash or some other reasons.

Packet. to the. Server Stub. Finally steps is the. Server. Stub Calling the Server Procedure with the. unmarshaled Parameters. The reply traces the same Path. in the other direction.

* It is also used to reject an invalid segment or refuse an attempt to open a connection.

SYN:-

* SYN bit used to Establish connections.
* If SYN=1 and ACK=0 to indicate that the piggy back ACK field is not in Use.
* If SYN=1 and ACK=1, the SYN bit is used to denote both CONNECTION REQUEST and CONNECTION ACCEPTED.

FIN:-

* FIN bit is used to release a connection.
* It specifies that the Sender has no more data to transmitt. however, after closing a Connection the closing process may continue to receive data indefinitely.
* Both SYN and FIN Segments have sequence numbers and are thus guaranteed to be processed in the correct order.

Check Sum:-

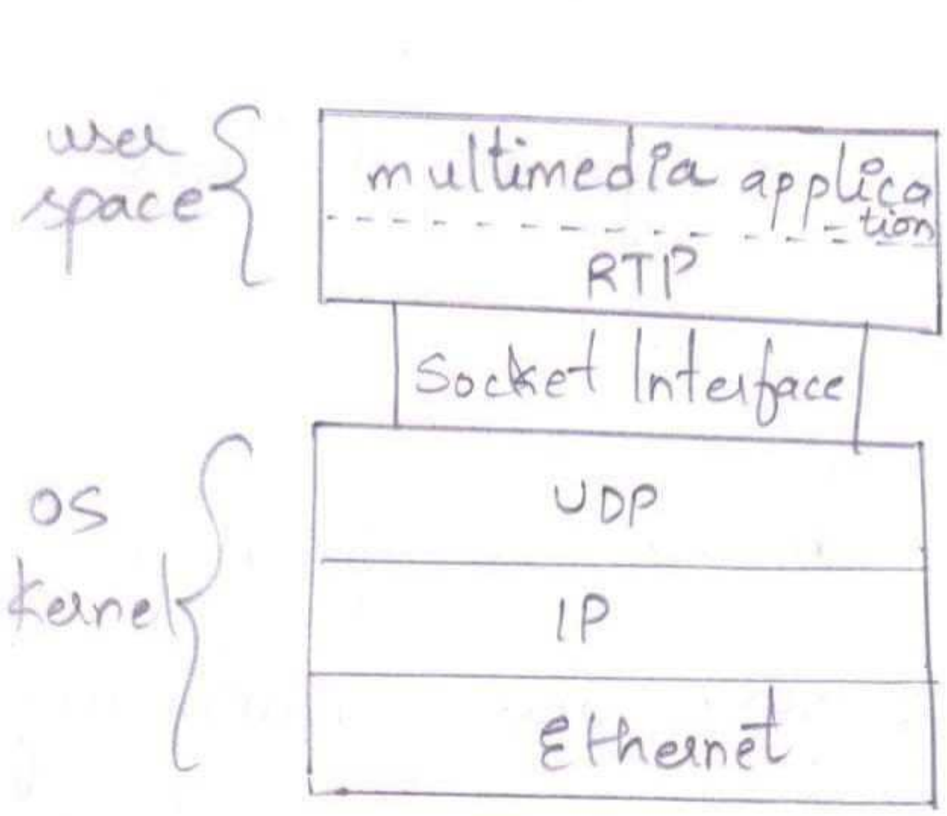
* A checksum is also provided for Extra reliability. It checksums the header, the data, and a Conceptual has pseudo header in exactly the same way as UDP.
* If Except that the pseudo header has the protocol number for Tcp and the checksum is Mandotory.

Window Size:-

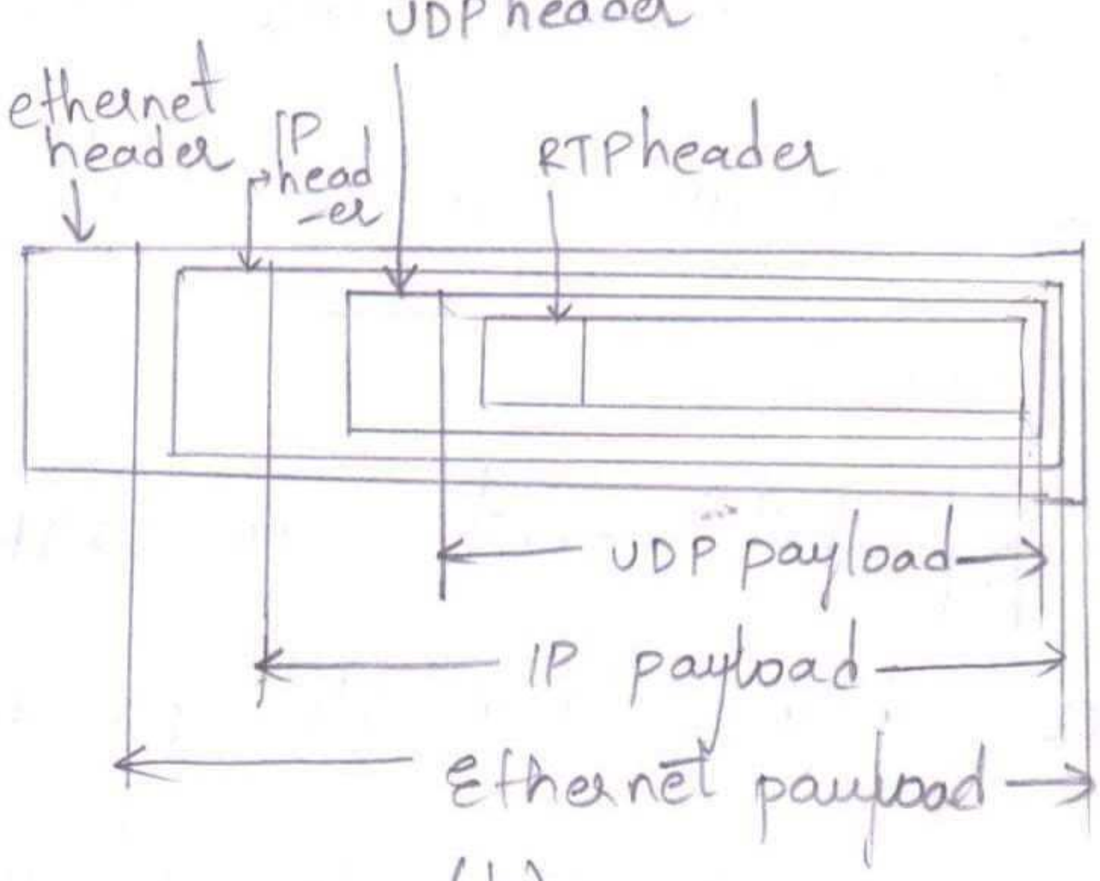
* Flow Control in Tcp is handled using a Variable-sized sliding window.
* This window size field tells how many bytes may be sent starting at the byte acknowledge.
* A window size field of 0 is legal says that the bytes includes Ack number - 1 has been recieved.

6.4.3 Real-Time Transport Protocols

- client server RPC is one area in which UDP is widely used.
- In particular, as internet telephony, music on demand, video conferencing are becoming popular, it is clear that a generic real-time transfer protocol would be a good idea.
- RTP (real-time-transfer protocol) born. It is described in RFC 3550 & now it is in widespread use for multimedia.
- 2 aspects of RTP are
  - i) The RTP protocol for transporting audio and video data in packets
  - ii) Second is the processing that takes place at reciever to play audio & video.



(a)



(b)

* It is also used to reject an invalid segment or refuse an attempt to open a connection.

SYN:-

* SYN bit used to Establish connections.
* If SYN=1 and ACK=0 to indicate that the piggy back ACK field is not in Use.
* If SYN=1 and ACK=1, the SYN bit is used to denote both CONNECTION REQUEST and CONNECTION ACCEPTED.

FIN:-

* FIN bit is used to release a connection.
* It specifies that the Sender has no more data to transmitt. however, after closing a Connection the closing process may continue to receive data indefinitely.
* Both SYN and FIN Segments have sequence numbers and are thus guaranteed to be processed in the correct order.

Check Sum:-

* A checksum is also provided for Extra reliability. It checksums the header, the data, and a Conceptual has pseudo header in Exactly the same way as UDP.
* If Except that the pseudo header has the protocol number for Tcp and the checksum is Mandotory.

Window Size:-

* Flow Control in Tcp is handled using a Variable-sized sliding window.
* This window size field tells how many bytes may be sent starting at the byte acknocoledge.
* A window size field of 0 is legal says that the bytes includes Ack number - 1 has been recieved.

