Lec-11 Protocol Verification → প্রথমে verify করে তারপর code করে implementation করতে হয়। 13-03-24

Techniques to verify Models

× i) Finite state machine

✓ ii) Petrinet Model – এর সাহায্যে verify করে দেখব fruitful হবে কিনা
↳ কিছু notions আছে।

○ → state.

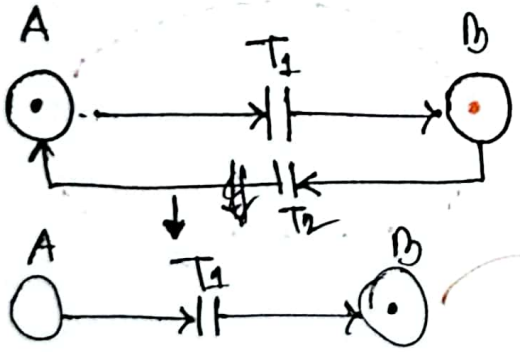
⊙ → a token within a state

• → token

—||— → Transition

→ Arc

complete example.



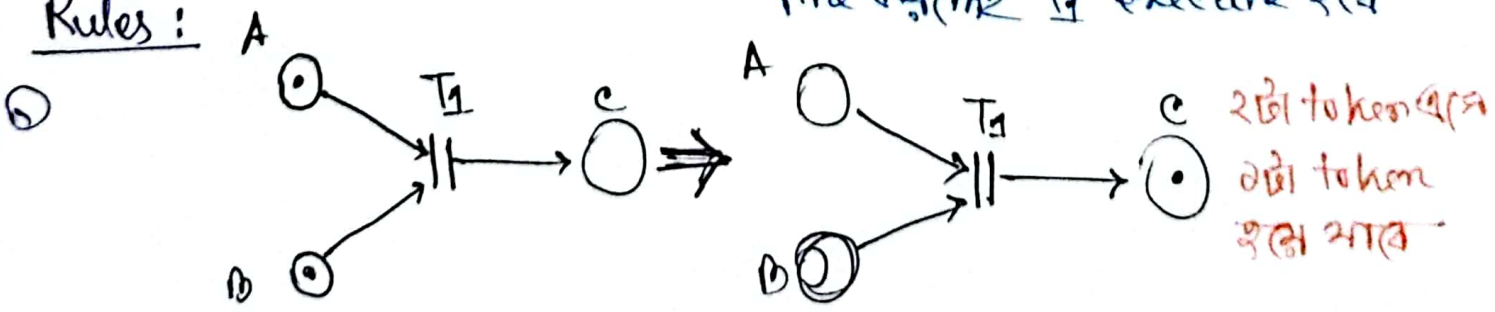
A থেকে B তে যেতে হলে T_1 transit/execute করে যেতে হবে। কেবলমাত্র A তে token থাকলেই তা যেতে পারবে

B তে আর কোনো state নেই তাই এখানেই শেষ হবে।

এখন A আর execute করতে পারবে না কারণ আর token নাই।

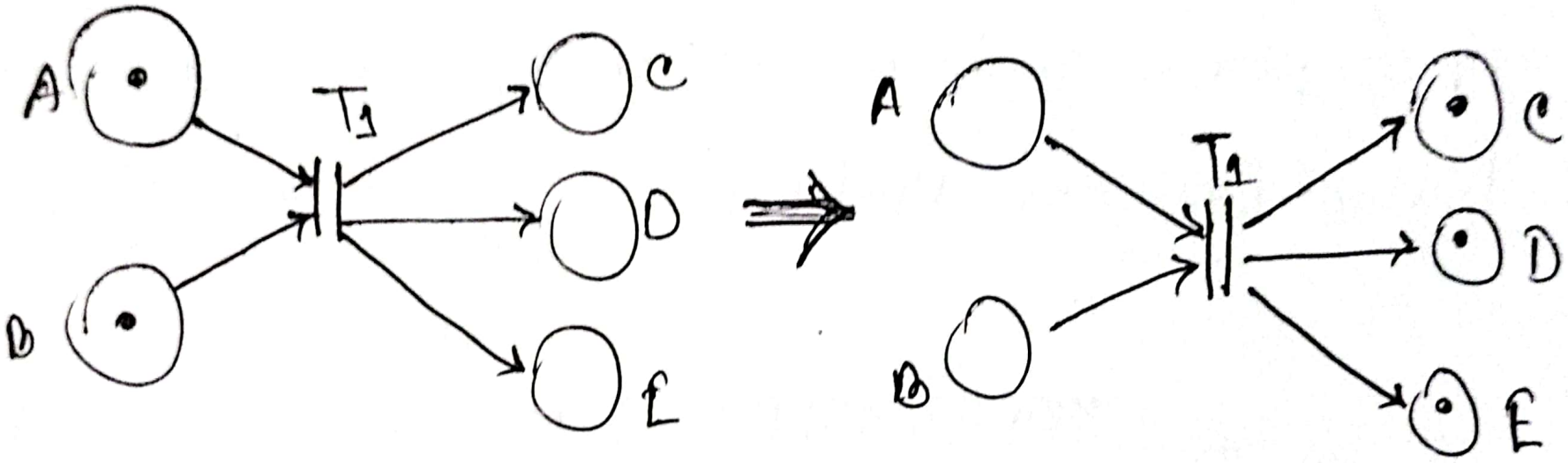
প্রথমে T_1 execute হবে না কারণ ২টা token থাকতে হবে, A, B ২টাই fire করলেই T_1 execute হবে

Rules :

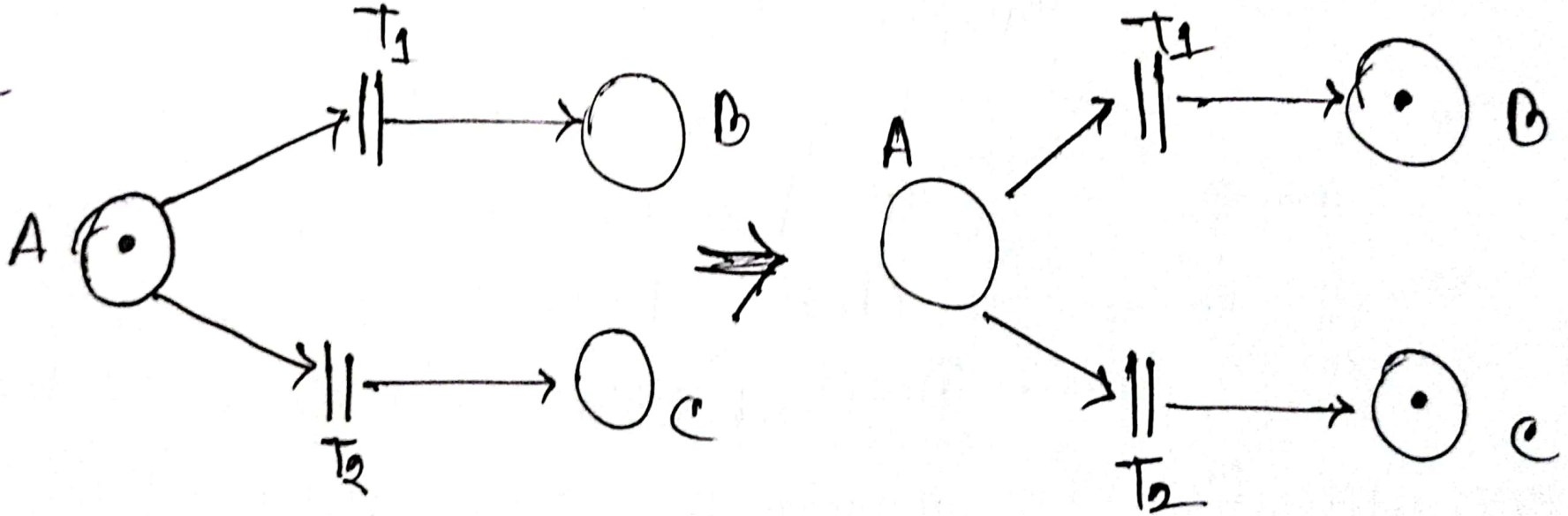


২টা token থেকে ১টা token হয়ে যাবে

(ii)



(iii)

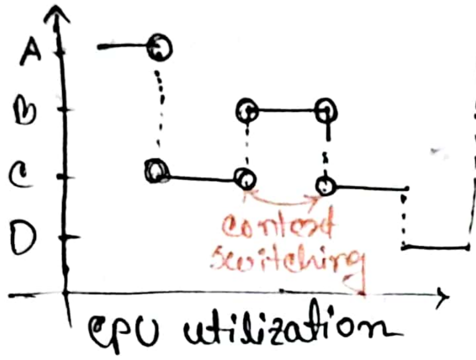


lec-12 18-03-24

By the help of petrinet model we will verify a protocol named 'Mutual Exclusion Protocol' – * very simple, mutually ২জনকে exclude করে, ২টা process একে অপরকে exclude করে

05 process

Ⓐ Ⓑ Ⓒ Ⓓ

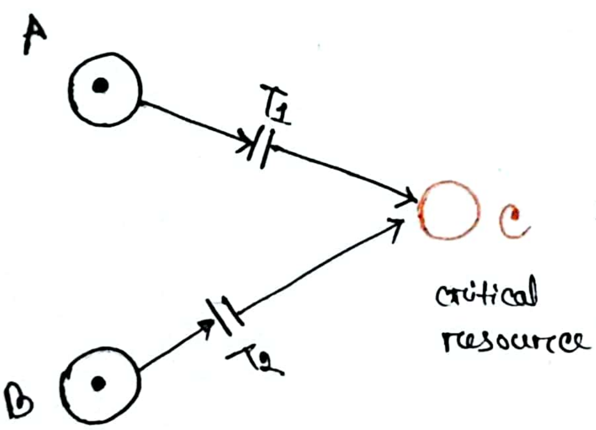


মনে হবে 4টা process parallelly কাজ করছে কিন্তু আসলে তা হয় না। cpu utilization কখনও overlap করে না। context switching এত fast হয় যে user বুঝতেই পারে না। (back & forth করে switch হয়)

Printer: ২জন একসাথে command দিলেও overlap হয় না কারণ printer job command নেয়। non sharable resource যাদেরকে বলে critical resource.

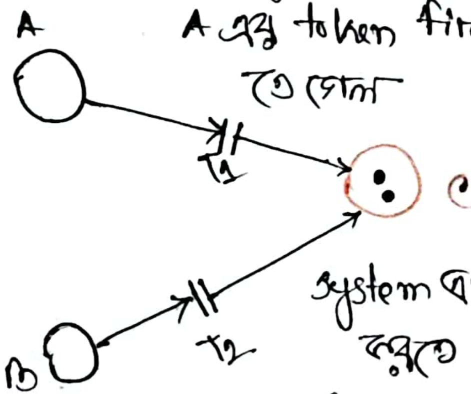
* all the hardwares & softwares are resource

* mutual exclusion এর মাধ্যমে critical resource কে শেয়ার করে থাকি
* একসময়ে শুধুমাত্র একজন use করবে
* A যখন use করে compell করে অন্যরা যেন use না করে

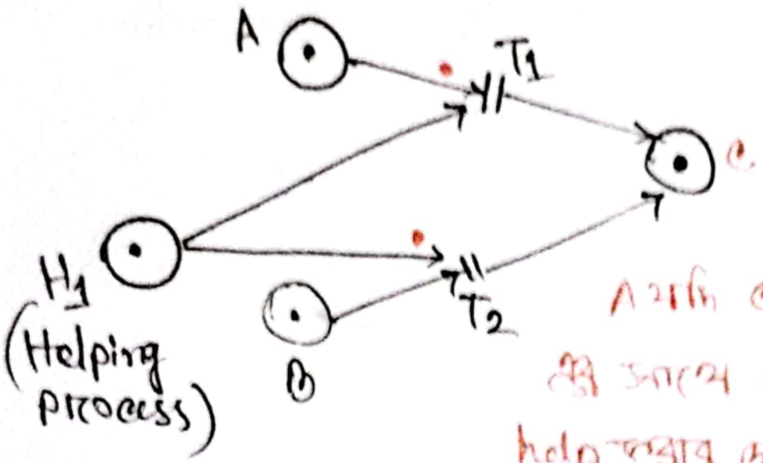


⟹

C is being used by A. কারণ A এর token fire করে C তে গেল

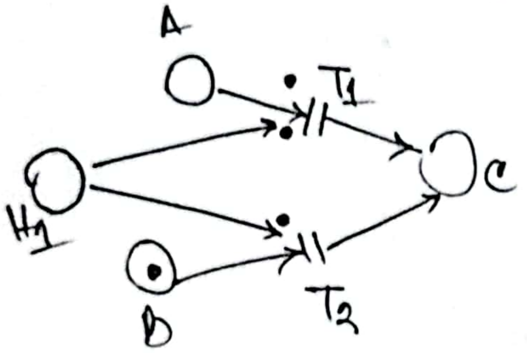


System একটাকে stop করতে পারল না। তাই বলা যায়, এই model useful না। A যখন use করছে B তখন access পাওয়ার কথা না। এখানে malfunction হচ্ছে।

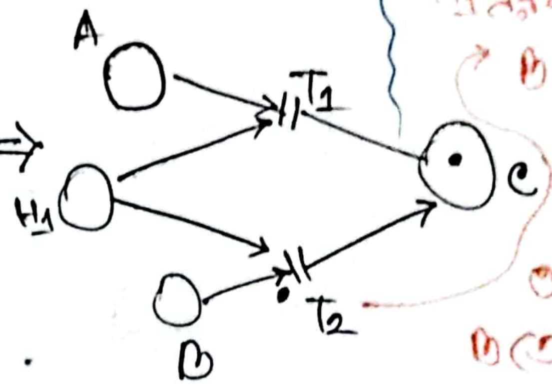


A, B = processes
C = critical resource
H1 = helping process (H2, H3)

A যদি communicate (fire) করতে চায় সে H1 এর সাথে communicate করবে। H1 অবশ্যই help করার জন্য। A, H1 কে বলবে I need to fire যেহেতু T1 এ token প্রয়োজন সেটা execute হতে পারবে না। B যেহেতু help চায় নাই তাই (•) / T2 execute হবে না (B fire করবে না)

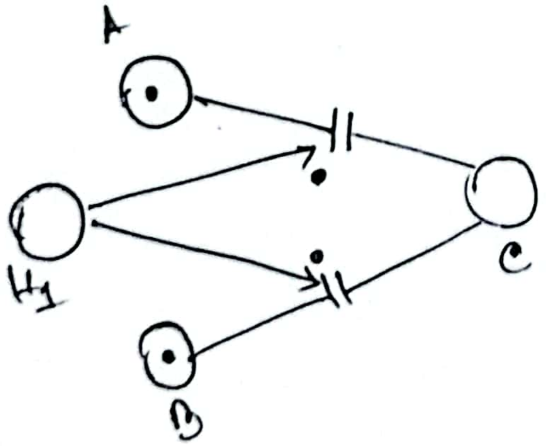


⇒

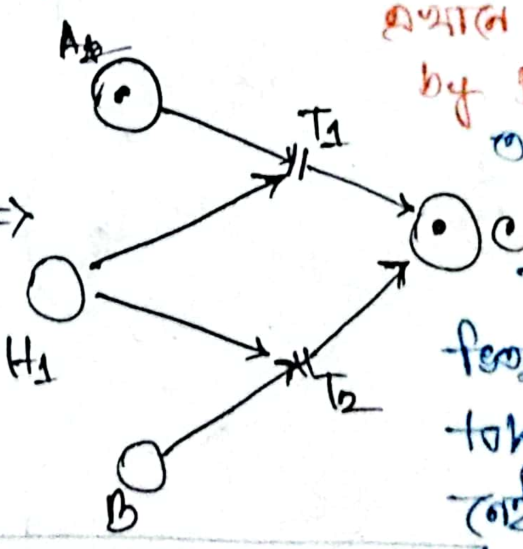


H1 এর মধ্যে আর token নেই B চাইলেও C কে use করতে fire করতে পারবে না তাই token আবার B তে চলে যাবে

