MID-1

Class-1

Computer Networks- When multiple computers are connected and communicate with each other.
A series of nodes connected in a system where they can transmit, receive and exchange data.

How computer networks started- ARPANET was the 1st computer network.

Data- Continuous bit string or collection of bits.

Connecting devices- Switch, Hub etc.

Protocol- Rules and regulations.

EMI- Electromagnetic interference.

Router- A device that shows a path to the data from source to destination.

Bandwidth- It is the capacity of a channel.

IP Address - A 32 bit long address which is needed for networking of computer devices. IPv4 = 32 bits and IPv6 = 128 bits

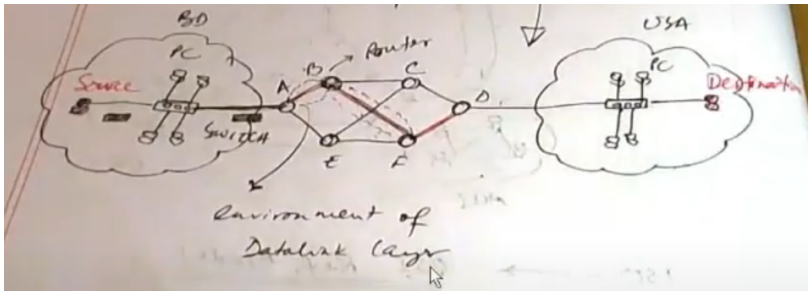
Transmission media - A medium which is used to transmit and receive data within devices.

OSI Reference Model-

The International Standard Organization (ISO) gave a standard design of computer networks called Open System Interconnection Reference Model (OSI Reference Model). It has a total of 7 Layers. Each layer is required for computers to communicate with each other and form a computer network. Each layer can come in use multiple times.

1. **Application Layer -** It holds all the software, algorithm, protocols, applications needed to have a network system. It is the opposite of the physical layer.
2. **Presentation Layer -** It holds the syntax and semantics of the data which helps how the data should be displayed. (Example: HTML tag being converted by the browser to show us the data etc.)
3. **Session Layer -** Maintaining the session/connection from the source to destination for that instance of data communication. Sometimes data might need to be sent over multiple sessions.

4. **Transport Layer -** Supervises the whole process of the data generation and transmission from source to the data being received on the destination.
5. **Network Layer -** Shows the path from source to destination.
6. **Data link Layer -** Creating an error free link between adjacent 2 devices.
7. **Physical Layer -** All the hardware specifications required to transmit, create, receive the data.(Example: Transmission media, router, Network interface card etc.)



Class-2

TCP(Transmission control protocol)/IP Reference Model-

It is derived from the OSI Reference Model. Total of 4 Layers. It is the most used model. **[** TCP is a protocol of the transport layer and IP is a protocol of the Internet layer **].**

1. **Application Layer(Application Layer + Presentation layer) -** Same as OSI Reference model.
2. **Transport Layer -** Same as OSI Reference model. (adds a 20 byte header to the data)
3. **Internet Layer/Internetwork Layer (Network Layer) -** Same as the network layer of OSI Reference model. (adds a 20 byte header with the destination IP address)
4. **Host to Network Layer (Data link Layer + Physical Layer) -** It's a combination of data link layer and physical layer. Same as the data link Layer and physical Layer of OSI Reference model. Host to Network Layer is responsible for the data moving from source to the network.

Application Layer(Application Layer + Presentation layer)
Transport Layer
Internet Layer/Internetwork Layer (Network Layer)
Host to Network Layer (Data link Layer + Physical Layer)

[Application protocol data unit, Transport protocol data unit, Network protocol data unit]

Hybrid Model-

The Hybrid Model is used to explain how the TCP/IP reference model works. The source and The destination both have these Hybrid Model layers. Now the simulation of how a data is sent using this model is given bellow,

In Case of the Source: Here we will simulate what happened in the source.

1. **Application Layer -** 1st writing the data to send. Then this layer sends the data to the transport layer.
2. **Transport Layer -** This layer wraps the data with a header. It has 2 parts. The 1st part is a 20 byte header which holds necessary information needed for the transport layer and the 2nd part is the payload which carries the data. In short the transport layer just adds a header to the data with the information for the management and control of a data stream. It is called the **Transport Protocol Data Unit** or **TPDU**. Then it sends the TPDU to the Internet Layer (Network Layer).
3. **Internet Layer (Network Layer) -** This layer again wraps the TPDU called a Packet. The Packet has a 20 byte header which is again added to the TPDU which holds an IP address of the destination along with some other information. Then It sends the Packet to the Data link layer in the Host to Network Layer.
4. **Host to Network Layer -**
    a. **Data link layer -** This layer again wraps the Packet which is called Frame. It has a header which holds necessary information, a payload which holds the Packet from the Internet layer and a Trailer which holds **CRC or Cyclic Redundancy Check** which fixes errors in the Frame if there are any. Then It sends the Frame to the Physical layer in the Host to Network Layer.
    b. **Physical Layer -** This layer sends the frame in bits through the network to the destination.

This whole wrapping process of the data is called **Encapsulation** which is done in order to send the data.

In Case of the Routers: Here we will simulate what happened in the Routers.Since routers work with adjacent devices. So most of the work will be done in the Data link layer.

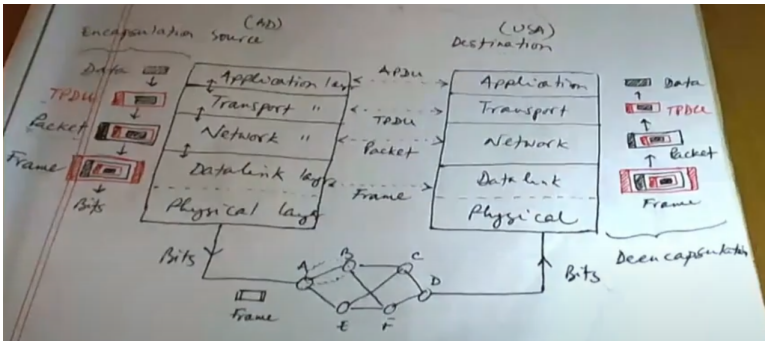
1. **Application Layer -** This layer will help the network layer with the routing algorithms to decide a path for the packet to reach its destination.
2. **Transport Layer -** This layer has no work for routers
3. **Internet Layer (Network Layer) -** This layer will receive the Packet from Host to Network layers data link layer. Then it will check the header of the packet for the IP address of the destination and check if the packet was for this router. Then it will use the routing algorithm and decide a path for the packet to reach its destination. After deciding the path it will send the packet again to the data link layer.
4. **Host to Network Layer -**

a. **Data link layer -** This layer will receive the Frame. Then check the header and trailer to check if the data was for this route and if the frame has any errors. If all is ok then it will send the Packet in the payload to the Internet Layer(Network layer). It will receive the Packet again from the Internet Layer(Network layer) when it confirms the destination path and wraps the Packet in a frame and sends it to the next destination router. That's how it will send the Frame to the destination.
b. **Physical Layer -** This layer will receive the data in bits meaning the router will receive the data. Then sends the Frame to the Data link layer. Again it will get the Frame form the Data link layer and send it to the next destination router.

In Case of the Destination: Here we will simulate what happened in the Destination. Here the process will start from the physical layer.

5. **Application Layer -** This layer will receive the data from the Transport layer. Then it will open the data and with the help of the presentation layer, it will look at the syntax and semantics and then display the data to the user.
6. **Transport Layer -** This layer will receive the TPDU from the Internet layer. It will check the information in the Header of the TPDU and if all is ok then it will send the data in the TPDU to the Application layer.(There are many applications so it will send to the designated application it needs to go from the header information).
7. **Internet Layer (Network Layer) -** This layer will receive the Packet from Host to Network layers data link layer. Then it will check the header of the packet for the IP address and check if the packet was for this destination. If it's happy with the packet then it will transfer the TPDU in the packet payload to the transport layer.
8. **Host to Network Layer -**
    c. **Data link layer -** This layer receives the Frame from the Physical layer. Then check only the header to check if the data was sent to this destination and trailer to check if there are any errors to fix. If it's happy with the frame then it will transfer the Packet in the Payload of the Frame to the Internet Layer.
    d. **Physical Layer -** This layer will receive the data in bits. Then sends the Frame to the Data link layer.

This whole unwrapping process of the data is called **Deencapsulation** which is done in order to receive the data.



Class-3

Frame Size- While creating frames we can not exceed the size depending on the frame mode. These modes can only transfer data frames of a certain size.

1. **Ethernet Frame -** Max size 1460 Byte which it can transfer.
2. **ATM (Asynchronous Transfer Mode) Network -** Cell(Frame) Max size 53 Byte which it can transfer.

How are Frames created? -

There are 4 ways of creating data frames,

Character count - Add a character count of the characters in the beginning of the character set you want to transfer in a frame. It will also include the count of the character which represents the character count. The receiver will get the frame and read the 1st character which is the character count and count the characters which will tell it how many characters are in the frame.

Example: Here the character data will be sent in 3 frames. Red marks are the character count which will tell the receiver the frame's ending point.

7	6	3	4	9	2	3	5	7	8	5	3	8	2	1

5	7	6	3	4	7	9	2	3	5	7	8	6	5	3	8	2	1

Sometimes there might be some errors while reading the character counter and the frame end point will get messed up and give wrong data to the receiving end.

Byte Stuffing - It is the same as character count but instead adds a flag in front and end of the byte. One will be considered as the beginning flag and another will be considered as the ending flag. These flags will indicate the Frame. The receiver will ignore the flags and take in the bytes in the middle as a frame.

Flag	A	B	C	D	Flag

Sometimes a Flag might be a character included in the frame and at that time the receiver will think of that flag as an end point.

Flag	A	B	Flag	D	Flag

So to fix this issue the sender can instruct an Ecs character bit pattern for the receiver to know which flag is not the end point of a frame. The Esc character has to be placed in front of the flag sent as a data.

Flag	A	B	Esc	Flag	D	E	Flag

Here, the receiver will read through the byte data and if it reads a Esc character then it will skip the Esc character and read whatever the next byte data is. So, for the above the data Frame is

A	B	Flag	D

Sometimes there might be a Ecs character bit pattern as a data.

Flag	A	B	Esc	C	Flag	D	E	Flag

So it can also be solved in the same way as the flag problem. The sender can instruct the Esc bit pattern at the beginning of the Esc and flag character data.

Flag	A	B	Esc	Esc	C	Esc	Flag	D	E	Flag

Here, the receiver will read through the byte data and follow the same instruction and skip the Esc character and read whatever the next byte data is. So, for the above the data Frame is

A	B	Esc	C	Flag	D	E

Example: This is the data,

A	B	Esc	Esc	C	D	E	Flag

This is the sender instruction for the receiver,

Flag	A	B	Esc	Esc	Esc	Esc	C	D	E	Esc	Flag	Flag

That is how bite stuffing works.

Bit Stuffing - Here, the data frame is considered as individual bits. It is the same as byte stuffing but it has a fixed flag. The fixed flags to identify start/end of frame is 01111110. The receiver will skip the start/end flag and take the data frame inside.

The data: 1101010111001000101

The frame: 01111110110101011100100010101111110

Sometimes it can have the same flag as data in the frame, and may read the wrong frame data.

The data: 10101110111111101110010

The convention to handle this type of situation is, if there are 5 consecutive 1's then the sender has to insert a 0 after those 5 consecutive 1's.

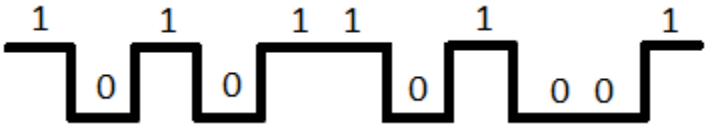
The data: 10101110111111101110010
The frame: 01111110101011101111101011100100111111 0

The receivers instruction will be, if the receiver reads a 0 after 5 consecutive 1's then the receiver has to omit the 0 and take whatever is after the 0. For the flags it will look for 01111110. If it reads 01111110 then the receiver will know that it is a flag and will take the data inside the start and end flag.

The frame: 01111110101011101111101011100100111111 0
The receiver: 10101110111111101110010

Example: Here there is a 0 after 5 consecutive 1's in the data
The data: 10101110111111010111001 0
The frame: 01111110101011101111100101110010011111 0
The receiver: 10101110111111010111001 0

Physical layer coding violation- In physical layer encoding we send data in a bit string in binary encoding, where 1=high and 0=low voltage. So, for 10101101001 the **binary encoding** is,



The receiver will know 1s and 0s depending on the high and low voltages but how will it know if there are consecutive 1's or 0's ? There is a thing called **bit time**. It is the time duration of the voltage.

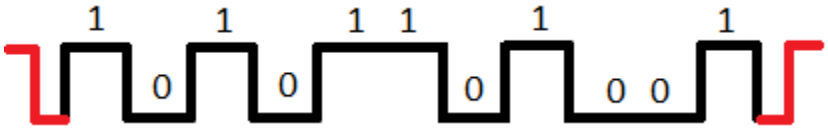
Say the high/low voltage duration is 5 microsecond. So if the receiver gets high voltage for five microsecond and then low voltage for 5 microsecond then it will know that the data is 1 and 0.

So, If there are 2 consecutive 1's then the bit time of high voltage will be 5*2=10 microsecond. And if there are 3 consecutive 0's then the bit time of low voltage will be 5*3=15 microsecond. So if it's 10 microseconds of high voltage then there are 2 consecutive 1's and if it's 15 microsecond of low voltage then there are 3 consecutive 0's.

So it's all a matter of signals. This is **binary encoding**.

Now the violation of this mechanism will be a framing technique. If we give the bit time of 2.5 microsecond for a bit when the selected bit time is 5 microsecond then it will be a violation.

So if we set a violation at the start and end of the data then it will become a frame for the receiver to understand.



To incorporate this violation one will need porper hardware. The design also matters, if the bit time is 5 microsecond and the violation time is 4.9 or 5.1 microsecond then the receiver might get confused even though theoretically it is a violation. So, the violation time can not be too close to the given bit time.