* But the receiver has not had a chance to consume the data.
* Later receiver can grant permission to send by transmitting a segment with the same Ack number and a non zero window size.

Option field:-

⇒ It provides a way to add extra facilities not Covered by the regular header.

⇒ the options are of Variable length, fill a multiple of 32 bits by using padding with zero's, and may extend to 40 bytes to accommodate the longest Tcp header that can be specified.

⇒ Some options are carried, when a connection is established to inform the other side capabilities.

⇒ other options are carried on packets during the life time of the connection.

⇒ Each options has a Type-length-Value encoding.

⇒ option is the one that allows each host to specify the mss (maximum segment size) it willing to accept.

⇒ The window scale option allows the sender and reciever to negotiate a window scale-factor at the start of a connection.

⇒ The time stamp option carries a time stamp sent by the sender and echoed by the receiver. It included in Every packet, once its use is established during connection setup, and used to Compute round-trip time samples that are used to estimate when a packet has been lost.

⇒ Paws (protection Against Wrapped sequence numbers) Scheme discards arriving segments with old timestamps to prevent this problem.

No Message Passing is visible to the application. Programmer. this technique is known as RPC (Remote Procedur call) and has become the basis for many networking applications. Traditionally, the calling Procedure is known as the client and the called Procedure. is known as. the server, and we will use those names. here too. The client program must be bound with a small. library Procedure, called the client stub, that. represents. the. Server. Procedure in the client's address space. Similarly, the server is bound with a. Procedure called the. Server Stub. These Procedures. hide the fact that the Procedure. Call from the client to the server is not Local. Packing the. Parameters is Called Marshaling. Step1 is the client calling the client stub. This call is a Local Procedure call, with the. Parameters. Pushed onto the stack. in the normal way. Step 2 is the client stub Packing the. Parameters is called. Marshaling. Step 3 is the Operating System Sending the. Message from the. client Machine to the Server Machine. Step 4. is the Operating system Passing the incoming

- RTP normally runs in user space over UDP. It operates as follows
- The multimedia applications consist of multiple audio, video, text & other streams, these are fed into library which is in userspace along with application
- The library multiplexes the streams and encodes them in RTP packets, which stuffs into a socket, on OS side of socket, UDP packets are generated to wrap the RTP packets & handed to IP for transmission over a link such as Ethernet. The reverse process happens at reciever.
- packet nesting is shown in (b) and protocol stack is defined in (a)

RTP:

- Basic function of RTP is to multiplex several real-time data streams onto single stream of UDP packets
- These stream can be sent to single (or) multiple destinations
- Each packet sent in RTP stream is given a number higher than its predecessor. This numbering allows the destination to determines if any packet is missing.

SACK (selective Acknowledgement) option lets a receiver tell a sender the ranges of sequence number that is have received.

* With SACK, the sender is explicitly aware of what data the receiver has and hence can determine what data should be retransmi -tted.

* SACK is defined in RFC 2108 and RFC 2883 and is increasingly used.

Tcp Connection Establishment:-

* Connections are established in Tcp by means of the three-way handshake discussed.

* To Establish a connection, one side say the Server passively waits for an incoming connection by executing the LISTEN and ACCEPTS primitives in that order.

* When this segment arrives at the destination the Tcp entity there checks to see if there is a process that has done a LISTEN on the port given in the Destination port field. If not, it sends a Tcp segment reply with the SYNS RST bit on to reject the connection.

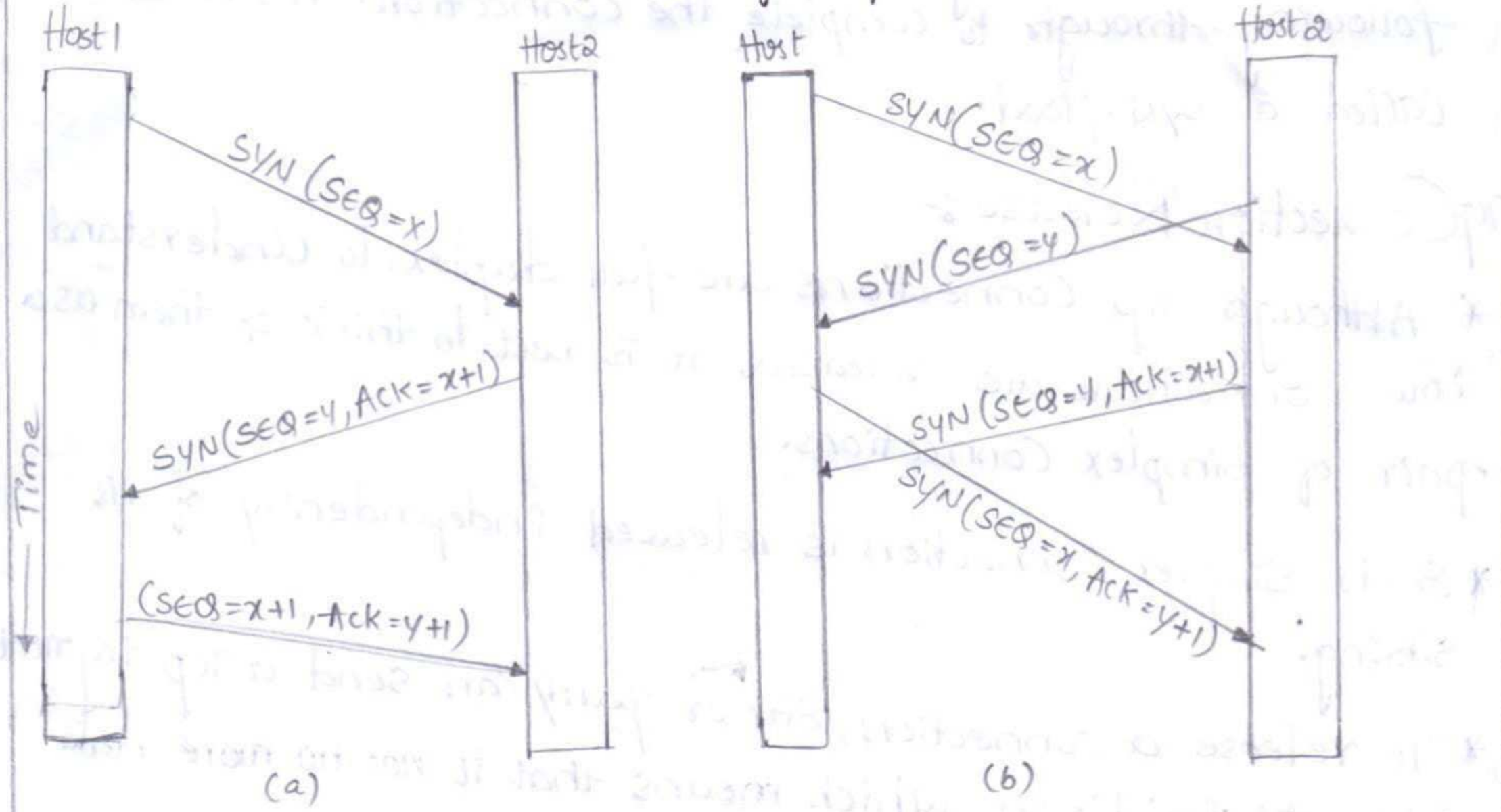
* the connection primitive sends a Tcp segment with the SYN bit on And Ack bit off and waits for a response.

* If some process is listening to the port, that process is given the incoming Tcp segment. It can Either accept or reject the Connection. If it accepts, an acc segment is Sent back.

* The SYN segment sequence of Tcp segments sent in the normal case.

A third Problem is that it is not always Possible to deduce the types of the Parameters, not even from a formal Specification or the code itself. An example is Printf, which may have any number of Parameters (at least one), and the Parameters, not even from a formal Specification or the code itself. An example is Printf, which may have any number of Parameters, and the Parameters can be an arbitary mixture of integers, Shorts, Longs, characters, strings, floating-Point numbers of Various Lengths, and other types. Trying to call Printf as a remote procedure would be Practically impossible because C is so Permissive However, a rule Saying that Rpc can be used Provided that you do not Program in C (or C++) would not be Popular with a lot of programmers form of Procedure Cells. Such an arrangement makes network applications makes Easier to Program and more familiar to deal with. For Example. Just imagine a Procedure named get-ip-address (host-name) that works by sending a UDP Packet to a DNS Server and waiting for the reply, timing out and trying ages and one is not forthcoming quickly enough.

The Internet Transport protocals · TCP ·



a) Tcp Connection establishment in the normal case.

b) simultaneous connection establishment on both sides.