এভাবে mutual exclusion কাজ করছে।



⇒

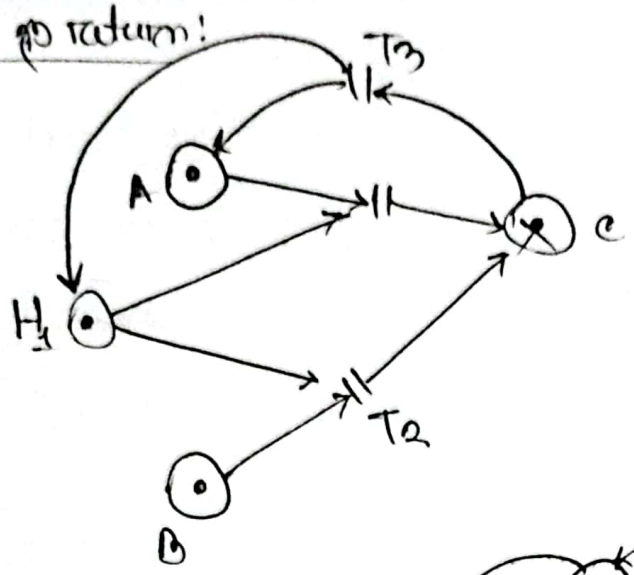


এখানে C is being used by B. A চাইলেও আর token fire করতে পারবে না কারণ H1 এর ভিতর আর কোনো token preserved নেই।

একজন use করলে আরেকজন use করতে পারবে না।

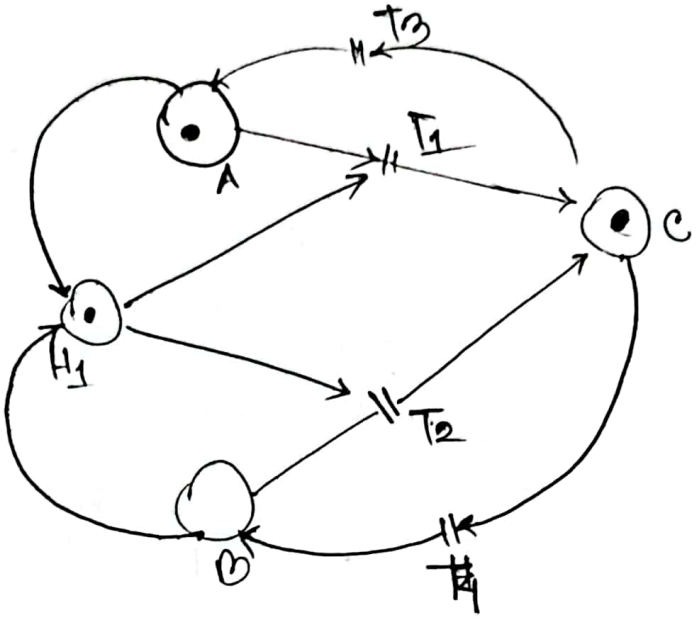
পুরো cycle complete করার জন্য আবার

A কে return!



এরর পাই হাত হলে আবার return করতে হবে।

B কে return



This model is not working properly ; তাই H_2, H_3 insert করা যায়

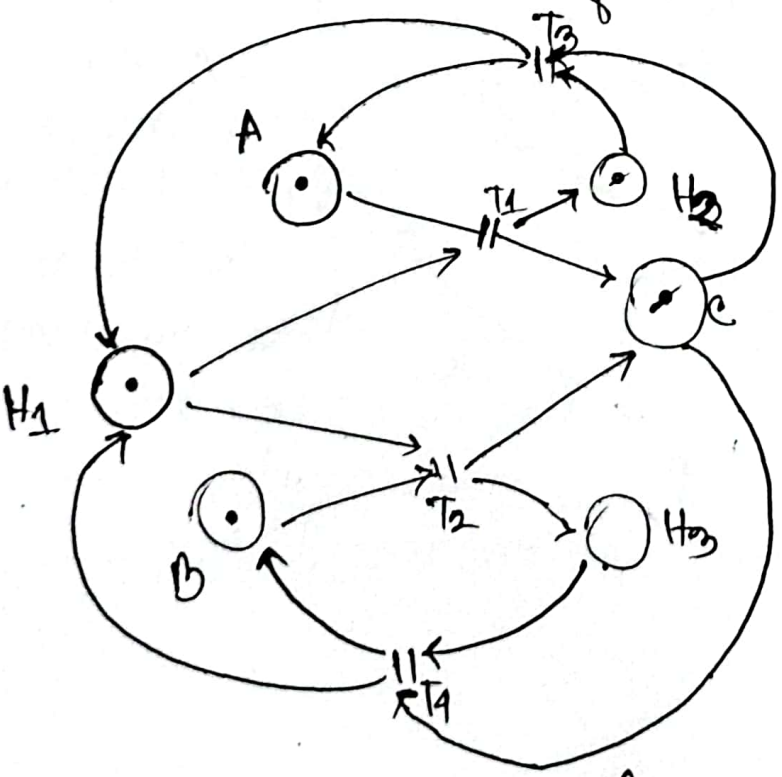


fig: Petrinet model of Mutual Exclusion

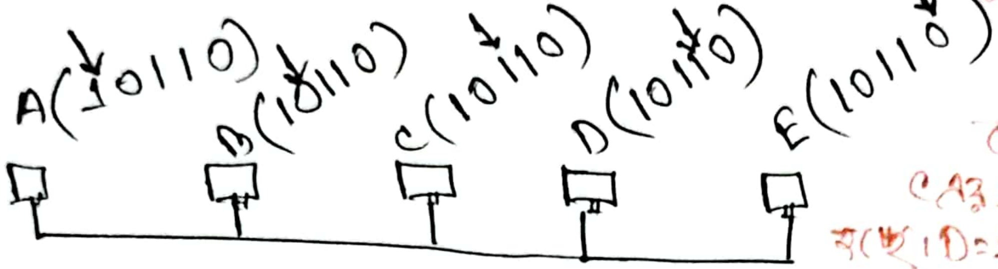
1-13/ 31-04-24

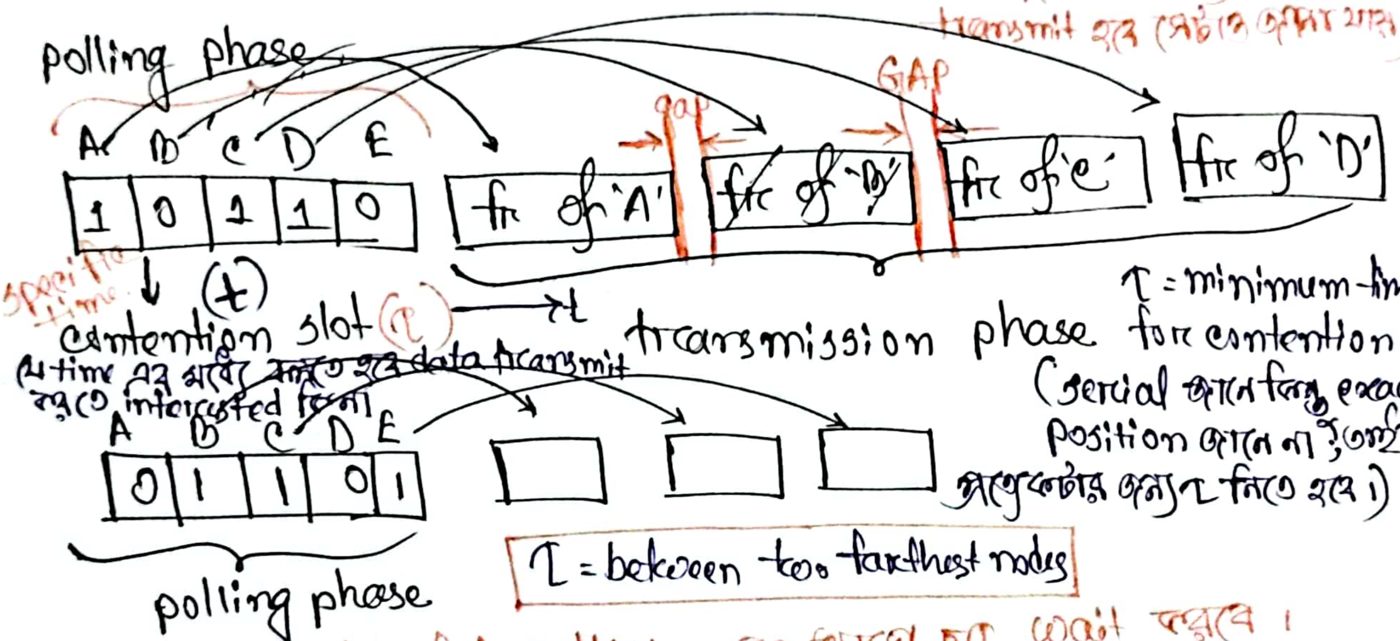
(collision free) (time consuming তাই CSMA/CD use হয়)

Channel Allocation এর (collision free protocols)

a) A bit map protocol (নিজেরাই decide করবে কে কখন transmit করবে)

(A যদি 1 transmit করে সেটা বাকি সবাইকে জানাবে; A এর পরে B transmit করবে। B 0 পাঠালে সেটাও সবার কাছে যাবে। C এর 1 সবার কাছে broadcasted হচ্ছে। D=1, E=0 ... এভাবে bit map create হবে। এর মাধ্যমে কে কখন transmit হবে সেটাও জানা যায়)





(t) contention slot (τ) → τ (যে time এর মধ্যে জানতে হবে data transmit করতে interested কিনা)

τ = minimum time for contention slot (serial জানে কিন্তু exact position জানে না, তাই প্রত্যেকটার জন্য τ নিতে হবে।)

τ = between two farthest nodes

একটার জন্য τ, 5 টার জন্য 5τ | প্রতিটা collision এর আগে 5τ wait করবে। — time consuming but collision free থাকবে

- polling phase শেষ হলে transmission শুরু হবে। যারা interested ছিল A, C, D শুধু data পাঠাবে এবং transmission শেষ করে আবার polling phase শুরু হবে। প্রতিবার polling এর পরে transmission হবে।
- এখন bit A এর জন্য কারণ ওর আগে 1 নেই। C এর আগে 1টা 1 তাই সে জানে 1টা trans হবে। D জানে ওর আগে 2টা trans হবে। A শেষ হলে সে আর signal পাঠাবে না তাই GAP create হবে। C, D sense করবে GAP আছে। C যখন detect করবে GAP আছে, তখন সে trans শুরু করবে, this is it's time to transmit. same as D. এভাবে collision free data transmit হয়।

b) Binary countdown Protocol

(bit length important) এটা হলো অনেক time নিবে convert হতে, এত বেশি তত faster)

A (1 00 01)	1	0	0	X	X
B (10110)	1	0	1	1	0
C (01001)	0	X	X	X	X
X (10000)	1	0	0	X	X
Y (10111)	1	0	1	1	1
Z (10010)	1	0	0	X	X
OR	1	0	1	1	1

bit length : 5 bits long of the address

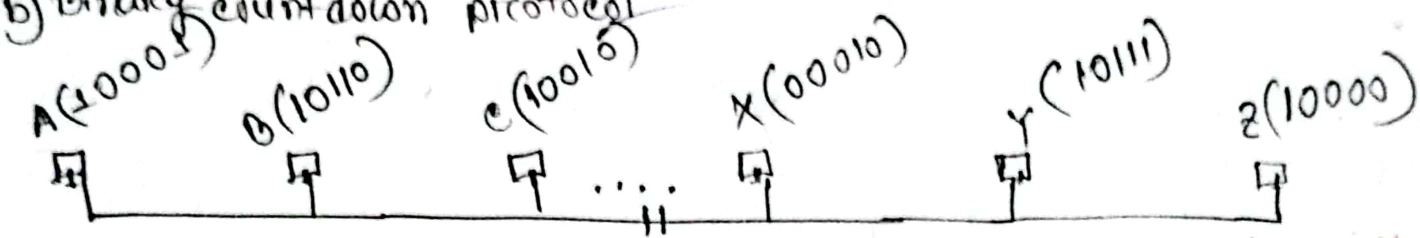
$2^5 = 32$

10111 (Y) → winner

যেটা OR এর সাথে match না করবে সেটা বাদ; এখানে যার decimal বেশি সে winner হয়েছে (interested দের মধ্যে highest number)

* channel না পেলে starvation

b) Binary count down protocol



প্রত্যেকে প্রত্যেকের MSB কে transmit করবে।

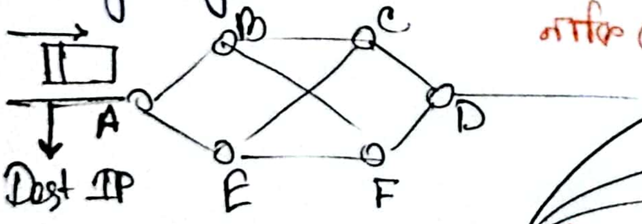
A(10001)	1	0	0	x	x
B(10110)	1	0	1	1	0
C(10010)	1	0	0	x	x
X(00010)	0	x	x	x	x
Y(10111)	1	0	1	1	1
Z(10000)	1	0	0	x	x
OR	1	0	1	1	1

bit length এর সমান করে রাখতে হবে।

→ (10111) winner 'Y'

00000 — lowest num; শুধুমাত্র একা থাকলেই data পাঠাবে।
— less chance of winning ~~[illegible]~~
(starvation) — CPU যখন process time পায় না

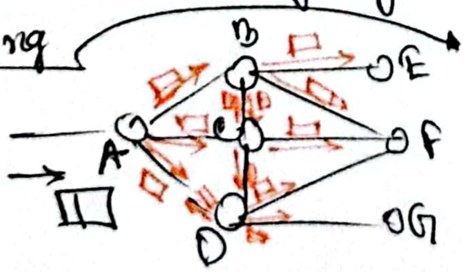
Routing Algorithm → এটা use করে A decide করবে packet B তে নাকি C তে যাবে।



→ Shortest path routing X
→ flooding (selective flooding)
→ optimality principle

— static routing algorithm (non-adaptive) — fixed route
— dynamic routing algorithm (adaptive)

flooding → incoming ছাড়া outgoing এ copy পাঠাবে

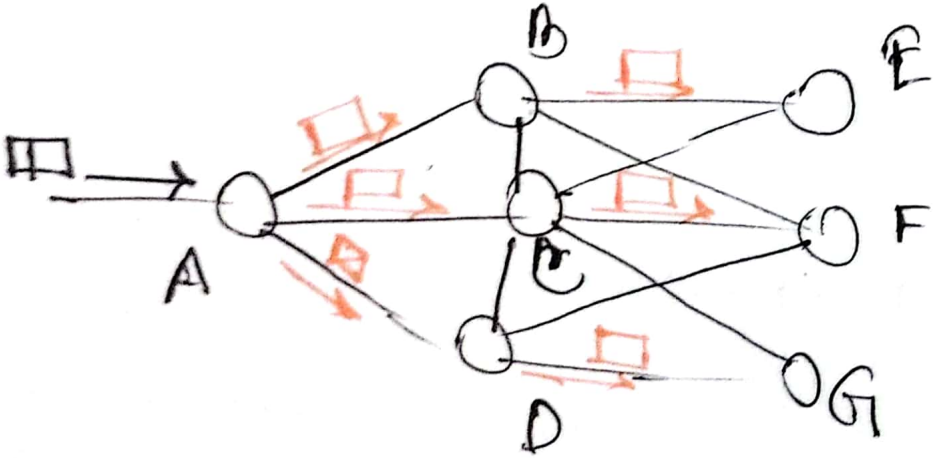


It's not a good concept

Ex: mobile networking

Selective flooding

B, C, D A এর neighbour. এই packet কপালের যাবে, যদি neighbour না থাকে তখন packet পাঠাবে।



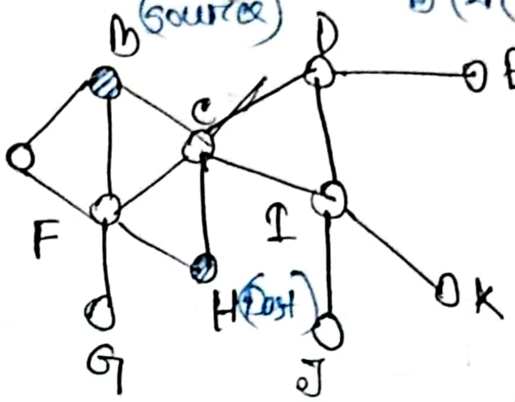
আবার C এর সাথে E, F, G common neighbour of C. এই...

