

CSE 321: Computer Networks

L-1

Text: Computer Networks (4th edition)

- Andrew S. Tanenbaum

Supplementary texts:

(1) Computer Networks: A Systems Approach (4th edn)

- Larry L Peterson, Bruce S. Davie

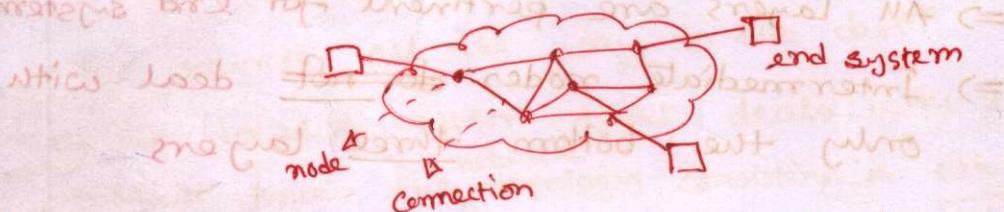
(2) Data and Computer Communications (7th edn)

- William Stallings

(3) Computer Networking: A Top-Down Approach (4th edn)

- James F. Kurose, Keith W. Ross

* Computer Network: A collection of nodes and connections



⇒ Communication over a network is governed by a set of rules and formats : Protocol

→ Protocol is necessary for any function that requires cooperation between different entities.

→ A network that provides many services needs many protocols. Some services are independent, but others depend on each other. Even a protocol may use another protocol in its execution.

→ The form of dependency in between protocols is managed by layering. Here, each layer implements a service

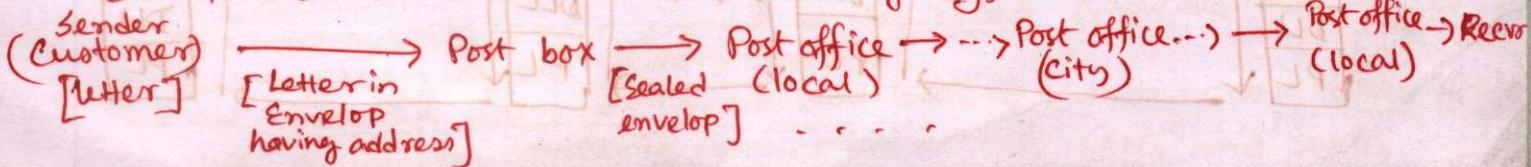
- Through its own internal-layer actions

- and by relying on services provided by the layers below.

→ Advantages of layering:

- Allows identification & relationship of complex system's pieces
- Eases maintenance and updating the system

Example scenario: classical mailing system



Open Protocols and Systems

→ A set of protocols is open if

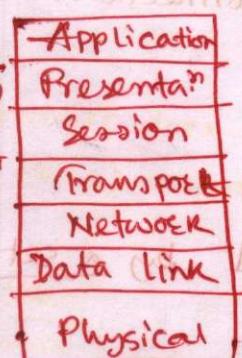
- Protocol details are publicly available
- Changes are managed by an ~~an~~ organization whose membership and transactions are open to the public

→ A system that implements open protocols is called open system

→ International Organization for Standards (ISO) prescribes a standard to connect open systems

- Open System Interconnection (OSI): widely adopted as a reference model for computer networks

