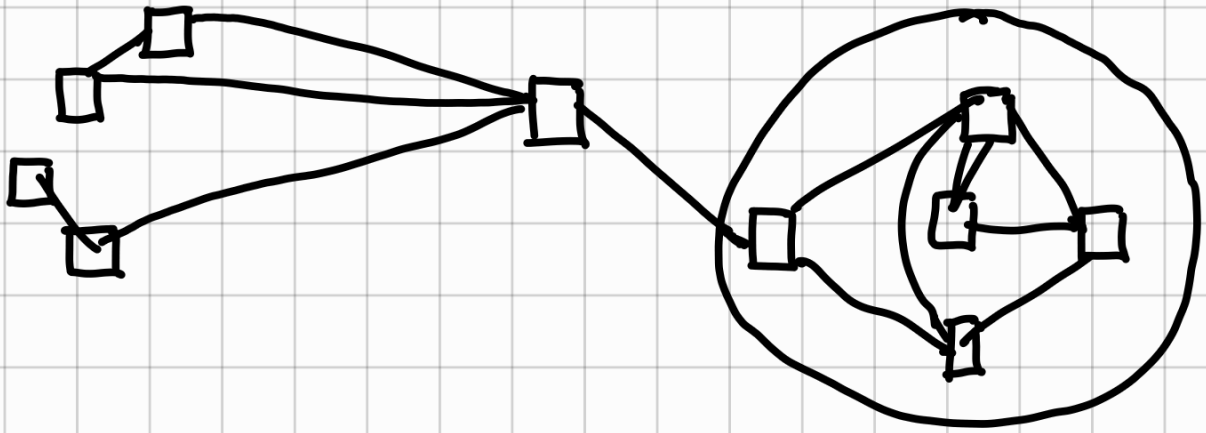
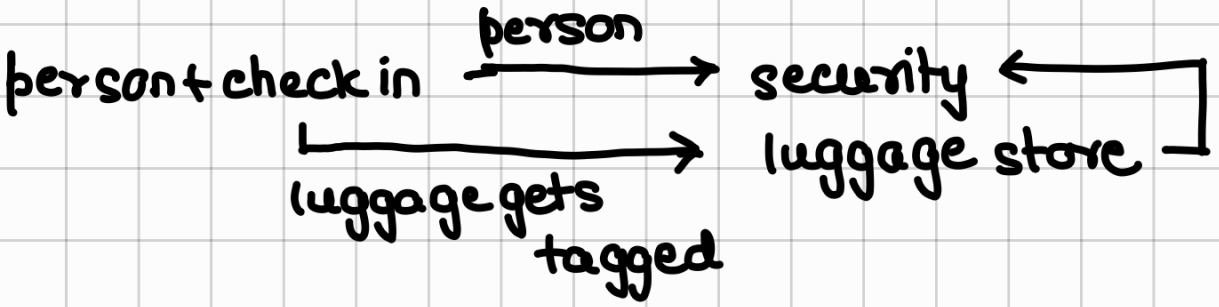
Rules governing a network **Protocol**



Computer Networks A network where nodes are computing devices (computers, cars, printers, ...) and a link between two node indicates possibility of communication.

Analogy to Airline Communication

1. Ticket/boarding pass of a flight from A1 to A2
2. Luggage
   - only person → security
   - person + check in
   - person + carry on → security
   - person + checkin + carry on



(If travelling with data of different forms, data might be broken and sent through different channels, and, then be repacked)

3. After security, person goes near the Gate.
4. Airplane on a fixed runway

5. Airplane routing
6. Landing on a specific runway
7. Exit from plane through a specific gate
8. Luggage
9. Ticket Enquiry

1. Entry Pass	10. Exit assurance
2. Luggage	9. Luggage
3. Gate Management	8. Gate Management
4. Runway management	7. Runway Management
5. Routing —— Routing 5½ ——	6. Routing

Both the sides are symmetric

5 Layer Protocol

Application
Transport Layer ≡ Division of Data
Network Layer ≡ Gate Management
Link ≡ Runway Management
Physical Layer

OSI Layer is 7-Layer Protocol instead of 5-Layer Protocol

Internet Layering Protocol

Application Layer
Transport Layer
Network Layer
Link

Application Layer
Transport Layer
Network Layer
Link

Physical Layer

OSI Mode

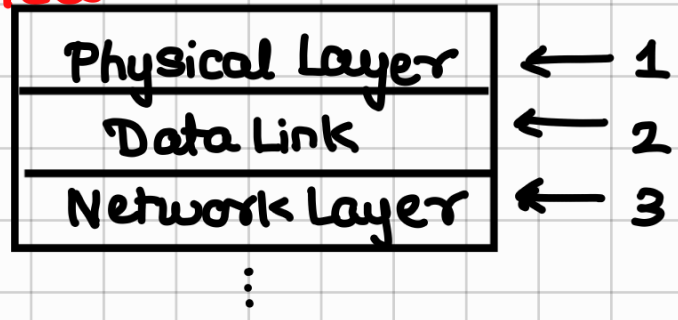
OSI → open systems Interconnection

7-Layer Model

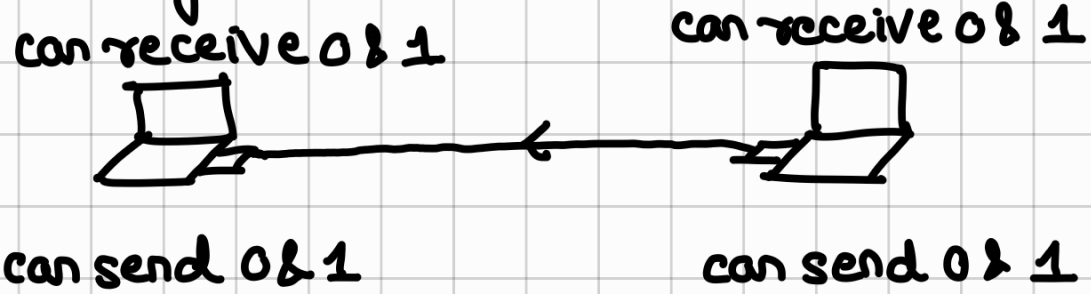
1. Application
2. Presentation — Data compression and encryption
3. Session — Data delimitation and security
4. Transport
5. Network
6. Link
7. Physical

- Presentation ≡ Baggage Management
- Session ≡ Removal of Banned Objects and security

Bottom Up View

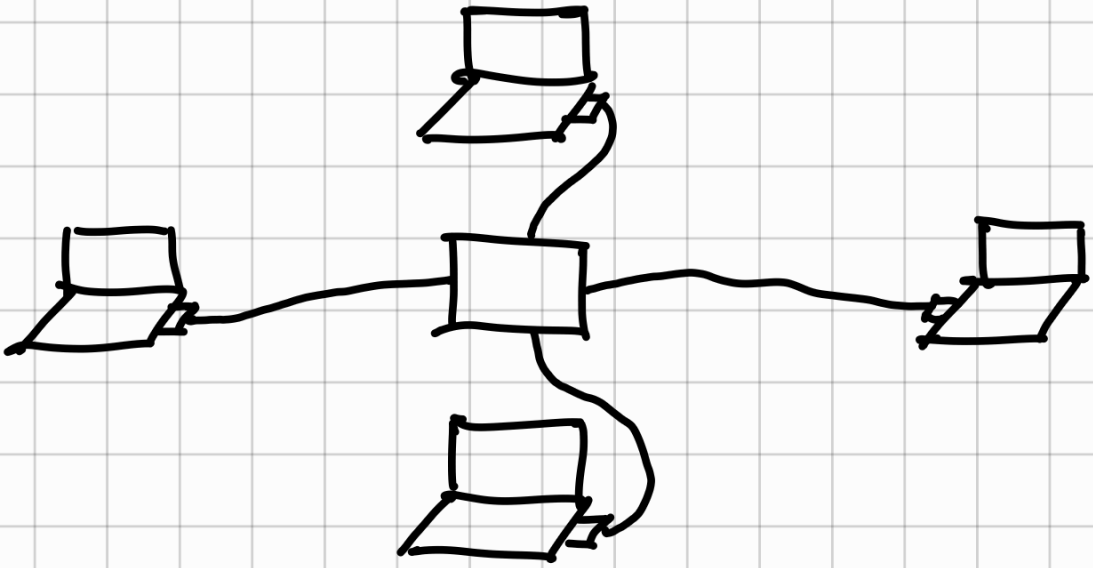


Physical Layer Through this raw data (seq. of 0's and 1's) get transmitted



Physical Layer has no media management
If both simultaneously send signals then they will

collide.



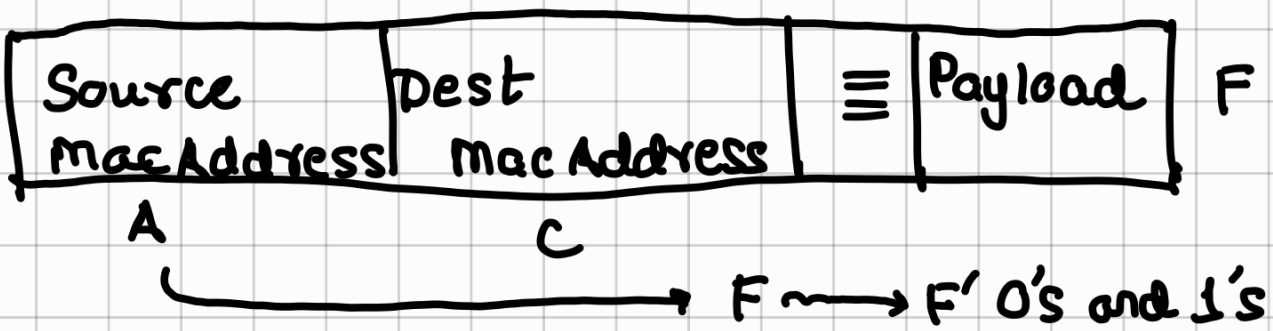
If two computers simultaneously send signals there will be collision

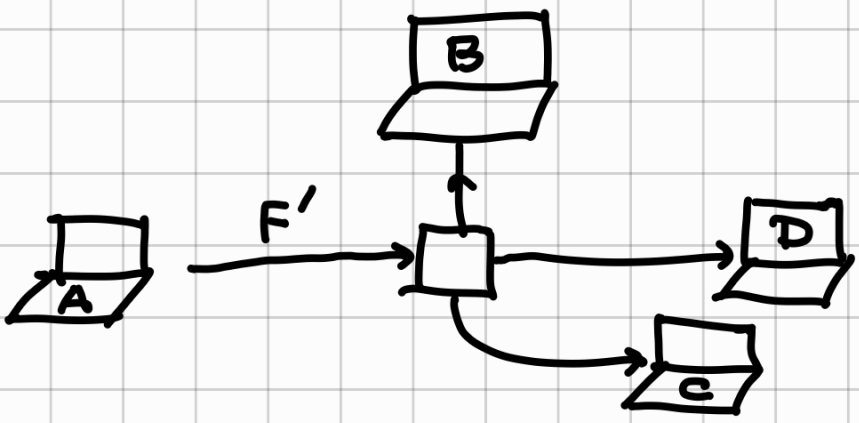
Data Link Layer

Mac Address unique manufacterers given code to every device

32e:ffc:7xx77..... 75

Data Link Layer's create "Frames"





B,C,D all get F'
All push it to their own DLL
- DLL's decide whether the message is for them or not. If yes, it reads, otherwise junks
- So only C reads the message.

Making the communication more secure,

Circuit Switching A fixed communication channel is created b/w two and that is used for continuous interactions

Advantages

1. Consistency
2. Routing not needed
3. Faster
4. Less Redundancy
5. More secure

Disadvantages

1. $O(n^2)$ channels needed for n people
2. Adding new user to a 2-way channel is very hard

Packet Switching

Data is broken into small parts and potentially broadcasted

Graphs graphs are pairs $G=(V,E)$ where V is a set & E is a set of size 2-subsets of V.

A graph is directed if every edge has a source &

a target.

P. Kumar

CSMA: send signal/speak when channel appears silent

Limitations of CSMA (carrier sense multiple access)

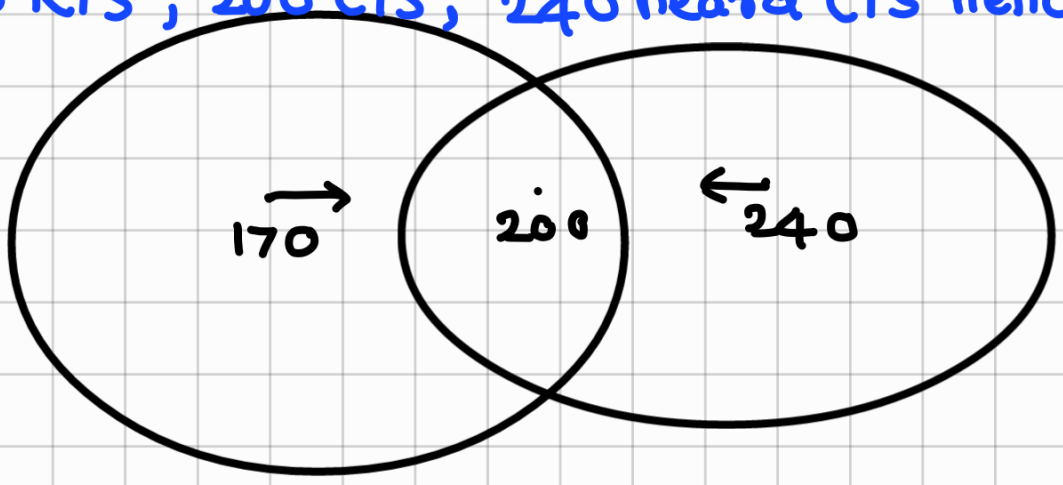
→ Transmission over limited distance:

Let transmission are heard over a distance of 50. A node at location 170 is transmitting for node at location 200.

Node at location 240 starts transmission since it finds the channel to be silent.

This is called Hidden Station Problem-1.

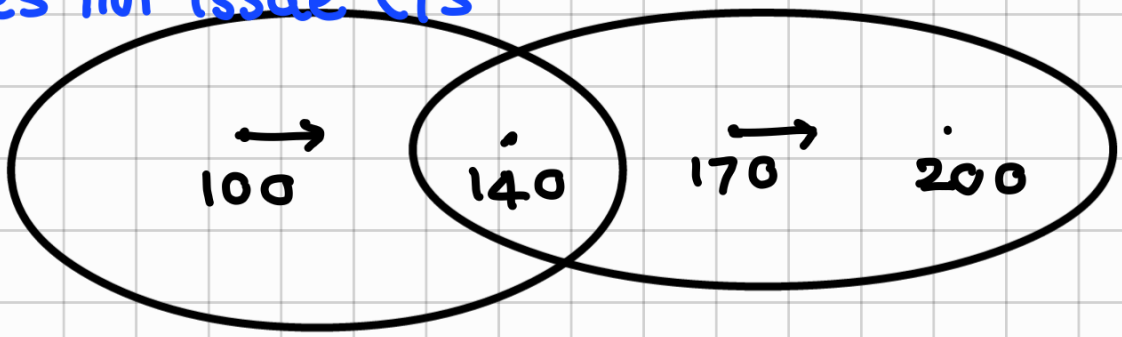
170 RTS; 200 CTS; 240 heard CTS hence no RTS



A node at location 170 is transmitting for node at location 200. Node at location 100 starts transmission for node at location 140.

Hidden Station Problem-2

170 RTS; 200 CTS; 100 RTS; 140 listening 170 hence does not issue CTS



A node at location 170 is transmitting for node at location 200. A node at location 140 wants to (doesn't) transmit for node at 100.

Exposed Station Problem

170 RTS ; 200 CTS ; 140 RTS (CTS of 200 not listened) ; 100 CTS (transmission of 170 does not reach 100)

Solution to the above problems

- When node wants to transmit, it issues RTS (request-to-transmit), if it does not hear any CTS.
- Listener issues CTS (clear-to-transmit) ~~if not listening any transmission~~.