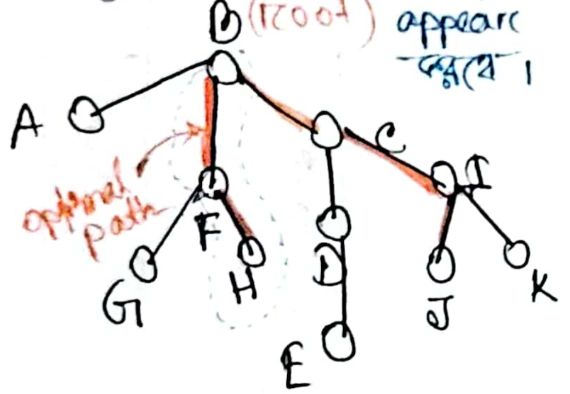
P-14 3-04-24

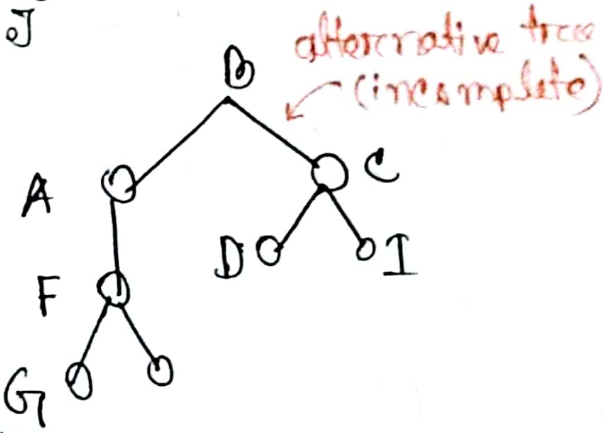
- Static routing algorithm
  - shortest path routing
  - flooding (selective flooding)
  - optimality principle
- Dynamic " "

প্রত্যেকটা node নিজেকে root মনে করে tree create করবে যাতে
কে প্রত্যেকটা router এ একবার করে appear করবে।

B থেকে H এ পাঠাতে হলে B নিজেকে source হিসেবে root ধরবে।



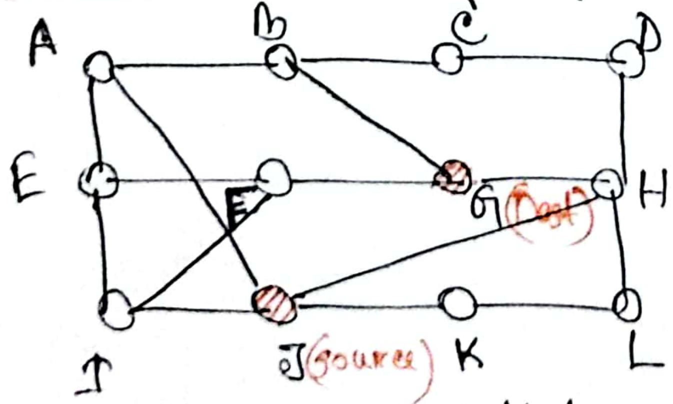




Dynamic Routing Algorithm

- Distance vector Routing Algorithm
- Link state Routing Algorithm

Distance vector Routing Algorithm

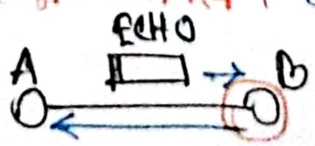


J's neighbours : A, I, H, K

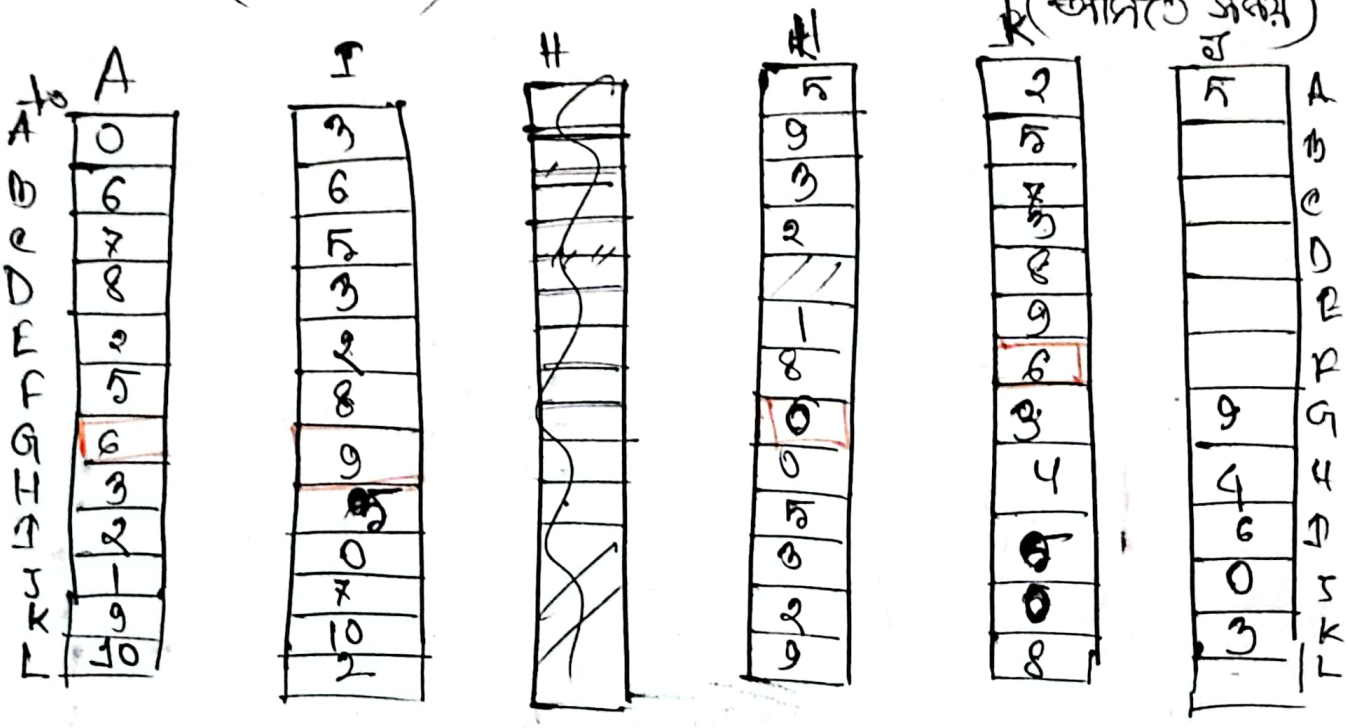
প্রত্যেকটা router এর মধ্যে routing table থাকে যেখানে info থাকে। J তে একটা routing table থাকে যার dest, neighbour এর থাকবে। G তেও same, কিন্তু তাদের মধ্যে direct connection নেই, 1 এর distance থাকবে তাদের নিয়ে কাজ করতে হবে।

ECHO Packet (controlling packet)

যার কাছে পাঠাবো সে সেখান থেকে আবার ফেরত আসবে। B তে এসে process করবে



latency : propagation delay + processing time + propagation delay
(যাওয়ার সময়) (আসার সময়)



ECHO

JA : 5 msec

JI : 6 msec

JH : 4 msec

Jk : 3 msec

JG : JA + AG = 5msec + 6msec = 11ms

JG : JI + IG = 6 ms + 9ms = 15 ms

JG : JH + HG = 4 ms + 6ms = 10 ms

JG : Jk + kG = 3 ms + 6 ms = 9 ms

JA packet ফেলে যাবে

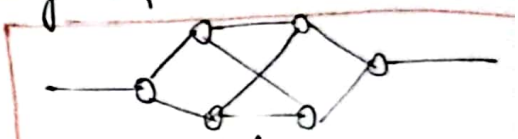
17-09-24

Lec-15

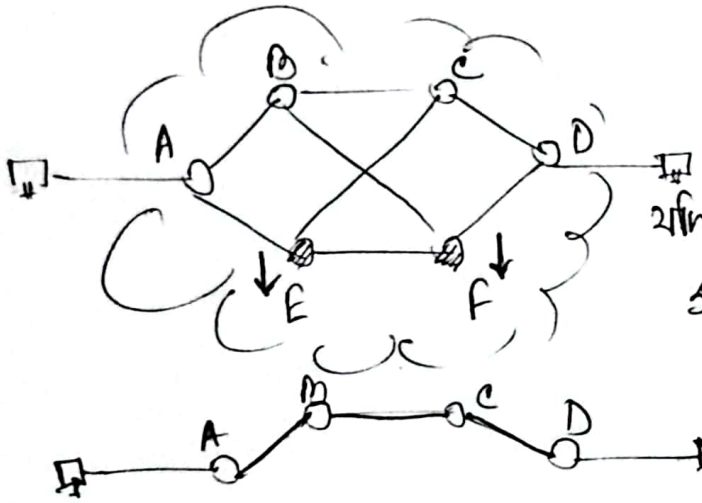
Dynamic Routing Algo
- Distance vector Routing algo
- Link state Routing algo

- we have a problem in Distance vector Routing Algorithm.
- যদি subnet টা linear হয় তাহলে problem হয়।

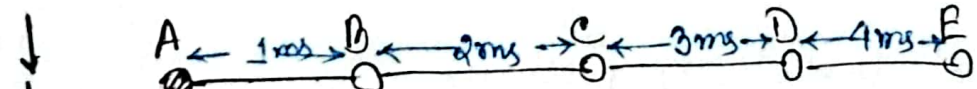
Linear subnet: যখন routers গুলো sequentially থাকে


not linear

এখানে path choosing-এ options আছে।



যদি E, F down হয় তাহলে যেটা linear subnet হিসেবে গণ্য হবে। তখন count to infinity problem হতে পারে।



Initially → 1 (2+1) =3 (3+2+1) =6 10

(A তে reach করার জন্য initial value)

A is Down initially

(null)

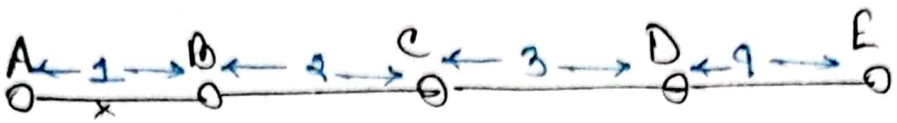
A is up (Good news)

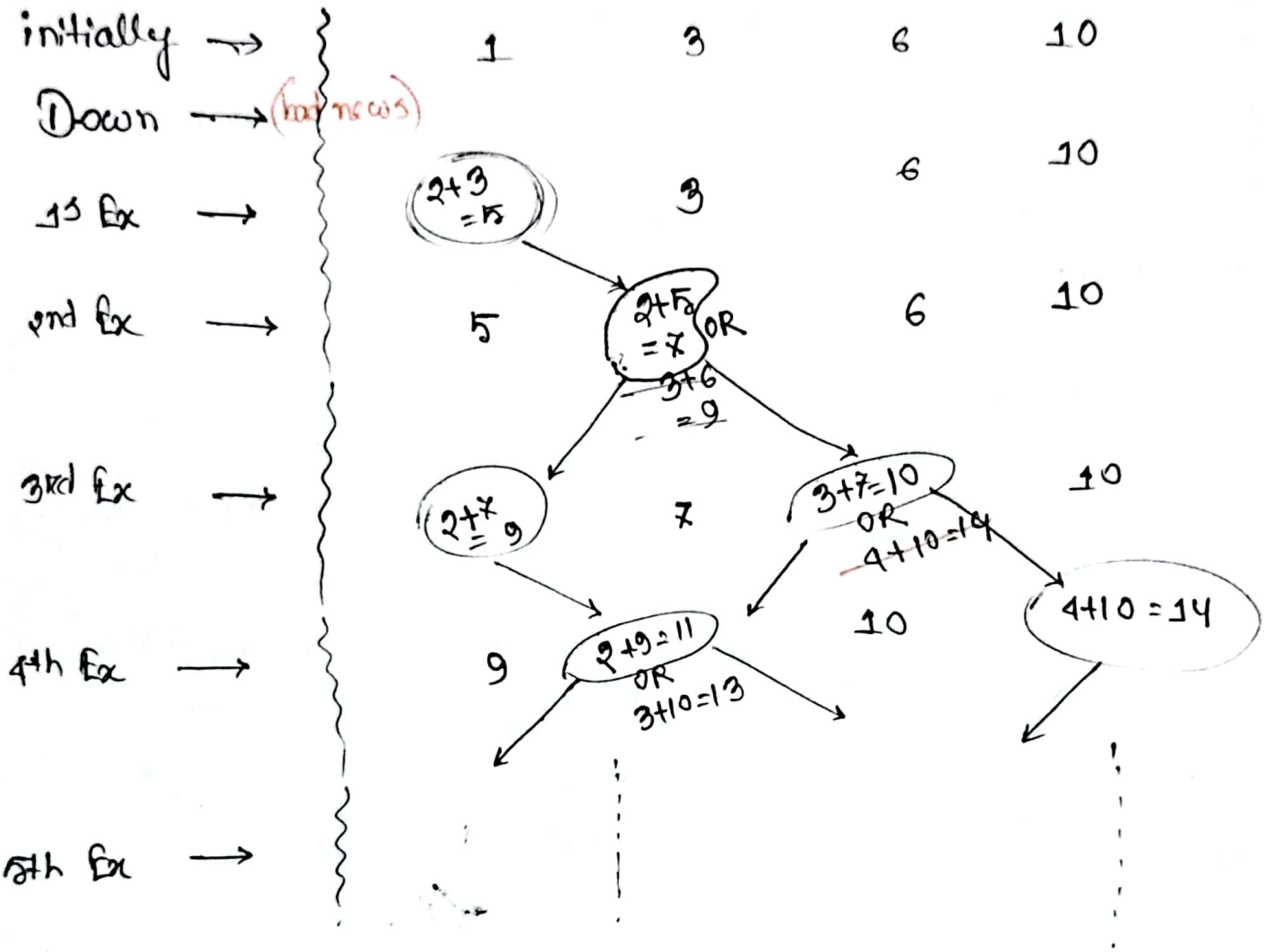
1st exchange → 1 (কারণ, neighbour এর update করবে)

2nd exchange → 1 (2+1) 3

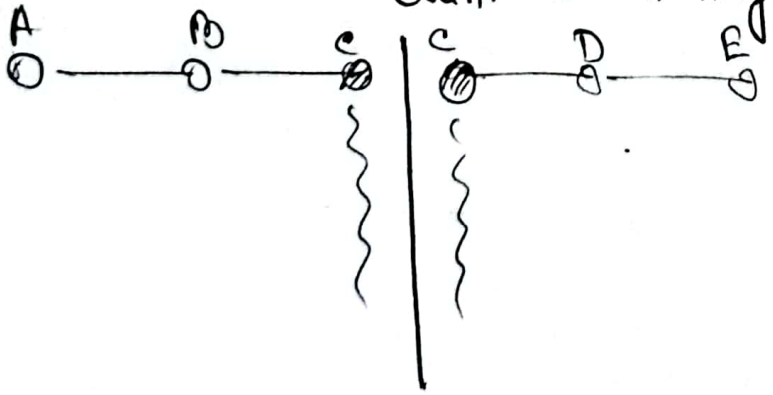
3rd ex → 1 3 (3+3) 6 (4+6)

(up to down after trouble)





Count-to-infinity problem



22-04-24

lec-16 Dynamic Routing algorithm
- Distance vector Routing algo
- Link state Routing algo

Link

Steps:

i) know your the neighbours [HELLOW packet provide/generate করে] (এখানে সে তার neighbour দের জানায় তার IP কী → আর কিভাবে IP আছে)

ii) Measure the delay [ECHO packet generate করে measure করা হয় (latency)]

