

# Computer Networks

Satvik Gupta

April 6, 2023

## Contents

<b>Introduction</b>	<b>2</b>
Communication link: . . . . .	2
Goals of Networks . . . . .	2
Data communication . . . . .	3
Components of Effective Data Communication . . . . .	3
Components of Data communication system . . . . .	3
Types of Communication . . . . .	3
Physical Structure . . . . .	3
<b>Physical Topology</b>	<b>3</b>
Bus Topology . . . . .	3
Ring Topology . . . . .	4
Star Topology . . . . .	4
Mesh Topology . . . . .	5
Tree Topology . . . . .	5
<b>Networks Based on Geographical Area</b>	<b>5</b>
<b>OSI Model - Open Systems Interconnection</b>	<b>5</b>
Data in layers: . . . . .	6
<b>Physical Layer</b>	<b>6</b>
Transmission Media . . . . .	7
<b>Wired/Guided Media:</b> . . . . .	7
<b>Wireless/Unguided Media:</b> . . . . .	8
Propagation Methods . . . . .	8
Bands . . . . .	9
Switched Networks . . . . .	9
Network Architecture . . . . .	9
Types of Communication on the basis of Connection . . . . .	9
Connection-Oriented . . . . .	9
Connectionless . . . . .	10
<b>TCP/IP Stack</b>	<b>10</b>
Internet Protocol (IP) . . . . .	10
Transmission Control Protocol (TCP) . . . . .	10
User Datagram Protocol (UDP) . . . . .	10
<b>Data Link Layer</b>	<b>10</b>
Delay . . . . .	10
Transmission Delay . . . . .	10
Propagation Delay . . . . .	10
Queueing Delay . . . . .	10

Processing Delay . . . . .	11
Framing Techniques . . . . .	11
Character Count . . . . .	11
Flag (Character Stuffing/Byte Stuffing) . . . . .	11
Bit Stuffing . . . . .	11
Error Detection and Control . . . . .	12
Parity Check . . . . .	12
Hamming Code . . . . .	12
Checksum . . . . .	14
CRC (Cyclic Redundancy Check) . . . . .	15
Flow Control . . . . .	16
Stop & Wait ARQ (Automatic Repeat ReQuest) . . . . .	16
Formulas . . . . .	17
Go Back N . . . . .	18
Selective Repeat (SR) . . . . .	18
Data Encoding Techniques . . . . .	19
<b>Media Access Sublayer</b> . . . . .	<b>21</b>
Media Access Control and Multiple Access Protocols . . . . .	21
Random Access/ Contention Methods . . . . .	21
Pure Aloha . . . . .	21
Slotted Aloha . . . . .	22
CSMA (Carrier Sense Multiple Access) . . . . .	22
Persistence Methods for CSMA . . . . .	23
CSMA with Collision Detection (CSMA/CD) . . . . .	23
<b>Network Layer</b> . . . . .	<b>23</b>
Classful Addressing . . . . .	24
Class A . . . . .	24
Class B . . . . .	24
Class C . . . . .	25
Class D . . . . .	25
Class E . . . . .	25
Classless Addressing . . . . .	26
Rules for Classless Addressing . . . . .	26
Subnetting . . . . .	26
Subnetting in Classful Addressing . . . . .	26
Subnetting in Classless Addressing . . . . .	27
Variable Length Subnet Masking (VLSM) . . . . .	28
Header Formats for IP Protocols . . . . .	29
IPv4 Header Format . . . . .	29
IPv6 Header . . . . .	33

## 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.

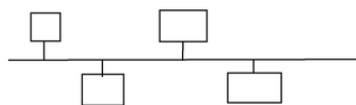


Figure 1: Bus Topology

### *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**

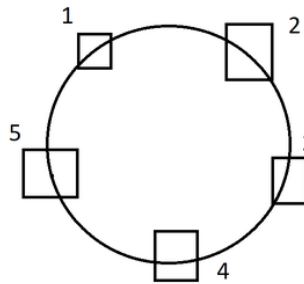


Figure 2: Ring Topology

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.

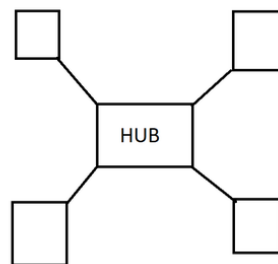


Figure 3: Star Topology

\_\_ Advantages and disadvantages same as of any centralized system\_\_ Hub can also be expensive.

## Mesh Topology

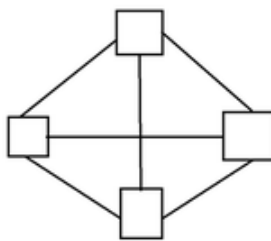


Figure 4: Mesh

### **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.

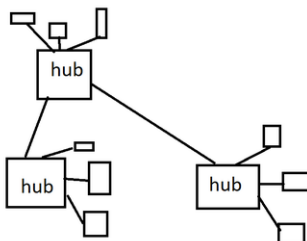


Figure 5: Tree Topology

## 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.

7	Application Layer	Human-computer interaction layer, where applications can access the network services
6	Presentation Layer	Ensures that data is in a usable format and is where data encryption occurs
5	Session Layer	Maintains connections and is responsible for controlling ports and sessions
4	Transport Layer	Transmits data using transmission protocols including TCP and UDP
3	Network Layer	Decides which physical path the data will take
2	Data Link Layer	Defines the format of data on the network
1	Physical Layer	Transmits raw bit stream over the physical medium

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

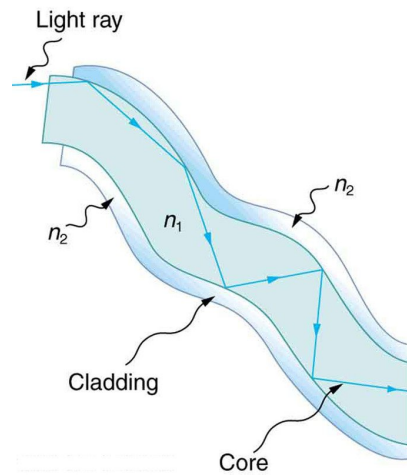


Figure 7: Transmission through Optical Fiber

- Coaxial Fiber

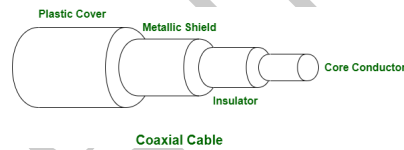


Figure 8: Coaxial Fiber

- Twisted pair cable



Figure 9: Twisted Pair Cable

- Unshielded Twisted Pair (UTP)