* These event that two hosts simultaneously attempt to establish a connection b/w the same two sockets.
* The result of these events is that just one connection is established, not two because connections are identified by their end points.
* If the first setup results in a connection identified by (x,y) and the second one does, too only one table entry is made. namely for (x,y).
* Recall that the initial sequence number choosen by each host should cycle slowly. rather than be a constant such as 0. This rule is to protect against delayed duplicate packets.

* If packets are missing, it may skip a videoframe if packets are carrying video data (or) approximate the missing value by interpolation if it is audio data.

- RTP provides several profiles for internetworking, for each profile multiple encoding is allowed.

Eg: single audio stream may be encoded as 8 bit PCM samples at 8kHz using delta encoding, predictive encoding so on.

- RTP header field in which the source can specify the encoding but is otherwise not involved in how encoding is done.

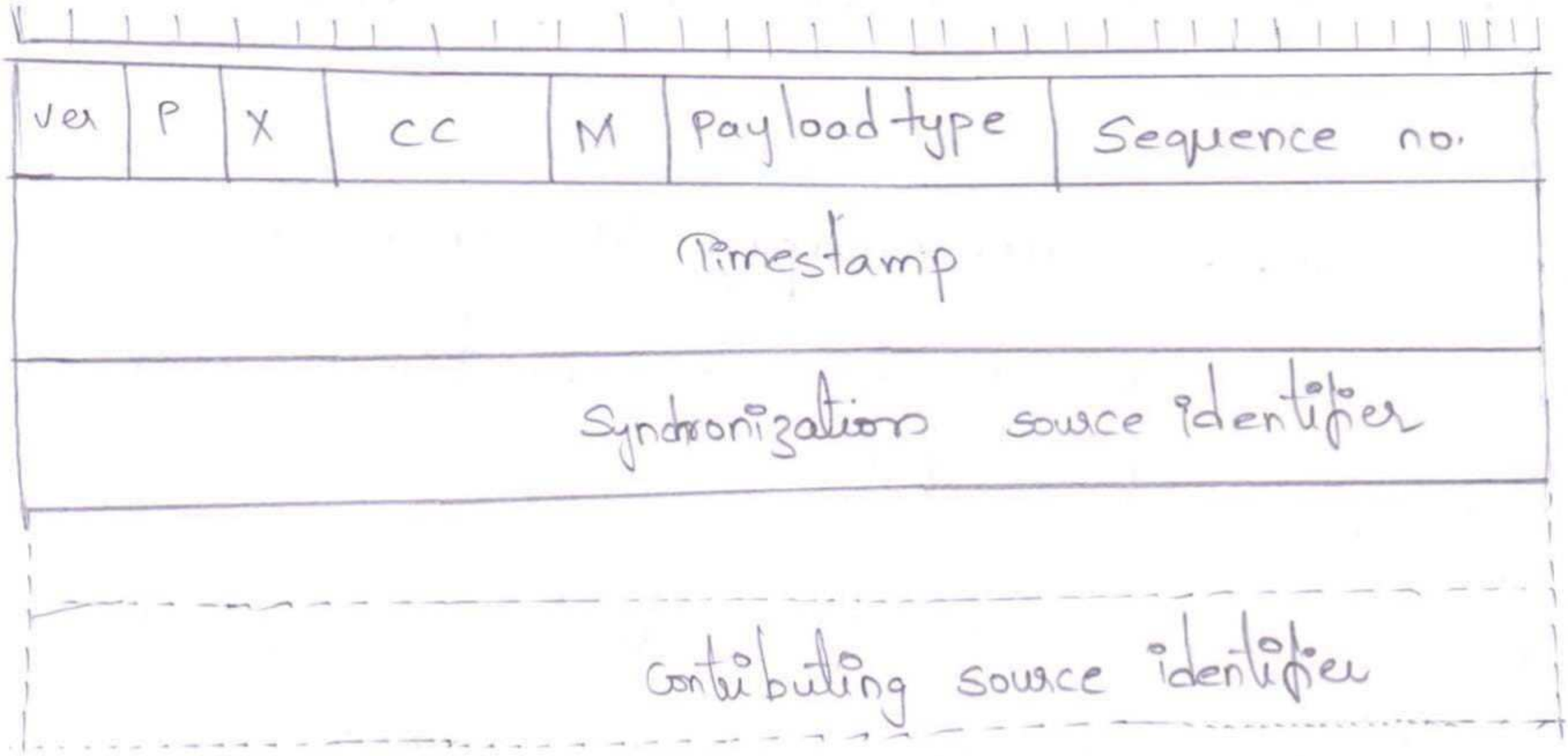
- RTP provides time stamping. The idea is to allow the source to associate a timestamp with first sample in each packet.

- Timestamping reduce the effects of variation in n/w delay but also allows multiple streams to be synchronized with eachother.

Eg: a digital television prgm might have a video stream & 2 audio streams, 2 audio streams could be for stereo could be for stereo broadcasts (or) for

handling films with digital language soundtracks.

- The RTP header is illustrated in below figure. It consist of 3, 32-bit words & potentially some extensions
- The first word contains version field, which is already at 2.
- P bit indicates that the packet has been padded to a multiple of 4 bytes.
- The last byte tells how many bytes are added.
- X indicates extension header is presents, fast word of extension gives length.



RTP header.

- CC field tells how many contributing sources are present. from 0 - 15.
- M bit tells application marker.

* The host by sending a stream of SYN segments and never following through to complete the connection. this attack is called a SYN flood.

TCP Connection Release :-

* Although Tcp Connections are full duplex, to understand how connections are released it is best to think of them as a pair of simplex connections.
* Each Simplex Connection is released independently of its sibling.
* To release a connection, either party can send a Tcp segment with the FIN bit set, which means that it has no more data to transmit.
* When the FIN is ack, the direction is shut down for new data. Data may flow in other directions.
* When both directions have been shut down the connection is released.
* Normally four Tcp segments are needed to release a connection one FIN and one Ack for each direction. it is possible for the first Ack and the second FIN is continued in the same segment.
* To avoid the two-army problem timers are used. If a response to a FIN is not forthcoming within two Maximum packet life time- the sender of the FIN releases the connection.

TCP Connection Release

Although TCP connections are full duplex, to understand how connections are released it is best to think of them as a pair of simplex connections.

→ Each simplex connection is released independently of its sibling. To release a connection, either party can send a TCP segment with the FIN bit set, which means that is has no more data to transmit.

→ when the FIN is acknowledged, that direction is shut down for new data. Data may continue to flow indefinitely in the orther direction, however, when both directions have been shut down, the connection is released.

→ Normally, four TCP segments are needed to release a connection:

one FIN and one ACK for each direction.

However, it is possible for the first ACK and the second FIN to be contained in the same segment, reducing the total count to three.

→ Just as with telephone calls in which both people say goodbye and hang up the phone simultaneously. both ends of a TCP connection may send FIN segments at the same time. These are each acknowledged in the usual way, and the connection is shut down.

well q to cut back the data rate when there is trouble in n/w.

- An issue with providing feedback is that the RTCP reports are sent to all participants. for multicast application with a large group, the bandwidth used by RTCP would quickly grow large. To prevent this from happening, RTCP senders scale down the rate of their reports to collectively consume no more than, say.

- RTCP also handles interstream synchronization. The prblm is that different streams may use different clocks, with different granularities q different datarates. RTCP can be used to keep them in sync.

- Finally RTCP provides a way for naming the various sources. This information can be displayed on reciever screen

→ There is in fact, no essential difference between the two hosts releasing sequentially or simultaneously.

→ To avoid the two-army problem, timers are used.

→ If a response to a FIN releases the connection. The other side will eventually notice that nobody seems to be listening to it anymore and will time out as well.

→ while this solution is not perfect, given the fact that a perfect solution is theoretically impossible it will have to do. In practice, problems rarely arise.

→ TCP 3 way handshake based setup and connection release.

TCP provides a reliable end to end service that delivers packets over the internet. The three way handshake to establish a TCP. Data transfer using the byte oriented sequence numbers.

PLAYOUT WITH BUFFERING AND JITTER CONTROL

- once the media information reaches the receiver, it mus
be played out at right time. In general this will not be
the at which RTP packet arrived at the receiver.
- even if packets are injected with exactly right inter
-vals b/w them at sender, they will reach the recei
-ver with different relative time. This delay is called
jitter.
- The solution to this is to buffer packets at receiver
before they are played out to reduce jitter. In the
below diagram we see stream of packets are delivere
with sustantial amount of jitter. Packet 1 is sent from
server at time t=0 and arrives at t=1 sec packet 2
undergoes more delay and takes 2 sec to arrive, they
are buffered on client machine.
- At t=10sec playback begins. 3 through 6 have
been buffered so that they can be removed from
buffere at uniform intervals for smooth play.

