Computer Networks

Satvik Gupta

March 25, 2023

Introduction

Network: Connection between objects or a group of objects

Computer network: A set of communication elements connected by communication link.

Communication link:

- **Wired**: Optical Fiber, Coaxial Fiber, Twisted pair cable.
- **Wireless**: Radio wave, satellite connection, microwave.

Goals of Networks

- Efficient resource sharing
- Scalable
- Reliability
- Communication
- Application of Networks
- Remote data access.
- Remote software access.
- Emailing
- FIle transfer

Data communication

It is the exchange of data between two or more devices via some transmission medium

Components of Effective Data Communication

- Delivery: The data should be delivered to the destination it was intended to.
- Accuracy
- Timeliness
- Jitter free

Components of Data communication system

- Sender
- Receiver
- Message
- Protocols
- Communication/Transmission medium

Types of Communication

- Simplex: Unidirectional communication.
- Half Duplex: Bidirectional communication but only one direction at a time.
- Full Duplex: Two simplex connections in opposite directions.

Physical Structure

- Point to point
- Multipoint

Physical Topology

It tells how systems are physically connected through links. It is a geometric representation of the network.

Bus Topology

Only one connection.

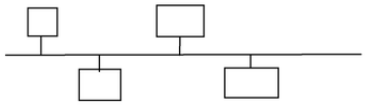


Figure 1: Bus Topology

Advantages

- Easy to install
- Cheap
- Easy to expand

Disadvantages

- Only one device can transmit at a time, which makes it low speed.
- Single point of failure - faulty cable can bring down the whole system.

Ring Topology

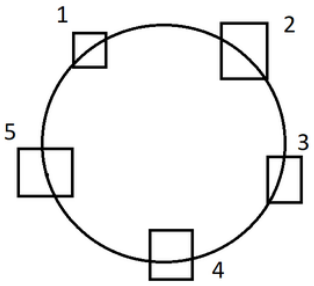


Figure 2: Ring Topology

Tokens are used to transfer data. Only one system can hold the token at a time. Token passing is done.

***Advantages**

- Cheap

Disadvantages

- Not easy to install.
- Not easy to expand.
- If one system/one link goes down the entire ring will go down.

Star Topology

Uses a central hub.

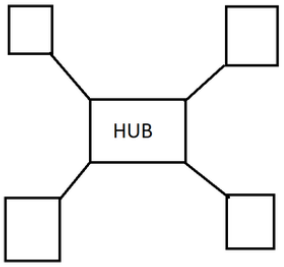


Figure 3: Star Topology

__ **Advantages and disadvantages same as of any centralized system**__ Hub can also be expensive.

Mesh Topology

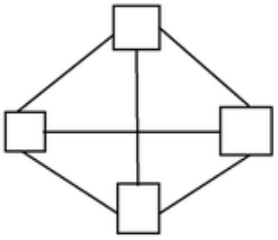


Figure 4: Mesh

Advantages

- Less traffic
- No single point of failure
- Messages can be sent directly without any routing

Disadvantages

- Cabling cost will be higher
- Maintenance cost will be higher.

Tree Topology

Tree structure.

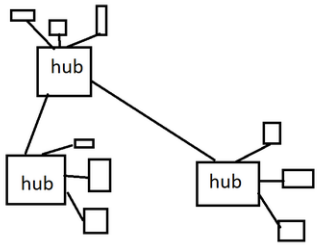


Figure 5: Tree Topology

Networks Based on Geographical Area

- LAN - Local Area Network
- MAN - Metropolitan Area Network
- WAN - Wide Area Network

Differentiate based on cables, cost, etc.

OSI Model - Open Systems Interconnection

Given by ISO.

The OSI model is a layered framework for the design of network systems that allows communication btw all types of computer systems.The purpose of OSI model is to facilitate communication btw different systems without requiring changes to the logic of underlying hardware and software.

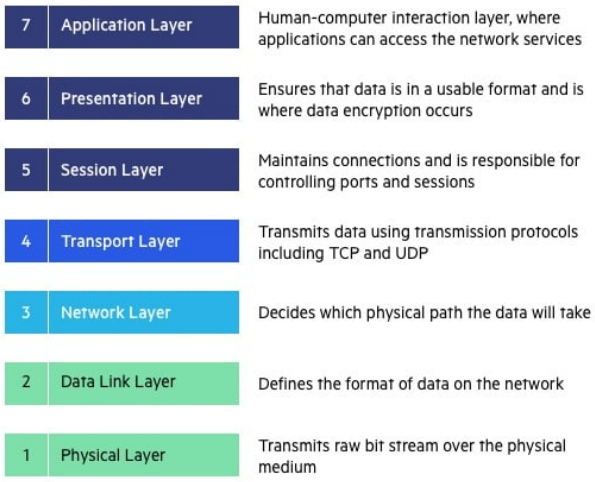


Figure 6: OSI

Data in layers:

S.No	Layer	Data	Responsibility	Protocols
1	Application Layer	Data	To allow access to network resources	Telnet, SMTP,DNS, HTTP
2	Presentation Layer	Data	To translate, encrypt and process the data	
3	Session Layer	Data	To establish, manage and terminate session	
4	Transport Layer	Segment	Process to Process msg delivery, error recovery	TCP, UDP (Port/Socket Address)
5	Network Layer	Packet	Move packet from source to destination.	IP, ARP, RARP, ICMP (Logical/IP Address)
6	Data Link Layer	Frame	Hop to hop delivery, organize the frames	IEEE 802 Std., TR,PPP (Physical/MAC Address)
7	Physical Layer	Bit	Transmit bits over a medium, provide mechanical and electrical specification	Transmission media

ARP - Address Resolution Protocol - Maps IP to MAC.