Figure 10: USTC

- Shielded Twisted Pair (STC)

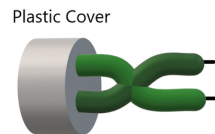


Figure 11: STC

### 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

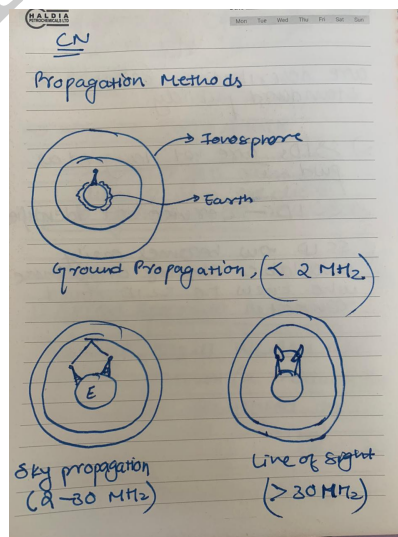


Figure 12: Propagation Methods



## 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.

## Data Link Layer

### 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.)

$$T_p = \text{distance}/\text{velocity}$$

#### 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

01101111111111111110010 → 011011111011111011111010010

## 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 and 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  $2^0, 2^1, 2^2, 2^3 \dots$  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

$$\text{Bit 3} = 0$$

$$\text{Bit 5} = 1$$

$$\text{Bit 7} = 0$$

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)

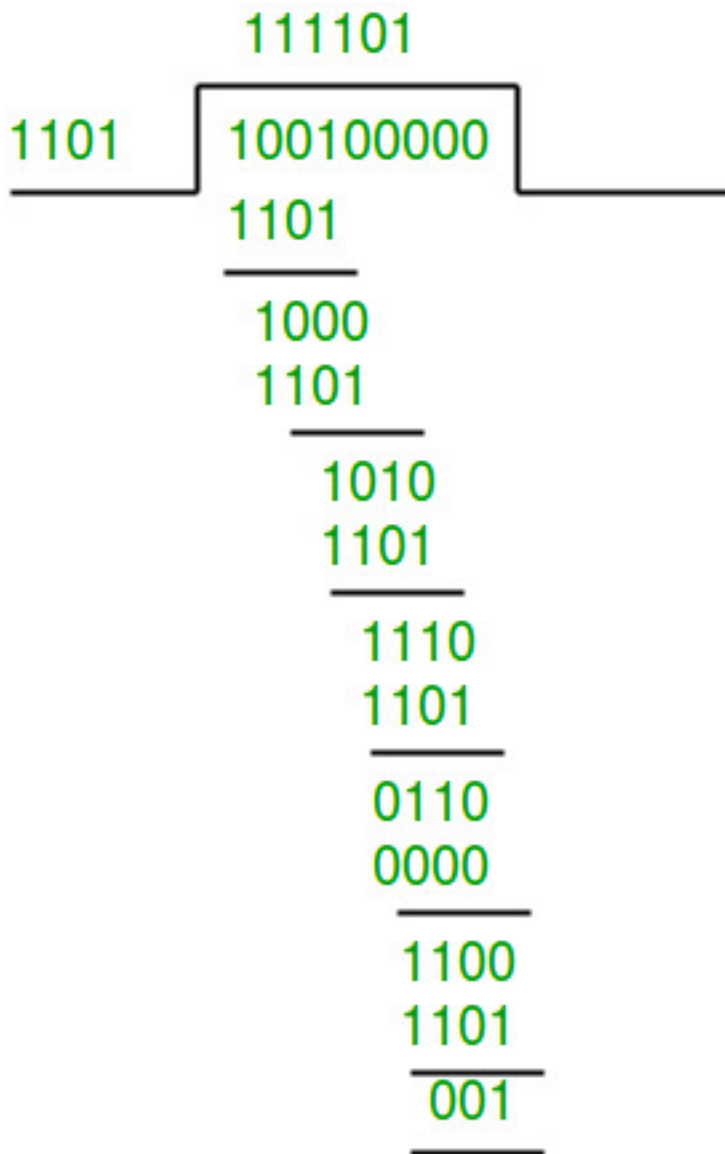
At receiver's side:

- 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

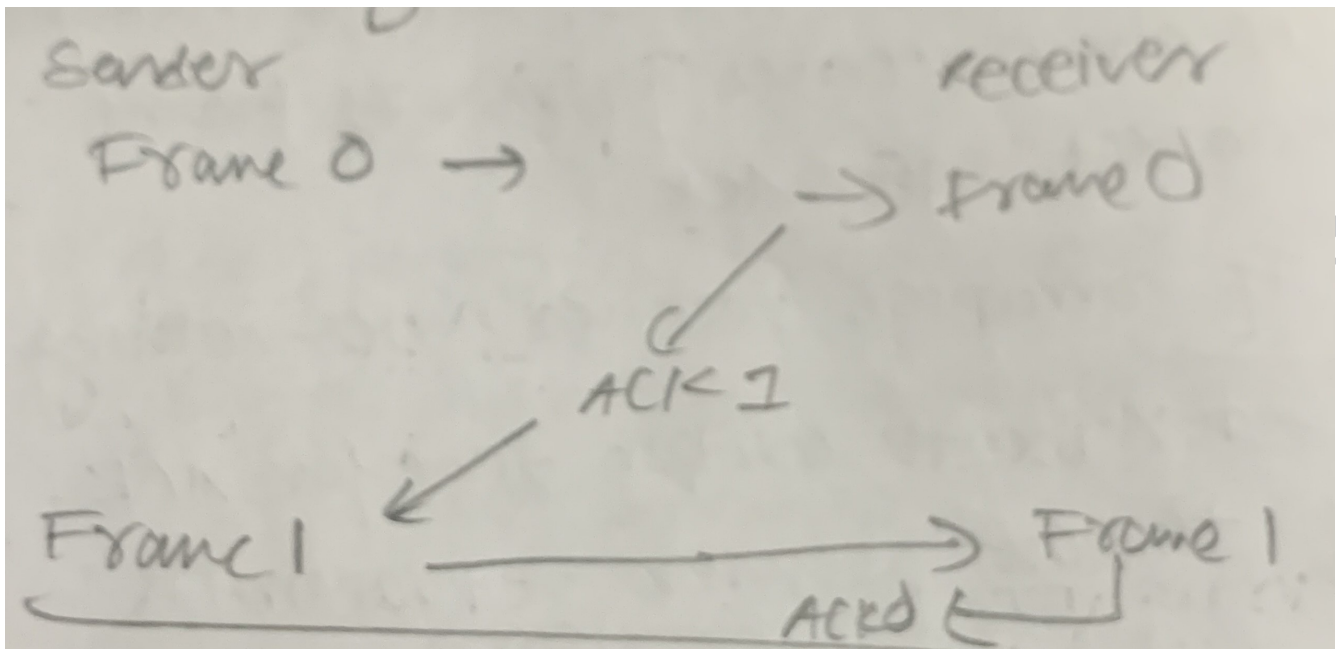
To make sure receiver receives all the data.

