CS 348: Computer Networks

Dion Reji

:

APPLICATION

TRANSPORT

NETWORK LAYER

DLL → FRAMING

DLL → MAC

PHYSICAL LAYER → Transfer of signals (bits)

DLL : Data Linked Layer

SIMPLEX : One way communication
eg: Radio

FULL DUPLEX : Simultaneous communication in both direction
eg: 4G channel

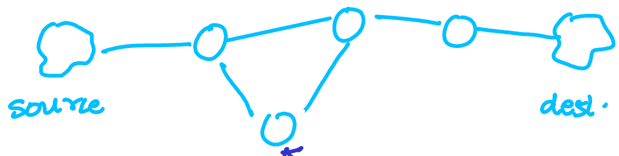
HALF DUPLEX : in both directions but not simultaneously
eg : WiFi

Which all autonomous systems to choose in path from source to dest

BGP : Border Gateway Protocol
across autonomous systems

IGP: Interior Gateway Protocol
inside autonomous system.

what path inside an autonomous system



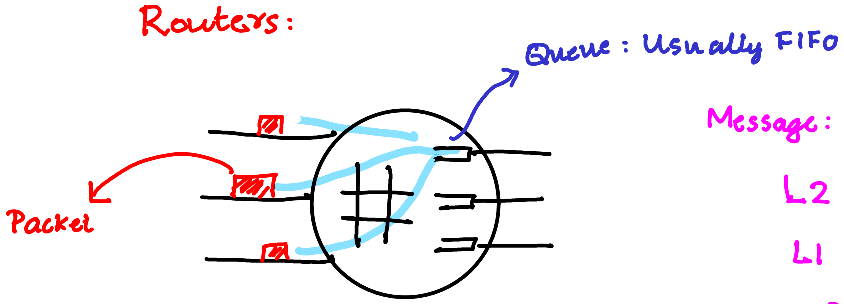
autonomous systems

lower level

IPv4 eg: 72.8 3. 5. 25

high level (like country in Postal address)

Routers:

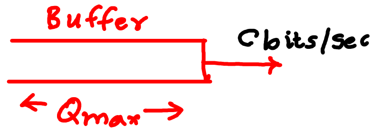


Message:

- L2 : FRAME (1 unit of data)
- L1 : SYMBOL
- L3 : PACKET
- L4 : TCP → SEGMENT
  - UDP → DATAGRAM

DROP: Packets discarded if input rate > output rate and queues are filled

CONGESTION CONTROL



Why can't we increase the size of buffer?

- Cost is heavy
- End to end delay of packet (from source to dest) $= \sum_{i=1}^{N} \frac{Q_{max}^{(i)}}{C^{(i)}} + \sum_i d_i +$ transmission delay

d_i → speed of light delay.

delay $= \frac{Q_{max}^{(i)}}{C^{(i)}}$

$C^{(i)}$ → ith router

'Better drop than delay'

Router Table (longest prefix match)

Eg:		EXIT
	72.*.*.*	R1
	72.68.*.*	R2

Choose the one with longer prefix

Layer 4: Transport Layer

Congestion Control → Transmission Control Protocol (TCP)

Reliability → Transmission Control Protocol (TCP)

(eg: file transfer)

(In video call, we don't care reliability)

There is also UDP in L4 : does pretty nothing.

Layer 5: Application Layer.

Web: HTTP

Email: SMTP

VoIP : TCP/UDP

Voice Over

Text Message

P_2P, ...

Specifications of protocol in RFC: Request for Comments

Overview: OSI5 layering

Application	L5	Web	Email	VoIP	Text	P_2P
Transport	L4	TCP		UDP		
Network	L3	IP (Internet Protocol)				
DLL	L2	WIFI, 4G, Ethernet, Bluetooth				
PHY	L1	Wireless, Optic Fibres, WiFi-PHY				

Design Protocols in modules,
Each subproblem handled by some protocol

Advantages of Layering/Modularity:

1. Ease of Developement -- look at only certain sets of problems handled by particular layer
2. Debugging
3. Many applications and physical technologies working together -- we can use any appliation with any physical device. Imagine what woud have happened if we could only use WhatsApp over wifi ?
   i.e, we can have different choices at different layers: some sort of compatibility
4. Ease of modification -- Only change 1 layer to address a problem

Disadvantages of Layering:

1. Opaqueness about other layers. This can cause some issues:
   For ex. TCP controls traffic congestion. If there is a packet drop, it assumes that there is some queue full at some router. However, that may not be the case. The reason is that routers doesn't talk to TCP. So, TCP might incorrectly predict packet drops.

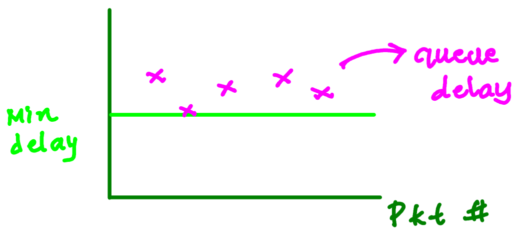
2. Redundancy of Tasks. Its not the case that every layer looks at different sets of tasks.
   For ex. TCP handles retransmissions as it needs to ensure reliability. But, there is also MAC layer retransmission.

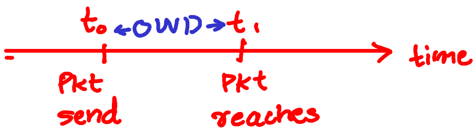
3. Suboptimality.

For ex. consider a VoIP application. we need the lowest possible delay from source to destination. But, there is no way to achieve that. We need to be satisfied with the path in the Layer 3 network we get in the internet. We can specify for the shortest path. What we get is the BEST-EFFORT, i.e, there is no guarantee on quality of service (given in terms of some metrics like amount of packet drop, latency(delay))

Types of Latency metrics:

1. One-way delay: Say we send a packet at time t0, that goes through the network and reches at t1.
   OWD — Then the one-way delay of the packet = t1 - t0.
   More the queueing delay, more the one way delay





2. Round Trip Time: Say we send a packet from source to destination, and the destination responds with an
   RTT — acknowledgement which is another packet that goes back to source. If the packet is sent at t0, and the acknowledgement is recieved at t2, RTT = t2 - t0

In VoIP, we need the delay (RTT/OWD), to be ~few 100ms at most

JITTER: Variability in OWD Latencies.

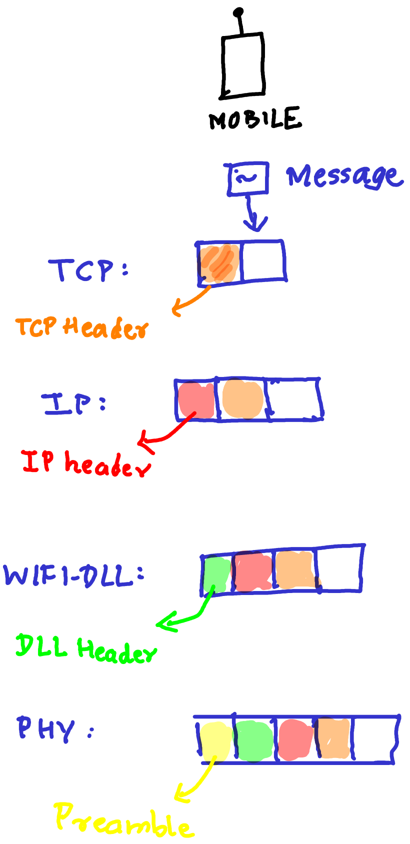
Consider some packets labelled 1,2,3,... with respective OWDs d1,d2,d3,... These packets are sent uniformly.

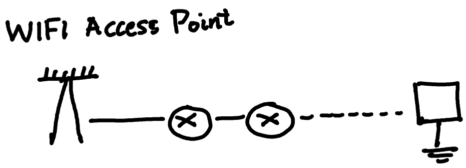
$$e_K = |d_{K+1} - d_K|$$

$$\text{Average Jitter} = \frac{1}{n-1} \sum e_K \quad \text{(for } n \text{ packets)}$$

Telephone Networks are designed for only Voice applications. Thus, they have lower jitter, RTT, OWD, no data loss, etc.

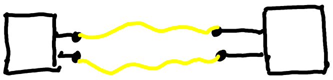
How different protocols interact in practice?



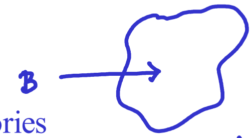


PHYSICAL LAYER

Wired Media :



Communication via electrical signals



$emf \propto A \frac{dB}{dt}$

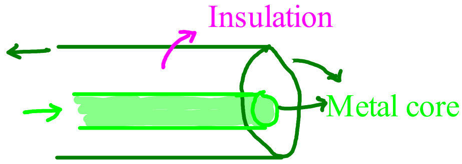
Need to minimize area so that opposing emf is reduced in the loop

In order to minimize the area, we use twisted loops(pairs). There are different categories of twisted loops:

- Cat 3: supports 10 Mbps upto 100 m
- Cat 2: supports 100 Mbps to 100 m
- Cat 1: supports 1000 Mbps to 100 m

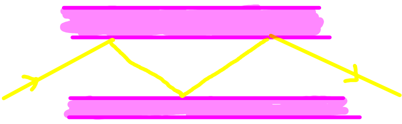
Usual ethernet cable contains multiple twisted pairs

We also have co-axial cables.



Thin Net Co-ax cables (0.2 inch in diameter): 100 MBps, 200 m
Thick Net Co-ax cables(0.4 inch in diameter): 100 Mbps, 500 m

Optical Fibres:

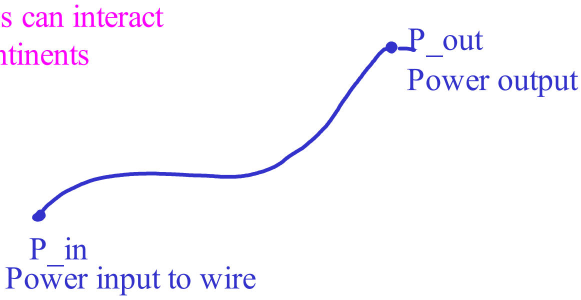


Critical angle = $\sin^{-1}\left(\frac{n_2}{n_1}\right)$

Single Mode optical fibres: only a single ray passes through
Mutli Mode optical fibres: multiple rays pass through

- Single mode is better for communication than mult-node
  Because in single mode only one ray goes, so it goes almost intact.
  In multi-node, the output pulse can be distorted as multiple rays can interact

Optic fibres are used in submarine cables connecting different continents



Attenuation: A paramter to measure the performance of a wire

We would like to consider $\frac{P\_in}{P\_out}$

But this ratio can take a wide range of values. So we look at

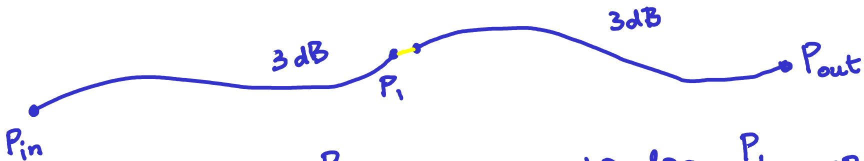
Attenuation = $10 \log_{10} \frac{P\_in}{P\_out}$ Unit: decibels

Ex: P_out = P_in/2

Attenuation = 10 log (2) = 3dB

$\frac{P_{in}}{P_{out}}$	dB
2	3
4	6
10	10
10^k	10k

Ex:



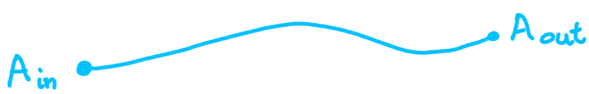
$10 \log_{10} \frac{P_{in}}{P_1} = 3$ $\qquad 10 \log_{10} \frac{P_1}{P_{out}} = 3$

$\Rightarrow 10 \log_{10}\left(\frac{P_{in}}{P_1} \cdot \frac{P_1}{P_{out}}\right) = 6 \Rightarrow$ Net attenuation = 6dB

We can also think of attenuation in terms of amplitude.

Power $\propto$ (Amplitude)^2

$$\text{Attenuation} = 10 \log_{10} \left(\frac{A_{in}}{A_{out}} \right)^2$$
$$= 20 \log_{10} \left(\frac{A_{in}}{A_{out}} \right)$$



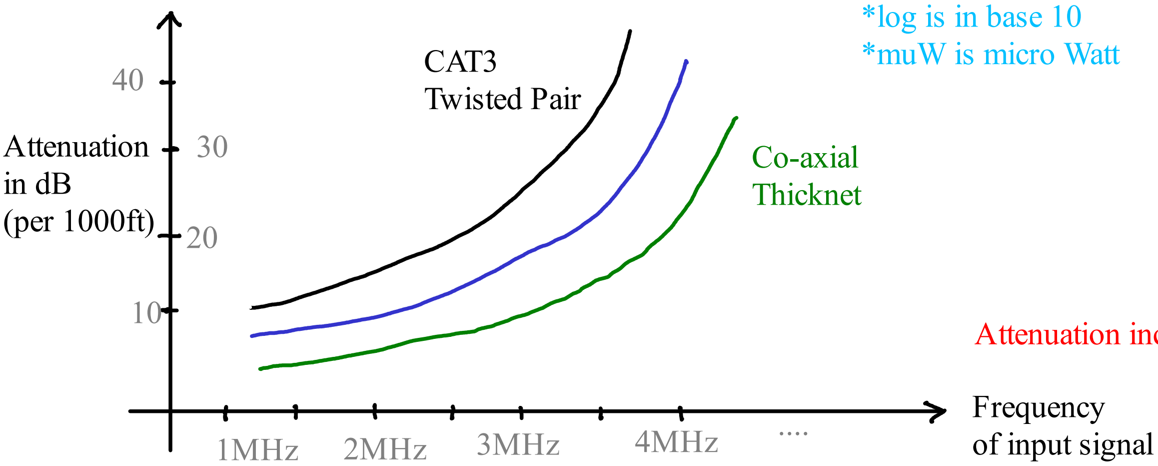
Absolute Power in Decibel Scale: We use 1mW as a reference.

For ex. we can express power P(watts) in dBm as follows:

$$10 \log_{10} \frac{P}{10^{-3}}$$

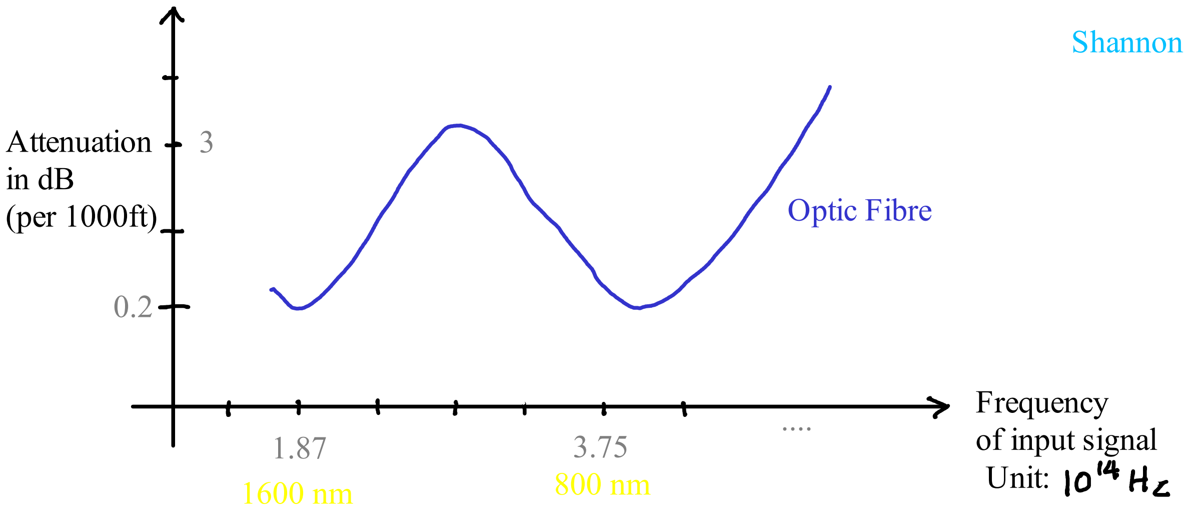
Ex. P = 1mW = 10 log(1) = 0 dBm
P = 2mW = 10 log(2) = 3 dBm
P = 10mW = 10 log(10) = 10 dBm
P = 1 muW = 10 log(0.001) = -30 dBm

*log is in base 10
*muW is micro Watt

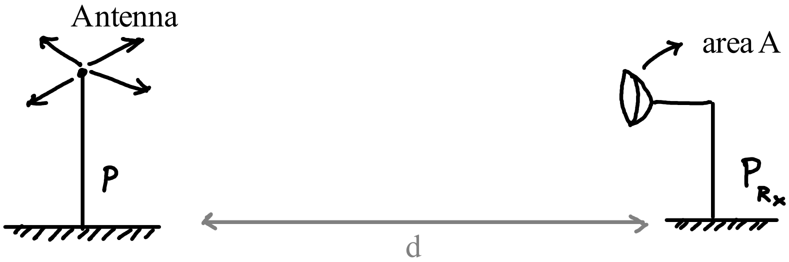


Attenuation increases with frequency!

Shannon → Entropy
Shannon → Capacity

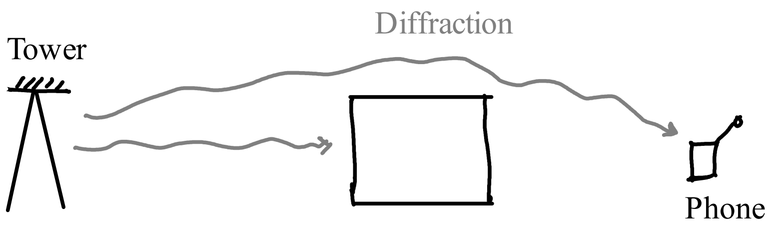


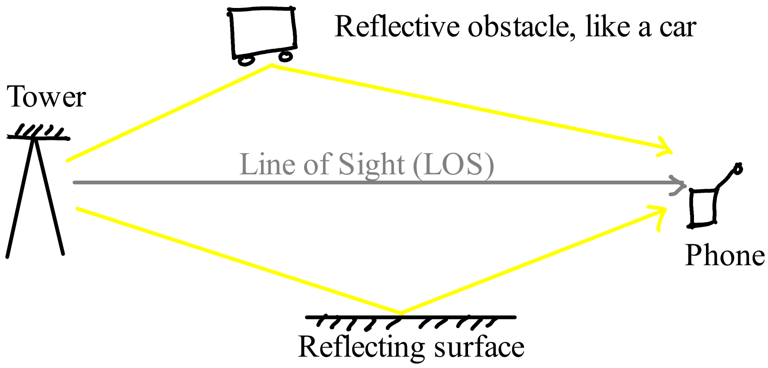
Wireless Channels



$$P_{Rx} = \frac{P \cdot A}{4\pi d^2} \quad \text{(ideal)}$$

Reality : $P_{Rx} \propto \dfrac{P}{d^{\alpha}}$ where $2 < \alpha < 5$





MULTI-PATH

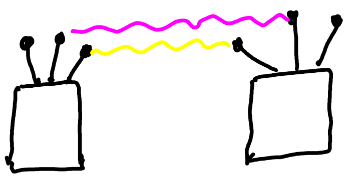
Comparison of wired and wireless channels:

1. Attenuation is much higher in wireless channels than in wired.
2. Interference can happen in wireless, between signals when two devices communicate in the same channel.

WiFi uses the unlicensed band, a 80 MHz frequency band at 2.4 GHz. Nobody pays in this spectrum. But interference can be high, as this can be used by anyone (unlicensed). There are liscensed bands in the spectrum, owned by some ISPs like Jio, Airtel, where they have sole control and thus less interference.

3. Diffraction: In wireless communications we use radio waves which can bend around obstacles.
4. Multi-path: LOS is the path straight from source to dest. There can also be other paths due to reflection from other objects. These multi-paths can be good or bad as they can result in constructive or destructive interferences.

Modern phones uses MIMO with multiple antennas.
This ensures that there is high chance of constructive interference between some antenna pair.
Massive MIMO contains a large number of antennas.
This can be used to ensure constructive interference in one direction and destructive everywhere else

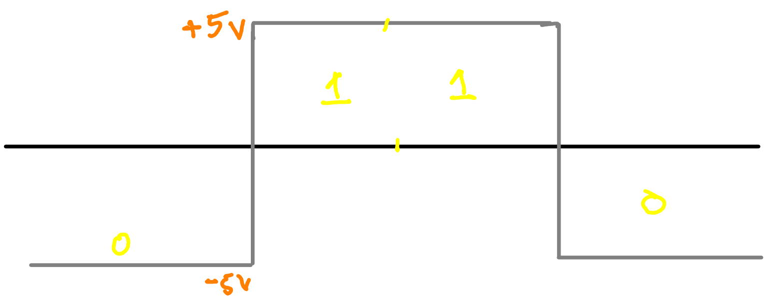


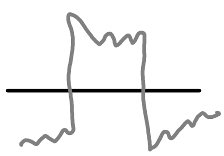
SIGNALLING:

Wired:

Use two voltage levels: +ve for 1 and -ve for 0

Non Return to Zero (NRZ)



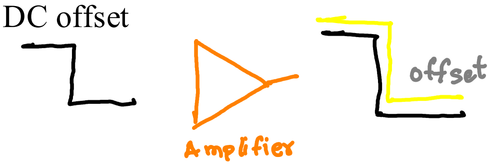


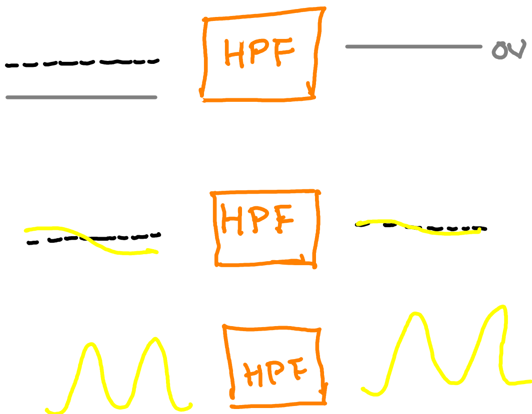
corrupted one with noise recieved at dest

Issues with NRZ:

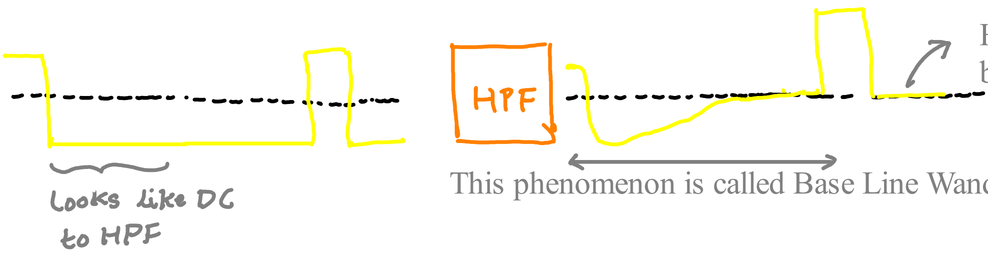
1. (Clocks not in SYNC - diff frequency at Tx and Rx):
   Suppose we send a signal like 1 0 0 0 0 0 0 0 0 0 ..... The reciever needs to count the number of zeroes. Say that the reciever has a faster clock, then he counts more zero bits, thus getting a wrong information.

2. (Base Line Wander):
   Consider the same case of sending 1 and consecutive 0's. We send this to an Amplifoer, which is not perfect and create an offset. So, when it reaches reciever it can be attenuated and noisy.
   To overcome this, we use a High Pass Filter, which removes DC offset and low frequency signals





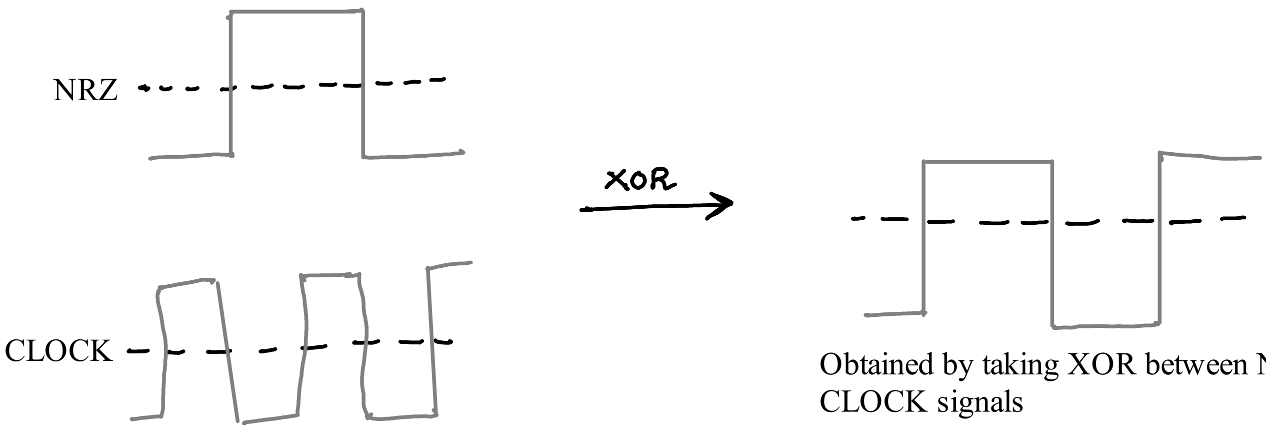
Band pass filters allow only signals of specific bands: say we need to separate Jio and Airtel sigs



High chance that this can be taken as a 1 instead of 0

This phenomenon is called Base Line Wander

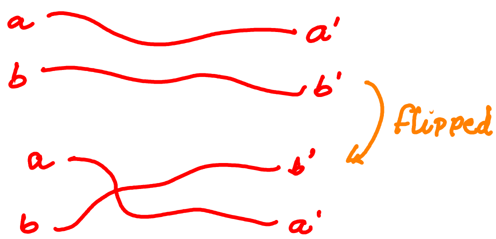
3. (Manchester Coding: Used in Ethernet)



Obtained by taking XOR between NRZ and CLOCK signals

Here, Signal transition every bit period and average signal per bit period is 0

One issue is that the wires gets flipped, if there are multiple wires.



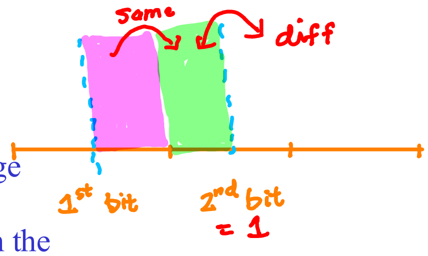
So, we need to know the polarity in Manchester coding.

Is there a way to know if we should take a'-b' and b'-a'?

One way is to communicate via preamble, but the beginning few bits of preamble may be lost. So do sth at the end of p

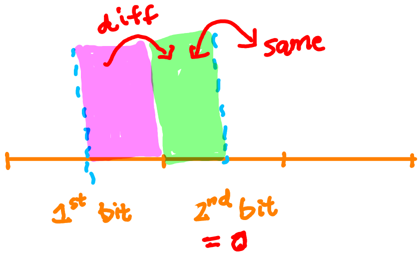
In ethernet, they send 1 0 1 0... etc, and at the end it sends two 11's

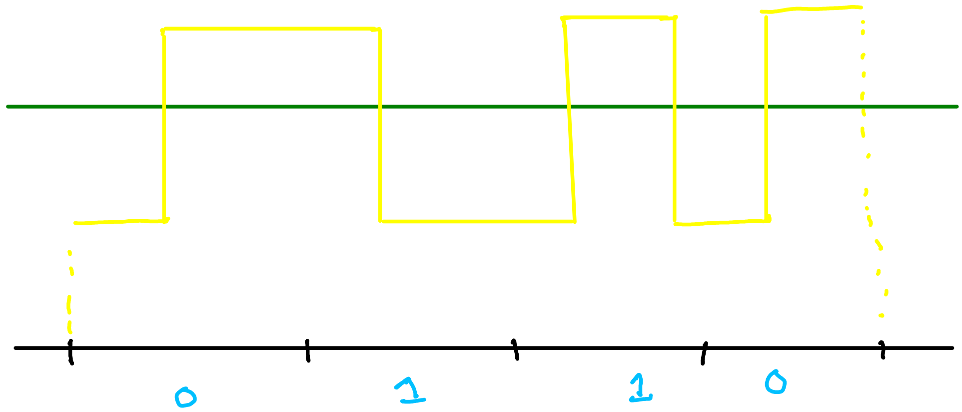
4. (Differential Manchester coding)



Rule:

- If bit = 1: voltage in the 1st half of the bit period is same as the voltage in the last half of previous bit period
- If bit = 0: voltage in the 1st half of the bit period is opposite of that in the last half of privous period

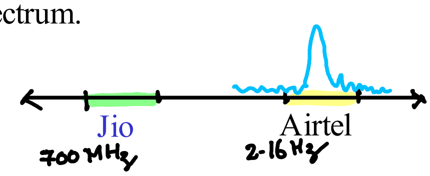




MODULATION IN WIRELESS CHANNELS

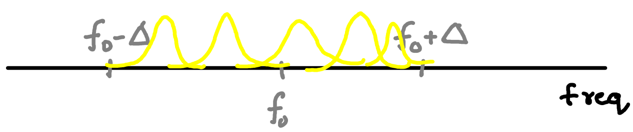
In wireless, Govt. decides which company uses which band in the spectrum.

Provided the transition power is less than a threshhold, it is possible to transmit in unliscenced bands.

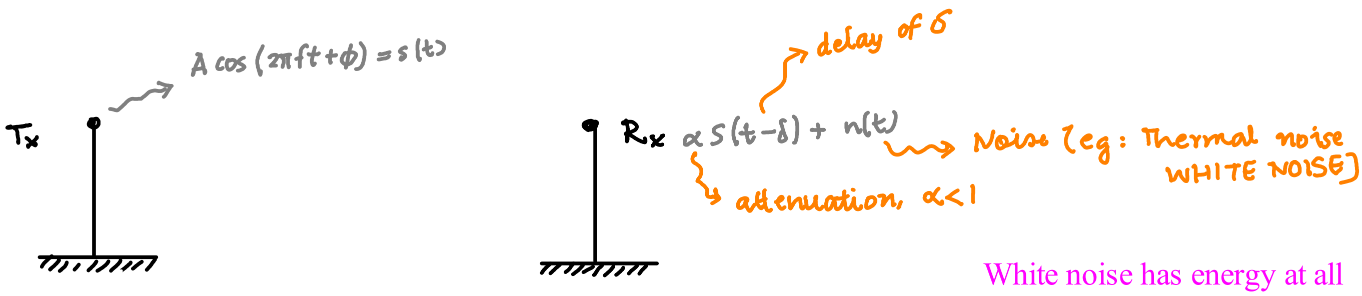


How to confine to communicate only in a fixed range/band?

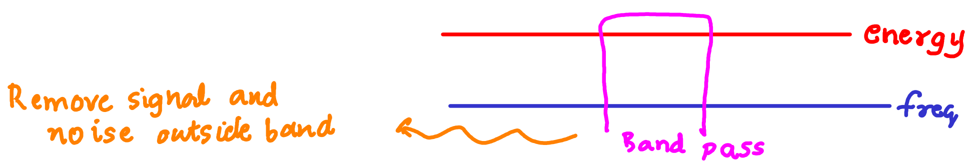
Wave is given by $A \cos(2\pi f t + \phi)$



Send not in just one frequency, but over multiple frequencies such that there is not much spill over oustside the band. Split the band into smaller bands, and send in each small band

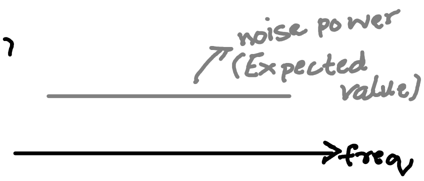


White noise has energy at all frequencies.



Noise: White, Gaussian, Additive (AWGN)

- Additive because it gets added to the signal: $\underbrace{\alpha s_1(t-\delta)}_{\text{SIGNAL}} + n(t)$
- White because its uniformly spread over the full band:



Qn: $r(t)$ is recieved.

$\rightsquigarrow s_0(t)$?

$\rightsquigarrow s_1(t)$?

$$r(t) = \alpha s_1(t) + n(t)$$
$$\text{or,} \quad \alpha s_2(t) + n(t)$$

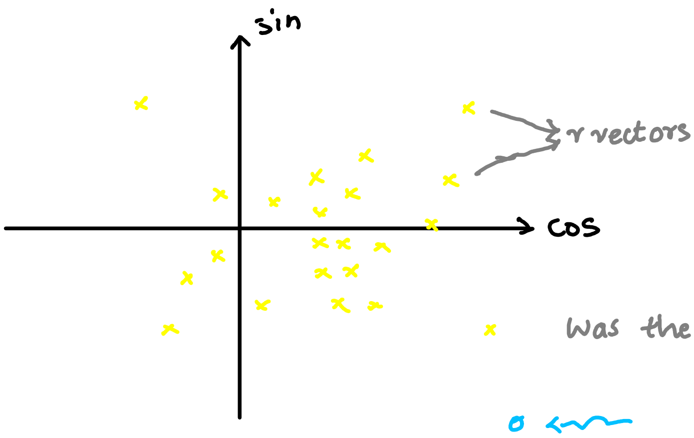
$$s_{1x} = \langle s_1(t), e_x \rangle = A\sqrt{\frac{\pi}{2}}$$
$$s_{1y} = \langle s_1(t), e_y \rangle = 0$$

$$r(t) \longrightarrow r_x, r_y$$
$$r_x = \langle r(t), e_x \rangle$$
$$r_y = \langle r(t), e_y \rangle$$

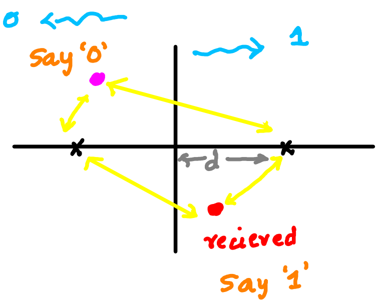
$$r(t) = \alpha s_1(t) + n(t)$$
$$n_x = \langle n(t), e_x \rangle$$
$$n_y = \langle n(t), e_y \rangle$$

Each is i.i.d Gaussian Random variables

Eg : 1, 1, 1, 1, ...



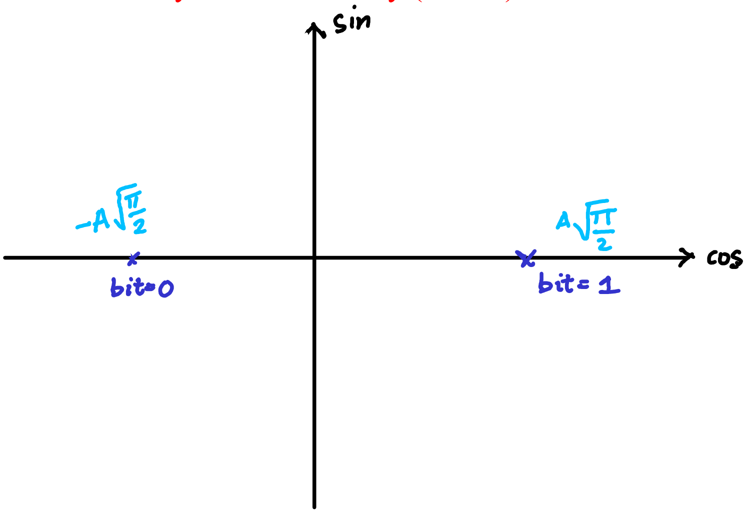
Was the bit 0 or 1?



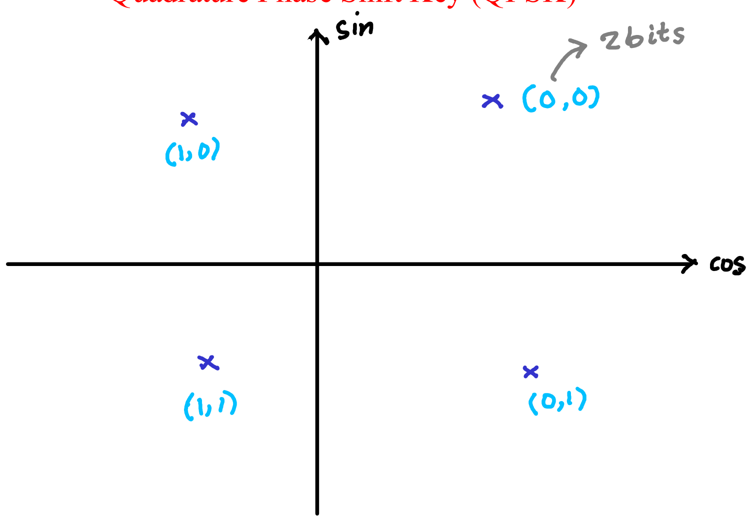
Suppose we know the constellation after attenuation and without noise.