RARP - Reverse Address Resolution Protocol - Maps MAC to IP.

Physical Layer

It is responsible for moving physical bits. It defines:

- a transmission medium (wireless/wired)
- types of encoding to be used
- data rate
- synchronization of bits
- physical topology

Transmission Media

Wired/Guided Media:

- Optical Fiber

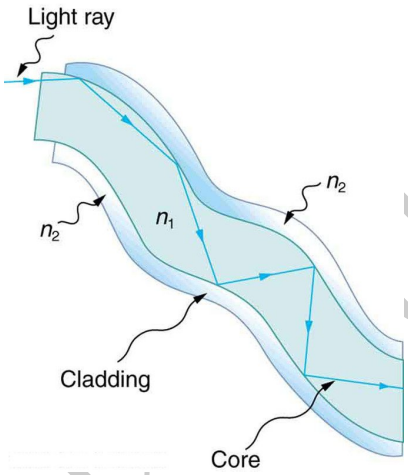


Figure 7: Transmission through Optical Fiber

- Coaxial Fiber

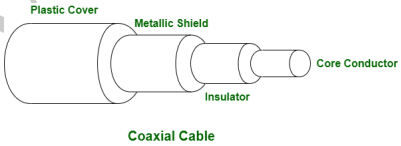


Figure 8: Coaxial Fiber

- Twisted pair cable

Figure 9: Twisted Pair Cable

- Unshielded Twisted Pair (USTC)

Figure 10: USTC

- Shielded Twisted Pair (STC)

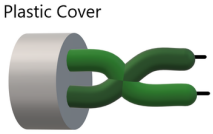


Figure 11: STC

Wireless/Unguided Media:

Radio wave, microwave, and infrared

Electromagnetic Spectrum for Wireless Communication

- 3KHz - 300 GHz => Radio Waves and Microwaves
- 300Ghz - 400 THz => Infrared
- 400 THz - 900 Thz => Light waves (not used for transmission)

Propagation Methods

Ground, Sky, Line of Sight

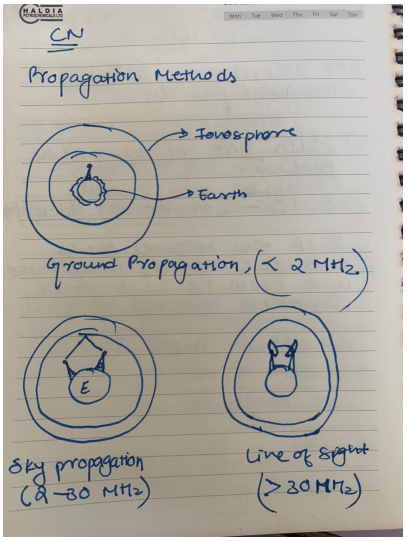


Figure 12: Propagation Methods

Bands

Band	Range	Propagation	Application
Very Low Freq.	3-30 KHz	Ground	Long range radio navigation
Low Freq.	30-300KHz	Ground	Radio and Navigation Locator
Medium Freq.	300KHz-3MHz	Sky	AM radio
High Freq	3-30 MHz	Sky	Citizen Band(Ship/Aircraft Communication)
Very High Freq.	30-300 MHz	Sky/Line of Sight	Cellular Phone, Satellite
Super high freq.	3-30 GHz	Line of Sight	Satellite Communication
Extremely high freq.	30-300 GHz	Line of Sight	Radar/Satellite Communication

High Frequencies cannot travel through walls, lower frequencies can.

High Frequency have less distance, lower frequencies have higher distance.

Switched Networks

Large networks cannot have all nodes directly connected with each other. Therefore, to send data from one node to another, it has to be sent through other nodes.

Suppose there's a network with many nodes, and node A wants to send some data to node B. There are two ways of doing so.

1. **Packet Switching**

   Data is divided into small sized packets for transmission. This increased efficiency, reduces chances of lost data,etc.

   1.**Virtual Circuit** - Source establishes a (virtual) path that the data will follow. Each packet goes through the same route.

   2. **Datagram Switching** - Source doesn't decide any route. It sends each packet to the next nodes. Each node can decide where to forward the packets. Each packet may end up taking a different route. Packets may be delivered in a different order.

2. **Circuit Switching**

   A special path is set up for the transmission, and the intermediate nodes are already decided before the data transmission takes place. There is a dedicated path set up for the transmission of that packet.

3. **Message Switching** - Entire message is transferred between nodes. Each node stores the message, then decides where to forward it. This is also called *store and forward*.

Network Architecture

Protocol is an agreement between two communicating parties on how the communication is to take place. Includes things like format of data, speed, etc.

Networks are organized as a stack of layers. Each layer offers its services to the layer above it (like OSI). Between each two layers there is an interface. **Services** are operations a layer provides to the layer above it. Protocols are used to implement services.

Set of layers and protocols is called the Network Architecture.

List of Protocols used by a certain system is called **Protocol Stack**. We generally have one protocol per layer.

Types of Communication on the basis of Connection

Connection-Oriented

- Similar to telephone.
- Establish a connection, communicate, and then release the connection.