### 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.

$$\text{Total time} = T_t + 2 * T_p$$

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

$$\text{Efficiency} = \eta = \text{Useful Time} / \text{Total Time}$$

$$\begin{aligned} &= \frac{T_t}{T_t + 2 * T_p} \\ &= \frac{1}{1 + 2(\frac{T_p}{T_t})} \\ &= \frac{1}{1 + 2a} \end{aligned}$$

where  $a = T_p/T_t$ .

$$\text{Throughput} = \eta * \text{Bandwidth}$$

## Go Back N

- Sliding Window Protocol
- Receiver window Size = 1
- Sender Window Size =  $2^m - 1$
- Sequence numbers for frames –  $[0, 1, \dots, 2^m - 1]$ , 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.

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

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.

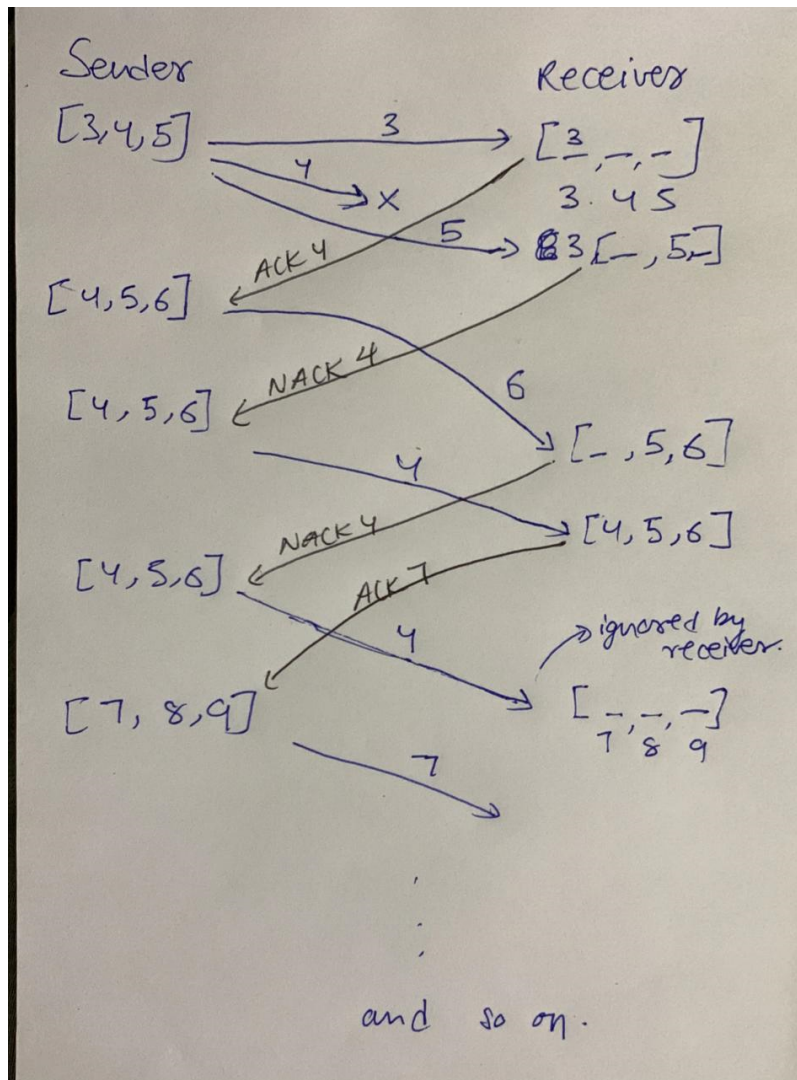


Figure 13: Selective Repeat Example

## Data Encoding Techniques

### NRZ-Unipolar

1 - +ve

0 - 0

### NRZ-Polar

1 - +ve

0 - -ve

### NRZ-I

Differential Encoding.

1 - Signal transition at start (high-to-low or low-to-high)

0 - No signal transition at start

### Manchester

Always has a mid-bit transition:

1 - Low to High

0 - High to Low

The start of the bit may also have a transition, if needed according to the bit's value.

For eg, if the bit is 1 (which means we need a low to high transition in the middle), and the interval starts with a high value, we will transition to low at the start.

(Eg - Encoding 11)

### Differential Manchester

Mid-bit transition is only for clocking purposes.

1 - Absence of transition at the start.

0 - Presence of transition at the start.