{ Ye kaatne ke peeche hi reason he :) }

Few problems still persist.

— x —

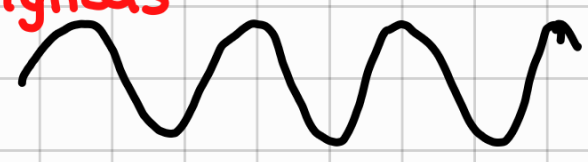
References

1. Computer Networks Andrew Tanenbaum
2. Data Communication and Networking by Behrouz Forouzan

Physical Layer Basics

2 types of signals → Digital (discrete)
→ Analog (continuous)

Analog Signals



Frequency
Amplitude
Phase
Wavelength

Digital Signals

Bandwidth (of communication channel)
The range of frequencies that go through that channel.

Types of Communication

1. Wired (e.g. classical Telephone)
2. Terrestial Wireless (e.g. Wireless Phones)
3. Satellite (e.g. Satellite Phone/Cable TV)

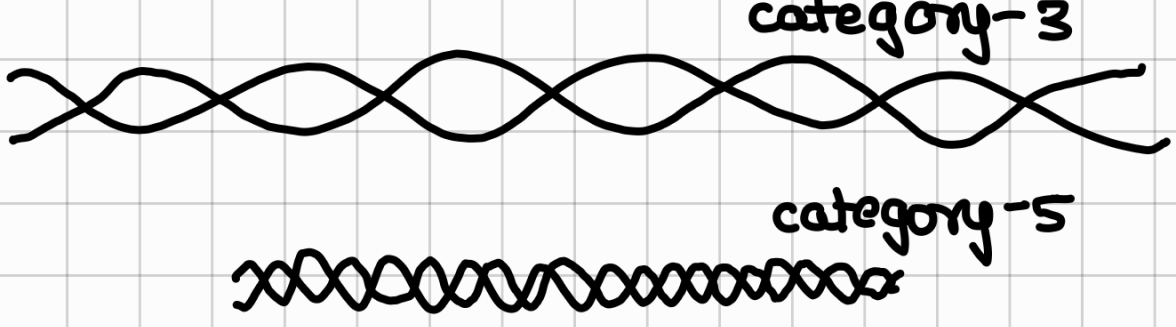
Better way to characterize,
1. Wired → Point to Point Communication
2. Terrestial Wireless, 3. Satellite → Broadcasting

Point to Point communication

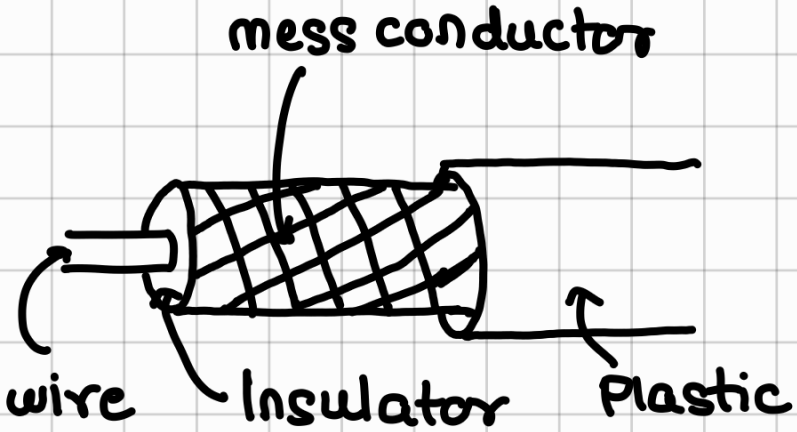
1. Magnetic Media:
   - Hard disk
   - USB
   - CD
   - Floppy disk

   } on an average, performing best

2. Twisted Pair wires:



3. Coaxial cables :



4. Fiber Optics

Processing power increased from 4.8 MHz to 2GHz. It increased 20 times/decade

Rate of communication

56 kbps ⟶ 1 Gbps

125 times/decade

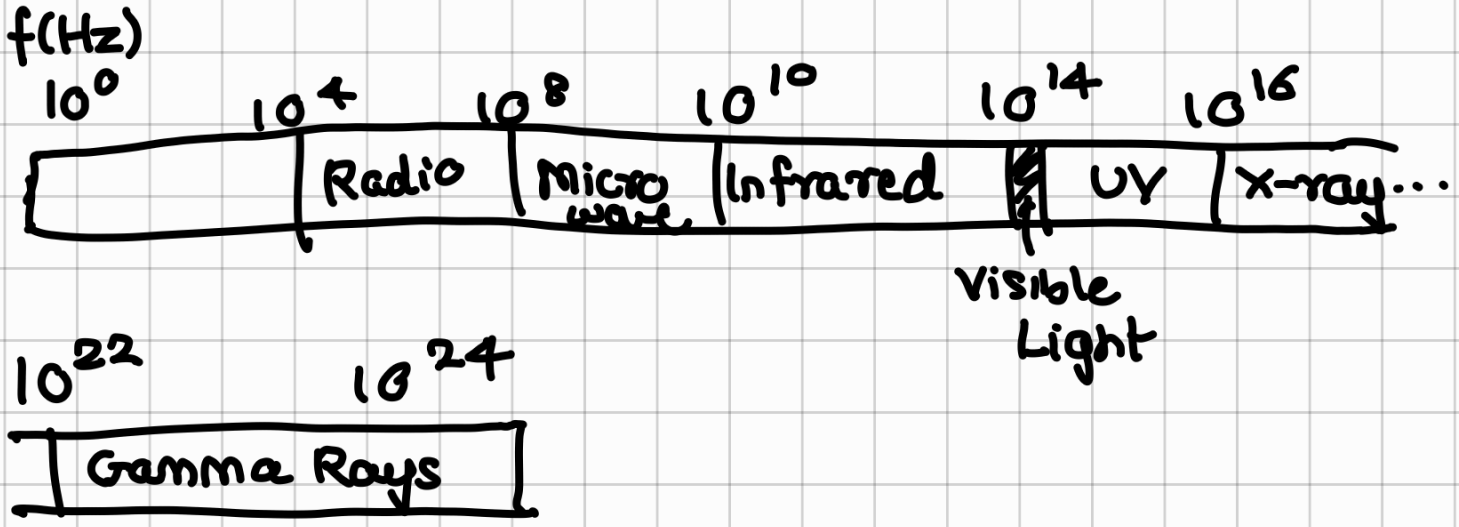
Advantages of Fiber optics :

→ More bandwidth

→ Much faster

→ Works for longer range.

Wireless Transmission



Radio Transmission

→ used by radio channel

1. High wavelength / small frequency
2. penetrates solid easily
3. Travels longer

Microwave Mobiles / TV

All the above hold albeit a little longer

Infrared TV remotes

can not penetrate solids

Visible Light Networking using LAN

easy to block

Satellites

1. Geostationary (farthest from earth & appears to stay fixed)
2. Medium-Earth orbit satellite (medium height & moves slowly)
3. Low earth orbit satellite (very close to earth)

1. Multiplexing
2. Switching

Separating two waves,

Fourier transform

$$g(t) = c/2 + \sum_{n=0}^{\infty} a_n \sin(2\pi nft) + \sum b_n \cos(2\pi nft)$$

Question : If $g(t)$ is given, then how do we find a_n, b_n, c?

$$\int_0^T \sin(2\pi n f t)\cos(2\pi k f t)\,dt = 0 \quad \forall\, n,k$$

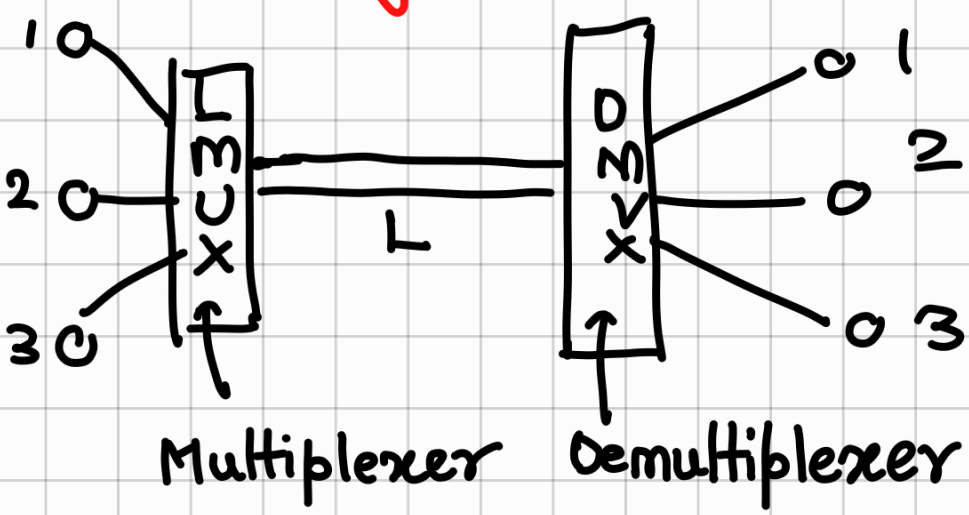
$$\int_0^T \sin(2\pi n f t)\sin(2\pi k f t)\,dt = \begin{cases} T/2 & n=k \\ 0 & \text{o.w.} \end{cases}$$

$$a_n = \frac{2}{T}\int_0^T g(t)\sin(2\pi n f t)\,dt$$

$$b_n = \frac{2}{T}\int_0^T g(t)\cos(2\pi n f t)\,dt$$

$$c = \frac{2}{T}\int_0^T g(t)\,dt$$

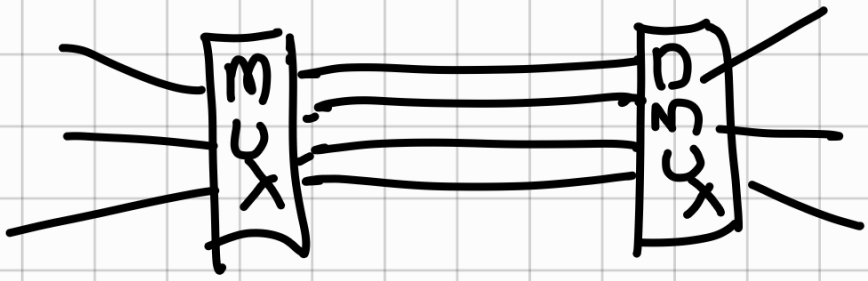
Multiplexing



1,2,3 want to send message to 1', 2', 3' simultaneously through L

Frequency division multiplexer

useful for analog signals



- Each wave is assigned a separate bandwidth
- Some space is kept between bandwidths

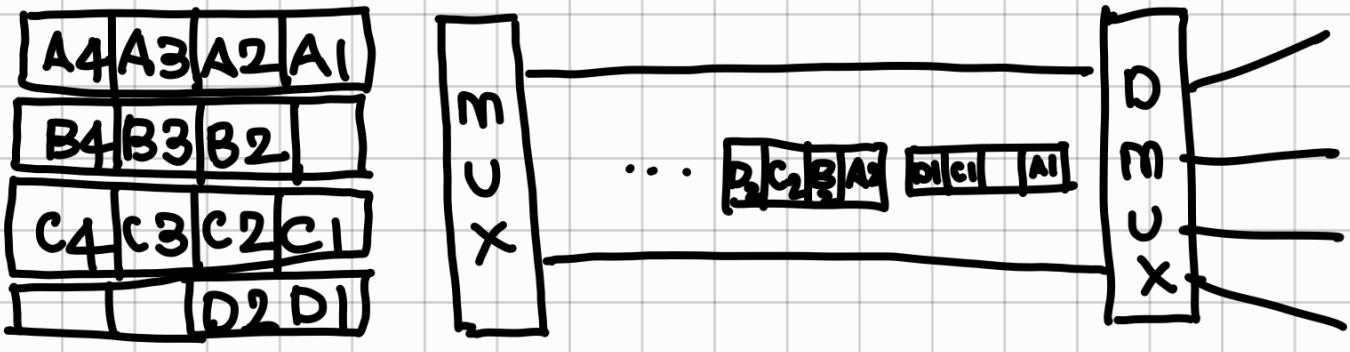
Note: There is a variant of this called wavelength division Multiplexing used for fiber optics mostly.

Time DVision Multiplexing

useful for digital signals

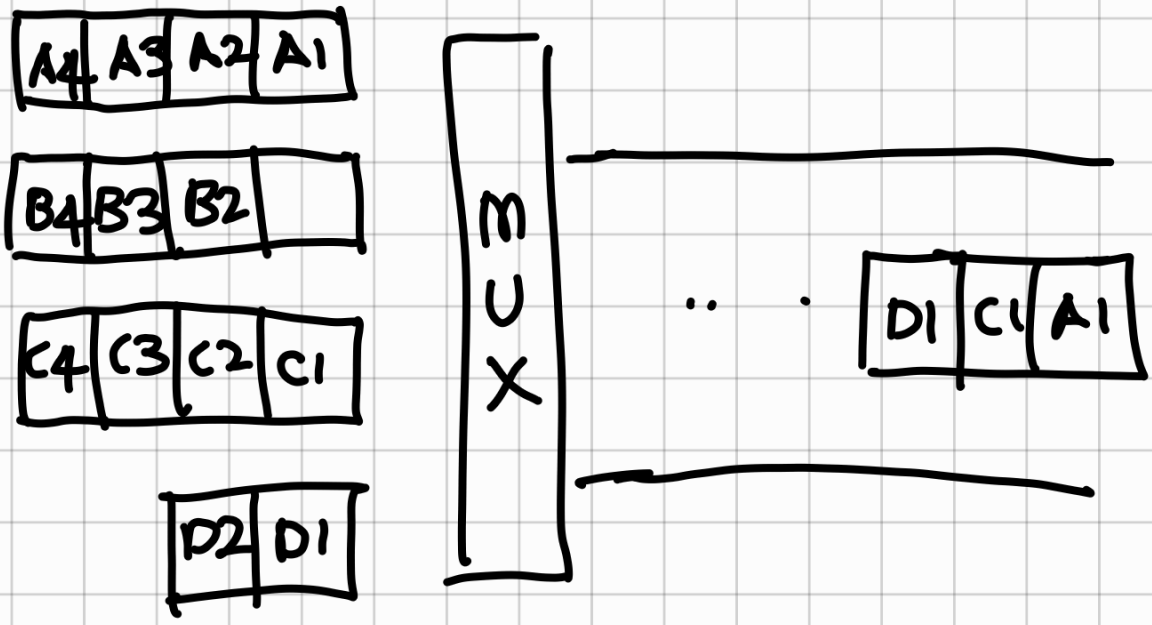
→ synchronous TDM
→ statistical TDM

Synchronous TDM



- fixed slots for each type. If no information, slots remains blank.
- takes up some unnecessary slots, and consumes more time.

Statistic TDM



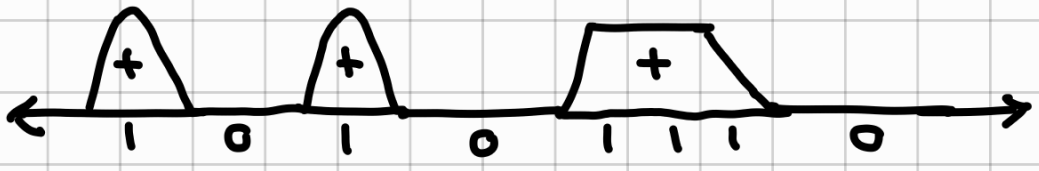
- Saves slot but messages needs addresses.

24/01 lecture skipped ...