We look at the nearest one: s_1 or s_0, and accordingly conclude. So, everypoint on left half plane will be called '0', and those on right half plane be called '1'.

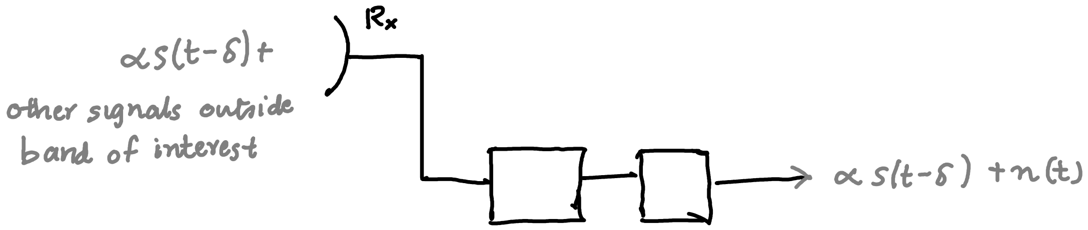
Binary Phase Shift Key (BPSK)



Quadrature Phase Shift Key (QPSK)

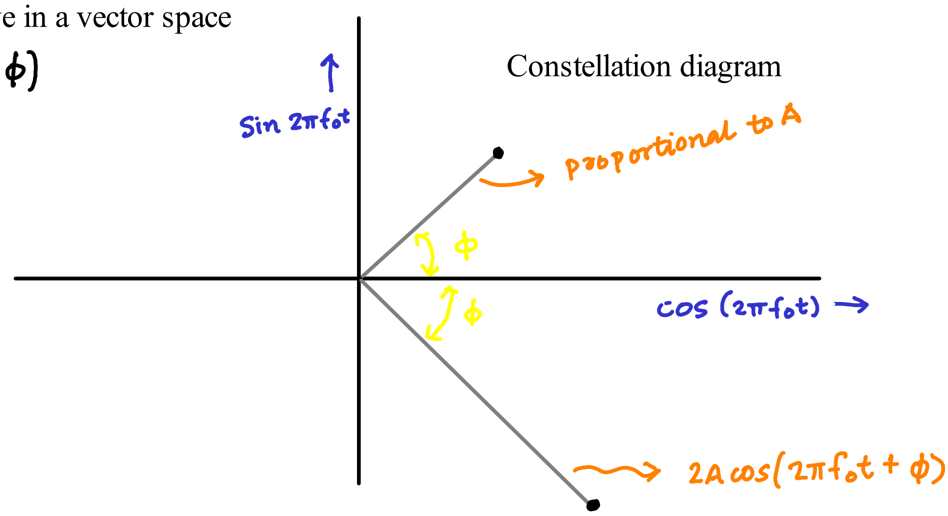


In $\alpha s(t-\delta) + n(t)$, $n(t)$ is noise after passing through Band pass filter.



Representing the wave in a vector space

$A\cos(2\pi f_0 t - \phi)$



In this vector space:

$a(t) = \ldots\ldots \quad 0 \leq t \leq T$

$b(t) = \ldots\ldots \quad 0 \leq t \leq T$

Inner Product: $\langle a(t), b(t)\rangle = \int_0^T a(t)b(t)\,dt$

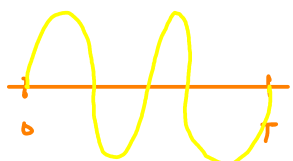
Unit vectors: $S_1 = \sqrt{\frac{2}{T}}\cos(2\pi f_0 t) \qquad T = \frac{1}{f_0}$

$S_2 = \sqrt{\frac{2}{T}}\sin(2\pi f_0 t)$

Q: Is the inner product of these unit vectors zero?

$$\langle s_1(t), s_2(t)\rangle = \frac{2}{T}\int_0^T \sin(2\pi f_0 t)\cos(2\pi f_0 t)\,dt$$

$$= \frac{1}{T}\int_0^T \sin(4\pi f_0 t)\,dt = 0$$



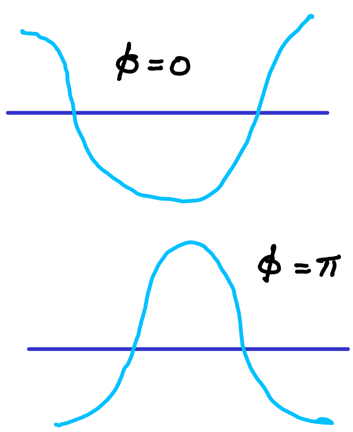
Q: Is the inner product of s1 or s2 with itself 1?

$$\langle s_1(t), s_1(t)\rangle = \frac{2}{T}\int_0^T \cos^2(2\pi f_0 t)\,dt$$

$$= \frac{1}{T}\int_0^T (1 + \cos(4\pi f_0 t))\,dt = 1$$

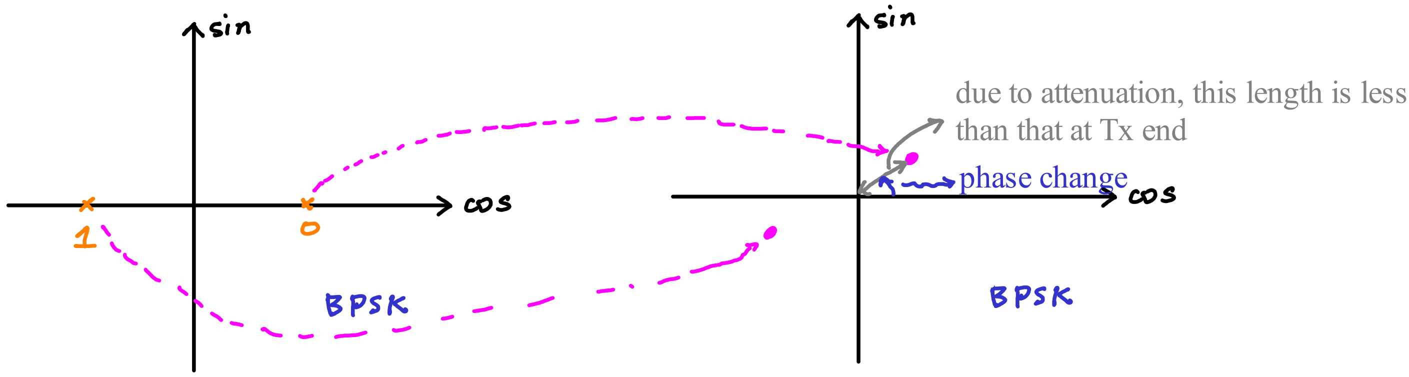
We need to modulate the carrier signal.

$A\cos(2\pi f_0 t + \phi)$ change one of these



The Tx sends s(t) and the Rx gets r(t), which contains both the signa and noise.

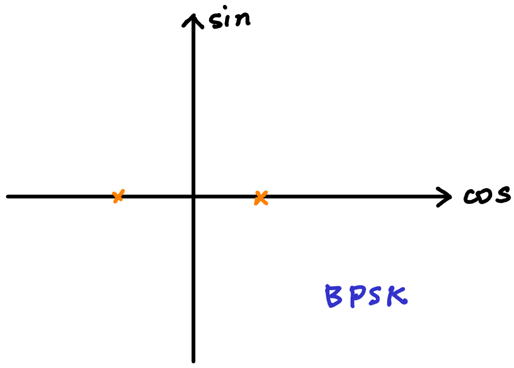
In practice, the signal goes through a channel to the Rx. The Tx would have either send 1 or 0. But at the reciever end, attenuation and/or phase change can occur.



Transmitter end

Reciever end

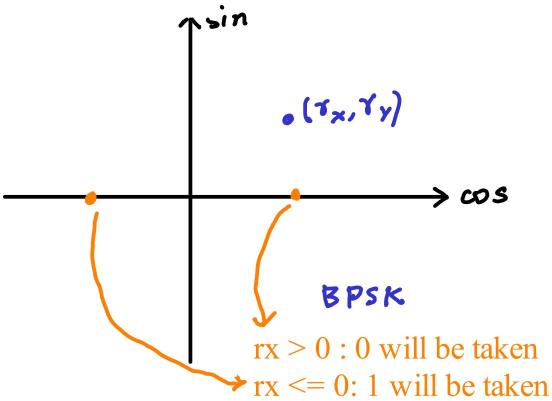
Rx corrects for the phase chenge, by rotating the vector back by \phi degrees



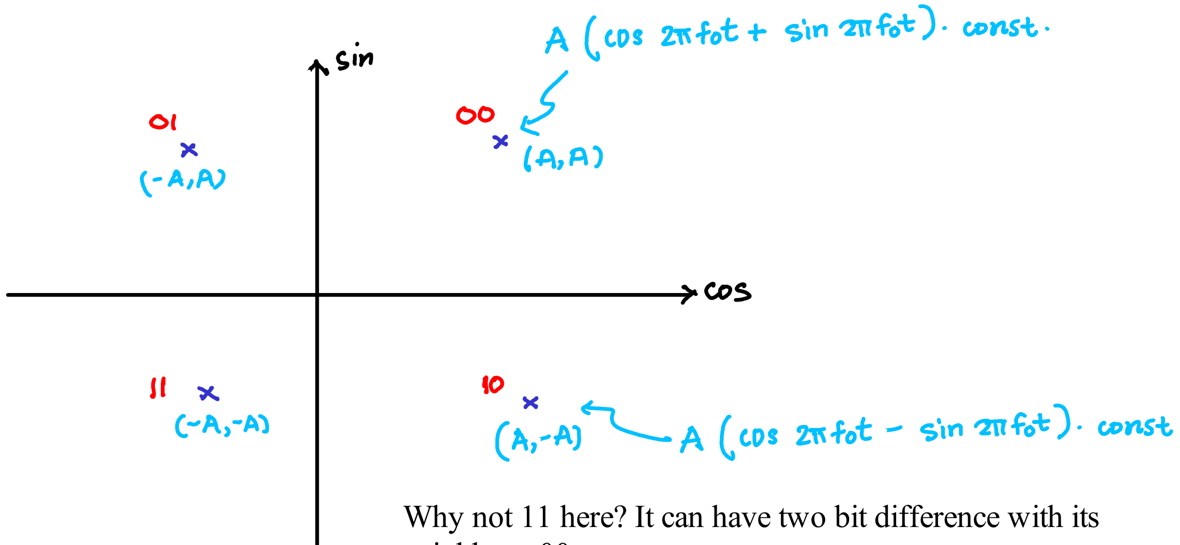
$$r(t) = (r_x, r_y)$$

$$r_x = \langle r(t), e_x(t)$$

$$r_y = \langle r(t), e_y(t) \rangle$$

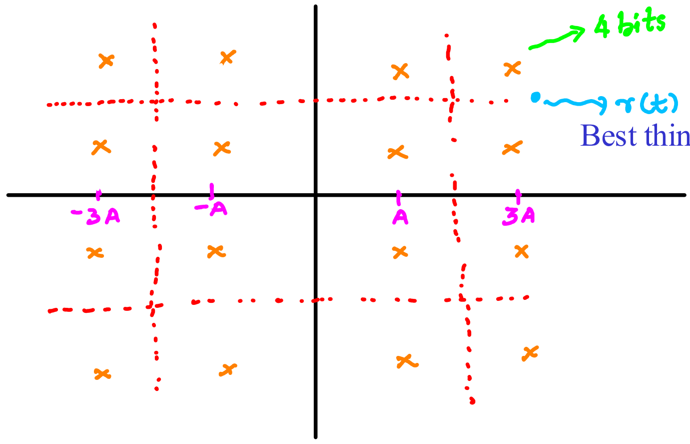


In QPSK, we send two bits of information at a time



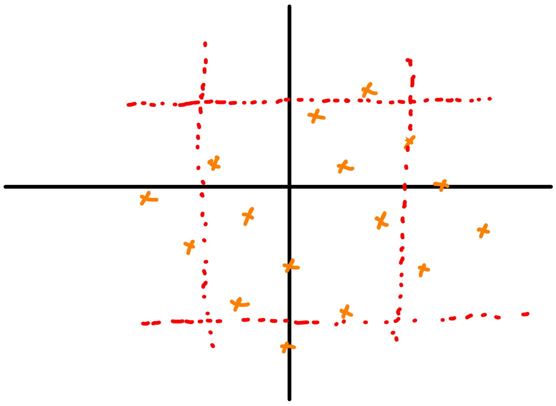
Why not 11 here? It can have two bit difference with its neighbour 00.

In QAM-16, we send 4 bits at a time, leading to 16 constellation points.

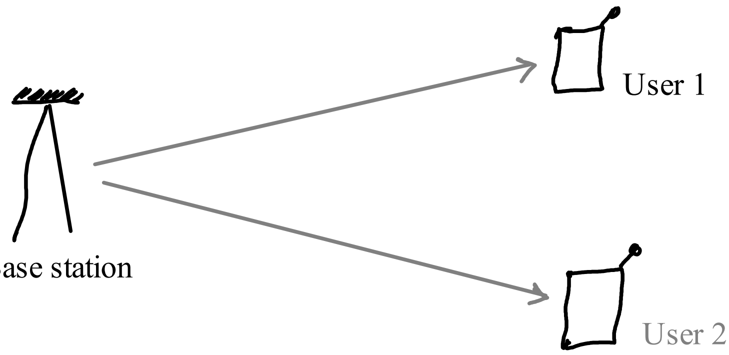


- what should we choose?
Best thing is to assign to nearest constellation point

Instead of clear straight version, there can be a rotated version:



This can result in wrong estimation



Q: Which modulation to be use?
BPSK,QPSK,QAM-16, QAM-64, QAM-256??

Ideally we would like to have higher modulation rates, i.e use the one which sends more buts at once. But we would be forced to have the same average power in any case, i.e we have the same average power available for both BPSK, and QAM-16. But, the bit error will be more in BPSK than in QAM-16.

So, given Pr(bit error) < eps, see what the attenuation is. If there is less attenuation go for QAM-16, otherwise go for BPSK.

DATA LINKED LAYER

Deals with Framing, Error detection, medium access, etc.

Consider a data send as below. We know that there is a 32 bit IP address here, where to look for these 32 bits?

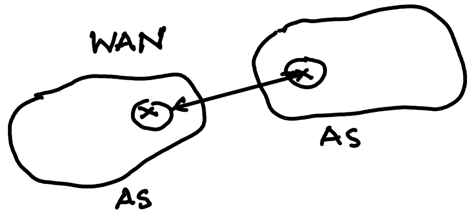
1 0 1 1 0 1 0 1 1 0 1 1 0 1 0

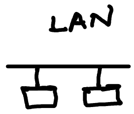
Framing: when does a frame begin and when does it end?

DATA is usually send as Frames: 1 chunk of meaningful data.

HDLC: High-level Data Link Control

Refer Peterson & Davie





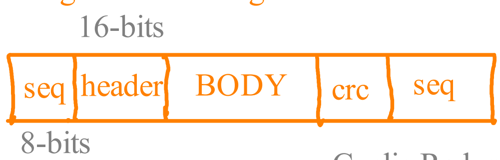
WAN: Wide Area Network
LAN: Local Area Network

Synchronous Mode HDLC:

We need to send continuously to sync the clock or do some channel estimation.
What to send? This is not actually useful info

HDLC has a default sequence for this silent region
DEFAULT SEQ: 0 1 1 1 1 1 1 0

When we move from a silent region to a frame, how to indicate that we are shifting to a useful segment now?



Cyclic Redundancy Check

Bit Stuffing:
If the body contains a sequence same as that the header, we add a few bits

At Transmitter:

END OF FRAM

0 0 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 1 .... 0 0 1 1 1 1 1 0

0 0 0

Rule : add a 0 after 5 consective 1s

At Reciever:
It is sure there can't be 6 consecutive 1s due to bit-stuffing.

Can it be sure that the zero after any 5 ones is originally present, or was it added as a part of bit-stuffing?

So, If we see 5 consective ones with 0 after it, Rx assume its due to bit stuffing.

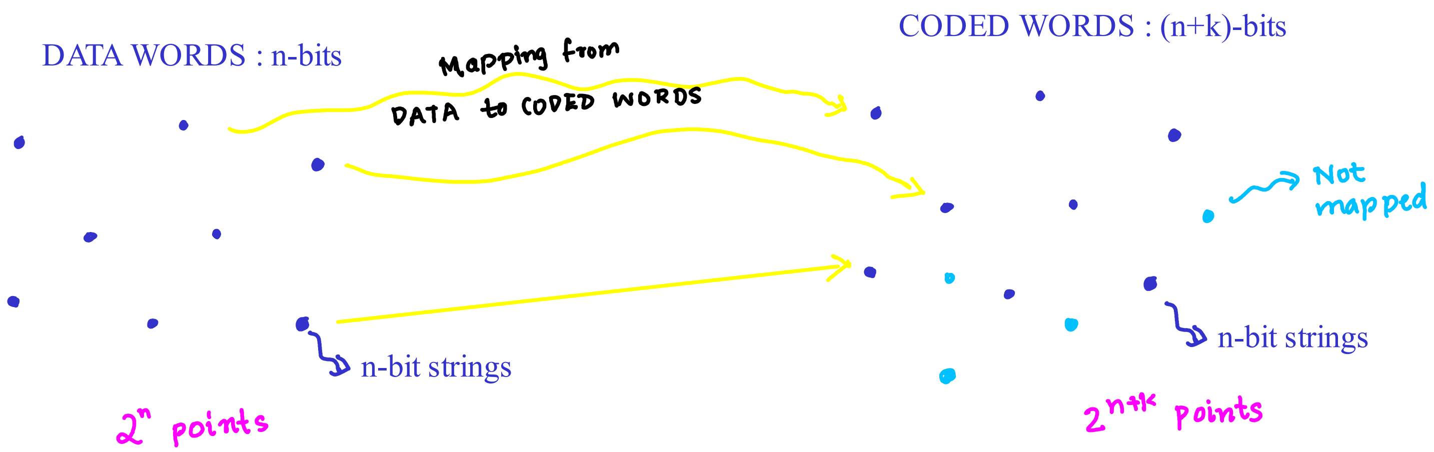
0 1 1 1 1 1 | 0 -- Assume due to bit stuffing
| 1 0 -- Seq at the end of the frame
| 1 1 -- Error has occured so discard the frame

Cyclic Redundancy Check

These are additional bits added at the end, used to detect errors. This is just one among many ways of error detection

Our requirements:

1. Want to be able to easily detect a large type of bit errors (single bit, a burst of consecutive errors, etc.)
2. Creation and verification of CRC has to be computationally efficient
3. For any given k (k represents the number of bits in CRC), CRC should be computable for any n (n is the number of bits in remaining part of the message)



If we recieve something which is not mapped, we can be sure of bit errors.

Here, we convert the data word to coded word, and transmit it.

But it is also possible that the recieved coded word has too many bit error that it is also a mapped one.

Hamming Distance

Given 2 codewords: a1 a2 a3 .... am
b1 b2 b3 .... bm

Hamming distance is the number of bits in which they differ.

Minimum Hamming distance of a code: Minimum of all Hamming distance between all pairs of code-words

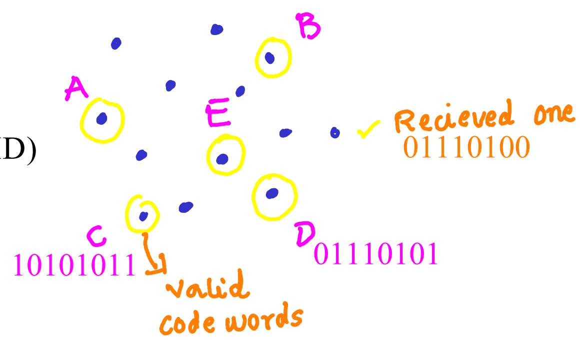
We desire to wish codes with high minimum Hamming distance.

Error Detection:

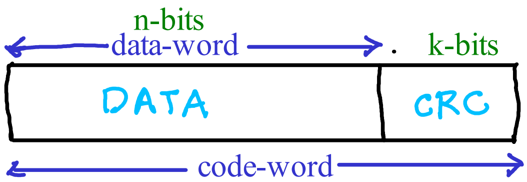
If min HD of a coding scheme is N, then we can always detect (N-1) or less bit erros.
Here, we detect that there were errors but may not know the position of errors.

Error Correction:

Suppose the minimum HD is (2t +1) and the number of bit errors is 't' or less, then by mapping the recieved code word to the nearest valid code word (distance is HD) corrects all errors



Choose a coding scheme with enough HD.
If t = 2, choose HD with min 5.



This results in a map from a data-word space with 2^n points to codeword space with 2^{n+k} points.

In codeword space we need high minimum Hamming distance, so that with an error we doesn't go to a valid code-word

BUILDING CRCS
GALOIS FEILDS:

$+, -$: use XOR $*$: as usual ($0 \times 1 = 0$, $1 \times 1 = 1$, $0 \times 0 = 0$, $1 \times 0 = 0$)

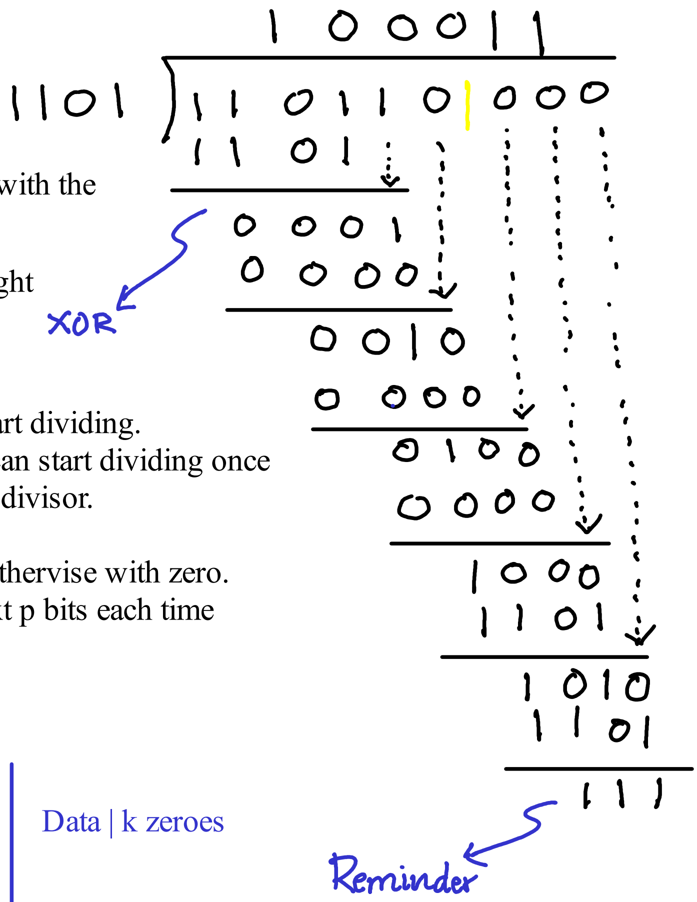
Long divison in binary world

Example:

Data : 1 1 0 1 1 0

k = 3

Divisor/Generator: 1 1 0 1 (k+1 bits)



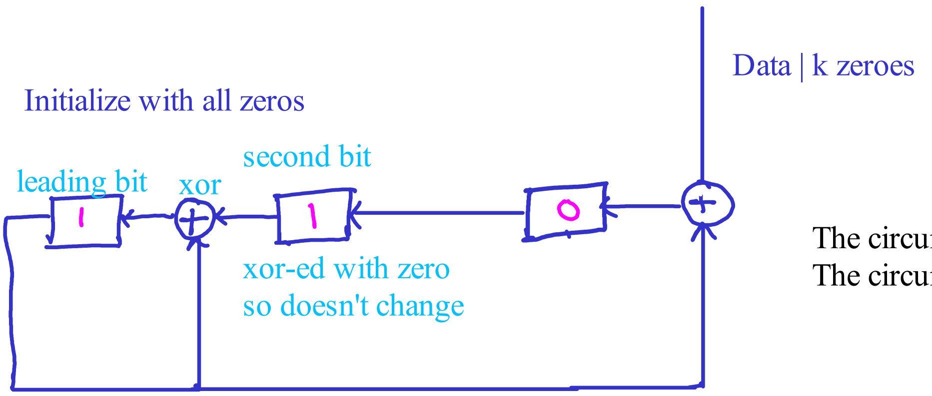
We divide the data word appended by k bits of zero with the divisor

The reminder after the long divison as seen on the right is the CRC, which we transmit after the data word

Observe that we don't need all the bits at once to start dividing. Even if the dataword is thousands of bits long we can start dividing once we get the first 'p' bits, where 'p' is no of bits in the divisor.

Also if the leading bit is zero we xor with divisor otherwise with zero. So we just need to keep track of leading bit and next p bits each time

This can be efficiently done using a shift register.



The circuit depends on the divisor we choose. The circuit on left is with divisor 1 1 0 1

How do we decide if the codeword is valid at the reciever?

At the reciever we get, $\underbrace{\text{Data'} \mid \text{CRC'}}_{\text{code-word}}$ ---> can be corrupted!

We can use the following two methods to see if there are errors.

Is the reminder 000? ← [Same Circuit] ← Data' | CRC

OR

If the answer is yes: NO ERROR

Is the reminder CRC? ← [Same Circuit] ← Data' | 000

Polynomial representation of bit strings:

We represent each bit correspond to a power of x.

$$\begin{array}{cccc} 1 & 1 & 0 & 1 \\ x^3 & x^2 & x & x^0 \end{array}$$

$$\downarrow$$

$$1\cdot x^3 + 1\cdot x^2 + 0\cdot x + 1\cdot x^0$$

$$\Longrightarrow \quad x^3 + x^2 + 1 = c(x)$$

Divisor/ Generetor polynomial

Multiplication:

$$c(x)(1+x) = (x^3+x^2+1)(1+x)$$
$$= x^3 + x^2 + 1 + x^4 + x^3 + x$$
$$= x^4 + x^2 + x + 1$$

Simulated using bit operations

$$\begin{array}{r} 1101 \\ 11 \\ \hline 1101 \\ 1101 \\ \hline 10111 \end{array}$$

What does it means to have errors?

Suppose we transimitted some code-word like
110110 111 = P(x)
Say two bits get flipped like. This can be done by adding following word
000001 001 = E(x) $= x^3 + 1$

Then, what we recieved is P(x) + E(x).

Q: Is ((P(x) + E(x))/C(x)) == 0??
If not zero, then detected errors.
Because for a valid code wordwe should get a zero

Types of Errors:

$$x^{n+k-1} \quad \underbrace{\qquad\qquad}_{n}\underbrace{\qquad}_{k} \quad x^0$$

1. Single bit error:
   E(x) = x^i ; for some
   $i \in \{0,1, ..., n+k-1\}$

$$\frac{P(x)+E(x)}{C(x)} = \cancel{\frac{P(x)}{C(x)}}^{\,0} + \frac{E(x)}{C(x)}$$

→ we don't want this to be 0.

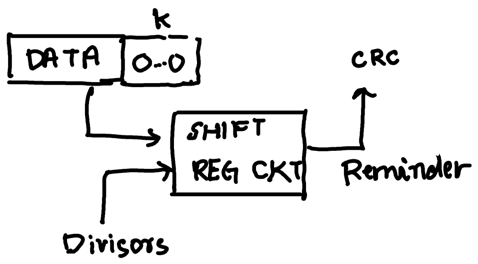
If $C(x) = x^k + \underbrace{\ldots\ldots}_{\text{anything}} + 1$,

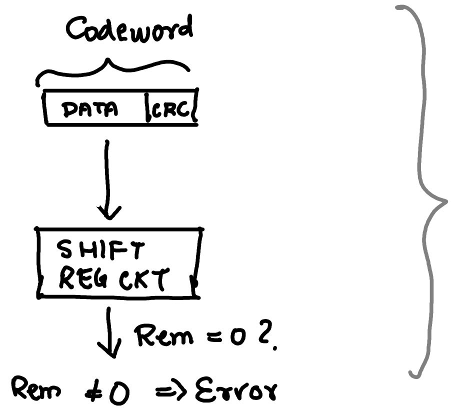
$C(x)\,[x^m + \ldots + x^q]$ will have this form $x^{m+k} + \ldots\ x^{m+q}$

$m > q$

highest power x^{m+k}
lowest power x^{m+q}

So, it cannot divide
E(x) = x^i





CRC Summary

Min HD = d, then definitely detect all errors less than d bits. May detect errors of greater than equal to d bits

RECALL: Polynomial Arithmetic $1101 \longrightarrow x^3 + x^2 + 1$

Use Galois Feild, GF(2): addition/subtraction as XOR

Ex: DATA: 110110 - $x^5 + x^4 + x^2 + x$

$1101 \overline{)110110000} \longrightarrow \dfrac{x^3(x^5 + x^4 + x^2 + x)}{x^3 + x^2 + 1} \xrightarrow{\text{Rem}} x^2 + x + 1$ (111)

$P(x) = x^3(x^5 + x^4 + x^2 + x) + x^2 + x + 1$
codeword (110110|111)

If A(x) is divisible by B(x) it means that A(x) = B(x).D(x)

P(x) -> codeword
C(x) ->Divisor/Generator
E(x) -> Error polynomial:
Represents positions of bit error

The polynomial recieved will be P(x) + E(X)

$$\frac{P(x) + E(x)}{C(x)} = \frac{P(x)}{C(x)} + \frac{E(x)}{C(x)}$$

For eg:
P(x) = 110110 111
E(x) = $x^6 + x^2$
Recieved polynomial: P(x) + E(x)
= 111111 111

P(x) chosen so that C(x) divides it.

Does C(x) divide E(x)? If no, we can be sure that there is error.

Types of Errors:

1. Single bit errors $E(x) = x^i$ for some $i \in \{0, 1, \dots, n+k-1\}$

Suppose $C(x) = x^k + \underbrace{\dots}_{\text{anything}} + 1$ then C(x) does not divide E(x)

Does there exist a D(x) such that $C(x)D(x) = x^i$ ← Is not possible!

Let $D(x) = x^m + \dots + x^q \quad , m \geq q$

$$C(x)\,D(x) = (x^k + \dots + 1)(x^m + \dots\ x^q) = \overbrace{x^{k+m} + \dots\ x^q}^{\text{atleast two terms}}$$

$(k+m > q) \qquad (k \geq 0)$

2. Two bit error:

$$E(x) = x^j + x^i \quad (j > i)$$
$$= x^i\,(x^{j-i} + 1)$$

Suppose C(x) is of the form $x^k + \dots + 1$, does C(x) divide E(x) ?

$$\frac{E(x)}{C(x)} = \frac{x^j\,(x^{j-i}+1)}{x^k + \dots + 1}$$

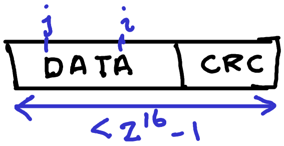
We know that x^i terms not cancelled out,

So the Qn is Does C(x) divide $x^{j-i} + 1$

Defn. (Order of a polynomial): The smallest 'r' such that the polynomial (say C(x)) divides $x^r + 1$ is called its ORDER

So we can find C(x) such that the order is very high, generally $2^k - 1$

Ex: k = 16; $C(x) = x^{16} + \dots + 1$, then find a C(x) such that it will not divide any $x^p + 1$ for $p < 2^{16} - 1$



→ We can detect all two bit errors

3. Odd number of Errors

$$E(x) = \underbrace{x^j + x^i + \dots}_{\text{odd no. of terms}}$$

ie, (1+x) is a factor

If C(x) = (1+x) (..)(.....), then all odd number errors are detected

If C(x) has even number of terms also, we can detect odd # errors.

HDLC CRC: $x^{16} + x^{15} + x^2 + 1 = x^{15}(1+x) + (1+x)(1+x)$

$$= (x^{15}+1)(1+x)$$

In GF(2) $x^2 + 1 = (x+1)(x+1)$

Show that E(x) in this case cannot be represented as E(x) = C(x)D(x) for any D(x).

Proof. E(x) has an odd number of terms.

Then, E(1) = 1 added odd number of times = 1.

But, C(1) = 1 + 1 = 0. (has a factor of 1+x) So, C(1)D(1) = 0 => E(1) != C(x) D(x).

If C(x) has even number of terms, C(1) = 1 added even number of times = 0. Again the same.

Thus there does not exist any such D(x)

4. Burst of errors

$$\text{Data|CRC} = 1\,0\,1 \ldots \underbrace{1\,1\,0\,1}_{} \ldots 1\,1\,1$$

Interference

i

consecutive errors.

$i+l-1$

$$E(x) = x^{i+l-1} + x^{i+l-2} + \ldots + x^i$$
$$= x^i(x^{l-1} + \ldots + 1)$$

$$\frac{E(x)}{C(x)} = \frac{x^i(x^{l-1} + \ldots 1)}{(x^k + \ldots 1)}$$

Recall that x^i does not cancel any factor in C(x).

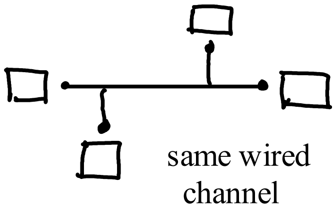
If l-1 < k then all factors of C(x) cannot be cancelled

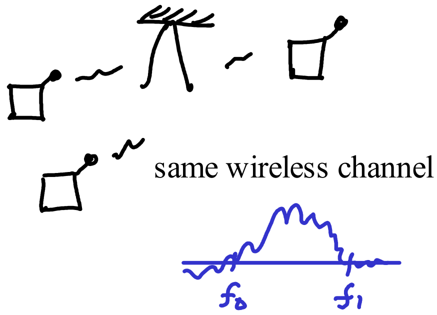
So, C(x) of the form $x^k + \ldots + 1$ detects all bursts of errors of length l such that l-1 < k

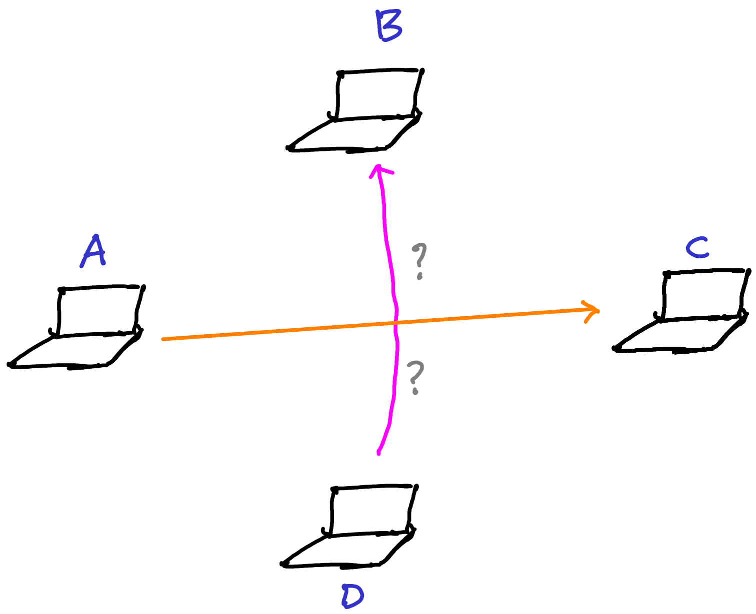
Ethernet uses a CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

MEDIUM ACCESS

How to communicate with each other such that the signals does not interfere?

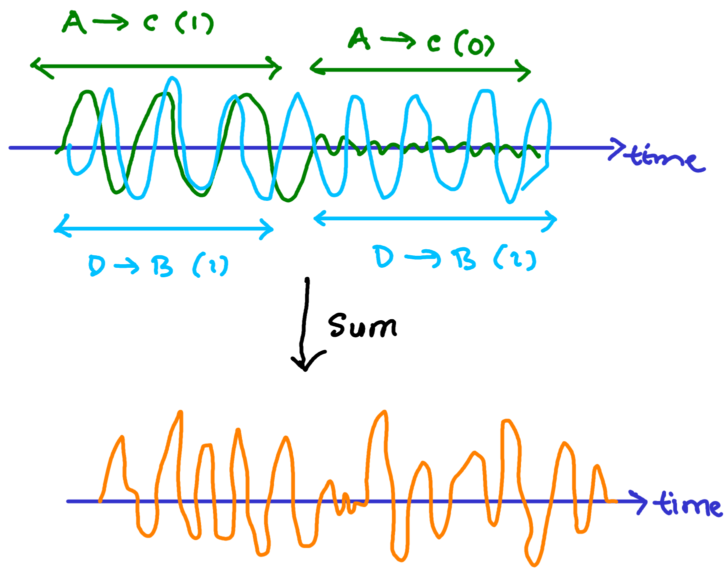






Say in the system to the left we use audio signals for communication. A tries to communicate with C and at the same time D communicated with B.