### Bipolar Encoding

1 - Alternating  $+1/2$ ,  $-1/2$  voltages

0 - 0 voltages

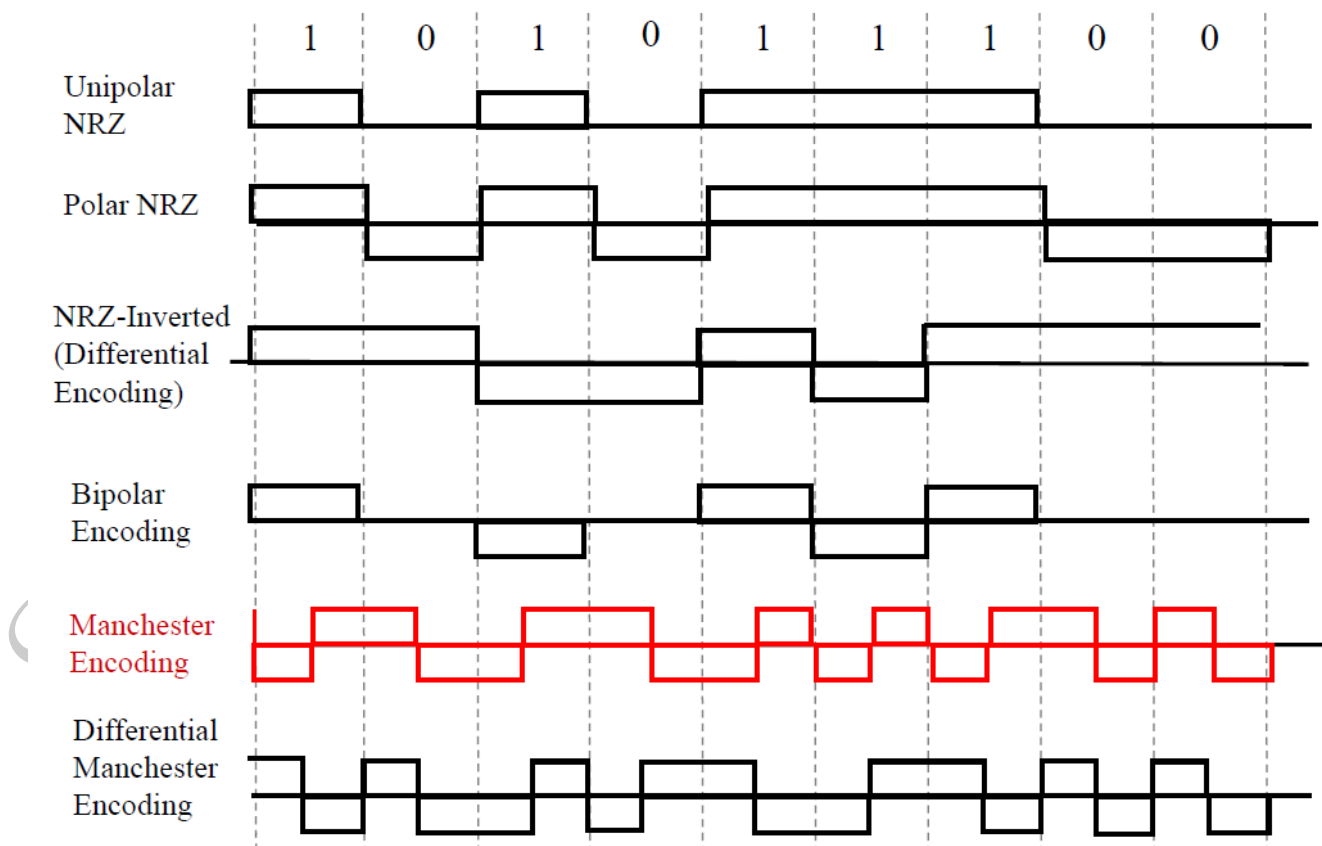


Figure 14: Data Encoding Techniques

## 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  $G = 1/2$ .

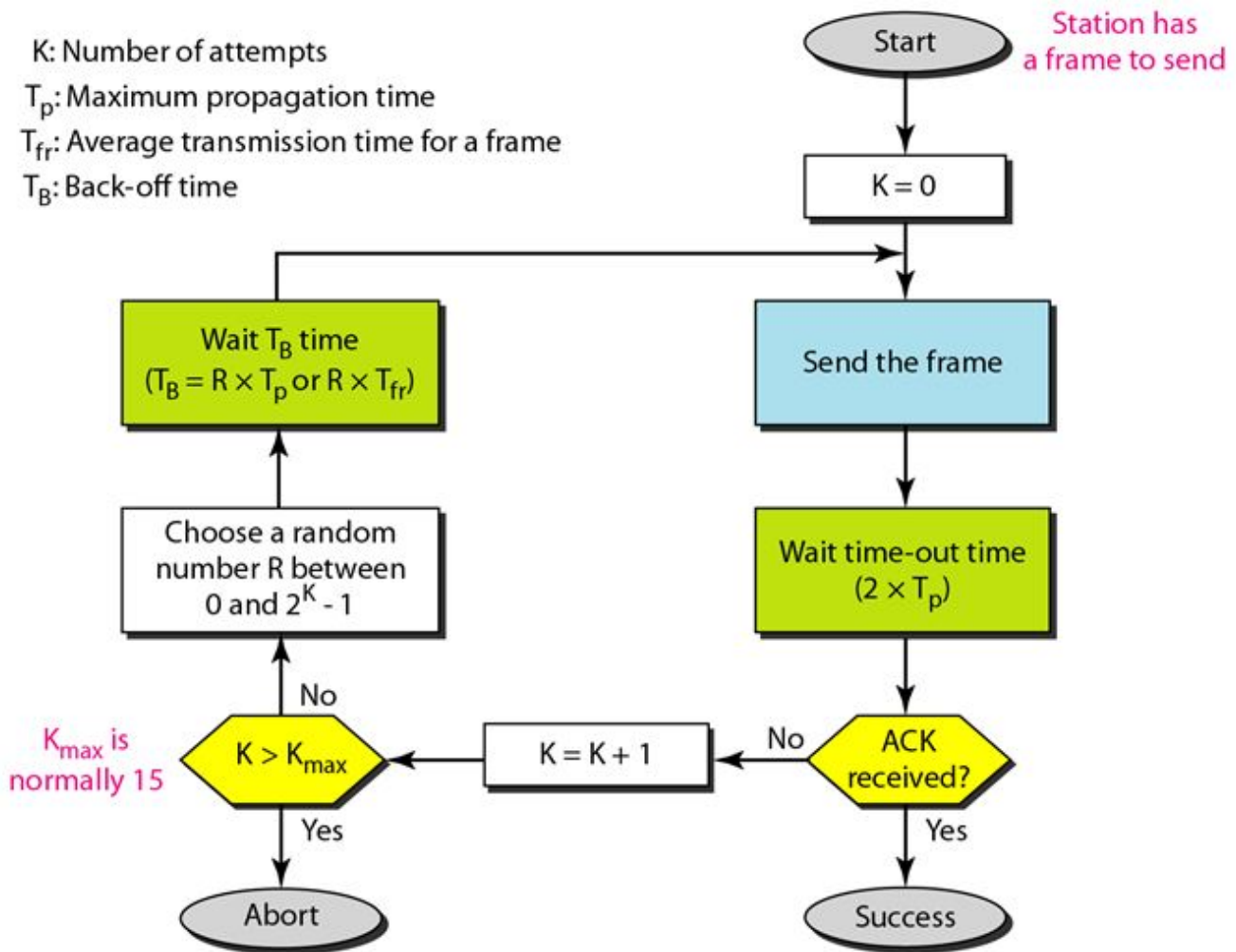


Figure 15: Flowchart for Aloha

The procedure of choosing a random number between 0 and  $2^K - 1$ , incrementing the value of  $K$ , and waiting an amount of time based on  $R$ , is called the **backoff procedure**.

### 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 backoff procedure.

## CSMA with Collision Detection (CSMA/CD)

- Continue sensing medium for time =  $2 * T_p$  after transmission.
- This will help us sense potential collisions.
- If a collision is detected, the node that detected the collision will send a jamming signal to the access medium. After that, backoff procedure will be applied.
- Condition to detect collision:

$$T_t = 2 * T_p$$

The transmission time for the frame should be long enough that we can detect collisions while we are transmitting it.

- Efficiency of CSMA/CD is

$$\eta = \frac{1}{1 + 6.44a}$$

where  $a = T_p/T_t$ .

## Network Layer

- Moves packet from source to destination
- Uses routers, bridges, switches.
- Deals with IP addresses.

Performs the following functions:

- Routing
- Fragmentation
- Congestion Control

IP addresses have two parts - network ID, and host ID. Network ID represents which network the IP address is part of, i.e, which organization controls it. Host ID represents which *computer* (or mobile, printer, etc.) in that network the IP address belongs to.

For e.g, Google has many servers. They will all have the same Network ID, but different Host IDs.

## Classful Addressing

IPv4 addresses were divided into various classes for easier addressing purposes.

An IPv4 address is a 32-bit address. For ease of representation, it is divided into 4 octets. Each octet contains 8-bits.

(8)(8)(8)(8)

It allows us to write the IP address in the following format - 192.168.10.1 Here, 192 is the first octet, 168 is the second, and so on. All are represented in decimal. This is called Dotted Decimal Representation.

### Class A

In this, the first bit of the first octet is always fixed as 0.

0 \_ \_ \_ \_ \_ . (octet 2) . (octet 3) . (octet 4)

The range of the first octet becomes 00000000 to 01111111, i.e. 0 to 127.

The 1st octet represents the Network ID (7-bits).

The 2nd, 3rd and 4th octet represent the Host ID (24-bits).

**Number of IP Addresses in Class A** =  $2^31$ , which is half of all IPv4 Addresses in the world.

**Total number of Networks** =  $2^7 - 2 = 126$ . Range is [1,126].

The network IDs 00000000 and 01111111 are reserved and unused. They aren't given to any organization.