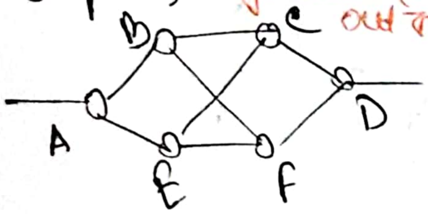
iii) build link-state packet

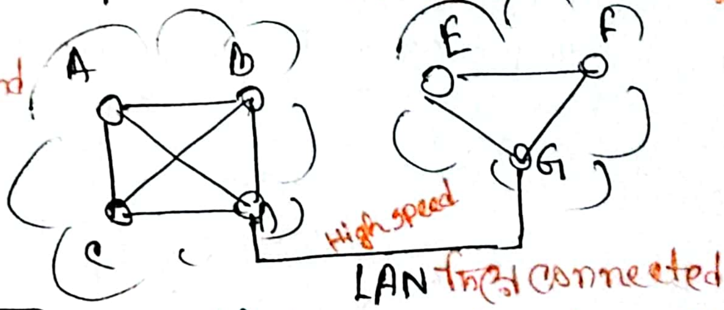
iv) distribute link-state packet

v) compute the shortcut packet

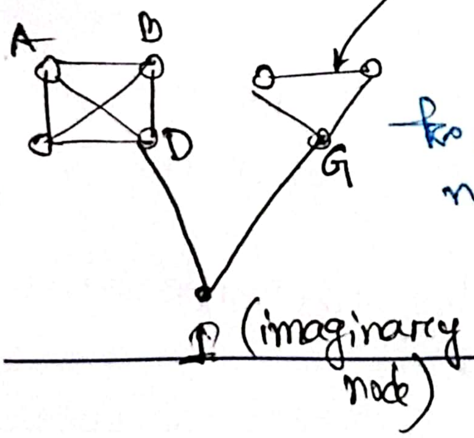
(এখানে LAN node হিসেবে কাজ করে। যেহেতু LAN এর ভিতর দিয়ে যেতে কত সময় লাগবে তা certain না তাই node count করবে।

for step 1, প্রথম কাজই হল neighbour কে find out করা





এখানে D এর সাথে ABC যেভাবে সরাসরি connect আছে G কিন্তু সেভাবে connected না কিন্তু D → G তে যাবে কখন সেটা define করা যায় না; G কে win হবে কি হবে তা জানিনা, এমন situation এ imaginary node connect করা হয় যা LAN এর প্রতিনিধিত্ব করে।

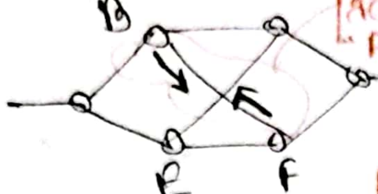


for step 2, latency count করে avg করে বের করা হয়

B → F

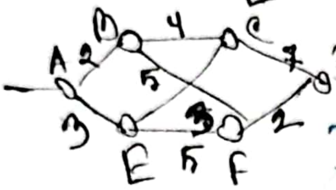
p. latency = propagation Delay (B → F) + processing time + propagation delay (F → B)

[যেহেতু P.D possibly same হয়]



= 2 propagation delay + processing time

B থেকে F এবং F থেকে পুনরায় B তে আসতে total যে time লেগেছে সেটা prop. delay। এরপর avg করে আবার করা হয় সেটা হচ্ছে latency



Delay calculate করতে latency consider করতে হয়

for step 3. (controlling packet যেটা routing table গুলো share করে) link-state packet

for B

B	
seq #	
Age	
A	2
C	4
F	5

→ কেন seq num generate করা হয়েছে?

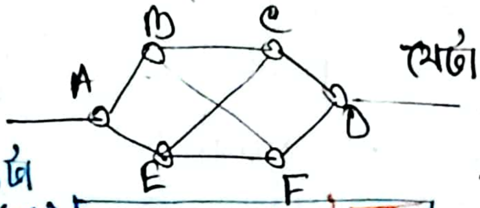
this is link-state packet of 'B'

x, x+1 ←

x+2, x

প্রতি 10 sec পরপর update generate হতে থাকে এবং neighbour কে update করে থাকে

যখন increment (x+1) হবে সেটা latest value carry করবে; same as (x+2) তেও হবে। update হলেই এর value নিয়ে নিবে x এর টা delete হয়ে যাবে। (x+2) compare হবে (x+1) এর সাথে then (x+1) delete হবে (x+2) এর info নিয়ে routing table এ রাখবে। এরপর অন্য কোথাও থেকে পরবর্তীতে x আসলেও (x+2) এর সাথে compare করে দেখবে (x+2) বড় তাই x delete হবে



যেটা bigger seq # সেটা রাখবে

seq # যত বেশি তত better

B	
sequence	
Age (80)	

length of the sequence# = 4 Bits long (16 টা)

X 0000
✓ 0001
✓ 0010
⋮
✓ 1110
✓ 1111
0000 X
0001
⋮
X 1111

compare করে আগেরটাকে delete করবে

বিশৃঙ্খলা হবে

receiver এর কাছে compare করে দেখবে যে এটা (0000) 1111 এর চেয়ে bigger না

নতুনগুলো latest info carry করলেও এগুলো receiver নিবে না

এখানে শেষ হয়ে গেলে আবার নতুন করে শুরু হবে; length তাই bit বেশি রাখতে হবে

নতুন যত কিছুই আসুক আগেরটাই রাখবে

Solⁿ হচ্ছে,

sequence # length : 32 bits (অর্থাৎ আর wrap up হবে না)

[2^{32} টি packet পাওয়া যাবে]

(122 years লাগবে এই number শেষ হতে)

হঠাৎ করে problem দেখা দিতে পারে . .

× Seq #1
× Seq #2
× Seq #3
✓ Seq #4
× Seq #5
× Seq #6
⋮
× Seq #44
✓ Seq #45

sender 4 send করেছে কিন্তু → read error → 44 → এখানে reader read করল 44) receiver এখন 44 কে 3 এর সাথে compare করে 44 নিয়ে নিল। তারপর sender 5 generate করবে। একটা সময় 44 generate করবে। তখন এই 44 আগের 44 এর সাথে compare করে, দেখবে তার তো আগে থেকেই 44 আছে। তাই নতুন 44 নিবে না বরং bigger number এর জন্য wait করবে অর্থাৎ 45 এর জন্য wait করবে।

seq 5 to 45 পর্যন্ত 40×10 = 400s পর্যন্ত less information নিয়েই কাজ করেছে। এখানে sender packet পাঠালেও receiver বুঝতে পারছে না যে সে less info/error নিয়েই কাজ করছে। সবগুলো seq ই latest info carry করেছে কিন্তু receiver accept করেনি।

seq 5 to 45 কে বলবে "backdated information"

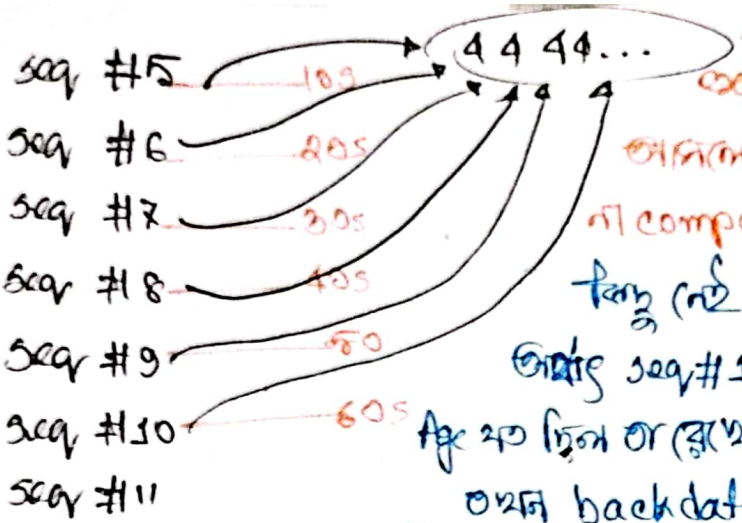
যদি 44 এর স্থলে আরও বড় number হত তাহলে trouble বেশি হত।

যদি MSB change হয় বাকী সব ঠিক থাকে তাহলে error হবে।

0000 0000
0000 . . . . 0100 (4)
1000 0100

(a single bit error can change huge)

(4444 . . . .) এমন situation এ উক্ত নাম্বার না আসা পর্যন্ত backdated information নিয়েই কাজ করতে থাকবে। এই ধরনের trouble solve এর জন্য "Age" add করে দিয়েছে। মানে হচ্ছে যেই information টা কতক্ষণ পর্যন্ত sustainable. (just like TTL) এখানে Age = 80 insert করলে এ সময়ের জন্য এ number টাকে রাখবে তার buffer এ, কিভাবে কাজ করে? → Age টা হচ্ছে একটা specific time কতক্ষণ পর্যন্ত এই information টা valid থাকে। যদি Age = 60 হয় তাহলে 60s এর জন্য এটা valid হবে। এটা রাখবে receiver তার routing table এর ভিতরে। routing table এর সাথে clock আছে; এর একটা power আছে। clock যখন expire হয়ে যাবে তখনই info টা delete হয়ে যাবে।



এখন 60s পার হয়ে গেলে এই packet expired হয়ে যাবে। তারপর seq #11 এলেও সেখানে আর কোনো packet থাকবে না compare করার জন্য। compare করার কিছু নেই তাই সে সেটাকে accept করে নিবে অর্থাৎ seq#11 accept করে নিবে। আর 11 এর জন্য Age যত ছিল তা রেখে দিবে। suffering for 60s যে তখন backdated info নিয়ে কাজ হচ্ছিলো। এরপর আর suffer হবে না। this is how যদি router-A reader যদি থাকতো তাহলে কিভাবে life শুরু করে। যদি 60s এর পরিবর্তে 10s/5s রাখা হত তাহলে suffering কম হত। কিন্তু তখন 10s পর পর packet update হওয়ার আগেই expired হয়ে যেত। তাই 10s নিবে না। (20s রাখা যেতে পারে) সেক্ষেত্রেও suffering থাকতে পারে। তাই সব বিবেচনা করেই 60s পর পর generate হওয়ার সুযোগ দিচ্ছে। consecutively 6টা packet (usual packets) miss করা মানে router-A huge trouble আছে। সেজন্য Age = 60s আর 32 bit রাখা হয়। this is how age saved you from getting into huge trouble.

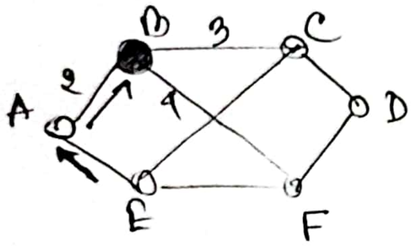
lec-17

Link-state Routing Algorithm

23-04-24

Steps:

i) know the neighbours (HELLO)
ii) Measure the delay (ECHO)
iii) build link-state packet
iv) distribute link-state packet
v) Compute the shortest path



controlling packet (heading is very simple)

B	
seq #	
Age	
A	2
C	3
F	4

A	
seq #	
Age	
B	–
E	–

Distribute link-state packet (প্রত্যেকটা router এই table তৈরী করবে)

for 'B'

(Owner: যার link-state packet; Source: যার কাছ থেকে এসেছে)

Owner	Source	Seq #	Age	Packet sent A	Packet sent C	Packet sent F	Ack (acknowledgement) sent A	Ack sent C	Ack sent F
A	A	(36)	60	0	1	1	1	0	0
E	A	62	60	0	1	1	1	0	0
F	C	42	60	1	0	0	0	1	1
~~A~~ (accept করবে না)	~~F~~	~~(31)~~	~~60~~	~~0~~	~~1~~	~~0~~			

(Packet sent and Ack sent: complement of each other)

→ A source তাই 0; C, F এ পাঠাবে তাই 1

→ C source তাই 0; F নিজেকে দিবে না তাই 0;

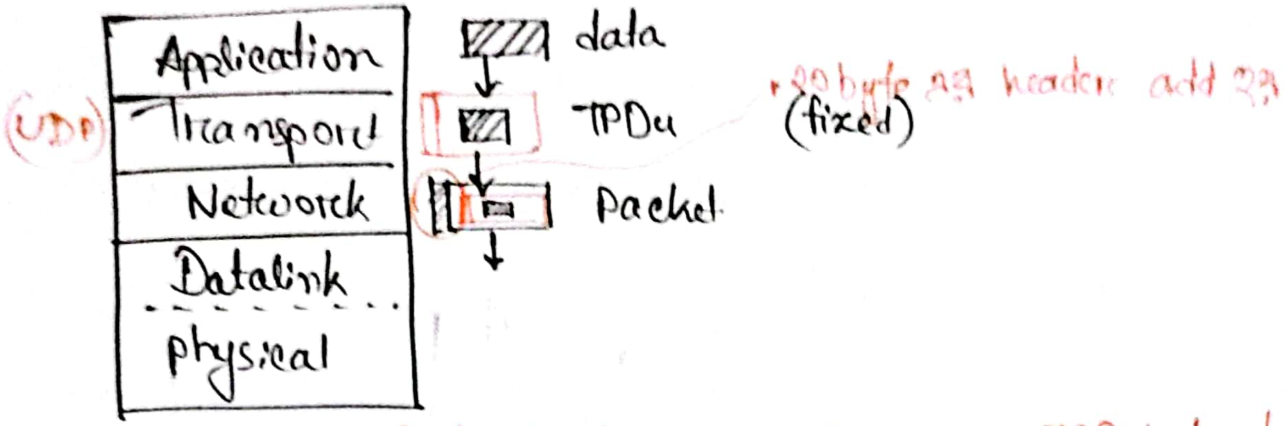
→ আগেই A থেকে যখন seq # 36 ছিল। পরে seq # 31 এর জন্য update হবে না অর্থাৎ accept ই করবে না

বাকী compare করে দেখলো আগে 36 ছিল

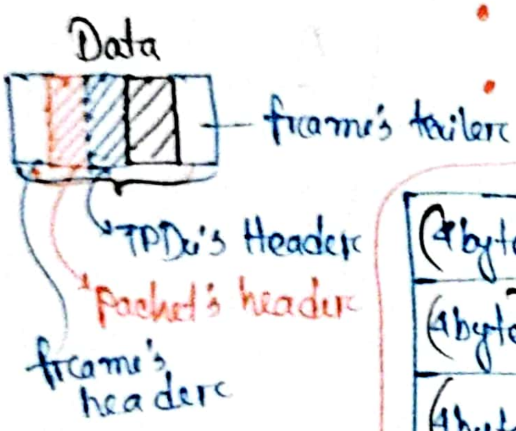
Compute the shortest path (~~Djistra algo~~)

(Dijkstra Algorithm)

TCP/IP Ref. Model



- Packet header হলো যেখানে TPDu's header শুরু
- TPDu's header " " Data "



(empty bit) 16 Bits

(4byte)	Version	IHL	Type of service	Total Length
(4byte)	Identification (?)	DF	MF	fragment offset (ব্যবহৃত bit)
(4byte)	TTL (128)	Protocol	Header check sum	
(4byte)	Source IP			
	Destination IP			
	Options			

20 Bytes

Precedence/priority

3 Bits	D	T	R

- IPV4 Header
- IHL (internetwork header length)
  ↳ 20 Byte

কোনটা wire হয়ে বেশি more reliable

D = Delay
T = Throughput
R = Reliability

(data rate) [per unit time এ কি পরিমাণ data পাঠাচ্ছি]

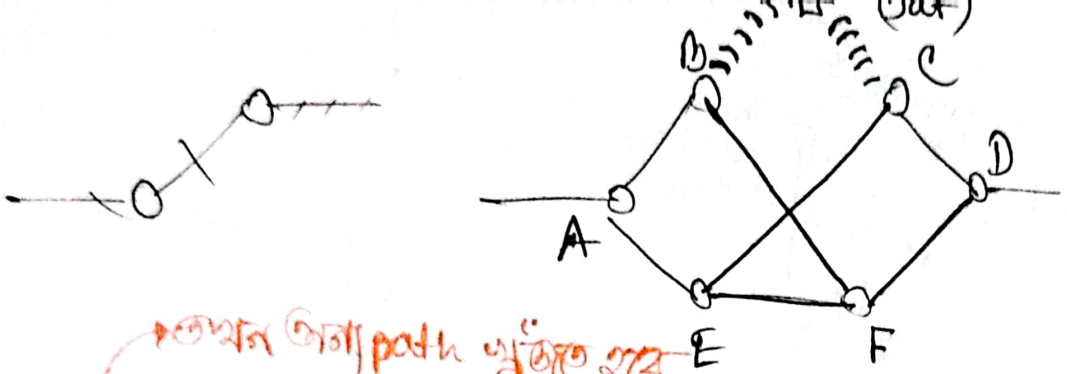
(router এর packet কিভাবে treat করবে → 3 bits এর উপর depend করে)