TCP Connection Management Modelling

The steps required to establish and release connections can be represented in a finite state machine with the 11 states. In each state, certain events are legal. when a legal event happens, some action may be taken. If some other event happens, an error is reported.

→ each connection starts in the closed state. It leaves that state when it does either a passive open (LISTEN) or an active open (connect). If the other side does the opposite one, a connection is established and the state becomes established.

→ Connection release can be initiated by either side. when it completes, the state returns to closed.

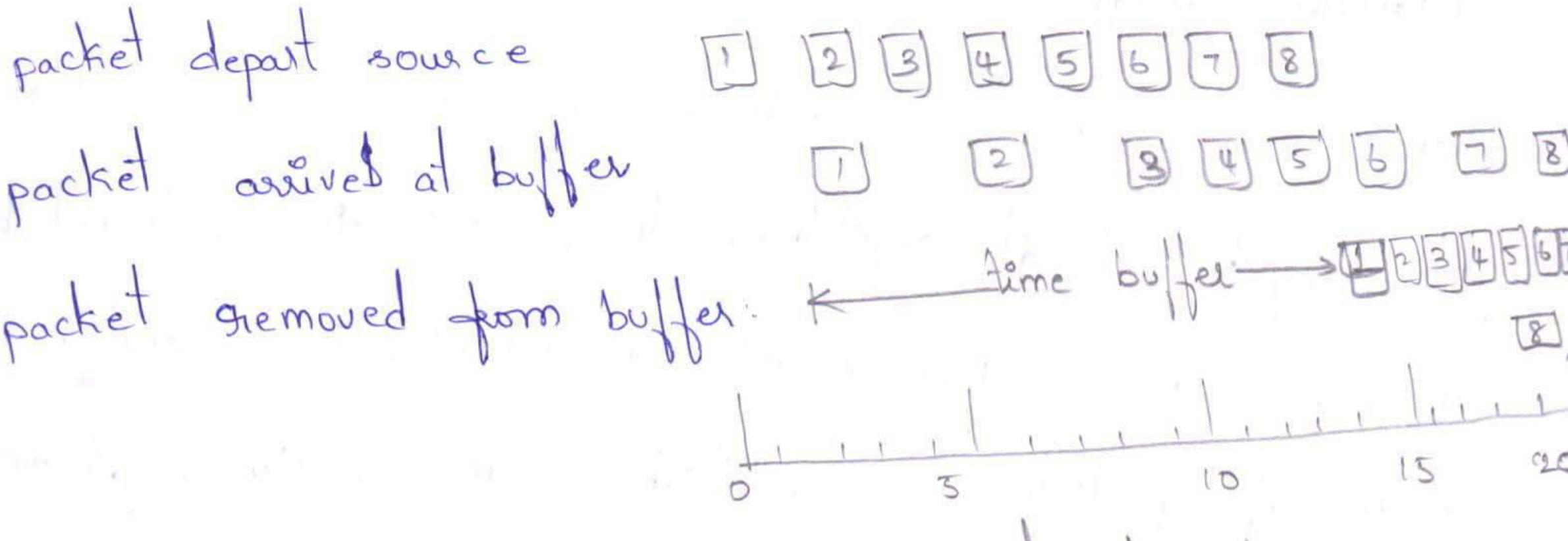
→ The finite state machine, the common case of a client actively connecting to a passive server is shown with heavy lines

→ solid for the client

→ dotted for the server

→ The lightface lines are unsual event sequences.

The states used in the TCP connection management	
State	Description
CLOSED	No connection in active or pending
LISTEN	The server is waiting for an incoming call
SYN RCVD	A connection request has arrived
SYN SENT	The application has started to open a connection
ESTABLISHED	The normal data transfer state.
FIN WAIT 1	The application has said it is finished
FIN WAIT2	The other side has agreed to release
TIME WAIT	wait for all packets to die off
CLOSING	Both sides have tried to close simultaneously
CLOSE WAIT	The other side has initiated a release
LAST ACK	wait for all packets to die off



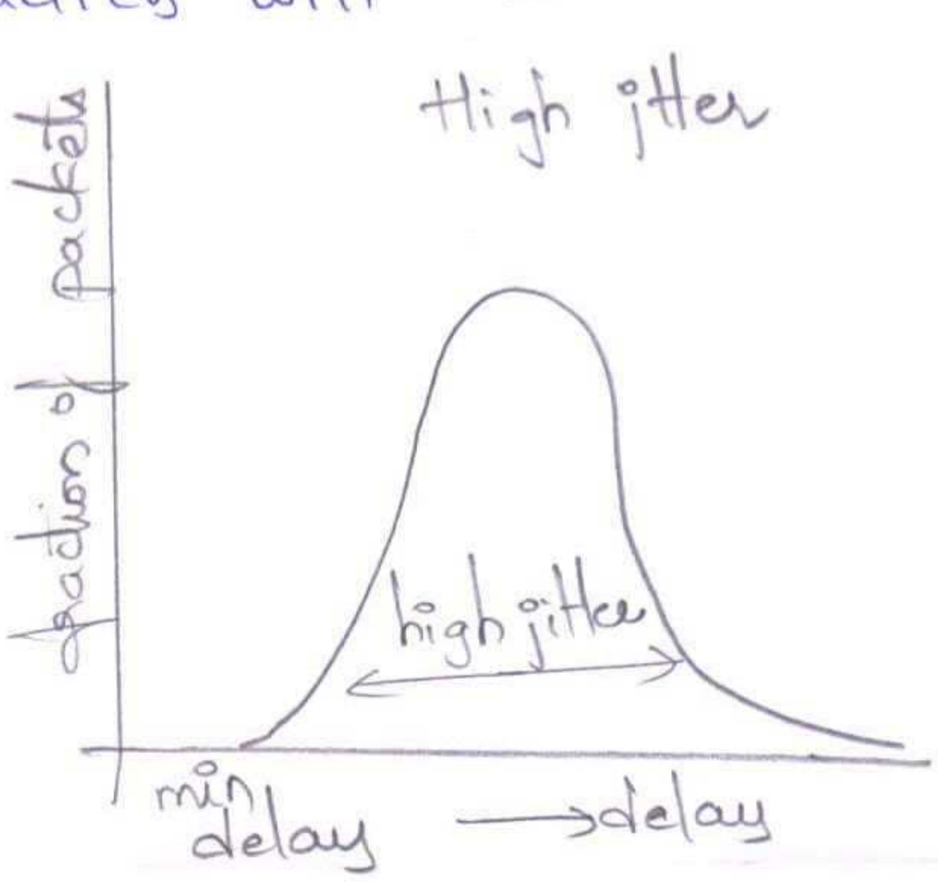
Smoothing the ouput streams.

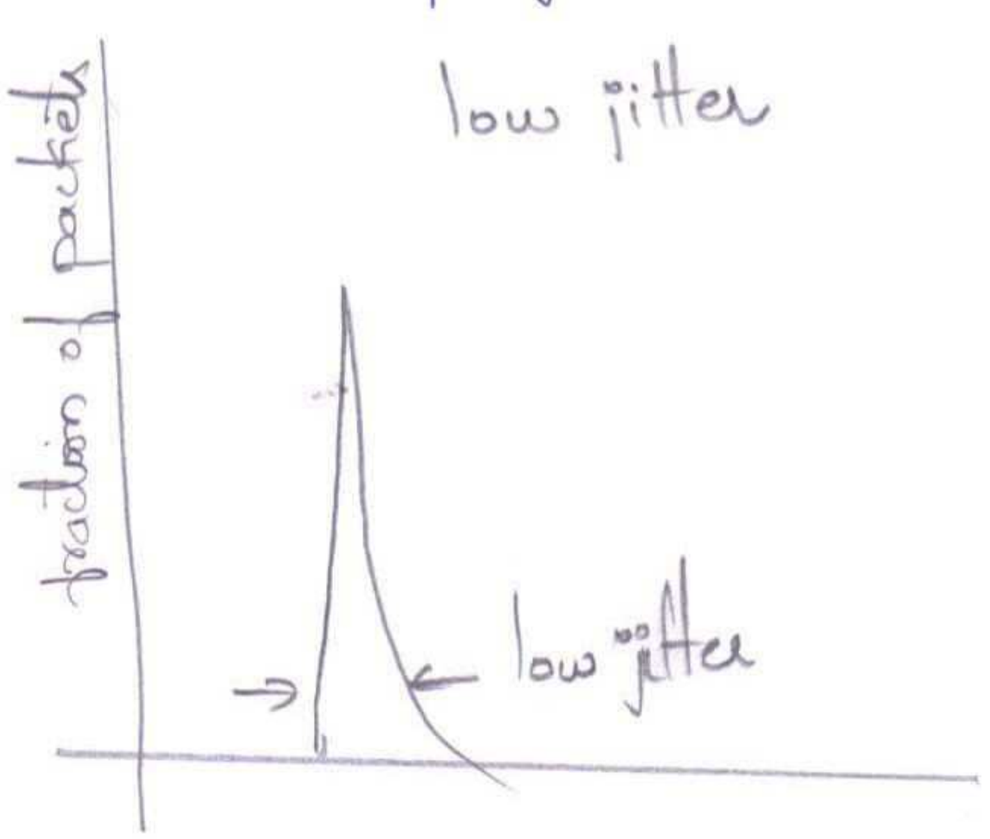
* unfortunately packet 8 has delayed so much that it is not available when it play slot comes up, there are 2 options packet 8 can be slipped by player can stop until 8 arrives creating annoying gap.