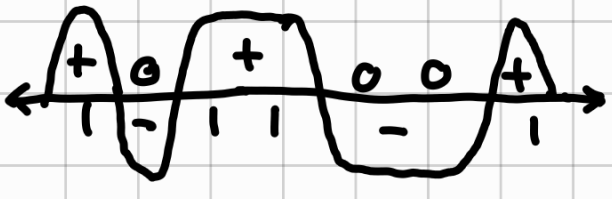
Modulation:
Data → signals
Demodulation:
Signals → Data

Modulation is done by varying the nature of signals over preferred time intervals

Return to Zero (RZ)

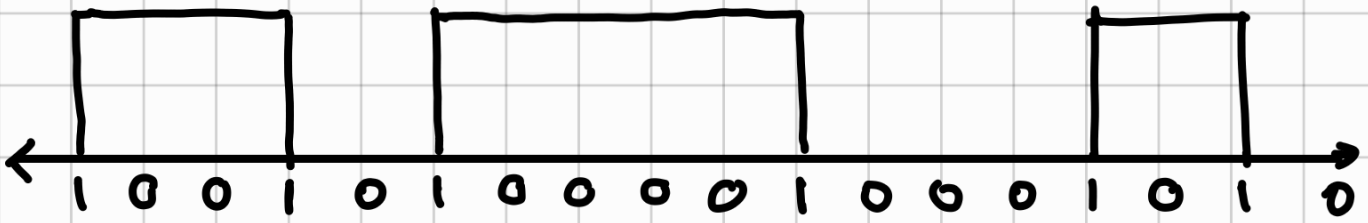


Non return to Zero (NRZ):



Problem: Sequence of 0's or sequence of 1's can be hard to detect.

Non return to zero inverted (NRZ)



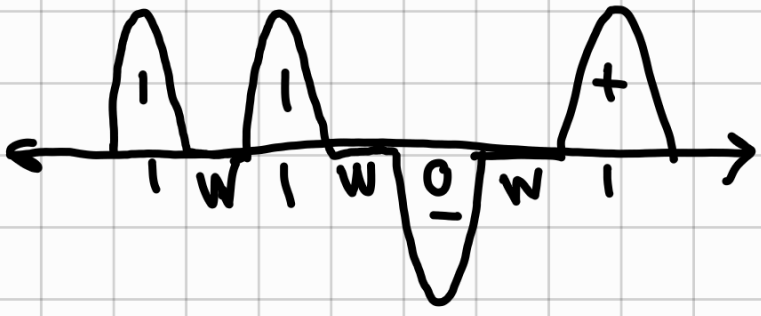
switch voltage when 1 appears

Solves the sequence of 1 problem
sequence of 0 remains

Manchester scheme

1 1 0 1

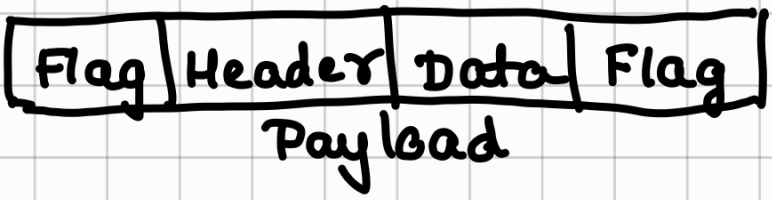
low to high : 1
high to low : 0



The waits consume half the time

Homework: Find a scheme that addresses both the weaknesses.

Framing → fixed size
→ variable size



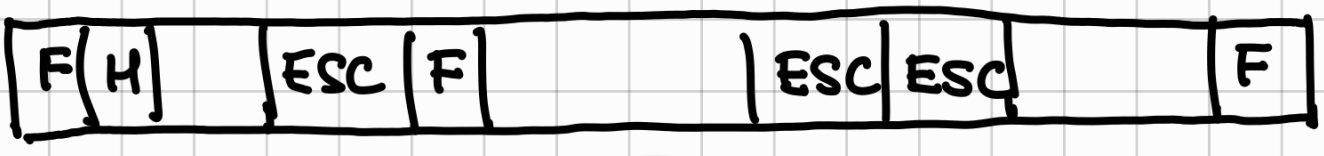
Header → ① Destination
② Protocol
③ Error Detection Code

Variable size Framing
Character oriented protocol
character → ASCII code

Use ESC character to denote beginning of a text used in flag in original body.

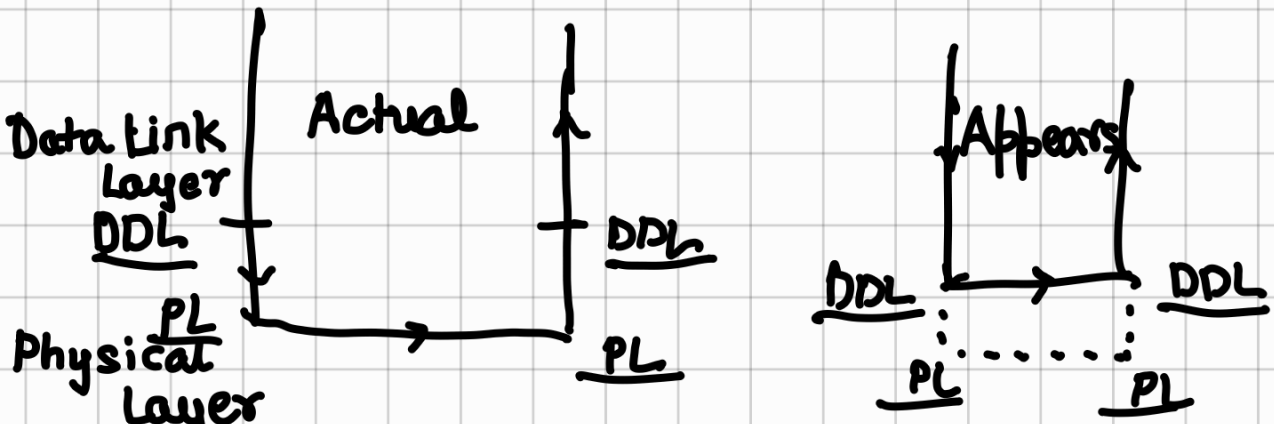
Frame sort

Frame Received

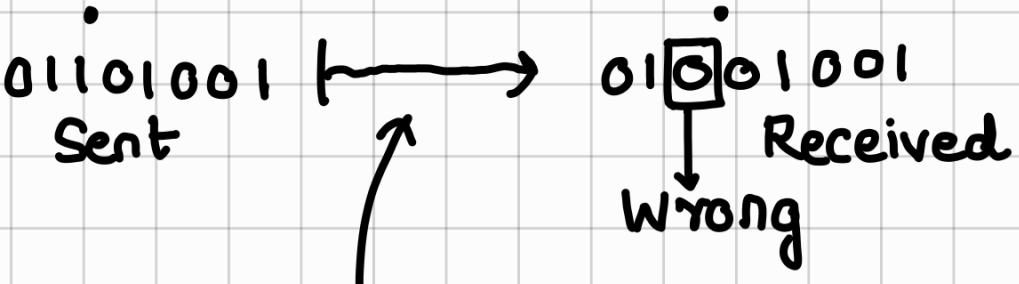


Bit oriented protocol
Flag: 01111110
If user uses 01111lyzw... in text always inserts a 0 after 01111 i.e. 01111lyzw ... shall be sent as 011111 0 yzw....

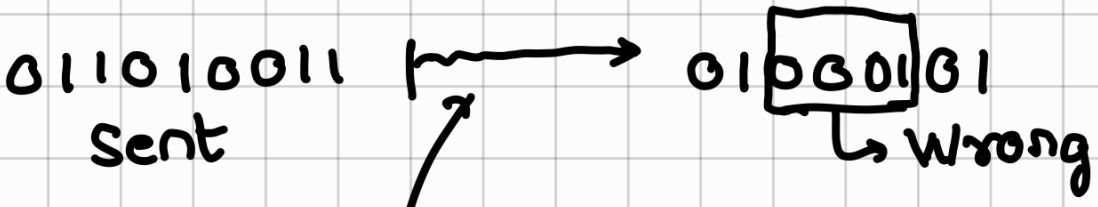
and receiver shall discard 0 & extract 01111yzw...



Error
- Single Bit error → single bit affected
- Burst Error → multiple bits affected



Single Bit Error



Burst Error

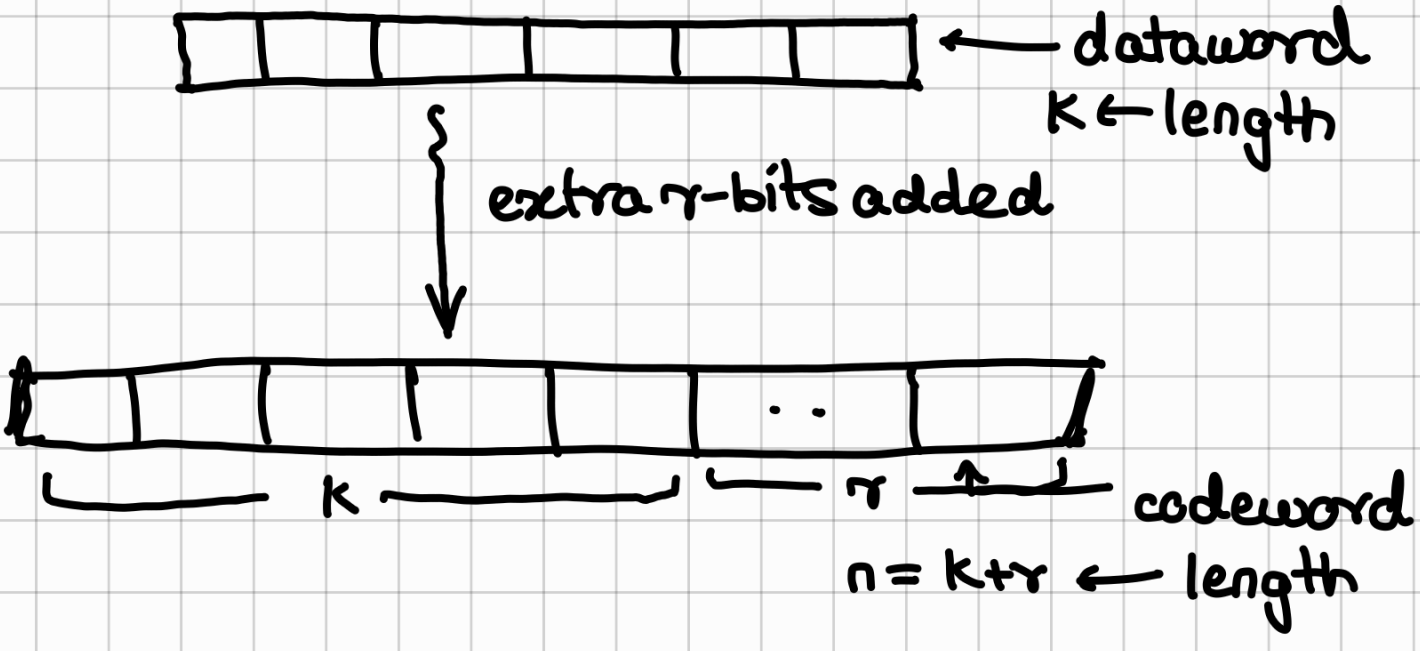
Remark: Single Bit Error is rather rare.

Suppose data is being used in 1Mbps, i.e., 1e6 bits per second.
→ fluctuation, if happens, would occur for time $t > \underline{1\mu s}$ with high probability.
(very small time)

(even 1 Mbps is very very slow for today's standard)
∴ single bit error will happen if noise lasts only for 1μs which is very unlikely.

→ Error Detection
→ Error Correction

- Forward error correction
  → receiver tries to guess the correct message.
- Retransmission
  → sends it back



$\mathbb{Z}/n\mathbb{Z} = \{\bar{0}, \bar{1}, \ldots, \overline{n-1}\}$

$x = rn + a \quad 0 \leq a < n$
$y = r'n + b \quad 0 \leq b < n$
Then, $(x+y) = r''n + c \quad 0 \leq c < n$
where $a+b = ln+c$

$n=2$

$\bar{0}+\bar{0}=\bar{0}$
$\bar{1}+\bar{1}=\bar{0}$
$\bar{0}+\bar{1}=\bar{1}$
$\bar{1}+\bar{0}=\bar{1}$

} XOR ⟹

$\bar{0}-\bar{0}=\bar{0}$ $(\ldots -\bar{0}=\bar{0})$
$\bar{0}-\bar{1}=\bar{1}$ $(\ldots -\bar{1}=\bar{1})$
$\bar{1}-\bar{0}=\bar{1}$
$\bar{1}-\bar{1}=\bar{0}$

$n=p$, p is prime, $\mathbb{Z}/p\mathbb{Z}$ is a field.

Exercise: Prove that if $n=p$ prime, then for all $\bar{a}\neq 0$ $\exists$ $\bar{b}$ s.t. $\bar{a}.\bar{b}=1$

We mostly study block codes.
generally if datawords are of size r and code words of size n then encoding scheme is denoted by $C(n,r)$.

Example: $n=3, r=2$

Dataword	Codeword
00	000
01	011
10	101
11	110

Observation:
1. If 1 bit is corrupted then receiver can detect it as they will no longer be in the list but can not correct.
2. can not detect 2-bit errors.

Example: $r=2$, $n=5$

00	00000
01	01011
10	10101
11	11110

1. All errors can be detected.
2. One-bit errors can be corrected. (since each codeword differs by atleast 3 positions)

Hamming Distance

Hamming distance b/w 2 bit strings X, Y of same size is defined by

$d(X,Y)$ = # of positions they differ
= # of 1's in $X \oplus Y$

Minimum Hamming distance of an encoding scheme is the minimum Hamming distance b/w two codewords.

Theorem: If Hamming Distance b/w any 2 codewords is atleast $s+1$ then upto s bit errors get detected.

Proof: Exercise.

Theorem: If Hamming distance b/w any two codewords is atleast $2s+1$ then receiver can detect and correct upto s bit errors.

Linear Block code

A block code is called LBC if XOR of any 2 codewords is a codeword.

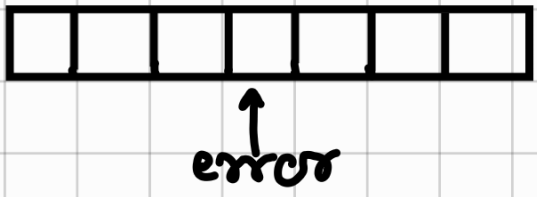
Simple parity check: $n = r+1$

Add one extra bit at the end to make sure # of 1's is even.

Exercise: Show this is LBC.

5/02 Class missed.

Error Detection in Cyclic Codes

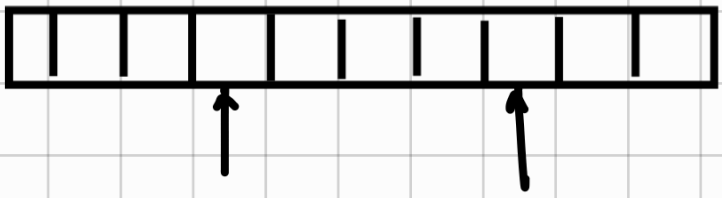


Single Bit Error

How to Detect?

What do you pick as generator? x^i+1, $i>0$.

Yes! As x^i is never divisible by x^i+1 if $i>0$.



Isolated 2 bit errors

Can x^3+x+1 divide $x^k+x^l = x^l(x^{k-l}+1)$ where $k > l+1$? [if $k = l+1$, error is not isolated]

Can pick $g(x)$ to be something that does not divide x^i+1 $\forall$ $i \in \{0,2,\ldots,n\}$ and x^i $\forall$ $j \in \{1,2,\ldots,n\}$.

↳ [$i=1$ because $x^2+1 = (x+1)^2$]

$$x^4+1 = x(x^3+x+1)+x^2+x+1$$
$$x^q+1 = x^{q-3}(x^3+x+1)+x^{q-2}+x^{q-3}+1,\ q>4$$

{ x^3+x+1 is a possible generator }
since it is eventually possible to write x^q+1 and reduce it to x^k+1, $k<q$. Through induction we can prove this result.

$$\begin{aligned} x^{10}+1 &= x^{10}+x^9+x^8+x^9+x^8+1 \\ &= (x^3+x+1)(x^7)+x^9+x^7+x^8+1 \\ &= (x^3+x+1)(x^7)+(x^3+x+1)(x^6)+x^8+x^6+1 \\ &= (x^3+x+1)(x^7+x^6)+(x^3+x+1)(x^5)+x^5+1 \\ &= (x^3+x+1)(x^7+x^6+x^5)+\underline{x^5+1}. \end{aligned}$$

Odd number of Errors.
Exercise:
Any multiple of $x+1$ catches odd no. of errors

Check Sum:
Sender needs to send k numbers sends k+1 instead where (k+1)-th is the sum of others.
Receiver cross checks the sum.