**Total number of hosts in each Network** =  $2^{24} - 2$

Host with values 0.0.0 and 255.255.255 (first and last hosts) are also reserved and unused.

The host with value 0.0.0 is used to represent the entire network. So, for a network with networkID = 28, 28.0.0.0 is used to represent the entire network (and not any particular host in the network).

Host with value 255.255.255 is used to represent Direct Broadcast Address. If anyone wants to send a particular message to *all* the hosts in a network, they will use this. For a network with networkID = 28, 28.255.255.255 represents its direct broadcast address.

**Default Mask** for class A is 255.0.0.0

**For any IP address, performing bitwise AND of the IP Address, and the Default Mask for its class will give us the network that IP address belongs to.**

For eg, let the IP address be 28.12.34.1

First bit of first octet is 0, so we know it's a class A Address.

Bitwise AND with 255.0.0.0

00011100.00001100.00100010.00000001 (28.12.34.1)

11111111.00000000.00000000.00000000

-----  
00011100.00000000.00000000.00000000

= 28.0.0.0

28.0.0.0 represents the network that the IP address 28.12.34.1 belongs to.

---

### Class B

First octet has first 2 bits fixed as 10

1 0 \_ \_ \_ \_ . (octet 2) . (octet 3) . (octet 4)



First 2 octets represent the Network ID.

Last 2 octets represent the Host ID.

**Range** is (128-191)

**Number of IP Addresses** =  $2^{30}$ , 25% of all IPv4 addresses in the world.

**Number of networks** =  $2^{14}$

**Number of hosts in each network** =  $2^{16} - 2 = 65534$

First and last hosts are excluded here, as they were in class A.

*Class A excluded first and last networks also. Class B excludes first and last hosts only. All networks in Class B are available for use.*

**Default Mask** - 255.255.0.0.

---

### Class C

First octet has first 3 bits fixed as 110

1 1 0 \_ \_ \_ . (octet 2) . (octet 3) . (octet 4)

First 3 octets represent the Network ID.

Last octet represent the Host ID.

**Range** is (192-223)

**Number of IP Addresses** =  $2^{29}$ , 12.5% of all IPv4 addresses in the world.

**Number of networks** =  $2^{21}$

**Number of hosts in each network** =  $2^8 - 2 = 254$

First and last hosts are excluded here, as they were in class A.

*Class A excluded first and last networks also. Class C excludes first and last hosts only. All networks in Class C are available for use.*

**Default Mask** - 255.255.255.0.

---

### Class D

First Octet has first 4 bits reserved as 1110

1 1 1 0 \_ \_ \_ . (octet 2) . (octet 3) . (octet 4)

**Range** is (224-239)

**Number of IP Addresses** =  $2^{28}$

In class D, all IP addresses are reserved. There is no network ID and no host ID.