— live options do not work well on hold. In streaming media application the player might pause. This problm can be alleviated by delaying the starting time even more, by using a large buffer. For streaming audio or video player buffers of about 10 sec are often used to ensure that the player recieves all packets in time.

— A key consideration for smooth playout is the playback point.

- The avg delay may need to be much further out to capture 99% of packets than if there is low jitter.
- To pick a good playback point application can measure the jitter by looking at difference b/w RTP timestamps and arrival time.
- one way to avoid audio is to adapt playback point b/w talkspurts in the gaps in a conversation. No one will notice difference b/w short & slight longer silence.
- m indicates the start of new talkspurt for lets apps
- No appn of can be done if propagation delay if a direct path is being used. The playback point can be pulled in by simply accepting a larger fraction of packets will arrive too late will be played.

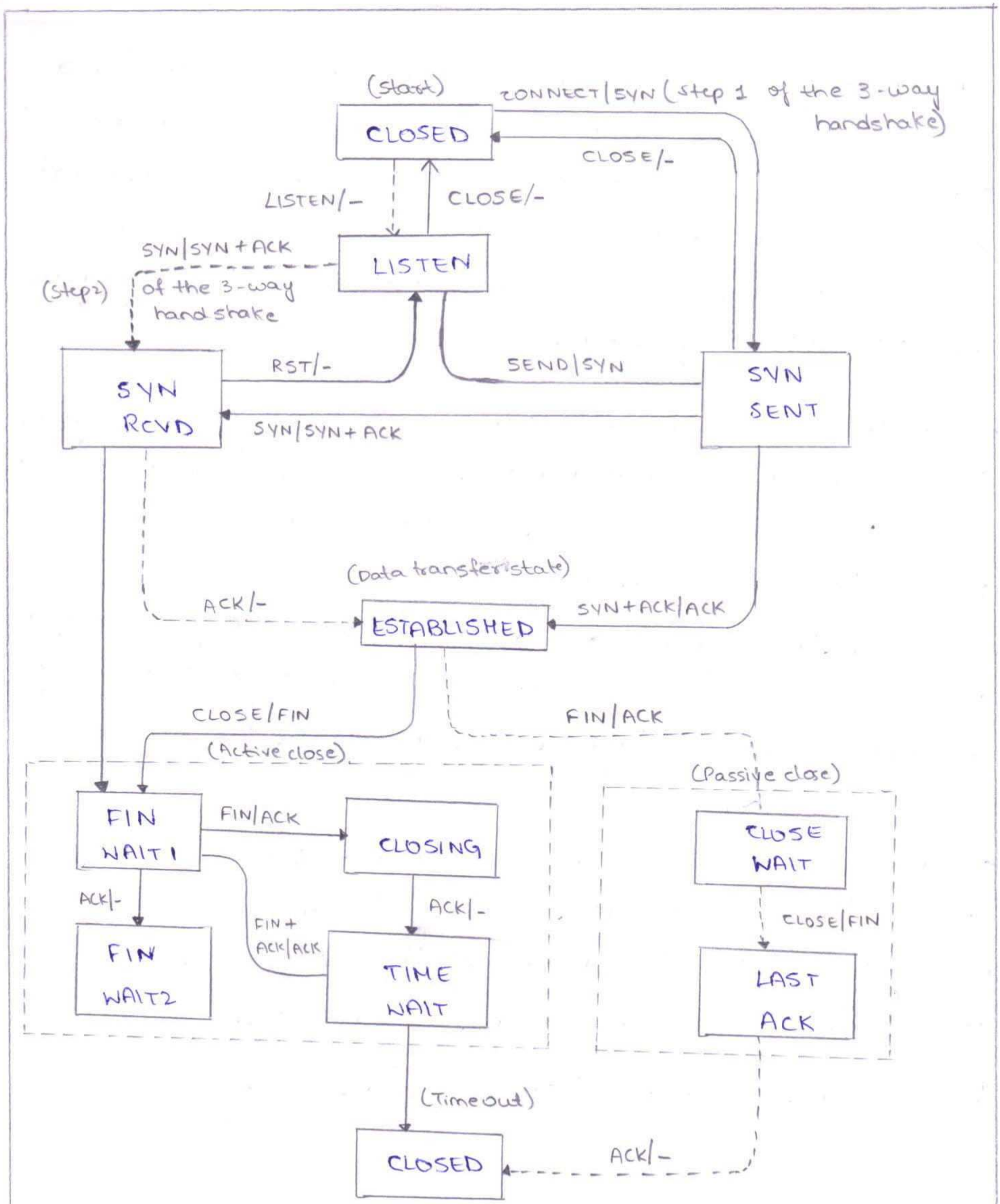




→ each line is marked by an event/action pair. The event can either be a user-initiated system call (CONNECT, LISTEN, SEND, or CLOSE), a segment arrival (SYN, FIN, ACK or RST), or, in one case, a timeout of twice the maximum packet lifetime. The action is the sending of a control segment (SYN, FIN, or RST) or nothing, indicated by —. Comments are shown in parentheses.

→ One can best understand the diagram by first following the path of a client (the heavy solid line), then later following the path of a server (the heavy dashed line). When a application program on the client machine issues a CONNECT request, the local TCP entity creates a connection record, marks it as being in the SYN SENT state, and shoots off a SYN segment. Note that many connections may be open (or being opened) at the same time on behalf of multiple applications, so the state is per connection and recorded in the connection record.

→ When SYN+ACK arrives, TCP sends the final ACK of the three-way handshake and switches into the established state.



TCP connection management finite state machine

THE INTERNET TRANSPORT PROTOCOLS: TCP

UDP is a simple protocol and it has some very important uses, such as client-server interactions and multimedia, but for most Internet applications, reliable delivery is required. UDP cannot provide this, so another protocol is required: It is called TCP and is the main workhouse of the Internet.

INTRODUCTION TO TCP

TCP (Transmission Control Protocol) was specially designed to provide a reliable end-to-end byte stream over an unreliable internetwork. TCP was designed to dynamically adapt to properties of the internetwork and to be robust in face of many kinds of failure.

TCP was formally defined in RFC 793 in September 1981. To give us a sense of extent of TCP, the important RFCs are now RFC 793 plus.

Each machine supporting TCP has a TCP transport entity, either a library procedure, a user process, or most commonly part of the kernel.

A TCP entity accept user data streams from local processes, breaks them up into pieces not exceeding 64KB & sends each piece as separate IP datagram. When data containing TCP data arrives at a machine, they are given to TCP entity, which reconstructs original byte streams. In short, TCP must furnish good peformance with reliability that most applications want & IP does not provide.

THE TCP Service Model:—

TCP service is obtained by both the sender and the receiver creating end points, called sockets. Each socket has a socket number consists of IP address of host and a 16-bit number local to the host, called a port. A port is the TCP name for a TSAP.

Port numbers below 1024 are reserved for standard services that can usually only be started by priviliged users. they are called well-known ports.

The list of well-known ports is given. Over 700 have been assigned. A few of the better-known ones are listed.

→ TCP sends the final ACK of the three-way handshake and switches into established state.

→ Data can now be sent and received.

→ when an application is finished, it executes a CLOSE primitive, which causes the local TCP entity to send a FIN segment and wait for the corresponding ACK (dashed box. marked "active close"). when the ACK arrives, a transmission to the state FIN WAIT 2 and one direction of the connection is closed. when the other side closes, too, a FIN comes in, which is acknowledged. Now both sides are closed, but TCP waits a time equal to twice the maximum packet lifetime to gurantee that all packets from the connection have died off, just in case the acknowledgement was lost.