000 → Normal Data Packet

⋮

111 → Controlling packet

D=1 মানে you're expecting some delay



তখন অন্য path খুঁজতে হবে

DF : Don't fragment → যেখানে সেখানে ভাগে ভাগ করা

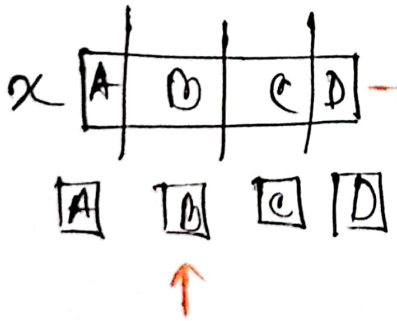
MF : More fragment

Ethernet : 1460 Byte (max)

ATM(max) : 53 Byte

বড় packet থাকলে fragment করে দিতে পারবে যদি permission থাকে (depending of type)

fragment offset : tells you the position of the fragments in the packet



যদি reassemble করতে হয় তাহলে position লাগবে

IPV4 Header

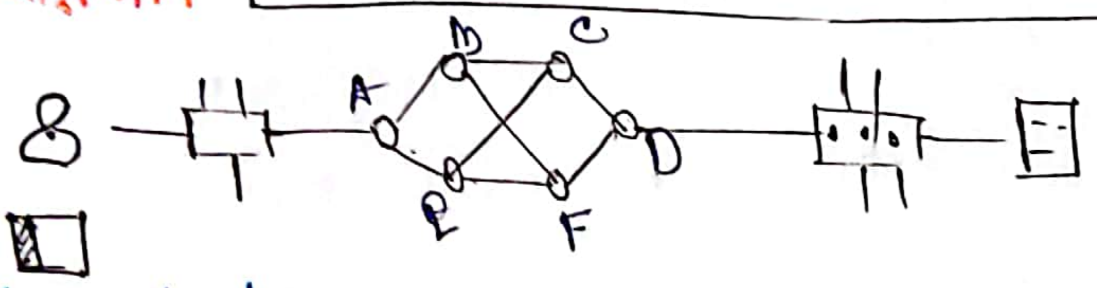
16 Bits

Ver	HL	Types of service	Total length		
Identification			DF	MF	fragment offset
TTL (04)		protocol (UDP)	Header Checksum		
Source IP (32 Bit)					
Dest IP (32 Bit)					
options					

Time-to-live (কতক্ষণ থাকবে sustain করে)

20 Byte

traffic হলে পারে তাই source & dest এখনো TTL কম হবে সেটা specifically জানা যায় না।



packet এর header

এই packet টা network এ রাখব → TTL এ unit নেই কেন?
(ওরা এ age কিন্তু decrement হয় না ; এ time পর্যন্ত into buffer এ রাখে তার unit থাকে)

→ এখানে clock দেয়া যায় না, তাই একটা number দেয়া হয় ৩৩ সংখ্যক hop count করবে। একটা device cross করলে ক্রমানুসারে decrement করবে।

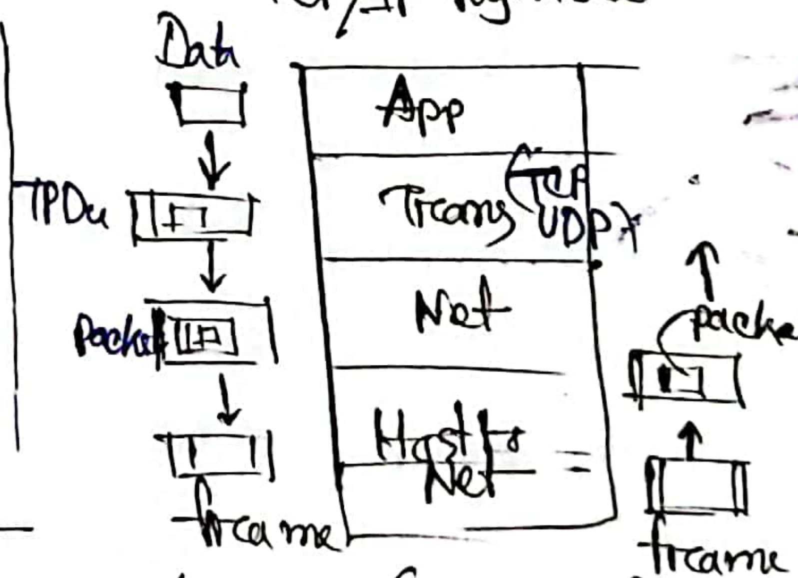
Protocol - (8 bit)

→ Packet's header এর ভেতর লেখা থাকে TPDu এর কোন IP কে handle করবে সেটা ; যদি
→ TPDu এর ভিতরে লেখা থাকে সেটা TCP নাকি UDP
— source থেকে multimedia থেকে throughput এর ভেতর দিয়ে যায় তখন তা lighter হতে হয়, যত বেশি simple তত বেশি data যেতে পারবে। এক্ষেত্রে UDP খুব liter. (TCP heavy)

[The TPDu made of UDP is more simple than the TPDU made of TCP]

Protocol এ যদি TCP থাকে সেটা .... UDP থাকলে ...

TCP/IP Ref Model

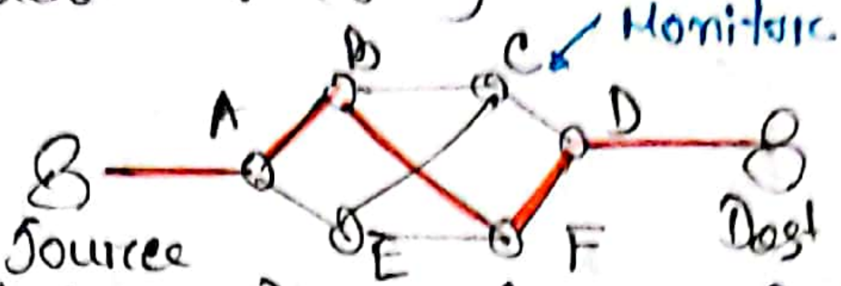


Header checksum: check করবে error আছে কি না [cyclic redundancy check (CRC)] → (16 Bit)

Options:
i) Security
ii) Strict Source Routing
iii) Loose Source Routing
iv) Record Route
v) Record time

Security: How secret this packet is? (শুধু কেউ এটা num দেখা যেতে পারে)

Strict Source Routing:
(Dedicated Path is given from source to destination) ex: (A, B, F, D) — dedicatedly follow করবে



Loose Source Routing:
(List of routers given that can't be missed) while going from source to somewhere — while the packets are being transferred.
→ ex: C ; source থেকে dest এ যেতে হলে কোনোভাবেই C miss করে
(কোনো specific router যেটা বাদ দেয়া যাওয়া যাবে না — যাবে না)

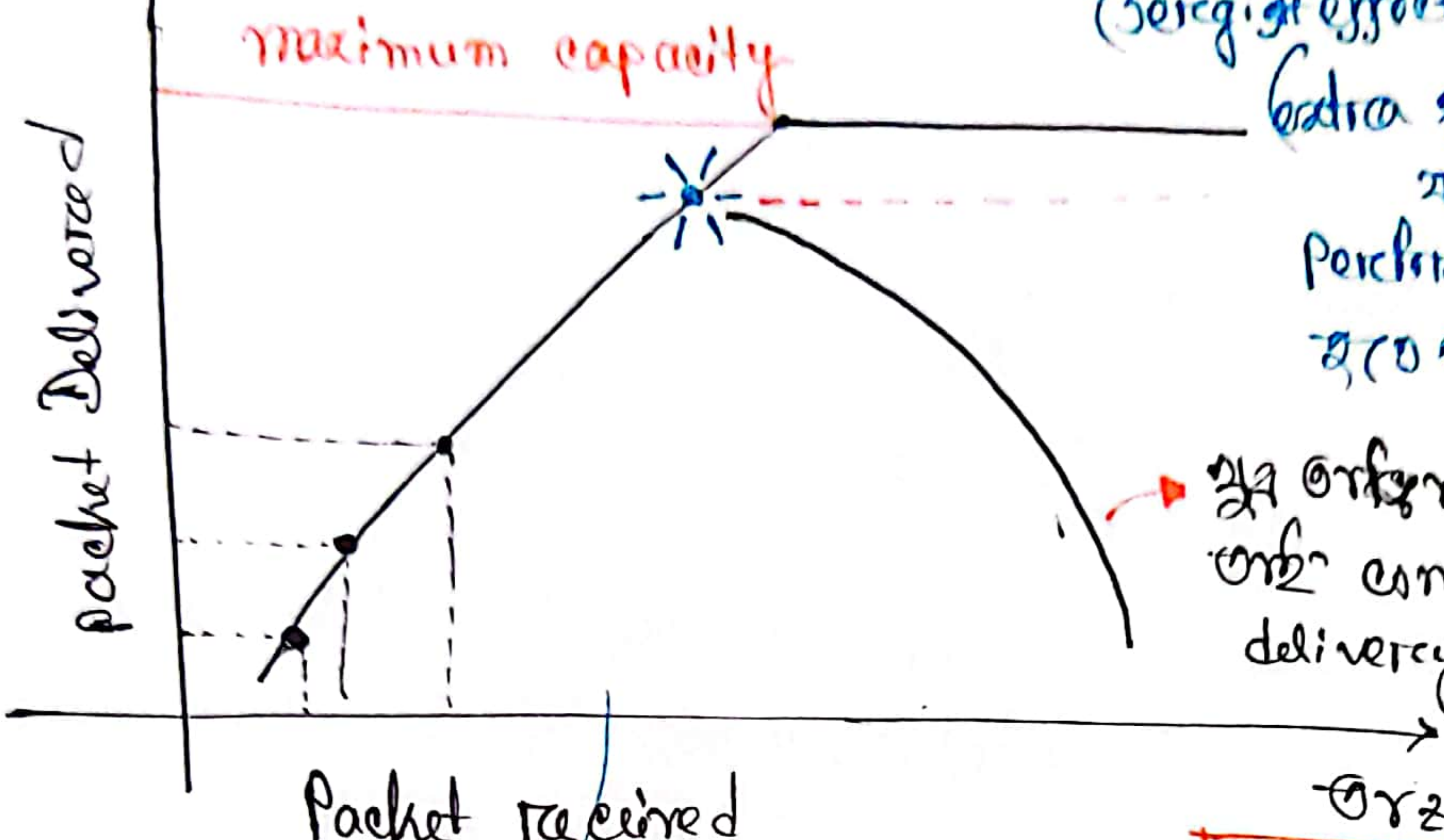
Record Route: Dest এ যাওয়ার পর দেখা যাবে কোন কোন Route হয়ে যাচ্ছে, or device হয়ে যাচ্ছে?
#tracert www.cnn.com → Route, time দেখাবে

Record time: কখন কোন device এর ভিতর দিয়ে উক্ত packet যাচ্ছে

Congestion Control Algorithm (in Network Layer)
→ like traffic jam, যেটাকে data traffic বলে

packet read → neighbour selection → path find →
process করতে কিছুটা সময় লাগে ... it requires some sorts of buffer queue থেকে একটা একটা নিয়ে নিয়ে buffer clear করবে। যদি buffer full হয়ে যায় next আর জায়গা দিতে পারবে না তাই ...
→ যখন 70% performance হয় তখন degrade করতে থাকে, delay হবে, এই সময় এই situation কে congestion বলে। → (বসে থাকতে হয় / দেরী হয়) অনেকক্ষণ wait করতে হয়। → Congestion

Why it is important to control congestion?
— control না করলে deliver করতে পারবে না



(Sercgight effort) + Combined effort (extra মূল্য buffer এর চলে যায়) এরপরে Performance degrade হতে পারে

(rapidly) এর অর্থাৎ fall শুরু হয় তাই control করা important delivery capacity drop হবে, এক সময় তা zero হয়ে যাবে

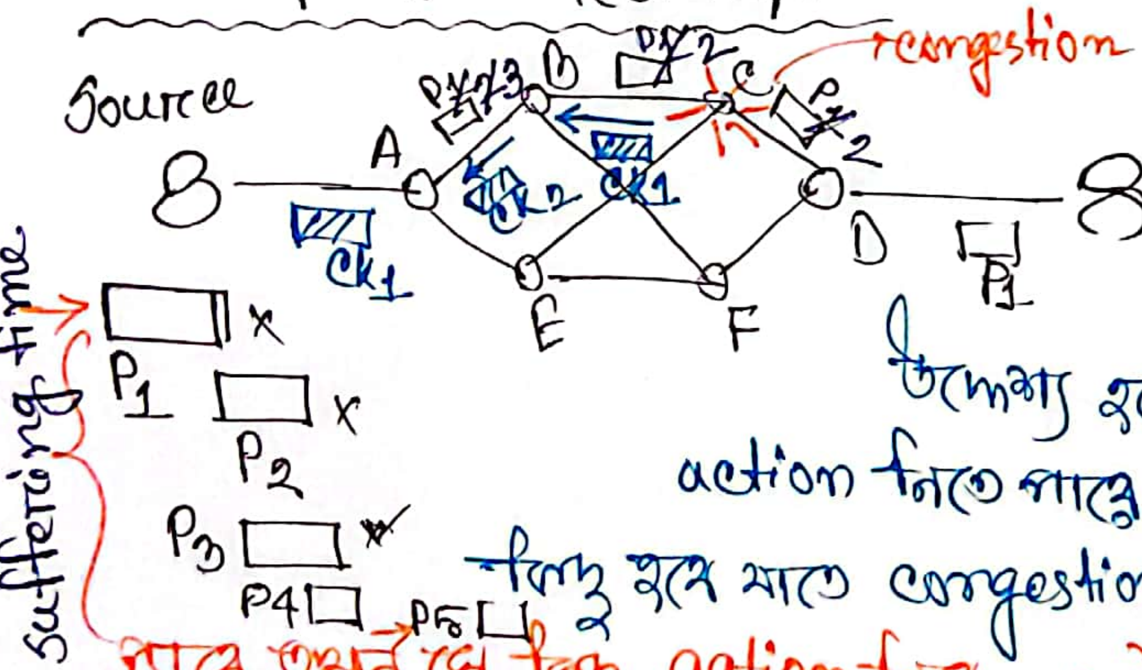
Dead Router

fig: How important to control Congestion

6-05-24

lec-17/ control করতে পারলে আস্তে আস্তে delivery ঠিক হবে। To control congestion algorithm —

- Choke packet Technique:

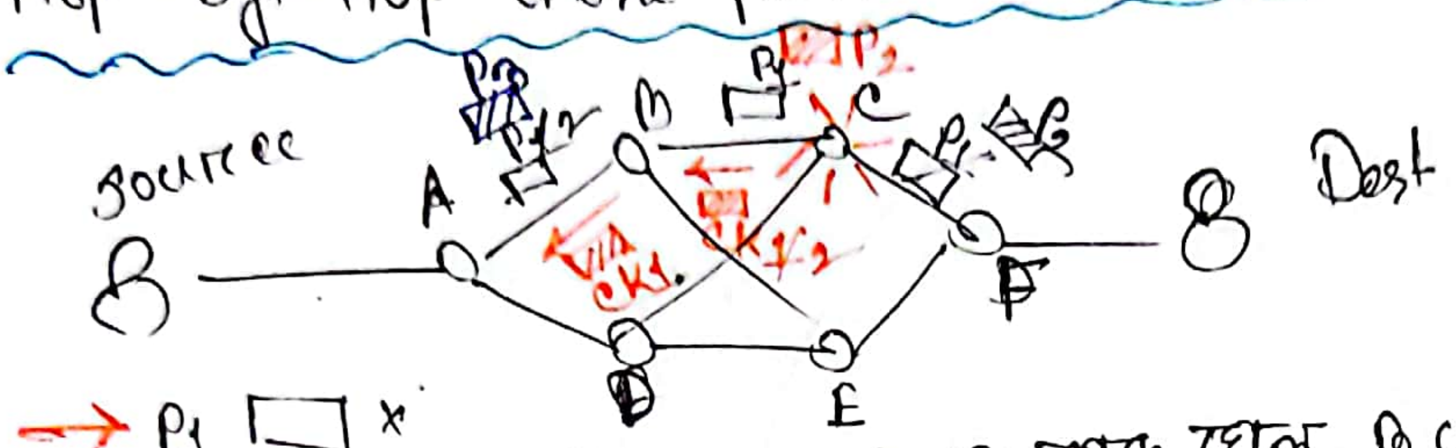


যখন P_1; C তে গেলে congestion feel করা শুরু হল। C এখন source এর জন্য choke packet 1 create করল তাকে যাতে inform করা যাতে source action নিতে পারে। i) path design এর সময় এমন কিছু হবে যাতে congestion feel না হয়। source যখন ck1 পাবে তখন সে কিছু action নিবে → i) path design এর সময় C avoid করা অর্থাৎ C কে relief দেয়া ii) কিছুক্ষণ দেরী করা

আবার P_2 কে D তে forward করার সময় C আবার congestion feel করে তখন আবার ck2 এর generate করবে, ck1 source এর পৌঁছাতে পৌঁছাতে P_5 generate হয়ে গিয়েছে। action নেওয়া হলো P_5 এর সময় মানে P_{1-5} C cross করেই যাচ্ছে; এতক্ষণ sustain করলে ভালো নয়তো router down হয়ে যাবে। trouble শুরু হওয়ার সাথে সাথে signal/remedy পাঠালেও পৌঁছাতে বলে suffering time.