C revieves two signals - from A and D.

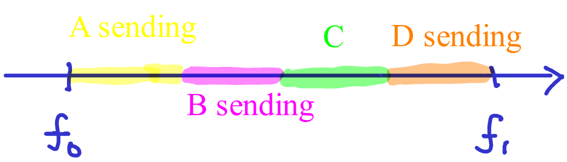


Q: What is being sent? By whom? For whom?

need some identifiers

Identifiers

One way is to use different parts of the frequency range for each communication.



Drawbacks of this:

1. What if new devices E,F want to join? We might need to revise the protocol
2. Performance issue: Throughput of A is 1/4 th compared to the case where A uses the full band.

So, another method is to introduce some sort of addressing scheme.

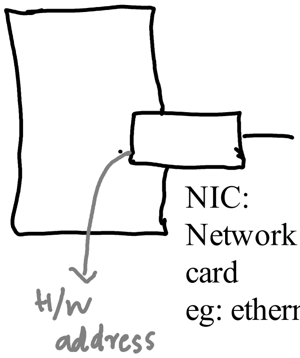
Assign bits to ecah node. But how many bits?
- use enough bits to accomodate devices which may join in the future

How to assign address? Should we do it manually, OR have a protocol to assign then automatically.

Y2K problem:
There was no enough bits to accomodate for additional bits when year changes from 1999 to 2000, and potentially causing overflow.

IPv4 has 32-bits thus could address 2^32 devices.
People initially though so many devices would not be in use, but soon enough 32 bits were not enough.
NAT box was a hack to cope with this

At the MAC layer we use hardware addresses.
This can ensure that there is no clash between address.

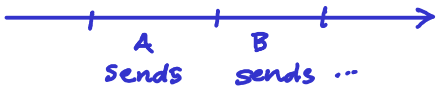


NIC:
Network interface card
eg: ethernet → 48 bits

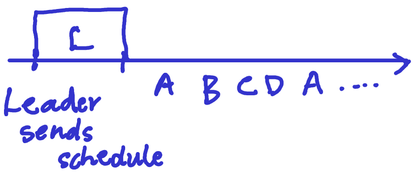
Interference

Can we use different frequency band? Then who uses which band?
We could also try to make sure that only two people communicate at a time, and no one else.

Idea: One node is made the leader. This can solve the issue of who can send first, and could decide who sends the upcoming messages, i.e it sends some sort of schedule.



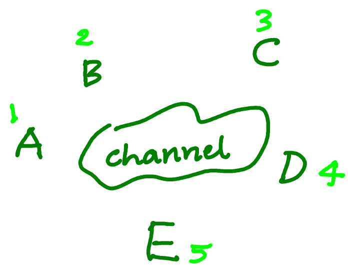
What will someone wants to join? There should be a means by which the newcomer can signal the leader.

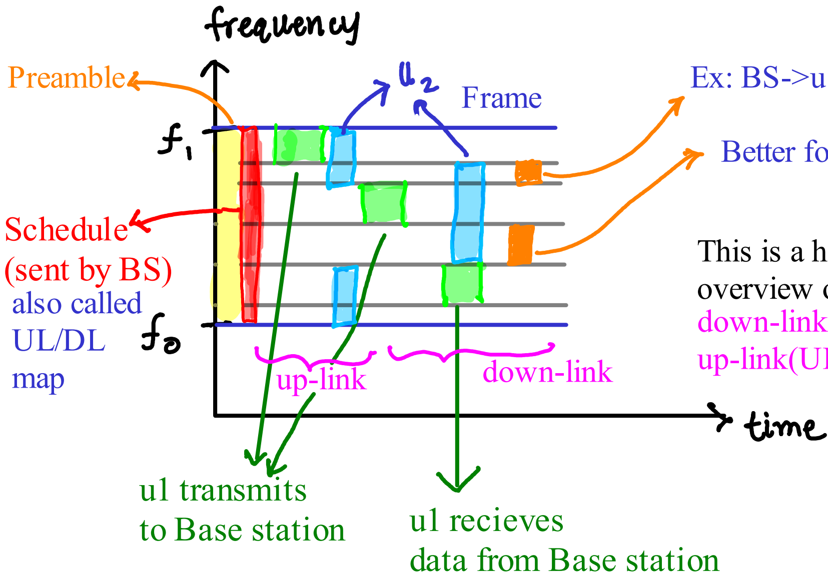


Another idea: Token based - there is no leader as such, but device with the token is allowed to transmit. Say A has the token in the beginning. It will transmit the message and then will be ready to release the token - Token release message. This also says whom to give the token to.
There will be some specific order among them. So if one gives up token, it can be given to next in order.

RECALL: the options available to avoid interference:
1. Central co-ordinator
2. No Central co-ordinator

A base station acts as central co-ordinator in most cases





Ex: BS->u1 (less attenuation) and BS->u2 (more attenuation)

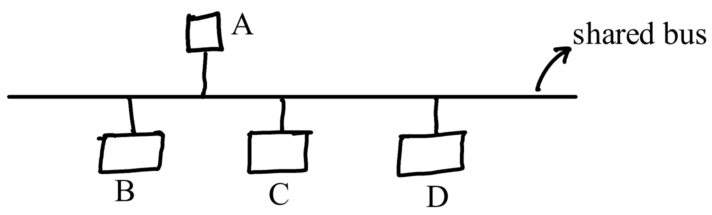
Better for u2 than u1 (less attenuation in u2)

This is a hogh level overview of how 4G works
down-link (DL): time when users recieves
up-link(UL): time when users send

Issues with central Co-ordinator

1. Single point of failure
2. May not be possible to have a central co-ordinator:
   eg: wireless with unlicensed band

What if we have no central co-ordinator?

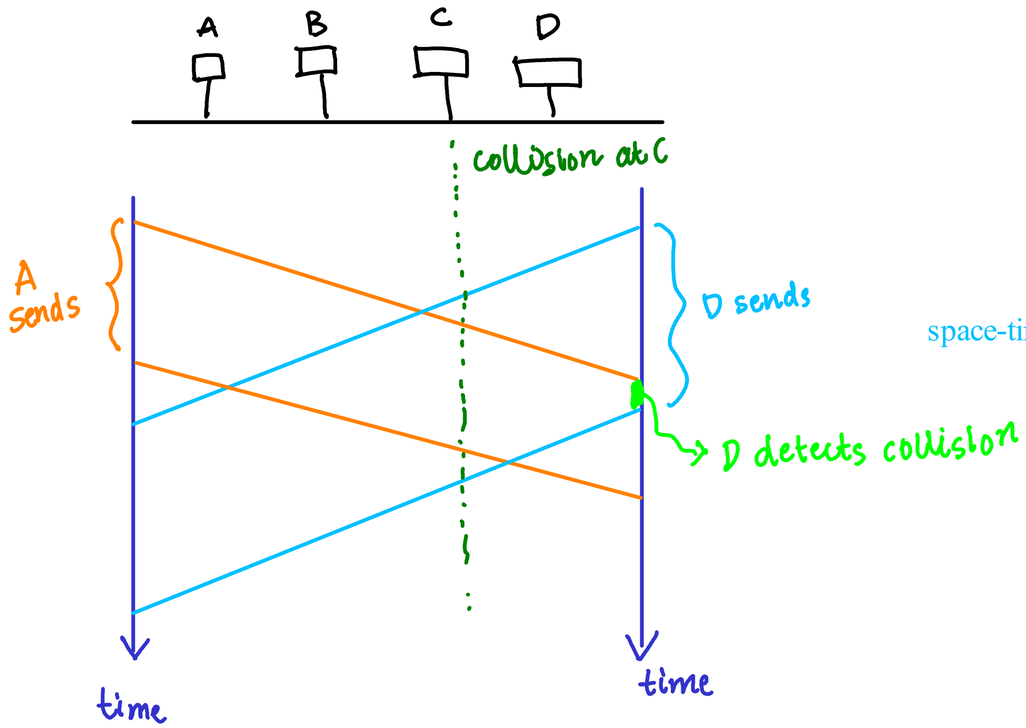


Requirements:
1. plug and play: we need to connect and disconnect to the bus
2. no central co-ordinator

Assume that most of the time only one system needs to transmit.
So they can right away send the message. But when multiple systems sends, there will be collisions (rarely).

Collisions: multiple nodes transmit simultanoeously. signals add up at the reciever and neither signal can be deciphered.

If collision occurs, stop transmitting

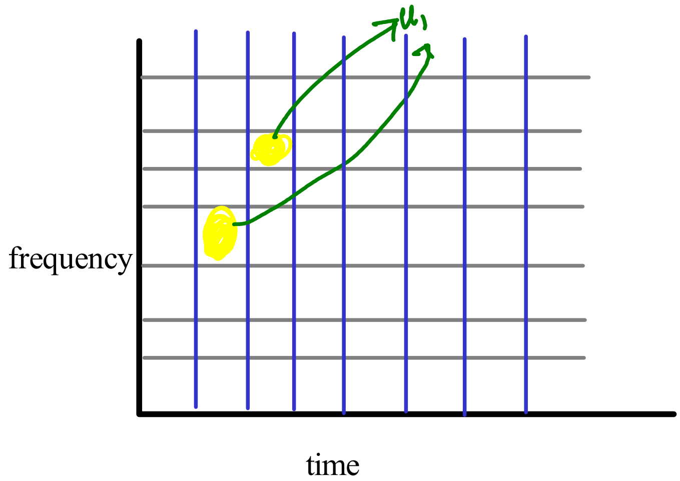


space-time diagram

TDMA: Time division multiple access
There is a central co-ordinator which decides to schedule

FDMA: Frequency division multiple access
All talks to a base station, which assigns different frequencies to each user

Combination of TDMA + FDMA: OFDMA(orthogonal....)



different tiles for different users

Token Passing: Only token holder can transmit -- Pass on the token every so often

collision detection

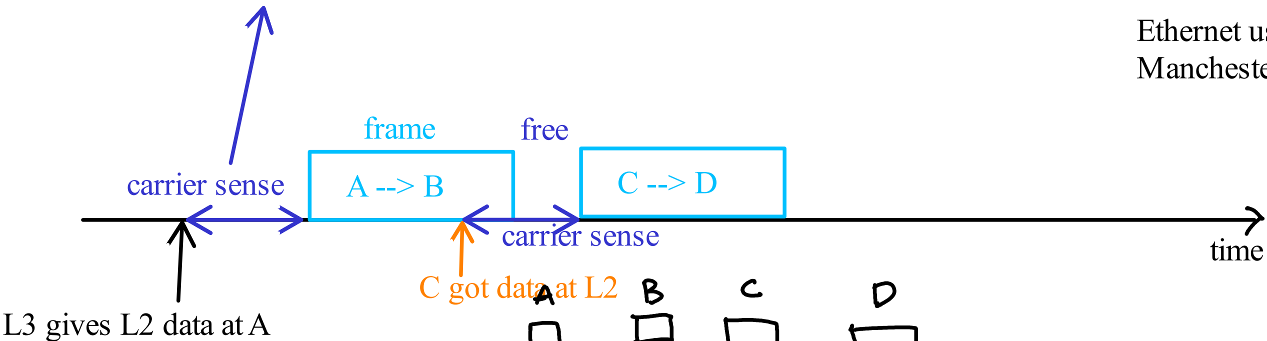
ETHERNET - CSMA CD No centralized co-ordinator

Carrier sense

How to detect collision ? If energy is higher that usual

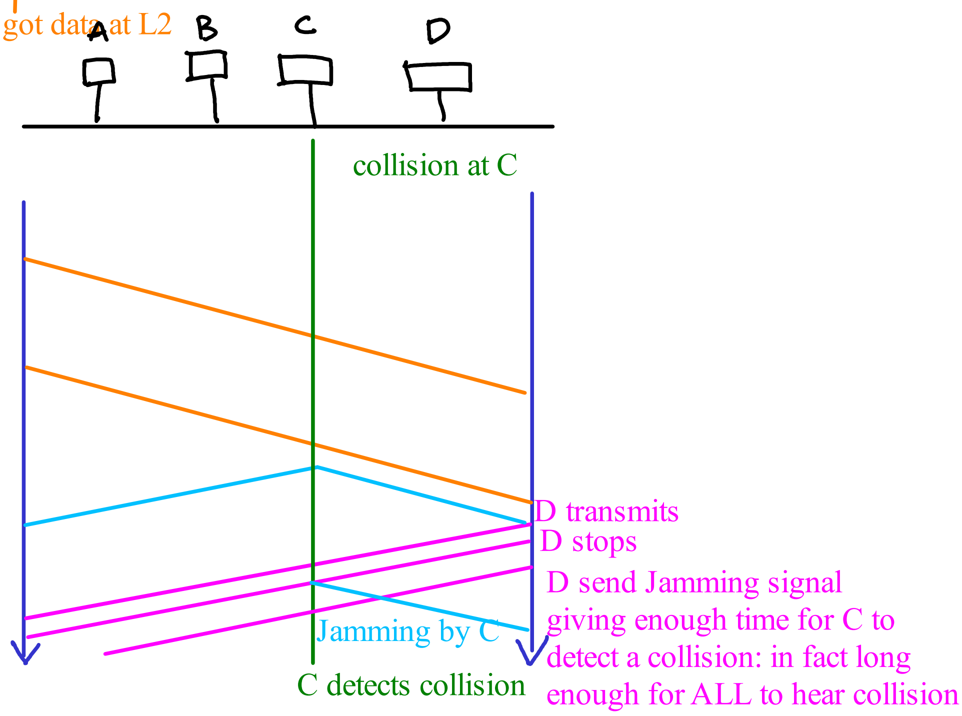
Li : Layer i

Ethernet uses Manchester encoding



L2 Frame

Header	L3	CRC

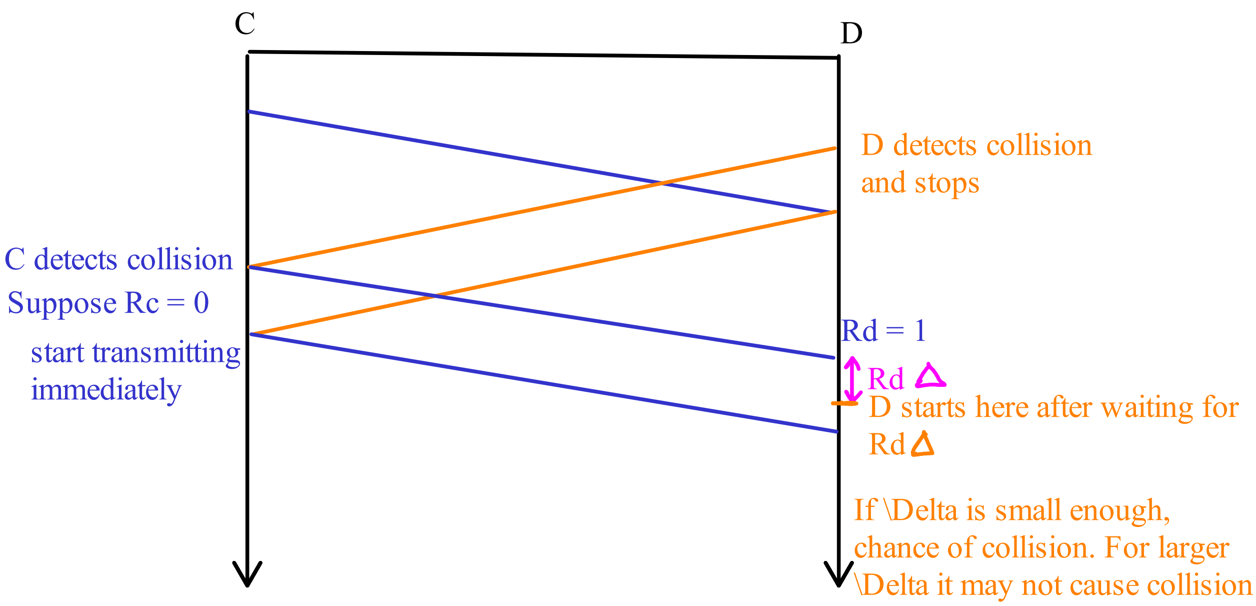


Random wait time: Each nodes throws random number
R - chosen uniformly from some range.

Each user required to wait for RΔ before trying to send again

Idea: If Rc < Rd, then C will transmit next without colliding with D.

What is Δ?



\Delta needs to be large enough
to hear C's frame before

Can show that if \Delta > round trip time, then D will not transmit before C

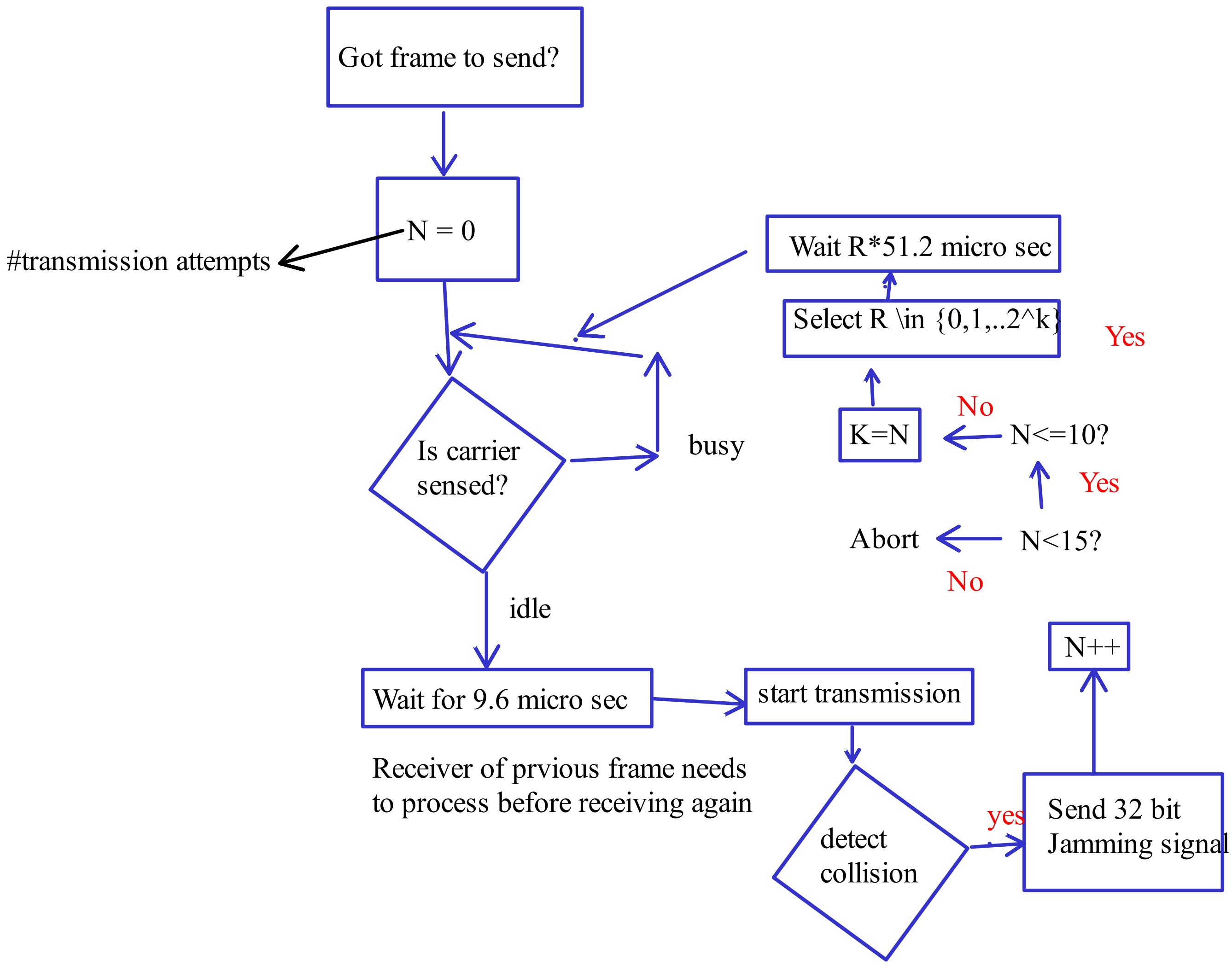
RTT(round trip time):
2*(time for signal to go from one
end of the network to the other)

Assumptions used in Ethernet:
- Max cable length : 2500m
- Speed of light : 2*10^8 m/s (~c)

So, one way delay (OWD) = 2500/(2*10^8) = 12.5 micro sec
Thus, RTT = 25 micro sec

C and D can have repeaters (max 4), which amplified in the path between C and D.
Including this ethernet assumes a RTT of 51.2 micro sec.

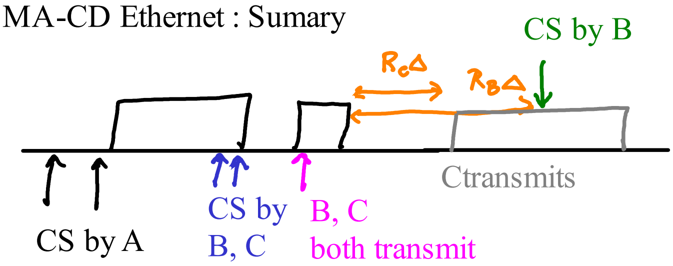
Flow chart (Run at every node)



Exponential back off:
- don't know how many colliding

Suppose M colliding.

CSMA-CD Ethernet : Sumary

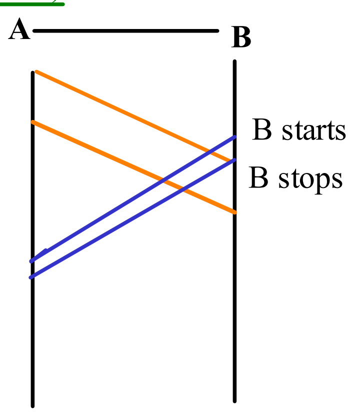


$R \in \{0, 1, 2, \ldots 2^k\}$

k increments upto 10 for every collision for same frame

Limits on frame size (Ethernet 802.3)

1. Minimum Frame size:



Want A to detect collision while transmitting

$\Delta =$ worst case RTT

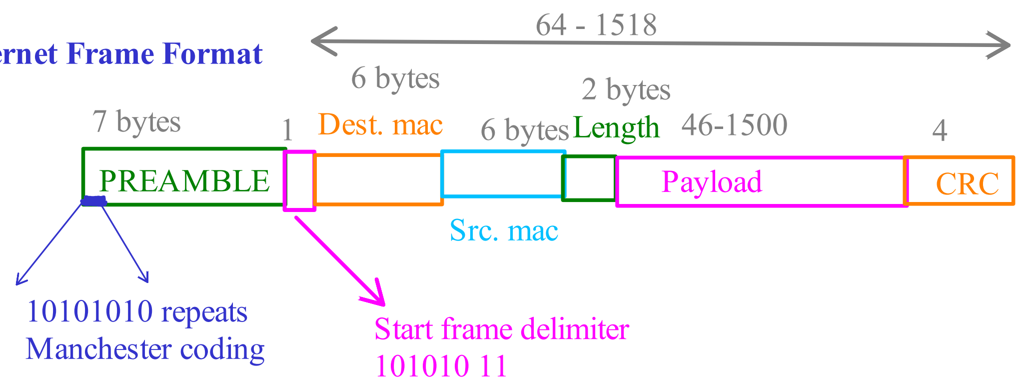
Original standard 10 Mbps
For a frame size of 54 bytes (512 bits),
time to ransmit at 10 Mbps = 512/10^7 = 51.2 micro sec
(Seen before!)

2. Maximum Frame size

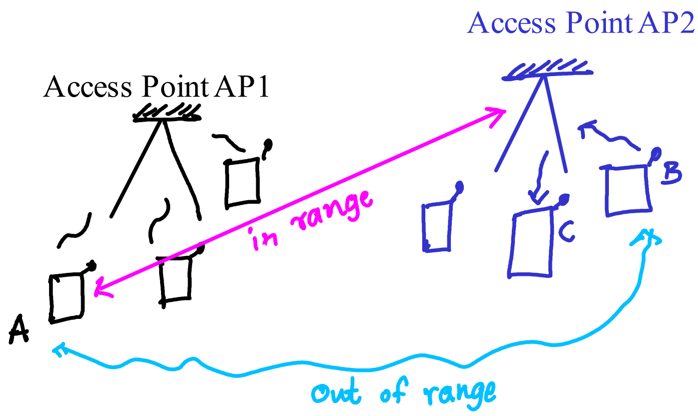
Why can't we send (very) large frames all together?
- bit errors :- high probability of atleast one bit error
- don't want a single user to monopolise the channel for very long

In Ethernet, the max frame size is 1518 bytes

Ethernet Frame Format



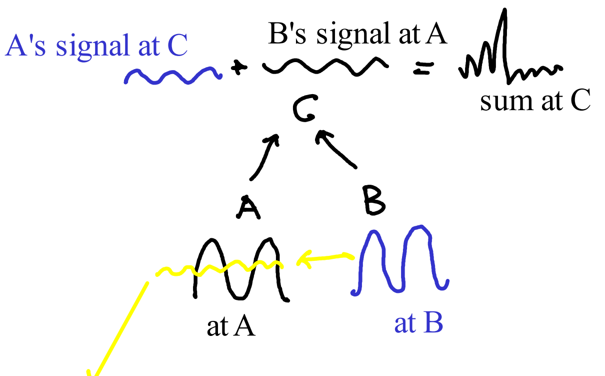
WIFI (IEEE 802.11)



Will CSMA- CD work here?

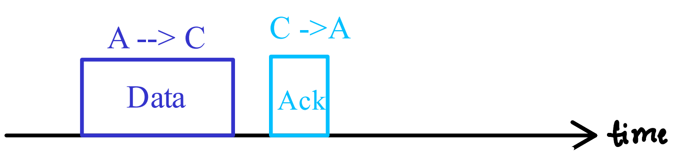
No. (Hidden Terminal Problem):
B sends message to C, but A cannot sense B's signal. So, there is a possibility of collision. So carrier sense does not work.

Collison detection also does not work. See below
because when one speaks, it can't hear anyone else



while transmitting A cannot hear anybody else

Solution (for collison detection):
Reciever sends an acknowledgement



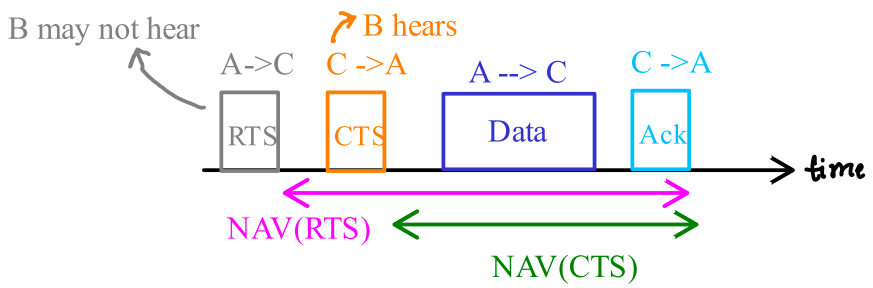
If A doesn't receive acknowledgement, it assumes there was some collision and thus retransmits

signal decay with distance d ~ $1/d^{\alpha}$

$2 < \alpha < 5$

How to handle Hidden Terminal?

First send a REQUEST TO SEND(RTS), then reciever sends a CLEAR TO SEND(CTS)



NAV: network allocation vector

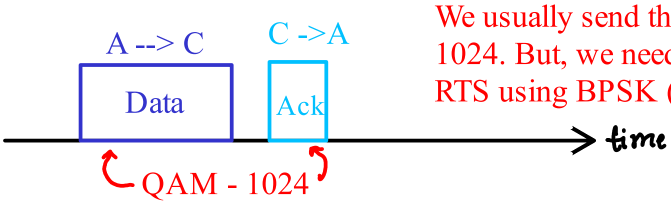
Rule: all hearing RTS and CTS (other than A and C) must remain silent for the NAV durations.
(Virtual Carrier sensing due to B)

What if A sends RTS but does not get CTS?
- Here, A assumes there was collision and it re-transmits

What if RTS,CTS,DATA are sent but ACK didn't reach A?
- Again A assumes there was collision and it re-transmits

The protocol in WiFi is called CSMS-CA
collision avoidance

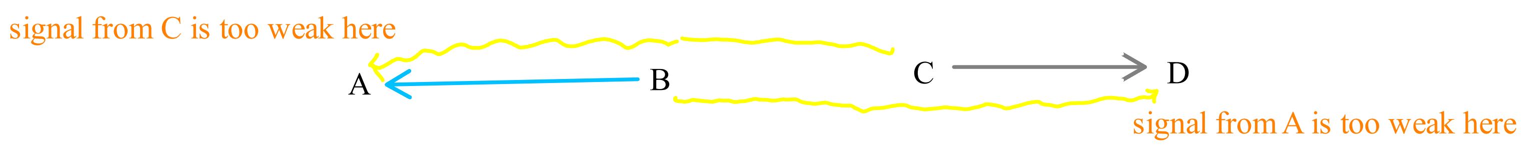
Here, we adopt Virtual Carrier sensing - using RTS,CTS. But these are optional and can be diabled.
This is disabled because it is an overhead and can affect performance. In practice, this occupied much time



We usually send the message at high modulation rate, like QAM-1024. But, we need the RTS to reach long distance so we send RTS using BPSK (because we need high SNR over long distance)

Exposed Terminal Problem:

Each node can only hear its neighbouring nodes.



A can only hear B, B hears A and C, D only hears C

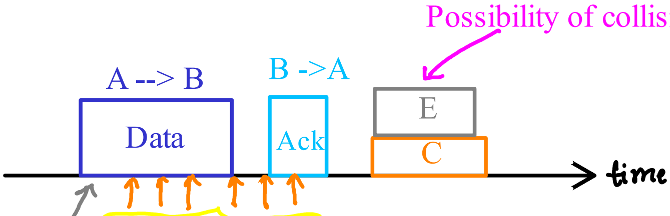
Q: Is it possible to transmit from B to A and from C to D together, in theory?

Case(i): RTS/CTS is enabled: If B starts first, C remains silent due to RTS.
Case(ii): RTS/CTS is disabled: B starts, C does Carrier sense. So C remains silent

This is because C is EXPOSED to B: exposed terminal problem

A C
B D
E

Possibility of collision

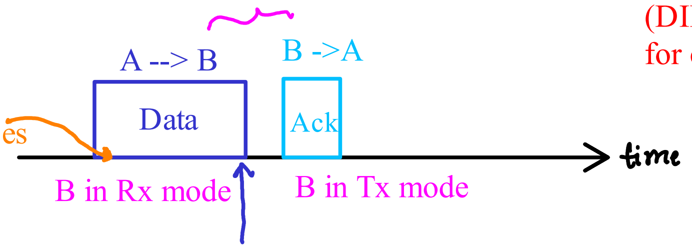


So, - we don't want E and C to start together.
- we don't want C and E to semd the data between A-> B and B->A

E also has data to send

C gets data
to send
from higher layer
- but it carrier senses

Short Inter Frame Spacing (SIFS)



Rule: C has to wait atleast DIFS
(DIFS = SIFS + (2*slot-duration))
for channel to be free before trying to transmit

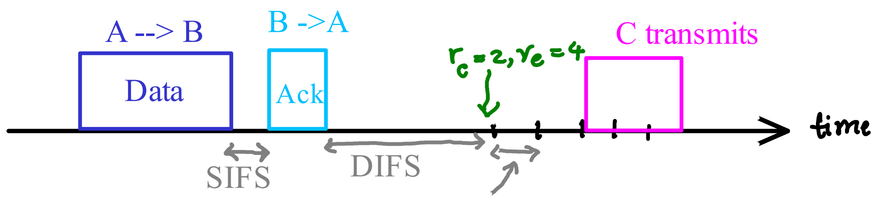
C carrier senses

B in Rx mode

B in Tx mode

B has to receive DATA
frame, check for errors,
read frame
B need to switch from Rx to Tx

C and E choose contention window random variable, $r_C \in \{0,1,\dots CW_{max,C}\}$
$r_E \in \{0,1,\dots CW_{max,E}\}$



slot (9 micro sec
in original standard)

Initially,
CWmax = 15.
For every collision,
CWmax = 2*CWmax+1

But it has an upper limit of 1023

Here, after DIFS, both keep decrementing its r value, and when one of them becomes zero, it starts transmitting. Say in this example rc= 2 and re = 4. After two slots C starts transmitting, while re value gets freezed. And it is decremented later on.

Fairness:
Suppose C has faced a couple of collisions, then, $r_C = \{0,1,\dots 63\}$
Let G be another node which has faced no collisions $r_G = \{0,1,\dots 15\}$

Typically, rc >> rg. So C has to wait for longer time on an average.

So, WiFi is not completely fair

IEEE 802.11 g --> 64-QAM, 3/4 coding rate (54 Mbps) - atmost 20 MHz
IEEE 802.11 n --> 256-QAM, 3/4 coding rate (866 MBps) - atmost 80 MHz
IEEE 802.11 ax ---> 1024 QAM, 3/4 coding rate

CDMA (Code Division Multiple Access)

used in 3G

BPSK signals

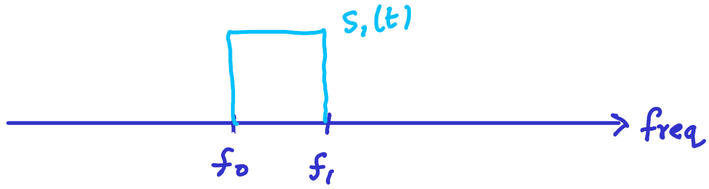
$P_1(t) \times S_1(t)$

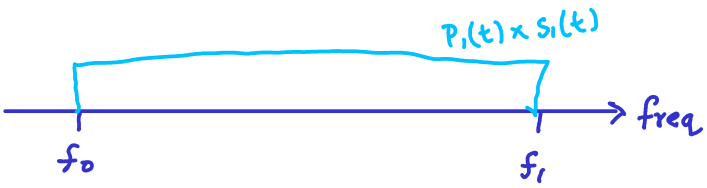
$S_2(t)$

$P_2(t)$

Spreading codes

We used different spreading codes for different users, and each spreading code is orthogonal to each other

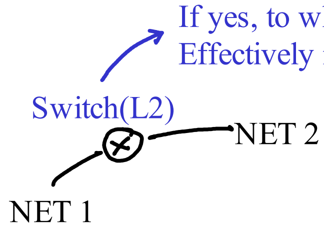




SWITCHING / ROUTING

Network
DLL } L2 L3 Switching/Routing
PHY

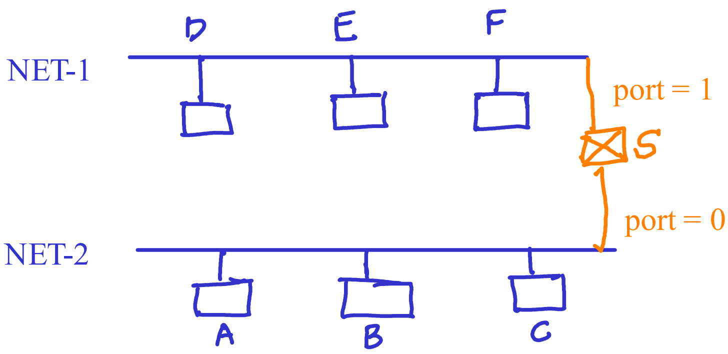
Asks if it should forward an incoming message?
If yes, to which output port?
Effectively isolates NET-1 from NET-2



NET 3

L3 switches forward messages based on IP addresses,
while L2 based on hardware addresses.

ETHERNET SWITCHING (L2)



We can't just connect them directly by ethernet
(length of ethernet has an upper limit,
also it can affect the thoroughput).

So, we connect them via a switch

One way is to use ports to identify the networks:
-A,B,C on port 0
-and D,E,F on port 1
Administrator can maintain a table mapping the device to its port.

If a new node comed in, the admin assigns a port. But, this has a lot of manual efforts - error prone.
Also, what if some one unplugs from 0 and plugs in 1. The value in table will be wrong

What Ethernet does?