These IP addresses are only meant to be used for multicasting, group email, broadcasting, etc.

---

### Class E

First Octet has first 4 bits reserved as 1111

1 1 1 1 \_ \_ \_ . (octet 2) . (octet 3) . (octet 4)

**Range** is (240-255)

**Number of IP Addresses** =  $2^{28}$

All IP addresses are reserved for military purposes.

## Classless Addressing

This divides 32-bit IP addresses into BlockID and HostID.

BlockID is similar to NetworkID.

**Format** - x.y.z.w/n

$n$  represents the number of bits used to represent the blockID.

It also represents the number of 1-bits that the mask for this IP address should have.

For e.g - 128.225.1.1/10

This means the first 10 bits represent the block ID, and rest 22 bits specify host ID.

The mask for this network would be - 11111111.11000000.00000000.00000000, (10 times 1, then all 0)

To find the network, we do bitwise AND of IP address and mask

```
10000000.11100001.00000001.00000001
11111111.11000000.00000000.00000000
-----
10000000.11000000.00000000.00000000
```

= 128.192.0.0

The network that IP address belongs to is 128.192.0.0/10

## Rules for Classless Addressing

- Addresses should be contiguous
- Number of addresses must be in power of 2.
- 1st address of every block must be evenly divisible by block size (number of hosts)

Block Size = number of hosts =  $2^{32-10} = 2^{22}$  First address of block is same as network's IP address = 128.192.0.0/10 = 10000000.11000000.00000000.00000000 = 10000000110000000000000000000000

We don't need to divide. We can just check that the last 22 (22 is number of bits used to specify host) bits are 0.

i.e, the 22 least significant bits (LSBs) should be 0, which is true in our case.

## Subnetting

Subnetting is the process of dividing a large network into smaller *subnets*. This helps in organization, security, fixing bugs, etc.

An organization may divide its network into separate subnets for finance department, legal department, etc.

We do this by dividing the hosts.

***NetworkID REMAINS UNCHANGED IN SUBNETTING, ALWAYS.***

## Subnetting in Classful Addressing

Suppose we have a network with the IP 200.10.20.0. This is a Class C IP address.

It has a total of 254 hosts. We want to divide it into 2 equal subnets.

- Network ID is 200.10.20. This will remain unchanged.
- The last octet is used to specify the host. Let's say we want 2 subnets -  $S_1$  and  $S_2$ .
- We will subnet by fixing the first bit in the last octet as either 1 or 0.

$S_1$

For the last octet, we will prefix the first bit for  $S_1$  as 1.

$S_1$  will have IP addresses of the form 200.10.20.1 \_ \_ \_ \_ \_ .

**Range** => 200.10.20.128 - 200.10.20.255

**Usable IP addresses** =  $2^7 - 2 = 126$

**Network IP address** for  $S_1$  is 200.10.20.128.

**Broadcast IP address** is 200.10.20.255

$S_2$

For the last octet, we will prefix the first bit for  $S_2$  as 0.

$S_2$  will have IP addresses of the form 200.10.20.0 \_ \_ \_ \_ \_ .

**Range** => 200.10.20.0 - 200.10.20.127

**Usable IP addresses** =  $2^7 - 2 = 126$

**Network IP address** for  $S_1$  is 200.10.20.0.

**Broadcast IP address** is 200.10.20.127

Thus, if the network receives a packet meant for 200.10.20.50, it will be sent to  $S_2$ .

A packet meant for 200.10.20.165 will be sent to  $S_1$ .

### Subnet Mask

Since we are indirectly using 1 extra bit to specify Network ID now, our subnet mask will also have 1 extra bit added to it.

Class C's Subnet mask is 255.255.255.0

Now, it will be 255.255.255.1 \_ \_ \_ \_ \_ .

We added 1 in the place where we prefixed an extra bit.

Subnet mask for our network is now 255.255.255.128

---

Our total number of usable IP addresses has become  $126+126= 252$ . Earlier, it was 254.

Due to subnetting, we have lost the use of 2 IP addresses.

**Number of usable IP addresses** = Number Of Original Usable IP Addresses -  $n * 2$

where  $n$  is the number of subnets.

### Subnetting in Classless Addressing

The procedure is mostly the same as for subnetting in classful addressing. The only extra thing that needs to be done is to change the value of  $n$ .

For.eg,

128.192.0.0/10 - let this be our network.

First 10 bits specify block. Last 22 bits specify host.

**10000000.11000000.00000000.00000000.** The bold part represents block ID.

To subnet into two equal subnets, we fix the first host-bit.

Subnet 1 will have IP addresses of the form 10000000.1110\_\_\_\_\_. (We prefixed 0)

Subnet 2 will have IP addresses of the form 10000000.1111\_\_\_\_\_. (We prefixed 1)

To find out IP Addresses for Subnet1 and Subnet2, we will have to increase the value of  $n$  by 1, since now we are using an extra bit to find out block ID.

**Subnet1 IP Address** = 128.192.0.0/11

**Subnet2 IP Address** = 128.224.0.0/11

Accordingly, mask will be changed.

### Variable Length Subnet Masking (VLSM)

In case we want subnets of different sizes, we use this technique. It works for both classless and classful addressing in similar ways. We'll explain using classful here.

Suppose we have a network with IP 200.10.20.0.

We want to divide it into 3 networks -  $S_1, S_2, S_3$ .  $S_1$  should have 50% of all hosts.  $S_2$  and  $S_3$  should have 25% each.

We will subnet in the following way:

$S_1$

**IP Addresses of the form** - 200.10.20.0 \_\_\_\_\_ (1 bit prefixed)

**Range** - 200.10.20.0 - 200.10.20.127

**Usable IP Addresses** - 126

$S_2$

**IP Addresses of the form** - 200.10.20.1 0 \_\_\_\_\_ (2 bits prefixed)

**Range** - 200.10.20.128 - 200.10.20.191

**Usable IP Addresses** - 62

$S_3$

**IP Addresses of the form** - 200.10.20.1 1 \_\_\_\_\_ (2 bits prefixed)

**Range** - 200.10.20.192 - 200.10.20.255

**Usable IP Addresses** - 62

For  $S_1$ , we prefixed only 1 bit. For  $S_2$  and  $S_3$  we prefixed two bits. We can see that the  $S_1$  has roughly double the number of IP addresses that  $S_2$  and  $S_3$  each have.