এই trouble সমাধানে রেখে আরেকটা technique আছে। . . .

Hop-by-Hop Choke packet Technique



→ P1 ☐ ×

P2 ☐ ×

P3 ☐

Ck_1 যখন B এর কাছে গেল, B will take some action. to relief C, B will give lower data rate. B কিছুটা wait করে P_2 কে forward করবে C এর কাছে যদি still trouble ... B is helping the adjacent one A এখন lower data rate এ B কে help করবে। এভাবে সবগুলো router ck packet পাবে এবং সবাই একে অন্যকে help করবে with lower data rate...

কিভাবে? → সাথে সাথে data না পাঠিয়ে কিছুক্ষণ wait করবে

তাই choke packet এর চেয়ে Hop-by-Hop more convenient.

উপরোক্ত technique কাজ না করলে তখন আরেকটা algo follow করে।

Load Shedding (to get out of congestion) (demand কে shed করে ফেলা) — এখানে suffering বেশি (cruel technique) — কিন্তু effective

buffer এ কিছু data delete করে দিলে packet lost হবে। — (bad side)

— good side — সাথে সাথে congestion থেকে বের হওয়া যায়

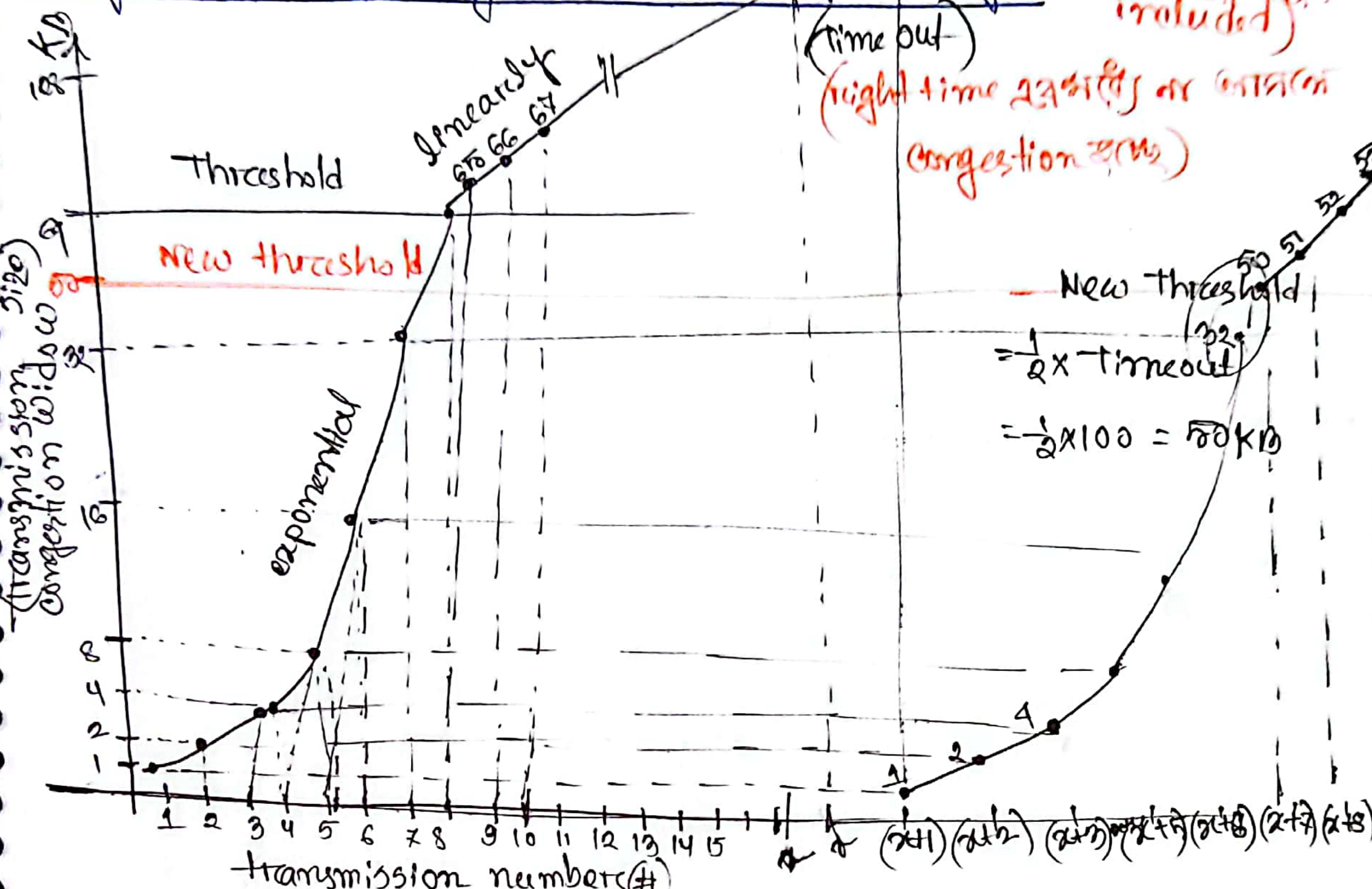
→ Milk Technique (Fresh is Good) — buffer এ আগে যে packet আছে আগেরগুলো delete

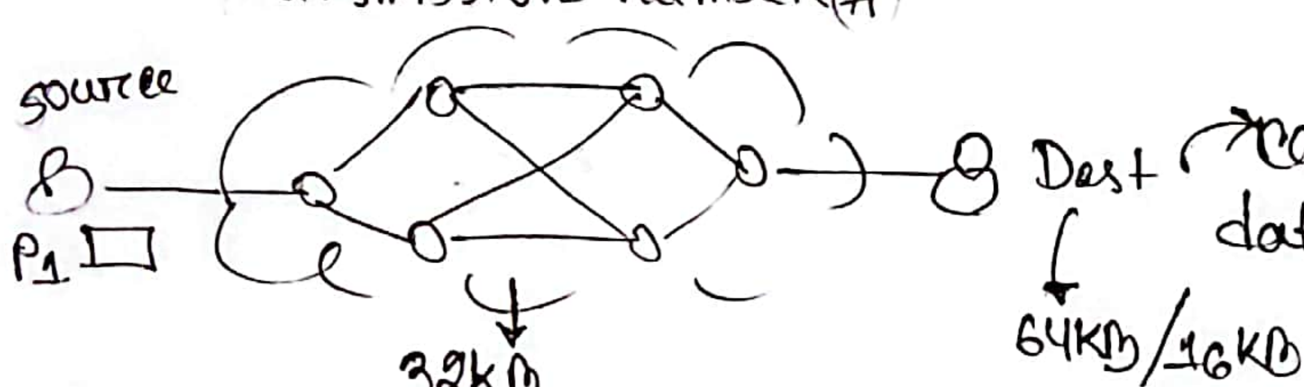
→ Wine Technique (Old is Good) — পুরাতন packet রেখে নতুনগুলো delete

Congestion কে অন্যান্য layer-এ control হয়ে যায়।

Congestion control Algorithm in Transport Layer

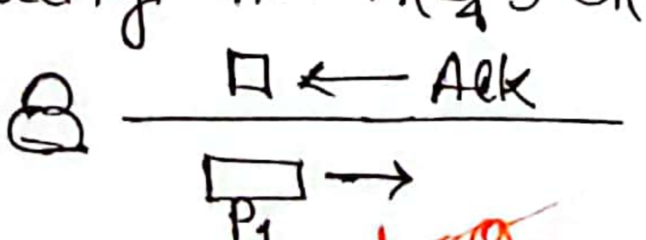
(Source & dest উভয় ক্ষেত্রে included)





Dest → capacity থাকে কি পরিমাণ data accept করতে পারবে

যেটা lower সেটা choose করলে trouble হওয়ার সম্ভাবনা থাকে না। capacity থাকা সত্ত্বেও অনেক সময় অনেক বেশি data rate আসতে পারে।



[source fixed করে দিবে]

P1 যাওয়ার আগেই lost হয়ে গেলে source ack এর জন্য wait করবে। আবার dest ack করার পরেও on the way তে lost হয়ে গেলে ঐ source P2 কে পাঠাতে পারছে না। একটা specific time দেয়া হবে যার মধ্যে ফেরত দিবে যদি return না আসে তাহলে P1 আবার পাঠাবে। (timer use হয়) যদি তাও return না আসে তাহলে সেখানে congestion হচ্ছে।

- 32KB না পাঠিয়ে 1KB পাঠাবে, যদি সমস্যা -trouble দেয় তাহলে 2KB পাঠাবে। এভাবে timeout এর মধ্যে ack আসলে তখন 4KB পাঠাবে। এভাবে

A capacity বাড়ার সাথে সাথে data পাঠানোর পরিমাণ বাড়তে থাকে।

'Slow Start Algo'

- 1KB
- 2KB
- 4KB
- 8KB ↓

- Threshold ——(limit)

Quality of Services (QOS)

(measurement unit)

Importance Matrix

Applications	Bandwidth	Delay	Reliability	Jitter
Email	low	low	high	low
video on demand	high	Medium	low	high
video conferencing	high	high	med/high	high

variations in packets' arrival time. for multimedia communication এর জন্য Important factor গুরুত্বপূর্ণ।

Scenario 1

Source → Dest

delay

P1	10 ms
P2	5 ms
P3	15 ms
P4	20 ms
P5	10 ms

Delay low

Variation high = high Jitter

Scenario-2

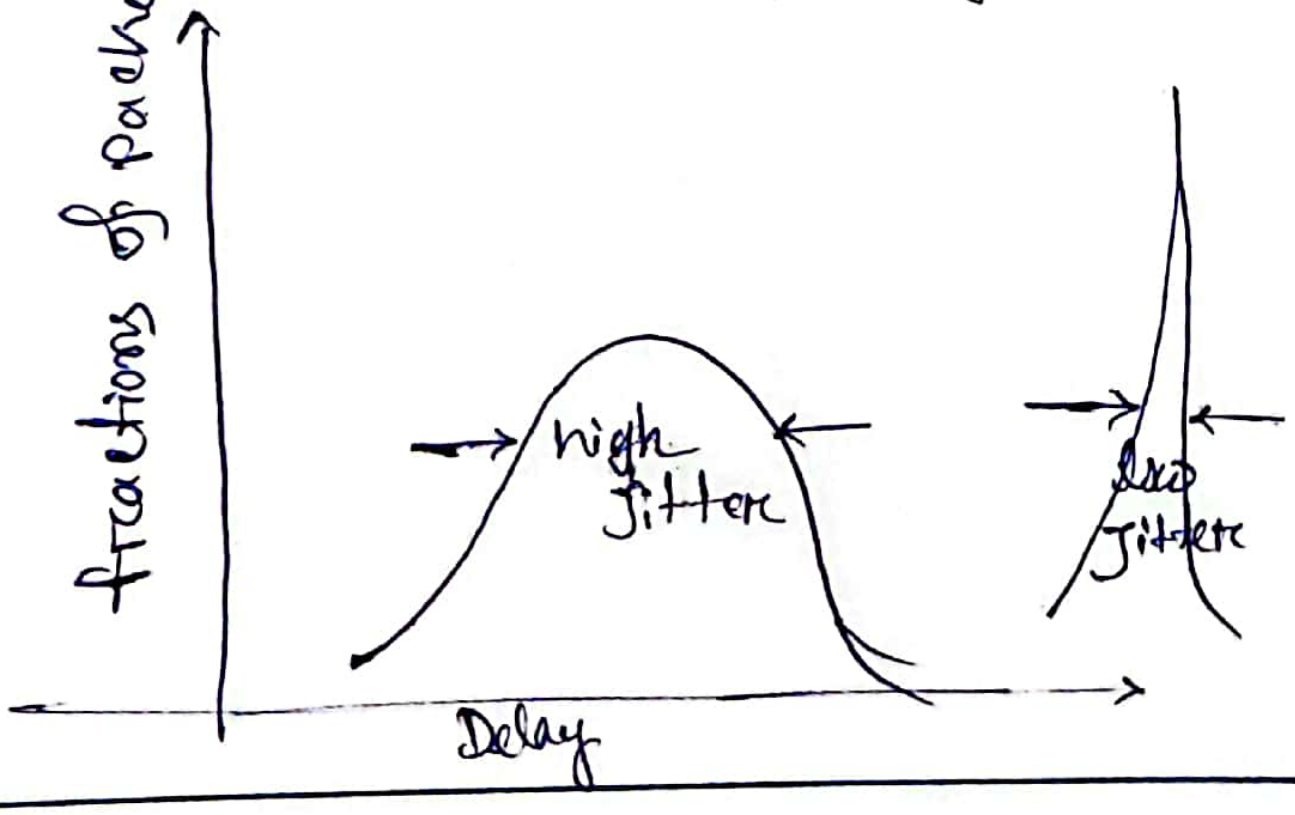
Source → Dest

Delay

P1	90 ms
P2	89 ms
P3	91 ms
P4	90 ms
P5	89 ms

Delay High

Variation low = low Jitter



Lijenta

Quality of services (QOS)

(measurement unit)

Importance Matrix

Applications	Bandwidth	Delay	Reliability	Jitter
Email	low	low	high	low
video on demand	high	Medium	low	high
video conferencing	high	high	med / high	high

Jitter → variations in packets' arrival time. for multimedia communication এর জন্য important factor বলতে এটাকে.