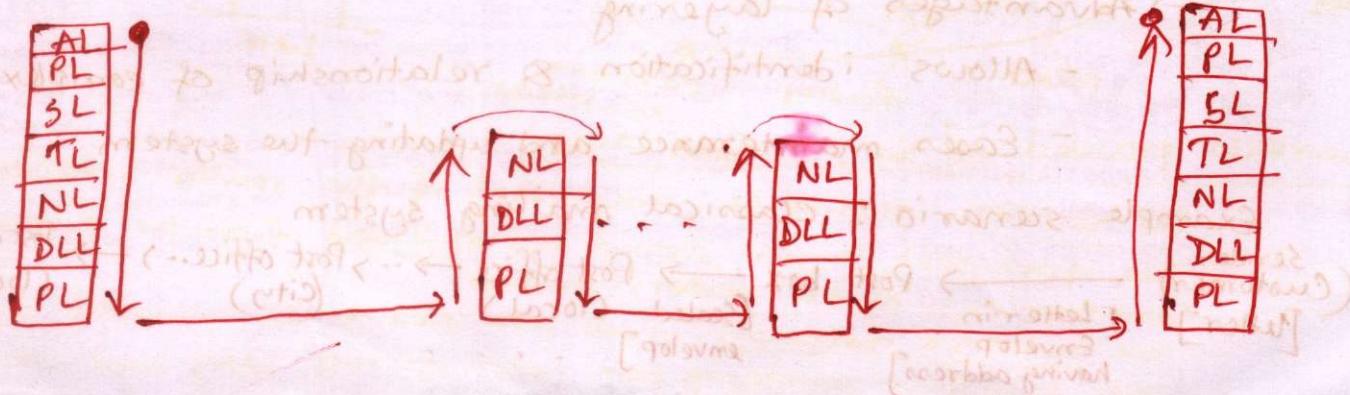
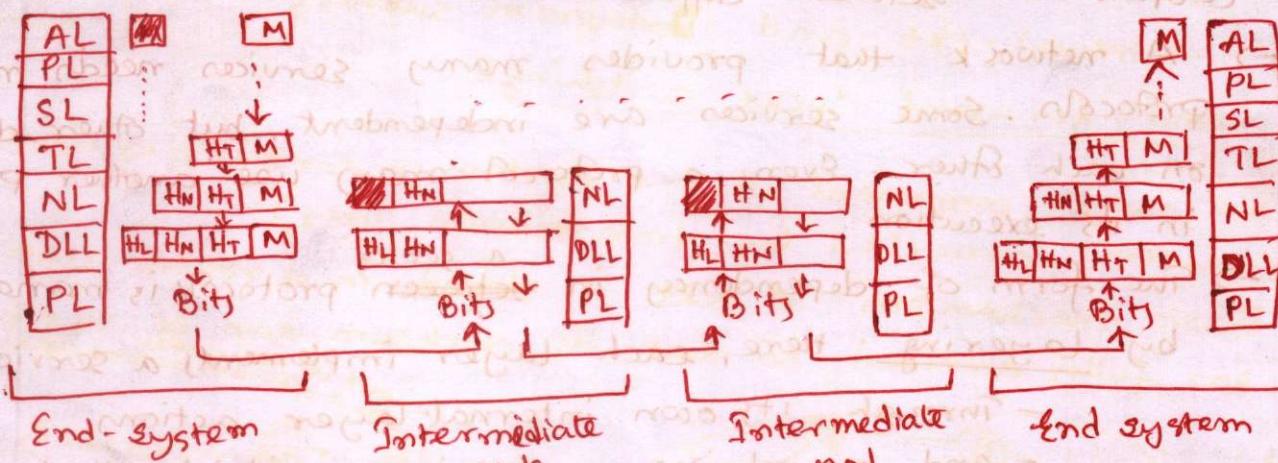
Layers in OSI



Not present
in Internet
protocol
stack

- ⇒ All layers are pertinent for end systems
- ⇒ Intermediate nodes do not deal with only the bottom three layers

Data transmission using OSI reference model



Approach of studying Computer networks

(3)

① Bottom up approach : Tanenbaum
(PL to AL)

② Top down approach : Kurose
(AL to PL)

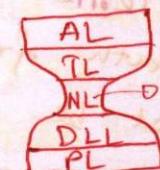
→ Our approach : hybrid of ~~both~~ them !

PL to DLL }
NL to TL } simultaneous

→ by Saifur Sir

Okay me

Chapter 5: Network Layer



Very simple functionality

- Design issues

- Network layer is concerned with getting/routing Packets from the source all the way to the dest?

→ The lowest layer that deals with end-to-end transmission

- Must know the ^{network} topology consisting a set of routers (called subnet) to choose # appropriate paths through it.