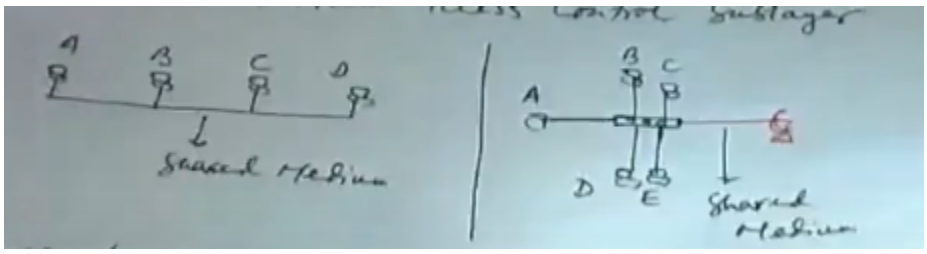
Class-4

Lab- CISCO Packet Tracer 7.2.1

Sub layers of Data link layer In TCP/IP Reference Model-

The Data link layer has 2 sub layers,
Logical Link Control (LLC) Sublayer - Frames are created in this sublayer.
Medium Access Control (MAC) Sublayer - It is a protocol that controls which device/PC will transmit at a time in a shared medium. (Example: Multiple Frames/data from 2 or more computing devices might collide with each other and create a garbage data bit signal when they try to transfer the data at the same time on the shared medium. To stop that MAC controls which device will transmit the data at a time)

In a bus topology if device 'A' transmits a frame and at the time of transmitting in the medium device 'C' transmits a frame, the both frames will collide and the collide signal will propagate and spread in the medium and go to the connected computing devices.



We can allocate channels to stop these data collisions. There are 2 ways.

1. **Static Channel Allocation in LAN (Not used anymore)**

2. **Dynamic Channel Allocation in LAN**

Dynamic Channel Allocation in LAN-

There are few assumptions which we have to take into consideration when creating protocols for Dynamic channel allocation. They are,

Station Model - N numbers of independent stations. Meaning there are N numbers of computing devices.

Single Channel - For all the N numbers of computing devices there will be only a single channel in the medium for all the devices to use.

Collision Assumption - When a computing device transmits data in the shared medium and at the same time another device transmits data then it is bound that the frames will collide and the collided signal will generate garbage frames which will propagate. And the devices involved in the collision will have to re-transmit the frame/data again. Even if one bit of a frame overlaps with another, it will count as a collision and turn into a garbage frame.

Time Assumption - There are 2 types of time assumptions.

1. **Continuous -** Can send data when they have/need to. There will be no problem.
2. **Slotted -** To send data the device needs to wait for its time specified time slot.

Sense - Sense assumption also comes in 2 types,

1. **Yes -** It means if the device has the capability to sense if the carrey/medium is being used by other devices. If device ‘A’ wants to send data then it will sense the medium to see if any other devices are transmitting data at that time or is the medium free to use. It needs good processing power and battery.
2. **No -** The opposite of Yes, meaning that the devices can not sense the medium.

Depending on these assumptions some protocols have been designed.

Protocols -

There have already been many protocols designed. They are,

ALOHA - It is a very primitive and old protocol. If there are multiple devices, a device can send whenever they have/need to. No need to wait for a specified time slot. Since not all the devices have or need to send data 24/7, there is a lot of idle time and the probability of multiple devices sending data at the same time is small. So the transmission will be successful most of the time. But the problems will occur if the number of computing devices starts to increase in a LAN. This is a setback for this protocol.

Slotted ALOHA - It is also a very primitive and old protocol. To solve the ALOHA setbacks Slotted ALOHA was created. It is the same as ALOHA but the device has to send data in a given slot. The devices have to wait for their slot even if they have data to transmit. Even if no other devices have data to send and the transmission media is free, the device which has to send data will have to wait for its time slot. This is a setback for this protocol.

Carrier Sense Multiple Access (CSMA) - Sense if the carrier/medium is not in use or busy. If the medium is idle then transmit data else refrain from transmission. If the medium is busy then 2 things can happen,

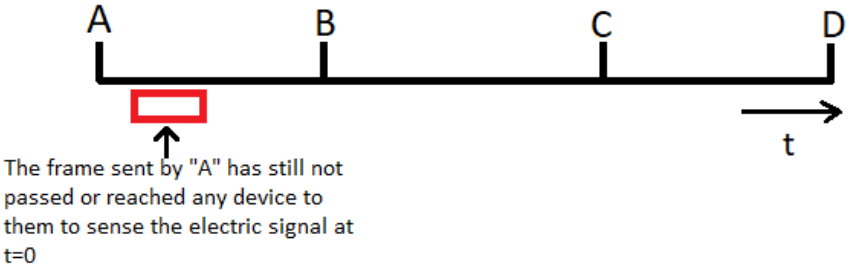
1. **Persistently Sense -** Don't do other jobs and continuously keep on looking to see if the channel is busy. If the transmission media is empty then transmit data. It guarantees that it will get a free medium to transmit the data later on. It uses processing power to continuously sense. If there is current/voltage in the medium then the channel is in use by other devices. That's how it senses/detects if the channel is in use.
2. **Non-Persistently Sense -** Do other jobs while the channel is busy and try again at a later time. It does not guarantee that it might get free medium to transmit the next time it tries to sense it. It saves processing prower.

Collisions still can happen following CSMA protocol.

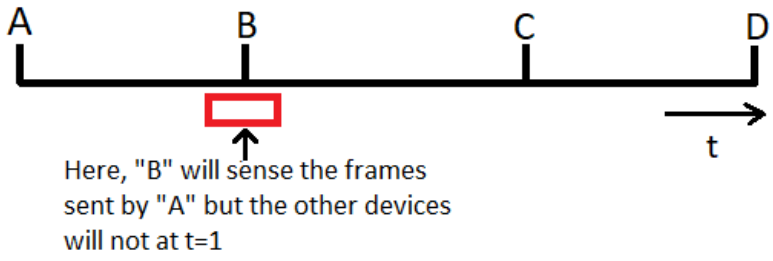
Collisions can occur following CSMA Protocol:

From a computing device, data is sent in an electrical signal. The speed is about 2.25 * 10^8 m/s.

At t=0 time “A” transmitted a frame. And at the same time t=0 the fram did not pass/reach any other device. So the other devices will sense that the medium is free at t=0 and will also transmit frames. Even though the speed of the electric signal is very fast, there is a thing called **propagation delay**. The signal will still take some time. And if the devices are far apart from each other then the propagation delay will be higher. So at t=0 time even though “A’” has sent a frame, other devices will sense the medium is idle/free and will also transmit their data/frame which will end up colliding and the collided signal will propagate through the medium.



Even if one device senses the frame at some point and sees that the medium is in use the other devices will still think that the medium is idle since the signal didn't pass/reach their sense area.



Class-5

Carrier Sense Multiple Access with Collision Detection (CSMA/CD) - How is the collision detected? Source transmits a signal which propagates through the transmission medium and also receives the same transmitted signal. If the transmitted signal and received signal are different to each other then the source will understand there has been collision between the signals.
It is inherently half-duplex and can not be used as a full duplex. Because the receiving end is busy receiving its own transmitted data.That is how it was designed.

Contention Period -
Here,

τ (tau) = propagation delay between 2 farthest devices(worst case)

ε = Epsilon, it is a tiny amount of time

$(\tau - \varepsilon)$ = is a tiny amount of time before the farthest device can sense that the medium is being used. If a collision happens then D will know instantly since its closest. But for A it will take the propagation time. $(\tau - \varepsilon)$ is the last moment to detect collision. After this time no collision will happen because the farthest device will start receiving the data and know the medium is in use. It will also take the same $(\tau - \varepsilon)$ time for the 1st device A to know that if a collision has occurred.
So,

$(\tau - \varepsilon) + (\tau - \varepsilon)$ [time to collision + time for 1st device to know collision]

$= 2\tau - \varepsilon$

$= 2\tau$

So, **Contention Period** = 2τ (tau). It is the total time for the 1st device to know if the data it sends has collided or not.

So after a frame transmission the sending device has to wait the **contention period** to know if a collision has occurred.
Contention Period is between 2 farthest hosts/devices. Because it is **the worst case**, so either the collision will occur between the 2 farthest devices or never occur. Because the farthest host/device might cause collision even if the transmission is between 2 closest devices since it will take more time for the frame to propagate and reach the farthest device.

After sending a frame the device will wait till the Contention Period. If nothing is detected within this time then the frame has been transmitted successfully. Then send another frame and wait till the Contention Period and so on. If it detects any signal during the Contention Period then the device will know that a collision has occurred.

What will happen after the collision in CSMA/CD?
In CSMA/CD if this situation occurs then, an algorithm called "**Binary Exponential Backoff (BEB) Algorithm**" will tell how and who will send data after the collision. After the collision each of the collided frame devices will locally run **BEB** which will tell them which device will send the data after their collison.

Binary Exponential Backoff (BEB) Algorithm
Here, The device will have to multiply a bit time with a random number from a set which it will create depending on the number of times a collision has occurred. Formula to create the set is, Set = {0,....,2^n-1} where n = number of times collisions have occurred. Then that device has to wait that multiplied amount of time to retransmit its frames.

Example: Suppose a collision occurred between "A" and "C" then,
For Device 'A' collision has occurred 1 time. So,

Set = {0,...., 2^n-1}
= {0,...., 2^1-1}
= {0,...., 1}
= {0,1}

Now device 'A' will randomly pick a number from the set and multiply the bit time.
If picks 0 then, time to wait = (0 * 512 bit) = 0 bit time. So 'A' will not have to wait and can retransmit.
If picks 1 then, time to wait = (1 * 512 bit) = 512 bit time. So 'A' will have to wait 512 bit time and then can retransmit.

For Device 'C' collision has occurred 1 time. So,

Set = {0,...., 2^n-1}
= {0,...., 2^1-1}
= {0,...., 1}
= {0,1}

Now device 'C' will randomly pick a number from the set and multiply the bit time.
If picks 0 then, time to wait = (0 * 512 bit) = 0 bit time. So 'C' will not have to wait and can retransmit.
If picks 1 then, time to wait = (1 * 512 bit) = 512 bit time. So 'C' will have to wait 512 bit time and then can retransmit.

Now in this case the chance of both devices picking the same random number is 50%. If they pick the same number then again collision will occur. If so then the same process will happen again. So,

Again,
For Device ‘A’ collision has occurred 2 times. So,

Set = {0,...., 2^n-1}
= {0,...., 2^2-1}
= {0,...., 3}
= {0,1,2,3}

Now device ‘A’ will randomly pick a number from the set and multiply the bit time and multiply the bit time to have a wait time And same for device ‘C’.

Here, they can again pick the same random number and occur a collision but this time the chances are 25%.
So, each time the chance of picking the same number will decrease and chances of no collision will increase. Because each time the set numbers will increase. Worst case the collision will keep occurring 10 times.
If the collision reaches 10th time then it will know that the problem is somewhere else and not in this protocol or allocation.

[**Note:** If ‘A’ collided with ‘C’ in the 1st time and then collided with ‘D’ in the 2nd time the number collision will be 2 for ‘A’ but 1 for ‘D’]
[**Note:** If a collision between 3 devices occurs and each device's collision number is 1 then it is guaranteed that 2 devices will collide in the next transmission. There can also be a chance that all 3 device will collide]

Collision Free Protocol-

There are 2 collision free protocols.

A Bit Map Protocol- Before transmission there is a polling phase which will be divided for each individual device. Each of these are called **Contention slots** which have a bit time for each of the devices.
In the polling phase the devices will send 1or 0 in their respected contention slots,

1 = if interested to transmit
0 = if not interested to transmit (have to data)

These 1 and 0 will also be transmitted to every other device where they will keep a note of these. After every device has sent their consent to the contention slot and the polling phase is over, each device will have generated a bitmap with the contention slot data they received.

Now the transmission phase starts and they maintain the polling phase serial of the contention slots to transmit data.

Example:

Contention slots

A	B	C	D

1	0	1	0

Here, each device will follow their bitmap of the contention slot and send data. A will transmit 1st then C will transmit. B and D will not transmit. After they are done transmitting, again another polling phase will start. This time the **Contention slots might be** different**.**

A	B	C	D
0	1	1	0

Here, A will not send, then B will send, then C will send and then D will not send.

How will the devices know when to start transmission?
In the transmission phase after each transmission of a device frame, there is a **gap** in between the transmissions where there is no signal in the medium. So the devices will sense and count the gaps till they can start transmitting. Since the devices will know the bitmap and will know how many devices will transmit before its own turn it will count the gaps after each transmission and wait for its turn to transmit.

Example:
In the above contention slot bitmap, ‘C’ knows that there is only 1 device which will transmit before ‘C’ and that is ‘B’. So ‘C’ has to count 1 gap then it will know that it's now ‘C’s’ turn to transmit.

This is how Bitmap Protocol Works.

Binary countdown- Here every device will get a unique number of the same bit length address. [5 bit length address for 30 hosts = 2^5 = 32 unique addresses.]

Example: Here, out of 30 devices A,B,C,D are interested to transmit data

	Bit time				
	0	**1**	**2**	**3**	**4**
A(10110)	1	0	1	1	0
B(10111)	1	0	1	1	1
C(10111)	1	0	1	0	x
D(10011)	1	0	0	x	x
OR	1	0	1	1	1

If a device bit does not match with the OR value bit in serial, then it will get kicked out from the chance to transmit. In the above we can see that device D's bit in bit time 2 did not match with the OR bit value and got kicked out from the next bit time. C also got kicked out in bit time 3.

The one device that does not get kicked out will have its unique ID be similar to the OR Id and that device will earn the right to transmit.
Here, The OR = 10111 which is the Unique Id of device 'B'. So 'B' will have the right to transmit.

Lower numbers have a chance to enter starvation.(00000 is the lowest)

Binary countdown can be implemented when there is a priority. The higher the ID number the Higher the priority. For the above the highest priority is Id-11111 and the lowest is id-00000

This protocol is very time consuming

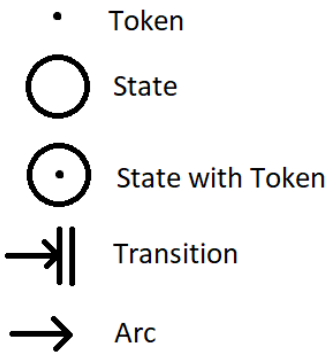
Class-6

Protocol verification -

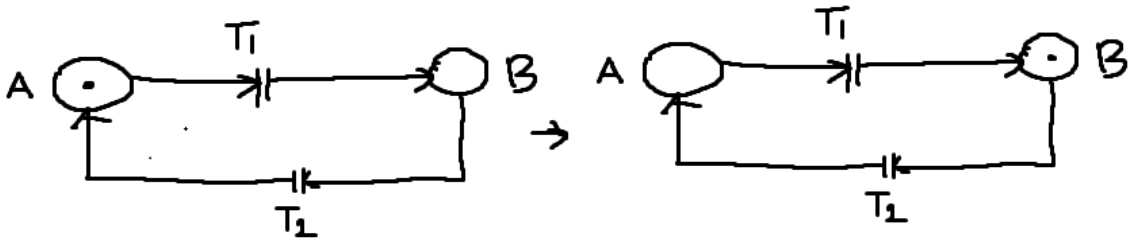
Theoretically checking if the protocol works then implementing it.

Finite state Machine- Sir, did not teach this. Just told the name.