→ when the timer goes off, TCP deletes the connection record.

→ Now let us examine connection management from the server's viewpoint.

→ The serves does not a LISTEN and settles down to see who turns up.

(b). The 2048 bytes of data delivered to the application in a single READ call.

When an application passes data to TCP, TCP may send it immediately or buffer it as its discretion. This event causes TCP to stop accumulating data and transmit everything it has for that connection immediately.

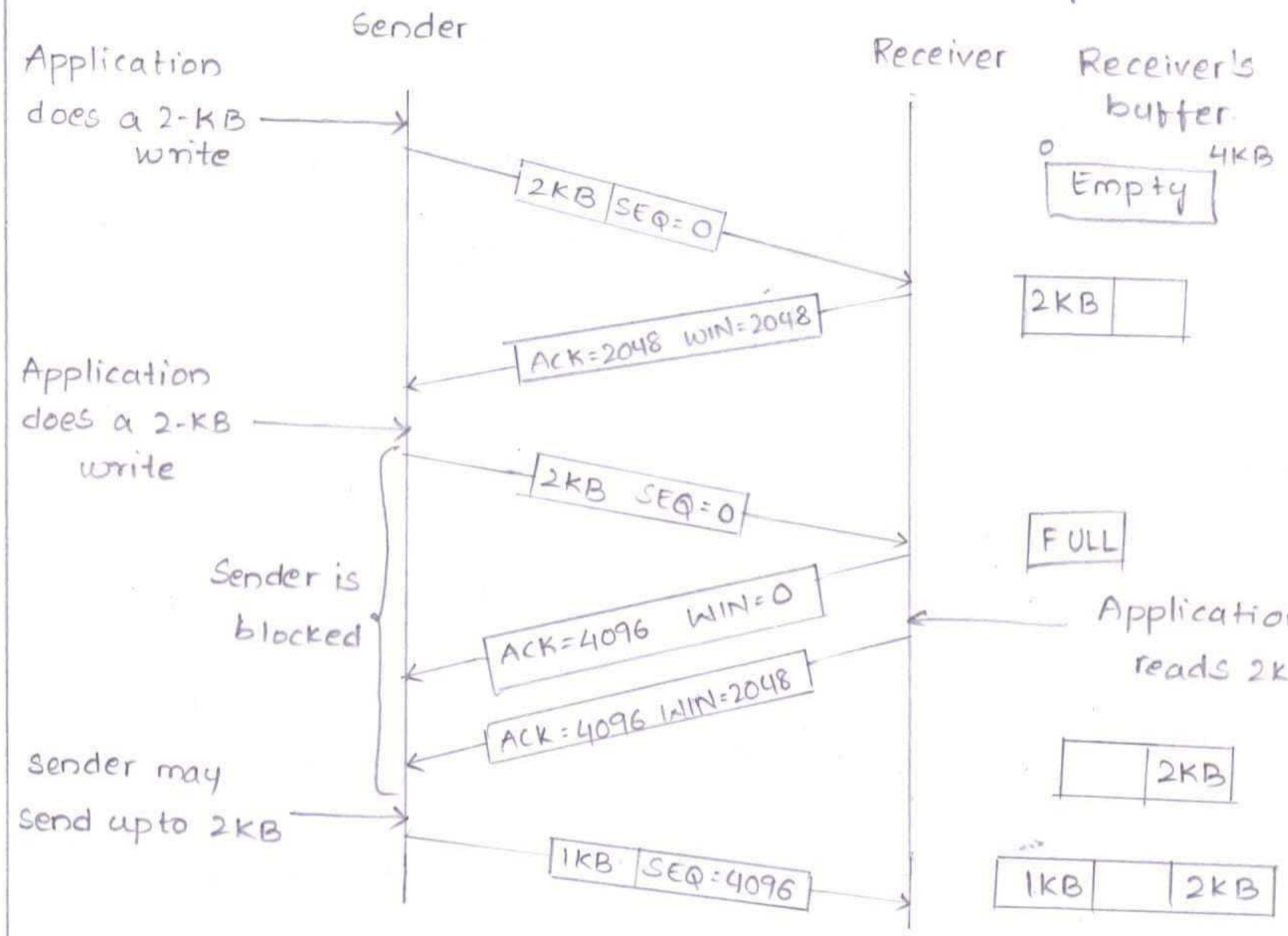
When the urgent data are received at the destination, the receiving application is interrupted, so it can stop whatever it was doing and read data stream to find the urgent data. The start of the urgent data is not marked.

This scheme provides a crude signaling mechanism and leaves everything else up to the application. However, while urgent data is potentially useful. it found no compelling application early on and fell into disuse. Its use is now discouraged because of implementation differences, leaving applications to handle their own signaling. Perhaps future transport protocols will provide better signalling.

TCP Sliding Window

Window management in TCP decouples the issues of acknowledgement of correct receipt of Segments and receiver buffer allocation.

For example, suppose the receiver has a 4096-byte buffer, as shown in the figure. If the sender transmits a 2048-byte Segment that is correctly received, the receiver will acknowledge the Segment. However, since it now has only 2048 bytes of buffer space (until the application removes some data from the buffer), it will advertise a window of 2048 starting at the next byte expected



Window Management in TCP/IP

- payload field tells which encoding algorithm has been used.
- sequence no. is just a counter that is incrimented on each RTP packet sent.
- Timestamp is produced by stream's source to note when the first sample packet was made. This value can help reduce time variability called jitter at receiver by decoupling the playback from arrival. time.
- Synchronization source identifier tells which stream the packet belongs to. It uses multiplexing and demultiplexing

RTCP- real time transport Control protocol.

RTP has a little sister protocol. called RTCP. It is defined along with RFC- 3550 and handles feedback, synchronization. & user interface. It doesn't transport any media samples.

The first function can be used to provide feedback on delay, variation in delay (&) jitter, bandwidth, congestion & other n/w properties to sources.

- This information can be used by encoding process to increase the datarate when the n/w is functioning

The heavy solid line is the normal path for a client. The heavy dashed line is the normal path for a server. The light lines are unusual events. Each transistion is labelled with the event causing it and the action resulting from it, separated by slash.

→ when the server's SYN is itsely acknowledged, the three-way handshake is complete and the server goes to the ESTABLISHED state. Data transfer can now occur.

→ when the client is done transmitting its data it does a CLOSE, which causes a FIN to arrive at the server (dashed box marked "passive close").

→ The server is then signalled. when it too does a close, a FIN is sent to the client. when the client's acknowledgement shows up, the server releases the connection and deletes the connection record.

Now the sender transmits another 2048 bytes, which are acknowledged, but the advertised window is of size '0' The sender must stop until the application process on the receiving host has removed some data from the buffer at which time TCP can advertise a larger window and more data can be sent.

When window is '0', the sender may not normally send segments with two exceptions. First, Urgent data may be sent. Second, the sender may send a 1-byte segment to force the receiver to reannounce the next byte expected and the window size. This packet is called a Window probe.

Senders are not required to transmit data as soon as they come in the from the application. Neither are receivers required to send acknowledgements as soon as possible

Example:

When the first 2KB of data came in TCP, knowing that it had a 4-KB window, would have been completely correct in just buffering the data until another 2KB came in, to be able to transmit a segment with a 4-KB payload. This improves performance.

Consider a connection to a terminal. In worst case, whenever a character arrives at the sending TCP entity,

*TCP creates a 21-byte TCP segment

*It gives to IP to send as a 41-byte IP datagram

*At receiving side, TCP immediately sends a 40-byte acknowledgement

* Later, when the remote terminal has read the byte, TCP sends a window update, moving the window 1byte to the right. This packet is also 40-bytes.

* Finally, when the remote terminal has processed the character, it echoes the character for local display using a 41-byte packet.

* When bandwidth is scarce, this method of doing business is not desirable

One approach that many TCP implementations use to optimize this situation is called delayed acknowledgements. the idea is to delay acknowledgements and windows updates for up to 500 msec in the hope of acquiring some data on which to hitch a free ride. Assuming the terminal echoes within 500 msec, only one 41-byte packet now need be sent back by remote side, cutting the packet count and bandwidth usage in half.

Although delayed acknowledgements reduce the load placed on the network by the receiver, a sender that sends multiple short packets is still operating inefficiently. A way to reduce this usage is known as Nagle's Algorithm.