→ Design issues for the network layer

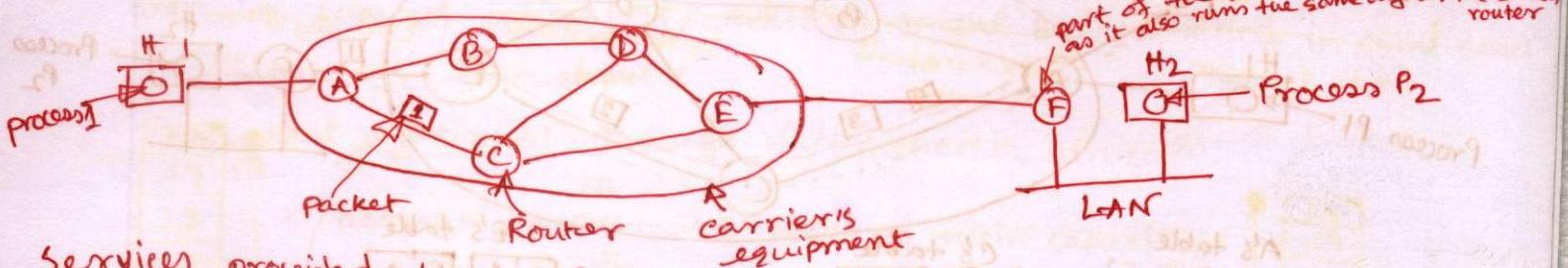
- Store-and-forward pckt switching
- Services provided to the transport layer
- Connectionless and connection-oriented service
- Virtual-ckt and Datagram subnets

Store-and-forward pckt switching

Tasks of the NL

- Know the topology
- Decide on routes to each dest?
- Make routing decisions for each packet accordingly

- The major component of the system is carrier's equipment (routers connected by transmission lines)
- Basic mechanism: A pckt is stored in a router until it has fully arrived so that its checksum can be verified



Services provided to the Transport layer

- The services should be independent of the router technology
- The transport layer should be shielded from the number, type, and topology of the routers present
- The network addresses made available to the transport layer should use a uniform numbering plan, even across LANs and WANs.

Connectionless vs Connection oriented service

① Connectionless service:

- packets are injected into to subnet individually and routed independently of each other.
 - ~~subnet~~ packets are termed as ~~host~~ Datagram
the subnet is a ~~host~~ datagram subnet
- In case of conn

② Connect-oriented service: ~~a path~~

- a path from the source router to the destⁿ router must be established before any data pkts can be sent.
 - the connection is termed as VCC virtual circuit
- the subnet is a virtual-ckt subnet