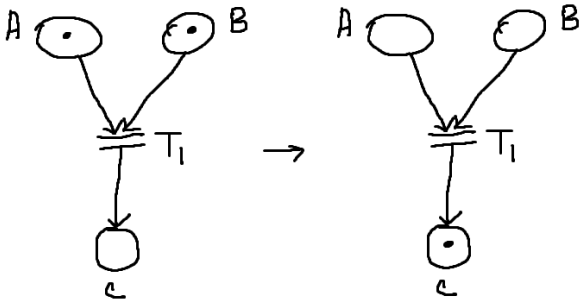
Petri Net model-



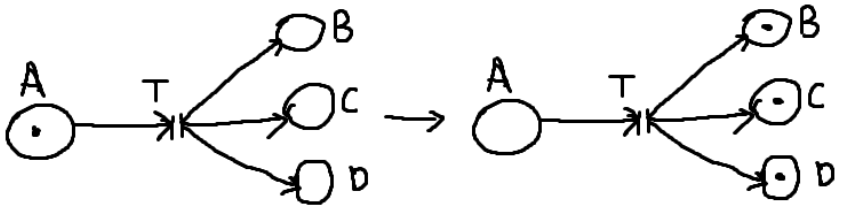
1. **State-** It is the goal of a process and it tells which state the process is in.
2. **Token-** With the token transitions can be executed. The number of tokens needed will depend on the transition. It is like a key.
3. **Transition-** Transitions are executed to move to the next state.
4. **Arc-** It shows the direction.



To go from state A to B, it has to go through by executing the transition T1. For A to execute the transition there has to be a Token inside A. After the transition the token will be transferred to B. So for A to process continuously the token has to come back to A. So using another transition T2 the state B can go to A and then the Token will go back to A.

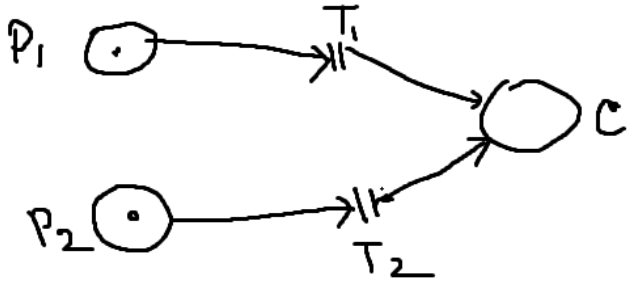


Now for A and B to go to C both have to execute the transition model. Tokens from both A and B have to come to execute the transition since In the transition 2 arcs are coming, 1 from A and 1 from B. And after the execution only one Token will go to C since there is only 1 arc coming to C from the transition.

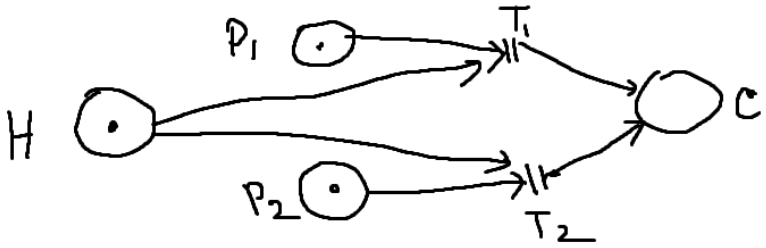


Here, multiple arcs are coming from the transition so the Token from A will have copies go to all the connected States from the transition.

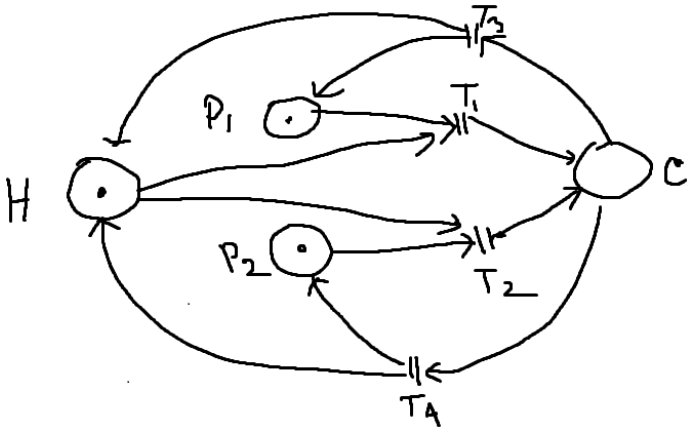
Mutual Exclusion- Mutual excursion is when multiple processes need to use one state. But the state can be used by one process at a time. This state is called a critical state/resource.



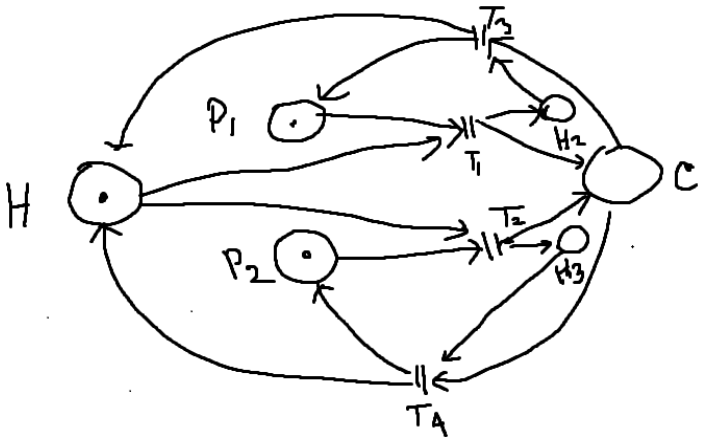
Here, P1, P2 are 2 processes and C is a critical state/resource. Both P1 and P2 satisfy the transition condition of having 1 Token to transit to C so they will both move to C state but C can only let a single process in at a time. So here malfunction will occur.



Here, to stop the malfunction we can use a helper state. Now we can see that to transit T1 and T2 needs 2 Tokens. When P1 need to use C it will get a helper Token from H to Transit T1 and go to C but at the same time if P2 wants to transit, It won't be able to transit T2 because T2 also need 2 Tokens and H already gave its token for T1s transition. So there the mutual Exclusion is intact.



Now C can also return the tokens it received for a process to start from the said process and H through T3 and T4. But it still has a problem when returning the token.
When returning a Token to P1(suppose) C will send Tokens to both P1 & H through T3 and also to P2 and H through T4 since 2 arcs are going out of C to the T3,T4 transitions. At that time P1 will receive its 1 token but H will receive 2 Tokens and P2 will also have 2 Tokens because it will receive another Token including its original Token which was already in P2. This will cause another malfunction.



Here, To combat the malfunction we will use another H2, H3 helper states. H2, H3 will receive a Token from T1 and T2 when whichever transition is being used. Then when C returns the Token the Transition T3 and T4 will need 2 Tokens to transit. So suppose when P1 wants to use C it will use the T1 transit and use the help of H and transit from T1 to C. When transiting from T1, 1 Token will go to H2 and one will go to C. Now when C returns the Token back to P1 it will return 1 Token to both T3 and T4 but now these Transitions require 2 Tokens. So for the case of T3 it will receive 1 Token from C and 1 from H2 and then from T3 1 Token will go to P1 and H. But T4 will only receive 1 Token from C but no Token from H3 because H3 did not receive any Tokens. So T4 transition will not occur.

Now we have the perfect and final model of **Mutual Exclusion** for the **Petri Net Model**.

MID-2

Class-1

Internet Protocol (IP)-

Ip address is a unique number which helps to find a device in a network. It is put in the 20 byte header of a packet in TCP/IP models Internet Layer (Network Layer) when creating the packet. IP has different versions.

1. IPv4(32 bits)
2. IPv6(128 bits)

IPv4 IP address(32 Bits long): Currently IPv4 is the one that is used. There have been a lot of quick fixes that help improve IPv4 that we are not changing to IPv6.

Example:

Bit string: 11000011 10101110 00010111 00001111
IP address: 192.168.19.15

IPv6 IP address(128 Bits long): It is still not used because it takes time to convert from IPV4 and will disrupt the network flow.

20 byte IPv4 header-

The structure of the **20 byte header** is given below. Each segment is a consecutive bit.

<table>
<tr><td>Version
(4 bit)</td><td>IHL
(4 bit)</td><td>Type of
Service
(6 bits)</td><td>Empty
bit</td><td colspan="4">Total length
(16 bit)</td><td rowspan="6">20
Byte
Hea
der</td></tr>
<tr><td colspan="4">Identification
(16 bits)</td><td>Empty
bit</td><td>D
F</td><td>M
F</td><td colspan="2">Fragment Offset
(13 bit)</td></tr>
<tr><td>TTL
(8 bits)</td><td colspan="4">Protocol
(8 bit)</td><td colspan="3">Header Checksum
(16 bit)</td></tr>
<tr><td colspan="8">Source IP
(32 bit)</td></tr>
<tr><td colspan="8">Destination IP
(32 bit)</td></tr>
<tr><td colspan="8">Options</td></tr>
</table>

Version(4 bit): It is for the devices and layers who need to read the Packet header. And when reading the header the 1st thing that is read is the version. Because depending on the Version the information in the Ip will be different.
Example: Network Layer and router always reads the header of a packet.

IHL/Internet Header Length(4 bit): It tells the length of the packet header. Even though the header is of 20 bytes, there is an Option section where we can add extra options which will increase the Header length above 20 bytes.
Example: If there are 2 byte options added then the length of the Header will be 22 bytes.

Type of Service (6 bit): 1st 3 bit is Priority and last are 3 individual bits of D T R.

Precedence (Priority) (3 bit)	D (1 bit)	T (1 bit)	R (1 bit)

Precedence gives the priority of the packet.
Example: If the precedence is 000 that means the priority is low and the packet is carrying normal data. If the precedence is 111 then the priority is high and is carrying controlling data like the data generated by ping, echo, hello etc.

D means **Delay.** If this bit is 1 that means that there will be a delay in the communication.
Example: Satellite communication.

T means **Throughput.** If this bit is 1 that means that there will be a lot of data sent through the intermediate devices.

R means **Reliability.** If this bit is 1 that means that the intermediate devices will send the packet through a reliable path. It might take time but still it will be reliable.
Example: If a source sends sensitive data then it should be sent on a reliable path.

Empty Bit(1 or 2 bits): A string of few empty bits

Total Length(16 bits): It is the total length of the packet. With this we can tell where the packet ends and where the frame trailer will start.

Identification(16 bits): It is an ID number used to identify the packet. After receiving the packet we need to acknowledge it and for that we need Identification.

DF/Don't Fragment(1 bit): If the bit is 1 that means that the intermediate devices can not fragment the packet. If one path needs to fragment the packet just to transfer it, then it will pick a different path.
Example: For sensitive data there might be a malfunction when reassembling the data from fragments. For these cases do not fragment the packet.

MF/More Fragment(1 bit): If the bit is 1 that means that the intermediate devices can fragment the packet.

Fragment offset(13 bit): It gives the position of the fragments in the packet.
Example: B fragment is after A and before C fragment.

TTL/Time to Leave(8 bits): How long the packet will last. It has no unit. It just counts the hop of devices it will jump. When the TTL is 0 then it will no longer hop and will get deleted..

Protocol(8 bit): It tells what protocol will be used in the upper layer. When the packet will get de-encapsulated in the network layer and transferred to the transport layer, It will tell which protocol the TPDU will be given to. So, the protocol which will handle the TPDU in the upper layer, its information is in the **Protocol**.
Example: If the protocol is TCP then the TPDU inside the packet will be given to the TCP protocol to be handled.

Header Checksum(16 bit): It has the CRC or Cyclic redundancy check. It will find if there are any errors in the Header.

Source IP(32 bit): It is the sources IP address

Example:

Bit string: 11000011 10101110 00010111 00001111
IP address: 192.168.19.15

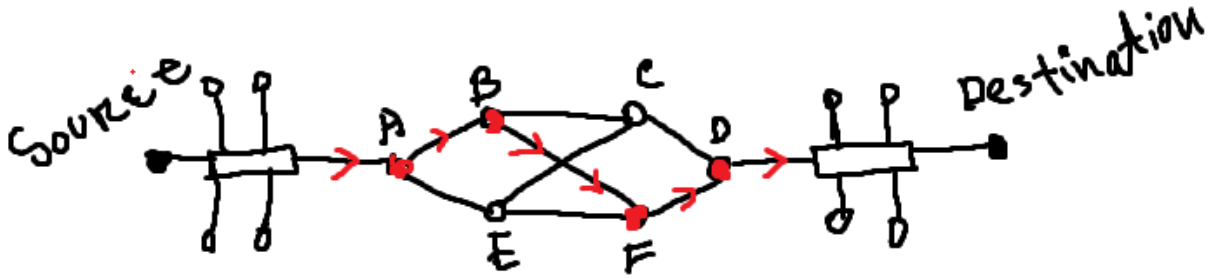
Destination IP(32 bit): It is the destinationsIP address

Example:

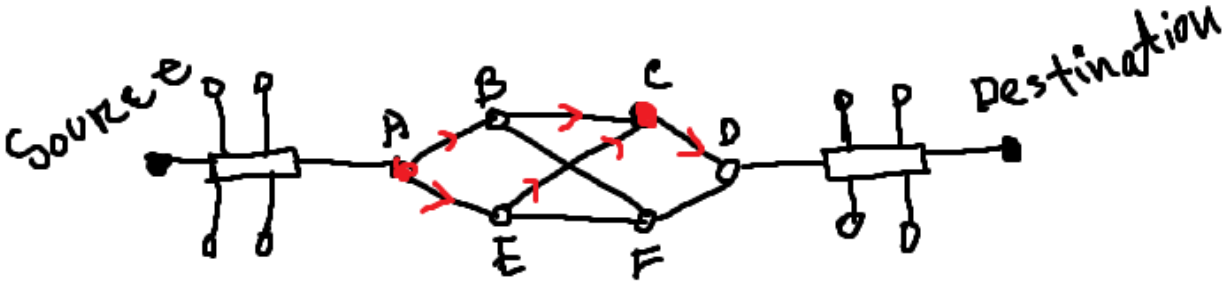
Bit string: 11000011 10101110 00010111 00001111
IP address: 192.168.10.15

Options: It is the optional part which is not required. There are many options,

1. **Security:** How secret the packet/communication is. How much secrecy should the data in the packet maintain.
2. **Strict Source Routing:** It tells the path the Packet must follow to go to its destination. Example: Go through the routers path (A->B->F->D)



3. **Loose Source Routing:** It gives a list of routers which can not be missed when going to the destination. Doesn't matter what path it choses but has to go through these routers. It can be used to monitor what data goes through this router. Example: Go through the routers list(A,C).