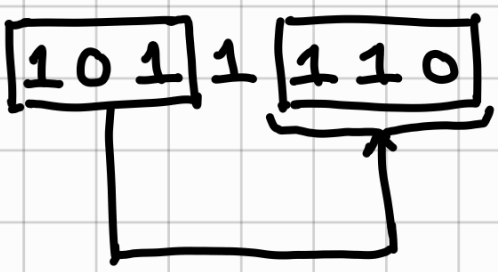
Modification: Send -ve of the sum.

Issue:
say $k=4$
and each number has to be sent in 4 bit stream

can hold 0, 1, ..., 15

If you want to send 8 7 7 8 the sum is 30 that does not fit 4 bit stream

1's complement
Suppose sum is



1011110 ⟶ 1110
+ 101
1011 ← wrapped sum

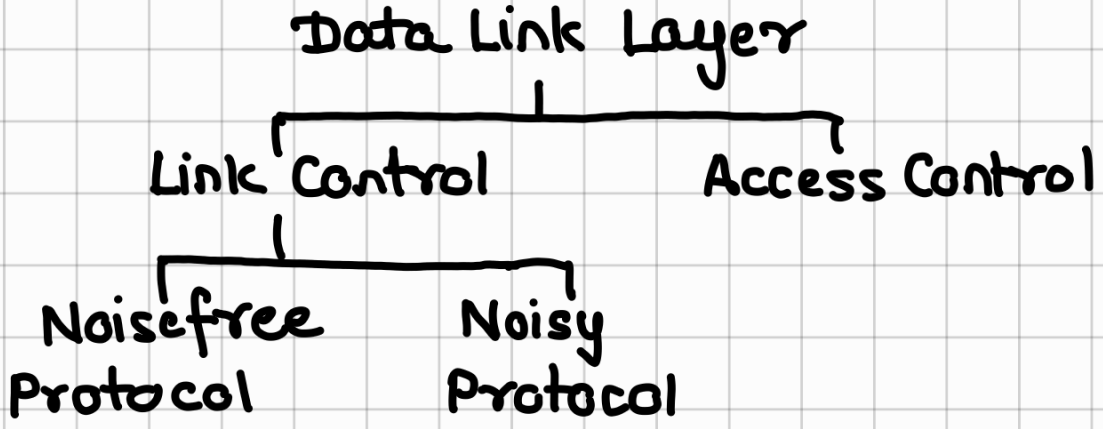
Take the -ve : 0100 ← check sum
Sender sends the check sum after wrapping the left most extra bits to the right.
Receiver reverses the process.

Example : Bit stream size : 5

Sender
7
11
12
0
6
Sum: 36
100100
wrapping 00101 : 5
checksum : 26

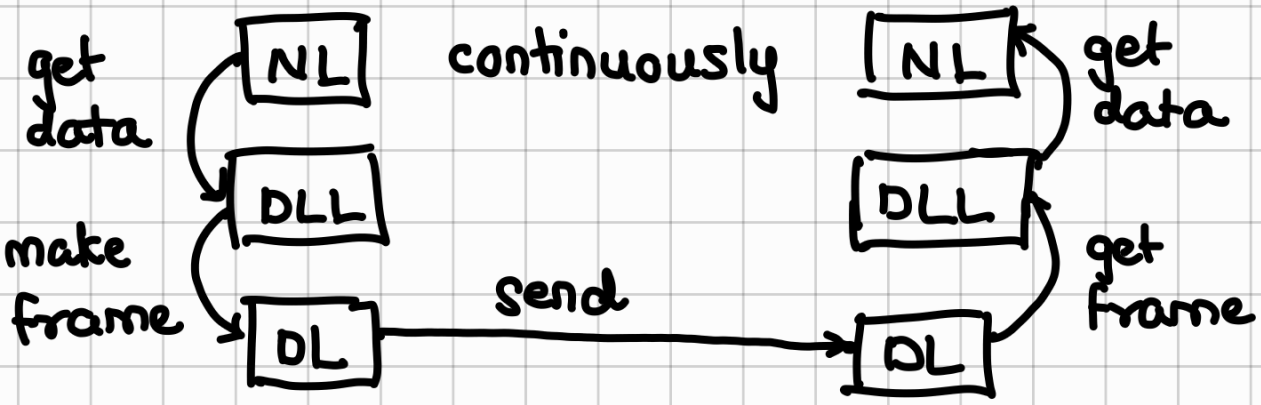
Receiver
7
11
12
0
6
26
Sum : 62
↓
111110
} wrapping
11111
} negation
0

Data Link Layer → Link Control
→ Access Control



Link Control →
- flow control (relevant for both noiseless and noisy)
- error control (relevant for noisy channel)

Simplest Protocol



Stop and Wait Protocol

1. Sender sends a frame.
2. Wait to get an acknowledgement frame back from receiver
3. Then once acknowledgement received, it sends

the next frame.

Noisy channel : Error can happen

Stop and Wait automatic repeat.

1. Sender sends a frame
2. Receiver receives and checks for error. If error not found an acknowledgement sent back, else receiver stays quiet.
3. Sender, if Acknowledgement received, sends the next frame. Else after sometime sends the previous frame back.

Go Back N and automatic repeat.

Number the frames mod 2^n.

Sender can keep sending frames in the following order

$0, 1, 2, \ldots, 2^n-1, 0, 1, 2, \ldots, 2^n-1, 0, 1, 2, \ldots$

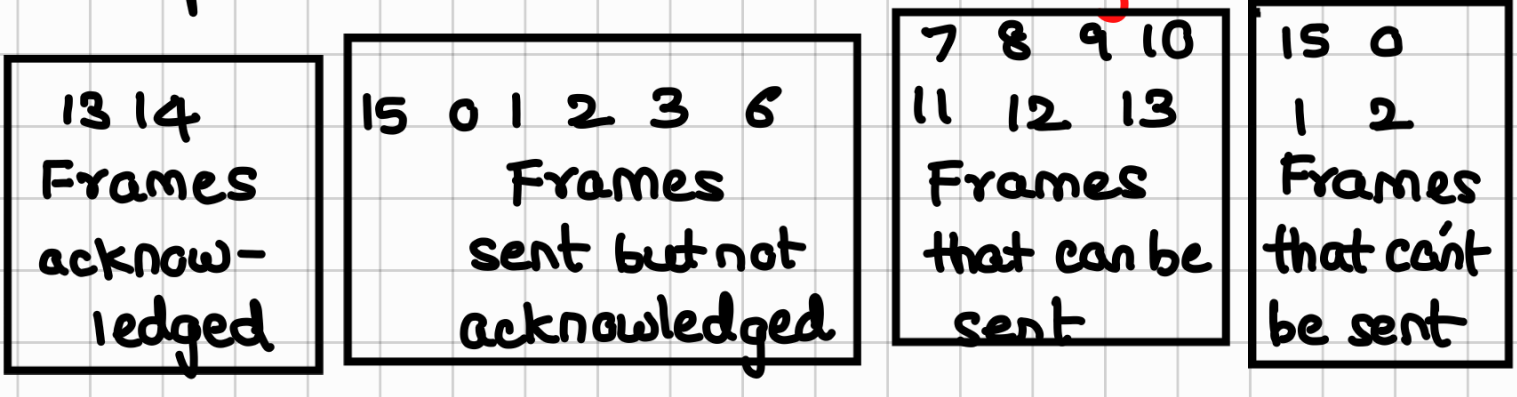
Suppose $i-1, i-2, i-3, 0, 2^n-1, \ldots, i+2, i+1, i,$ $i-1, i-2, \ldots$

has been sent and Acknowledgement has been received for $0, 1, \ldots, i$

then next i can be sent but not next i+1.

Example:

Sliding Window



Modulo Arithmetic

$14 * 33 = 62$ in mod 100

$62 \div 33$ in mod 100

$062 \div 33$ in integer not possible

$162 \div 33$ " " " "

$262 \div 33$ " " " "

$362 \div 33$ " " " "

$462 \div 33 = 14$

$98 * 33 = 34$

$34 \div 33$

⋮ 30 trials

$3034 \div 33 = 98$

finding inverse is not easy

Hari wants to send a 2 digit number to Anil.
Anil sends it by multiplying 77.
77 is public key (11,7) is private key
Hari wants to send 34

$\longrightarrow (34 * 77) \bmod 100 = 18$

Now, $18 \div 77$ is not easy

($41 \div 77$ requires 33 trials)

What if we know prime factorization of 77?

041 $63 \div 11$

141 $163 \div 11$

241 $263 \div 11$

341 $363 \div 11 = 33$

$441 \div 7 = 63$

So, division is much easier if we know prime factorization of the public key.

For our example,

$18 \div 7$	$74 \div 11$
$118 \div 7$	$174 \div 11$
$218 \div 7$	$274 \div 11$
$318 \div 7$	$374 \div 11 = 34$
$418 \div 7$	
$518 \div 7 = 74$	

getting factors from a number is hard.
making number from factors is easy.
hence, (11,7) is private key.

How do we know if the information being sent is actually sent by Hari?
he can send additional number for you to verify.

3742 divmod 77

3742 divmod 7 = 534 (x7) + 4

534 divmod 11 = 48 (x11) + 6

$$3742 \text{ divmod } 77 = \frac{(48*11+6)*7+4}{11*7}$$

$$= \frac{48*11*7}{11*7} + \frac{42+4}{77}$$

$$= 48 + 46/77$$

$3742 = 77*48 + 46$

We use this procedure as verification

Giving marks to students (Hashing)

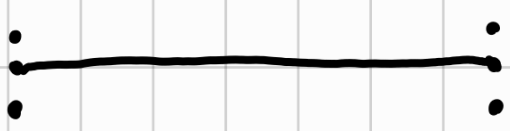
Student	Marks		
Anil	2137		
Dipu	947	12,19	(77*12+19)
Gyan			

divmod of marks and reverse is sent.
Mix roll no. and marks and khichdi karke bhej diya.

— x —

Network of computers

Sofar, we have studied only communication b/w 2 computer through single channel.



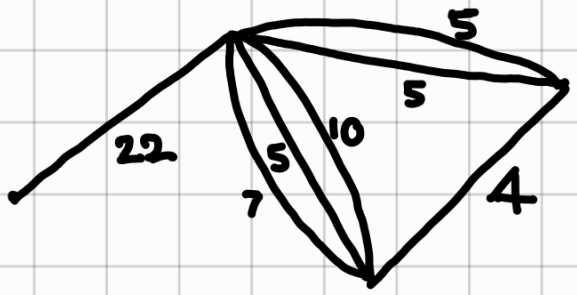
Routing

	C	F	?
1	2	5	6
2			
⋮			
n			

→ various routing algorithms to route a data packet

Spanning Tree Algthm.
→ Prim
→ Kruskal

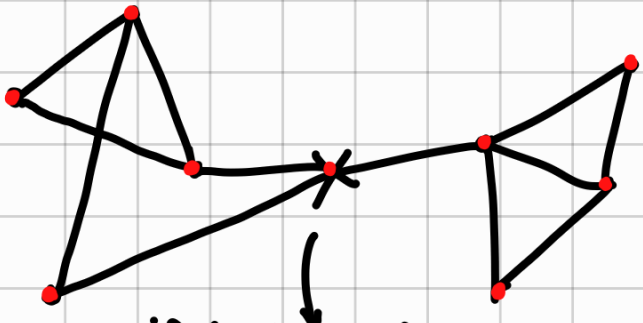
Flow Algthm.



Coloring edge weights

Routing

- Different packets b/w same origin and destination may follow different paths in standard computer network communication
  To govern this, properly planned routing is needed
- Generally routing is done by a complex set of algorithms cooperating with each other
- Routing demands communication (if needed) b/w any 2 pairs of nodes
- It needs to incorporate for the situations of node and link failure



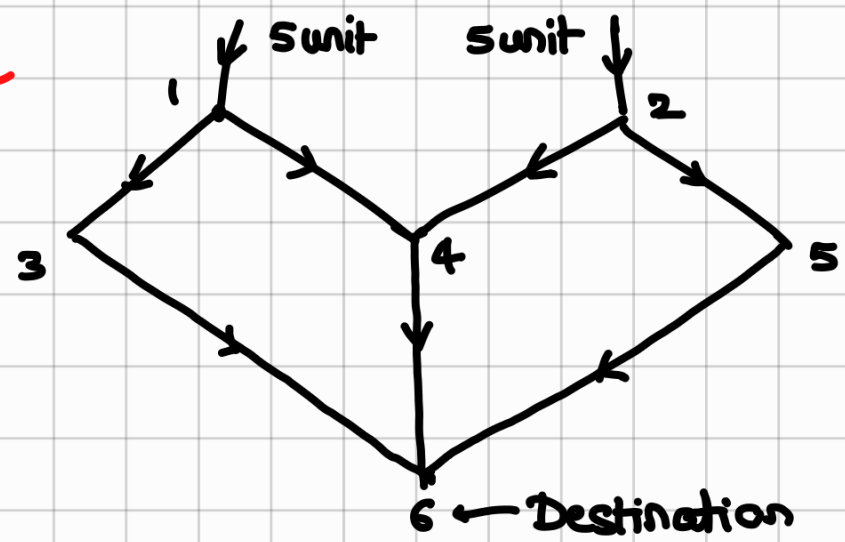
if this node breaks down
the network would be disconnected
Hence, it is essential for routing to incorporate for the situations of node.

∴ At times, it should incorporate when an information needs to be broadcasted for such situations.

Hence, communication is needed b/w the nodes and broadcasting is involved.

- Routing must allow modification of routes in case of congestion of network.