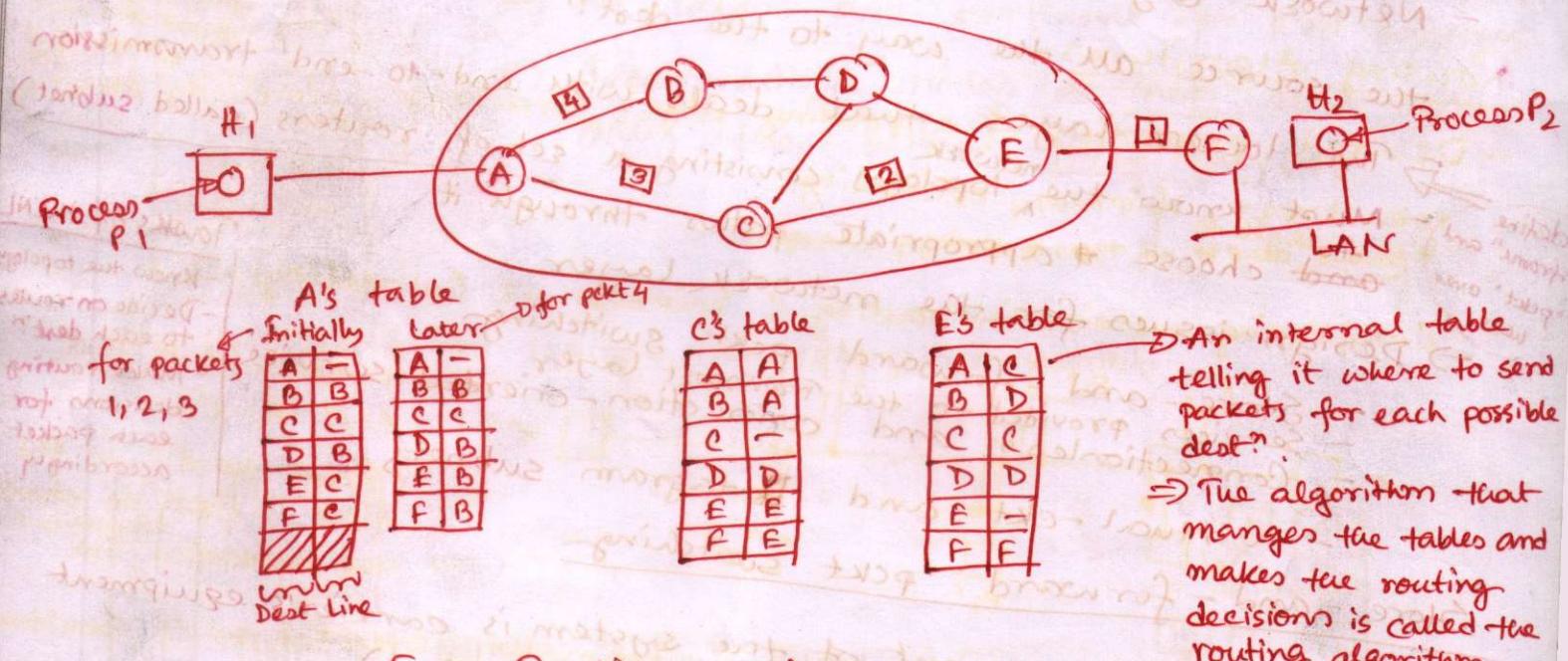


Fig: Routing within a datagram subnet

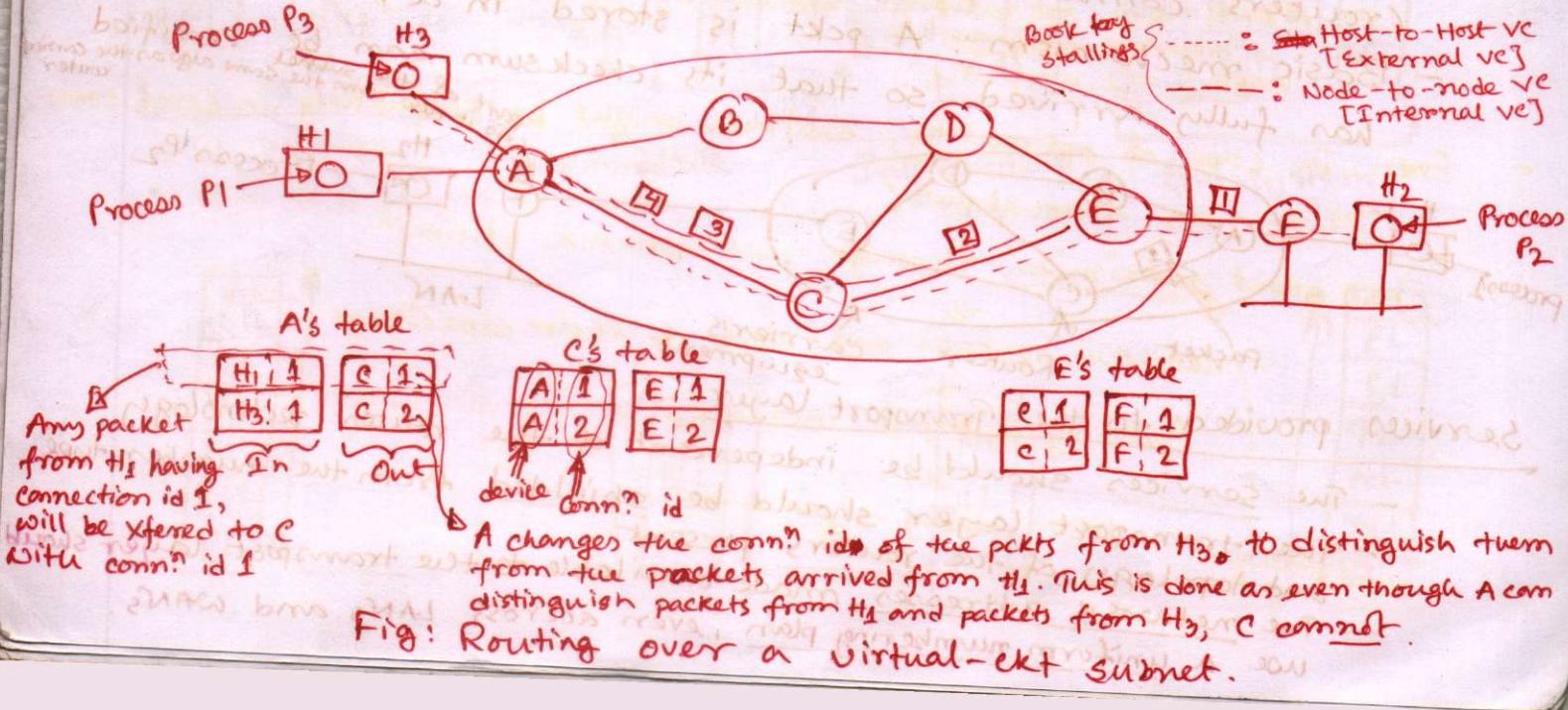


Fig: Routing over a virtual-ckt subnet.

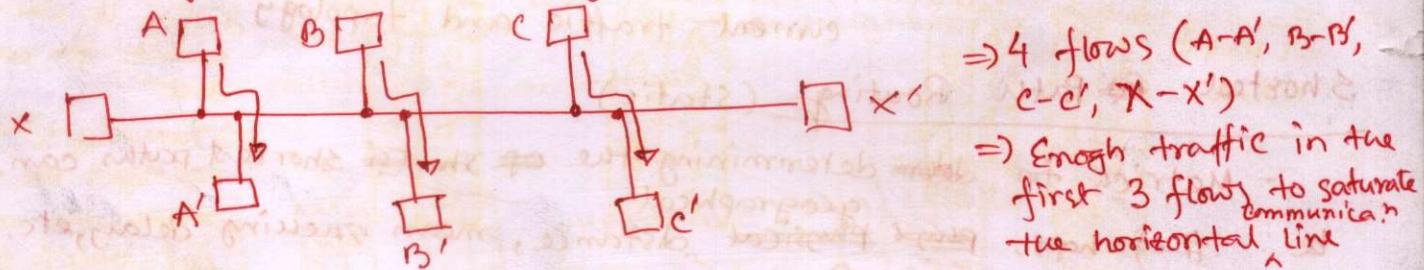
Comparison between virtual-ckt and datagram subnets

(5)

Issue	Datagram subnet	Virtual-ckt subnet
ckt setup req?	No	Yes
Control info in each pktr	Full src and dest ⁿ addresses	A short VC number
Routing of each pktr	Independent	following the route chosen while vc setup
Effect of route failure	None, only the pkts transmitted during the crash will be lost	All VCs that passed through the failed router are terminated
Dos maintenance	Difficult	Easy, if enough resources can be allocated for each VC in advance
Congestion control		

Routing Algorithms:

- Desired properties of a routing algo: Correctness, Simplicity, stability, robustness, fairness, optimality
 (should be able to adapt to all types of topology changes)
- Conflicting metrics in routing: Fairness vs Optimality



→ Consequence: Network is optimally operated as 3 simultaneous flows are operating. However, X-X' may never see any resulting unfairness!
 hope! => unfair!

- Even in case of optimality, metrics might be conflicting in some cases.
 Delay vs throughput:

Throughput maximization through full network utilization

network queues are close to their capacities

Queuing delay is increased

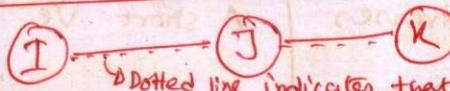
Total delay is increased

Conflicting

→ However, in some cases delay and throughput might not be conflicting. Ex: # of hops ↓ (delay ↓ BW Consumpt.) Throughput

Optimality principle:

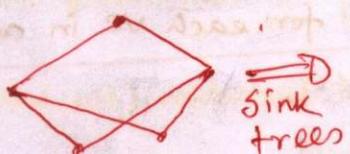
Consistent



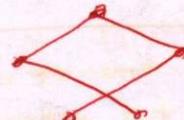
optimal implies $I \rightarrow J \rightarrow K$ to be optimal.

→ If router J is on the optimal path from router I to router K, then the optimal path from J to K also falls along the same route.