DEST. MAC Address	Port	Expiry (sec)
A	0	10
B	0	15
D	1	17
C	0	22

for forwarding but we populate this using src address

These sitches also called BRIDGE

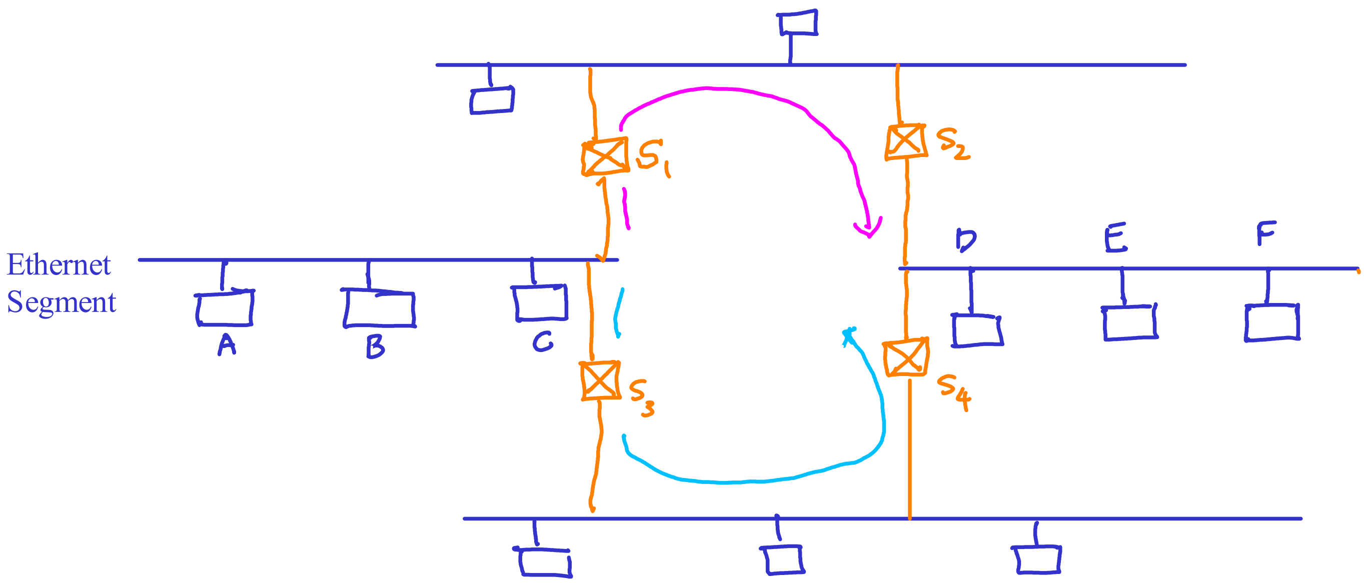
The switch listens for some message. Then
it can identify which port that node is.

(t=0) A initially sends a message to B. S recognises
A on port 0. But it send the message to port 1,
because it does not know where B is. (in case
of more than 2 ports, it sends it to every port.

(t=5) Now assume that B sends a message to C.
Then it realises that B is in port 0. Again
it forwards to port 1(don't know where C is)

(t=7) D-->A, It forwards because A is in port 0,
and D on port 1

(t=12) C-->A, The first entry in the table for A
gets expired (hence removed). So S can't find an
entry for A, so it forwards on port 1.



What if A sends message to someone who is not present?
Then S1, and S3 will forward, these frames reaches S2, and S4 respectively, S2 forwards to S4 and S4 to S2, and it keeps circultaing.
So two copies of the same frame keeps looping inside the network

LOOPS ARE BAD FOR NETWORKS!

One idea: Time To Live (TTL). Each frame has an initial Time assigned to it, each time it passes through a switch (each hop), its time is reduced by 1 -- Used in Layer 3.

What is actually done? RADIA PERLMAN (1985) introduced a protocol.
SPANNING TREE PROTOCOL

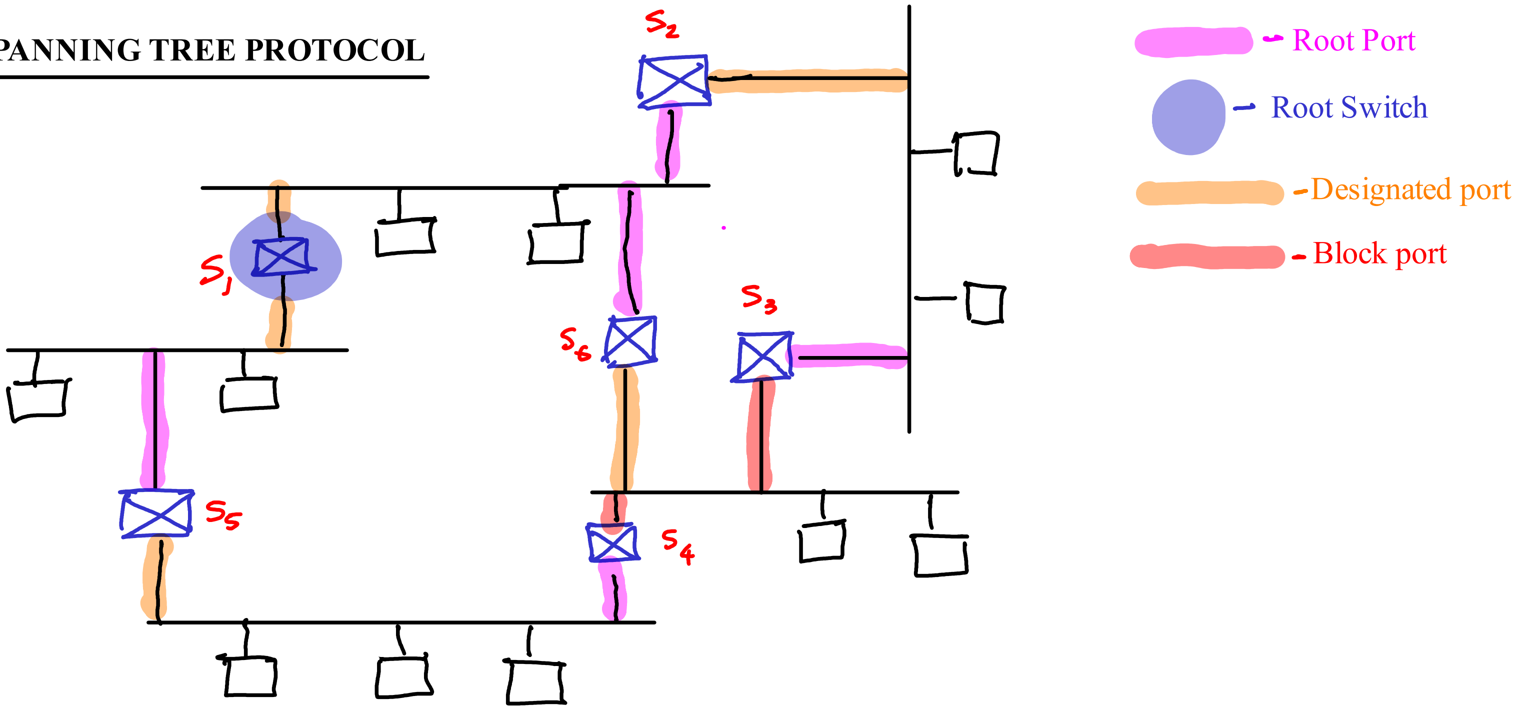
Since we need to break loops, we will disable some of the ports. We this get a spanning tree, and with enabled ports we can reach any node from any node.

STEPS:

1. Elect Root switch. (we will have id for each switch and choose using that (most probably the one with lowest id) but in this case the root may not be central- might be on a corner)
2. Every switch chooses a root port (RP) - the port closest to root
3. All switches connected to a segment, chooses one among all of their ports on this segment to be the designated port (DP) for that segment. (again the port closest to root is chosen as in 2nd step)
4. The the remaining port(s) which is neither RP/DP becomes a block port and is disabled

Note on expiry time in ethernet switching: it should not be too small or too large. Small expiry time would mean that the devices are removed too fast, while long time can lead to error if a device changes its port.
Inside the switch there would a CPU to forward the message accurately as per the requirement

SPANNING TREE PROTOCOL



Our idea is to disable some ports of switches for forwarding DATA FRAMES.

ROOT SWITCH: Each bridge has an ID. It 8 bits long. 2 bits are configurable and 6 bits are determined by MAC address (it is hardware address). For each switch we choose the smallest MAC address on all ports.
The one with lowest bridge ID is usually chosen to be the ROOT SWITCH.

Each bridge sends its neighbours a message of the form (Y,d,X), where
- Y is the bridge ID of who they think is the root
- d is the distance of that bridge to Y
- X is the bridge ID of that bridge.

Additionally, it may also send a Port ID

In the diagram above, let the order of bridge IDs be S1<S2<S3<S4<S5<S6.

At t=0, all say (Si,0,Si) to neighbours, i.e they says they are themselves the roots, because they haven't heard anyone.

After hearing this, they will update the lowest one they have heard. For example, S6 will realise (S1,1,S6), and so on. In the second round S2,S5,S6 realise that S1 is smallest. In third round S3, and S4 will realise that S1 is smallest.

Thus after some time this converges and ultimately S1 is selected as the root.

Now we need to form a tree with S1 as the root such that all the switches can reach the root bridge

ROOT PORT: Every port has a root port and this is kept active.This ensures that we can travel up the tree towards the root

How to choose the root port? One idea is to choose the port which is closest to the root bridge (close in terms of number of hops to reach the root). There can be cases where either port has same distance to root, then choose the one connected to a switch with lower bridge id (tie-breaking rule). What if both the port goes through same switch? In that case, we choose the one with lowest port id.

In the diagram above, the Root Ports are highlighted by

DESIGNATED PORT: Needed to ensure that each segment can reah the root.
Each bridge hears on a port message from others. If it is closest to root, then that port is chosen as the designated port. Usually the port of each segment closest to the root. Here also, in case of ties break it using bridge ids (choose the one with low bridge id)

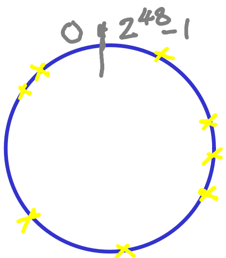
The remaining ports (i.e, Non-RP, Non-DP) are blocked ports which are disbled.

NETWORK LAYER

In DLL, we used 48 bits MAC addresses to forward messages. Suppose everyone (in the whole internet) starts using 48 bits addresses and everyone uses the distinct addresses, why can't we connect them using switches and run the Spanning tree + Learning Bridge protocols?

Scaling issues with L2 ethernet switches

a. Flat addressing : No relationship between MAC addresses on any network (on a LAN).



So, in the forwarding table we cannot do away with any entry,
If there are N machines, there will be one entry for each machine, O(N) there is no way to compress the table. But this doesn't scale because N = O(billions) in internet. Thus the routing table becomes too large.

We need to somehow maintain a relation between addresses to compress the table.

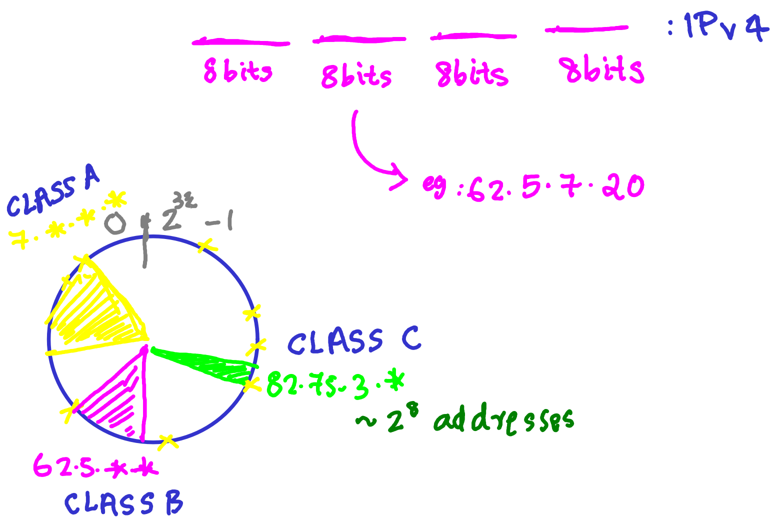
b. Broadcast may happen often. This would travel to the end of the network. This is also costly.

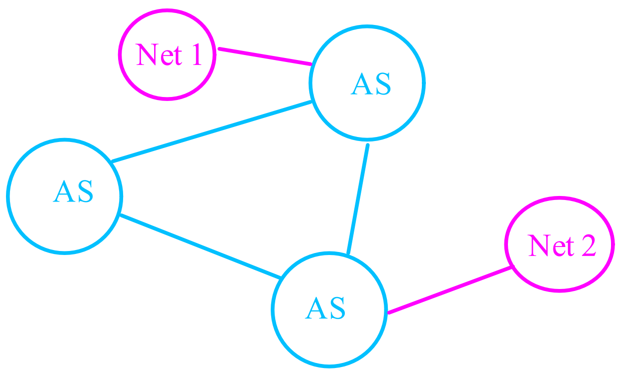
c. If the root fails or the link/node fails, the network is down for some time till the spanning tree protocol is run again to elect the root. In that case the data transfer is stopped till the protocol is run completely. This happens quite often in large networks.

d. The paths in the network may not the optimal because we have created a tree (in the tree we might not have the shortes path)

What should we do in Layer 3?

We will have IP addresses which are universal. Earlier we have IPv4 with 32 bits, but now we use IPv6 with 128 bits.



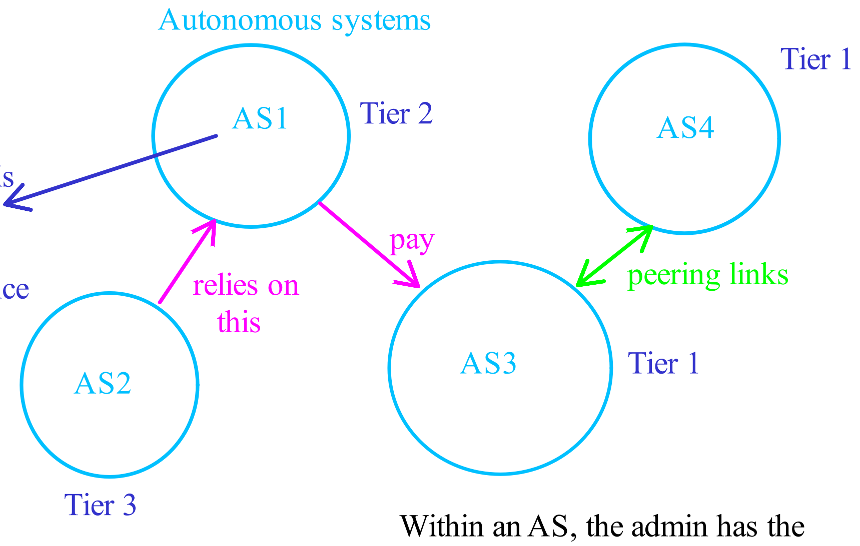


In the routing table we can now compress. For example, let the IP of NET 1be 62.5.*.* In NET 2 it can just add an entry corresponding to 62.5.*.* and can forward the packet to port correponding to that entry.

If we move our device from one part of the internet, we need to adopt a new IP address

Overview of IP routing

Use shortest path routing is followed inside each AS but each AS may follow different metrics for distance



Within an AS, the admin has the autonomy to choose whatever routing protocol to use.

INTRA-DOMAIN: Within AS (RIP, OSPF, ISIS)
INTER-DOMAIN: across AS (BGP)

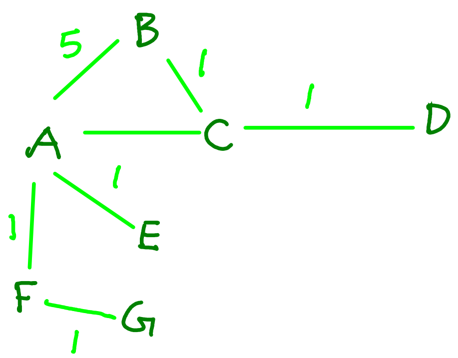
Service Level Agreement:
(Between an ISP and its customer)

Ex: An ISP might say that the customer can send at 100 Mbps, and the Up-time is > 99%
It may also give a guarantee on delay, like 30 ms latency within own AS. guantee on packet drop rate (say < 1%)

INTRA DOMAIN ROUTING (INTERIOR GATEWAY PROTOCOL)

Typically, shortest path routing: using common graph traversal algorithms like Bellman Ford, Dijkistra.

Distance vector - Bellman Ford



At A initially:

DEST.	COST	NEXT-HOP
A	0	-
B	5	B
C	1	C
E	1	E
F	1	F

At C initially:

DEST.	COST	NEXT-HOP
C	0	-
A	1	A
B	1	B
D	1	D

Min cost known to various destinations

The nodes will nown share the first two column of the table.

If C and F sends the table to A, we will have:

DEST.	COST	NEXT-HOP
A	0	-
B	2	B
C	1	C
E	1	E
F	1	F
D	2	C
G	2	F

Look at the distance vectors recievd from the neighbours, update the table accordingly using the shortest path. Eventually we will end up in a stable routing table

Distance vector is exchanged periodically with the neighbours. After few iterations, table converges.

What if one link, say F--> G fails?
If that happens, F should immediately tell A that it cannot reach G, i.e (G, \inf) is the new distance.
A will update the table and then send updates to B, C etc. that it cannot reach G and all of them should update the tables accordingly.

Count-to-infinity Problem

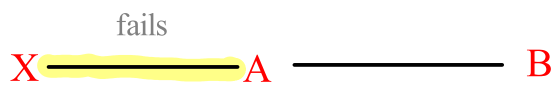


Table at B initially

DEST	COST	NEXT
X	2	A
A	1	A

Table of A

DEST	COST	NEXT
X	1	X
B	1	B

due to failure →

DEST	COST	NEXT
X	inf	-
B	1	B

from B →

DEST	COST	NEXT
X	3	B
B	1	B

from B

DEST	COST	NEXT
X	4	B
B	1	B

← keeps increasing

The DV of B is also getting updated due to A.

RIP: Routing Information protocol, where we take \inf = 16.

A does not know the whole graph/topology of the network. How can we do away with this?

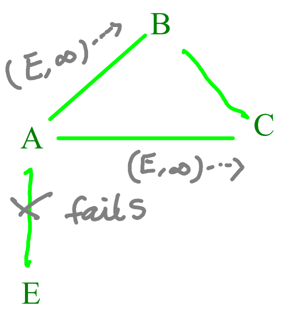
Spit-Horizon

Do not share the distance vector entry with a particular neigbour if that neighbour is the next hop to the destination concerned. So, after failure, B will not send (X,2) to A since A is the next hop

Poison Reverse

Tell the next hop neighbour that my distance (cost) to the destination is \inf.
Ex: B tells to A: (X, inf) instead of (X,2)

Failure of Split Horizon:



At B:

DEST	COST	NEXT
A	1	A
C	1	C
E	2	A

At C:

DEST	COST	NEXT
A	1	A
B	1	B
E	2	A

Not send to A

After failure:

At B:

DEST	COST	NEXT
A	1	A
C	1	C
E	inf	-

At C:

DEST	COST	NEXT
A	1	A
B	1	B
E	2	A

tells this to B

After C sends:

At B:

DEST	COST	NEXT
A	1	A
C	1	C
E	3	C

Now shared with A

Now A gets a new entry (E,4) via B and shares this with A and updates the table with (E, 5) via 5.

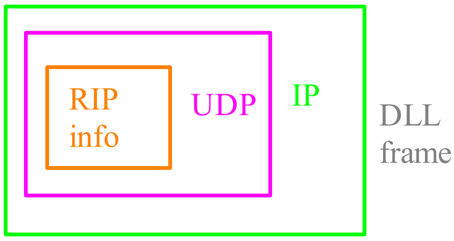
This results in a count to infinity problem again

Advantages of Distance Vector:
1. Very simple

Disadvantages of Distance Vector:
1. Count to infinity resulting in loops
2. Takes high time to converge

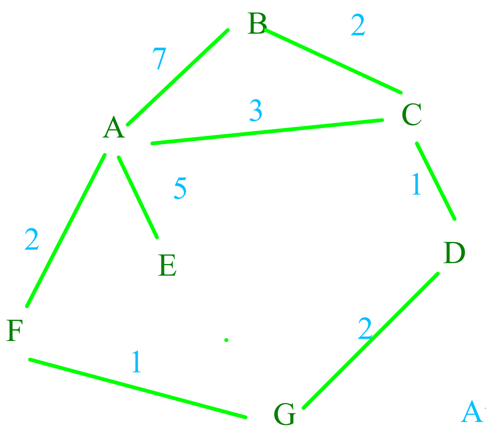
We will typically use this if the network is small, or when link failure is low.

Altenatives: Link State Routing (OSPF - Open shortest path first, ISIS)



Link State Routing(LSR) uses Dijkstra:
Each node tells all others in the network its distance to neighbours.
This is essentially a broadcast.

Example of LSR:



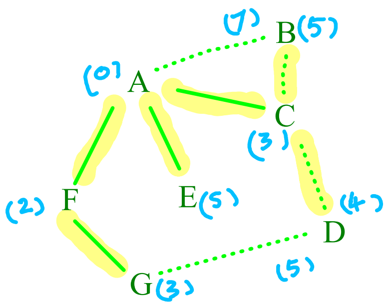
A tells everbody that it is connected to B,C,E,F with the weights as shown. Basically, it gives the local topology.

A --> C --> D --> G .... (forwarded throughout the network).

We need to do broadcast. so one obvious thing is to send the message recieved on one edge to all other edge. How long should we forward? We need to forward once on a particular link, we only want this once! Only send to neighbours who wouldn't have recieved this (probably).

At the end, all gets full graph, and run Dijkstra.

Shortest path tree from A to all others. Each time choose the link with least distance from root (highlighted).



DEST	COST	NEXT
A	0	-
B	5	C
C	3	C
D	4	C
E	5	E
F	2	F
G	3	F

Time for A to fill its routing table
= Time for broadcast + Time to run Dijikstra
-- Fast compared to DV

Assume that a link, say F-G fails. F sends a broadcast that it is no longer connected to G (Similarly G also tells (F,\inf) to all.
Everybody gets the update, and needs to re-run the Dijkistra.
-- May have loop during this time

Advantages of LSR:
1. Fast convergence
2. No count to infinity problem

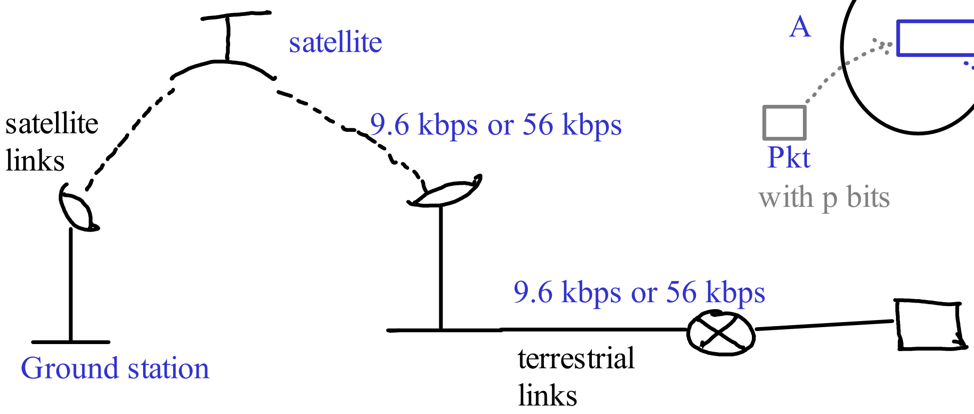
Disadvantages of LSR:
1. Need to run Dijkistra
2. Need full topology info
3. Need to do full broadcast for any change (this can be costly)

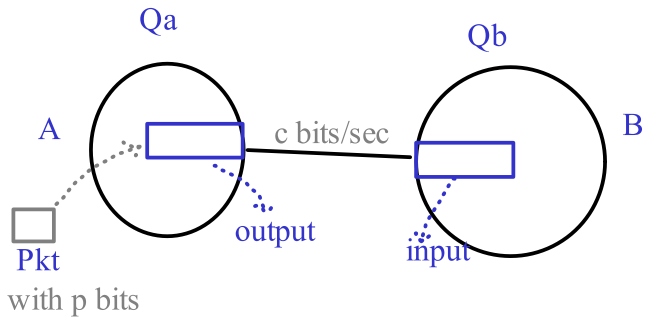
Q: What should be the link weights be? When and how should we update weights?

Some protocols have defaut weights. For ex. RIP has a default weight of 1 foe every link.

OSPF(LSR) has weight = max $\left(\frac{10^8}{\text{link speed in bits/sec}}, 1\right)$

Recall that ARPANET (1969) was the first internet.





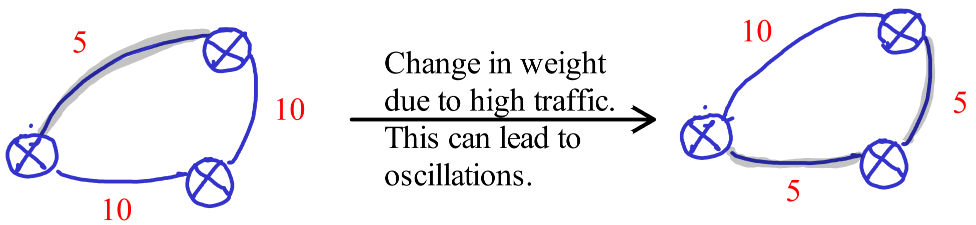
t0 = time when last bit of the pkt enters Qa
t1 = time when last bit of bpkt reaches Qb
transmission delay = p/c

t1-t0 = Queueing delay at Qa + speed of light delay + transmission delay

In ARPANET, they averaged these delays over some time period and the link weights were set equal to this average.

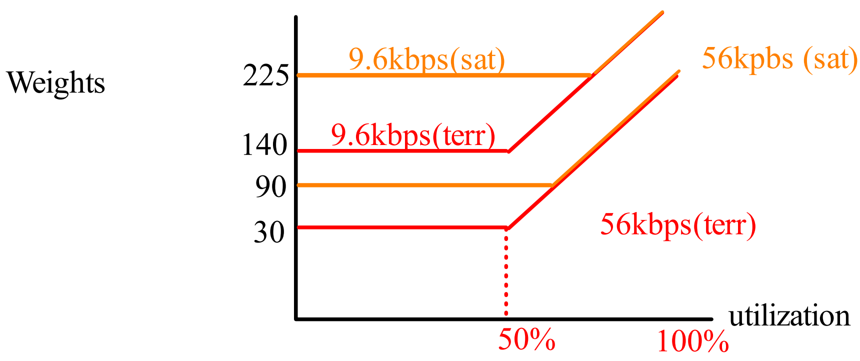
Issues :

- Oscillations. If some queue had high queueing delay and there is another one with low delay. After re-run of dijkistra the path gets diverted. Now the conjested become less conjested and vice-versa. This leads to another round- so oscillations
- Satellite link were penalized a lot, due to high speed of light delay (as distance was large). 9.6 bps (low-speed) terrestrial links had less weight than 56 kbps(high-speed) satellite links
- Range of link weights were very high. Low speed(9.6 kbps) was penalised too much compared to high speed(56kps) So, one heavy loaded 9.6 kbps link had the same weight as 126 lightly loaded 56kbps links. As a result in effect, pretty much we might never use this 9.6 kpbs.
- Queueing delay + Transmission delay was high for slow (9.6 kbps) links.

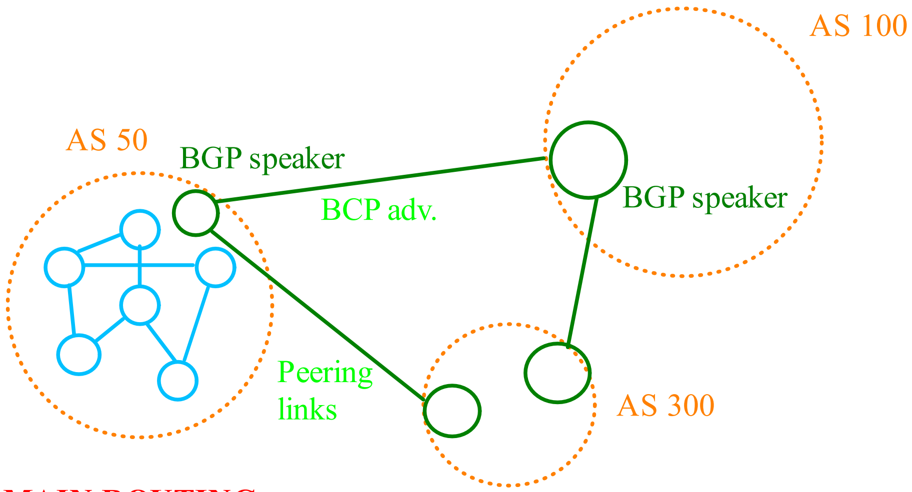


Utlilization = % of bandwidth being used.

Due to the issues above, they decided to set the weights based on the utilization.



So far we have seen Intra-Domain routing where we will implement some common shortest path algorithm



INTER-DOMAIN ROUTING (BORDER GATEWAY PROTOCOL)

SLA- Service level agreement

ICANN gives a IP prefix of the form x.y.w.z/n (eg: 72.25.3.0/24) each autonomous system.
Each AS needs to let others know their IP address. Each AS has a BGP speaker, and each of them sends out a BGP advertiser, which contains their Prefix, name and attributes. It also sends a digitally signed certificate from ICANN.

The protocol used for this transfer is called Exterior BGP (eBGP).

Interior BGP (iBGP) for within an AS

SideNote: All routers in an AS can also be BGP speakers. But this might be expensive.

Procedure to share info:

1. Use eBGP to share AS Level paths known to your AS with the neighbouring ASes
2. BGP speakers share eBGP learned info with other BGP speakers in own AS using iBGP
3. BGP speakers within a particular AS select paths to various IP prefix destinations using ATTRIBUTES of various advertisers.
4. BGP inserts external routing info into IGP routing tables.

SUMMARY

Intra-domain routing dealt with sending data within an AS. Each router inside an AS maintains a Routing table which contiains, next hop and cost, as we saw before.

Each AS is free to give any advertisement it wants. The protocol used for this is eBGP. For example,Suppose AS 300 contains the IP 83.5/16. Then the advitsement send by AS 300 will be of the form

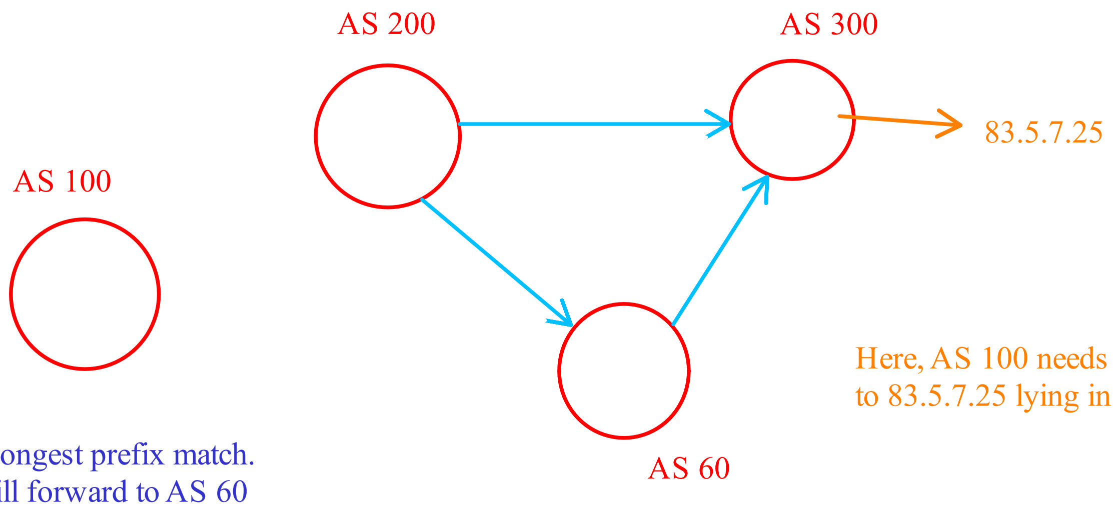
---IP prefix----AS Path-----BGP Attributes-----
---83.5/16-------AS300------~~~~~~~~~~~~~-----

Thus, AS 300 announces it contains that IP prefix.
Now, when AS 300 sends this advetisement, one of the BGP speakers (of another AS, say AS 200) could understand that this prefix is in AS 300. How will other speaker(s) know about this? For this they use iBGP.

Now AS 200 is free to adverise the same IP to others. In most cases it doesn't do this because it can result in high traffic. If they do advertise, the same procedure happens.
Now, AS 200 will also add its name to the AS PATH portion of the advertisement

Golden Rule: If a BGP speaker in an AS (say AS X) sends an advertisement for <prefix> X-Y-Z ... to a neighbour AS, and if the neighbour sends this BGP speaker a packet with destination IP matching the prefix, then AS X will forward it along the AS path which was advertised

Exceptions: Suppose we have something like ----83.5/16----AS200-AS300------
and similarly ----83.5.7/24----AS200-AS60-AS300------ and let the destination be 83.5.7.25.
The second one is a longer prefix. There are two different prefixes which match with the destination IP. Now what does the Golden rule mean here? What should AS 200 do?



Here, AS 100 needs to send to 83.5.7.25 lying in AS 300

Solution: Follow the longest prefix match.
Thus, here AS 200 will forward to AS 60

SUPER-NETTING

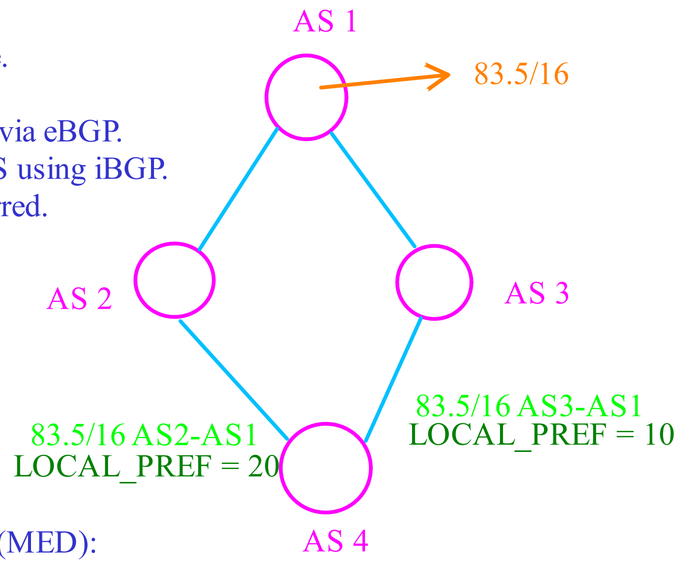
Why longest prefix match? FinalG is also connected to R4 in AS6 with weight 3ly when we reach destination we need a fu
we need to send to those who has a longer prefix. As you get further from original AS we have shorter prefix (in general). So in order to get closer to the destination, we better follow the longest prefix.

BGP ATTRIBUTES

1. LOCAL_PREF: Suppose an AS (AS 1) has a prefix and it has advertised that to AS 2 and AS 3. Suppose there is another AS 4.

AS 4 can add LOCAL_PREF attribute.

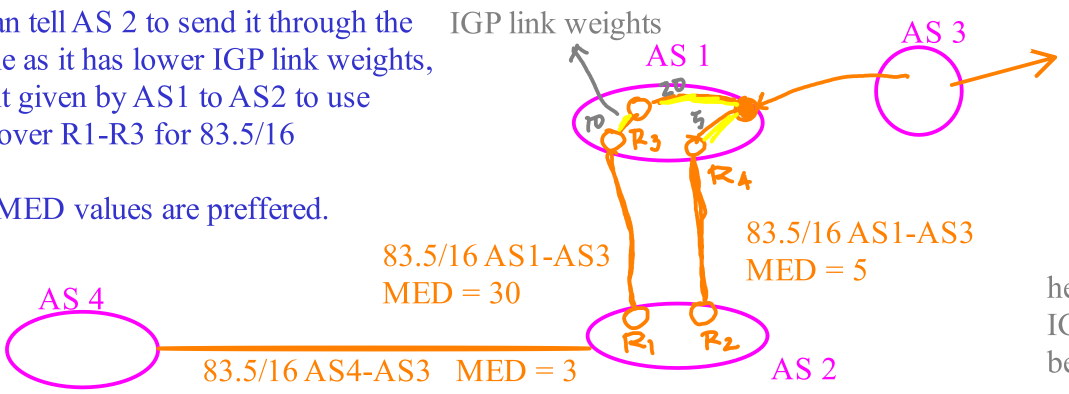
-Added locally to advertisement heard via eBGP.
-Send to other BGP speaker in own AS using iBGP.
-Larger LOCAL_PREF value is preferred.
(decided by the admin of AS).
-Not forwarded to another AS.



2. MULTI-EXIT DISCRIMINATOR (MED):

AS 1 can tell AS 2 to send it through the right one as it has lower IGP link weights, i.e a hint given by AS1 to AS2 to use R2-R4 over R1-R3 for 83.5/16

Lower MED values are preffered.