Connectionless

- Like postal system
- Each packet has the destination address,
- Each packet is routed independently through the system.

TCP/IP Stack

Internet Protocol (IP)

- Hosts can inject packets into any network.
- They will travel to destination independently.
- They may arrive out of order.
- Connectionless.
- Similar to postal service.

Transmission Control Protocol (TCP)

- Transport Layer
- Source and Destination communicate using this.
- It is connection-oriented, reliable. Provides no-error delivery.
- Handles flow control
- Converts incoming byte-stream (continuous data like video or large amount of text) into discrete messages. Destination's TCP reassembles them.

User Datagram Protocol (UDP)

- Transport Layer
- Connectionless, unreliable
- Faster than TCP.

Delay

Transmission Delay

The delay taken for the host to put the data onto the transmission line.

$$T_t = L/B$$

where L is the size of the data, and B is the bandwidth.

Propagation Delay

Time taken by the last bit of the data to reach the destination (after it has been transmitted from host to transmission media at the source.)

$$T_p = distance/velocity$$

Queueing Delay

Each packet waits in the buffer (at destination) before it is processed. The amount of time it waits is known as the queueing delay.

Processing Delay

This is the time taken to process the packet. Includes checking headers, updating TTL, deciding where to forward it,etc.

Queueing and Processing delays are generally taken to be zero, unless mentioned otherwise.

Framing Techniques

One of the major issue in framing is to decide how to specify the start and end of a frame.

One way to do this is to use **fixed-size frames**. For eg, if we say one frame is 50 bytes, then the destination's data link layer will know that after the first 50 bytes, the next frame has begun.

This can lead to wastage of space. If the data in a frame is only 10 bytes, we have to add 40 bytes of empty space.

Thus **variable-sized frames** are more preferred.

Framing techniques for variable-sized frames are given below.

Character Count

It involves simply adding the number of characters in a frame to the data-link header. Before each frame starts, the number of characters in it is present.

For eg, we have 4 frames with lengths 3,4,2,5 characters respectively. Our data would look like:

`(3)(frame1)(4)(frame2)(2)(frame3)(5)(frame4)`

This isn't used anymore, because a single bit error in the character count could lead to miscalculation of frames. Count variable could also only hold a certain limit of number. For e.g, if `count` was specified as an 8-bit number, the maximum value it can hold is $2^8 - 1 = 255$. If our frame has more than 255, we would face issues.

Hence this isn't used.

Flag (Character Stuffing/Byte Stuffing)

We use a special **flag byte** at the start and end of each frame. It is fixed so it can be recognized.

An issue with this is that the flag byte may occur "accidentally" in the data itself. This may cause the DLL to assume the frame has ended even when it has not.

To solve this, we use a special **ESC** byte, which is also fixed. Accidental flag bytes have the ESC sequence inserted before them, to tell the DLL that this FLAG is data and not the end of a frame.

If ESC occurs within the data "accidentally", we escape it with another ESC.

Examples

A Flag B –> A ESC Flag B

A ESC B –> A ESC ESC B

A ESC Flag B –> A ESC ESC ESC Flag B

A ESC ESC B –> A ESC ESC ESC ESC B

Doesn't work if the data isn't 8-bit.

Bit Stuffing

A special bit pattern denotes start and end of frames. Generally, this is taken to be `01111110`.

If the sender encounters the starting of this pattern in the data, it adds a 0 or a 1 before it ends so that the pattern never occurs. The receiver will do the opposite and remove the *stuffed* 0s or 1s.

For e.g, for `01111110`,

If the sender encounters a 0 followed by 5 consecutive 1's, it adds a 0 before continuing. This ensures that `01111110` never occurs in the data.

The receiver will *destuff* these extra zeroes on its end.

`01111110` –> `011111010`

`0110111111111111111110010` –> `01101111101111101111010010`

Error Detection and Control

Parity Check

Parity check is the simplest method of detecting errors. It involves using a single check bit after or before the frame.

We count the numbers of 1s and 0s in a particular frame. If it's odd, check bit value is 1. If it's even, check bit value is 0.

101011 - Check bit=0 101001 - Check bit=1

This can only detect single-bit errors. If two bits (a 0 and a 1) are flipped, the number of 1s remains the same. 2D parity check is more useful. The data is divided into rows and columns. Parity bit is calculated for each row and column.

```
1 0 1 0 1 | 1
1 1 0 0 0 | 0
1 0 1 1 0 | 1
_ _ _ _ _ |
1 1 0 1 1 | 0
```

The final data becomes

```
101011
110000
101101
110110
```

Hamming Code

Data - the original data to send **Redundant bits** - bits that aren't part of the original data. They have been added for error detection and correction. **Codeword** - The final result that is sent to the receiver - combination of data and redundant bits. **Code** is a collection of codewords.

Hamming Distance between two strings is the number of positions where the symbols are different. Only valid for strings of equal length. We will use it for binary numbers.

For eg, hamming distance btw 101 and 100 is 1 (only last bit is different) btw 101 and 110 is 2 (2nd and 3rd bits are different).

Hamming Distance of a Code is the minimum hamming distance between any two codewords in the code.

A code with a hamming distance of d can *detect* $d-1$ bit errors, and *correct* $\lfloor (d-1)/2 \rfloor$ errors.

For example, consider a code with the codewords `000` and `111`. The hamming distance of such a code would be 3. We should be able to detect 2-bit errors, and correct 1-bit errors.