→ A set of optimal routes from all sources from to a given destination form a tree rooted at the destination: Sink tree (may not unique)



Sink trees



Routing algs

Adaptive: Dynamic routing decisions based on estimates on current traffic & topology

Static/Non-adaptive: Does not base their ~~decis~~ routing decisions on measurements or estimates of the current traffic and topology

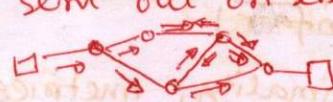
Shortest Path Routing (static)

- Metrics for determining the shortest paths can be: # of hops, geographical distance, mean queuing delay, etc.

- mechanism: Several algorithms (ex: Dijkstra)

Flooding (static)

- Every incoming packet is sent out on every outgoing line except that one it arrived on.



- Disadvantage: Generates vast numbers of duplicate packets. (In fact infinite # ^{in case of taking no measure})

- Avoidance:

① Use a ~~header~~ hop count in the header. The header is decremented in each hop. A packet is discarded when the counter becomes zero.

② Source router puts a seq # in each packet it receives from its hosts. A list of seq # for each source router is maintained in each router. An incoming pkkt is discarded if it is not flooded for the first time. To prevent the list from growing without bound, each list is augmented by a counter K, which summarizes the list meaning all seqs up to K have been seen.

⇒ Selective flooding (a variation): Incoming pkts are sent only on those lines that are going approximately in the right direction.

APP's of flooding:

(7)

- ① Military: for extreme robustness
- ② Distributed DB: To update data concurrently
- ③ Wireless networks: All messages transmitted by a stat can be received by all stats
- ④ Metric or Benchmark for other algos (as shortest path must be travelled in this case)

Distance Vector Routing (DVR)

- Each router contains a routing table
- Each routing table contains one entry for each router in the subnet
- Each entry contains two info:

① Preferred outgoing line/next-hop node for corresponding destination routers

② Estimate of the time or distance to that destination (the best-known distance)

- Different metrics used for estimating distance:

- # of hops (=1 for a neighbor)

- Time delay (by sending ECHO pkts, which initiates timestamping and sending back from the neighbors)

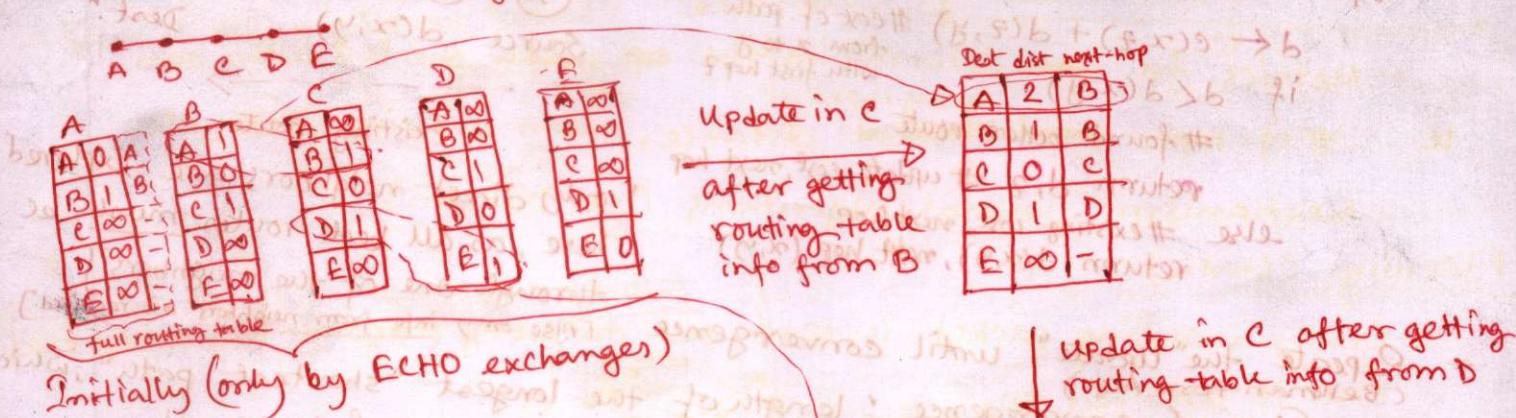
- Queue length (by simply examining queues of other nodes)

- Algo: Each router sends routing info to neighbors to update its own table

① The best known distance: Each router exchanges its own routing table info with its neighbors.