Scenario 1

Source → Dest

delay

P1	10 ms
P2	5 ms
P3	15 ms
P4	20 ms
P5	10 ms

Delay low

Variation high = high Jitter

Scenario-2

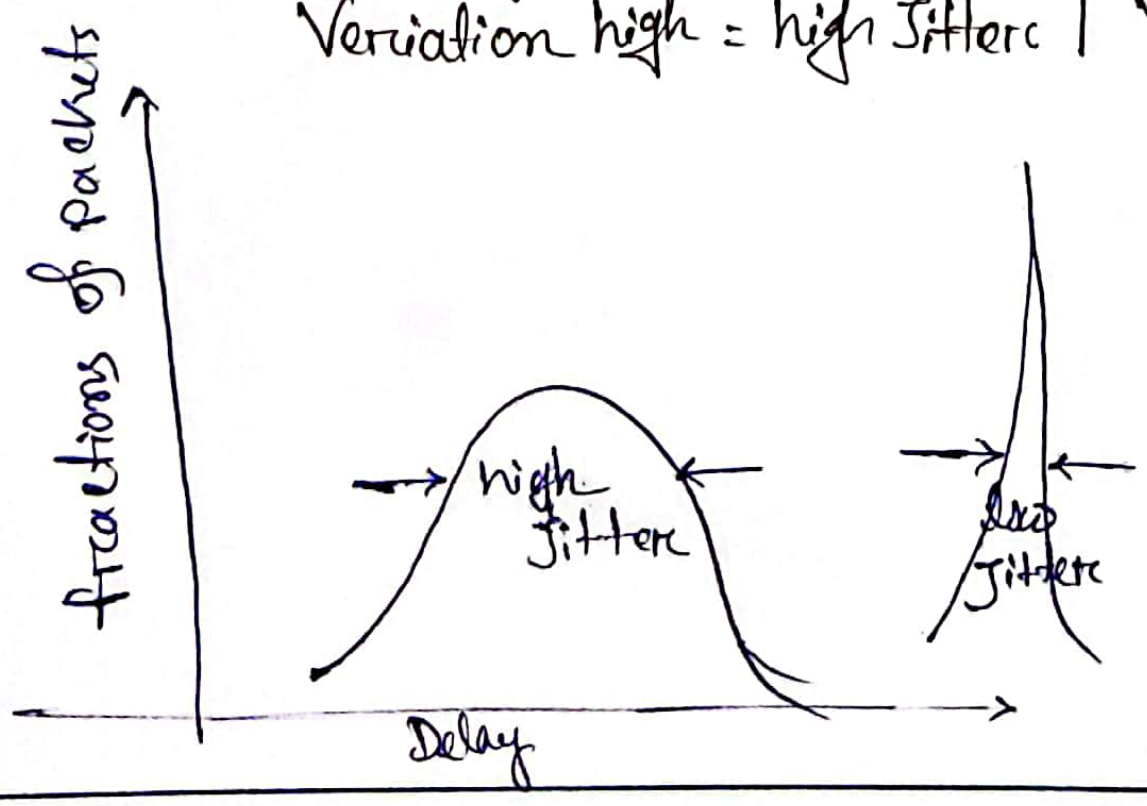
Source → Dest

Delay

P1	90 ms
P2	89 ms
P3	91 ms
P4	90 ms
P5	89 ms

Delay High

Variation low = low Jitter

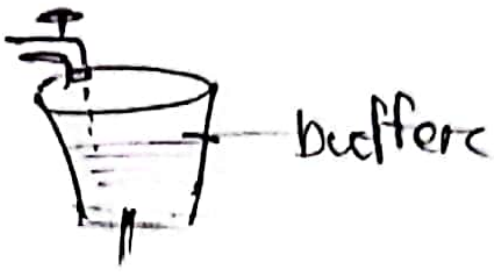


Techniques to achieve QoS

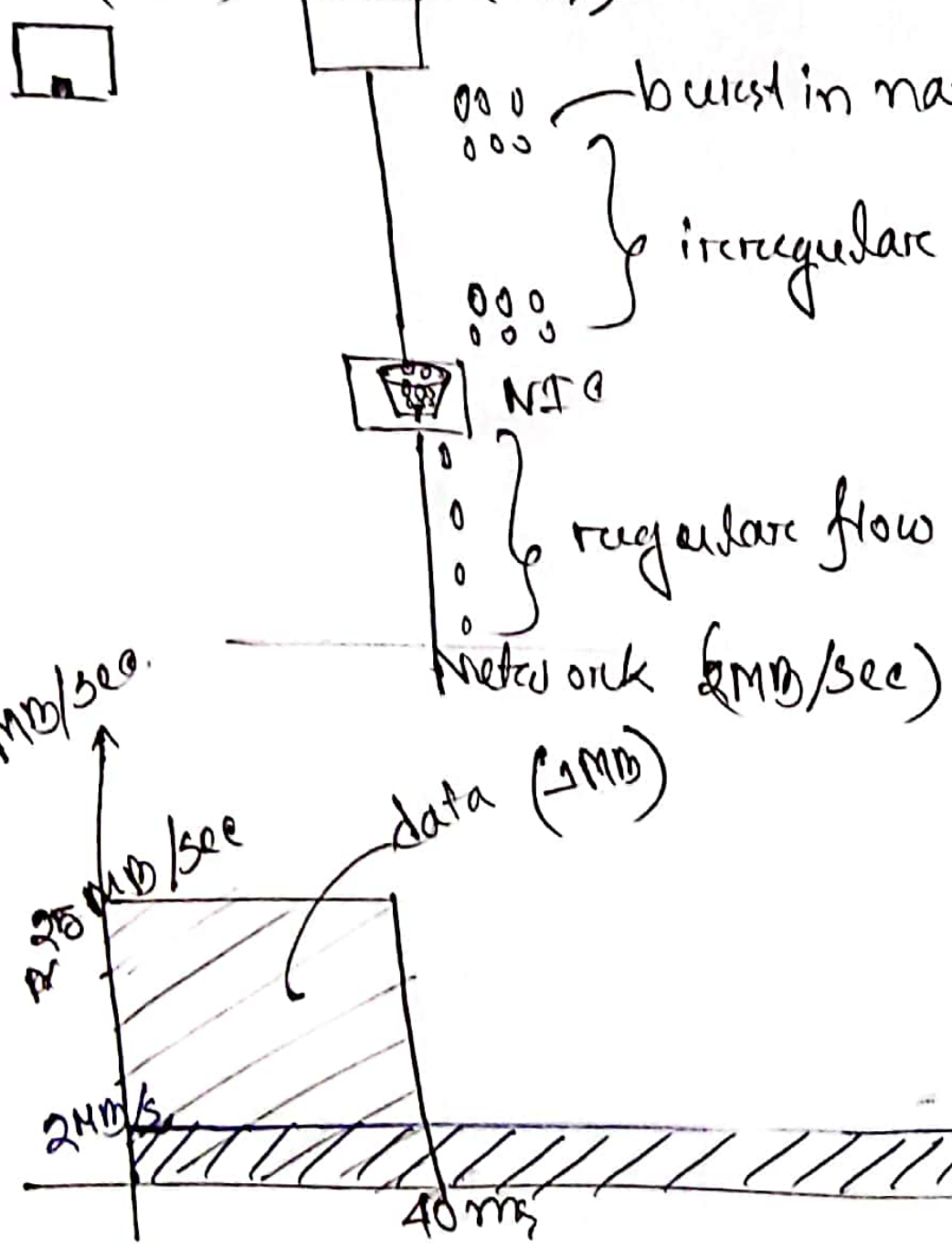
→ Over Provisioning (one of the best)

→ Flow control

* Leaky Bucket Algorithm



PC (Host) PC (Host)



[Network এর capacity অনুযায়ী leak করানোর/ কমানোর মধ্য]

Data = Rate × Time

= 25 MB/sec × 40 msec

= $\frac{25 MB}{1000 msec} \times 40 msec$

= 1 MB

Time = $\frac{Data}{Rate}$

= $\frac{1 MB}{25 MB/sec}$

= 0.5 sec

= 500 msec

(0-1023) →

client 1

http://www.CNN.com

IP

Protocal
(Port #80)

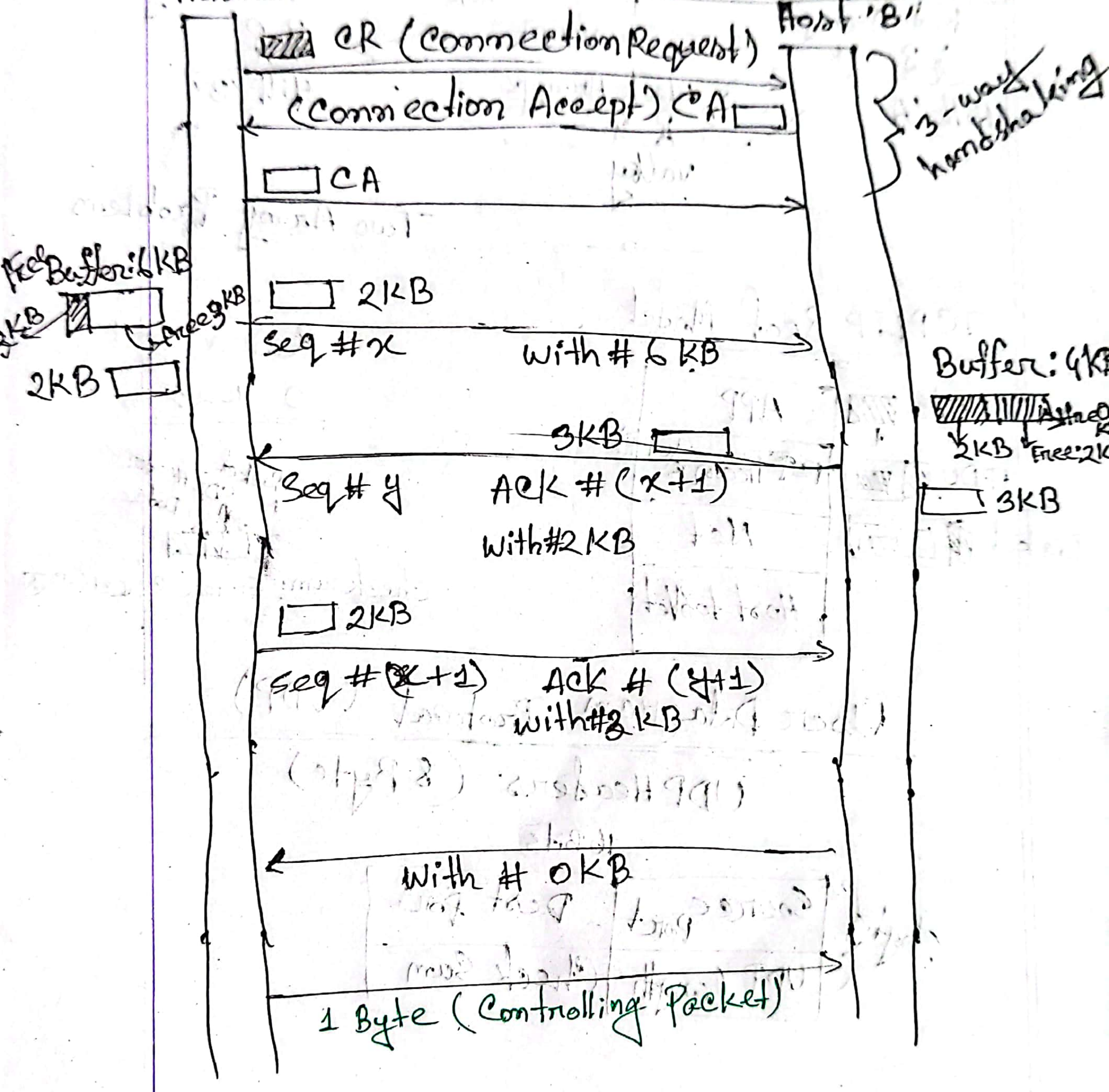
client 2

TCP/IP Ref Model

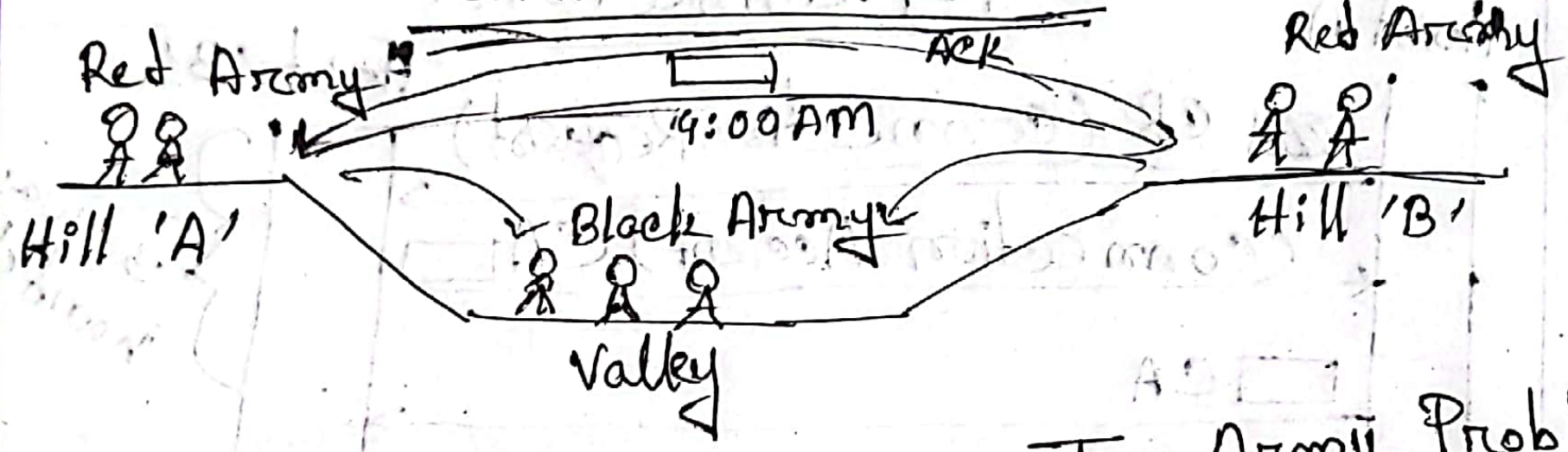
Host (Ex. CNN Web server)

Data Transmission

TCP Connection Establishment +



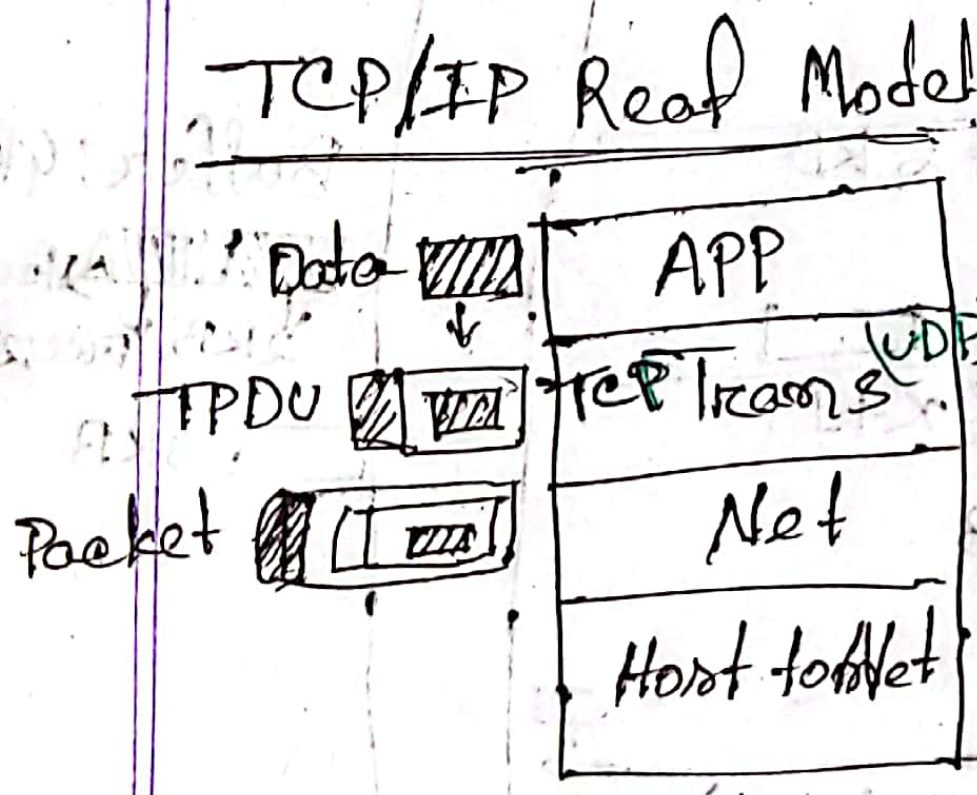
Connection Release



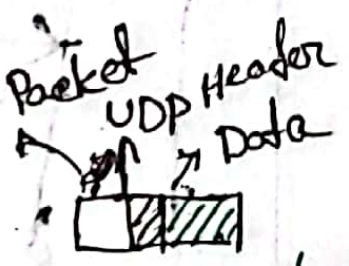
Two Army Problem

—o—

TCP/IP Reaf Model



Reserv port
0 - 1023



checksum: Error check করা

User Datagram Protocal (UDP)