Suppose we were transmitting the codeword `000` ande during transmission bit flips occurred:

- If the bit flip resulted in `001`. We know this is not a valid codeword (only `000` and `111` are valid), so we know an error has occurred.

  The hamming distance between `111` and `001` is 2.

  The hamming distance between `000` and `001` is 1.

  Therefore, we can guess that the original word must have been `000`. Similarly, we would be able to detect and correct errors if the bit flip resulted in `010` or `100`.

- If the bit flip resulted in `011` (2-bit error).

  Once again we can see it isn't a valid codeword, so we are able to detect the error.

  But if we try correcting it, we would think that the original codeword was `111`, since the hamming distance of (`111`,`011`) is less than that of (`000`,`011`).

- If the bit flip resulted in `111` (3-bit error)

  We wouldn't be able to detect the error, since `111` is a valid codeword.

Thus we saw - we can detect 1-bit and 2-bit errors. We can only correct 1-bit errors.

Hamming Codes can also be used to detect and correct errors in a different manner.

Let's say we have data of length m. First we need to calculate the number of redundant bits needed to transmit this.

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

The smallest value of r satisfying this equation can be used as the number of redundant bits.

Format of the Codeword

- Total number of bits = $n = m + r$.
- The bits at positions $2^0, 2^1, 2^2, 2^3$... and so on are *check bits*.
- The bits in the rest of the positions are data bits.
- Bits are numbered from 1.

Calculating the values of check bits

- Suppose we have m=7. We can see that r=4 will work. Codeword length = 7+4=11.
- Bits at positions 1,2,4,8 are check-bits.
- Bits at positions 3,5,6,7,9,10,11 are data bits.

Suppose our data is 0100110 (7-bit) Final data will look like:

```
_ _ 0 _ 1 0 0 _ 1  1  0
1 2 3 4 5 6 7 8 9 10 11
```

Write the position number of data bits as sum of powers of 2.

3 = 1+2

5 = 1+4

6 = 2+4

7 = 1+2+4

9 = 1+8

10 = 2+8

11 = 1+2+8

To calculate value of check-bit 1:

- Check which equations have `1` on the RHS.
- Equations for bits 3,5,7 and 9 have `1` on RHS.
- So, we will do a parity count of bits 3,5,7,9

Bit 3 = 0

Bit 5 = 1

Bit 7 = 0

Bit 9 = 1

- Parity count = 0 (even number of 1's). So, the value of check bit 1 is 0.

Value of check bit 2:

- Equations for 3,6,7,10,11 have 2 on RHS.

- Parity count:

  Bit 3 -> 0

  Bit 6 -> 0

  Bit 7 -> 0

  Bit 10 -> 1

  Bit 11 -> 0

  Parity Count = 1

- Value of check bit 2 = 1

Similarly we can do for check-bit 4 and 8

Check bit 4:

Equations : 5,6,7 Parity check : 1,0,0 Parity count = 1

Value of check-bit 4=1

Check bit 8

Equations: 9,10,11 Parity Check: 1,1,0 Parity Count = 1

Value of check-bit 8 = 1

Final Codeword:

```
_ _ 0 _ 1 0 0 _ 1  1  0
1 2 3 4 5 6 7 8 9 10 11

becomes

0 1 0 1 1 0 0 1 1  1  0
```

Checksum

- Sum all the bits in the data word.
- If the checksum goes beyond n bits, add the extra leftmost bits to the n rightmost bits. (***Wrapping***)
- Take 1's complement of the sum so found. This becomes our checksum
- Send the checksum along with the data to the receiver.

At receiver's end: - Add all the data received (including the checksum) - Perform wrapping if necessary. - The end result should be `1111....1`, i.e, n times 1 - If the end result is anything else, it means the data is wrong.

Example

Let n=4. We are sending 4-bit data.

Let the data be 4 numbers : 7,11,12,6.

- Add all the numbers: 7+11+12+6 = 36
- 36 cannot be represented in 4-bits, so we perform wrapping.

```
36 in binary 100100

Extra bits are the leftmost 2 bits (10)

Take them and add them to the rightmost 4 bits

0100
  10
----
0110
```

```
----

Take 1's complement

1001
In decimal = 9
```

9 becomes our checksum which we send to the receiver.

At receiver's end:

All the data is added.

7+11+12+6+9 = 45

45 cannot be represented in 4-bits, so we will do wrapping

```
45 in binary is 101101

Take extra digits - leftmost 2 (10)
Add to rightmost 4

1101
  10
----
1111
----
```

We received the sum as 1111, so we know our data is correct.

CRC (Cyclic Redundancy Check)

Data - k-bit Codeword - n-bit Divisor - (n-k+1) bits. Divisor should be mutually agreed between sender and receiver.