We divided the network into two equal halves -  $S_1$ , and let's call the other part  $S'$ .

Then, we further divided  $S'$  into  $S_2$  and  $S_3$ .

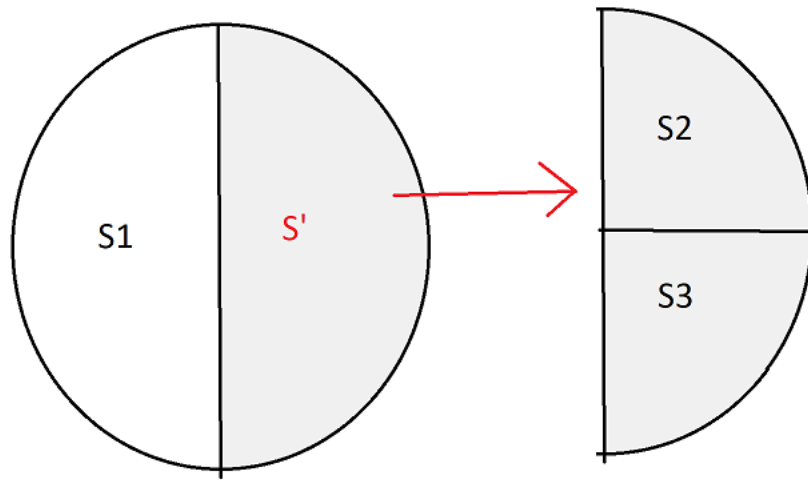


Figure 16: VLSM

## Header Formats for IP Protocols

Whenever a packet is sent using IP (IPv4 or IPv6), it includes data (payload), as well as a header. The header doesn't contain the actual data, but it contains metadata such as destination, priority, source, etc.

### IPv4 Header Format

- IPv4 is a **connectionless datagram** service.
- Header size is between 20-60 bytes.
- The total datagram size is at max 64 KB, or 65535 bytes.
- The payload size is maximum 65515 bytes. This happens when the header size is 20 bytes. In case the header size is larger than 20, the max payload size will be decreased accordingly.

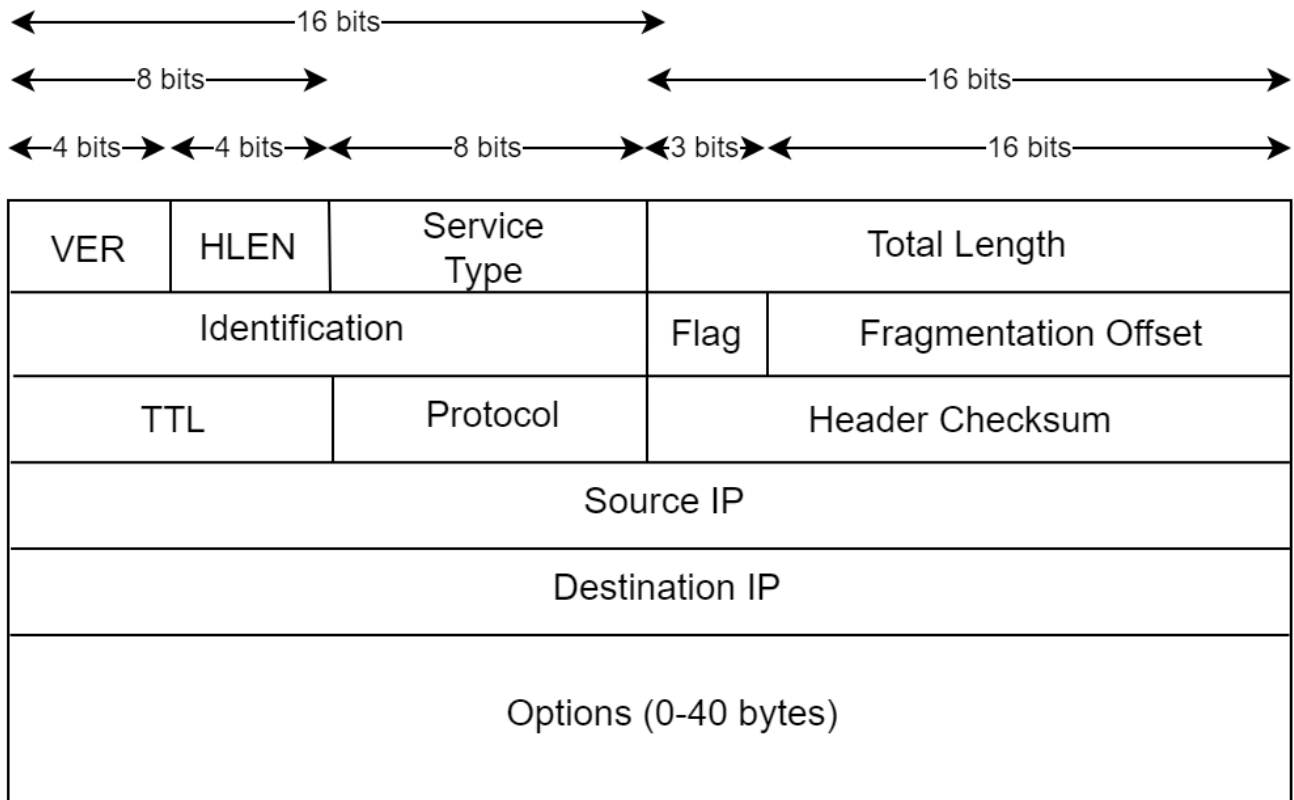


Figure 17: IPv4 Header Format

## VER

- Stands for version. It is a 4-bit value. It contains the value of version, i.e, which IP version is being used.
- Almost all transmissions either use IPv4 or IPv6.
- Version for IPv4 is 4 = 0100
- Version for IPv6 is 6 = 0110

## HLEN

- Contains the header-length.
- 4-bit value.
- Uses a factor of 4.

Header size =  $HLEN \times 4$ .

As the minimum header size for IPv4 is 20 bytes, HLEN can never be 0,1,2,3 or 4.