4. **Record Route:** The packet records the Ip address of the routers it goes through. The Ip address gets inserted in the header. Then after reaching the destination these Ip address can be seen to see which routers the packet went through. These can be used to backtrack to find the source of the packet.
5. **Record Time:** The packet records the time when it goes through the routers. Then after reaching the destination these Time can be seen to see which routers the packet went through and what time. These can be used to backtrack to find the source of the packet.

IP Address (IPv4)-

The IP address of IPv4 is 32 bit long. The combination of the address can be 2^32 = 4294967296.

IP address in Decimal notation: 192.168.10.15
IP address in Binary bits: 11000000 10101000 00001010 00001111

In the packet header there will be the Destination IP address. When the packet travels through the intermediate transport devices, the devices will read the destination Ip and decide a path for the packet to go to its destination.

The destination is part of a network system. So to find the destination device the packet 1st needs to find in which network system the device is in. After finding the network then the packet will search which host device is the destination. So the Ip address will indicate the destination device and which network is the device in.

Different classes of IP address: (Lecture 7 time: 1:07:00) -

To denote a network the IP has been divided into 4 classes.

Class A: Here, The Network is 8 bits meaning 2^8 number of networks in the whole world and The host devices are 24 bits meaning 2^24 numbers of hosts in the whole world under each network.
Example: here blue is network and orange is host IP

Binary Bits: 01111111 11111111 11111111 11111111
Decimal: 127.255.255.255

Now how will we know which IP is of Class A?
For Class A the 1st bit of the network is fixed and that is 0. So the range of the network address for 1st octet is from 00000000 to 01111111 = 0 to 127 (In Decimal)
And for the Host IP all is flexible. So the range is from 00000000 to 11111111 = 0 to 255(In Decimal)
So the **total range** of IP for Class A is,
From,

Binary Bits: **00000000** 00000000 00000000 00000000
Decimal: 0.0.0.0

To,

Binary Bits: 01111111 11111111 11111111 11111111
Decimal: 127.255.255.255

[Though 00000000 00000000 00000000 00000000 is a special IP so the IP will start form 00000001**]**

Class B: Here, The Network is 16 bits meaning 2^16 number of networks in the whole world and The host devices are 16 bits meaning 2^16 numbers of hosts in the whole world under each network.
Example: here blue is network and orange is host IP

Binary Bits: 10111111 11111111 11111111 11111111
Decimal: 191.255.255.255

Now how will we know which IP is of Class B?
For Class B the 1st 2 bits of the network is fixed and that is 10. So the range of the network address for 1st octet is from 10000000 to 10111111 = 128 to 191 (In Decimal)
And for the Host IP all is flexible. So the range is from 00000000 to 11111111 = 0 to 255(In Decimal)
So the **total range** of IP for Class B is,
From,

Binary Bits: 10000000 00000000 00000000 00000000
Decimal: 128.0.0.0

To,

Binary Bits: 10111111 11111111 11111111 11111111
Decimal: 191.255.255.255

Class C: Here, The Network is 24 bits meaning 2^24 number of networks in the whole world and The host devices are 8 bits meaning 2^8 numbers of hosts in the whole world under each network.
Example: here blue is network and orange is host IP

Binary Bits: 11011111 11111111 11111111 11111111
Decimal: 223.255.255.255

Now how will we know which IP is of Class B?
For Class C the 1st 3 bits of the network is fixed and that is 110. So the range of the network address for 1st octet is from 11000000 to 11011111 = 192 to 223 (In Decimal)
And for the Host IP all is flexible. So the range is from 00000000 to 11111111 = 0 to 255(In Decimal)
So the **total range** of IP for Class B is,
From,

Binary Bits: 11000000 00000000 00000000 00000000
Decimal: 192.0.0.0

To,

Binary Bits: 11011111 11111111 11111111 11111111
Decimal: 223.255.255.255

Class D(Out of class scope): For Multicast range(224.0.0.0 - 239.255.255.255).
Class E(Out of class scope): Future use range(240.0.0.0 - 255.255.255.255)

Class-2

Finding the Destination-

Given the destination IP address 192.168.10.15, 1st we need to find the Network. Here we can see that the IP is of Class C so the 1st 3 octet is part of the network and the last octet is part of the host. So we found the network by its class. Now we need to follow a convention to find the host. (**This Convention is for all Classes of IP address**)

Network address:
The convention to get Network address is to keep the Network as it is and turn the host Octet to 0 (binary- 00000000). So we get (for class C),

Network address:192.168.10.0

Broadcast IP address:
If the data/packet has to be broadcasted/sent to all the hosts in a Network, then the broadcast address needs to be used.
The Convention of the Broadcast Ip address is that The network address stays the same but the host octet has to be 255 (Binary- 11111111). So (for class C),

Broadcast address: 192.168.10.255

Host address:
Convention to get the host address is, anything of the Host octet between 0 to 255 is the host address. Meaning the host address in between the Network address 163.146.0.0 and the Broadcast address 163.146.255.255. So, 2^16 - 2 = 65,536 - 2 = 65,534 is the host range/count(for Class B since in Class B the Host is 8+8=16 bits). So (for class C),

Possible 1st host: 192.168.10.1
Possible 2nd host: 192.168.10.2
…….
Among these host one of them is: 192.168.10.15
…….
Possible 254th host: 192.168.10.254

Subnet Mask (Later will learn in depth):
Convention to get Subnet Mask is for the network Octet to be 255 (binary-11111111) and for the host Octet to be 0 (binary-00000000).
So if the Address is 192.168.10.16 a Class C then the subnet mask is,

Subnet Mask: 255.255.255.0

And if the Address is 163.46.10.24 a Class B then the subnet mask is,

Subnet Mask: 255.255.0.0

Example 1: Given 163.46.10.24
Here, The IP class is B. Now following the convention,

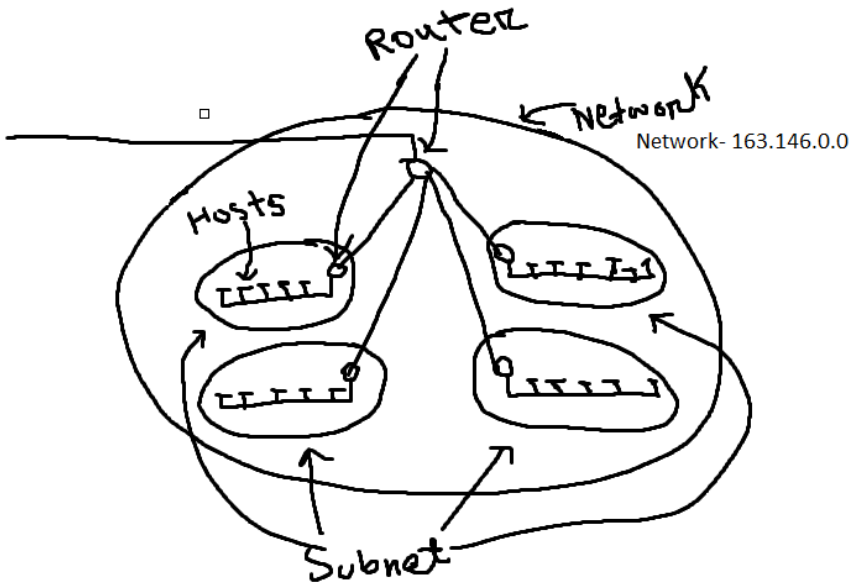
Network Address: 163.46.0.0
Possible 1st host: 163.46.0.1
Possible 2nd host: 163.46.0.2
……...
Among these host one of them is: 163.46.10.24

Broadcast address: 163.46.255.255
Subnet Mask: 255.255.0.0

Example 2: Given 24.46.13.56
Here, The IP class is A. Now following the convention,
Network Address: 24.0.0.0
Possible 1st host: 24.0.0.1
Possible 2nd host: 24.0.0.2
……..
Among these host one of them is: 24.46.13.56
Broadcast address: 24.255.255.255
Subnet Mask: 255.0.0.0

Subnet (Sub network):(Lecture-8 time- 50:00 watch if you don't understand)-

In a network there are alot of hosts. So for better management and to identify the host faster, a Subnet can be used. A subnet is a smaller network under the original network. So a network can have many subnets under it to organise all the hosts.



Here, the network is 163.146.0.0. The router is connected to a bunch of smaller networks with routers which are then connected to hosts. These smaller networks are subnet.
So when a packet comes to the main router in the network, the router has a table of subnet addresses which tells the main router which subnet has the destination IP. Then it passes the packet to that subnets router and that subnet router finds the destination host under the subnet hosts and sends the packet to the destination host. That is how a subnet is used to make the search for the host faster.

The convention of the subnet is, We can not change the network octets address But we can change the host octet.
So, we have to make any number of bits from the host octet reserved for the subnet address. But it has to be 1<N<total host bit. The more bits we take for the subnet the more number of subtes will be created and there will be less number hosts per subnet.

In this case we take 4 bits from the host. So the total number of subnets is 2^4 = 16(Not really) and under each subnet there are a total of 2^16-4 = 4,096 hosts. Network address is 163.146.0.0 and it is in Class B so we take 4 bits from the 1st octet of the host address. So,
Network Address,
Decimal: 163.146.0.0
Binary: 10100011 10010010 00000000 00000000

Now, taking the 1st 4 bits of the host the subnet will start from
Binary: 10100011 10010010 00010000 00000000 [decimal of 0001 = 1. 1st subnet]
Decimal: 163.146.16.0
2nd subnet will be,
Binary: 10100011 10010010 00100000 00000000 [decimal of 0010 = 2. 2nd subnet]
Decimal: 163.146.32.0
3rd subnet
10100011 10010010 00110000 00000000 [decimal of 0011 = 3. 3rd subnet]
163.146.48.0
……
Last Subnet will be,
Binary: 10100011 10010010 11100000 00000000
Decimal: 163.146.224.0

The Subnet will be between the Network IP 163.146.0.0 and The Broadcast IP 163.146.255.255. **So there will be a total of** 2^4 - 2 = 16 - 2 = 14 Subnets (For Class B).

So the **hosts** under the 1st subnet will be between the 1st Subnet IP 163.146.16.0 and 1st Subnet Broadcast IP 163.146.31.255. And the host of the 2nd subnet will start after 2nd Subnet IP 163.146.32.0. **So for class B the hosts under each subnet is** 2^12 - 2 = 4,096 - 2 = 4,094.

Example: If the IP address is 163.146.42.16 then,
Network Address: 163.146.0.0
1st possible Subnet: 10100011 10010010 00010000 00000000
163.146.16.0
1st host in 1st Subnet: 10100011 10010010 00010000 00000001
163.146.16.1
2nd host in 1st Subnet: 163.146.16.2
…….
Last host of 1st Subnet: 163.146.31.254
Broadcast IP of 1st Subnet: 10100011 10010010 00011111 11111111

163.146.31.255

2nd possible Subnet: 163.146.32.0

3nd possible Subnet: 10100011 10010010 00110000 00000000
163.146.48.0

1st host in 3rd Subnet: 10100011 10010010 00110000 00000001
163.146.48.1

2nd host in 3rd Subnet: 163.146.48.2

…….

Last host of 3rd Subnet: 163.146.63.254

Broadcast IP of 3rd Subnet: 10100011 10010010 00111111 11111111
163.146.63.255

…….

Last possible Subnet: 10100011 10010010 11100000 00000000
163.146.224.0

Broadcast IP of last Subnet: 10100011 10010010 11101111 11111111
163.146.239.255

Possible host under a subnet is : 163.146.42.16 [host is under the 2nd subnet]

Broadcast address: 163.146.255.255

Subnet Mask: 255.255.0.0

[** Subnet Mask Is the separation between the network address, the subnet address and the host address in the IP Address.]

Class-3

Subnet Mask-

Subnet Make is very different from Subnet. Subnet Mask Is the separation between the network address, the subnet address and the host address in the IP Address. And if there is no Subnet then, it's the separation between just the Network and the Host.

Subnet Mask IP if there is No Subnet:

The convention is for the network Octet to be 255 (binary-11111111) and for the host Octet to be 0 (binary-00000000).

Example:

Suppose the IP address is 130.146.190.25

Subnet MaksIP: 11111111 11111111 00000000 00000000
255.255.0.0

Subnet Mask IP if there is Subnet:

The convention is for the network Octet to be 255 (binary-11111111), all the Subnet bits to be 1 (If the subnet is 4 bits then binary-1111) and for the host Octet to be 0 (binary-00000000).

Example:

Network IP: 10000010 10010010 00000000 00000000
130.146.0.0
Subnet MaksIP: 11111111 11111111 11110000 00000000
255.255.240.0

Classless Inter Domain Routing Notation(CIDR):

The convention is,

The network IP/ The bit number where it separated

Here, the Network IP is 130.146.0.0 and The separation point is The total bits of the Network IP + the Subnet bits (8 + 8) + 4 = 20. So,

CIDR: 130.146.0.0/20

Another convention is,

The network IP. Host IP/ The bit number where it separated

Suppose the host is 130.146.190.25. So,

CIDR: 130.146.190.25/20

How does a router sends a Packet to the Destination When There is Subnet-

The router under the Network has a table which has a list of Subnet Mask IP addresses. When a Packet comes to the router, The router gets the destination IP address from the header of the packet and then in Does an AND operation with the Subnet Mask IP address and then I find the IP address of the Subnet and send the Packet to the Subnet router. That subnet router then sends the packet to the destination since that router has the destination address of all its hosts under that subnet.

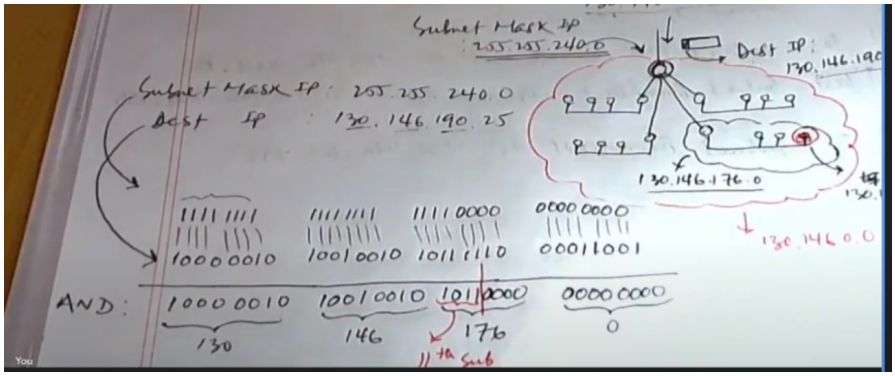
Example 1:

Suppose a destination IP: 130.146.190.25
Subnet Mask IP: 255.255.240.0

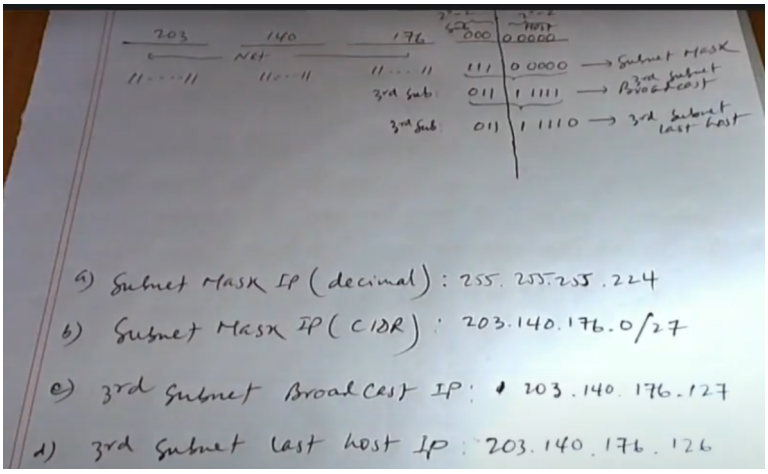
Now we do AND operation on binary,

Subnet Mask IP: 11111111 11111111 11110000 00000000
Destination IP: 10000010 10010010 10111110 00011001
AND Operation: 10000010 10010010 10110000 00000000
AND Operation Decimal: 130.146.176.0

Now 130.146.176.0 is the Subnet IP address. So the Host is under this Subnet. Now the Main router will pass the Packet to this Subnets router and then the router will send the Packet to the Destination.