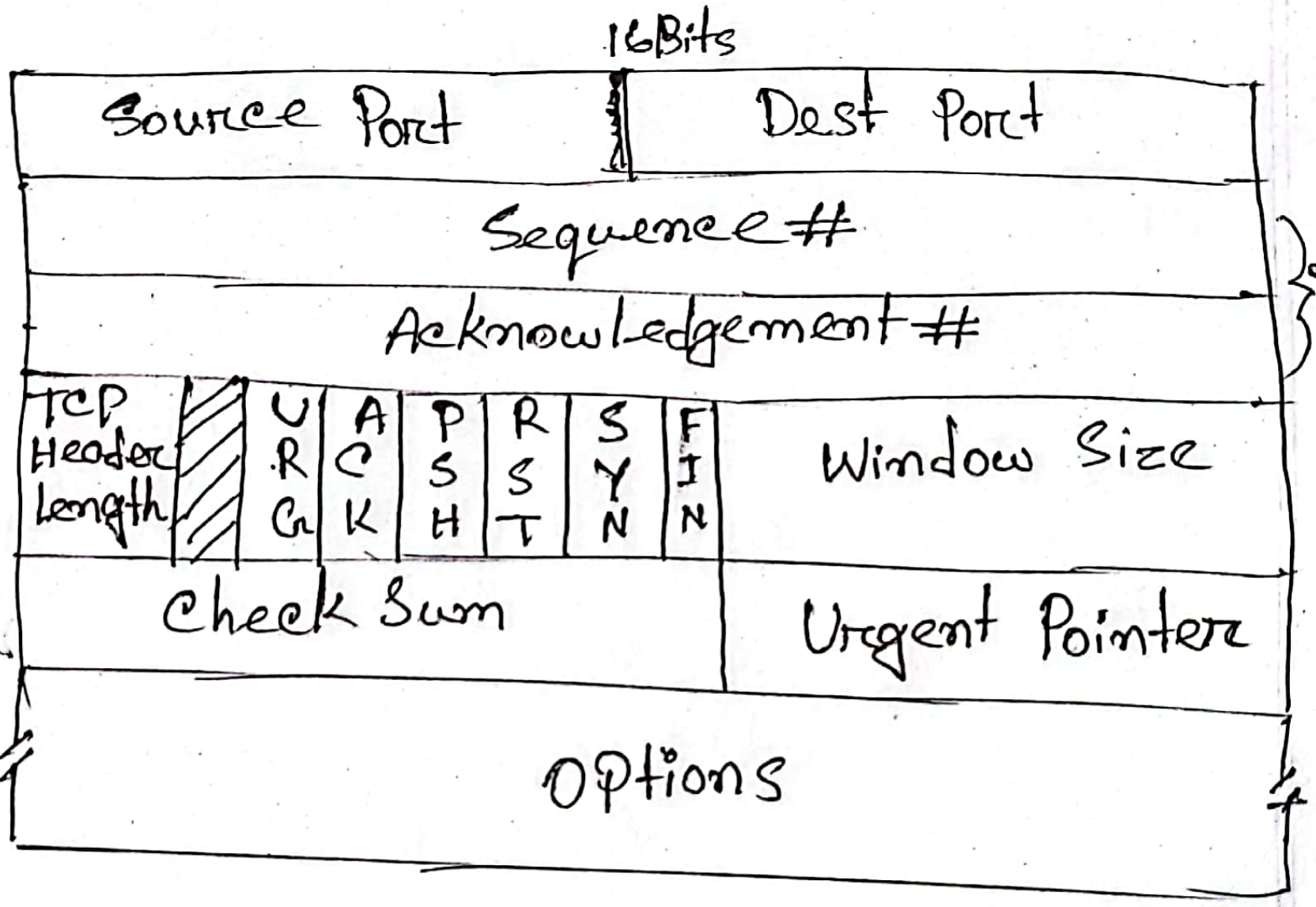
UDP Headers (8 Byte)

16 Bits

Source Port	Dest Port
UDP length	Check Sum

8 byte

TCP Header (20 Bytes)



Window : Buffer time

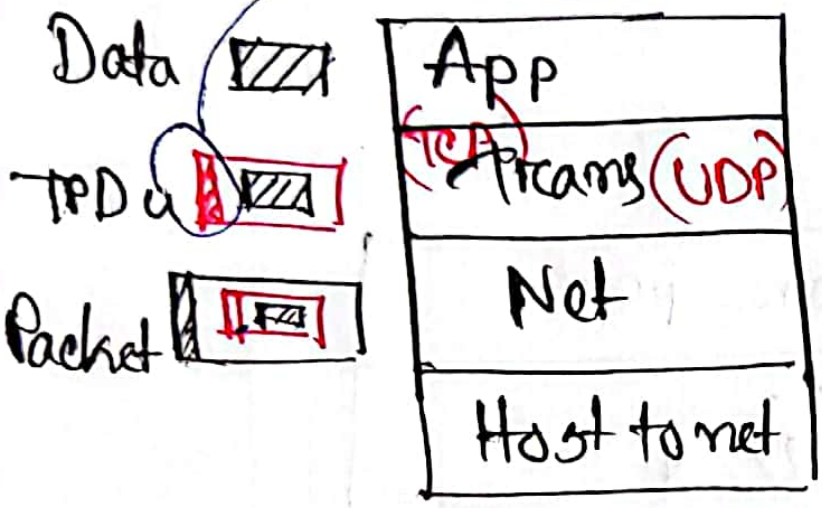
Urgent pointer: কি Type এর urgent

lec-22

20-05-24

TCP/IP Ref Model

Transport protocol Data Unit (TPDu)



User Datagram Protocol (UDP)

lighter- প্রচুর পরিমাণ data pass করতে হয়

UDP follow করলে 8 byte

TCP follow করলে 20 byte, heavy তাই data pass করতে দেরী হয়

UDP Header (8 Byte)

যদি UDP follow করে তাহলে 8byte এর একটা header যুক্ত হবে।

16 Bits

Source port	Destination port
UDP length	Checksum: specific মেসেজ আছে কিনা error check করে (CRC)

8 Byte

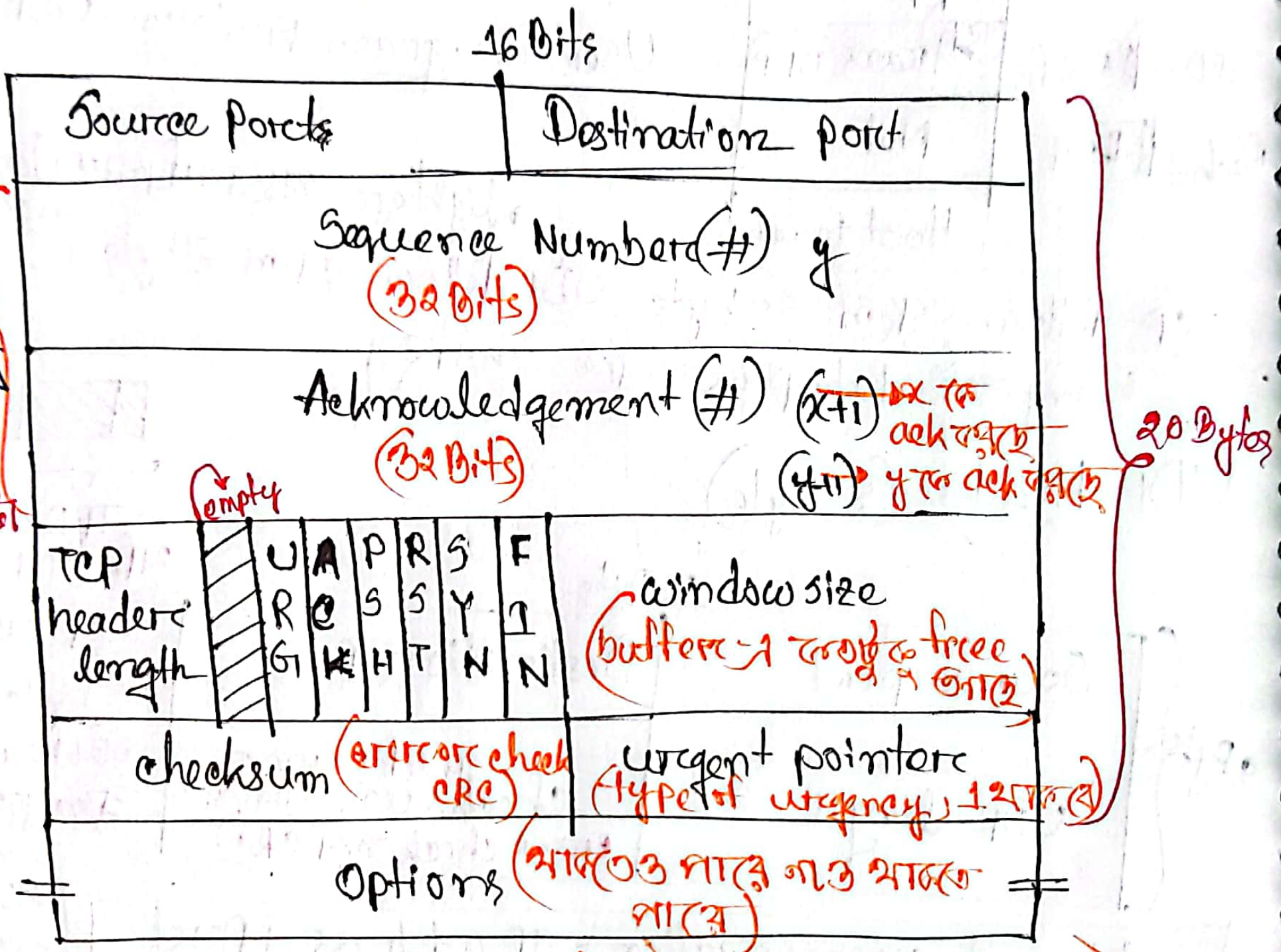
end to end communication = IP address + Port

216 ports আছে

Reserved ports = 0 - 1023
= 2^{10}
= 1024

~~TCP Protocol~~ X

TCP Header (20 Bytes)



Piggy backing এটার মধ্যে আরেকটা পাঠানো

URG = urgent Bit (খুব তাড়াতাড়ি পাঠাতে হলেই করে দিয়ে পাঠানো হয়)
(urgent হলে bit = 1 হবে)

ACK = acknowledgement (bit = 1 হলে ...)

PSH = push (1 = packet কে forward করবে) + variation আসতে

RST = reset (1 = ...)

SYN = synchronization (নতুন connection set)

FIN = finish (1 = inform করবে disconnect হতে যাচ্ছে)

lab

lab exam → 22-05-24 (wednesday)

- 3/4টা ques
- open book, (notes দেখা যাবে)
- Design

TTL কী?

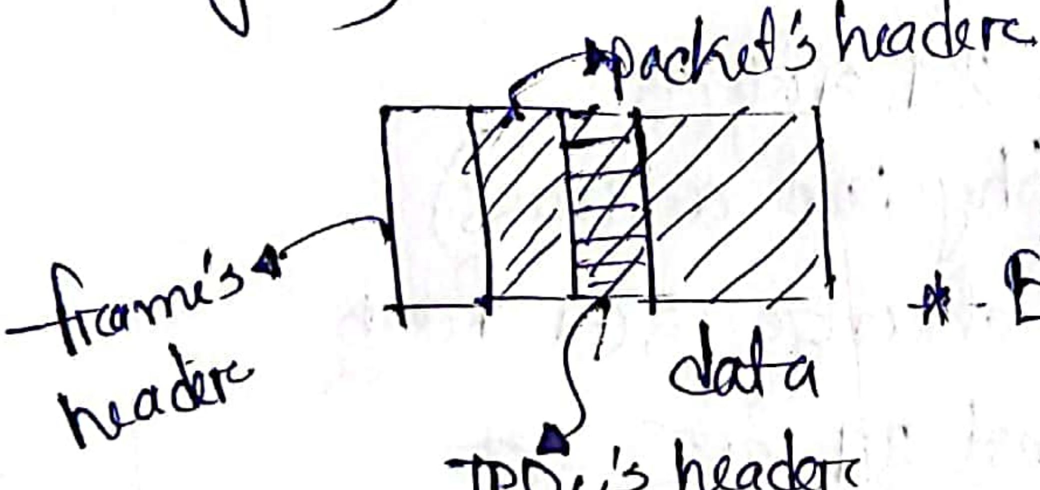
subnet mask?

straight through cable straight?

Wire Shark (Network protocol analyzer) (sniffing tool) → real packet capture করে দেখাতে পারি এবং ভিতর কি কি আছে?

(ARP = Address resolution protocol)

(Ethernet = 1460 Bytes)



* Epoch time = second wise specifically বলছে

Ethernet II → 01-00-5e-00-00-fb (01-00-5e = company, 00-00-fb = serial #)

48 Bits (MAC) — physical address

My PC (MAC) : 40-8D-5C-78-35-7C

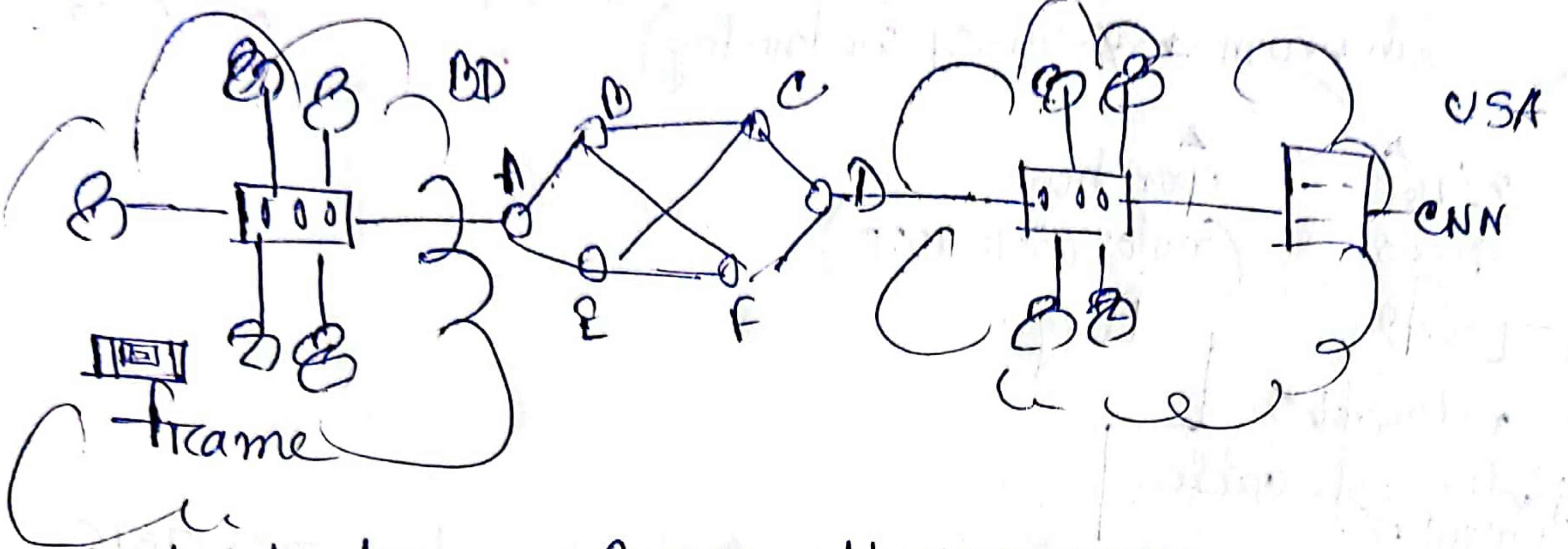
My PC (IPV4): 192.168.15.52

My Gateway: 192.168.15.250

your PC (MAC): 1C-1B-0D-DD-5D-E6

your IP : 192.168.15.30

MAC address দিয়ে source to Dest communicate করে



- packet header-এর মধ্যে MAC add দেওয়া থাকে
- adjacent device এর মধ্যে error free communication পাঠানোর জন্য datalink layer-এর
- পরবর্তী device এ যাবে MAC এর মাধ্যমে (physical address)
- frame এর MAC always change হবে কিন্তু
- packet " শুধুমাত্র Dest এর চেঞ্জ হবে

icmp = internet control massage protocol

IPV4 = 45 — [4 bit দিয়ে 20 byte হয় না তাই 8 bit দিয়ে 45]
IHL = 20 bytes — 32 এর জন্য 5

Total length = 16 bit

(00 3C — 12×1 = 12, 3×16 = 48 → 60 bytes)

- frame = 74 bytes, packet = 60 bytes

identification =
DM = 0
MF = 0
fragment offset (16 byte) : 0
TTL : 128

protocol : ICMP(1)
header checksum
source add : ~~Maruf~~ My 192.168.15.50
dest : Maruf 192.168.15.52

ICMP পাঠাবে না

Data → ৬২ bytes

Distant network-এ পাঠাতে পারলে next adjacent device
হলে হলে CMN-এ যাবে।

lab

FASTEST 4G banglalink

192.168.30.0

Fa0/0

192.168.30.254

R3

192.168.60.0

S0/2/0

192.168.60.2

S0/3/0

(192.168.50.1)

192.168.50.0

192.168.50.1

S0/3/0

192.168.20.254

S0/2/0

192.168.10.254

R1

S0/2/0

(192.168.40.1)

(192.168.40.2)

R2

Fa0/0

Fa0/0

192.168.40.0

192.168.10.1

192.168.10.2

192.168.20.1

192.168.20.2

192.168.10.0

192.168.20.0

(dynamic) 6th network

1) Dynamic Link-State Routing algo

OSPF (open shortest path first)

Process ID

#router △ OSPF △ 1

255.255.255.255

(-) 255.255.255.0 (subnet mask)

0.0.0.255

wild card Mask IP