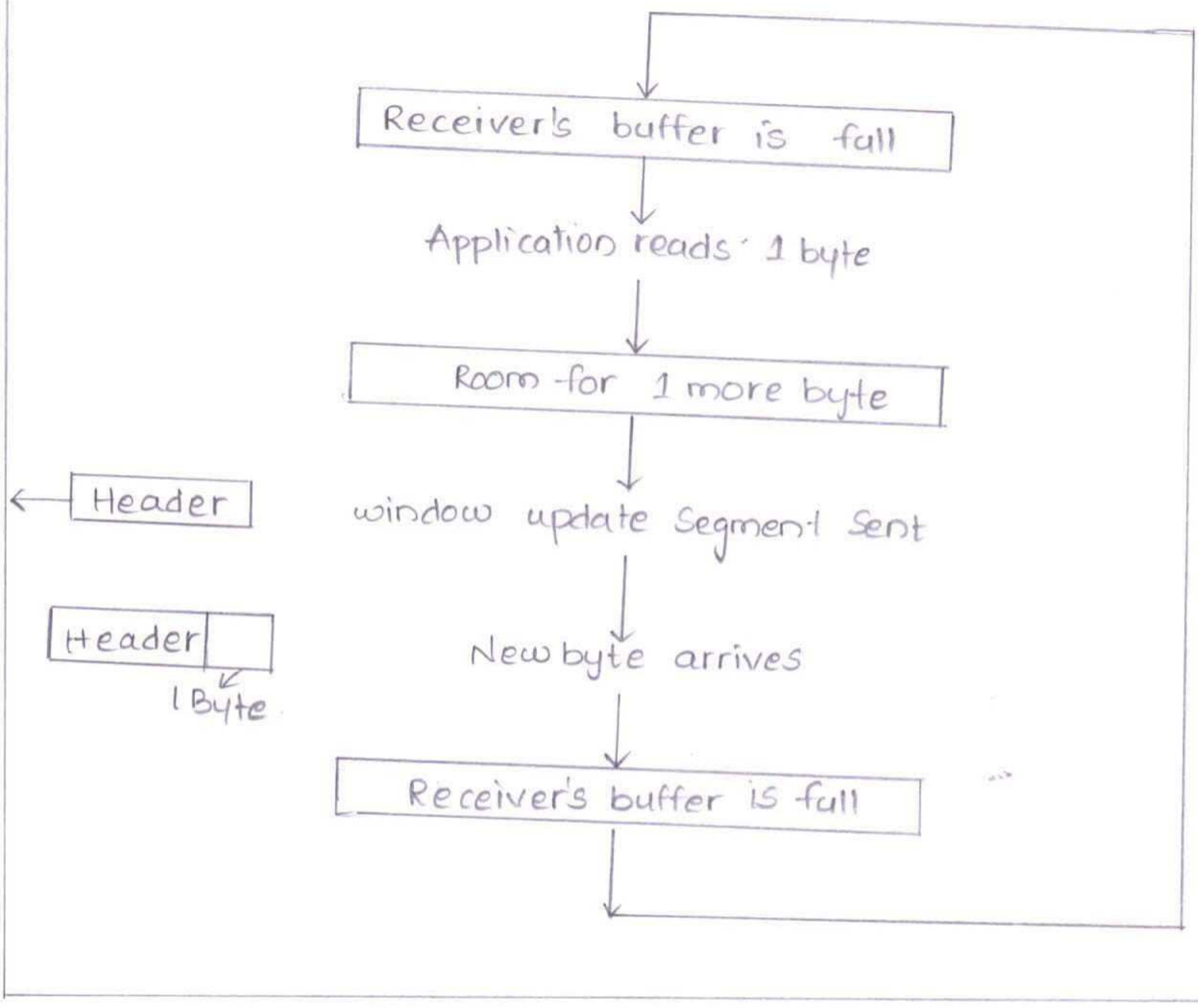
Nagle's Algorithm:

Nagle suggested to reduce the above problem. What nagle suggested is simple.

* When data come to the sender in Small pieces, just send the first piece and buffer all the rest until the first piece is acknocoledged.
* Then Send all the buffered data in one TCP segment and start buffering again untill the next segment is acknowledged.
* Only one short packet can be outstanding at any time
* If many pieces of data are sent by the application in one round-trip time
* Nagle's algorithm will put many pieces in one segment, greatly reducing the bandwidth used.
* The algorithm additionally says that a new segment should be sent if enough data have trickled in to fill a maximum segment.

The receiving TCP can go further in improving the performance than just doing window updates in large units. like the sending TCP, it can also buffer data, so it can block 'READ' request from application until it has large chunk of data for it.

Another issue that the receiver must handle is that Segments may arrive out of order. The receiver will buffer the data until it can be passed up to application in order. Actually, nothing bad would happen if out of order Segments were discarded since they would eventually be retransmitted by Sender, but it would be wastefull.



are caused by congestion and monitor timeouts and look for signs of trouble the way miners watch their canaries. A good retransmission timer is needed to detect packet loss signals accurately and in a timely manner.

Fixing this (many) timer, by including the variation factor, was an important step in Jacobson's work. Given the good retransmission timeout, the TCP sender can track the outstanding number of bytes, which are loading the network. It simply looks the difference between the sequence numbers that are transmitted and acknowledged.

Now track the congestion window using sequence and acknowledgement numbers and adjust congestion window using sequence and (acknowledgement null) AIMD rule. This behaviour might be a good idea for a protocol designed to cause congestion, but not for a protocol to control it.

It turns out that we can use small burst of packets to our advantage. The below figure shows what happens when a sender on a fast network sends a small burst of four packets to a receiver on a slow network — that is the bottleneck or slowest part of the path.

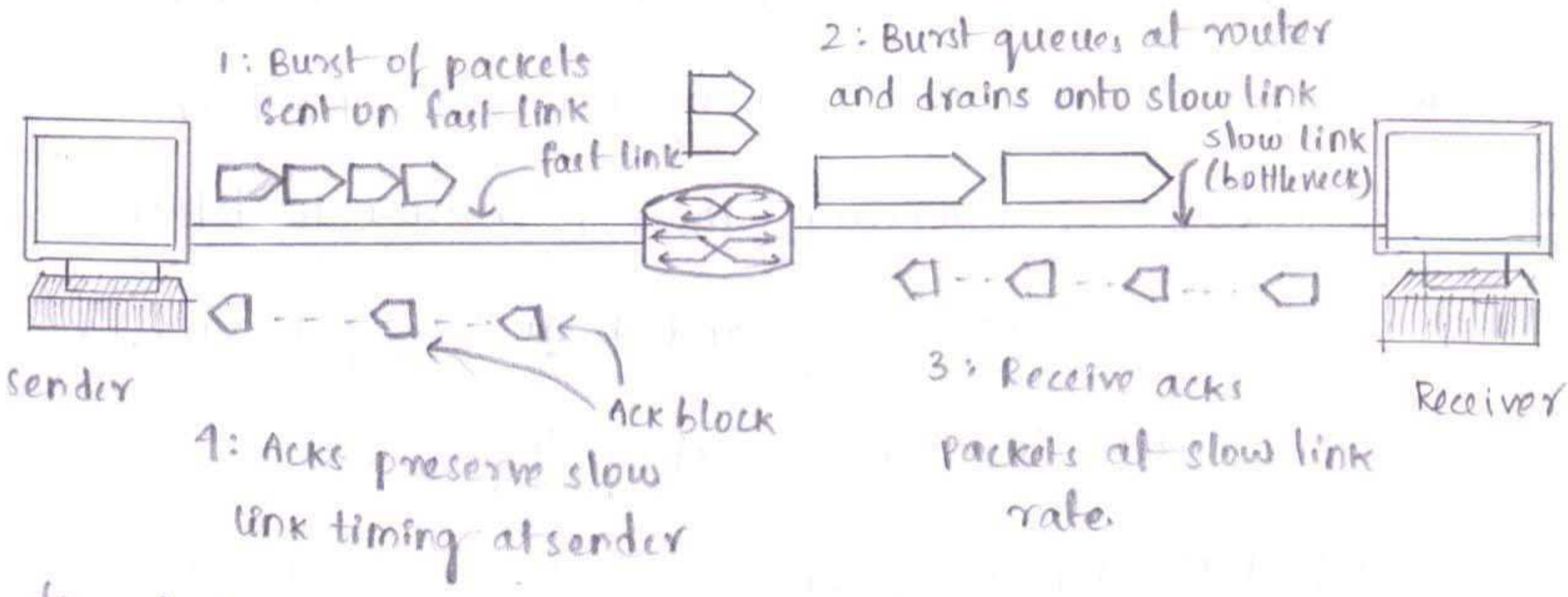


fig. A burst of packets from a sender and the returning ack clock.