Example 2:



Special IP-

These IPs are not used for Host and are used for special purposes.

1. 0.0.0.0 is a special Ip used for Booting purposes of the OS. It is called “This Host”. The OS can not see the IP in the Network Interface Card before booting. So when requiring an IP the OS uses This host IP.
2. 255.255.255.255 is a special Ip used for Broadcasting on the local Network.
3. (Works for any class but as an example using Class A) The network Ip is 0 but not the host ip. So for class A, 0.120.146.18 it means a specific Host on that local network meaning a host in class A.
4. When an Ip starts with 127 then it is a “Loopback” IP. It is used when a packet will go to the destination and then return back. (Example: 127.10.0.40) These IPs are used on Linux.

Class-4

Quick Fixes for IPv4-

Before transitioning from IPv4 to IPv6 there have been some quick fixes for the IPv4 which is now being used and for it being really popular the transition to IPv6 has been at halt.

Type of IP:

Since most of the clients take services from servers, the clients do not need to have a unique IP address. So depending on this there are 2 types of IP.

Private IP: This is a range of local IP that is private and has no uniqueness. The outside world can not find this IP. This IP is used for clients and servers under a local Network. These IPs can not leave that specific network. Since these private IPs can not leave a network, other networks can also have these same IPs. So a private IP can be accessed from that specific Network but not from the outside network. Private IP uses a range of IP from all the Classes. They are,

Class A - The range is from 10.0.0.0 - 10.255.255.255. (more than 16x10^6 addresses)
Class B - The range is from 172.16.0.0 - 172.32.255.255. (more than 1x10^6 addresses)
Class C - The range is from 192.168.0.0 - 192.168.255.255. (more than 65x10^3 addresses)

Example: Clients/Servers on a local Network.

Real IP: When we need a server/client from a specific Network to connect to other servers/clients in the outside world or from the outside Network, then we need a unique IP. That is a Real IP. All the remaining IPs from the Classes after the private IP are Real IPs.
Example: CNN webserver

How can we access outside networks using Private IP? -

There is a concept called NAT(Network Access Transmission) which helps private IP to access other Networks. This concept came when the scarcity of IPv4 started.

NAT(Network Access Transmission):
NAT is incorporated in the Gateway of the network. When a request/data goes through NAT, it maps the Real IP of the ISP to the Private IP of the request/data. Then the request/data gets the privilege of a Real IP and can connect to other outside networks.

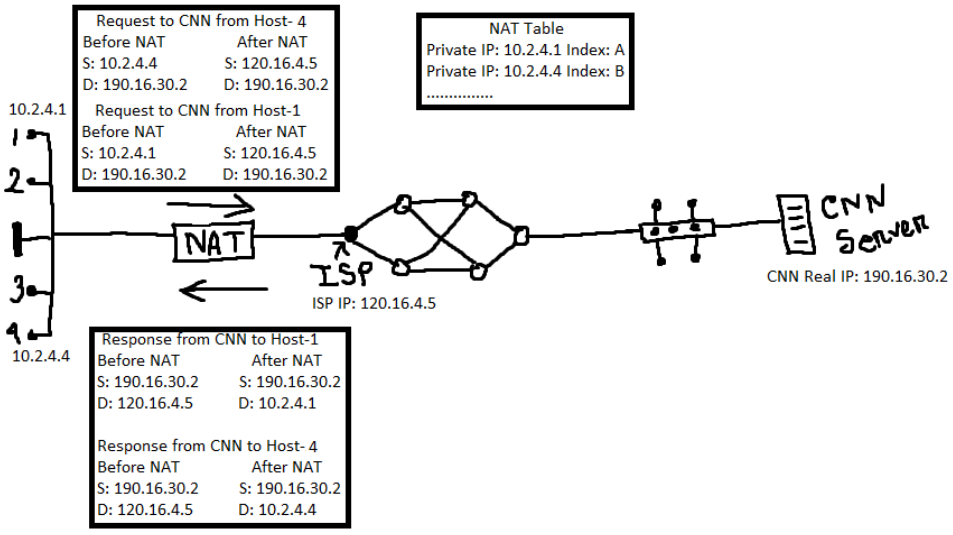
NAT also manages a table of Indexes for each request passing through it from the Private IPs. Each Index gets a corresponding Private IP assigned to it to keep track of which Private IP the NAT has to send the Response data when it comes from the outside network.

Example:
Here, Host 1 and 4 are Private IPs and they want to request to CNN server. In the Packet the source is the Private IP and the destination is CNN Real IP. So their request 1st goes through NAT. Here NAT replaces their Source IP from Private to Real IP of the ISP in their Packet

Headers. Now these packets have the privilege of Real IP and can go outside of the current network to the outside network.

NAT also puts the Private IP to a corresponding Index in their NAT table to keep track of the devices in that network. And also insert the corresponding Index in the packet header.

The intermediate routers route the packet to the destination which is CNN. Then CNN replies to the response and now the reply packet has the CNN Real IP as source and the ISP real IP as destination with the NAT index. Then from the ISP the response will come to the NAT. Now NAT will compare the Index of the response with its NAT table index and again replace the destination ISPs Real IP with the corresponding Private IP from the matched Index. Then NAT will send it to the destination hosts with that response from CNN.



Where do NAT inserts the Index?-

So to know this we 1st need to know about ports.

Ports:

It is a specific memory location where different services run. Port address is 16 bits. So, every machine has 2^16 = 65536 ports.
With the IP address we go to the machine and then with the 16 bit port address we can go to the specific location of the memory to get that required service.

Reserved port: There are reserved ports which are from 0 - 1023. Few of these are,

1. http = 80
2. Telnet = 23
3. FTP = 21
4. SMTP = 25
5. Pop3 = 110

Non-Reserved port: The rest 64513 are non-reserved ports.

Example:
Here, http://190.16.30.2/filename is a link. The 190.16.30.2 is the IP address and http even though it's a protocol but there is also a port here. This http runs on port 80.
So, the request will go to the machine with the IP address and then using http it will go to the port (80) in the machine and then get the reply for the Request from CNN.

Where is the port address stored?:

It is stored in the header of the TPDU. The TPDU has a header of 20 bytes. The header holds 16 bits size of both Source and Destination port.

Source Port = 16 bits	Destination Port = 16 bits
Some other data…..	

So when the packet is encapsulated the port number of http is stored in the header of TPDU.

Back to the question of “Where does NAT insert the Index?”:

Now that we understand ports, we can see that the **reserved** ports are from 0-1024 which only requires a bit size of 10 (2^10 = 1024). But since the port address is 16 bits, there are 6 bits which are empty.
So, the Index is stored in the 6 empty bits.

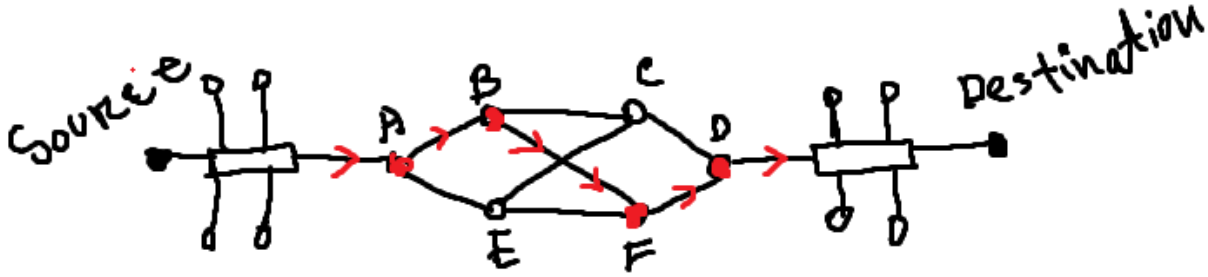
Class-5

Types of Network:

There are different types of networks. And depending on these networks different routing algorithms are used.

1. **Virtual Circuit Network (Subnet) -** There will be a dedicated path when sending a packet from source to destination. So for that session all the packets will have to follow that specified path using a routing algorithm. So initially a routing algorithm will be used in that session to decide a path. Packets from source will have a VC value in the header and the router will check these values and send the packets to the dedicated path to the destination. Easy to provide quality of service since the path is fixed and these specified

routers will provide the services.



2. **Datagram Network (Subnet) -** There is no dedicated path. Each packet can take whatever path to go from source to destination. So every packet has to be routed using an algorithm individually. Here the packet will have a lot of information in the header since each time the packets will go through different paths to go to the destination. Hard to provide quality of service since each time the packet will take a different path so the services have to be provided by all the routers.

Routing Algorithm-

An algorithm used for the purpose of routing is called a routing algorithm. There are 2 types of routing algorithms.

1. Static Routing Algorithm
2. Dynamic Routing Algorithm

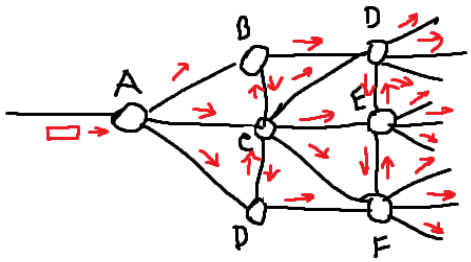
Static Routing Algorithm-

It is a pre-defined algorithm. It has no scope of adaptability so it is a non-adaptive algorithm. There are many different types of Static Routing Algorithms.

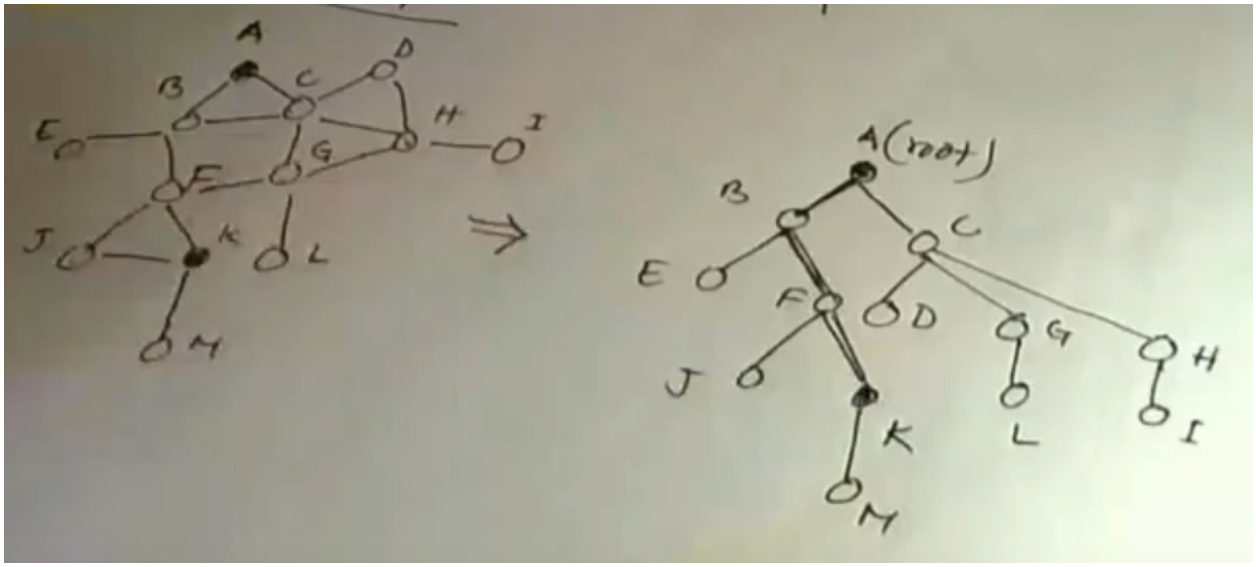
1. Shortest path routing algorithm
2. Flooding
3. Optimality Principle
4.

Shortest Path Routing Algorithm: The shortest path algorithms aim to find the optimal paths between the network nodes so that routing cost is minimized.

Flooding: Here if a router has a packet, it will send a copy of it to all the outgoing lines from that router to other adjacent connected routers and those routers will do the same till the packet has spread everywhere. At one point since the packet has spread everywhere it will surely reach its destination.
Flooding does not need a good mathematical algorithm and is very simple since it floods the connected routers with the packet till the packet reaches its destination.
Flooding is not good for wired network connection, since it creates traffic in the wired medium. But it is good for Wireless network transmission. Since wireless devices are not always stationary, it is hard to detect the device's position. So A wireless router will flood the packet in its transmission range.



Optimality Principle: Here, when sending a data/packet from a source, 1st create a tree from that source to other routers. Then we will get an optimum path from source to the destination.



Here, the source is from A to K so, after creating a tree from source A we can see an optimal path from A to K.

Dynamic Routing Algorithm-

It is the opposite of the Static Routing Algorithm. It is an adaptive algorithm. It will adapt and take a different path if there is any problem in the current path. There are a lot of Dynamic Routing Algorithms. 2 of them are,

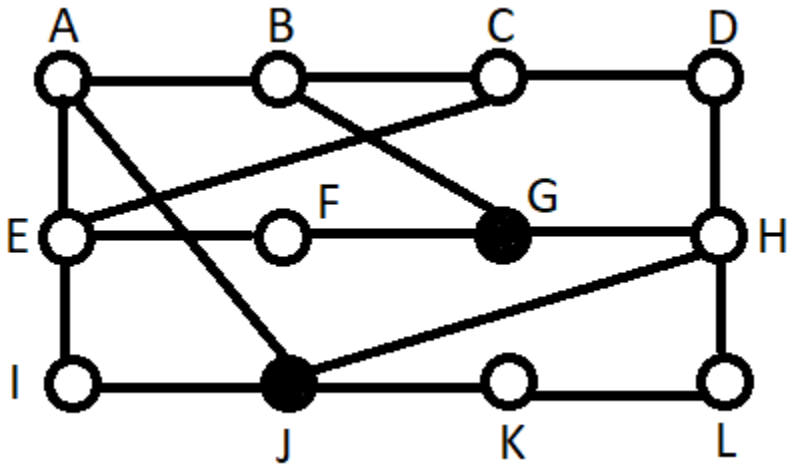
1. Distance Vector Routing Algorithm.
2. Link State Routing Algorithm.