- Add (n-k) 0s to the dataword.
- Divide dataword by divisor using *modulo-2 division.*
- Append the remainder found to the original dataword (without the extra zeroes)

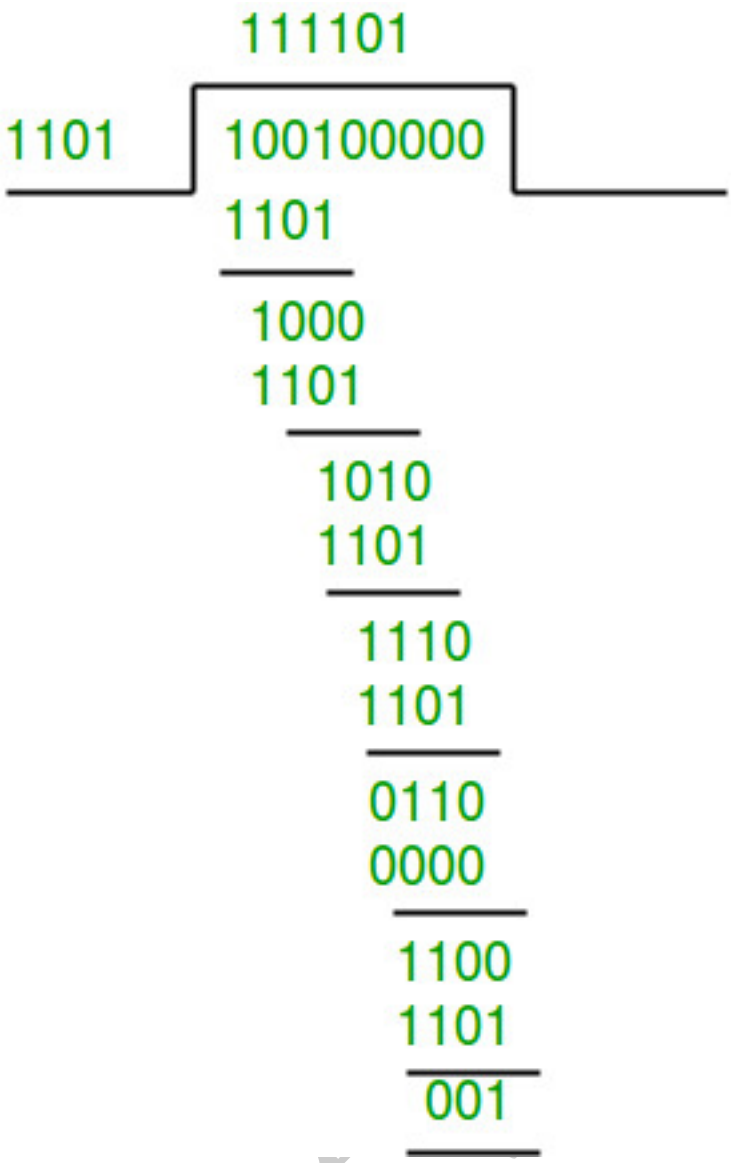
At receiver's side:

- Perform modulo-2 division of the received code-word and divisor.
- If the remainder is 0, the data is correct. Otherwise it's incorrect.

Modulo-2 Division

It's a method of dividing 2 binary numbers.

It follows the rules and logic of normal division, with subtraction step replaced by bitwise XOR.



Flow Control

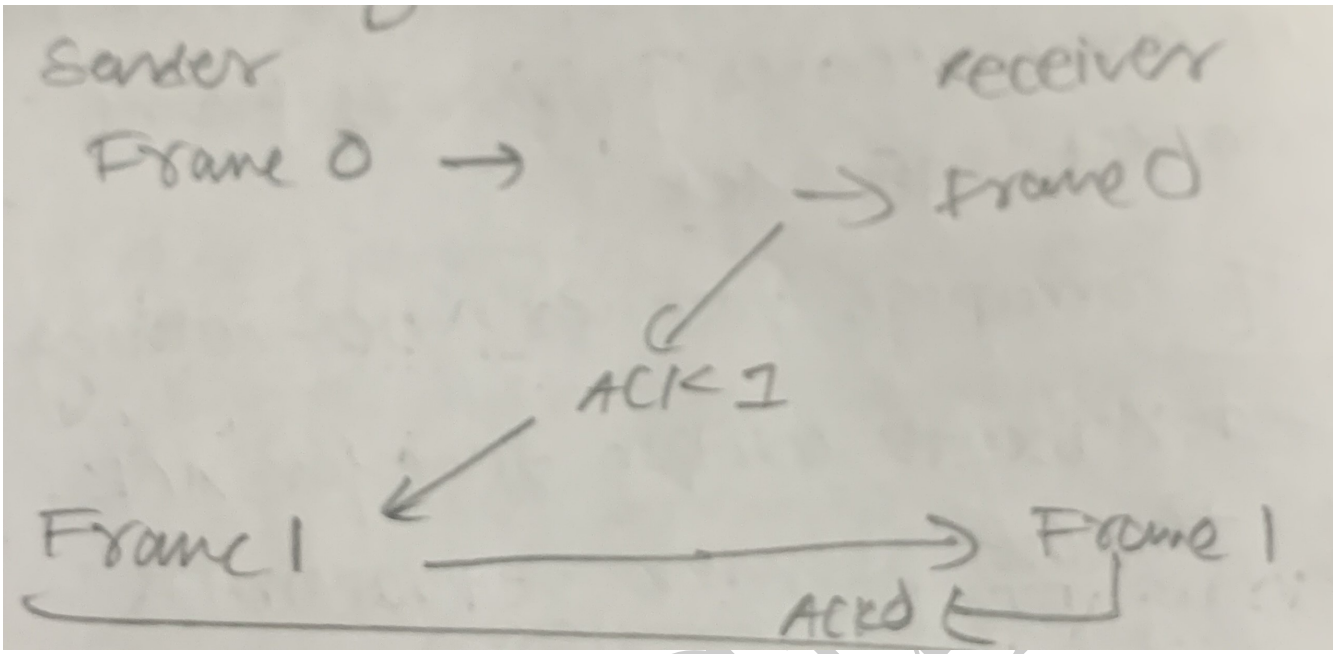
To make sure receiver receives all the data.