Eventually the packets get to the receiver, where they are acknowledged. The times for the acknowledgement reflect the times at which the packets arrived at the receiver after crossing the slow link. They are spread out compared to the original packets on the fast link, As these acknowledgements travel over the network and back to the sender they preserve this timing.

The key observation is this: The acknowledgements return to the sender at about the rate that packets can be sent over the slowest link in the path. This is precisely the rate that the sender wants to use. If it injects new packets into network at this rate, they will be sent as fast as the slow link permits, but they will not queue up and congest any router along the path. This timing is known as an ack clock. It is an essential part of TCP. By using an ack clock, TCP smoothes out traffic and avoids unnecessary

THE FUTURE OF TCP

* As the workhorse of the Internet, TCP has been used for many applications & extended over time to give good performance over a wide range of networks. Many versions are deployed with slightly different implementations than the classic algorithms we have described, especially for congestion control & robustness against attacks.

* It is likely that TCP will continue to evolve with the Internet.

* One of its issues is that TCP does not provide the transport semantics that all applications want.

* For Example, some applications want to send messages or records whose boundaries need to be preserved.

*(i) A Review of TCP Performance :-

* Within any packet-switched network, when demand exceeds available capacity, the packet switch will use a queue

Acknowledgements can be sent only when all the data upto the byte acknowledged have been received. This is called a cumulative acknowledgement. If the receiver gets segments 0, 1, 2, 4, 5, 6 and 7. it can acknowledge everything up to and including the last byte in segment 2. When the sender times out it then retransmits Segment 3. As the receiver has Buffered Segments 4 through 7. upon receipt of Segment 3 it can acknowledge all bytes up to the end of Segment 7.

* It is widely used iby TCP implementations, but there are times when it is betler to disable it.
* In interactive games that are run over the internet, the players typically want a rapid stream of short update packets.
* A more subtle problem is that Nagle's algorithm can sometimes interact with delayed acknowledgements to cause a temporary deadlock: The receiver waits for data on which to piggy back an acknowledgement.
* Another problem that can degrade TCP performance is the silly window sydrome.

Silly window syndrome:

This problem occurs when data are passed to the sending TCP entity in largeblocks but an interactive application on receiving side reads data only 1 byte at a time. TCP buffer on receiving side is full and the Sender knows this. Then the interactive application reads this. Then the interactive application reads one character from TCP stream. This action makes the receiving TCP happy. So it sends a window update the sender saying that it is all right to send 1 byte. The sender obliges and sends 1 byte.

to hold the excess packets.

* When this queue fills, the packet switch must drop packets.

* Any reliable data protocol that operates across such a network must recognize this possibility & take corrective action. TCP is no exception to this constraint.

* TCP uses data sequence numbering to identify packets, & explicit acknowledgements (ACKs) to allow the sender & receiver to be aware of reliable packet transfer.

* This form of reliable protocol design is termed "end-to-end" control, because interior switches do not attempt to correct packet drops.

* Instead, this function is performed through the TCP protocol exchange between the sender & receiver.

* TCP uses cumulative ACK's rather than per-packet ACK's, where an ACK referencing a particular point within the data stream implicitly acknowledges all data with a sequence value less than the ACKed sequence.

The buffer is now full, so receiver acknowledges 1-byte Segments and sets window to '0'. This behaviour can go on forever.

* Clark's Solution is to prevent the receiver from Sending a window update for 1 byte.
* It is forced to wait untill it has a decent amount of Space available and advertise that instread..
* the receiver should not Send a window update until it can handle maximum Segment Size it advertised when the connection was established or until its buffer is half empty, whichever is Smaller.
* Nagle's algorithm and clark's Solution to Silly window Syndrome are complementary. Nagle was trying to Solve the problem caused by Sending application delivering data to TCP a byte at a time.
* clark was trying to solve the problem of receiving application Sucking the data up from TCP a byte at a time.
* Both Solutions are valid and can work together. The goal is for the Sender not to Send Small Segments and receiver not to ask for them

*(iii) TCP Evolution

* The Evolution of TCP is a careful balance between innovation & considered constraint.

* The evolution of TCP must avoid making radical changes that may stress the deployed network into congestion collapse, & also must avoid a congestion control "arms race" among competing protocols.

* The Internet architecture to date has been able to acheive new benchmarks of network effeciency, & translate this carriage effeciency into ground-breaking benchmark prices for IP-based carriage services.

* Much of the credit for this must go to the operation of TCP, which manages to work at that point of delicate balance b/w self-optimization. & cooperative behaviour.

* Widespread deployment of transport protocols that take a more aggressive position on self-optimization will ultimately lead to situations of congestion collapse, while widespread deployment of more conservative transport protocols may well lead to lower jitter

TCP congestion Control

When the load offered to any network is more than it can handle, congestion builds up. The internet is no exception. The network layer detects congestion when queues grow large at routers and tries to manage it, if only by dropping packets. It is up to the transport layer to receive congestion feedback from the network layer and slow down the rate of traffic that is sending into the network.

* In Internet, TCP plays a main role in controlling congestion, as well as the main role in reliable transport. That is why it is such a special protocol.

* TCP congestion control is based on implementing this approach using a window with packet loss as the binary signal. To do so, TCP maintains a congestion window whose size is the number of bytes of the sender may have in the network at any time. The corresponding rate is the window size divided by the round-trip time of connections. TCP adjusts the size of the window (s) according to the AIMD rule.

Modern congestion control was added to TCP largely through the efforts of Van Jacobson (1988). It is a fascinating story. Starting from 1986, the growing popularity

of the early Internet led to the first occurence of what became known as a congestion collapse, a prolonged period during which goodput dropped precipitously due to congestion in the network. Jacobson set out to understand what was happening and remedy the situation.

* The interesting part is how he added this to an existing implementation without changing any of the message formats, which made it instantly deployable. To start, he observed that packet loss is a suitable signal of congestion. This signal comes a little late but it is quite dependable. After all, it is difficult to build a router that doesnot drop packets when it is overloaded.

However, using packet loss as a congestion signal depends on transmission errors being relatively rare. This is not normally the case for wireless links such as 802.11, which is why they include their own retransmission mechanism at the link layer. Because of wireless retransmissions, network layer packet loss due to transmission errors is normally masked on wireless networks. It is also rare on other links because wires and optical fibres typically have low bit-error rates.

All the Internet TCP algorithms assume that lost packets

* TCP also uses ACKs to clock the data flow. ACK's arriving back at the sender arrive at intervals approximately equal to the intervals at which the data packets arrived at the sender.

* If TCP uses these ACK's to trigger sending further data packets into the network, then the packets will be entered into the network at the same rate as they are arriving at their destination.

* This mode of operation is termed "ACK clocking".

* (ii) Better than TCP ?

* Recently, numerous "better-than-TCP" protocol stacks have appeared on the market, most commonly in conjunction with Web Server Systems, where the performance claim is that these protocol stacks can interoperate with standard TCP clients, but offer superior download performance to a standard TCP protocol implementation.

* This level of performance is achieved by modifying the standard TCP flow control systems in a no. of ways. The modified implementation may use a lower initial RTT estimate to provide a more aggressive startup rate, & a more finely grained RTT timer system to allow the sender to react more quickly to network state changes.

* Other modifications may include using a larger initial congestion window size or may use an even faster version of slow start, where the sending rate is tripled, or more, every round-trip time interval.

* The same technique of incremental modification can be applied to the congestion avoidance state, where the linear-rate increase of one segment size per round-trip time interval can be increased to some multiple of the segment size, or use a time base other than the round-trip time for linear expansion of the congestion back-off.

* Resetting the TCP session to slow-start mode following the ACK timeout also be avoided in such modified protocol implementations.

Eg lower packet retransmission rates, but at a cost of considerably lower network efficiency.

* The challenges faced with the evolution of TCP is to maintain a coherent control architecture that has consistent behavior within the network, consistent interaction with instances of data flows that use the same control architecture, & yet be adequately flexible to adapt to differing networks characteristics & differing application profiles.

* It is highly likely that we will see continued innovation within internet transport protocols, but the bounds of such effort are already well recognized.

* We can now state relatively clearly what levels of innovation are tolerable within an Internet Network model that acheives its efficiency not through enforcement of rigidly enforced rules of sharing of the N/w resource, but through a process of trust b/w competing user demands, where each demand is attempting to equilibrate its req. against a finite N/w capacity.

* This is the essence of the TCP protocol.