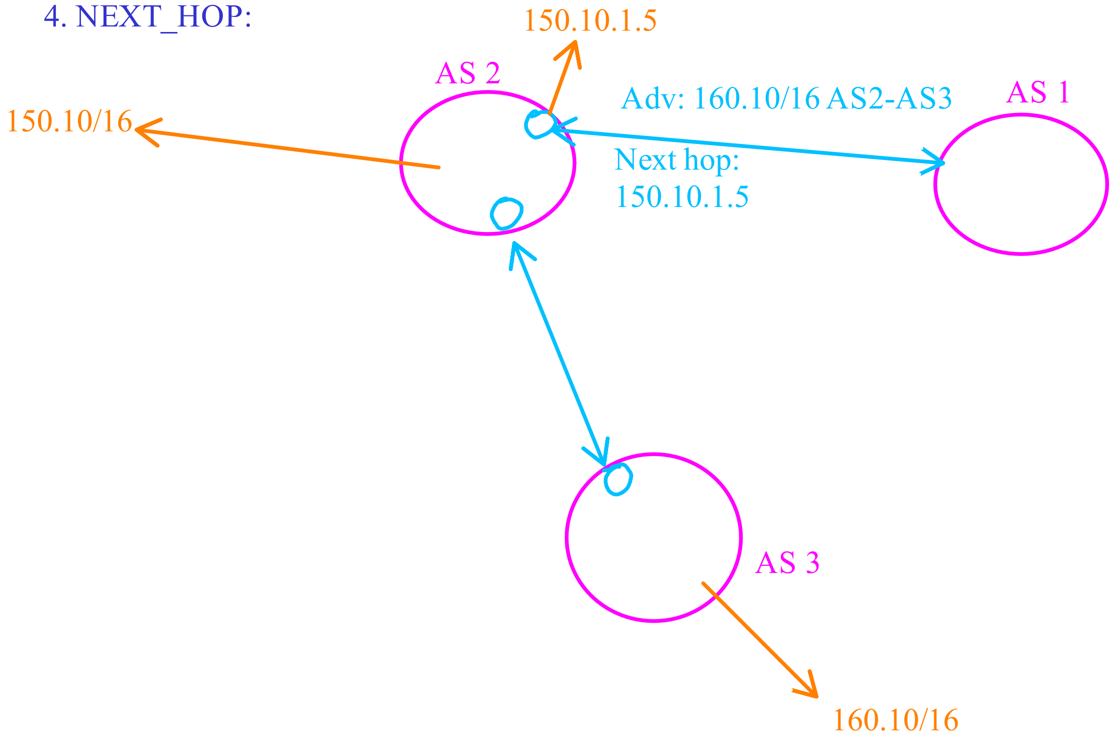


here, MED is equal to IGP link weights. Need not be always like this

LOCAL_PREF is chosen before MED. Most of the time we only need MED to comapre IP prefix send by same AS. But it can also be used in cases were another AS also sends the same prefix.

3. AS_PATH: This is the list of ASes to the destination AS which has the prefix. Shorter AS_PATH is preferred.

4. NEXT_HOP:

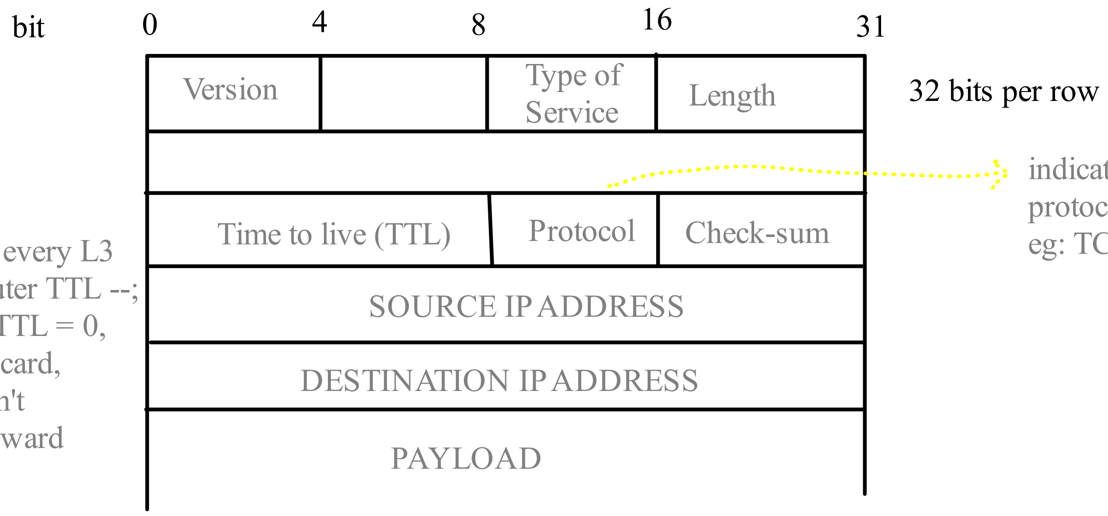


RULES FOR CHOOSING PATHS
(used at each BGP speaker for each unique prefix)

a. Choose the route with largest LOCAL_PREF
b. Choose the route with the shortest AS path (Shortest in terms of the number of ASes)
c. Choose the path with lowest MED
d. Choose the path learned via eBGP over path learned over iBGP
e. HOT-POTATO ROUTING: Choose path with lowest IGP metric to the NEXT_HOP
f. Choose path whose exit router(in the same AS) has the lowest ROUTER_ID (= highest IP address on all interfaces of the Router)

IP v4 HEADER

Best-effort service : no guarantee in terms of latency, etc.

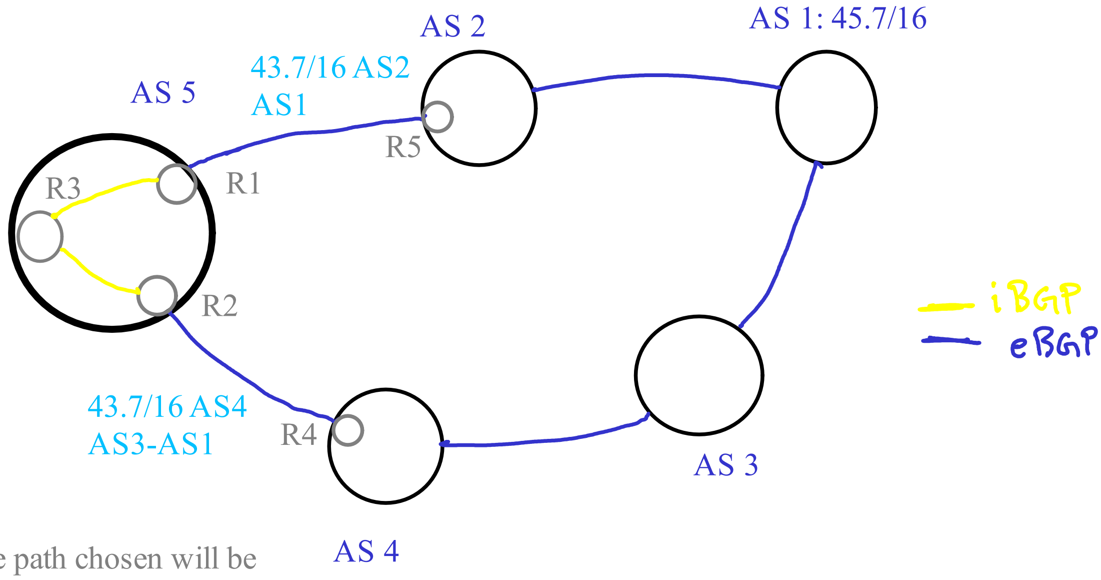


indicates next higher layer in network protocl stack,
eg: TCP:6, UDP:17, ICMP:1

At every L3 router TTL --; If TTL = 0, discard, don't forward

RECALL:
BGP FORWARDING RULES

1. Largest LOCAL_PREF
2. Shorter AS_PATH
3. Smaller MED
4. eBGP over iBGP
5. HOT POTATO routing
6. Lowest ROUTER_ID

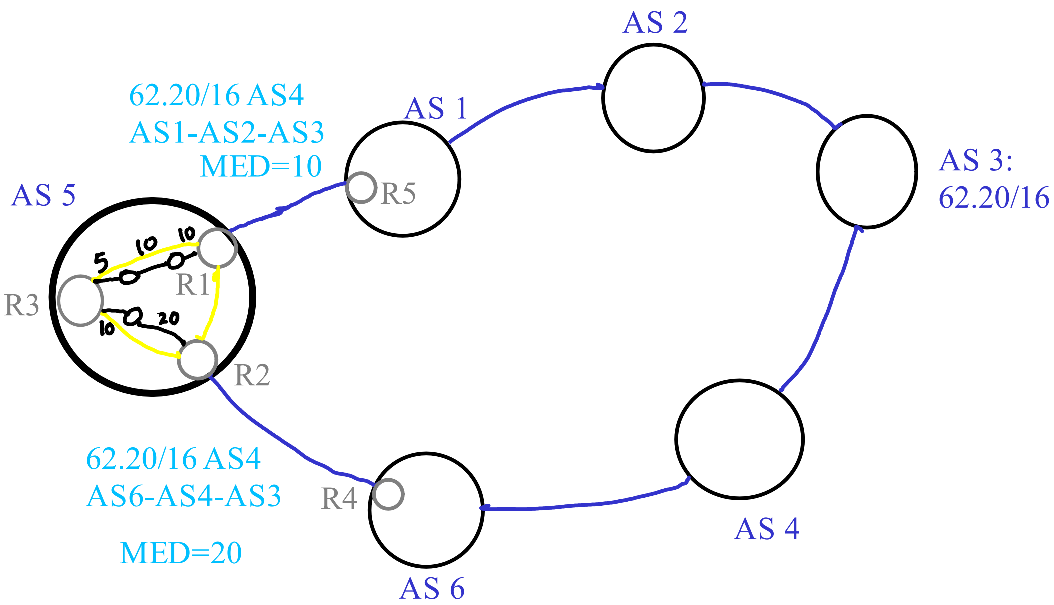


Admin of AS 5 want traffic for 43.7/16 to go via R2->AS4

By default due to shorter path, the path chosen will be via AS2. If the admin wants otherwise, the admin can set the LOCAL_PREF of the other path high.

R2 forwards 43.7/16 AS4-AS3-AS1 with LOCAL_PREF = 10 (larger than that sent by R1)
R1 forwards 43.7/16 AS2-AS1 with LOCAL_PREF = 5.
This ensures that the traffic is along AS4-AS3 path.

If the LOCAL_PATH is left as the default value, as told above the shorter path will be chosen.



If LOCAL_PREF was not set, it needs to rely on MED to choose the path beacuse both the paths AS1-AS2 and AS6-AS4 are of the same length.

Here, MED = 10 is chosen, so AS1-AS2.

If the MED is equal or not set, the routers they look at iBGP vs eBGP:
So, R1 prefers AS1-AS2-AS3, while R2 prefers AS6-AS4-AS3.
R3 has heard about the IP from both R1 and R2 using iBGP, so it need to adopt HOT-POTATO routing.

For R3 the NEXT_HOPS are R4 and R5, and it needs to choose which one is the closest to it.
The distance is chosen in terms of the sum of the weights in IGP.

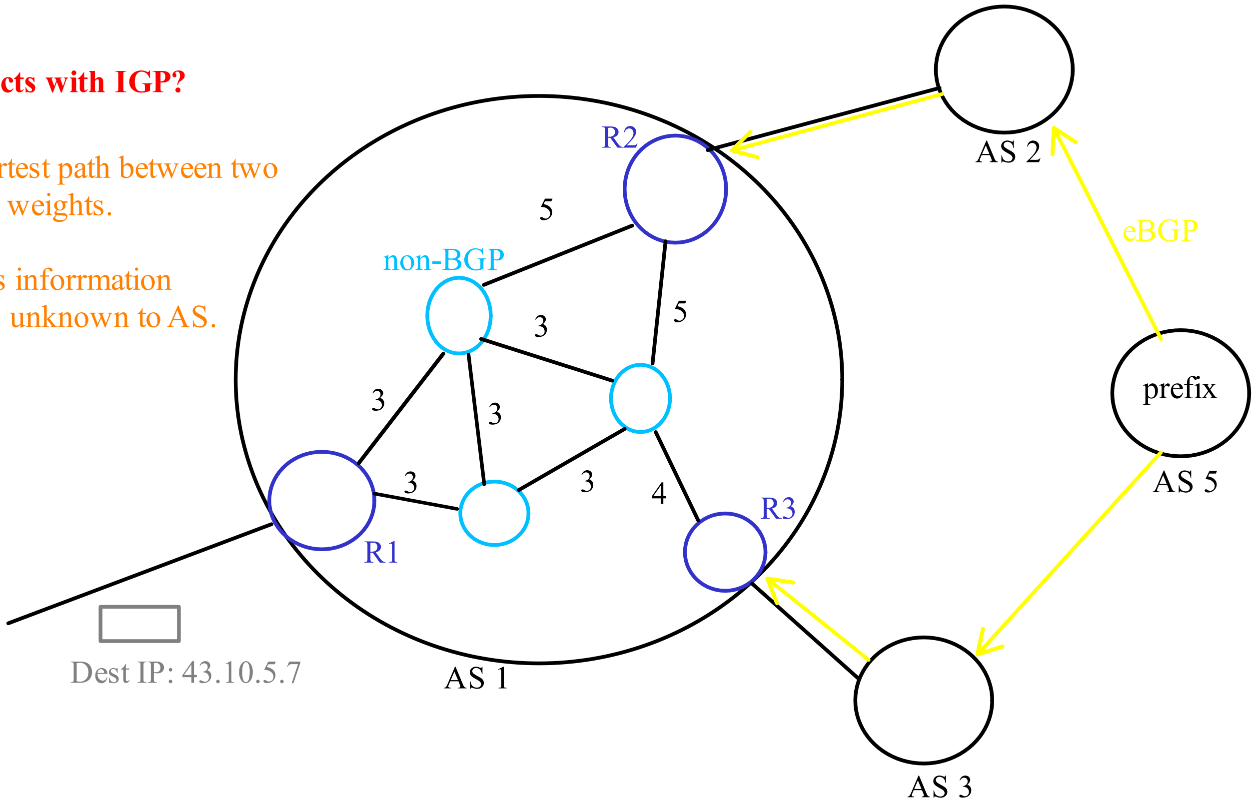
Clearly from fig. above R5 is closer to R3 (weight 25) than R4 (weight 30). So it choses the path AS1-AS2-AS3

So far we were working with BGP speakers. What about other routers. We need to incorporte BGP info to IGP

How BGP interacts with IGP?

IGP tells the shortest path between two routers given the weights.

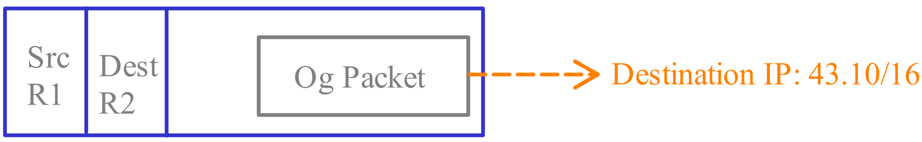
While BGP gives informmation about IP prefixes unknown to AS.



METHODS:
1. Encapsulation
2. Pervasive BGP
3. Tagged IGP

1. Encapsulation

Consider the prefix 43.10/16. Suppose R1 has get a packet with destination IP 43.10.5.7
We encapsultae the packet like below:

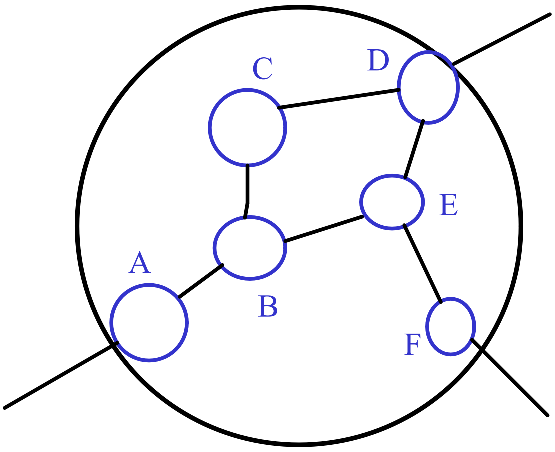


Now this packet takes the shortest path via IGP to R2.

R2 removes the encapsulation and forward the original packet to R5 and so on.

2. Pervasive BGP

This works assuming all the (internal) routers run BGP, and every prefix has a unique exit router in AS.



A has both a BGP and IGP tables:

Dest	Exit
47.5/16	D

All have the same BGP table

IGP table

Dest	Next
D	B

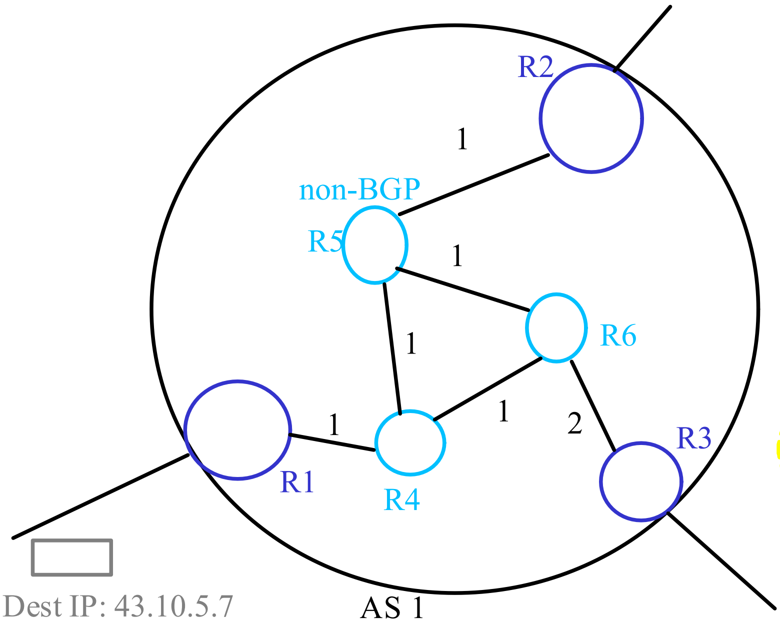
A does two table look up and decide to which node its hould send next.
Here the packet finally exit thru D, and to reach D A need to forwrd it to B as per its IGP table.

3. Tagged IGP

This can be used with Hot Potato routing.

In earlier methods BGP was not talking directly to IGP. However here there will be a direct interaction.

BGP speakers insert some tagged information into IGP about prefixes learned via BGP

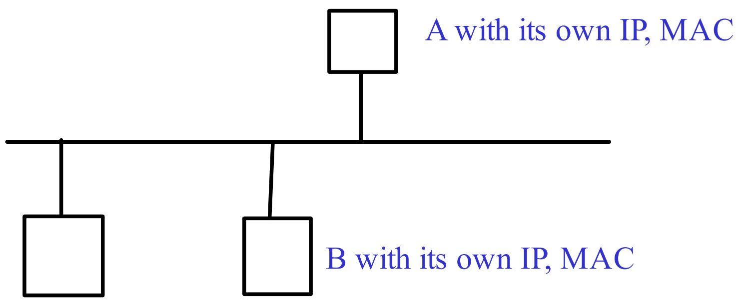


At R4, there is a IGP table :

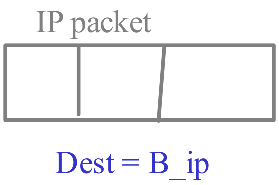
Dest	Next	Tag	Cost
R1	R1		1
R7	R5		3
R8	R6		4
R5	R5		1
R6	R6		1
43.10/16		R7	
43.10/16		R8	

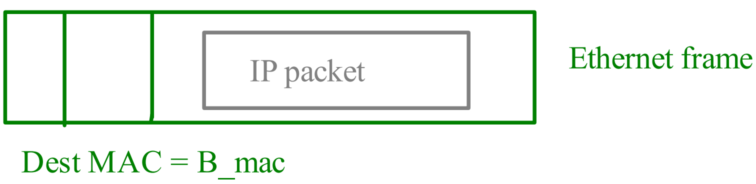
When a destination with a prefix arrives it looks at the tag
If multiple tags exist it chooses the one with lowest cost

Layer 3 - Layer 2 Interaction



IP ARP
DLL
PHY

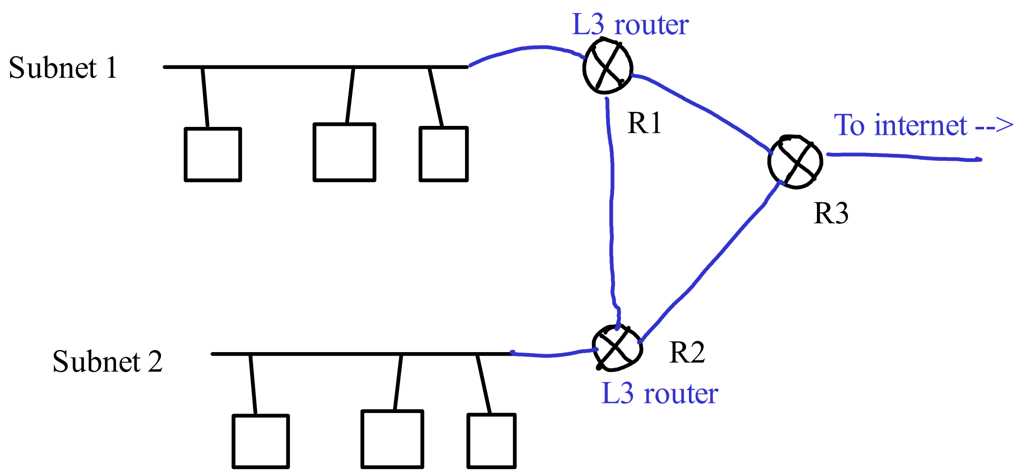




ARP: Adrress resolution protocol
- Need to be able to broadcast

SUB NETTING:

Suppose a network was given a IP prefix. This needs to be split among the different lans



Each subnet has a subnet number and
a subnet mask

Subnet mask identifies which bits of IP address
to consider.
Example: Mask= 255.255.255. 1 0 0 ....
(all ones)

Let the prefix
for this network be
75.37.3/24

25th bit
Subnet 1 -- 0 _ _ _ _ _ _ ...
Subnet 2 -- 1 _ _ _ _ _ ...

Here for S1, subnet number, S1 = 75.37.3.0
for S2, subnet number, S2 = 75.37.3.128

Given an IP address say X:
If (X and M1) == S1 then X is in Subnet-1
If (X and M2) == S2 then X is in Subnet-2 #Here, "and" means boolean-and

Table at R1:

Subnet num	Mask	Next	
S1	M1	-	Matches Subnet 1
S2	M2	R2	Matches Subnet 2
-	-	R3	Doesn't match either - can be sth external

ARP - helps Layer 2 and layer 3 to interact.

BROADCAST

1. L2 Broadcast (Ethernet)

Preamble	DST MAC	SRC MAC	Type/ Length	DATA	CRC

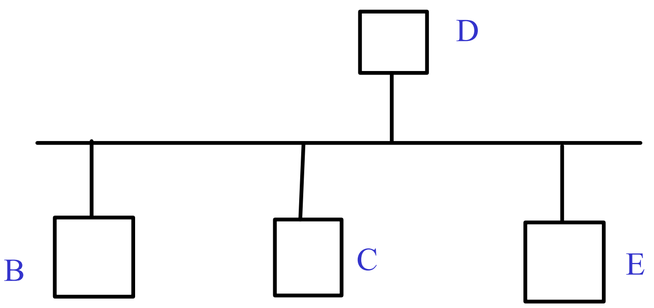
All 1's for broadcast-
All machines on LAN accept the frame
and send DATA to higher layer

3. L3 Broadcast (IP)

IP Packet:

....	SRC IP	DEST IP	....	DATA

i) Limited broadcast :- Set destination IP to all 1's. This is heard (read) by all the L3 devices at Layer 3 on the same subnet.



Recall that every message in a subnet has a subnet number and subnet mask.

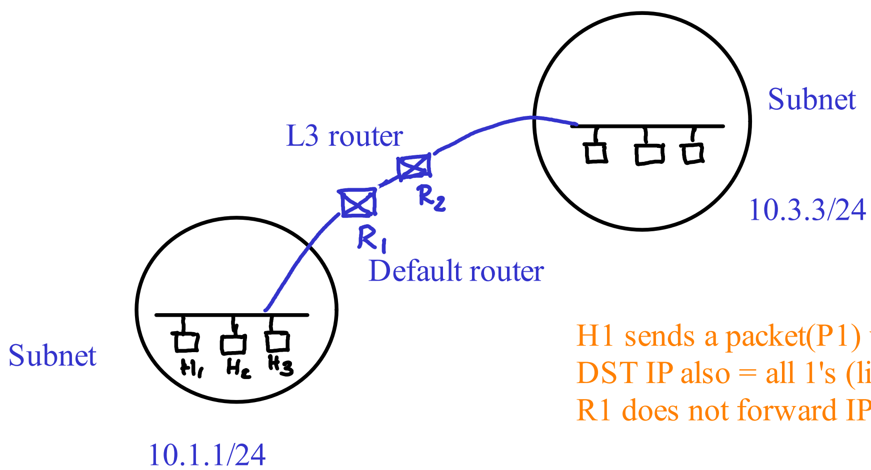
If a machine has IP 'A' and A \land MASK == Subnet number, then that machine belongs to that subnet

In the above ethernet LAN say B has to send message to all others.

Suppose in the ethernet frame, DST MAC = D_mac and in the IP packet DST_IP = all 1's.
Broadcast doesn't actually happen even though we put all 1's in IP. We need to put all 1's in DEST MAC field also.

ii) Directed broadcast :-

Default router's job is to forward pkts to the rest of Internet



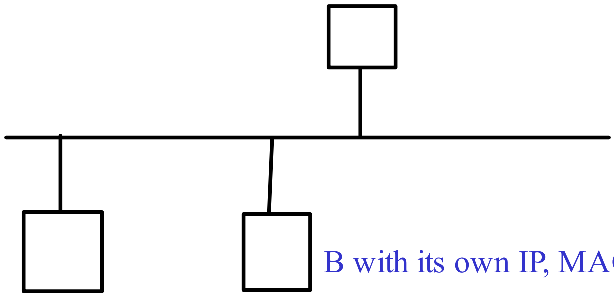
H1 sends a packet(P1) with DST MAC = all 1's and DST IP also = all 1's (limited broadcast).
R1 does not forward IP pkt to R2 since it is a limited broadcast

H1 in LAN 1 wants to send an IP broadcast to LAN 2 (10.3.3/24)
In case of directed broadcast, we set the DEST IP: $\underbrace{\text{subnet\_num}}_{10\cdot 3\cdot 3}$ ---- $\underbrace{\text{all\_ones}}_{11111\ldots}$

Consider another packet P2 with DEST IP : 10.3.3.255 , SRC IP: 10.1.1.5 (H1's IP)
SRC MAC: H1's MAC and DEST MAC: MAC address of R1

R1 will now forward P2. R2 also forwards P2 but now with a new MAC header.
This packet is now made to L2 broadcast

ADDRESS RESOLUTION PROTOCOL(ARP)



B with its own IP, MAC

A with its own IP, MAC

A has the destination IP = B_ip, but it doesn't have the B's MAC address.
ARP is used to find this out.

a) ARP request.
This will be a L2 broadcast,
everyone gets the message but only B should respond with its MAC.

Preamble	DST MAC	SRC MAC	Type/ Length	ARP REQUEST	CRC

In the ethernet frame shown, if the value in the Type/Length field > 1536, it means type otherwise it means the length of the frame. For ARP, this field is set to 0x0806

The ARP REQUEST field contains:
- i) Sender MAC : A_mac
- ii) Sender IP : A_ip
- iii) Target MAC : all 0's (because A doesn't know B_mac)
- iv) Target IP : B_ip

Everyone gets the message and compares the target IP with theirs. B will respond now with an ARP reply

b) ARP reply.

Preamble	DST MAC	SRC MAC	Type/ Length	ARP REPLY	CRC

A_mac B_mac

The ARP reply field contains:
- i) Sender MAC : B_mac
- ii) Sender IP : B_ip
- iii) Target MAC : A_mac
- iv) Target IP : A_ip

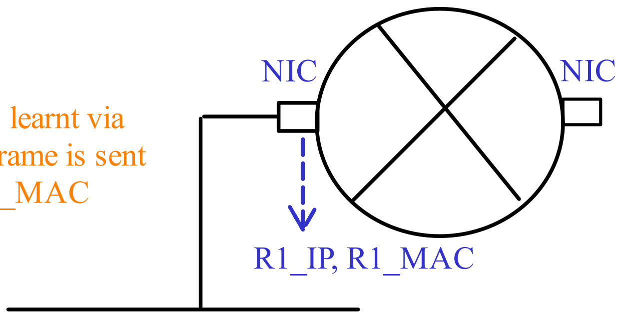
This message now goes up the layers and at the IP layer, A populates its ARP table with B_ip and B_mac.
There will be an expiry time for the entries

ARP table

IP	MAC
B_ip	B_mac

What if the DEST IP is not there in the same LAN?
We need to send it to default router (say R1 in this case).
Now what if R1_mac is not known

In this case, R1_mac is learnt via ARP and the ethernet frame is sent with DST_MAC = R1_MAC



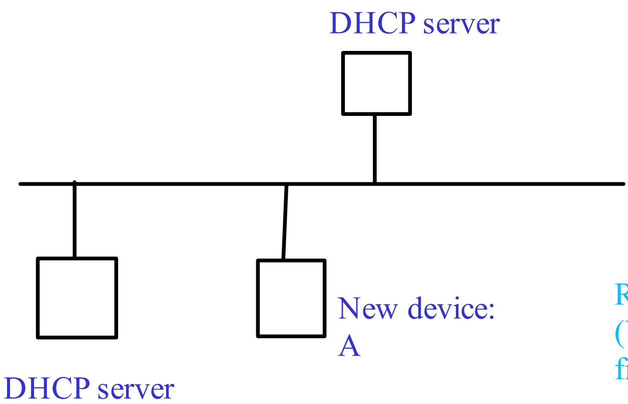
This is the case where we know the IP and don't know the MAC.

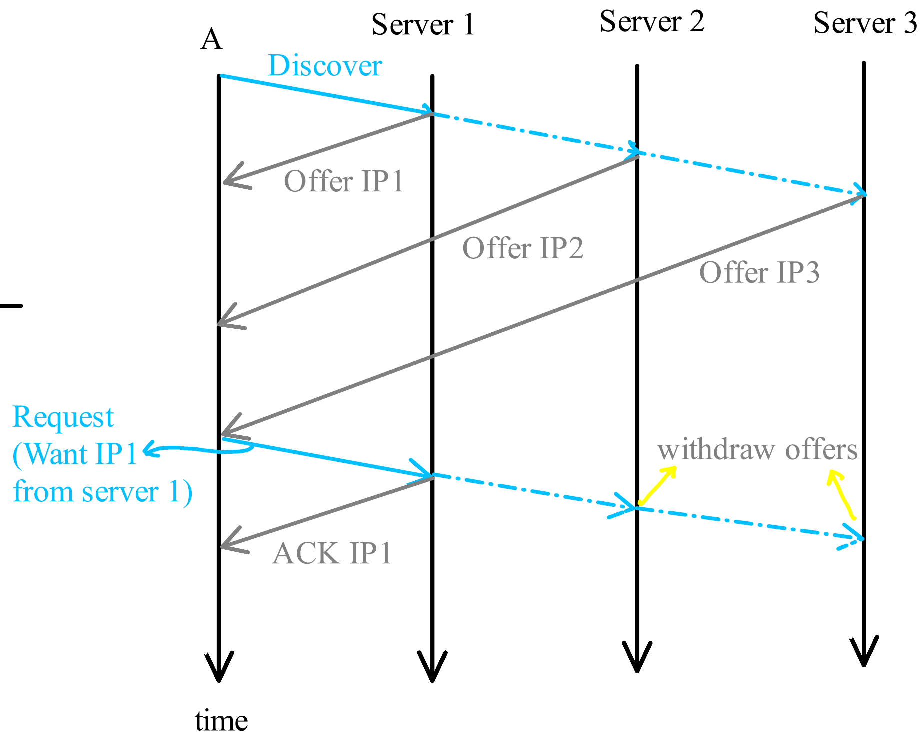
Consider the case where a new device joins a network. The new machine knows its MAC but don't know its IP.
How to get this IP address? DHCP protocol comes to the help!

Since IPs are allocated globally, we don't want them to be hardware-dependent.

Dynamic Host Configuration Protocol (DHCP)

There will be a DHCP server in the network, which can give the new device an IP. But, the device maynot know what the server is. So, it will do a broadcast





Finally A uses IP1 sent by server 1

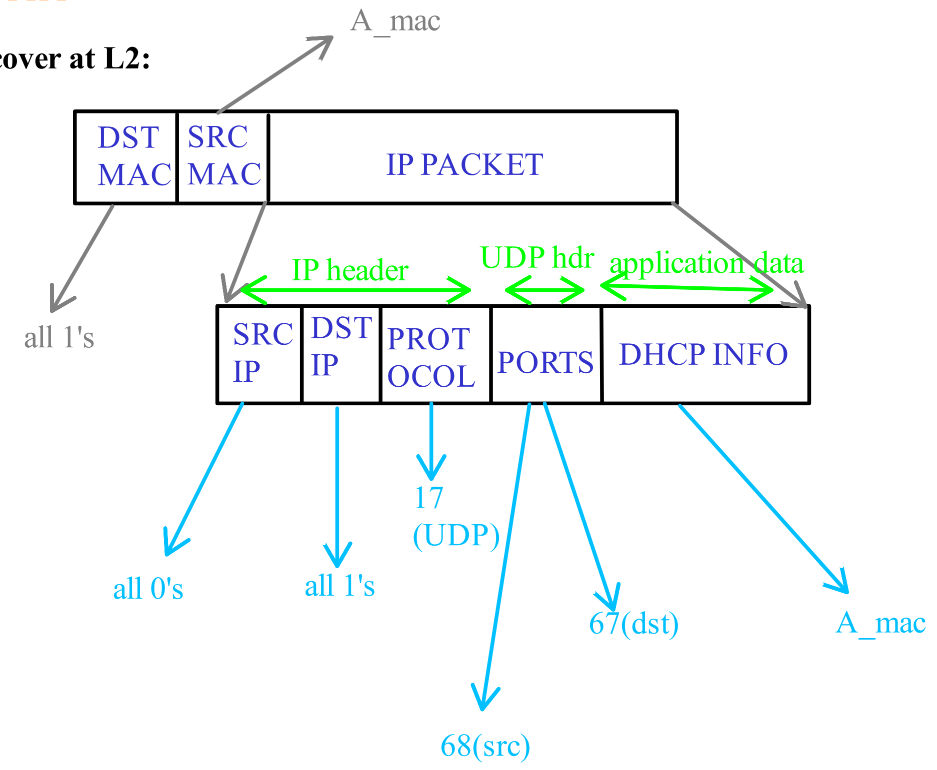
Where does DHCP sit in the network stack?