Stop & Wait ARQ (Automatic Repeat ReQuest)

- Sender sends a frame and waits for ACK (acknowledgement) for the frame from receiver.
- Receiver receives the frame. If the frame is correct, receiver sends ACK.
- If the frame is corrupt, receiver drops the frame and does nothing.
- Sender waits a certain amount of time for ACK from receiver. After this, it times out and resends the frame.
- ACK message may also get lost, then the sender will assume the original frame was corrupted or lost. It will retransmit, which means the receiver may get duplicate data.
- To avoid this, frames are numbered.
- We only need to differentiate between a frame and its immediate successor. I.e, we need to differentiate between frame x and x+1. We don't need to differentiate between frame x and x+2.

  x+2 will never be sent unless both x and x+1 have been sent AND acknowledged.

- Therefore, 1 bit sequence number is enough. If the first frame is 0, the second frame is 1, the third is again 0, and so on.



Formulas

Total time taken = Transmission time of data + transmission time of ACK + propagation time of data + propagation time of ACK + Queuing delay + Processing delay.

$$= T_{t(data)} + T_{t(ACK)} + T_{P(data)} + T_{P(ACK)} + Delay_{Queue} + Delay_{Processing}$$

We take queue delay and processing delay to be 0. As ACK is very small, we take transmission time of ACK to be 0.

$$= T_{t(data)} + T_{P(data)} + T_{P(ACK)}$$

Propagation time of ACK and data will be same.

Total time = $T_t + 2 * T_p$

Where T_t is transmission time of data, and T_p is propagation time.

Efficiency = η = Useful Time/ Total Time

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

where a = T_p/T_t.

Throughput = η * Bandwidth

Go Back N

- Sliding Window Protocol
- Receiver window Size = 1
- Sender Window Size = $2^m - 1$
- Sequence numbers for frames – $[0, 1, ...2^m - 1]$, 0 and $2^m - 1$ inclusive.
- Window Size = WS/W
- We send up to W frames at a time, and keep them in memory until the receiver ACKs them.
- Receiver only receives one frame at a time.
- Receiver can send a single ACK for many frames. For eg if sender sent 7,8,9 and receiver received all, it can simply send ACK 10.
- If receiver receives wrong frame (e.g receiver was waiting for frame 3 and frame 4 came), or a corrupted frame, it stays silent.
- Sender's timer will timeout. Sender will resend all frames in the window.

For e.g, if WS=3 and sender has sent 1,2,3,4,5,6 and timer for 3 times out (1 and 2 ACKed successfully), sender will send 3,4,5 again.

Efficiency = $\eta = \frac{WS}{1+2a}$

where $a = T_p/T_t$

For maximum efficiency (100% usage),

- WS = 1+2a
- No. of bits needed for sequence number = $\lceil log(1 + 2a) \rceil$

Selective Repeat (SR)

- Only lost/corrupted frames are resent.
- Sender window size = Receiver Window Size
- Window Size = $2^m/2$
- Receiver buffers frames that are within its window range. Others are dropped. For e.g, if receiver's window is waiting for frames 3,4,5 and sender sends 6, it will be dropped.
- ACK is only sent after frames are received in order. If receiver window is 3,4,5 and we receive 4,5 - 4,5 will be stored and buffered, but no ACK will be sent.

  Instead, receiver will send a negative acknowledgement (NACK) for 3 - NACK 3.
- This tells the sender that receiver hasn't received 3.
- Sender will resend 3 (only 3).
- When 3 is received, receiver will send ACK 6.
- If we had received 3 in the beginning, we would have immediately sent ACK 4. This would also make the receiver move its window to 4,5,6.

Example case:

- Sender window = [3,4,5],6,7. Sender sends 3,4,5.
- Receiver receives 3 and sends ACK 4. 4 is lost. Receiver buffers 5 and sends NACK 4. Receiver moves its window by 1 to [4,5,6],7.
- As soon as the sender receives ACK 4, it knows receiver has received 3.
- It will move its window by 1 to [4,5,6],7. As 5 has already been sent, sender will now also send 6.
- Sender will receive NACK 4 and resend 4.
- Receiver will receive 6 from sender, and buffer it. Receiver still hasn't received 4. It will send NACK 4.
- Receiver will receive 4 from sender. It has 5,6 so it will send ACK 7.
- Sender will receive the second NACK and again send 4
- Receiver will receive the 4 (that the sender sent earlier), and accept it. 4,5,6 have all been received. Receiver will send ACK 7, and move its window to [7,8,9]

- Receiver will receive the second 4 sent by the sender. It will be ignored since it's out of the window.
- Sender will receive ACK 7 and move its window to [7,8,9].
- Transmission will continue as normal from here.