Distance Vector Routing Algorithm: Here in the network, each router will have a routing table which will contain the time for a packet to reach all the routers from that specific router. And each router will also share their routing table with their adjacent routers. Then the source router will calculate which path will take the shortest time for the packet to reach the destination. Then will send the packet to that path.

How will routers know the time to reach other routers?

A router will send a ECHO packet to the specific adjacent router and will count how long it takes for that ECHO packet to return back. This will depend on the other router's packet queue, since that router might be busy processing other packets as well. Also the propagation distance will also affect the Time.

Example: Here, a packet will go from Source J to Destination G.



Now, The adjacent routers of J (A, I, H, K) will share their routing table with J and J will calculate the minimum time to go from these adjacent routers to the destination.

Table: Shared Routing Table

To (ms)	A	I	H	K	J (Source)
A	0	3	3	5	8
B	6	4	10	6	
C	7	5	3	11	
D	10	6	4	11	
E	11	11	5	6	
F	3	11	6	7	
G	10	11	9	3	?
H	3	10	0	3	16
I	4	0	6	6	20
J	5	7	7	4	0
K	6	10	10	0	12
L	11	11	3	2	

Now, The calculation is,
Time from Source to adjacent router + Time from Adjacent to destination.

J to G: JA + AG = 8 ms + 10 ms = 18 ms
J to G: JI + IG = 20 ms + 11 ms = 31 ms
J to G: JH + HG = 16 ms + 9 ms = 25 ms
J to G: JK + KG = 12 ms + 3 ms = 15 ms

Here, the shortest time is 15 ms which is in Adjacent router K. So source J will choose Adjacent router K.
So source J will send the packet to K. Then if K also follows the DIstance Vector routing algorithm, then K will also do the same calculation and decide which adjacent router will give the shortest path, then send that packet to that router.

[Note: Since it is a Dynamic routing algorithm, this calculation process and the time in the routing table will be updated in every session of packet transfer **]**
[Note: Also by the time J send the packet to K, the time form K to G might change from when J calculated the shortest path **]**

Table: J's Routing table after calculation

To (ms)	J (Source)
A	8
B	
C	
D	
E	
F	
G	15
H	16
I	20
J	0
K	12
L	

[Note: In this instance in J's routing table, 5 values have been updated **]**

Final

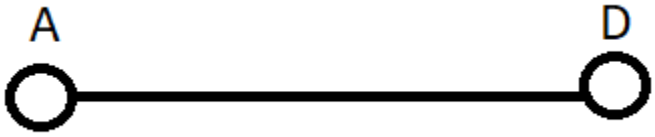
Class-1

Problem in Distance Vector Routing Algorithm-

The main problem in the Distance Vector Routing Algorithm is the **Counting Infinity problem**. This Counting Infinity problem may occur(**not 100%**) if the subnet is a **linear subnet**.

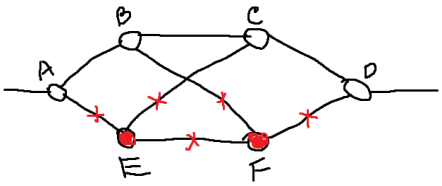
linear subnet:
A linear subnet is when routers are sequentially connected. Suppose routers A, B, C, D are connected sequentially then it is a linear subnet.

So, here if a packet is sent from A to D then the routers B and C have no work to do to send the packet. A router's job is to show a path/route from many different paths, but here we can see that there is only a single path from A to D .

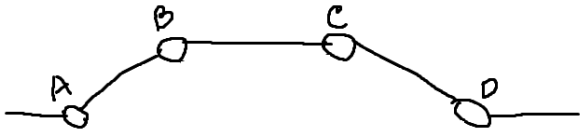


So both routers A and C do not have to use any routing algorithms to show the packet a path and are basically useless here. So it can be said that only A and D are connected

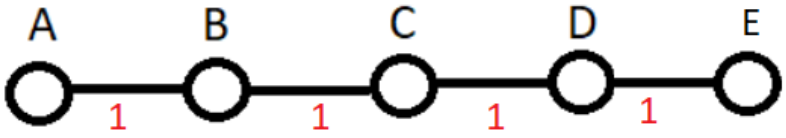
Now we can ask why design a linear subnet since it is wasteful and useless? The answer is that we actually do not design a linear subnet but it automatically gets created. How?



Suppose in the above network, router E and F are down. Then all the paths connected to E and F are also not accessible. And Router A, B, C, D will create a linear subnet and there might be a Count Infinity Problem.



Counting Infinity problem:



Suppose A,B,C,D,E routers are in a linear subnet and each routers are 1 millisecond apart from each other. Now suppose a router A is down. So router B can not reach A. Now the information value in the routing table of router B about A will be null. Other routers also can't reach A.

States	A	B	C	D	E
Initial	Down	null	null	null	null
1st Ex	Up	1	null	null	null
2nd Ex	Up	1	1+1 =2	null	null
3rd Ex	Up	1	1+1 =2	1+2 = 3	null
4th Ex	Up	1	1+1 =2	1+2 = 3	1+3 = 4

So, initially router A is down and in all the other routers routing tables the information value of a will be null since no router can reach router A.
Then when router A goes UP again it will send a **'Hello' (ICMP)** packet with various information to the other routers to let them know that it's back up again.

1st Exchange of information,
B will be the 1st to get the hello packet from A since B comes after A in the linear subnet. When B gets the hello packet it will send a ECHO packet to calculate the distance to reach A and that will be 1 millisecond (given) and insert it instead of null. At that time except B the other routers don't know that A is up since the hello packet has to go through router B.

2nd Exchange of information,
B will let its adjacent routers A and C know that it can reach A. Except A, C will receive this information and will know that C can also reach A through B. Now since C knows the distance to B is 1 millisecond and from B the distance to A is 1 millisecond, So C will update its routing tables A information from null to 1+1 = 2 millisecond. Since it will take 2 milliseconds to reach A from C through B.

3rd Exchange of information,
C will let its adjacent routers B and D know that it can reach A. B will see that information but will igort it since B can reach A in less time compared to C. But D will receive this information and will know that D can also reach A through C. Now since D knows the distance to C is 1 millisecond and from C the distance to A is 2 millisecond, So D will update its routing tables A information from null to 1+2 = 3 millisecond. Since it will take 3 milliseconds to reach A from D through C.

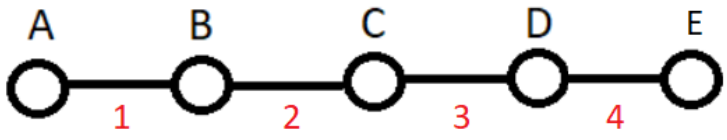
4th Exchange of information,
D will let its adjacent routers C and E know that it can reach A. C will see that information but will igort it since C can reach A in less time compared to D. But E will receive this information and will know that E can also reach A through D. Now since E knows the distance to D is 1 millisecond and from D the distance to A is 3 millisecond, So E will update its routing tables A information from null to 1+3 = 4 millisecond. Since it will take 4 milliseconds to reach A from E through D.

Now all the routers know the time to reach A. This is how the distance vector routing works since there is no counting infinity problem even though there is a linear subnet, Since it is not always guaranteed that counting infinity problem will 100% occur if there is a linear subnet.

So when will the counting infinity problem occur?

When a router comes back UP from being DOWN, that is a **good news situation**. So in this case the good news beautifully propagates through all the routers, and the counting infinity problem doesn't occur.

But when it is the opposite of good news, meaning the router goes DOWN from UP it is a **bad news situation**. This situation is given below,



Suppose A,B,C,D,E routers are in a linear subnet and each routers are 1,2,3,4 millisecond apart from each A to E.

States	A	B	C	D	E
Initial	Up	1	2+1 = 3	3+3 = 6	4+6 =10
1st Ex	Down	2+3 = 5	2+1 = 3	3+3 = 6	4+6 =10
2nd Ex	Down	2+3 = 5	2+5 = 7	3+3 = 6	4+6 =10
3rd Ex	Down	2+7 = 9	2+5 = 7	3+7 = 10	4+6 =10
4th Ex	Down	2+7 = 9	2+9 = 11	3+7 = 10	4+10 = 14
5th Ex	Down	2+11 = 13	2+9 = 11	3+11 = 14	4+10 = 14
Infinit	Down	…..	…..	…..	…..

Initially router A is up and the other routers have their routing tables A information to, B = 1, C = 2+1= 3, D=3+3= 6 and E=4+6 =10.

1st Exchange of information,
A is down. So B then gets the information that A is down and can not connect to A. B then checks other ways to get to A and finds that C can still reach A, so B thinks that B can connect to A through C. But B does not know that C is reaching A through B since the information that A is down did not reach C Yet and that in the distance vector routing it is not recorded how a router is getting its connection to another router and through which router is it dependent on for this connection. If B inserted a null value for A then this would not have happened since then C also would change its routing table information of A. But B looks for better options rather than putting null value and that is to look for other better possible ways to connect to A. Now B will update its routing table information of A so that it can go to A through C to B to C is 2 and C to A is 3 so, 2+3 = 5 will be inserted in B routing table in A information.

2nd Exchange of information,
Now B will send the new routing table information to adjacent routers A and C. C will get this information and will re-calculate the routing table for information of A. C will see that it will take C to b = 2 and B to A = 5 a total of 2+5 = 7 milliseconds to reach A through B and C to D = 3 and D to A = 6 a total of 3+6 = 9 milliseconds to reach A through D. So C will choose B and insert C = 7 in the routing table.

3rd Exchange of information,
Now C will send the new routing table information to adjacent routers B and D. B will again see that B to A is not possible directly, and choose C. B will get this information and will re-calculate the routing table for information from A through C again. Now since the C routing table information of A changed from 5 to 7, then B routing table will change, So from B to C =2 and C to A = 7 a total of 2+7 = 9 will be inserted in the routing table.

Now D will also get this information from C and re-calculate D's routing table. D will see that it takes less time to go to A through C 3+7=10 and more time through E 4+10 = 14. So D will insert 3+7=10 in its routing table.

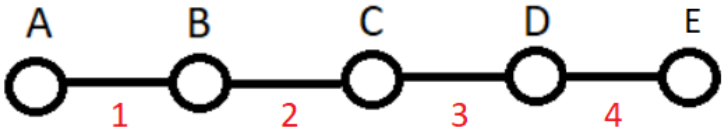
4th Exchange of information,
Now, both B and D will give their adjacent routers their updated routing table. So 1st C will get the updated table from B and D and will recalculate and see that connecting to A through B is more efficient then through D. So C will insert C to B = 2 and B to A = 9, so 2+9 = 11.
And for E this it can only use D to got to A, E will change its routing table and insert, E to D = 4 and D to A = 10, So
4+10 = 14.

5th Exchange of information,
Now C and E will give their information to adjacent routers. Then B will change to 2+11 = 13 and D will change to 3+11 = 14.

As we can, it is stuck in an infinite loop and will keep doing the process forever until there is a change in the environment, like A turning back UP. This is the count infinity problem.

[**Note:** Sometimes it's not the router that is down but the path to that router is down and there might be other paths to reach that router. So for that reason B did not put Null in its routing table and rather chose a different path to connect to router A]

Example: Hare, C is Down



States	A	B	C	D	E
Initial	1+2 = 3	2	Down	3	4+3 = 7
1st Ex	1+2 = 3	1+3 = 4	Down	4+7 = 11	4+3 = 7
2nd Ex	1+4 = 5	1+3 = 4	Down	4+7 = 11	4+11 =15
3rd Ex	1+4 = 5	1+4 = 6	Down	4+15 = 19	4+11 =15
4th Ex	1+6 = 7	1+4 = 6	Down	4+15 = 19	4+19 = 23
5th Ex	1+6 = 7	1+7 = 8	Down	4+23 = 27	4+19 = 23
Infinit	…..	…..	Down	…..	…..

Class-2

Link State Routing Algorithm- (Lecture Dec:28)

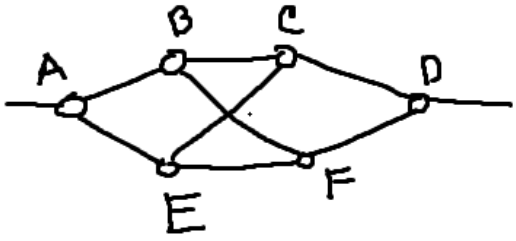
There are some steps for the Link state routing algorithm.

Steps:

1. Know your neighbor (HELLO packet- a Internet Controlling Message Protocol or ICMP packet)
2. Measure the delay (ECHO packet)
3. Build link state packet
4. Distributed link state packet
5. Compute the shortest path

How will routers work following the link state routing algorithm?

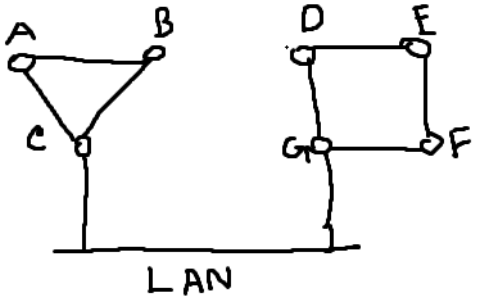
Suppose A,B,C,D,E,F is a subnet.



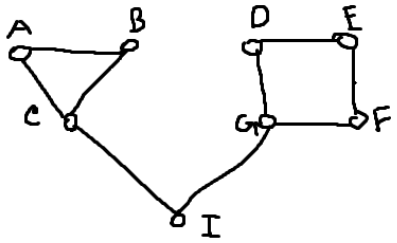
1st step is to know all the neighbors-

All the routers in a network will send a HELLO packet to its adjacent routers with various information like IP etc. By doing this the routers will know who their neighbors are.

There are some exceptions. Like the subnets below. Here, 2 networks are connected to each other with a high speed LAN.



Since LAN is not a specific device, we can not call LAN a neighbor of C and G. LAN also follows some other protocols. So for C and G to connect they have to follow the LANs protocol. So for this case we can image LAN as a node I.



Then we can say that C's neighbor is A,B,I and G's neighbor is D,F,I.