## Type of Service

- Also known as DSCP (Differentiated Service Code Point)
- 8-bit value
- Contains different values specifying the type of service we wish to use.

The 8 bits are:

[P] [P] [P] [D] [T] [R] [C] [O]

The first 3 bits (P) are used to set the precedence, or priority of the packet.

*D* - Delay.

0 means normal delay. 1 tells router this packet needs minimal delay.

*T- Throughput*

0 - normal

1 - maximize

*R - Reliability*

0 - normal

1 - maximize

*C - Cost*

0 - normal

1 - minimize

The last value is reserved as 0. It's fixed for future use.

### **Total Length**

Contains total length of the packet. 16-bit value.

### **TTL (Time to Live)**

- 8-bit value
- Source sets it to max value (255)
- At each node it encounters (router/switch/etc.), the value is reduced by 1.
- When it becomes 0, the packet is dropped.
- This helps in case a packet is getting stuck in loops in the network, causing congestion.

### **Protocol**

- 8-bit
- Tells which protocol is being used, TCP,UDP, etc.

### **Header Checksum**

- Contains the checksum value for the header.
- 16-bit

### **Source IP**

- Contains IP address of the source
- 32-bit

### **Destination IP**

- Contains IP address of destination.
- 32-bit

---

Fragmentation is done for packets that are larger in size than the permitted size.

The packet is broken down into many smaller packets, then sent over the network. It is reassembled at the source.

The IPv4 header contains values to identify the fragments.

### **Identification Bits**

This is a 16-bit unique packet ID. It identifies a group of fragments that belong to a single IP datagram.

## Flag

3-bit value.

[R] [D] [M]

The first bit is reserved as 0.

The second bit (D) stands for **Do not Fragment**. If this bit is 1, no node will try to fragment this bit. But, if some node doesn't allow a packet of a large size, and we set D=1, the node may drop that packet altogether.

The third bit (M) stands for **More Fragments**

If M=0, either this packet is the last packet in its datagram, or it's the only fragment.

If M=1, it means more fragments are coming after this packet.

## Fragment Offset

This represents the number of data bytes ahead of this particular fragment, i.e., the position of this fragment in the original unfragmented datagram.

It uses a factor of 8.

I.e., number of bytes ahead = Fragment Offset \* 8.

For eg, if a datagram is broken into 4 fragments of 80 bytes each (excluding the header length).

Fragment offset value for

1st fragment - 0 (No bytes ahead of it)

2nd fragment - 10 (80 bytes of data ahead of it.  $80/8=10$ )

3rd fragment - 20

4th fragment - 30

## Options

These contain optional headers and metadata. For example, they may be:

### Record Route

Tells the nodes to record the route this packet has taken in the header.

### Source routing

The source defines the route that the packet will take.

It is of two types.

1. Strict - The source defines the exact specific nodes the packet will travel through to get to its destination.
2. Loose - The source defines only loose requirements, not specific exact nodes.

Users cannot set source routing, only routers are allowed to do this.

## Padding

This is added in case the header size is not in multiple of 4.

We need to store the header length in HLEN, which uses a factor of 4.

So, if the header length is 21, we will add 3 bytes of padding to make it 24. Then HLEN will store  $24/4 = 6$ .



## IPv6 Header

- IPv6 uses 128-bit IP addresses instead of 32-bit.
- Only source can fragment packets. Intermediate nodes cannot.
- Header length is fixed at 40 bytes.

VER (4)	Traffic Type (8)	Flow Label (20)		
Payload Length (16)		Next Header (8)	Hop Limit (8)	
Source IP (128)				
Destination IP (128)				

Figure 18: IPv6 Header. Length of fields written in brackets.

**VER** is the same as in IPv4.

**Traffic Type** is the same as Type of Service in IPv4.

### Flow Label

20-bit value.

For continuous data that travels in a flow, such as video streaming, or live updates, we use flow labels. Even large files may be sent using flows.

Intermediate routers use source IP, destination IP and flow label to distinguish one flow from another. A source and a destination may have multiple flows occurring between them.

For e.g, if you're downloading two files together from Google Drive, they may be sent through 2 different flows.

So, we give them different flow labels to help identify which flow is which.

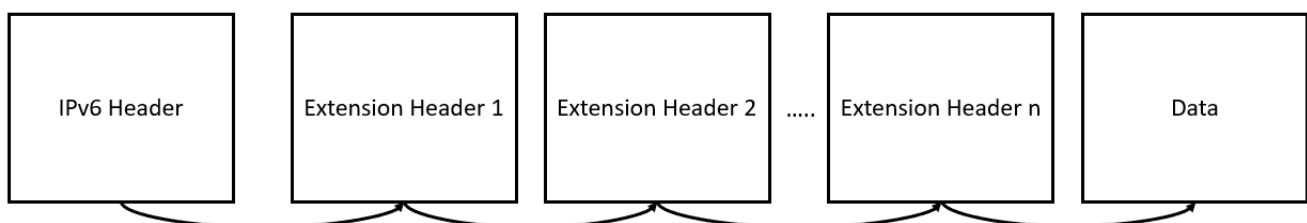
### Payload Length

Length of the payload

### Next Header

IPv6 uses extension headers instead of options. Metadata for routing, authentication, fragmentation, etc. are set in special extension headers.

This field is a 8-bit field that contains the type of the extension header (if present), that comes immediately after the IPv6 header. Each extension header contains its own "Next Header" field. The extension headers are thus chained together like this.



In some cases, this field is also used to indicate protocols in upper layers, such as TCP or UDP.

### **Hop Limit**

Same as TTL in IPv4 Header

**Source IP** and **Destination IP** are 128-bit IPv6 addresses for the source and destination respectively.

### **Extension Headers**

Extension headers may be used for many purposes. Some common extension headers used in IPv6 are:

- Routing headers, used if source wants to determine the route the packet should take.
- Authentication headers, used for security purposes.
- Fragmentation Header, used for fragments.
- Destination options - For data that we want only the destination to open, no intermediate routers
- Hop by Hop options, used to specify delivery parameters at each hop on the path to the destination.

---

Routing Protocols, ARP - better explained through videos.