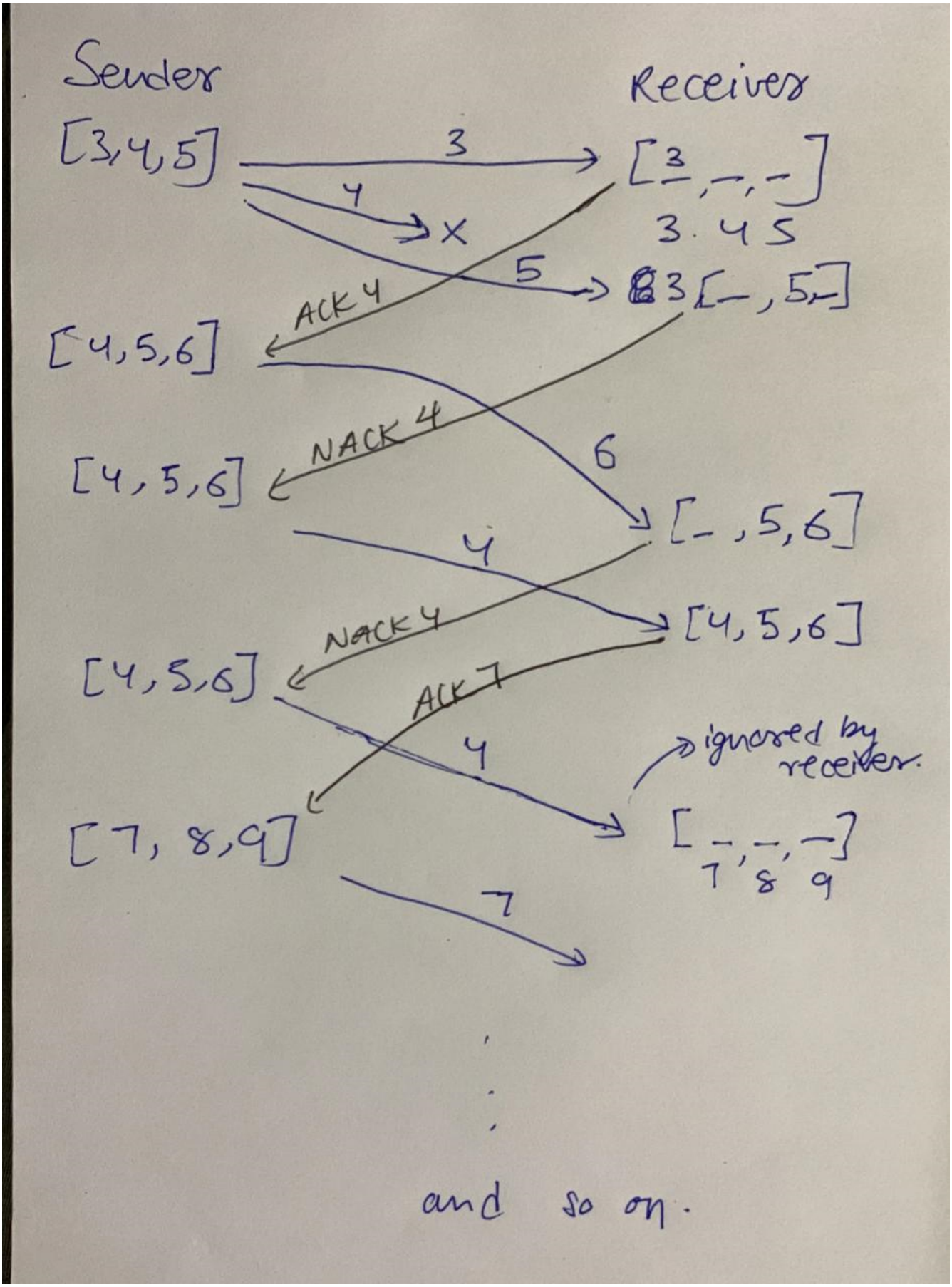


Figure 13: Selective Repeat Example

Media Access Sublayer

The data link layer is divided into two sublayers.

1. **Media Access Control (MAC)** - Defines the access method for each LAN.
2. **Logical Link Control (LLC)** - Flow control, Error control, etc.

Framing is handled by both.

Media Access Control and Multiple Access Protocols

Handle how multiple nodes can communicate on a single link.

Random Access/ Contention Methods

- All nodes are considered equal.
- No scheduled transmission.
- Transmission occurs randomly.
- Nodes compete for access.

Pure Aloha

- Each node sends a frame when it has a frame to send.
- Obviously, we will have collisions in case 2 nodes decide to send a frame together.
- Aloha expects the receiver of the frame to send ACK for the frame.
- Vulnerable time for Aloha is $2 * T_t$. This is the time frame in which collisions can happen.

  For eg, A sent a frame at 12:05

  Let transmission time = 5 minutes.

  B wants to send a frame. But it cannot send a frame until 12:10, because till 12:10 A will be transmitting its frame. A collision will occur if B sends before 12:10.

  Similarly, if C had earlier sent a frame anytime after 12:00, A's frame will collide with it.

  Therefore, the vulnerable time is 12:00 - 12:10, which is 10 minutes = twice of transmission time.
- In case a collision occurs, the node waits a random amount of time before retransmitting. How much time to wait is explained in the flowchart below.
- Maximum number of attempts are fixed.This value is called K_{max} For eg, if max attempts = 15, if a node has transmitted the same frame 15 times and always gotten collision, it will abort and try again some time later.
- K_{max} is generally set to be 15.

Efficiency of Pure Aloha

$$S = G.e^{-2G}$$

where G is the the average number of frames created by the *entire system* (all nodes combined), during the transition time of a single frame.

For eg, if T_t is 1ms, G is number of frames produced per millisecond.

$S_{max} = 0.184$ at $G = 1/2$.