2nd Step is to Measure the delay-

Here, all the routers generate an ECHO packet and send it to the adjacent routers. Then the ECHO packet goes to the adjacent router and waits for its queue to be processed and that router processes it and sends it back. The router counts the time it takes to do all that and measures the propagation delay and the latency to the adjacent routers. That's how the delay is measured.

3rd Step is to Build Link state packet-

Here, all the routers will create a Build Link state packet and send it to its neighbors. This packet is generated every certain amount of time(Ex: every 10 second).The routing table will be created using this. In the packet there will be,

IP address of the router which created the packet
Sequence Number
Age
Information of its neighbors and their distance

Table: What a Build Link State Packet

IP address of the router: It will hold the IP address of the router that created the Build Link state packet

Sequence Number: Since the router will keep on sending the packet updated with new information of its neighbors and their distance, the neighbors that will receive it will have to delete the previous packet they got from the router. But sometimes this packt will take a longer path to reach the neighbors (downed line/connection) and by that time a new updated packet was sent which took the shorter path and reached the neighbor early. So in that case the neighbor will think that the packet which arrived late is the new updated packet and not the original one which took the shorter path. For this Sequence Number is used.

So every time a router generates a packet a sequence number will be added(Ex: sequence number is x). After 10 seconds if another packet is generated then the sequence number will be different (Ex: sequence number is x+1). So the sequence number will be incremented every time the packet is created.

The receiver will compare this number and take the packet with a larger sequence number and

delete the lesser one.

A problem will occur relating to sequence length. Since if the sequence number length is small and if the packet is generated every 10 seconds, the sequence will quickly reach the maximum number and the after that, the router will restart the sequence and send updated packets with a sequence number from the start but since that number is lower that the max sequence the receiver has, the receiver will not accept it even though the packet is new and updated(Ex: sequence number length of 4 bits). This situation is called a **Wrap up**.

So the solution is to add a larger length. That is 32 bits equal to the IP address length. Then the Wrap up will not occur any time soon since it will take 122 years to reach the max sequence number even if the packet is generated every 10 seconds.

Another problem may occur, which is the **Read error**. Sometimes the receiver may face a read error where the receiver will read the new updated packets sequence number wrong, say reading no-6 as no-66. Then the receiver will not take the new updated packet since it read the number wrong the 1st time and will reject any packet that is below no-66. So, if the read error is a very big number then it will create a huge problem.

So to counter that **Age** is used where the packet will stay for a specific time and then get deleted. Then the next pack will get happily accepted.

Age: It is Time to leave for the generated packet. It is not hop count rather it is time. Say 60 seconds then for 60 seconds the receiver will keep the packet in the buffer then it will delete it.

If the Age is shorter than the generation time of the new packet, or If the Age is shorter and the new generated packet takes longer to arrive then there will be no packet in the receivers buffer and the receiver router will be in the dark with no information till the new packet arrives.

Same goes if the Age is too big then, the receiver will suffer a lot by missing a lot of new packets and using the old outdated packets.

Information of its neighbors and their distance: It holds the information of neighbors and their distance of the router which generated the packet. This will be updated after every specific time and will start the generation of a new packet containing these updated values and the packet will be sent to the receiving neighbor routers.

4th Step Distributed link state packet-

The packet created in the 3rd step needs to be distributed. So this step optimizes where these packets need to be distributed, who needs these packets and who doesn't.

Every router will create a Distribution table. This will keep check of the packets a router received and the router will send to.

				Packet Sent (Link state packet)			Acknowledgement sent		
Owner	Source	Age	Sequence	router	router	router	router	router	router

Table: Distribution table for routers

Owner: Generator of the link state packet.
Source: The source where the router got the packet from (Ex: B got the packet generated by router A from router F. Since F is B's neighbor).
Packet Sent: This shows the routers who the packets need to be sent to. The router will not send a packet to the router from which it got it from or the owner of the packet. Meaning except the Owner and Source the packet will be sent to other neighbors. **[0 = no 1 = yes]**
Acknowledgement sent: This shows the Acknowledgement to routers the packet came from and the owner. Meaning Acknowledgement to the Owner and the source. **[0 = no 1 = yes]**

Example:

				Packet Sent (Link state packet)			Acknowledgement sent		
Owner	Source	Age	Sequence	A	C	F	A	C	F
A	A	60s	56	0	1	1	1	0	0
A	F	60	63	0	1	0	1	0	1
A	C	60	61	0	0	1	1	1	0
E	A	60	35	0	1	1	1	0	0

Table: Distribution table for router B

Here, the 3rd row data will not be inserted in the distribution table since the packer owner is router A and the sequence number is lower then router A's previous packet.

Here, the 4th row date will be inserted in the distribution table even though the sequence is lower then the 2nd row sequence, since the packet owner is router E.

5th Step Compute the shortest path-

In this step the routers will use Dijkstra's algorithm to calculate the shortest path to the destination. Then send data to the router who has the shortest path to the destination.

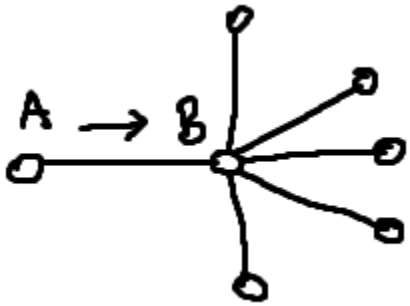
[**Note**: Link state routing algorithm is used in a form called Open Shortest Path First/OSPF]

Class-3

Congestion control Algorithm-

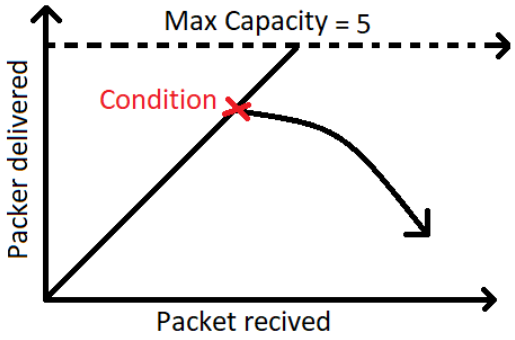
What is the congestion?

The congestion is nothing but traffic jams in a network caused by data packets.



Here, a lot of devices are connected to router B. B is redirecting all the data coming from A and other devices to their respective devices connected to B. Now, If B gets overloaded with a lot of data which surpaces the delivery capacity of router B, then data will accumulate in B. The data will have to stay in queue in router B's buffer till B can process all the data one by one. At some point even that buffer can get overfilled with data and at that moment some data will get lost. In this case congestion has been created.

Why do we need to control congestion?



When congestion is created, the delivering capacity of the router starts to degrade. Meaning that if before the congestion, the routers delivering capacity was 5 data at a time then after the congestion the capacity will start to go lower and at one point will be stopped and won't be able to deliver anymore. Then the router will be considered as down or dead and that will be a bad sign for a network. That's why when congestion occurs we have to control it.

We try to control the flow of data so that the congestion does not occur in the 1st place. But still sometimes congestion might occur and to control that we need to use some algorithms.
[**Note:** Congestion are controlled in the Network Layer]

Algorithms to control congestion (in Network Layer)-

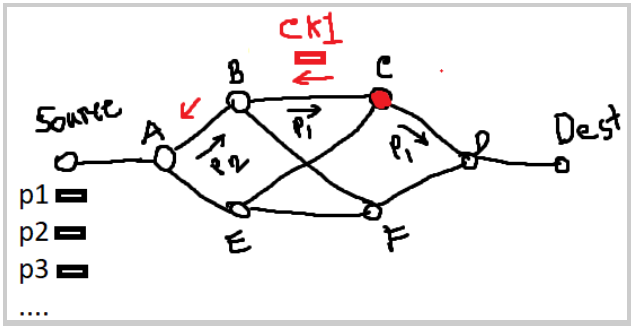
There are many algorithms. They are,

1. Choke Packet Technique
2. Load Shedding Technique

Choke Packet Technique:

It is a controlling packet. When a source is creating data packets and sending them to a destination through a router, and at that moment if that router faces a congestion, then that router will create a choke packet and send it to the source to tell it that the router is facing some sort of congestion. Then the source that is creating the data will either choose a different path or stop making data packets till the congestion is controlled. That is how a Choke Packet works.

Example:



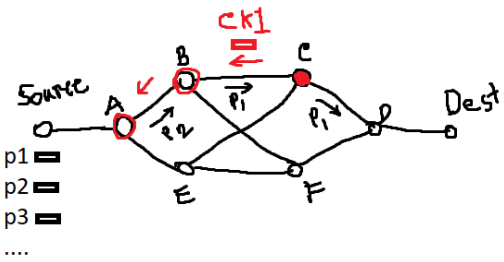
Suppose in a network above, the source is creating and sending data to the destination. At one point when the p1 packet reaches router C, router C faces congestion. After some time the p1 packet will be sent to the destination by router C but at the same time a Choke packet ck1 will be created and sent to the source telling the source that C is facing congestion. Then the source will change its path to send data or stop creating data till C gets controlled.

What are the problems with choke packets?

If the source distance is very far compared to the router facing a congestion, then by the time a choke packet reaches the source, the source will already create many more data packets and have sent it to that router. So in this case the source router will suffer by losing a lot of data packets .

So to solve this problem another choke packet technique is created called hop by hop Choke packet Technique.

Hop by hop Choke packet Technique (Improved):



It is an improvement over the normal Choke packet technique to remove the suffering. Here, the congestion affected router will send a choke packet to the source. This packet will hop from the intermediate routers between the congestion affected routers and the source. When these intermediate routers receive this choke packet, they will know that a router is facing congestion and to help that router these routers will **lower the data rate** by holding incoming data packets in their buffer. So in the path between the source and the affected router, all the other routers will receive this choke packet and lower their data rate to help the affected router to control its congestion. Then when the source receives the coke packet, it will either stop making data packets or take a different path.

That's how hop by hop choke packet technique works where every hop takes a step to control the congestion.

Load Shedding Technique:

This technique is very effective but a cruel method. Here, If a router faces any congestion then it will delete a few data packets to control its congestion. It is effective but compromises have to be made by deleting a few data packets. So the router shedded some of its load and controlled its congestion. This technique is effective in crucial moments.

Which packets to delete?

There are 2 algorithms which tells which packets to delete

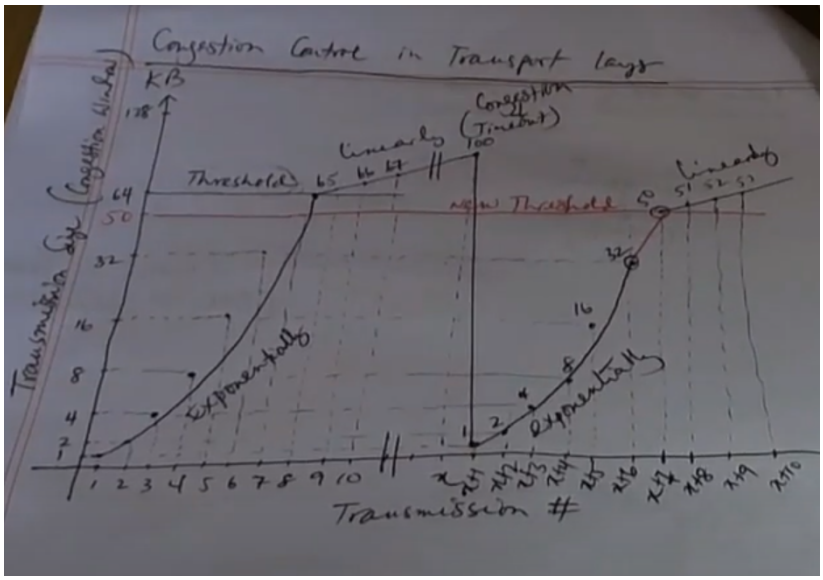
1. **Milk technique**: Fresh is good. Meaning that delete packets which are old and keep the packets which are new in the buffer.
2. **Wine technique**: Old is good. Meaning that delete packets which are new and keep the packers which are old in the buffer.

Congestion control Algorithm (in Transportation Layer)-

There are many algorithms. They are,

1. Slow start algorithm

Slow start Algorithm:



Start by sending a small size of data to the destination then increasing the size gradually to see if the network can handle the data size to the destination.

So 1st transmission will send data with size of 1. After it receives an acknowledgement of that sent data after a specified amount of time, then we can say that the network is ok and double the size of data for the next transmission. This will go on till we reach a threshold of that network.

After reaching a threshold the size increases linearly. Meaning it will increase by 1. At one point the network will give a timeout meaning that a congestion has occurred in the network.

When congestion has occurred, a new threshold will be created. This new threshold will be half of the size where the timeout occurred.

New Threshold = ½ * Size of Timeout

[**In the above graph the new threshold is ½ * 100 = 50 kb**]

Then from the next transmission the size of the data will start from 1 again and double till it reaches the threshold and repeat the process.

That is how congestion is controlled in the transport layer.

[**Note:** Time Out depends on the condition of the network. If it's in good condition the the timeout will take longer to occur]

Class-4

Quality of services (QOS)-

Need to have some Quality of services to make the user experience better. So below are some given things which are important for specific applications.

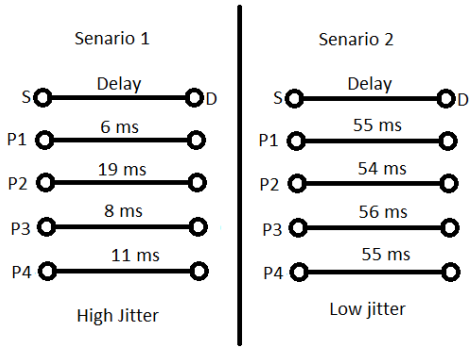
Applications	Reliability	Delay	Jitter	Bandwidth
Email	High	Low	Low	Low
Video on Demand	Low	Low/Medium	High	High
Telephony	Low	High	High	Medium
Video conferencing	Low	High	High	High

Table: Important Matrix Table

Reliability: It is how secure the data is on the network when going to the destination. Less data loss and read errors or bit errors.

Delay: It is the time delay till the information reaches the destination.

Jitter: Very important for multimedia communications. It means variations in packet arrival time. It is not Delay.



Here, there are 2 scenarios where in boath 4 packers are sent from source to destination. We can see that it takes little time delay in scenario 1 compared to scenario 2. But the delay variation is high in scenario 1 compared to scenario 2.
In multimedia communication we try to keep low jitters so that all the packets reach the destination between a specific time range which are closer to each other even if the delay is big.

Because audio and video frames have to have synchronization. Or audio will come 1st and video later or some missing frames in both audio and videos.

Bandwidth: If we need to send a very large amount of data then we need high bandwidth.

Techniques to achieve quality of service-

There are many techniques to achieve good QOS. They are,

1. Over provisioning
2. Traffic shaping

Over provisioning:

Doing more than required. Not caring about the application and trying to better everything. Using the best devices, best transmission media, and high bandwidth. It's not cost effective but will work for all types of applications and solve the QOS.