Only the distance part (excluding the preferred next-hop) suffices

② After receiving an info from a neighbor, a router updates its own routing table accordingly



	A	B
A	2	B
B	1	B
C	0	C
D	1	D
E	00	-

update in C after getting routing-table info from D

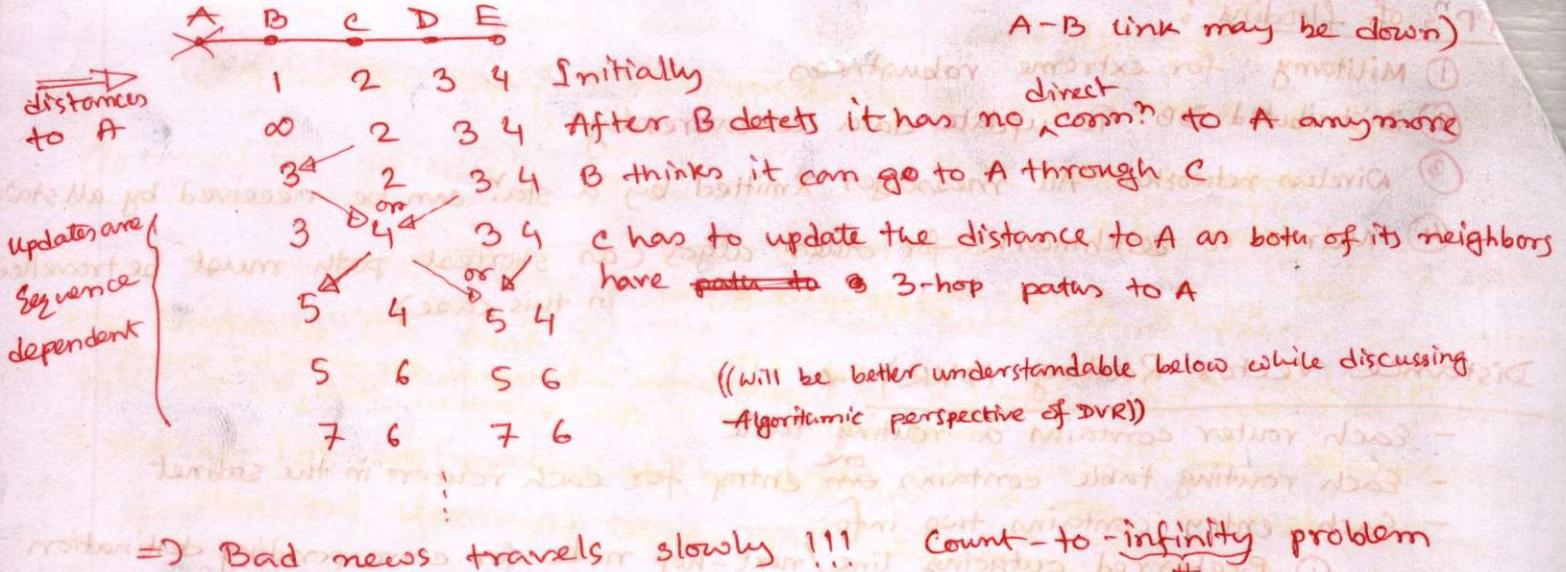
	A	B
A	2	B
B	1	B
C	0	C
D	1	D
E	2	D

similar updating continues in each router

⇒ Good news is spreading at the rate of one hop per exchange
 Time of convergence { - N hop length subnet → Everyone will know good news in N exchange (presence of a router in subnet),
 & diameter of the graph

Ques: What about bad news?

Let, A fails in the linear topology (A may be down or A-B link may be down)



\Rightarrow Bad news travels slowly!!! Count-to-infinity problem

Can be set as (longest path + 1)

Reason: Nob idea about whether I am the next-hop node in the neighbors path (only knows its own next-hop)

\Rightarrow Elaboration & soln from algorithmic perspective [book from Patterson]

Algorithmic perspective of DVR (from the book of Patterson)

- Initially routing table in each node will have finite distance for only directly-connected nodes