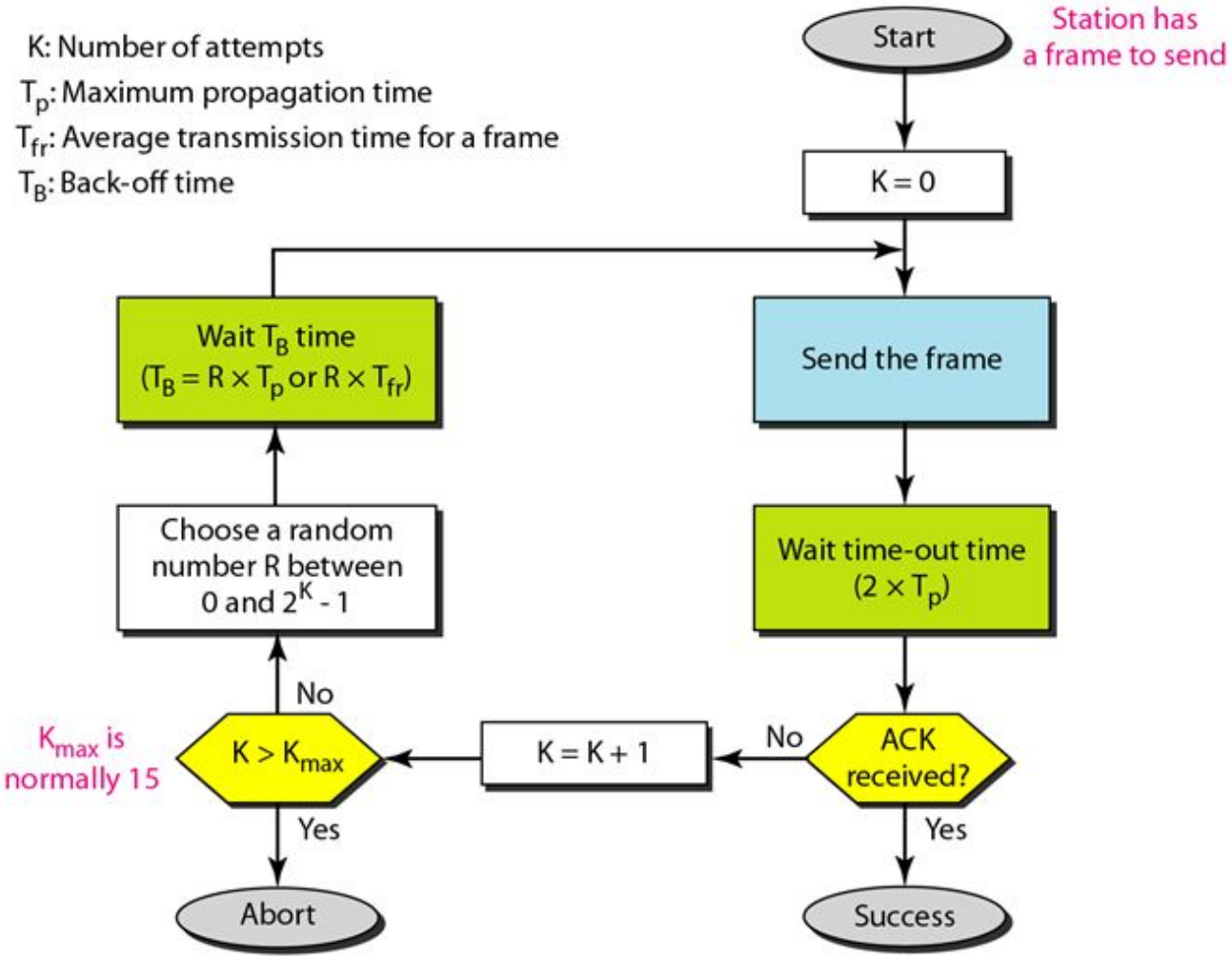


Figure 14: Flowchart for Aloha

Slotted Aloha

- Same as Aloha, but time is divided into slots.
- Time is discrete and globally synced (all nodes have same value of time)
- Frames can be sent *only at the beginning* of a time slot.
- Vulnerable Time = T_t

Efficiency of Slotted Aloha

$$S = G.e^{-G}$$

$S_{max} = 0.368$ at $G = 1$.

CSMA (Carrier Sense Multiple Access)

- Each node will sense the medium before sending.
- If the medium is idle, send the data. Otherwise wait.
- Collisions may still occur due to propagation delay.

For e.g, if A sent a frame at 12:01, and propagation time from A to D is 2 minutes. If D checks the medium at 12:02, it will find it idle and send the frame. A's frame and D's frame will then collide.

- Vulnerable time = T_p (Max propagation time).

Persistence Methods for CSMA

Persistence methods decide when and how to send data after sensing medium.

1-Persistent

- Continuously Sense the medium.
- As soon as the medium is idle, send the data immediately.

Non-Persistent

1. If medium is idle, send the data immediately.
2. If medium busy, wait a random amount of time, then sense the medium again and repeat from step 1.

- Less efficient, as the channel may remain idle in the random waiting time.
- Less chance of collision.

P-Persistent

Uses a value p that is fixed by the network administrator for each node. Different nodes have different values of p 1. Continuously check medium till idle. 2. If idle: - Generate random number r - If $r < p$, then transmit the data. - Else: - Wait for a time slot, and then check the line. - If line is idle, go to step 2. - If the line is busy, act as if a collision occurred, and follow the buyback