Example



All links have capacity 10 units.
Both 1 & 2 at 5 unit input
1-3-6 and 2-5-6

No problem

1-4-6 and 2-4-6

Some problems

1 gets 5 units and 2 gets 15 unit
Potential Routing
1-3-6 : 5 unit
2-4-6 : 7.5 unit
2-5-6 : 7.5 unit

Routing process needs to address:

- selection of route
- delivery of message through selected route (keeping record of selected route ← routing table)

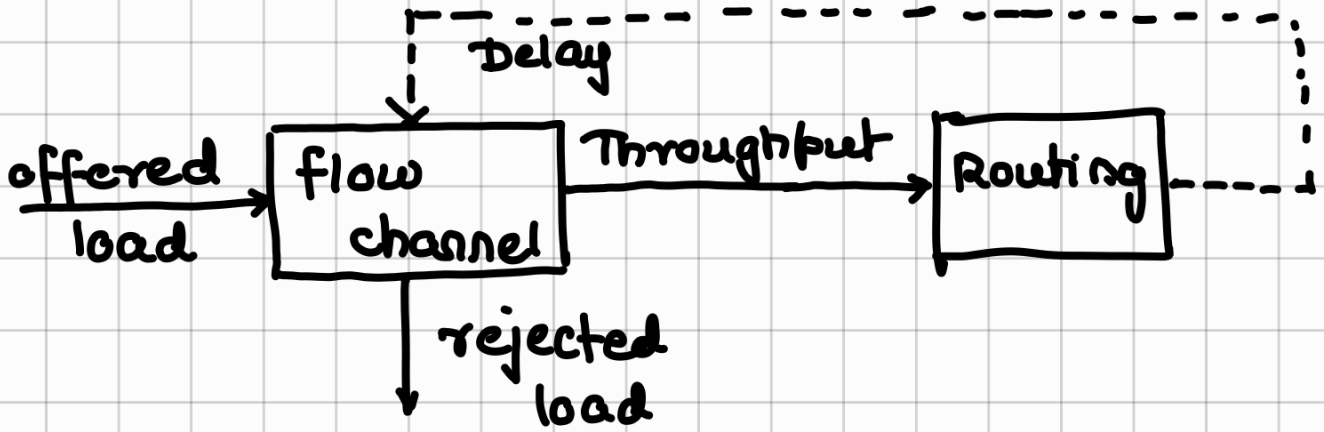
Two main performance measures → Throughput, Avg. packet delay

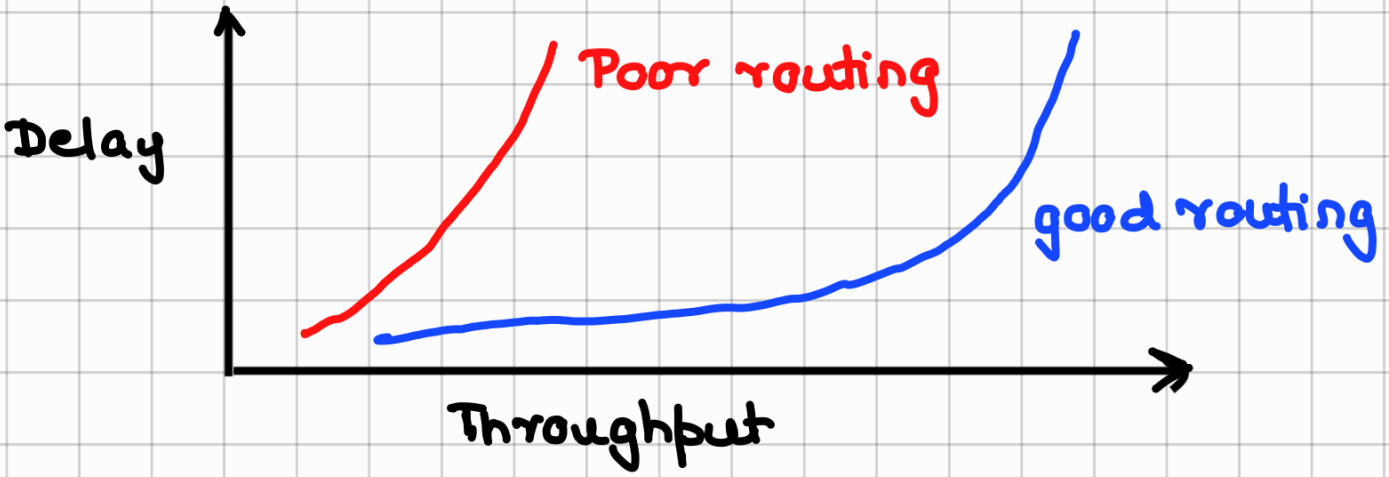
(quantitative measure)

Throughput = offered load - rejected load

(qualitative measure)

Average Packet Delay: Avg. Time needed to send one packet from one node to other





Classification:

1st centralized ← calculation of path happens at one node

v/s

distributed ← calculation of path happens at various nodes

2nd Static ← paths remain same for same origin and destination
v/s
Adaptive ← paths change for diff. packet

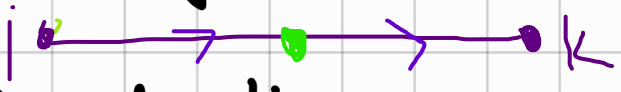
Broadcasting and Flooding

Nodes may need to communicate with all other nodes (or some other multiple nodes) time to time (to inform failure, change in protocol etc) and hence they need to broadcast from time to time
One such method is called flooding.

- origin communicates to all its neighbours
- each node once receives a broadcasting message communicates to all its neighbours

Rules:

1. No node communicates the message back to the node from which it received it.
2. If a node has communicated jth message of ith node to kth node to its neighbours already then it won't communicate lth message of ith node to kth node to its neighbours for $l \leq j$.



Spanning Tree Broadcasting

- Construct a spanning tree and call the origin of the message its root. All nodes preserve the info. about spanning tree.

spanning tree- tree with min number of undirected edges

- Root communicates message to all its neighbours
- Every node, if receives a broadcasting message communicates to all neighbours away from root.
- If N is the no. of nodes, the complete communication happens in N-1 steps.

Shortest Path Algorithms

using network as an edge weighted graph

weighted

Interpret edge weights as lengths.

- A link can have different length in different directions. length may change as line progress
- Each path between to nodes has a length equal to the sum of the lengths of the links
- A shortest path routing algorithm routes each packet along a minimal length path between origin and destination.

Example

Bellman-Ford method:

D_i := Estimated shortest distance of node i to the destination and d_{ij} is length of link (i,j).

Then $D_i := \min_j (d_{ij} + D_j)$

- Each node periodically does this calculation.
- $d_{ij} + D_j$ may be viewed as the estimate of shortest

distance from node i to the destination going through j.

Convention: 1 is the destination

D_i^h : Distance of shortest path from i to 1 that contains $\leq h$ edges

$D_i^h : 0 \quad \forall h$

$D_i^0 : \infty \quad \forall i \neq 1$

We shall prove

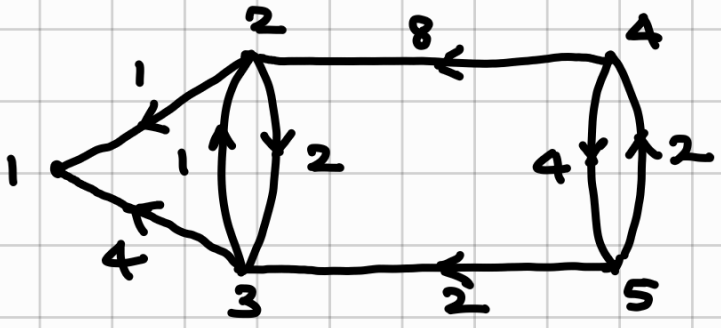
$$D_i^{h+1} = \min_j [d_{ij} + D_j^h]$$

The algorithm terminates if $D_i^h = D_i^{h-1} \forall i$ for some h.

Why??

$$D_i^{h+1} = \min_j [d_{ij} + D_j^h] = \min_j [d_{ij} + D_j^{h-1}] = D_i^h$$

So, $D_i^{h+l} = D_i^h \quad \forall \, l \geq 1$



I : $D_1^1 = 0 \quad D_2^1 = 1 \quad D_3^1 = 4 \quad D_4^1 = \infty \quad D_5^1 = \infty$

II : $D_1^2 = 0 \quad D_2^2 = 1 \quad D_3^2 = 2 \quad D_4^2 = 9 \quad D_5^2 = 6$

III : $D_1^3 = 0 \quad D_2^3 = 1 \quad D_3^3 = 2 \quad D_4^3 = 9 \quad D_5^3 = 4$

IV : $D_1^4 = 0 \quad D_2^4 = 1 \quad D_3^4 = 2 \quad D_4^4 = 8 \quad D_5^4 = 4$

V : $D_1^5 = 0 \quad D_2^5 = 1 \quad D_3^5 = 2 \quad D_4^5 = 8 \quad D_5^5 = 4$

Algorithm Haults

Step-IV is stable.

Theorem:

Consider the B-F algorithm with initial conditions $D_i^0 = \infty \ \forall \ i \neq 1$, then

(a) The scalars D_i^h generated by the algthm. are equal to the shortest ($\leq h$) walk lengths from node i to node 1.

(b) The algthm. terminates after a finite no. of iterations. Furthermore, the no. of iterations needed is $\leq N$ (no. of nodes) and at termination, D_i^h is the shortest path length from i to 1.

Proof: By induction

Recall

$$D_i^{h+1} = \min_j (d_{ij} + D_j^h) \ \forall \ i \neq 1$$

$$D_1^h = 0 \ \forall h$$

$$D_i^0 = \infty \ \forall \ i \neq 1$$

Note, $D_i^1 = d_{i1} \ \forall \ i \neq 1$

So, D_i^1 = shortest (≤ 1) walk from i to 1 !!

Suppose D_i^k is equal to shortest ($\leq k$) walk length from i to 1 $\forall \ k \leq h$. We shall show that D_i^{h+1} is the shortest ($\leq h+1$) walk length from i to 1.

Observe a shortest ($\leq h+1$) walk from i to 1 either consists of less than $h+1$ links in which case its length is equal to D_i^h, or else it consists of $h+1$ links where first as (i,j), $j \neq 1$ followed by an h-link walk from j to 1 where node 1 is not repeated. The

latter walk must be a shortest ($\leq h$) walk from j to 1 !!

Thus, shortest ($\leq h+1$) walk

$$\text{length} = \min\{D_i^h, \min_{j \neq 1}[d_{ij} + D_j^h]\} \longrightarrow (*)$$

By induction,

$$D_j^k \leq D_j^{k-1} \quad \forall\ k \leq h$$

[since the set of ($\leq k$) walks from node j to 1 contains the corresponding set of ($\leq k-1$) walks]

Therefore,

$$D_i^{h+1} = \min[d_{ij} + D_j^h]$$
$$\leq \min_j [d_{ij} + D_j^{h-1}] = D_i^h$$

Also, $D_i^h \leq D_i^1 = d_{i1} = d_{i1} + D_i^h$

So, from (*),

shortest ($\leq h+1$) walk for length = (#)

$$(\#) := \min\{D_i^h, \min_j (d_{ij} + D_j^h)\}$$

$$= \min\{D_i^h, D_i^{h+1}\}$$

Hence, $(\#) = D_i^{h+1}$

This proves that D_i^h generated by the algorithm is the shortest ($\leq h$) path from i to 1.

Next, we observe that if BF terminates after h iterations we must have $D_i^k = D_i^h \ \forall$ i and $k \geq h$. so we cannot have the length of shortest path

reduced by allowing more and more links.

Now, $\forall i$ and h $\exists$ a path that is shortest ($\leq h$) walk from i to 1 (search in a finite set) and the corresponding length as D_i^h by part @. As paths have no repeated nodes to be shortest it must be of length $\leq N-1$.

Hence, it follows that

$$D_i^N = D_i^{N-1} \quad \forall i$$

So, algthm. terminates in step $\leq N$.

Q.: What is the complexity of BF algorithm?

So, we saw

Routing
- Broadcasting (Flooding)
- Point-point shortest path (Bellman-Ford)

Bellman Ford Ctd.

1. Complexity
2. Shortest-Path construction
3. uniqueness

Theorem: No. of computation required in BF algorithm is $O(N^3)$

Proof: In every step one computes D_i for N nodes via minimization over atmost N-1 neighbours. So each step takes $O(N^2)$ computations.

- The algorithm converges in atmost N steps
- So total no. of computations is $O(N^3)$.

Shortest Path Construction in Bellman-Ford:

Bellman Ford Eqn:

$$D_i = \min_j [d_{ij} + D_j] \; ; \; D_1 = 0$$

BF Alg. gives a soln to

(*) Path from i to 1: For each i, pick a neighbour j s.t. D_j is soln. for (*) in BF algorithm.

So basically apart from assigning a value D_i to each node i BF algorithm also assigns a neighbour j to each i

(j is the one that gives the minimization in $D_i := \min_j [d_{ij} + D_j]$)

To get the path from i to 1 from each vertex go to the neighbour designated by BF algorithm. — (#)

Theorem: (#) gives a path from each node to 1

Proof: Enough to show $\forall i$ once the process (#) starts no nodes get repeated.

By contradiction:

Assume $i, j_1, j_2, \ldots, j_k, j_{k+1}, \ldots, j_\ell$ appears where $j_k = j_\ell$

Note: $D_{j_k} = d_{j_k j_{k+1}} + D_{j_{k+1}}$

$$D_{j_{k+1}} = d_{j_{k+1} j_{k+2}} + D_{j_{k+2}}$$

$$\vdots$$

$$D_{j_k} = D_{j_\ell} = d_{j_{\ell-1} j_\ell} + D_{j_\ell}$$

$$\therefore D_{j_k} = d_{j_k j_{k+1}} + d_{j_{k+1} j_{k+2}} + \ldots\ldots + d_{j_{\ell-1} j_\ell} + D_{j_k}$$

$$\therefore \sum_{i=k}^{l-1} d_{i_i i_{i+1}} = 0$$

$\Rightarrow \Leftarrow$ (since we assume all edge lengths to be positive in the network)

Theorem: (#) gives shortest length path from each i to 1.

Proof: Proved earlier that D_i gives length of shortest path from i to 1

And observe that (#) gives a path of length D_i.

Q: Potentially BF eqns may have multiple solns?

Ans: No !!

Thm: BF-eqns have unique solns

Pf: Suppose it has another set of soln. $\tilde{D}_i$.

So, $\tilde{D}_1 = 0$

As from $\tilde{D}_i$ too one can construct a path from i to 1 $\tilde{D}_i \geq D_i \ \forall i$

Now it is enough to show $D_i \geq \tilde{D}_i \ \forall i$

But BF algorithm first with the usual initial condition $D_i^0 = 0$, $D_i^0 = \infty$ to get $D_i \ \forall i$

(At each step minimizes $D_i^{h+1} = \min[d_{ij} + D_j^h]$)

Next run BF algorithm with initial conditions $\tilde{D}_i^0 = \tilde{D}_i \ \forall i$

Converges in 1st step itself at $\tilde{D}_j$'s !!

$D_i \geq \tilde{D}_i$

Step 1

$$D_i^1 = \min_j [d_{ij} + D_j^0] \geq \tilde{D}_i^1 = \min_j [d_{ij} + \tilde{D}_j^0]$$

$$D_i^2 = \min_j [d_{ij} + D_j^1] \geq \tilde{D}_i^2 = \min_j [d_{ij} + \tilde{D}_j^1]$$

Dijkstra's Algorithm

Target Node 1, All edges have +ve length

- The shortest of the shortest paths to node 1 must be a single edge from some neighbour of 1 (closest neighbour)
- The next shortest of the shortest paths must either be the single edge path from the next closest neighbours of 1 or shortest 2 edge path through the previously chosen closest neighbour
- To formalise this we view each node i as being labelled with an estimate D_i of the shortest path length to node 1
  when the estimate gets certain we regard this node as being permanently labelled.

Let P be the set of permanently labelled nodes $d_{ij} = \infty$ if i and 1 not connected.

Start with $P = \{1\}$

$D_1 = 0$

$D_j = d_{j1}$ for $j \neq 1$

Step 1. Find $i \notin P$

s/t $D_i = \min_{j \notin P} D_j$

$P := P \cup \{i\}$

If P contains all nodes, STOP.

Step 2. $\forall j \notin P$

$Dj := \min [Dj, dji + Di]$

GO TO STEP 1

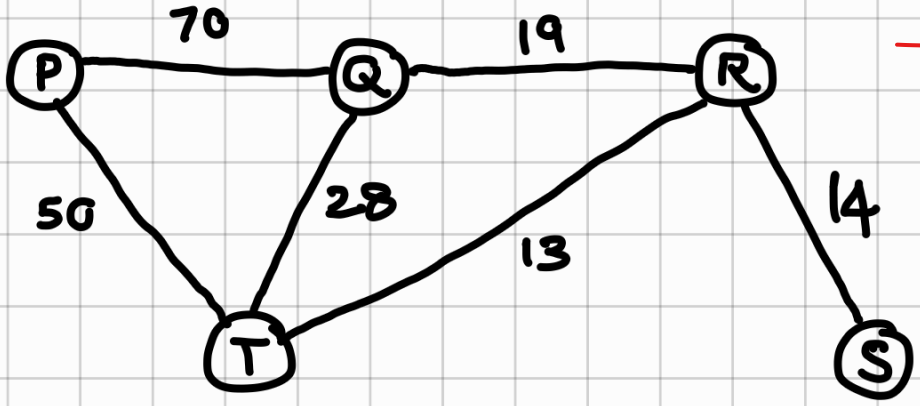
Exercise:

Prove that this converges to correct shortest lengths $\forall i$.

P. Kumar

Routing Tables

Keeps information of how much time the node will take to send information to other nodes.