Port number feild in UDP/TCP is used
to choose which application in application layer
has to be chosen

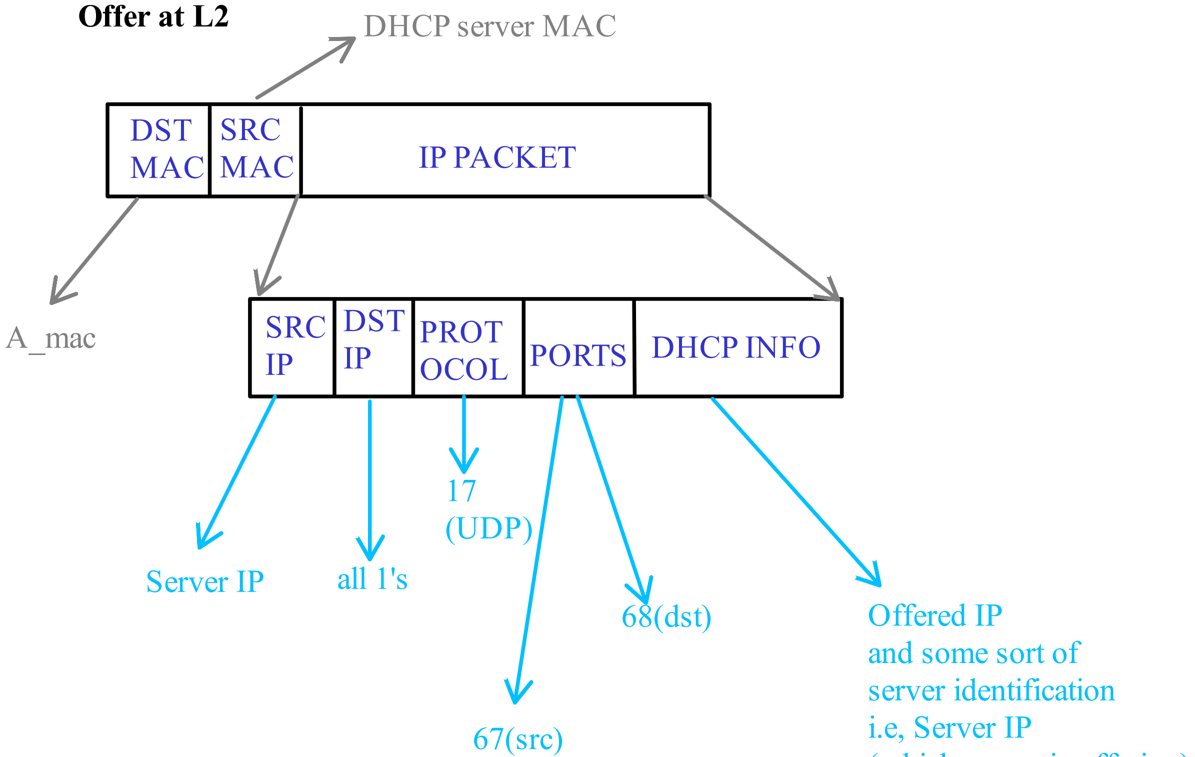
DHCP (port =68 for client A; for server port=67)

UDP TCP
IP
DLL
PHY

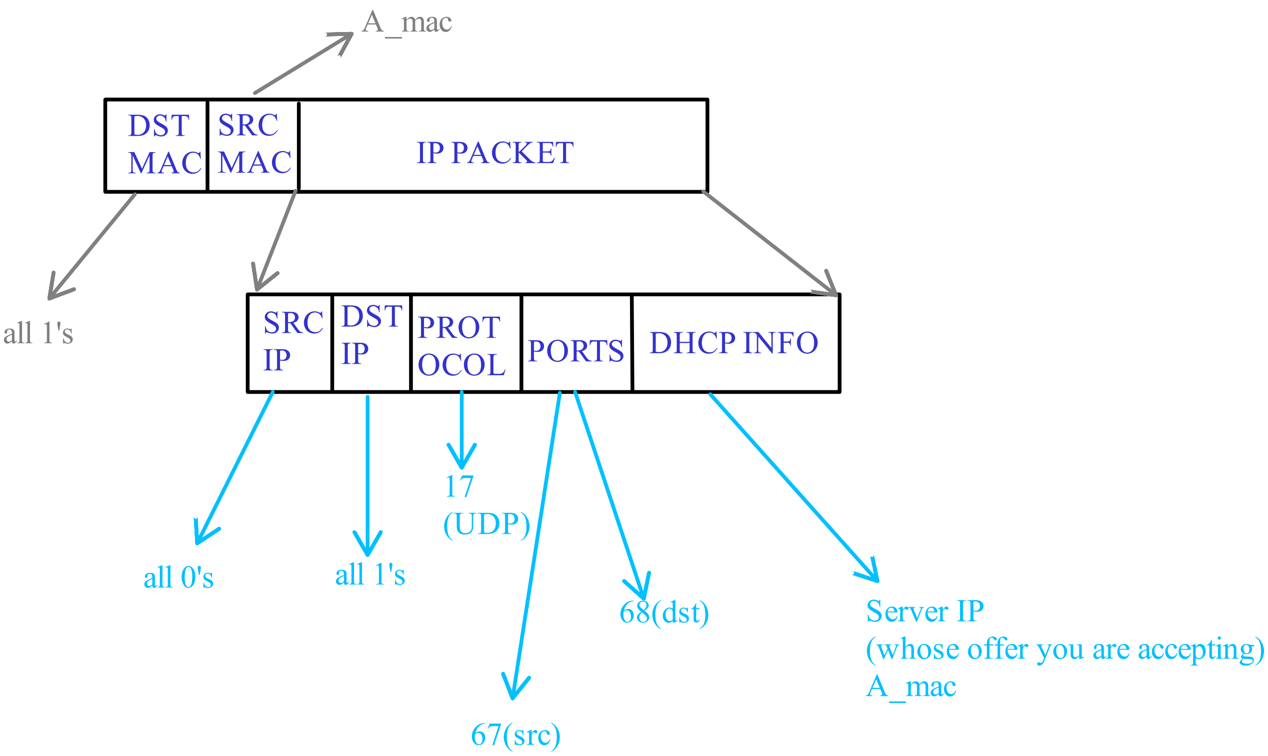
Dicover at L2:



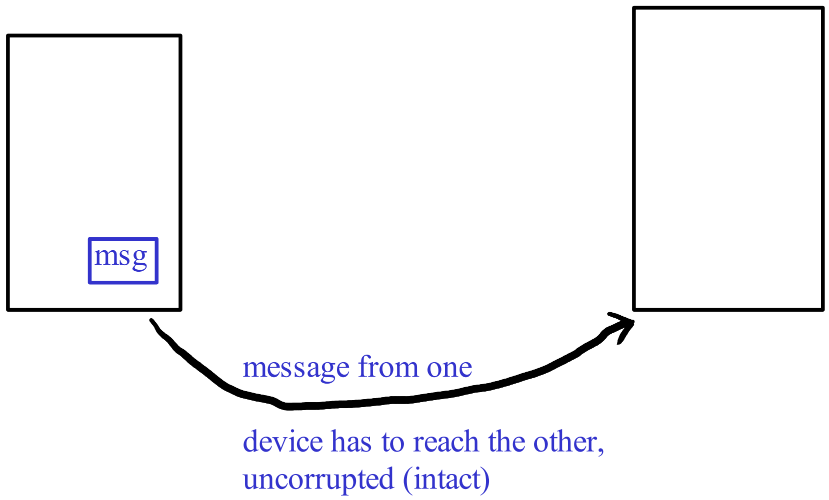
Offer at L2



Request (A --> Sent to all)



TRANSPORT LAYER



In Layer 3, often packets can get dropped. So, there is no Lyaer 3 gurantee that the message can reach the other device. Certain MAC protocols (like WiFi) will try re-transmissions, but they will only try a few times, after which they give up.

In order to ensure the message to reach the other end, we do re-transmission. What about uncorrupted?

What if the message is a large file? There is sth called Ethernet MTU (typically 1500 bytes) and the IP pkt has to fit inside MTU. So, the message has to be split up into smaller chunks and each will go as a separate IP packet.

Suppose the original packets were divided into IP packets as P1,P2,P3,... and they are being send in the order P1,P2,P3... but at the reciver the order might change. The recieved one may be P1,P3,P2,....
This is beacause the link weights can change in between and so the packets might take different paths. In some cases all the packets might take the same path, but some of them might be dropped in between. For ex. P1 managed to get through the queue, but P2 was dropped since queue was dropped. When P3 reached the queue, P1 has been emptied and so it managed to get thru. So, finally when P2 was retransmitted they reach in the order P1,P3,P2.

Guarantees about the message being transmitted intact is also not given by Layer 3.

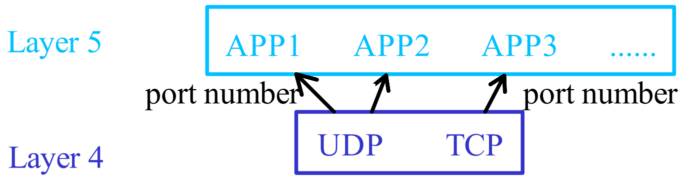
What does Transport Layer do?
- Retransmission to ensure that message reaches
- Ensure message reaches intact
- Congestion control
- Flow control (similar to congestion control)

Congestion control.
In Layer 3 if there is some queue which is full, the packets are dropped. We don't this to happen.
So in case of congestion we want the APP layer to slow down the sending rate. Similarly if there is no congestion we would like to maximize the usage of bandwidth, i.e increasee the data rate if unused bandwidth is available

The two protocols implemented in Layer 4 are UDP and TCP. Typically the applications running in APPL layer uses one or more of these.

TCP :- a) retransmission for guaranteed delivery
b) in-order delivery to the application layer at the reciever.
TCP will send the message to APPL layer only when all packets till a particular number has arrived.
c) congestion control, flow control

If our application needs all of these it uses TCP

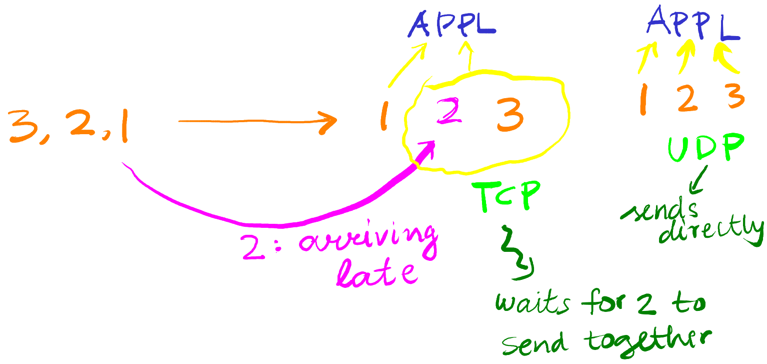


UDP :- Practically does nothing. Only sends out 1 Datagram to a given Destination IP and some port number

DHCP only used UDP. Why not TCP?
Where to used UDP instead of TCP?

1. If the message is not so long and can fit in 1 packet
2. Low latency

Recall: Port number of DHCP is 67,68
Some port numbers are reserved for some applications, ex for WEB: 80



Now-a-days Google Chrome uses sth called QUIC, which sits over UDP, and seems to be faster than TCP

UDP Header

0 ... 31

SRC PORT	DST PORT
LENGTH	CHECKSUM
DATA (PAYLOAD)	

TCP Header

Sequenc number is for data in one ditrection and Ack number is for data in other direction

0 ... 16 ... 31

SRC PORT		DST PORT
SEQUENCE NUMBER		
ACKNOWLEDGEMENT NUMBER		
	FLAGS	ADVERTISED WNDOW
CHECKSUM		URGENT POINTER
DATA (PAYLOAD)		

SEQUENCE NUMBER → MAX data is 2^32 bits. If the message is bigger, it wraps around to zero.

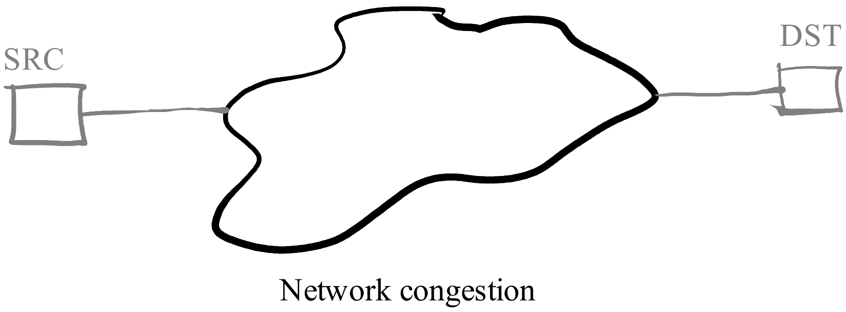
ACKNOWLEDGEMENT NUMBER → To convey which packets are recieved and not

Flag Bits:

SYN BIT | FIN BIT | RESET | PUSH | URG | ACK BIT

ex: ACK is set 1 if ACK is included in segment
SYN is send to set up connection,
FIN used to close connection

Advertised Window:



DST can be congested,
if buffer gets full and APPL doesn't read fast enough

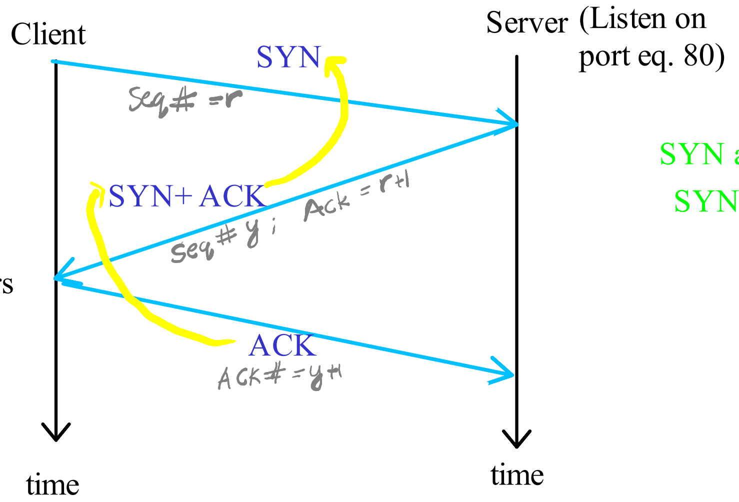
FLOW CONTROL

Advertised window is used by the destination to say how much free buffer lefy

Note: For UDP we use Datagram socket, while for TCP we use stream socket.

CONNECTION: Sender $\xleftrightarrow{\text{TCP}}$ Receiver

TCP connection Establishment



Here,
r and y are
random numbers

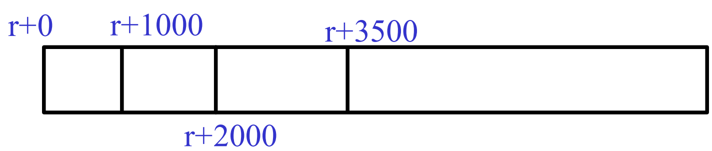
SYN and ACK are set in the flag feild.
SYN is considered as one byte

3-way handshake

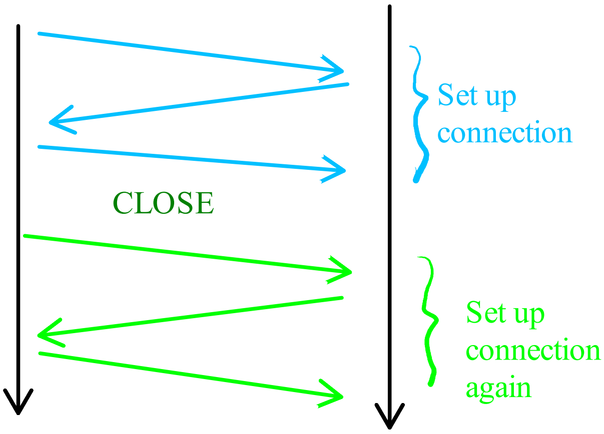
In TCP communication we consider DATA as a stream and the DATA is divided into segments.

TCP header contains a feild for SEQ NUMBER. If we put SEQ. # as 1,2,3... continuously to each segment, it can have some issues. The segments may not be of the same size.(So if we lose one segment the receiver doesn't know how much bytes is lost. So we think of segments as a stream of bytes.
Now we send the first byte in every segment as the sequence number.

Also, we don't start numbering the bytes from zero, but from a random number, r



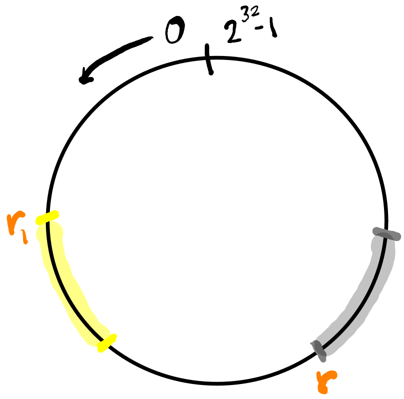
If ACK number = A => Got everything from the start (i.e, SYN SEQ #) till Byte A-1, and we are expecting byte A next.



Consider this case where we have back-to-back connections.
It is possible that an old segment from previous connection (which was floating around in network) and turns up at the receiver.

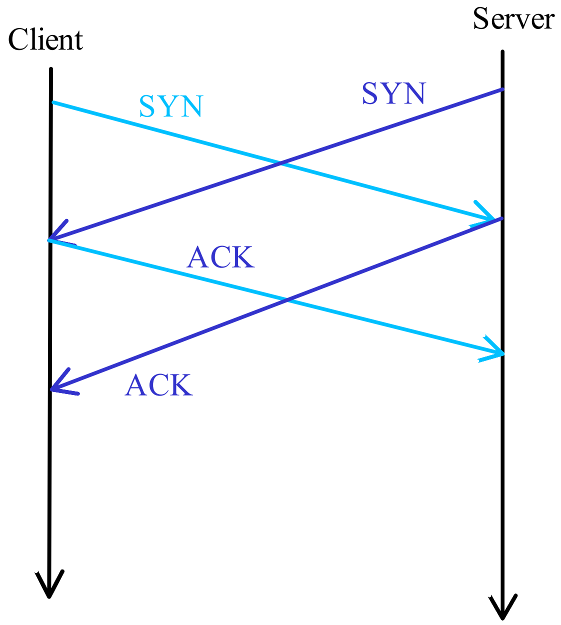
To avoid this we start numbering bytes from a random number. so its unlikely that the old sequence has same SEQ. # as a new one.

The random number is send to the server in the SYN packet



Unlikely that these regions overlap.

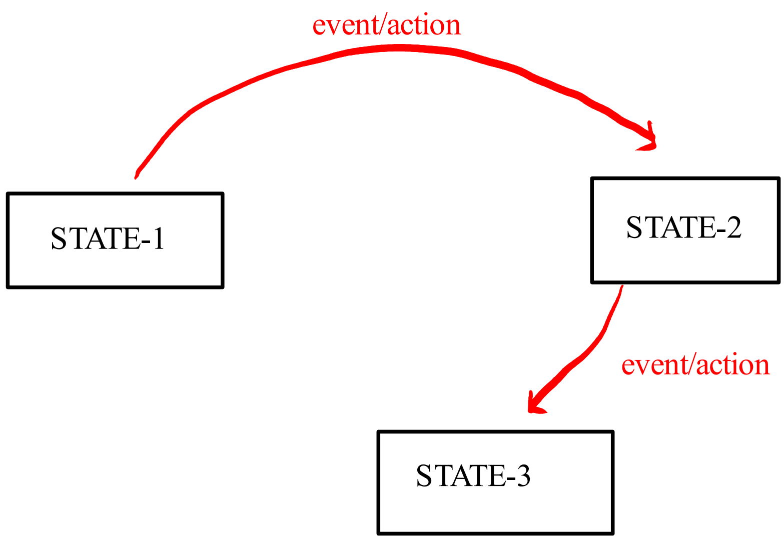
4-way Handshake



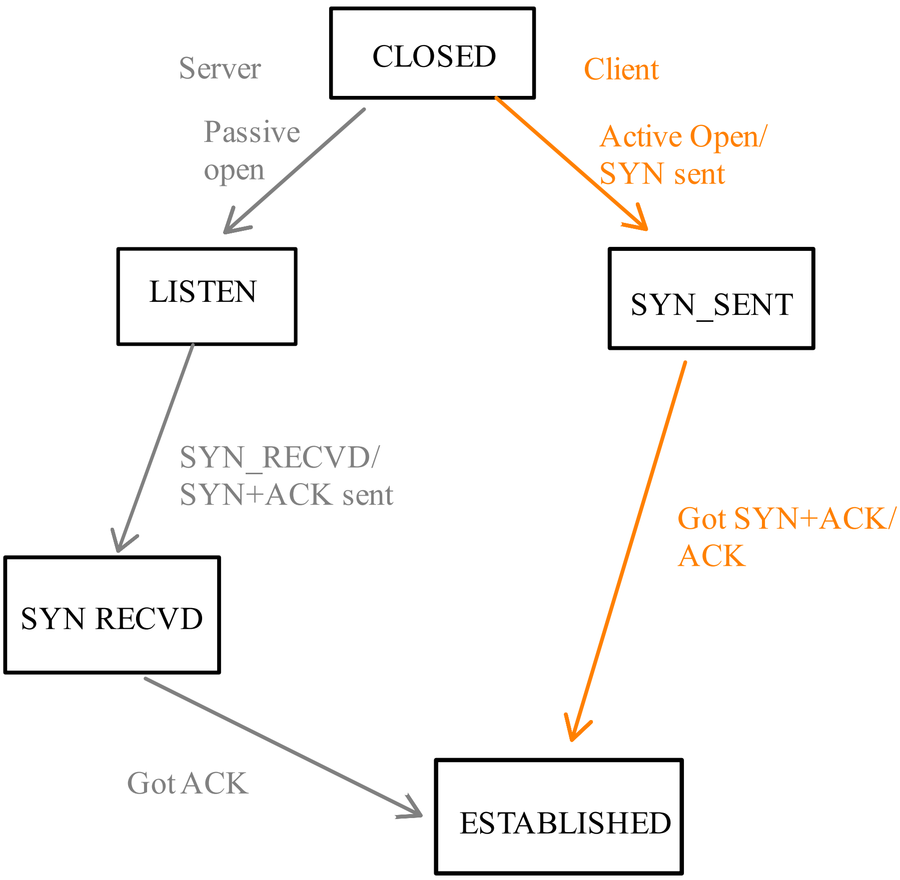
How to identify a particular TCP connection?
First look at the IP addresses: SRC IP and DST IP
Ports: SRC PORT and DST PORT
Protocol feild in IP Header: Indicates that it is TCP

Thus we have a 5-tuple:
<SRC IP, DST IP, SRC PORT, DST PORT, Protocol>
This defines each TCP connection.

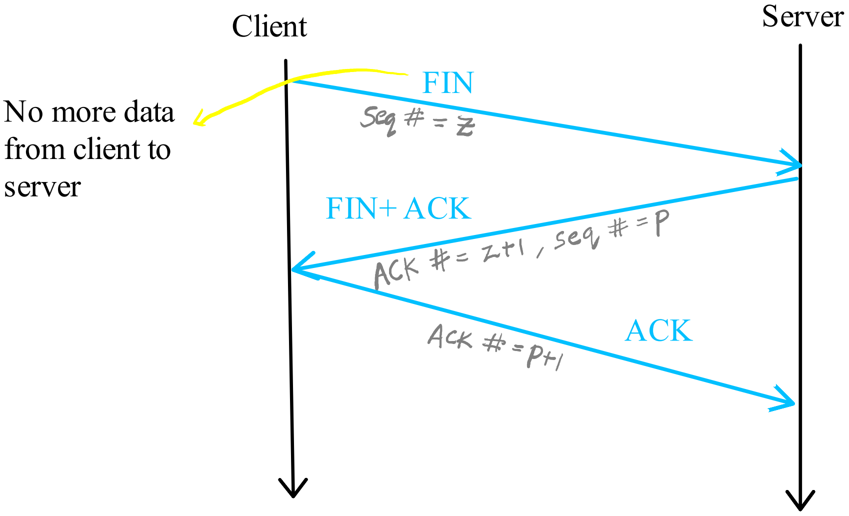
STATE DIAGRAM:



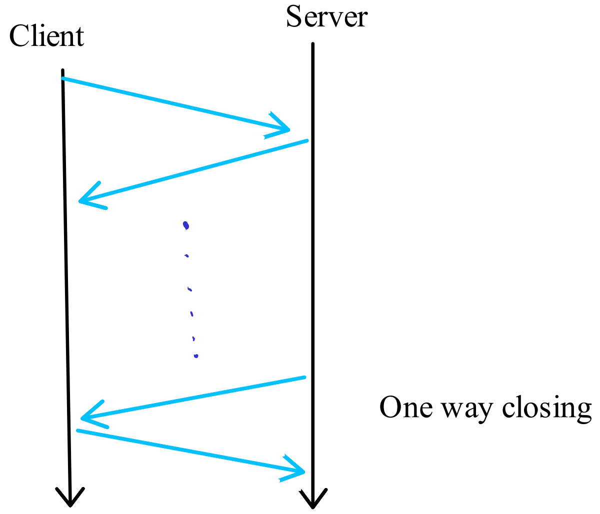
State diagram for 3-way handshake:



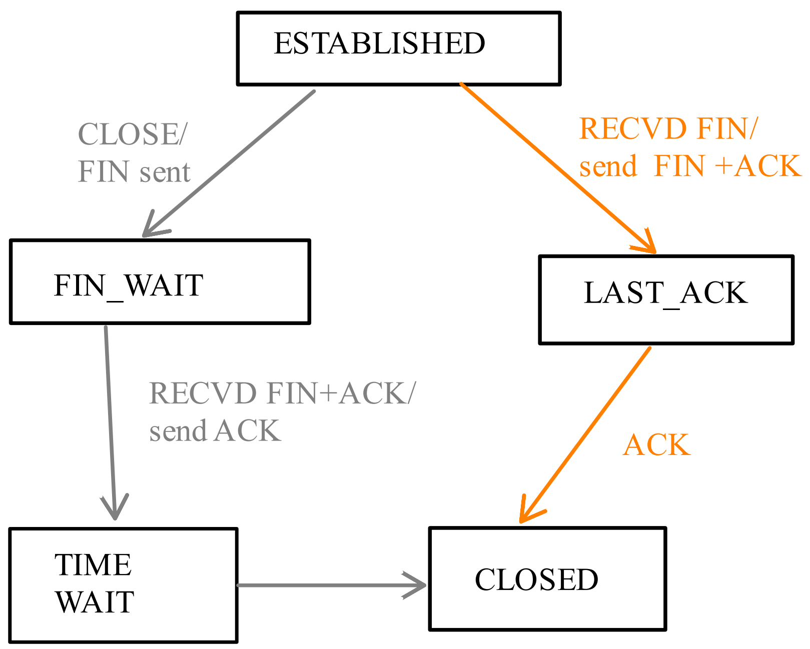
TCP connection Termination



3-way Handshake



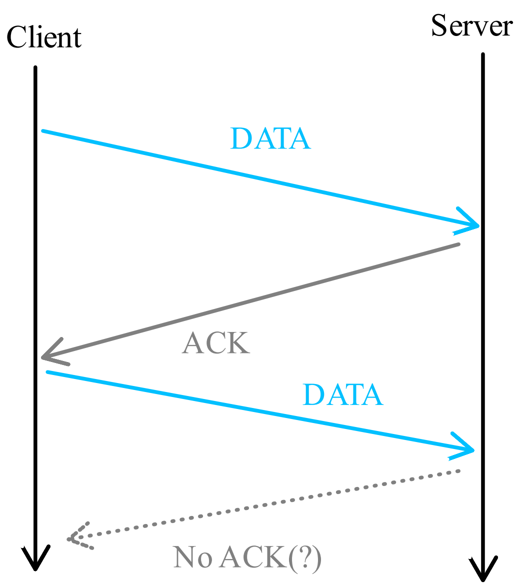
State diagram for termination (3-way hanshake)



wait for
2*MAX_SEGMENT
LIFETIME ~ 120 sec

Whatever was there in the network would be cleared from network after this time, and the port number will be not in used

Retransmission of lost packets

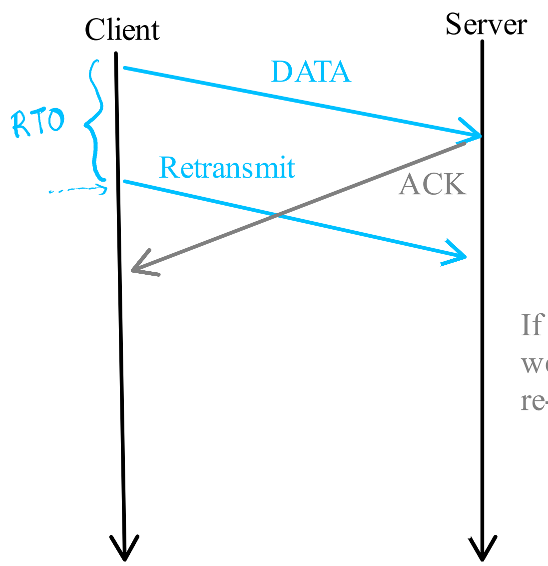


Recall that if ACK number is x,
it says that we have received everything
from start to (x-1) and is expecting x next

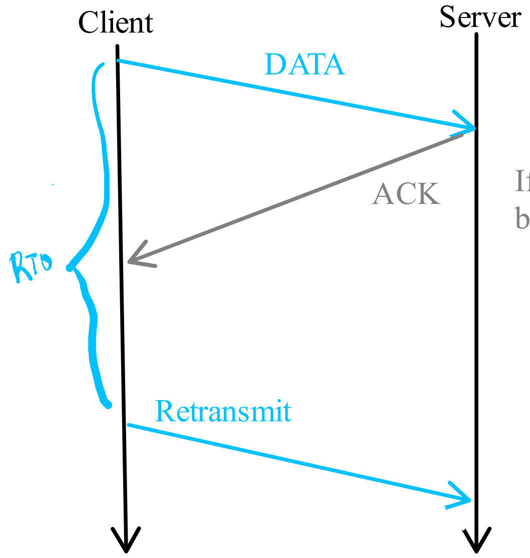
Assume that there was no ACK after a data segment was send. There are two cases:

1. The DATA segment was itself lost
2. ACK was lost.

How long to wait till we re-transmit?
i.e, What should re-transmission timeout (RTO) be?



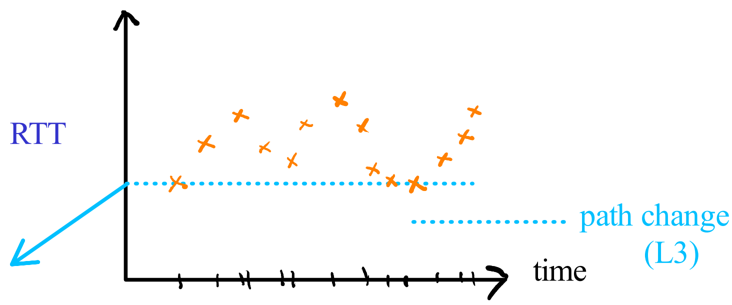
If RTO is too small,
we unnecessarily
re-transmit



If the RTO is too large we might take too much time before we re-transmit.

Issues:

1. RTT can be of the order ms-secs (across connections) in the internet depending on the location of source and destination
2. RTT for the same connection us variable

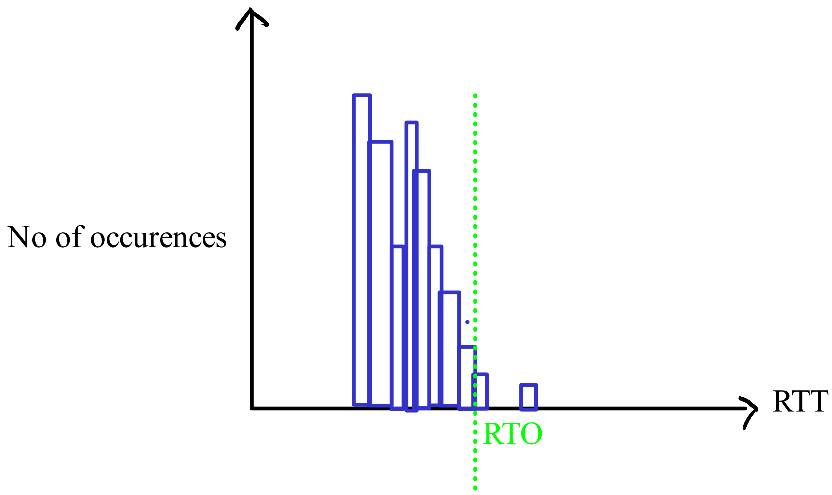


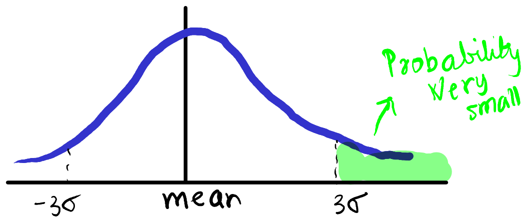
Time at which data pkts are sent out

min value:
speed of light delay
+n transmission delay
(if queueing delay is zero)

Transmission delay:
time at which data pkts sent out
= p/c
(STORE AND FORWARD:
wait for the full pkt to reach the router before forwarding to output queue)

Histogram of RTT:





Idea: We measure the RTT over time and we set the RTO as mean + const*(standard-deviation)

Let x1,x2,...xn be the estimates of some random number, then:

Mean estimate, $M = \frac{1}{n}\sum_{i=1}^{n} x_i$

Estimate of standard deviation = $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - M)^2}$ (can be computationally intensive if 100s of ACK arriving per sec)

So we adopt sth called, mean deviation = $\frac{1}{n}\sum_{i=1}^{n}|x_i - M|$

ALGO used by TCP (for RTO)

RFC: Request for comments

SampleRTT;
(latest RTT estimate)

Current estimate of mean:
EstimRTT

1. Difference = SampleRTT - EstimRTT // like x_i - M

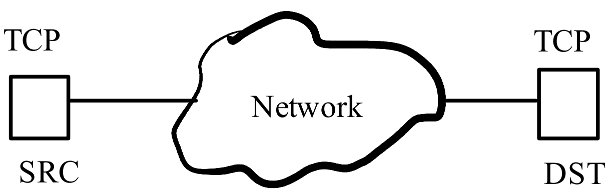
2. EstimRTT = $(1-\alpha)$EstimRTT + α SampleRTT, where α \in (0,1)

3. Deviation = $(1-\beta)$ Deviation $+\beta$ |Difference|

4. Timeout = μ EstimRTT + ϕ Deviation

By default $\alpha = 1/8$, $\beta = 1/4$, $\mu = 1$ and $\phi = 4$

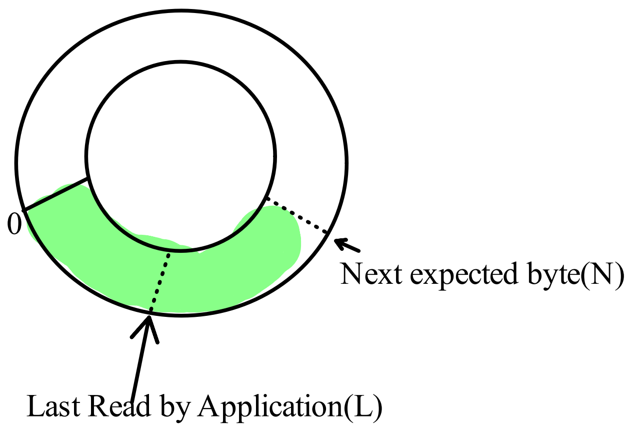
Congestion and Flow Control



What if congestion occurs at DST?

FLOW CONTROL

At the TCP layer at Destination, there is a cicular buffer:



We have an advertised window, which is a feild in the TCP header. This is set as the free buffer available.

If the total buffer size = M,
N-L-1 bytes have been used (see fig.).
Then, remaining buffer size is (M - (N-L-1)).
This is ok, if N>L. But it is possible that N can wrap around so that N<L. So, if N<L, the remaining
Buffer size = (M - (L-N+1)).
Combining these two, we have:
Adv. Window = (M-(N-L-1)) mod M

SRC uses sth called a window which is the maximum bytes of un-acknowledged SRC that can be sent out
Window = min (congestion window, Advertised window)

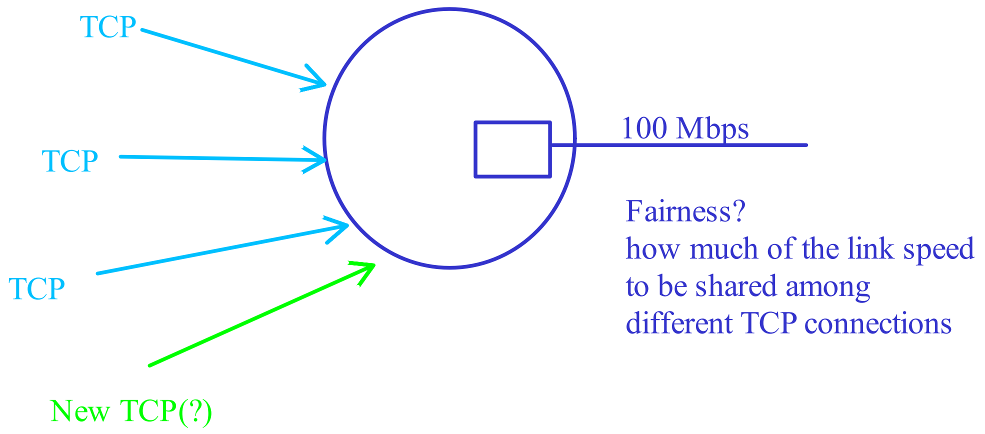
TCP CONGESTION CONTROL

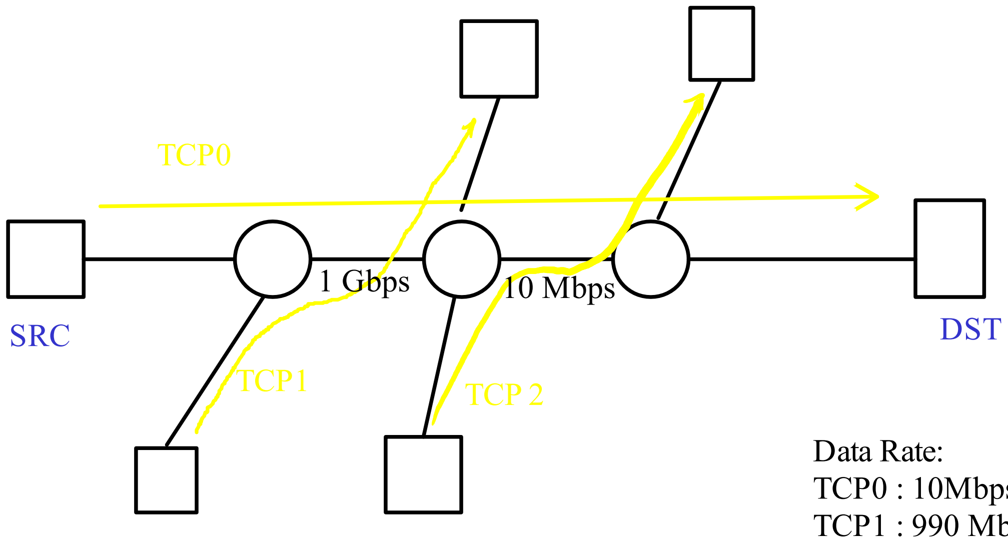
In the early ARPANET the decided to have no state info about connectons (No Application or Transport state is stored) . The interediate routers only care about packet forwarding. The complexity is pushed to the ends - SRC and DST. As far as SRC and DST is concerned, the network looks like a black box. Some TCP segment is sent out, and finally ACK is sent out.