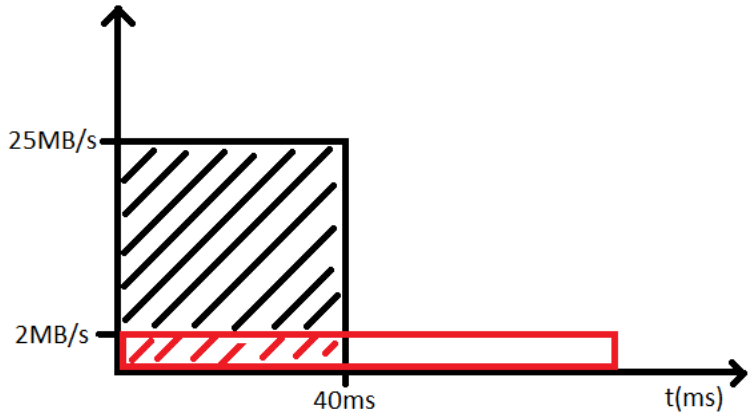
Traffic shaping:

Shaping the data traffic, Regularizing the irregular flow of data, controlling the flow of data. **The Leaky-Bucket** algorithm is used here.

The Leaky-Bucket algorithm: It is like a bucket filled with water with a leak leaking water in small amounts. Pc devices create bursting nature data. Meaning it creates a huge amount of data at one time and sends it, then the other times creates small amounts of data. The network can not always handle this brusty data and will create congestion. So it needs to be shaped. So since a network interface card is the last point between a PC and network, here we apply The Leaky-Bucket algorithm.

So, in the algorithm there the bursty data will pool in a buffer and there will be a leak which has the size the network can handle the data. And from that leak in a steady fashion the packets will go to the network at a lower data rate. So the bursting nature data is regularized. The buffer has to have sufficient size to hold the huge pool of data.

Example: PC generated data at 25MB/s rate in 40ms. The network capacity rate is 2MB/s. What is the time to send the data at the rate of 2MB/s in the network?



Given,

Rate = 25MB/s
Time = 40 ms

Now, the total data generated from the PC in 40ms,

=> Data = Rate * Time
=> Data = 25MB/s * 40ms
=> Data = 25MB/1000ms * 40ms
=> Data = 1MB

So, 1MB data has been generated in 40ms.

Now the time it will take to send the data at rate of 2MB/s,

=> Time = Data/Rate
=> Time = 1MB/2MB/s
=> Time = 0.5s
=> Time = 500ms

So, 500ms time is required to send the data at 2MB/s rate.

Ans:-

Fragmentation-

It means fragmenting data in little chunks. If the data we want to send is too large for the network to handle then we fragment the date.

There are 2 types of Fragmentation,

1. Transparent Fragmentation
2. Non-Transparent Fragmentation

[**Note:** Which is good and which is bad depends on the type of network. Ex: Transparent Fragmentation works good in the virtues circuit network where it used virtues circuit number]

Transparent Fragmentation:

When a packet enters a network and that network can not handle the whole packet, then the packet will fragment according to the network's capacity in the route from which it enters to the network. Then will reassemble in the router from which the packet will leave the network. The packet will keep on doing this through many other networks till it reaches the destination. This fragment and reassemble is called Transparent Fragmentation.

Here, it has to do a cumbersome job of reassemble every time it leaves a network

Non-Transparent Fragmentation:

When a packet enters a network and that network can not handle the whole packet, then the packet will fragment according to the network's capacity in the route from which it enters to the network. But it will not reassemble when leaving that network. Instead it will leave and enter other networks by being fragmented. It will only reassemble when it reaches the destination. The fragments will take whatever path they choose but will reassemble in the destination.

The fragments can be again fragmented to enter other networks if required but will not reassemble until it reaches the destination.

Here, it doesn't have to do the cumbersome job of reassembling every time it leaves a network. But every fragment has to have a header containing the information of destination. So overhead will increase here.

Class-5

UDP (User datagram protocol) Header-

It is connectionless and more like a postal service. No need for connection to send data to the server. It is unreliable meaning packets can get lost. No order so packets will not maintain any sequence.

UDP is fixed at **8 byte**

Source port (16 bits)	Destination port (16 bits)
Length (16 bits)	Checksum (16 bits)

Table: UDP Header

Source port: Port number of source

Destination port: Port number of destination

Length: It is the length of the header+payload.

Checksum: It is used to control errors

What is required for a client to connect to a web server through Socket & TCP connection?

TCP Header-

TCP header can be a minimum of 20 bytes/160bits or a maximum of 60 bytes.

Source port Number (16 bits)				Destination Port Number (16 bits)				
Sequence Number (32 bits)								
Acknowledgement Number (32 bits)								
Header Length	Reserved	U	A	P	R	S	F	Window Size (16 bits)

(4 bits)	(6 bits)	R G	C K	S H	S T	Y N	I N	
TCP Checksum (16 bits)				Urgent Pointer (16 bits)				
Options(0 to 10 words if 32 bits)								

Table: TCP Header

Source port Number (16 bits): Port number of source

Destination Port Number (16 bits): Port number of destination

Sequence Number (32 bits): It is the number to ID each bite/packet sent to the destination. (Ex: x)

Acknowledgement Number (32 bits): It is sent by the receiver to acknowledge the packet/bit it received from another host. The number is the next expected packet/bit. (Ex: x+1)

Header Length(4 bits): It is the length of the header. The decimal conversion of the 4 bits will be then multiplied by 4. That is the real header length (Ex: 1111 = 15, now 15 * 4 = 60 bytes)

Reserved(6 bits): These 6 bits are reserved for later use.

URG(Urgent flag): If some of the data in the packet is urgent data then this bit will be 1 else 0.

ACK(Acknowledgement flag): If sending acknowledgement then this bit will be 1 else 0.

PSH(Push flag): If this bit is 1 then the data will be pushed to the application layer and not wait till the buffer is full. (Ex: important data will be pushed to the receiver rather wait till the buffer is full)

RST(Reset flag): The connection will reset in the bit is 1 else 0.

SYN(Synchronization flag): if a host wants to start a connection the 1st data will have this bit at 1. The rest will have 0.

FIN(Finish flag): if a host wants to terminate its connection then this bit will be 1 else 0.

Window Size (16 bits): It is the size of the buffer. It tells how much size of data the host can receive. Host sends it to another host when sending data.

Checksum (16 bits): It is used to control errors.

Urgent Pointer (16 bits): It tells the length of the urgent data.

Options & Padding(0 to 10 words if 32 bits): If extra data needs to be sent it can be through this. Like MSS (Maximum Segment size).

Transmit Entity-

Network Service Access Point (NSAP): It is a single point connected to TSAP in the transport layer from the network layer.
Transport Service Access Point (TSAP): Transmit entity talks to alot of points of the application layer. And one of the points is TSAP.

Transmit entity:
Transmit entity talks to alot of points of the application layer. Transport Entity is both connected to NSAP and TSAP. The Transmitting entity listens to the IP address in the network layer to see what kind of request comes. Transmit entity also communicates with multiple points in the application layer.

Endpoint: It is a combination of IP address and port. Endpoint = port + IP

Sockets: When a request like http://www.cnn.com, finds the machine with the IP address, 1st the request will go to the NSAP, then the Transmit entity will listen to NSAP for the request and see what port the request needs. Since the above is a http:// request so the Transmit entity will make a connection to that port. That is how an endpoint will be created. This endpoint is called **socket**.
Multiple sockets can be created depending on the port type like ftp, http etc. Multiple sockets can be created on the same port as well since multiple clients may request the same port. Under a single socket multiple sockets can be created called socket identifiers.

So that's how the Transport Entity is connected to the network and application layer.

TCP Connection establishment-

It is a connection between 2 hosts, Host A and B. To establish a connection, 1st TCP uses a three-way handshake to establish a connection. Then data transmission happens where they send data to one another.

Three-way handshaking:
1st host A will send a connection request (CR) packet containing all the required information **(**like encoding mechanism, protoco, buffer size etc.**)** about that host to host B and request a connection.

After receiving the request of host A, host B will check if host B can work with the information of host A. If the host can work with the information then host B will send a Connection Accept (CA) packet else host B will not accept and release the connection (Ex: if you want to connect with a restricted machine then the connection request will not be accepted).

Then after receiving the CA packer from host B, host A will also check if host A can work with the information of host B. If so then host A will send a CA packet to host B. Now the ultimate connection is established and both hosts will start to send data to each other.

This whole process is called **Three-way handshaking**.

Data transmission:
Host A will send a data packet with a sequence number of x and the size of its window/buffer. Host B will accept it and store the packet in its own buffer and send host A a packet with a sequence number of y and acknowledgement of the next packet x+1 from host A and host B's size of window/buffer. Host A will receive and store it in its buffer and send another packer with sequence number of x+1, acknowledgement of the next packet y+1 from host B and host A's size of window/buffer.

This will go back and forth till one host buffer gets full. At that time the other host will send a 1byte of Controlling packt to remind the other host that this host is waiting to send data to that host so it should clear its buffer.

This whole process is the **Feedback based flow control**.

Example:
Host A has a buffer of 4kb and host B has a buffer of 6kb.

Host A will send a data packet of size 1kb with a sequence number of x and its window/buffer size 4kb.

Host B will receive the packet and store it in its buffer. Now the free space of the buffer is 5kb. Now host B acknowledges the data packet and will also send a data packet of size 2kb with a sequence number of y, the size of its window/buffer 5kb and acknowledge number of x+1. Here x+1 means that host B acknowledged the previous packet x and is expecting the next packet x+1. It is a little bit of optimization.

Host A will receive the packet and store it in its buffer. Now the free space of the buffer is 2kb. Now host A will also send a 3 kb data packet with the sequence number of x+1, ts window/buffer size 2kb and acknowledge number of y+1.

Host B will receive the packet and store it in its buffer. Now the free space of the buffer is 2kb. Now host B will also send a 2 kb data packet with the sequence number of y+1, its window/buffer size 2kb and the acknowledged number of x+2.

Host A will receive the packet and store it in its buffer. Now the free space of the buffer is 0kb. Now host A will also send a 2 kb data packet with the sequence number of x+2, its window/buffer size 0kb and acknowledge number of y+2.

Now host B can not send any more data to host A until host A frees its buffer space. But host B will send a 1byte controlling packet every now and then to let host A know that host B has more data to send and it is waiting for host A's buffer to be cleared.

Now the above will happen if host A and B do not clear their buffer after every data sent.

TCP connection Release:

TCP connection release is to disconnect the host's connection. Even if host A wants to finish their connection, host B might still have data to send. So a single host can not end their connection. Both sides need to agree to disconnect meaning when both sides have no data to send only then can they disconnect. So they need to find a specific time to end the connection.

Two army problem Time: lecture vod time: 30:00

Two army problems occur till this day. So to end their connection the hosts have to send acknowledgement multiple times to each other to agree to end their connection. Nowadays we can send data very fast so the probability of the data getting lost is lessened. So sending acknowledgement multiple times to each other works. Still this is not a 100% guarantee.

Class-6

Domain Name System (DNS) -

Maps the domain name to its IP.

The whole internet is divided under a DOT server into some top level domain. Like .com, .edu, .mil, .gov, etc. There are also domains divided for countries like .us, .bd etc.

These top level domains also have named servers or dns under them. There are also sub domains under these top level domains. Ex: CNN.com is under the .com domain. These websites are registered under these top level domains. There can not be duplicate websites under the same domain name.

Example:
If a client wants to browse CNN website from EWU then they will type http://www.cnn.com. To start a TCP connection with the website it will need a port and the IP of the website. The port is given on the http:// protocol so the only thing it needs is the IP. So, 1st the DNS server will check its cache if the site has visited previously, if not then it will look at the upper level DNS in .edu.

Since the website is from the .com domain it will then go and check the DOT server and find the .com domain. There it will search for the cnn.com domain. When it finds it it will go to that domain and get the IP address of the website from the cnn.coms domain. Then it will return to the client. Now the client has both the **port** and the **IP** to start a TCP connection with the cnn website.

Web Server-

Www: WWW is an architectural framework to access linked documents. It is also the host name of the web server. It can be anything like abc, xyz etc.

What happens in the client side when a client connects to a server-

A simple flow is given below, even though a lot more detailed work is done when connecting.

1. **Sept-1:** Browser determines the URL.
2. **Sept-2:** Browser asks its local DNS for the IP against the URL.
3. **Sept-3:** DNS replies with the IP. If not in the local DNS then the DNS will look for the IP in the upper levels.
4. **Sept-4:** Browser makes a TCP connection with the machine with the IP address and port.
5. **Sept-5:** The website machine servers the requested files to the client.
6. **Sept-6:** After the data transmission the TCP connection is released.
7. **Sept-7:** Browser displays the file/page.

In the middle of all that the client side may use many plugins. The browser may use helper applications if the browser faces a problem interpreting any packets.

What happens in the server side when a client connects to a server-

A simple flow is given below, even though a lot more detailed work is done when connecting like checking authenticity, authorisation to see the file etc.

1. **Sept-1:** Accepts the TCP connection.
2. **Sept-2:** Gets the requested file from the HDD.
3. **Sept-3:** Returns it to the client
4. **Sept-4:** Releases the TCP connection.

Optimization of the Web Server-

The most limiting factor in a web server is the HDD. Since the HDD has mechanical components, the process gets slowed if the HDD is used under a lot of pressure.

Incorporating Cache memory:
If a cache memory is incorporated in a web server, then the heavily used pages can be stored in the cache memory from the HDD. Then the requests to those pages to the web server can be taken from these cache memory rather than the HDD and server to the client much quicker.

Incorporating Multi threading:
Multiple threads help to work parallely. These threads will be connected to the cache memory. Then all the requests will go to these multi threads and the threads will connect to the cache memory and serve requests to the client parallely.

Server Farm:
Multiple optimized web servers are connected to a SAN and will be connected to a single front-end and a router with a high speed LAN. With this type of setup the number of requests to the server to the clients will increase a lot.

Storage area network (SAN): It is the connection of multiple HDDs in a network.

Optimization of the Server Farm-

TCP handoff:
In a normal server farm the front-end will accept the requests and will send it to one of its servers. Then the server will get the requested file from its cache and if not it cache then from the HDD to its cache to the front-end. Then the front-end will reply to the client.

This request acceptance and reply process makes the front-end too busy. So to make it easy for the front-end these individual web servers/nodes can directly connect to the client to give them replies to their requests rather than give it to the front end. That way the front end will get some of its load off. It is called **TCP handoff**.

Specific requests will be handed over to a specific machine:
If a web server has worked on a specific request previously then that specific type of request will only be given to that specific machine/node since that request is already on that machine's cache memory.