Routing Table for P :

Q	70
R	63
S	77
T	50

Table P

Routing Table for R :

P	63
Q	19
S	14
T	13

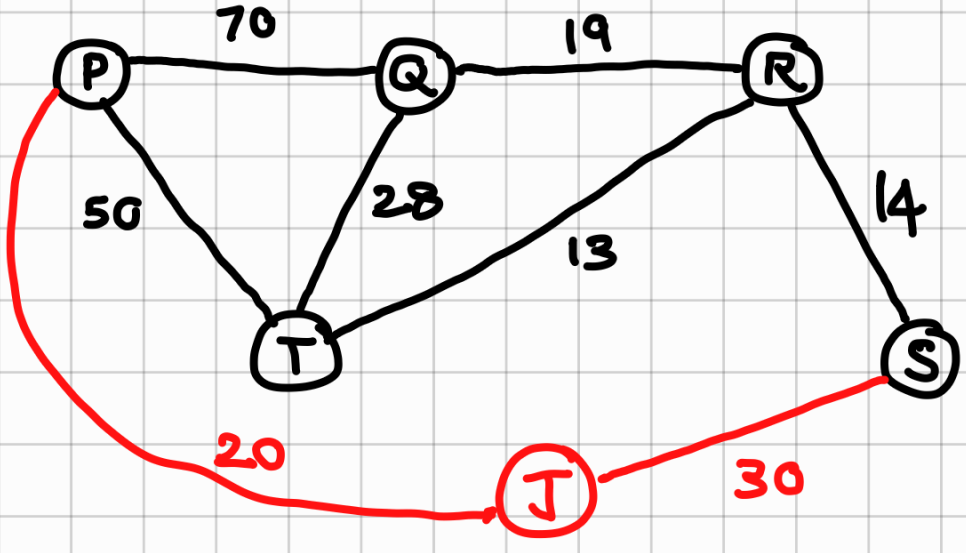
Table R

Routing Table for S:

P	77
Q	33
R	14
T	27

Table S

Let us add a new node J. What will happen to both routing tables?



Routing Table for J:

P	0+20 (Table P), 77+30 (Table S)	≡	20
Q	70+20 (Table P), 33+30 (Table S)	≡	63
R	63+20 (Table P), 14+30 (Table S)	≡	44
S	77+20 (Table P), 0+30 (Table S)	≡	30
T	50+20 (Table P), 27+30 (Table S)	≡	57

Routing Table for P :

Q	70
R	63
S	77
T	50
J	20

Table P

Routing Table for S :

P	77
Q	33
R	14
T	27
J	30

Table S

T=0 : updation/creation of routing table of node J
T=1 : updation of routing table of J's neighbours
T=2 : updation of routing table of their neighbours

≡

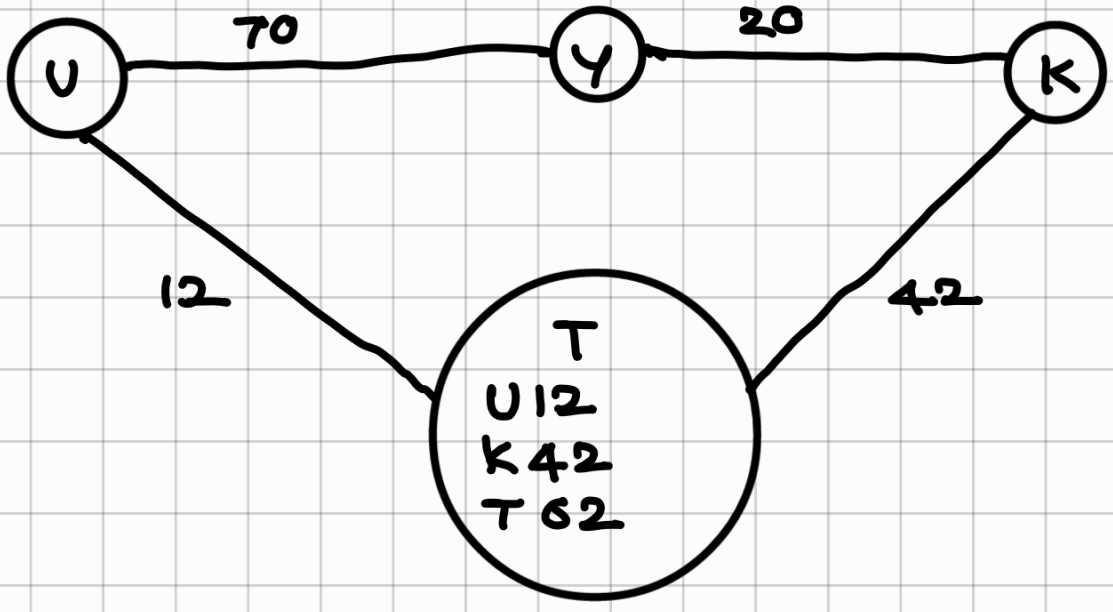
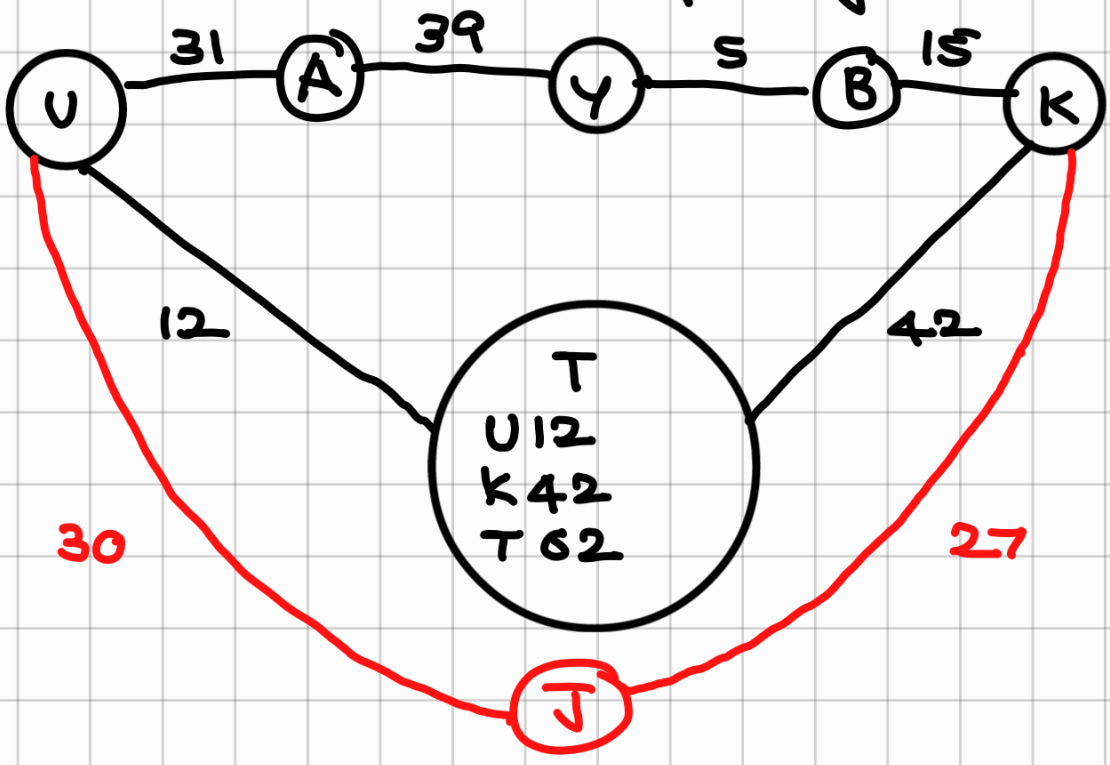


Table U : Y 70
K 54
T 12
J 30

Table K : U 54
Y 20
T 42
J 27 at t<0.

Suppose at t=10, node J appears.
Consider A and B to be temporary nodes not in routing table.



t=11 Table J :

U	30
K	27

t=12 Table J :

	U	K	
Y	70+30	20+27	= 30
K	54+30	0+27	= 27
T	12+30	42+27	= 47
U	0+30	54+27	= 42

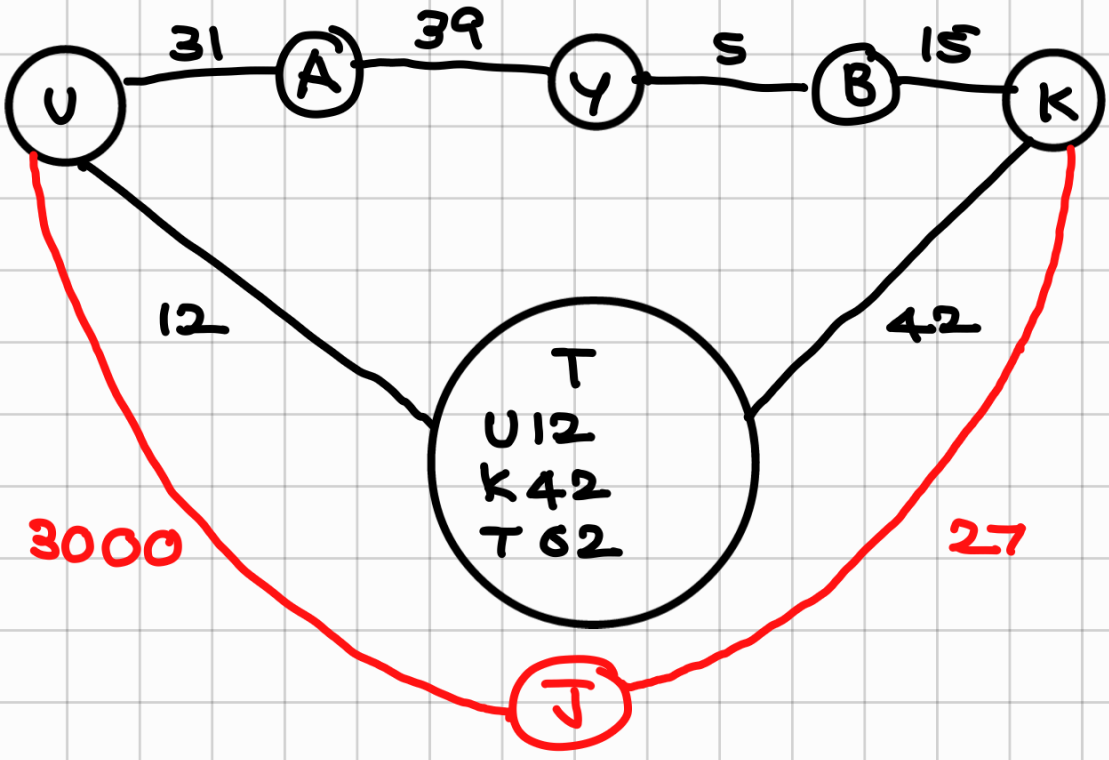
Table of T will also get updated at t=12.
(J ka value aajayega)

Table T :

U	12
K	42
T	62
J	min(30+12, 27+42)

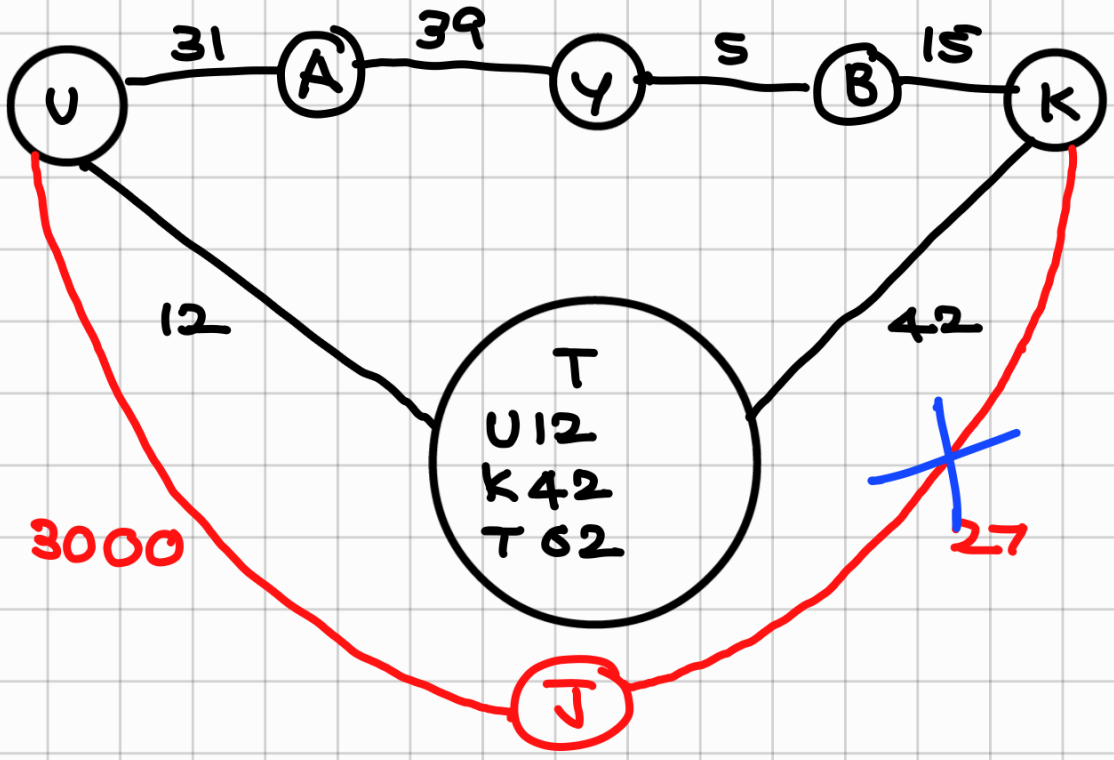
t = 13: Table Y will get updated as node Y will see node J after 3 timesteps.

This table will get stabilized after the routing table has all shortest paths.
For example,



In this case, node J will eventually reach the stable routing table but after a longer time.

What if after this, a node link is broken?



This news will spread much much slower and then, eventually, routing table is stabilized.

Bad news traverses much much slower than good news.

Method to update routing table :

→ Count-to-infinity

→ poisson-reverse ....

———×———

Dijkstra's Algorithm

$P = \{1\}$: $D_1 = 0$, $D_1 = d_{j1}$ $\forall$ $j \neq 1$

Step 1: Find $i \notin P$ s/t $D_i := \min_{j \notin P} D_j$

set $P := P \cup \{i\}$

If P contains all nodes then STOP

Step 2: (Updating of Labels)

For all $j \notin P$,

set $D_j := \min[D_j, d_{ji} + D_i]$

Go to Step 1.

Why the algorithm works?

At the beginning of each iteration of step 1:

a. $D_i \leq D_j \quad \forall \; i \in P \quad d_j \notin P$

b. D_j, $\forall j$, is the shortest distance from j to 1 using all nodes from P other than j.

Condition a. is satisfied initially and since $d_{ji} \geq 0$ and $D_i = \min_{j \notin P} D_j$ it is preserved by the formula

$D_j := \min[D_j, D_i + d_{ji}] \quad \forall \; j \notin P.$

$\longrightarrow$ proves (a).

We show condition (b) by induction.

It holds initially suppose that it holds at the beginning of some iteration of step 1.

Let i be the node added to P at that step and let D_k be the label of each node k at the beginning of that iteration. k is any node in P other than i

Then (b) holds for $j = i$ by induction. It also holds $\forall$ $j \in P$ by (a) and induction.

Finally, for $j \notin P \cup \{i\}$, consider a path from j to 1 which is shortest among those with all nodes except $j \in P \cup \{i\}$ and D_j' be the corresponding shortest distance

Such a path must consist of an edge (j,k) for some $K \in P \cup \{i\}$ followed by a shortest path from k to 1 with nodes in $P \cup \{i\}$.