For ex. If there is a queue is filling up in a router, it doesn't tell the SRC or DST directly about this. So, the TCP might not have any idea of bottleneck link speed..

The End-Hosts which are runnig TCP :

1. Don't know link speed on the paths
2. Don't know the link utilization
3. Don't know the number of TCP connections sharing the path





Data Rate:
TCP0 : 10Mbps
TCP1 : 990 Mbps

CONGESTION CONTROL ISSUES:

Each TCP connection:

- Wants to use bandwidth resources efficiently
- Do not want to cause congestion (packet loss, queue filling up)
- Fairness - One TCP connection should not grab most of the bandwidth at the expense of other connections

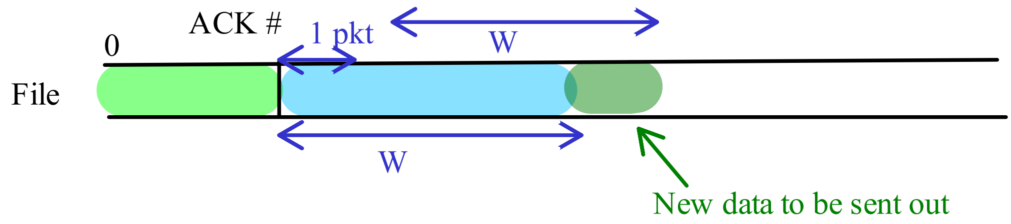
Q: How to set data rate of TCP?

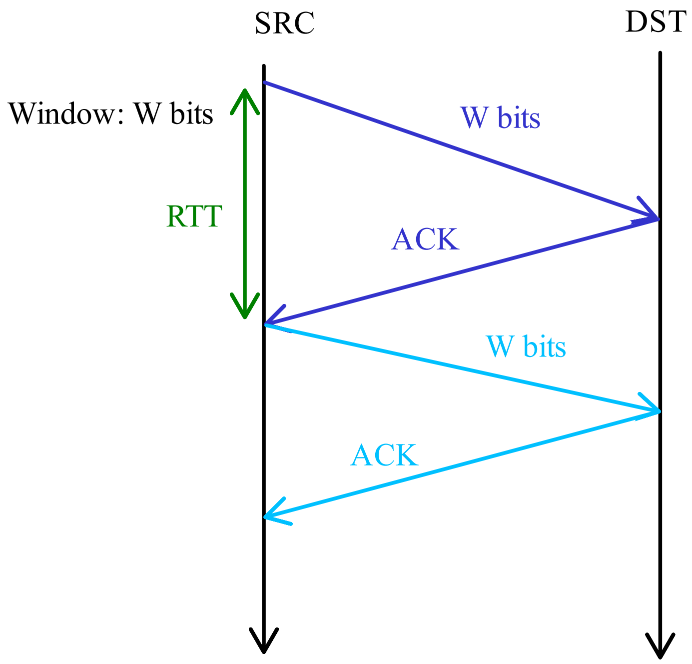
Consider a SRC sending data at 10 Mbps.
We can divide time into slots (say as 1 sec intervals). If each packet is of 10,000 bits, to maintain 10 Mbps, 1000 packets can be send in that 1sec interval.

We can also make the slot 10ms and send 10 packets of 10,000 bits (instead of 1s granularity).

WINDOW - BASED DATA RATE CONTROL:

We have a window W = max amount of un-acknpwledged data you can have in flight in the network





Data rate = W/ RTT

We increase or decrease to vary our data rate

When source sends W bits, it means that atmost W bits are un-acknowledged. These W bits can be in some queues.
W also limits the amount of congestion in the network.

If W is smaller we are filling up the queues to a smaller extend.

W = min (Congestion Window, Advertised Window)

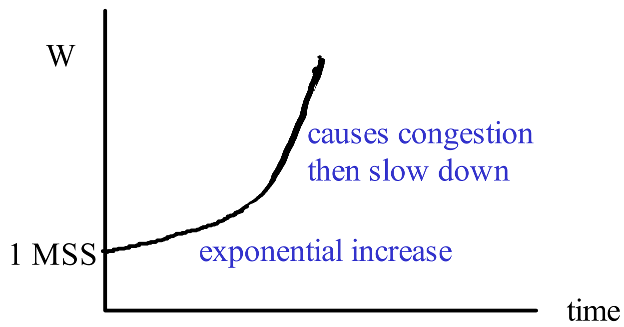
Initial value of W?

Suppose we set W = 1000 packets, and each paket contain 10,000 bits, then effectively W has 10^7 bits. Suppose RTT is 1 ms, then data rate = W/RTT = 10^7/10^(-3) = 10 Gbps. This is so large that it can congest the network.

In practice, W is initially set to be 1 MSS (Maximum segment size) [RFC5681]
1 MSS ~ 10^4 bits

SLOW START: Start with 1 MSS.

Assuming W = Congestion Window



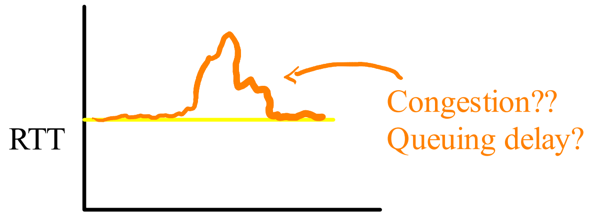
How to know about congestion?
How to slow down?

Summary of TCP Congestion Control so far

TCP sits at the end hosts, and they cannot communicate directly with intermediate routers and conclude how much congestion is present. So they consider the network as a blackbox and try to infer sth.
How to do? We are anyway sending DATA and getting backs ACKs. We can get some idea about what is happening in the network using this. (Example of congestion: Queue of an intermediate router might get filled up.)

The signals of congestion in the network:

1. Increase in RTT
2. Increase in packet loss
3. ECN: Explicit congestion notification



ECN: Explicit congestion notification

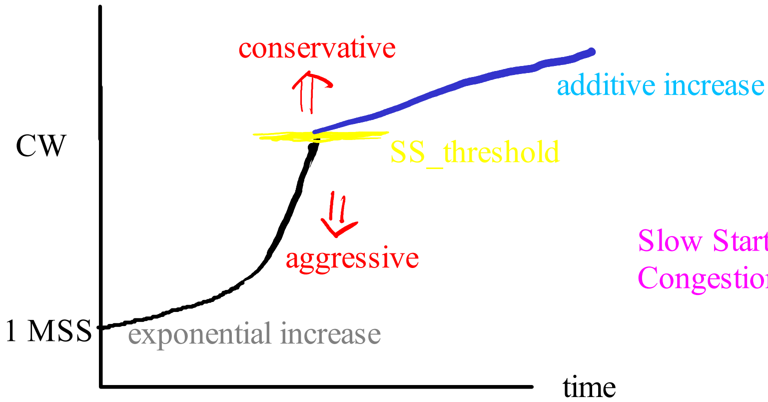
Explicit information from the routers indicating that there is congestion is happening.

How to send this message without changing the protocol?
If there is congestion the router sets appropriate bits in the header to signal to the source about congestion

How do we know if Packet loss occured?
We have a Timeout, and if there is no ACK before the timer is out, we can conclude packet loss. This timeout can be larger than RTT.
Can we detect packet loss earlier?

With ECN we can convey information of congestion must faster than other methods

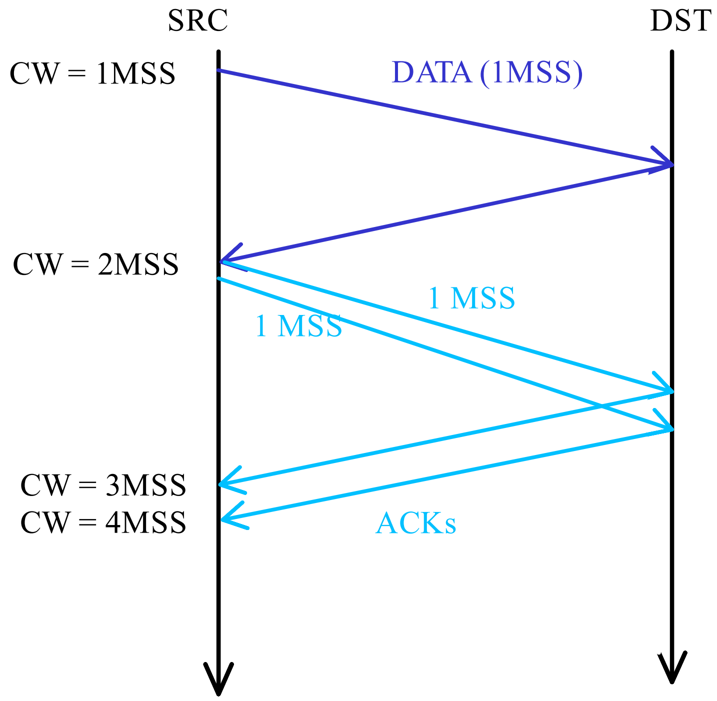
Another Issue: Range of available bandwidth is very high. It can vary from 10kbps and 10Gbps. So at what rate should TCP start sending the data? We saw that we adopt a SLOW START.



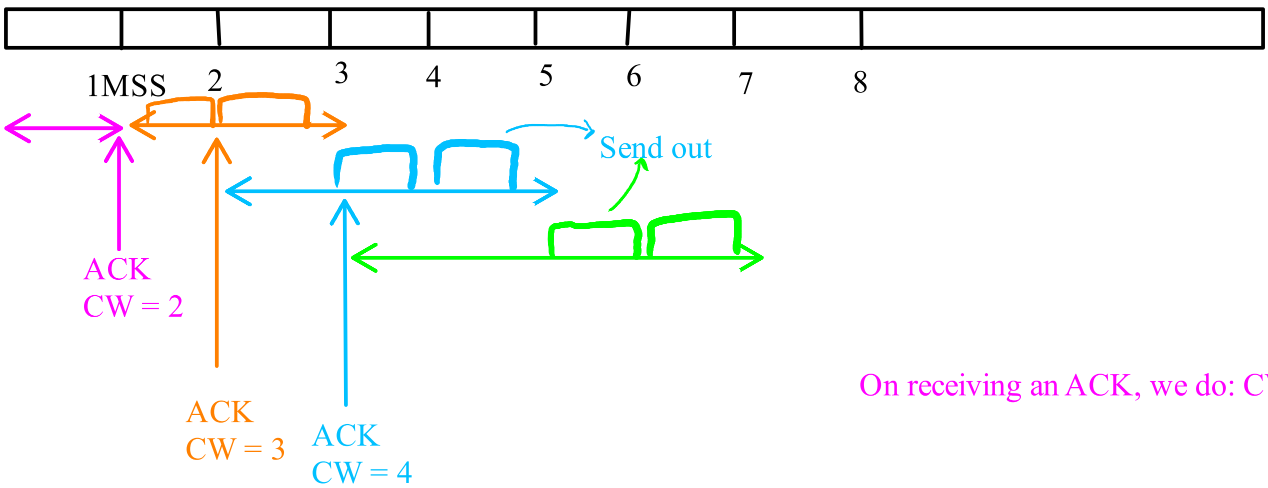
Slow Start: when CW < SS_threshold
Congestion avoidance: CW >= SS_threshold

How to practically increase the window size (CW)?

SLOW START: Here we have exponential increase - double the CW every RTT



Initially my CW is 1MSS, I could only send that much. But now I can larger chunk of data as I got an ACK for the first DATA I sent.

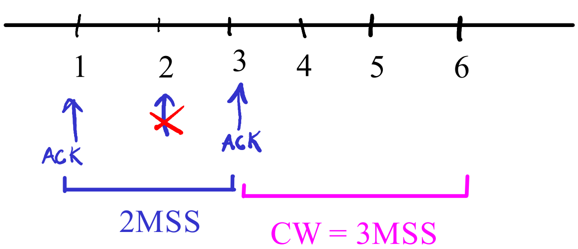


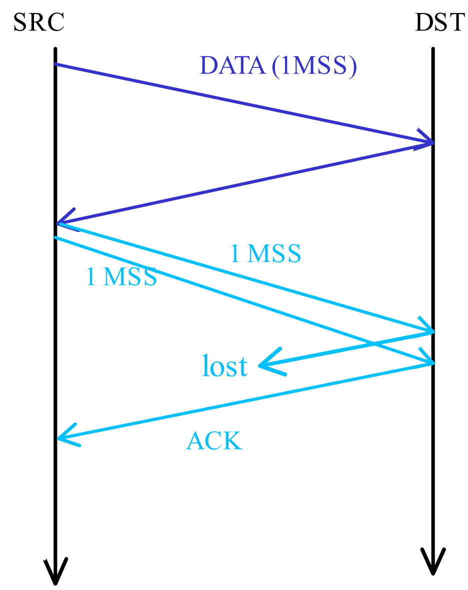
On receiving an ACK, we do: CW += 1 MSS

What if one the ACK gets dropped?

RFC 5681:

In SLOW START (CW < SS_threshold) then if an ACK acknowledges "N" (new) bytes,
then CW += min(N,MSS)





How to do Additive Increase?

Here we need to do CW+= 1MSS per RTT, while in case of SLOW START it was CW+=1MSS per ACK

We need to do sth like CW += x per ACK ===> CW += 1MSS per RTT

In a CW, the number of segments per RTT, n = CW/MSS. Assuming that x is fixed, then effectively we want

nx = 1 MSS => x = MSS/n = MSS/(CW/MSS) = (MSS^2)/CW

So, in CONGESTION AVOIDANCE (ADDITIVE INCREASE):

If CW >= SS_threshold, then on receiving an ACK we do:

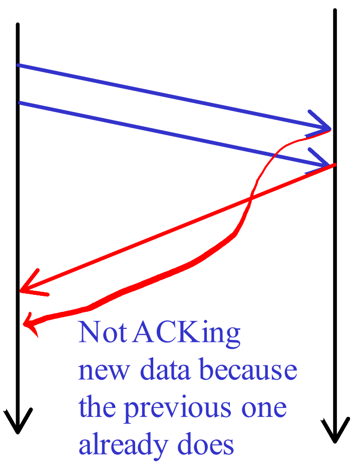
$$CW \mathrel{+}= \frac{(MSS)^2}{CW}$$

RFC 5681: If CW>=SS_threshold, on getting ACK for new data (non-zero number of new bytes ACKed)

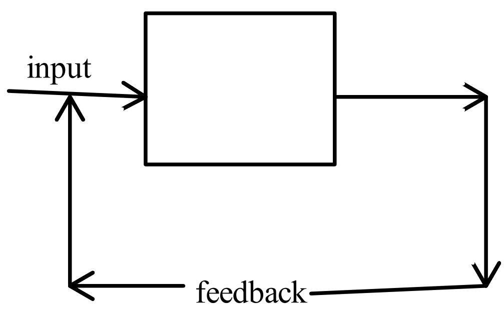
$$CW \mathrel{+}= \frac{(MSS)^2}{CW}$$

How to detect and react to congestion?

In CA phase



STABILITY: We don't want any prolonged congestion



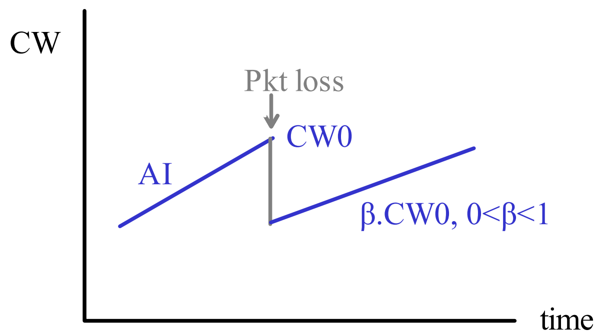
A typical Control system

AI-MD : Additive Increase Multiplicative Decrease

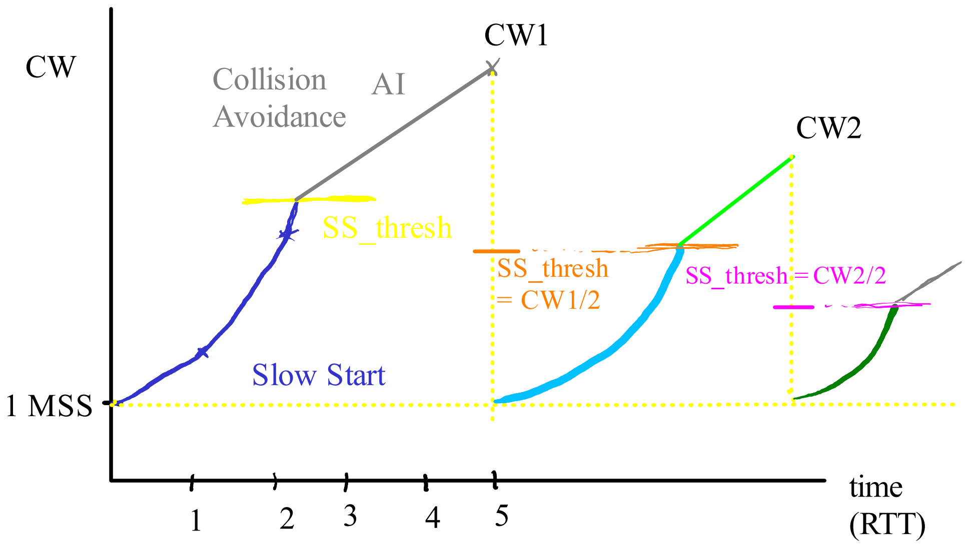
Drastic decrease when congestion occurs

Increase conservatively

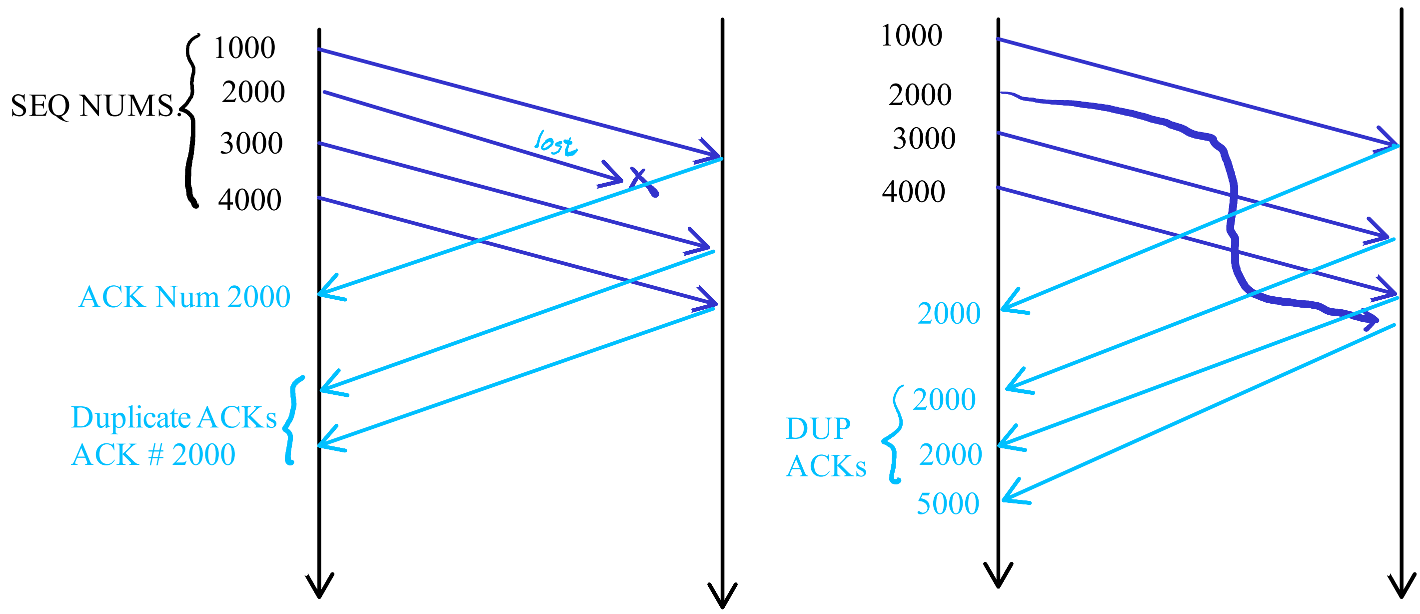
AI-MD is found provide stability as we needed



TCP TAHOE



Can we detect Loss before time-out?



See from diagram that there can be some correlaion with number of DUP ACKs and Pkt loss, but not necessarily

Rule: On getting 3 Duplicate ACKs assume packet is lost

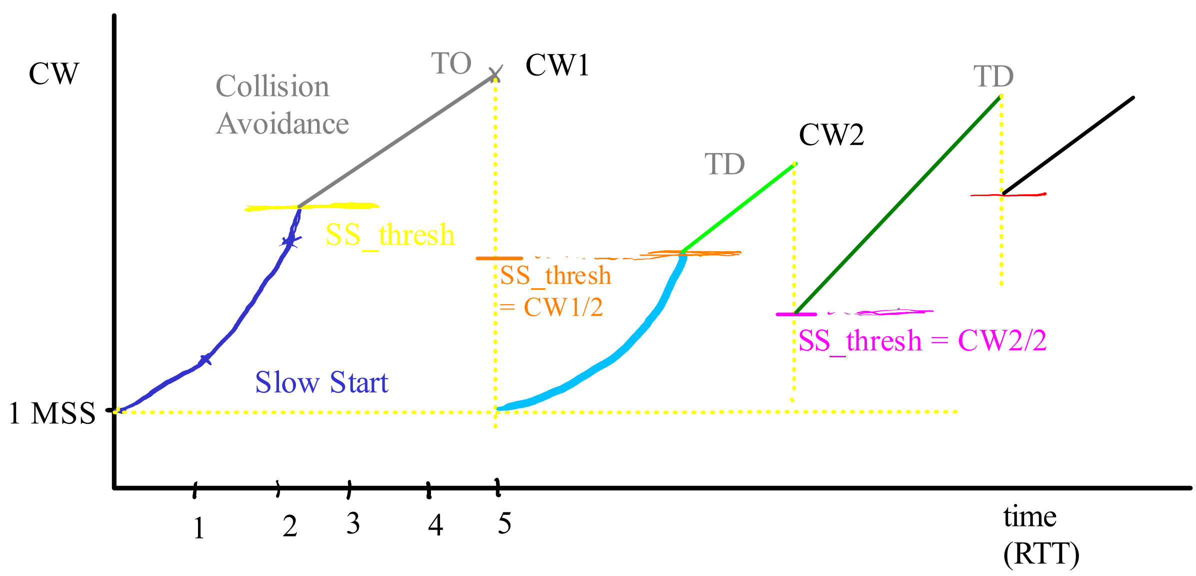
Should we reduce CW to 1MSS for every packet loss?

If we get a Timeout before a TD(Triple duplicate ACKs), it is considered a very drastic congestion (no DUP ACKS => pkts not getting through)

TCP RENO

Fast Retransmission: If 3DUP ACKs recieved -> Assume segment lost and retransmit it.
Fast Recovery: On getting 3DUP ACKs,

SS_threshold = CW/2
CW = CW/2



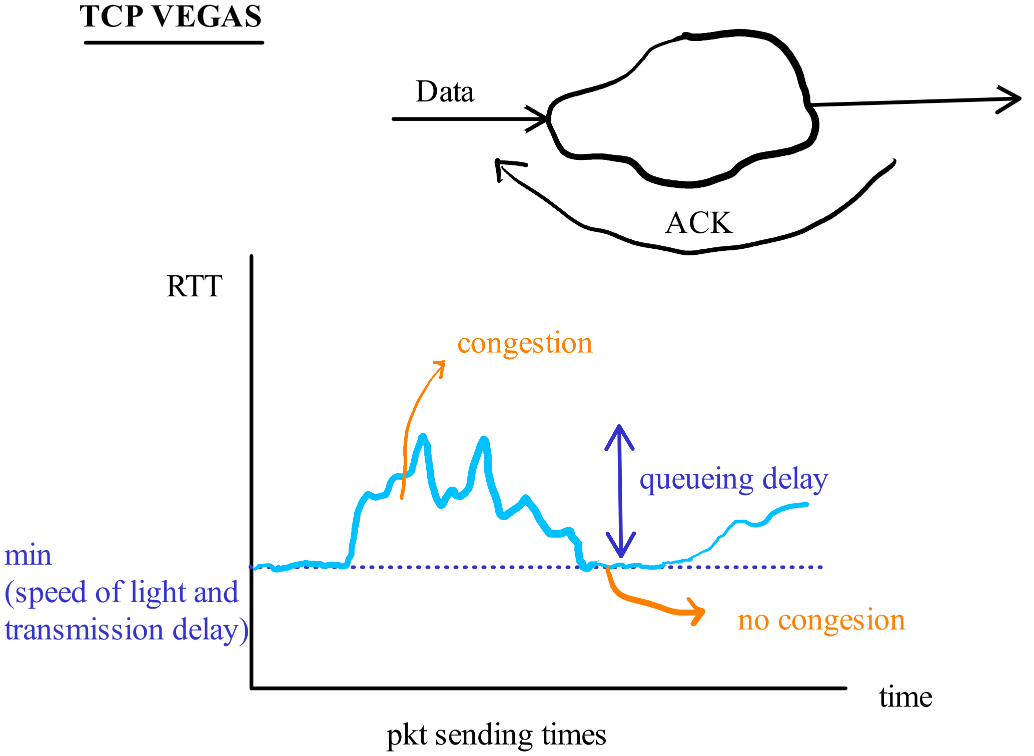
Issue with Tahoe and Reno:

1. Designed to cause congestion.
2. It increases the jitter.
3. Overall OWD and RTT increase

This can be problematic for some other applications.

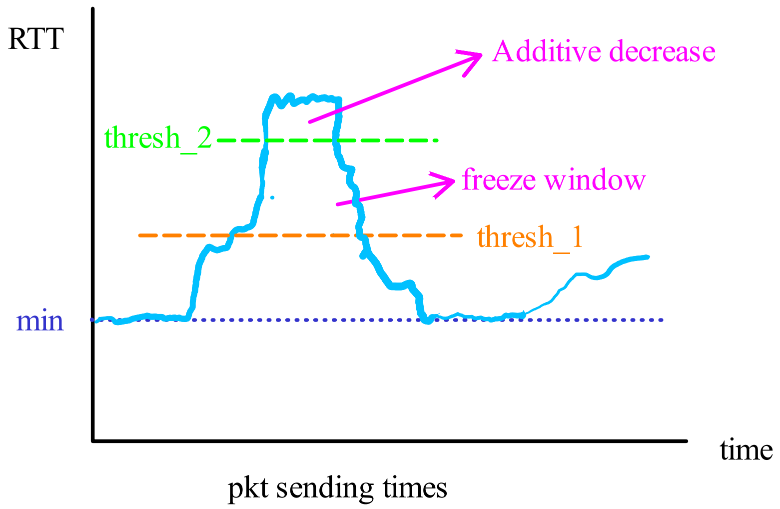
Good thing: They are designed to utilize the entire bandwidth.

TCP VEGAS



Their idea was:

1. Begin with SS as in Reno
2. In case of TO, behave as in Reno
3. In case of TD loss, behave as in Reno
4. Change congestion avoidance (CW >SS_thresh).



VEGAS RULES (for congestion avoidance)

- BaseRTT = min observed RTT in some recent time window.
- RTT = current (smoothed) estiate of RTT.
- W = Window

Suppose no congestion, then RTT = BaseRTT.

- ExptRate = W/BaseRTT
- ActualRate = W/RTT
- Diff = ExptRate - ActualRate = W (1/BaseRTT - 1/RTT) (>=0 because RTT>=BaseRTT)

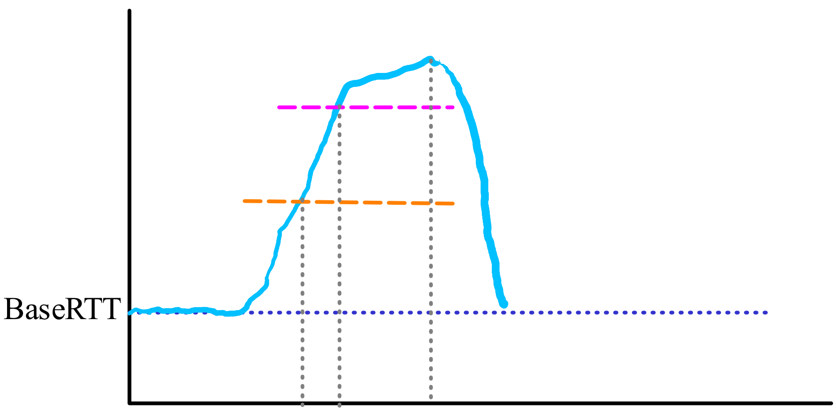
= W (RTT - BaseRTT)/(BaseRTT. RTT)

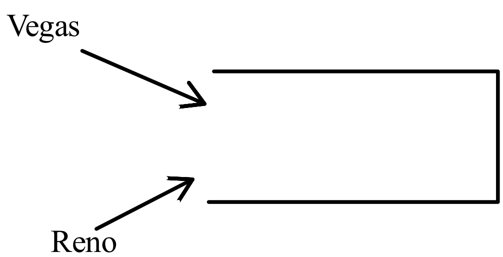
Queueing delay

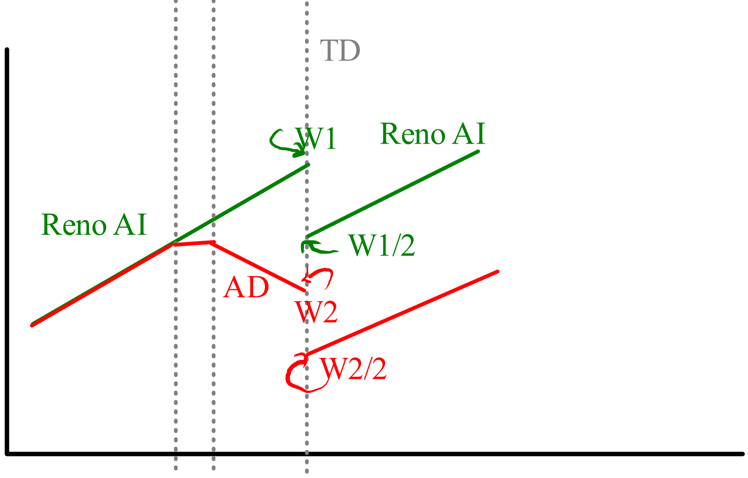
- If Diff < α, Additive increase as in Reno.
- If α < Diff < β, Freeze the window
- If β < Diff, Decrease W by 1MSS per RTT.

They suggested α = 30kbps, and β = 60 kbps

Can TCP Vegas and TCP Reno co-exist together?





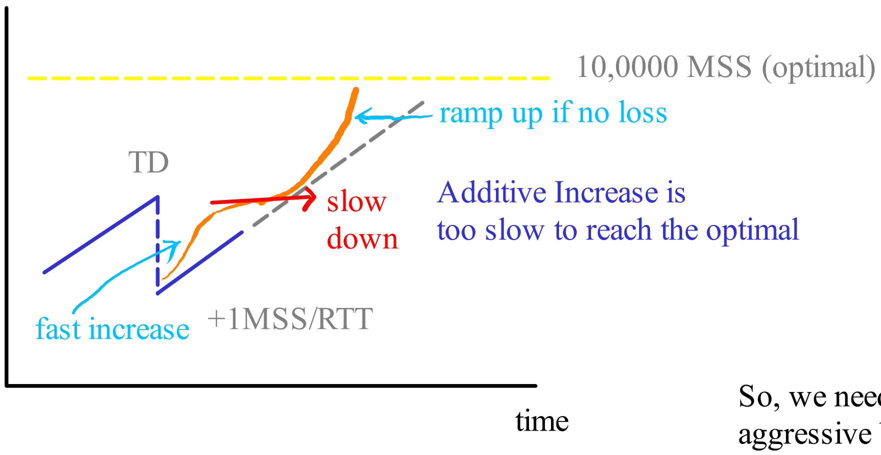


TCP Reno -> 1988-2012 (default by many OSes)

TCP CUBIC is the default now (Linux/MacOS)

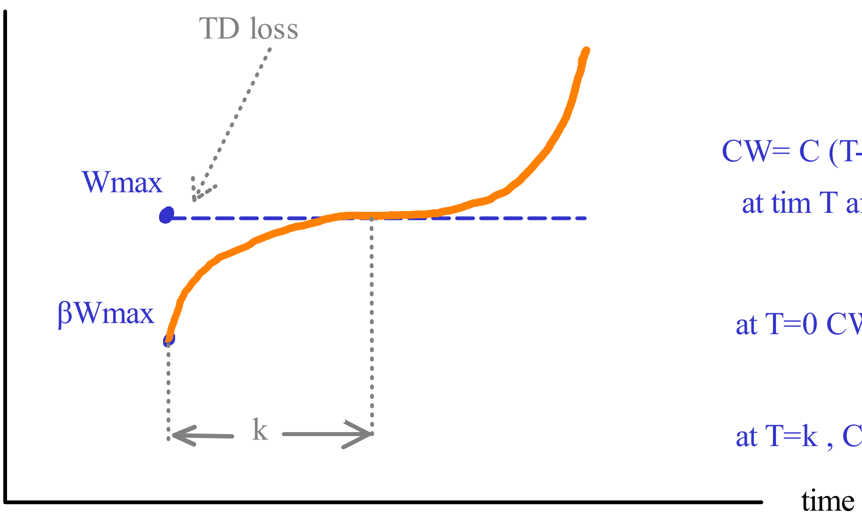
Why do we need CUBIC? Why was RENO not good enough?

Because we have HIGH-SPEED NETWORKS.



So, we need to speed up, and become aggressive but we shouldn't end up killing the RENO flow.

So need to increase fast, and then slow down



$CW = C\,(T-k)^3 + W_{max}$

at tim T after loss events.

$$K = \left[\frac{W_{max}(1-\beta)}{C}\right]^{1/3}$$

at T=0 $CW = -W_{max}(1-\beta) + W_{max}$
$= \beta\, W_{max}$

at T=k , $CW = W_{max}$

$\beta = 0.7$, $C = 0.4$

APPLICATION LAYER

If you don't want all properties of TCP, we need to develop own transport protocol in the Application layer on the top od UDP. Example. QUIC by Google, built on top of UDP, with congestion control (like TCP CUBIC).

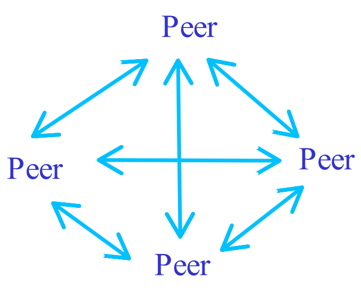
There are huge innovations in Application Layer, and the potential is also big.

Lot of the Applications is based on the Client-Server model, where the Server is offering some service, and the client utilises it. Ex: Email, Search, WhatsApp.

PEER TO PEER NETWORKS

Peer to Peer (P2P) networks are different from Client-Server. In Client-Server, there is some assymetry. But in P2P the hosts are equals. Example. Sharing music. A sort of Give-and-Take.

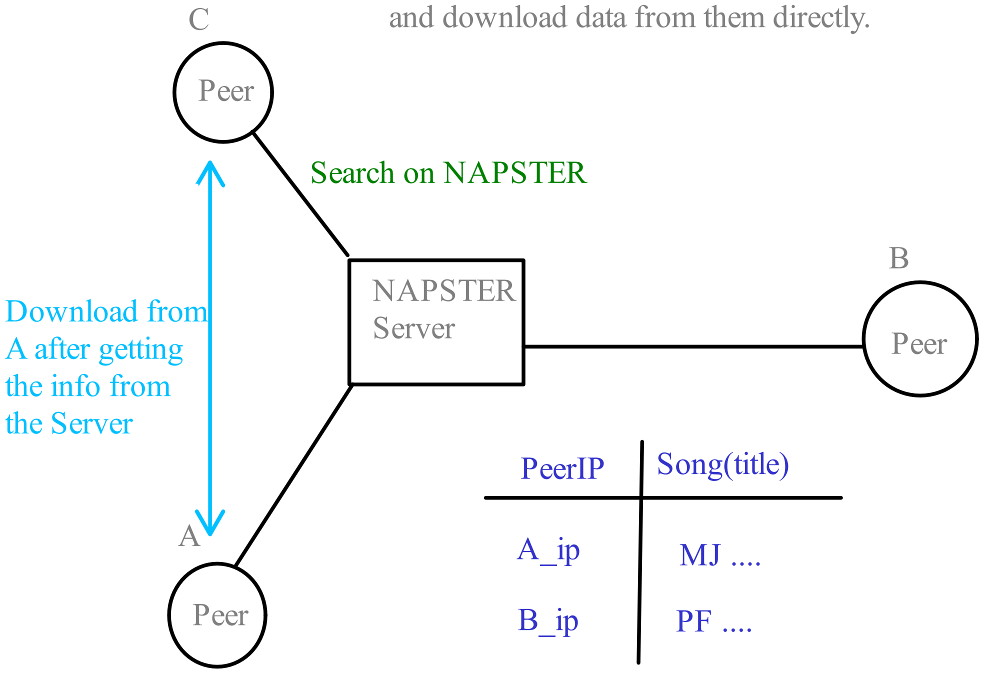
The web (HTTP) was developed in early 90's by High Energy Physics Researchers, who did particle accelator experiments (CERN). Their labs around the world would develop a huge amount of DATA, which they wanted to share.



1st Generation of P2P networks - NAPSTER

How to do a serach in P2P network? How to know who has the data we want?

Napster had a server. So, if you are a Peer, and wanted to participate, we could connect to Napster Sever. The server would maintain a list. So, if you need some music, query the server and get the info about the peer who has the music and download data from them directly.



PeerIP	Song(title)
A_ip	MJ ....
B_ip	PF ....

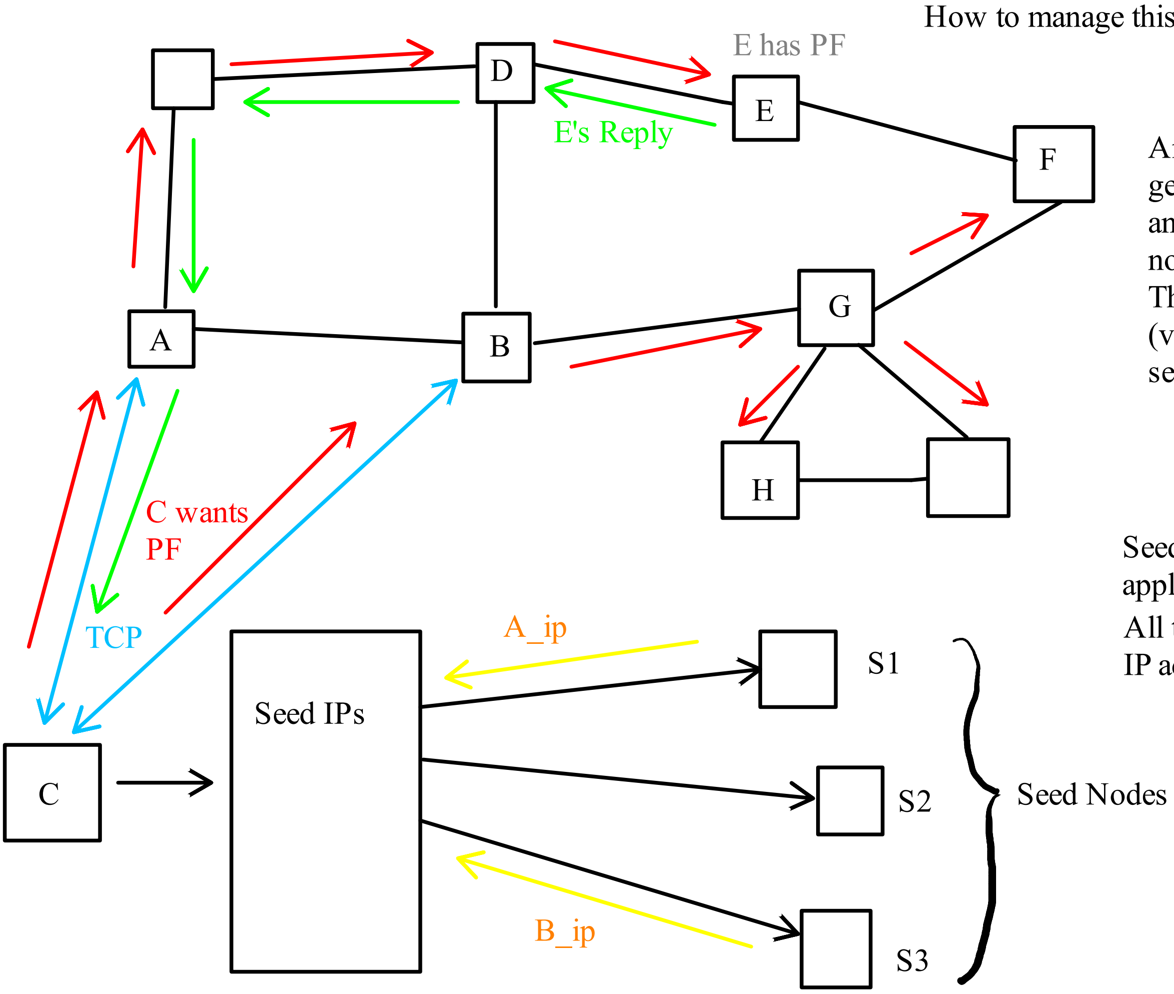
The Legality of NAPSTER was question in court, they eventually lost and had to shut down :(

Problems with Centralised Server:

Single Point of Failure :
- (technical) :
  - a. power failure
  - b. DDoS Attack (to bring down by a server by sending a lot of TCP queries)
- (legally) :
  - If there was any legal issues, the govt could force to stop it

2nd Generation: GNUTELLA

Anyone can join and leave at any time.
How to manage this without a centralised server?



Any newly joined node will get info from seed nodes, and will connect to some node in the network.
The IP for this connection (via TCP) is given by seed nodes.

Seed Nodes info is present in application software.
All the Nodes are aware of the IP addresses of Seed Nodes.

Napster gave both IP and what info is stored at that IP. But seed not doesn't give what info is stored. It just gives IP.
How do we know what info is stored in some node?
C gets connected to A and B, and can know what is in A and B. But may not know what is in D,E,...

Say C is doing a serach, then this search is broadcasted. Ex. C is seraching for Pink Floyd(PF) which E has.
E will reply to whoever has forwarded the request to it. The reply will have E's IP. The reply will then be forwarded back and it get backs to C. Then C can set up a TCP connection with E and download the file. Then E and C would become peers and will keep the connection alive (new edge in the graph).

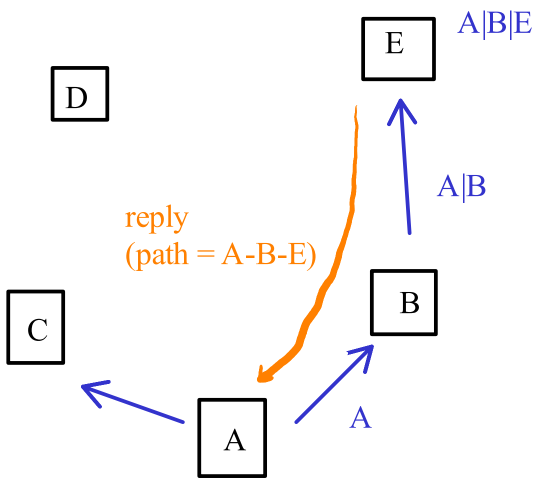
If C gets multiple replies, it can choose to download from whom.

Expanded Ring Broadcast

Broadcast to depth "n". n= 1 means only send to your neighbours, etc. It is part of the message like a TTL, which is decremented at each step. Increase "n" if no reply.

One of the issues with GNUTELLA is that we are doing broadcast, which is not efficient. So, we are bothering thousands of peers for some file which maybe very few peers have.

ASIDE: MOBILE AD HOC NETWORK (MANET) -



tanks in a battlefeild

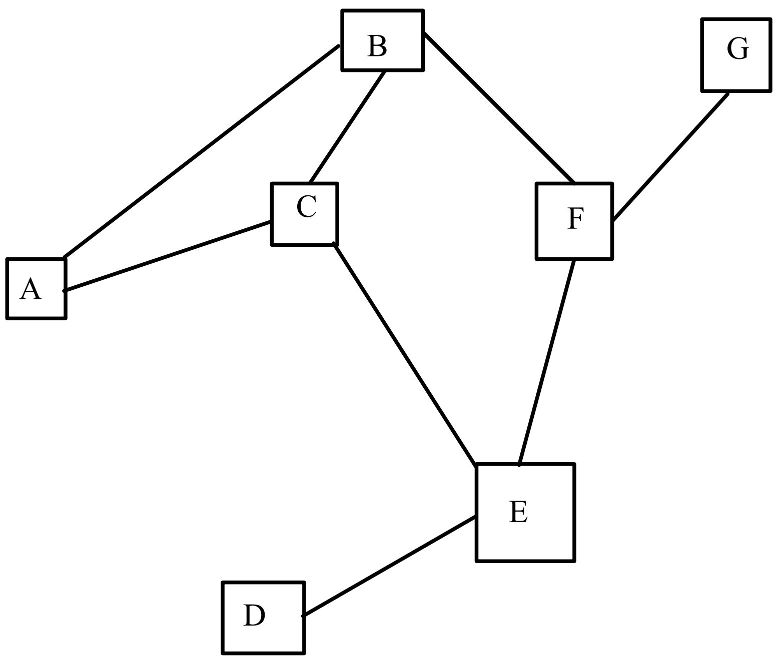
These tanks keep moving. So the path between nodes might break down often.

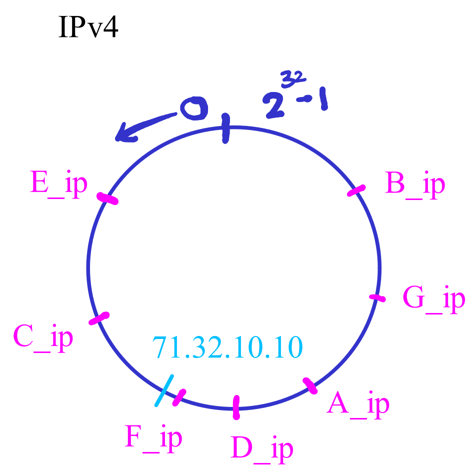
So, in this case, the sender will do broadcast and find a path.

If A is searching for E it sends a search message, and as the message propagates, the path gets added. E will then reply along the path after it recieves the full path.

Here we use Source Routing. If A sends a message to E, it specifies the full route from A to E in the message

3rd Generation: DISTRIBUTED HASH TABLE (DHT)



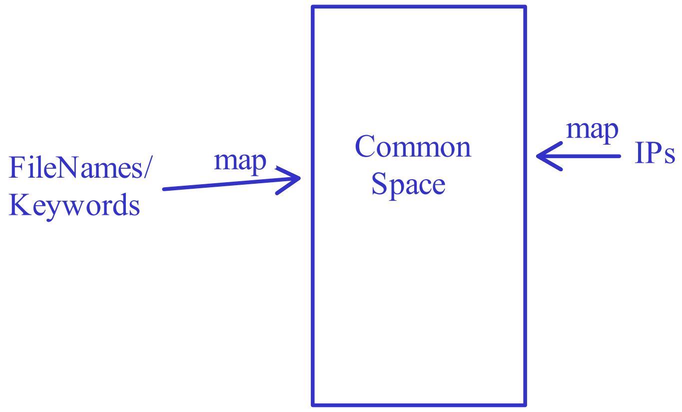


A will send a Search request to its neighbour closest to the IP 71.32.10.10.
So, A sends to C. C will try to send to its neighbour which also closest to the given IP. But E and B are taking it farther from the IP.

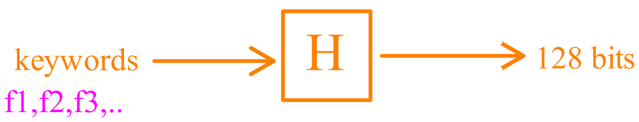
If we had an edge between C and F, this should have worked.

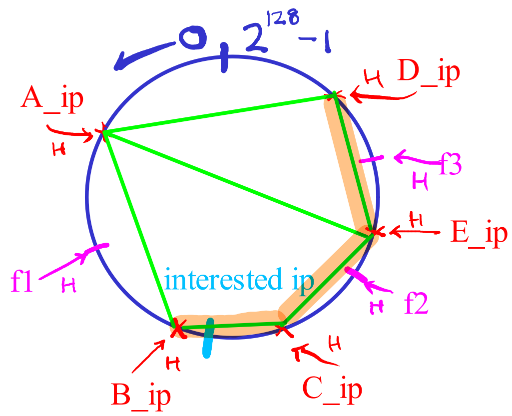
Idea: Maintain a mapping from FileNames/Keywords to IP addresses.

But people designing the netowk weren't happy about IP addresses because, IP addresses might not be well distributed. So they decided to have a common space, and map file names to it, and also IP addresses.



For this, use a hash function, H.





If D wants to know who has file f4. Map f4 to this space and try to find out which node has IP address closest to f4.

If B know where f4 is then we are done.

At a high level:

Suppose a node A has a file,"f". A finds node (say B) whose IP address is closest to "f" in the common space. (This means that the dist(H(f),H(B_ip) <= dist(H(f),H(X_ip)), for all nodes X in the network).

A tells B that it has "f". B will mainatain a table which maps file "f" to node "A_ip".

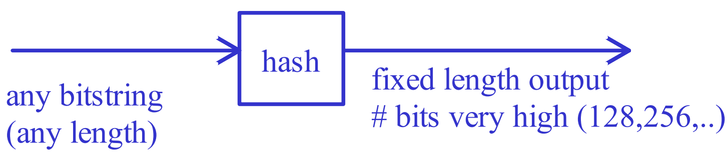
Suppose somebody else (say C) wants file "f". C finds the node which is closest to "f" in the common space. Here, it will be B.

C asks B: "Who has "f"?". B gives A_ip to C. C can now connect to A and download the file "f".

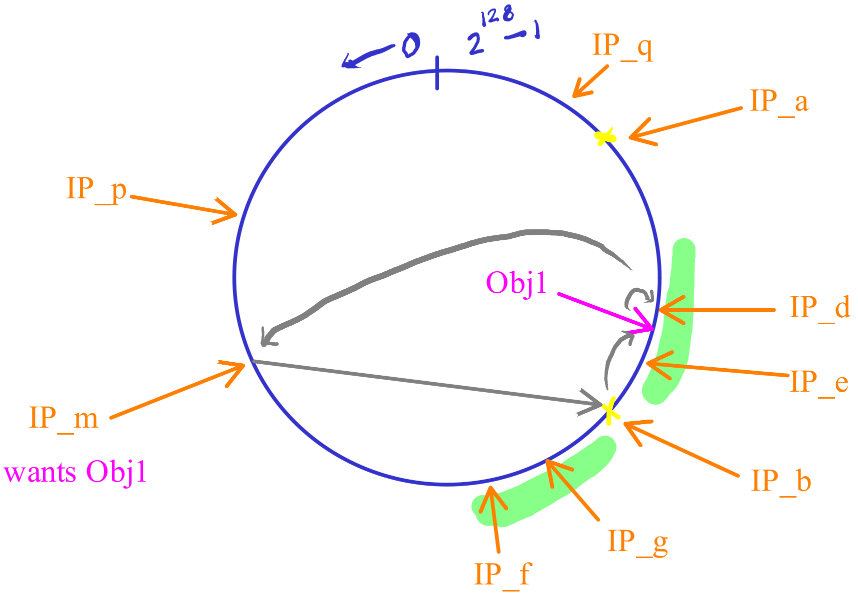
RECAP OF P2P SO FAR

NAPSTER - too cnetralised
GNETELLA - Limited broadcast, O(N) nodes may get the query

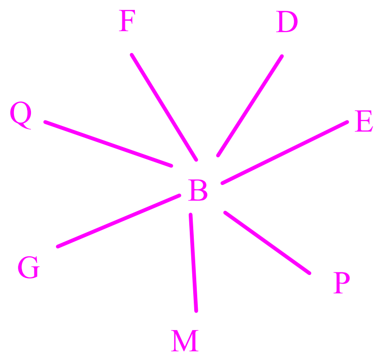
Can we do better? Can we get O(log N)? Yes, Using DHT.
We map the hash of IP and hash of keyword (ObjID) to a common space, where we can search much efficiently.



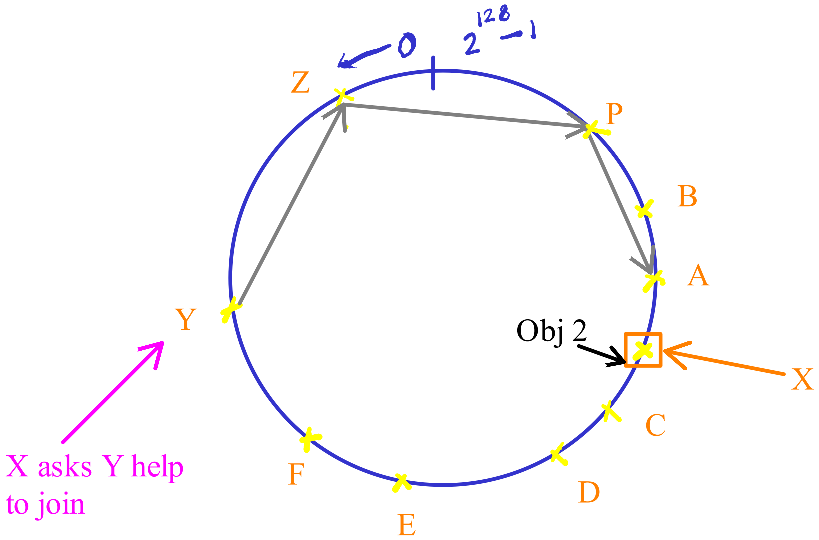
PASTRY



LEAF SET : Neighbours in P2P networks of any node



Suppose a new node X wants to join the network.

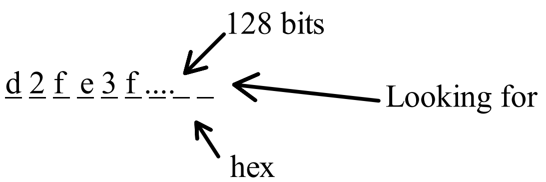


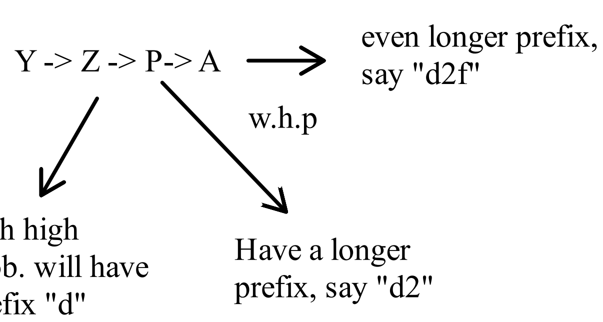
- X contacts some other node already in the network, here Y.

- Y "routes" this message to node closest to hash(IP_x).

- Each node on the Path (Y,Z,P,A) become part of the leaf set of X.

- A shares info about L/2 nodes on the left and right. These nodes join leaf set of X.

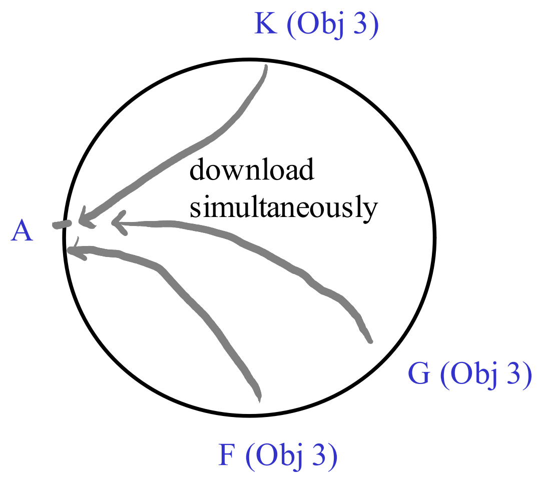




Consider Obj 2. Earlier C was closest to it and had info about Obj2. After X joins, it should have the info. So, X queries its neighbours, get the info and store the info as needed.

Suppose one node, say A fails. How will X know that A has failed?

BIT-TORRENT



DOMAIN NAME SYSYTEM (DNS)

The common IP addresses are of the form 32.75.5.9... etc which are not Human friendly.
We are more comfortable with common urls likes "google.com", "iitb.ac.in", etc. But these urls are built into lower layer protocols.

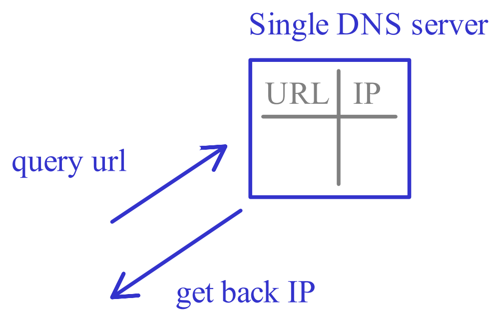
DNS maintain a mapping from IP to urls.

Say google.com has IP as 8.8.7.5 now, but will change the IP tommorrow. Our DNS must be able to identify and change mapping correspondingly.

Bringing down DNS can thus bring down the internet. So, DNS has to be very very robust.

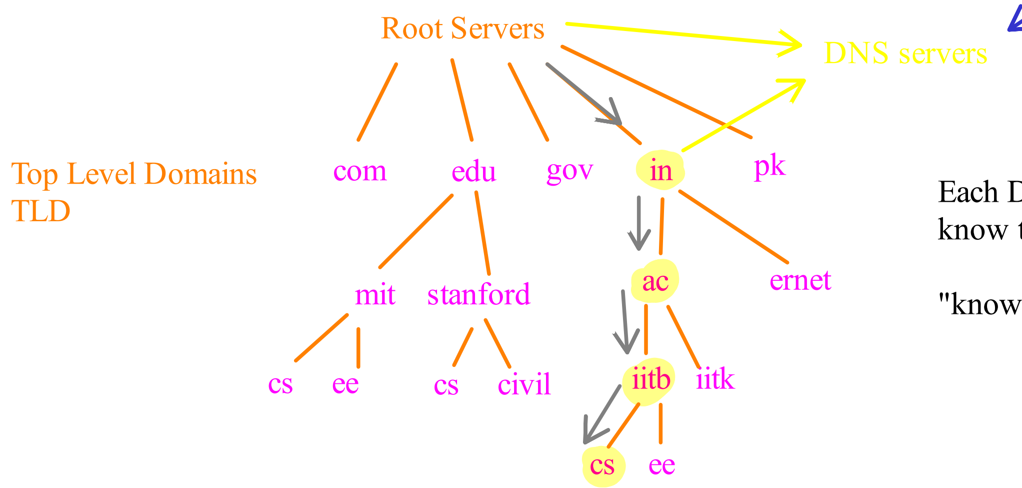
Naive approach

There is a single server and everyone queries this to get the IP. But this is not good as it is a single point of failure. Moreover it is not scalable.



Hirerarchical Solution

So, it was proposed to have some hirearchy.



Each DNS server of parent domain should know the DNS server of the immediate children

"knows" - know the name of server, IP address

If I need to change the IP of some machine say "surya.cse.iitb.ac.in", I could just change in "cse" DNS server.

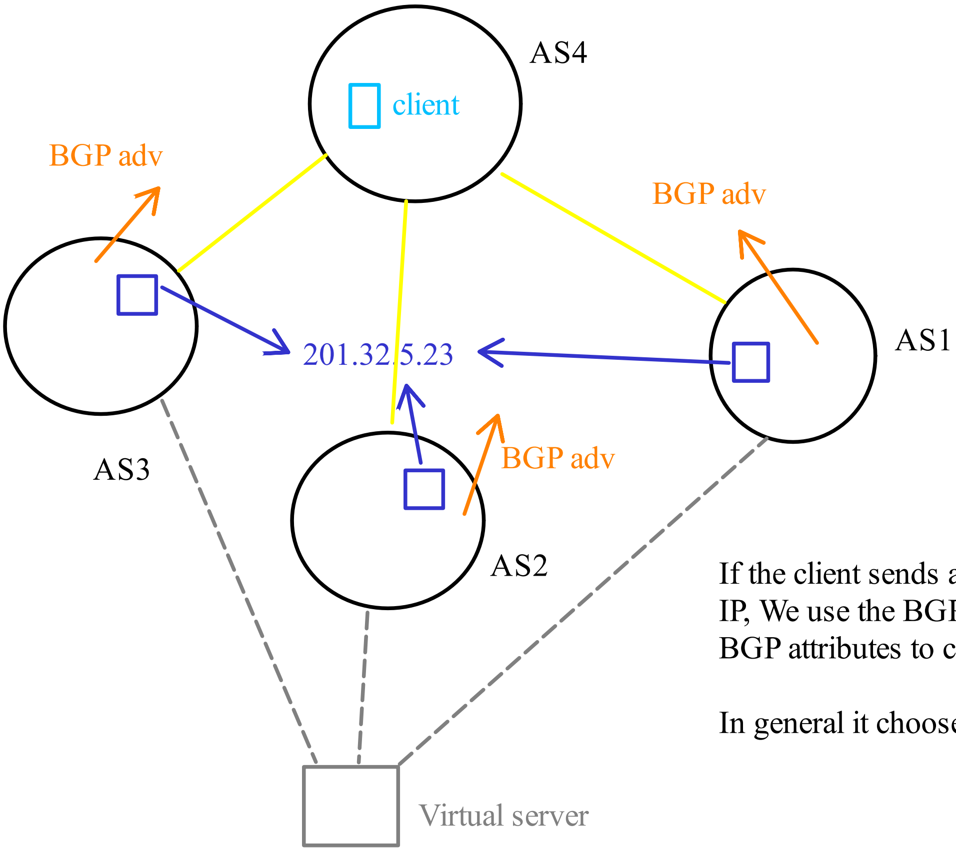
ROOT SERVERS

Initially, DNS packet sizes were of 512 bytes. (The packets set for DNS query and response).
They finally chose 13 Root Servers, since the info fits in 512 bytes.

Names:	IP	
A.root-server.net	-	
B.root-server.net	-	"root hints" file
:	:	
:	:	
M.root-server.net	-	

Clearly 13 is better than 1. But 13 may not be enough. What if someone brings down all 13?

Each of the names (A.root.server.net) is not a single machine - they are multiple machines with same name.
Say its IP is 201.32.5.23. How to have multiple machines with same IP? How will Layer 3 work?



If the client sends a packet targeted this IP, We use the BGP rules using BGP attributes to choose the host.

In general it chooses the shortest AS-path

ANY CAST

UNICAST - single destination
BROADCAST - all nodes are destination
MULTICAST - a subset of nodes are destination
ANYCAST - any one machine in a group is destination

RESOURCE RECORDS

<Name, Value, Type, Class, TTL>

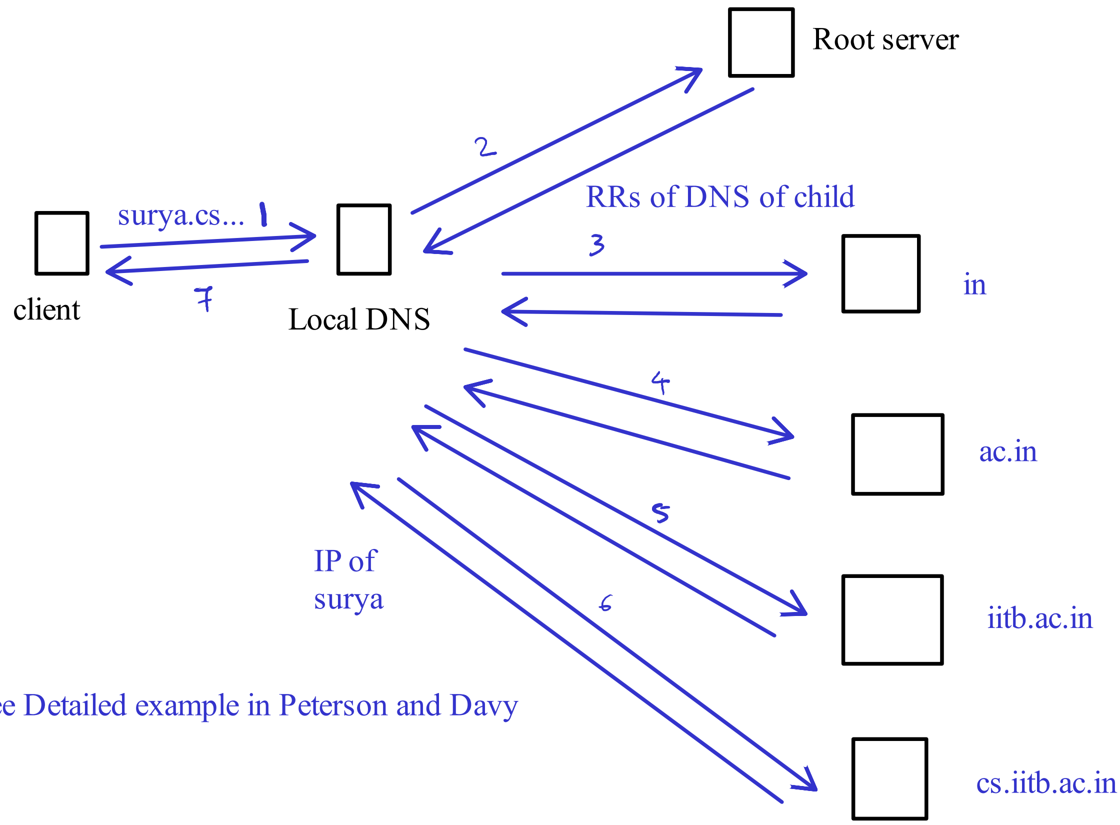
- Name: eg: surya.cse.iitb.ac.in
- Value: type of the value usually IP address
- Class: IN(internet)
- TTL: how long is the record valid

Types:

- A :- IP adrress
- NS :- Name of the DNS server of the domain in "Name" feild in the record
- CNAME :- Canonical name (alias)
- MX :- Name of Email server of the domain in "Name" feild in the record

Ex: 1. <iitb.ac.in, dns.iitb.ac.in, NS, ...> means that dns.iitb.ac.in is the name of the DNS server of iitb.ac.in
2. <surya.iitb.ac.in, 138.56.7.34, A, ..>
3. <cs.iitb.ac.in, mail.cs.iitb.ac.in, MX, ...>
4. <mail.cs.iitb.ac.in, 138.56.7.45, A....>

Ex for CNAME: "cs.mit.edu" and "ee.mit.edu" are aliases

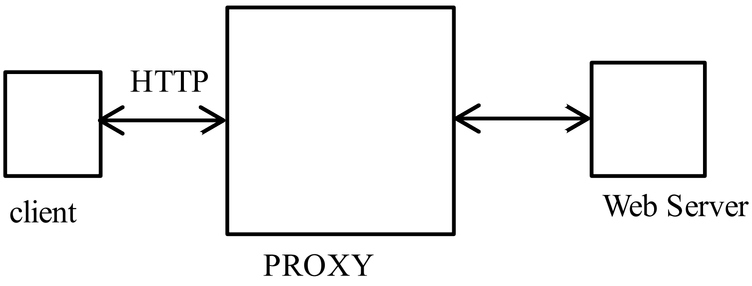


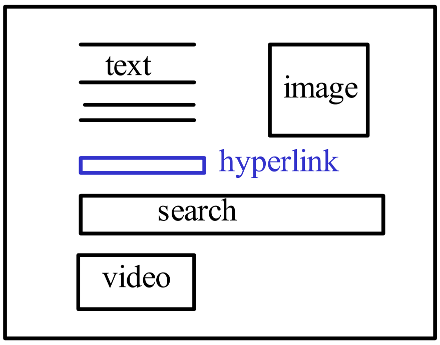
See Detailed example in Peterson and Davy

Hyper Text Transfer Protocol (HTTP)

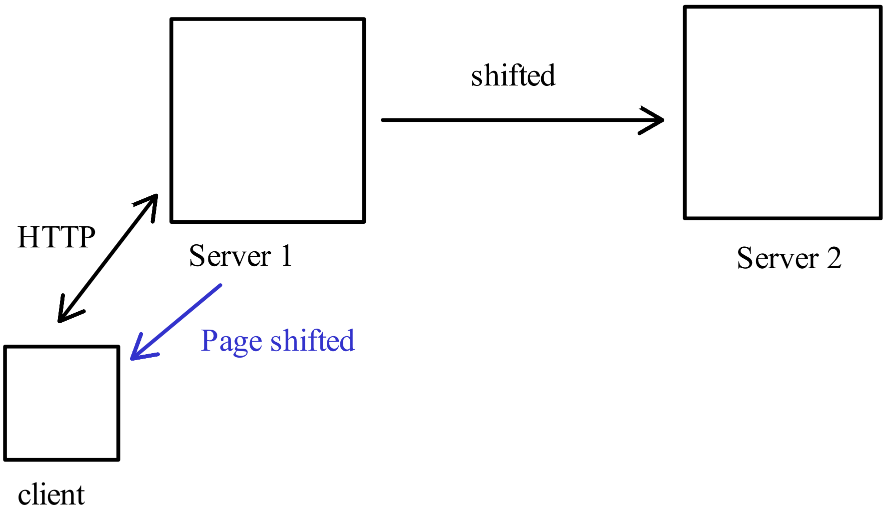
HTTP help us download a webpage or share or submit information with a webpage

HTTP also allows things like PROXIES.





Advantages of proxy:
1. Proxies can maintain a cache. If two clients are using the same Proxy, and need to download the same webpage Proxy can directly send the page (stored in cache) after downloading once.
2. Proxies also ask as some sort of firewall



HTTP versions 1.0,1.1,2 use TCP while HTTP 3 (came out in 2022) uses QUIC over UDP

HTTP message format

2 ASCII chars

(STARTLINE) .... <CRLF>

(MSG HDR) .... <CRLF>
.... <CRLF>

HTTP and TCP interaction

PORT used in HTTP is 80 while HTTPS it is 443

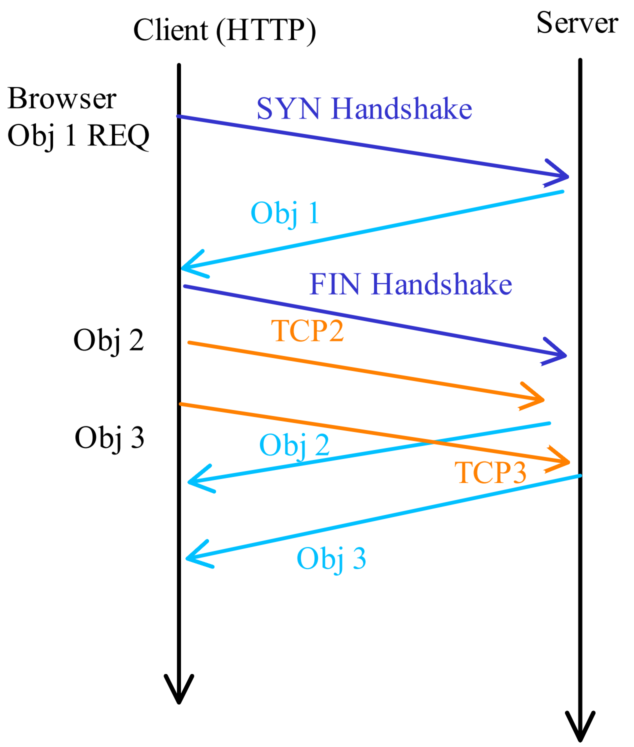
obj 0

obj 1
obj 2
:
obj n

Browser
↓
HTTP
↓
TCP

Send all requests at once or separately?
One Request per TCP connection?

HTTP 1.0 : New TCP connection for each "GET"



Issues of creating newer TCPs:
1. Each TCP connection takes time to learn optimal CW
2. Overheads (some time wasted in Handshake)
3. Server state is large due to multiple TCP connections being open