Since we just saw that length if this path from k to 1 is D_k we have

$$D_j' = \min_{k \in PU\{i\}} [d_{jk} + D_k]$$

$$= \min \left[\min_{k \in P} [d_{jk} + D_k], d_{ji} + D_i \right]$$

Induction implies

$$D_j := \min_{k \in P} [d_{jk} + D_k]$$

So, we get

$$D_j' = \min [D_j, d_{ji} + D_i]$$

Thus, in step 2,

D_j is the shortest distance D_j' from j to 1 using all nodes in $PU\{i\}$ except j.

Remark: $\forall\ i \in P$, D_i is the shortest distance from i to 1.

Proof: Exercise

Exercise:

If # of nodes is N, then show that # of computations needed is $\leq O(N^2)$.

Exercise/Homework: Read about synchronous distributed Bellman Ford Algorithm and its difficulties.

Since the network is huge, it becomes infeasible to use routing algorithms discussed since then every node needs to know the connection and cost to every other node. Also, there is no central database to compute

the connections for every 2 nodes in the network. Hence, we need a distributed, synchronous algorithm to tackle this issue.

Distributed Asynchronous Bellman-Ford Algorithm:

Setup:

1. Target Node : 1
2. At each time t and each $i \neq 1$,

   $D_j^i(t)$:= Estimate of the shortest distance of each ngb. node j ($j \in N(i)$) that has been communicated to i.

   $D_i(t)$:= Estimate of shortest distance of node i which was last computed at node i according to B.F. iteration
3. Process starts at t_0

   $D_i(t) = 0 \quad \forall\ t \geq t_0$

   $D_i^i(t) = 0 \quad \forall\ t \geq t_0$ and $1 \in N(i)$
4. Each node i has d_{ij} $\forall\ j \in N(i)$ which is assumed to remain constant between t_i's:

   $0 < t_1 < t_2 < t_3 < \ldots$

   $t_m \longrightarrow \infty$ as $m \longrightarrow \infty$

Algorithm:

1. i updates $D_i(t)$ according to:

   $$D_i(t) := \min_{j \in N(i)} [d_{ij} + D_j^i(t)]$$

   and leaves $D_j^i(t)$, $j \in N(i)$ unchanged
2. Node i, receives from one or more node neighbours $j \in N(i)$ the value of D_j which was computed at some earlier time, updates the estimate

D_j^i and leaves all other estimate unchanged.

Let T^i be the set of times for which an update by a node i as in case 1 occurs and T_j^i be the set of times when a message is received at i from node j, as in case 2.

We assume:
Assumption 1: Nodes never stop updating their own estimates and receiving messages from neighbours [i.e. T^i and T_j^i have an infinite no. of elements $\forall i \neq 1$ and $j \in N(i)$]

Assumption 2: Old distance information is eventually purged from system [i.e. for $\bar{t} > t_0 \exists \tilde{t} > t_i$ s.t. D_j computed $\bar{t}$ are not received after $\tilde{t}$]

Asynchronous distributed Bellman ford continued...
Theorem: Let the initial conditions $D_i(t_0)$ and $D_j^i(t_0)$ be arbitrary $\forall i = 1 \ldots N$ and $\forall j = 1 \ldots N$ Then $\exists$ a time t_m s.t. $D_i(t) = D_i \ \forall \ t \geq t_m \ \forall \ i = 1 \ldots N$
and D_i is the shortest distance from i to 1.

Proof: $\forall i = 1 \ldots N$ define 2 sequences $\{\underline{D_i^k}\}$ and $\{\overline{D_i^k}\}$ with

1. $\underline{D_i^k} = D_i = \overline{D_i^k} \quad \forall \quad k >> 0$
2. $\underline{D_i^k} \leq \underline{D_i^{k+1}} \leq D_i \leq \overline{D_i^{k+1}} \leq \overline{D_i^k}$

These are obtained from Bellman Ford algorithm by starting at 2 different initial conditions and then it

is shown that $\forall k$, the estimates $D_i(t)$ satisfy

$$\underline{D_i}^k \leq D_i(t) \leq \overline{D_i}^k$$

Note that from Bellmanford if for some $\overline{D_j}$ and $\tilde{D_j}$ $\overline{D_j} \geq \tilde{D_j}$ $\forall j \in N(i)$

Then

$$\min_{j \in N(i)} [d_{ij} + D_j] \geq \min_{j \in N(i)} [d_{ij} + \tilde{D_j}]$$

As a result if D_i^k are generated by Bellman Ford with initial conditions D_i^0 and suppose we have $D_i^1 \geq D_i^0$ $\forall i$, then

$$D_i^{k+1} \geq D_i^k, \forall i,k$$

Similarly if $D_i^1 \leq D_i^0$ then $D_i^{k+1} \leq D_i^k$ $\forall i,k$

$\overline{D_i}^k :=$ kth iteration of BF with initial conditions

$\overline{D_i}^0 = \infty$ $\forall i \neq 1$,

$\overline{D_1}^0 = 0$

Also, Let $\underline{D_i}^k$ $\forall i = 1 \ldots N$ be the kth iteration of BF when the initial condition is

$$\left.\begin{aligned} \underline{D_i}^0 &= D_i - \delta \\ \underline{D_i}^0 &= 0 \end{aligned}\right\} \forall i$$

where δ is a +ve no. large enough so that $\underline{D_i}^0$ is smaller than all initial node estimates $D_i(t_0)$, $D_j^i(t_0)$, $j \in N(i)$ and all estimates D_i that were communicated by node i before t_0 and will be received by its neighbours after t_0.

Claim: These satisfy:

1. $\underline{D_i}^k = D_i = \overline{D_i}^k$ $\forall$ $k >> 0$

2. $\underline{D_i}^k \leq \underline{D_i}^{k+1} \leq D_i \leq \overline{D_i}^{k+1} \leq \overline{D_i}^k$

Claim-2 is shown by induction and monotonicity of BF.

[if $\{\mathcal{D}_i^0\}$ and $\{\mathcal{F}_i^0\}$ are initial conditions of BF and $\mathcal{D}_i^0 \geq \mathcal{F}_i^0$ then $\mathcal{D}_i^h \geq \mathcal{F}_i^h \ \forall i,h$] $\longrightarrow$ (#)

Note: $\underline{D_i}^1 = \min_{j \in N(i)} [d_{ij} + \underline{D_j}^0]$

$= \min_{j \in N(i)} [d_{ij} + D_j - \delta]$

$= \min_{j \in N(i)} [d_{ij} + D_j] - \delta$

$d_{ij} + D_j$ gives length of path from i to 1

Min. possible length of path from i to 1 $\leftarrow D_i$

$\geq D_i - \delta \Rightarrow \underline{D_i}^1 \geq D_i^0$

Hence, we have $\underline{D}^k < \underline{D}^{k+1}$ —— (!)

Compare this with BF initial conditions

$D_i^0 = \infty$, $D_1^0 = 0$ and use (#)

$D_k = D_i \geq \underline{D}^k \ \forall \ k \geq N-1$ using (!)

From this, we get $\forall k \ \underline{D_i}^k \leq D_i$

This proves Claim-2.

For (!) observe that $\overline{D_i}^k = D_i \ \forall \ k \geq N-1$

So only thing remains to show is $\underline{D_i}^k = D_i \ \forall k >> 0$.

Note that $\underline{D_i}^k$ is the length of a path from i to some other node j involving no more than k links than $\underline{D_j}^0$.

Let $L(i,k)$ and $n(i,k)$ be the length and no. of links of this walk respectively.
We can decompose this walk into a path from i to j involving no more than N-1 links.
Therefore if $\lim_{k \uparrow \infty} n(i,k) = \infty$ for any i, we would

have $L(i,k) = \infty$, which is impossible as $\underline{D_i}^k \leq D_i$. !!

So, $n(i,k)$ is bounded w.r.t. k and hence number of possible values of $\underline{D_i}^k$ is finite as $\underline{D_i}^k \leq \underline{D_i}^{k+1}$
This implies $\underline{D_i}^k = \underline{D_i}^{k+1}$ $\forall$ $k \geq h$ for some h.

Hence, $\underline{D_i}^h$ satisfies Bellman ford eqn. which has unique soln.
So, $\underline{D_i}^h = D_i$!!

We only discussed the proofs when the network has no cycle, as well as undirected paths.

For general case, refer 1. Any graph algm. book
2. Data Network
R. Gallagar et. al.

Note: $d_{ij} \gtrless 0$

BF Eqns: $D_i = \min_{j \in N(i)} [d_{ij} + D_j]$; $D_i = 0$ $\longrightarrow$ (#)

BF Iteration: $D_i^{h+1} = \min_j [d_{ij} + D_j^h]$

$D_1^h = 0$; $D_i^0 = \infty$ $\forall$ $i \neq 1$ $\longrightarrow$ (*)

We saw:

1. (*) converges in almost no. of vertices step to D_i's that are minimum length paths from i to 1.
2. (#) has a unique soln.
3. Bellman-Ford Path : $\forall i$ take j that gives min in
$$D_i := \min_j [d_{ij} + D_j]$$
4. BF path has no repetition of nodes.
$$\begin{aligned} D_i &= d_{ij} + D_j \\ &= d_{ij} + d_{jj_1} + D_{j1} \\ &\quad \ldots \\ &= d_{ij} + d_{jj_1} + \ldots + d_{jk-1jk} + d_{jki} + D_i \end{aligned}$$
$\Rightarrow d_{ij} + d_{jj_1} + \ldots + d_{jk-1jk} + d_{jki} = 0$ (contradiction)

5. From here we observed that if
$$D_i^{h+1} = \min_j [d_{ij} + D_j], \quad D_i^h = 0 \ \forall h$$
$$D_i^0 = c_i$$
converges then it must converge to the correct solutions, i.e. minimum length paths!

Asynchronous Distributed Bellman Ford :
$t_0 < t_1 < \ldots < t_m < \ldots \quad m \to \infty, t_m \to \infty$
i) Things remain same between t_m, t_{m+1}
ii) Each vertex i, has the values d_{ij} $\forall j \in N(i)$ and gets information when d_{ij} changes. Also gets D_j from all its neighbours and computes
$$D_i := \min_{j \in N(i)} [d_{ij} + D_j]$$
$D_i(t)$:= Estimate of min. dist from i to 1 computed via BF iteration at time t
$D_j^i(t)$:= Estimate of the shortest distance of j to 1
$j \in N(i)$

available to i at time t

$D_i(t) := D_i^i(t)$

$D_i(t) = 0 \quad \forall\ t \geq t_0$

$D_i^i(t) = 0 \quad \forall\ t \geq t_0$ and $i \in N(1)$

At time tm's, one of 3 things happens:

1. i updates $D_i(t) = \min_{j \in N(i)} [d_{ij} + D_j]$ and keeps $D_j^i(t)$ unchanged.
2. updates D_j^i after getting changed values of D_j's for some neighbours and keeps D_j's unchanged for other neighbours.
3. i sits idle

T^i = set of times when 1 occurs

T_j^i = set of times 2 occurs

Assumption 1: Nodes do not stop sending its D_i's and calculates its D_i's

i.e. T^i and T_j^i both are infinite sets.

Assumption: Old informations eventually deleted

i.e. $\forall\ \bar{t} \geq t_0\ \exists\ \tilde{t} > \bar{t}$ s/t D_j computed at node j prior to time $\bar{t}$ are not received at any neighbour of node i after time $\tilde{t}$.

Fact: $D_i(t)$'s converge to correct distance in finite time.

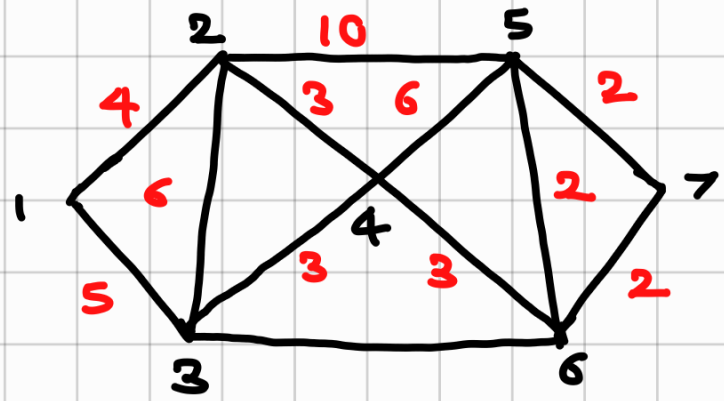
We did this by constructing $\{\underline{D_i^k}\}$ and $\{\overline{D_i^k}\}$ s/t

$\underline{D_i^k} \leq \underline{D_i^{k+1}} \leq D_i \leq \overline{D_i^{k+1}} \leq \overline{D_i^k}$

(proved in last class)

Problems :

1. Consider



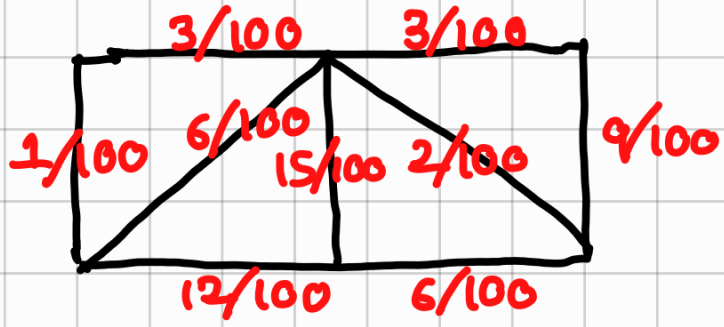
Run BF and Dijkstra algorithms.

2. Consider BF Algorithm on a tree. For each node $i \neq 1$, take the node j_i s/t j_i gives the minimum on

$$D_i^{h_i} = \min_j [d_{ij} + D_j^{h_i - 1}]$$

where h_i is the largest h s/t $D_i^h \neq D_i^{h-1}$. Consider the subgraph consisting of all ij_i and show that this is a spanning tree consisting of shortest paths.

3.

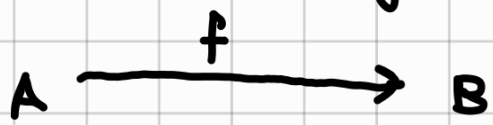


The number shown in each link is the probability of the link failing during the lifetime. It is assumed that links fail independently of each other. Find the most reliable path b/w each pair of node.

4. Can you think of a distributed version for Dijkstra's algorithm ?

Crypt-analysis :

A = {messages} B = {encrypted messages}



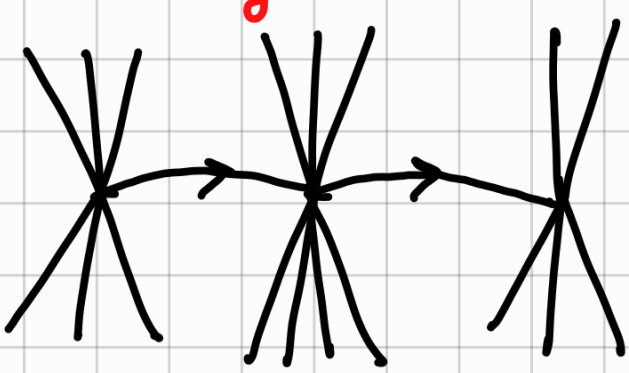
f ← 1-1, onto

Finding f^{-1}?

page 8 - part 2 notes

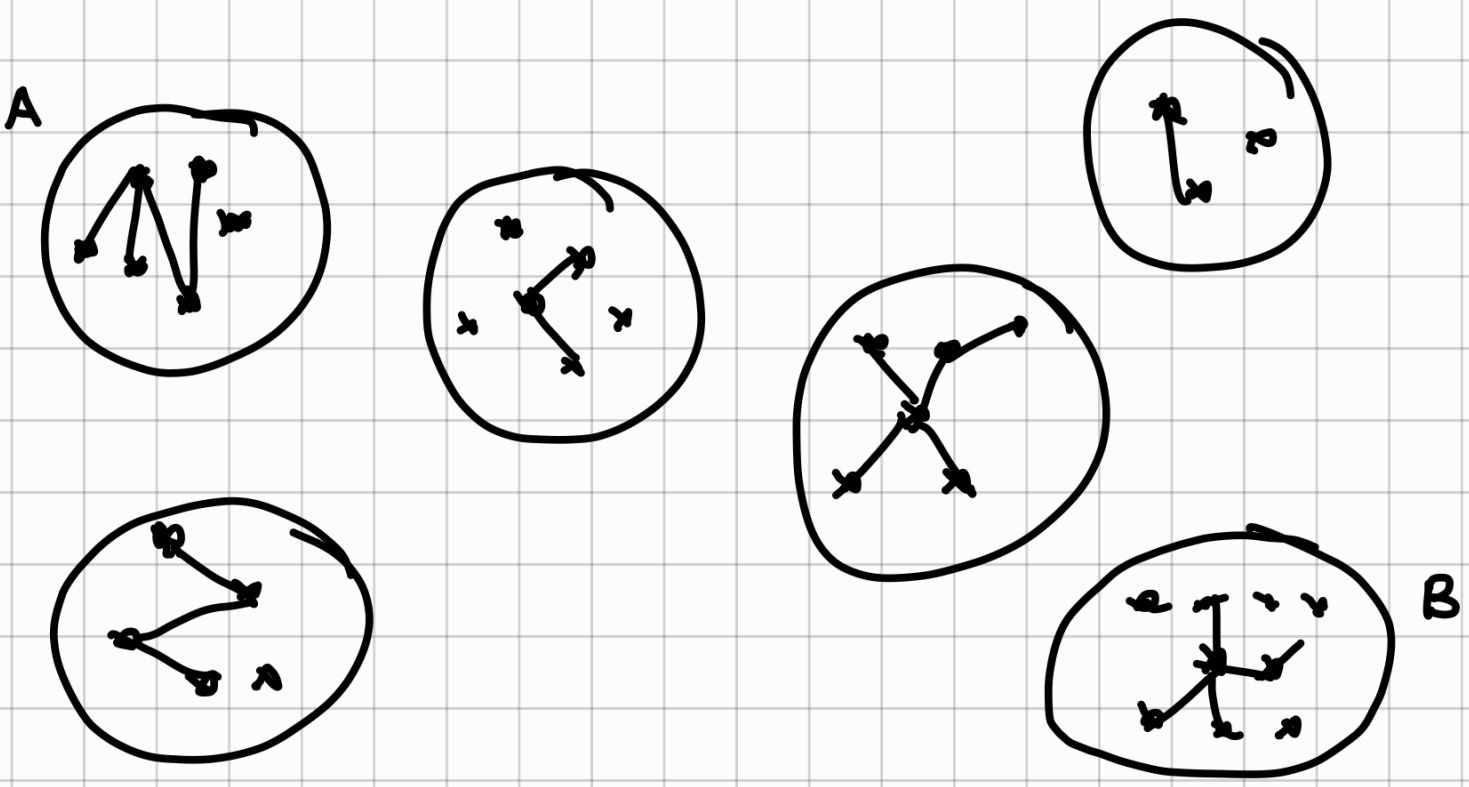
1 Apr / 3 Apr Class missed

Internetworking

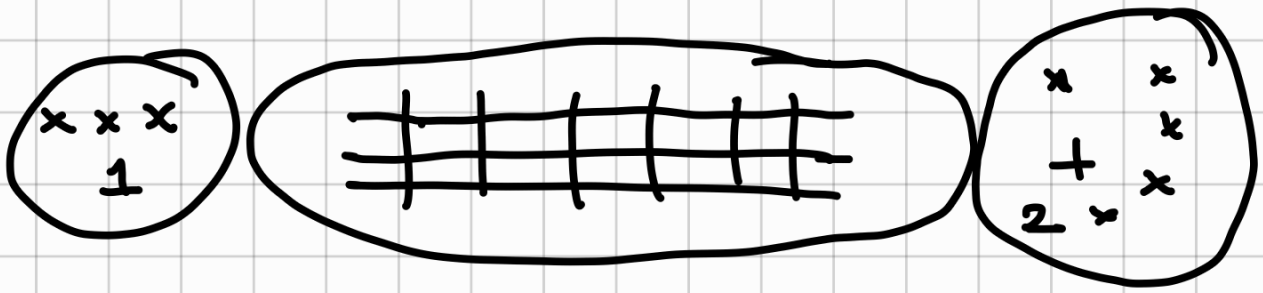


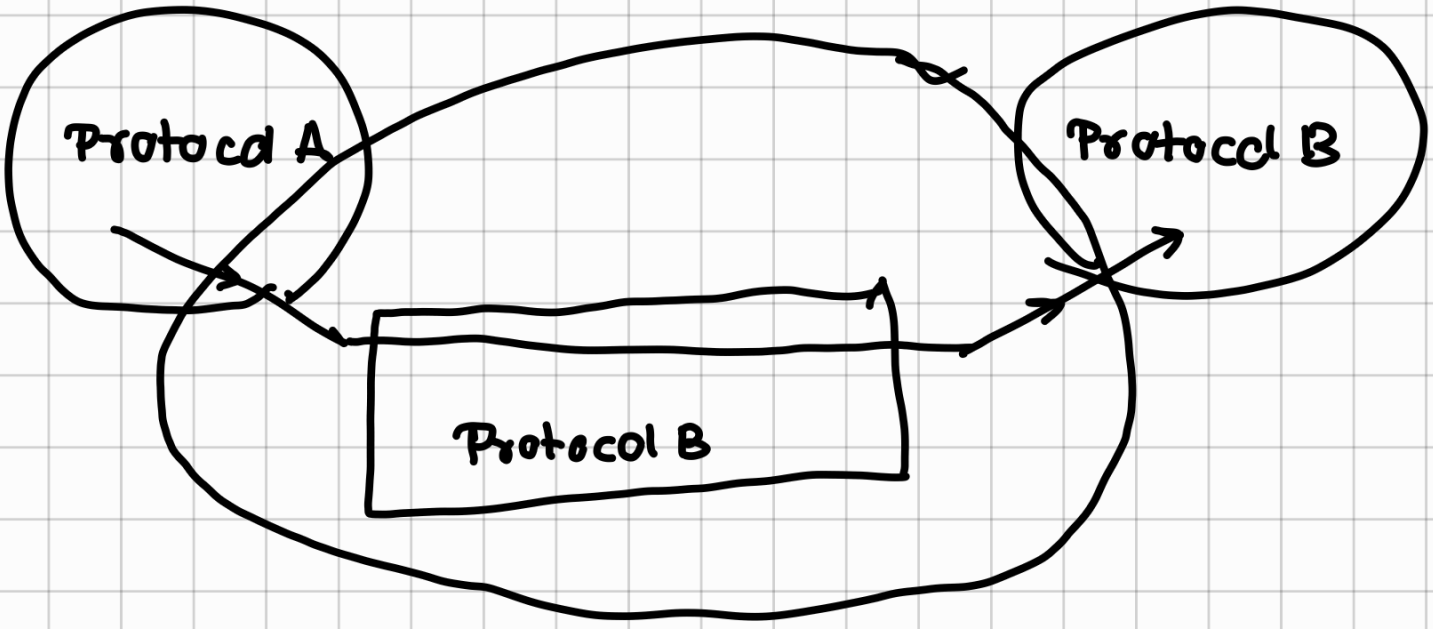
Different networks use different protocols

How to communicate ?

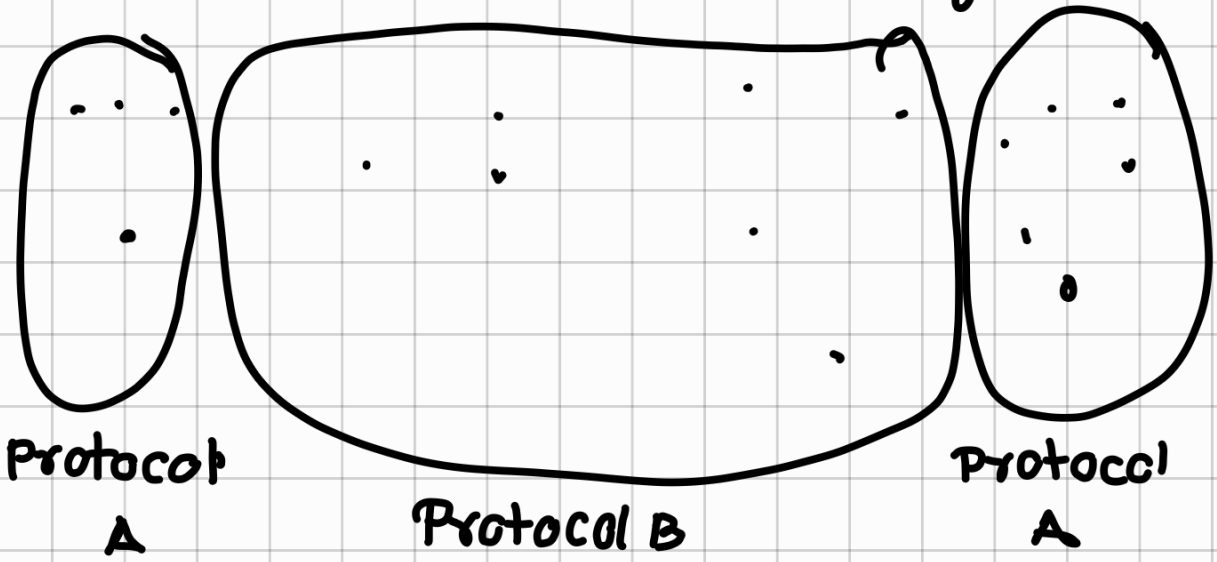


Tunneling

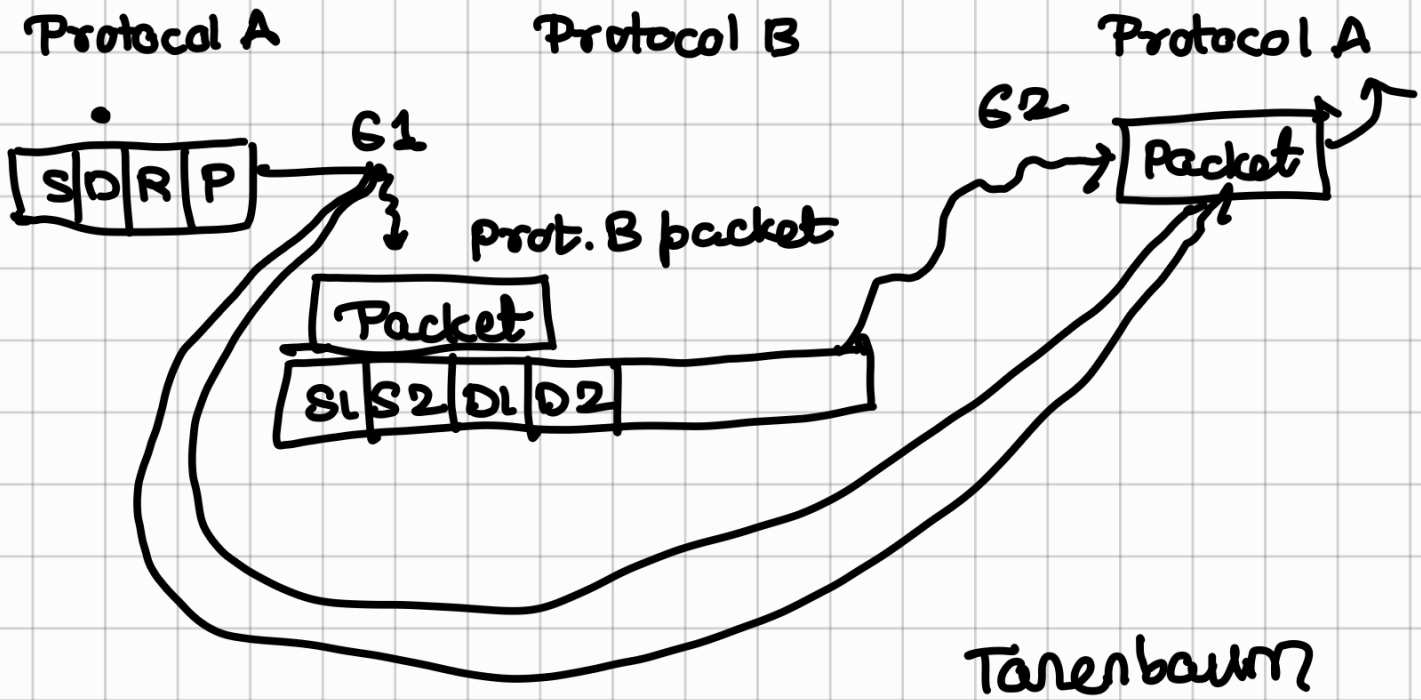




Refer Tanenbaum : Internetworking + Tunneling

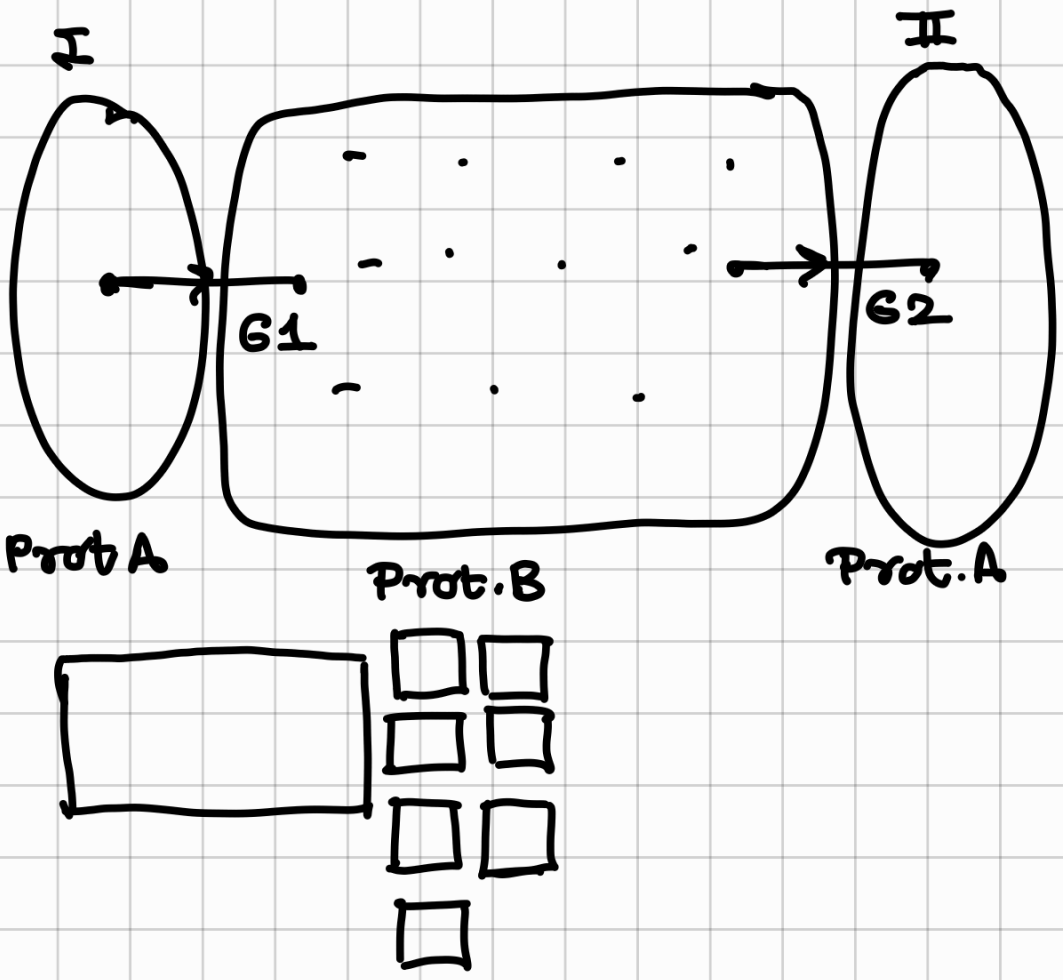


Tunneling process :



Tanenbaum

Internetworking + Tunneling + Packet Splitting



10 April Class Missed