- distances might be # of hops or anything else that acts as weights in a graph

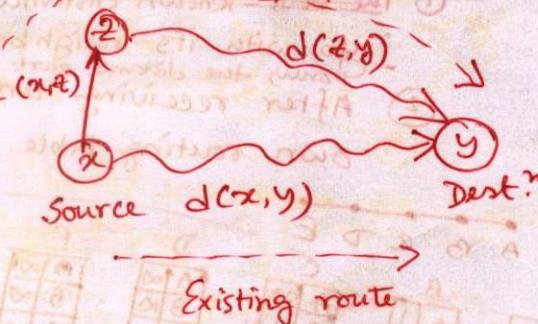
- Algo for update

Update (x, y, z)

```

 $d \leftarrow c(x, z) + d(z, y)$  #cost of path
if  $d < d(x, y)$                          from x to y
                                         with first hop z
                                         #found better route
                                         return  $d, z$  #update cost, next hop
                                         else #existing cost, next hop
                                         return  $d(x, y)$ , next hop  $(x, y)$ 

```



[only direct neighbors are explored here, as all best routes must be through one of the neighbors]

- Repeat the "Update" until convergence (Bellman-Ford Algo)

- Time of convergence: length of the longest "shortest path", which is termed as diameter of the graph

c) Impact of node failure:



\Rightarrow ~~1 2 3 4 5 6~~

in both F and D's table

B's table arrives
at D and F

\Rightarrow ~~1 2 3 4 5 6~~

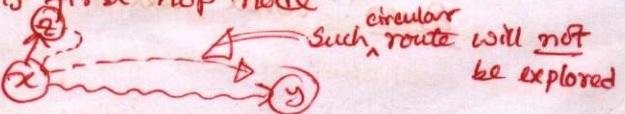
Updated entry in D's table

\Rightarrow ~~1 2 3 4 5 6~~

Updated entry in F's table

(WRONG!!)

\rightarrow Soln: While updating, a node has to make sure that it is not the next hop on the path from its first-hop node



Update (x, y, z)

$$d \leftarrow c(x, z) + d(z, y)$$

if $d < d(x, y)$ & $x \neq \text{nexthop}(z, y)$ } "split Horizon Rule"
 #found better route
 return d, z
 else
 return $d(x, y), \text{nexthop}(x, y)$

→ Now fully OK? NO, as the entry in D's table still mistakenly gets updated. Reason: B's table entry to C still remains stale as the algo only decreases the cost/distance, however, does not increase it.

→ Soln: Always accommodate the update from next-hop node

Update (x, y, z)

$$d \leftarrow c(x, z) + d(z, y)$$

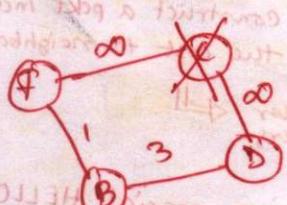
if $\text{nexthop}(x, y) = z \mid (d < d(x, y) \& x \neq \text{nexthop}(z, y))$

#forced update

return d, z

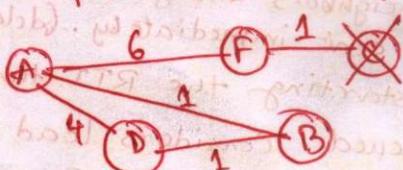
else

return $d(x, y), \text{nexthop}(x, y)$



c 5 B	\Rightarrow	c 0 -	\Rightarrow	c 0 -
in D		in B	in D	OK

→ Ques: All problems solved? Let's see... to what if there is a link failure?



c 7 F	c 8 A	c 9 B	c 1 C
-	-	-	c 0 -

c 0 -	-	-	c 0 -
-	c 0 -	-	-
-	-	c 0 -	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

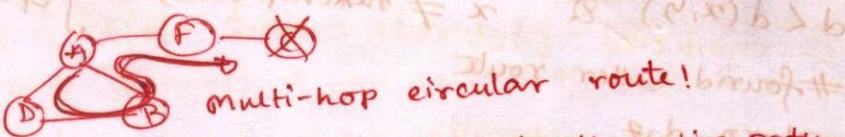
c 0 -	-	-
-	-	-

c 0 -	-	-
-	-	-

~~c	0	-~~	-	-

<tbl_r cells="3" ix="2" maxcspan="1" maxrspan="1

⇒ Reason behind sequence-dependent convergence: Only next-hop is known, entire path is NOT known
 "A" adopts the path $D \rightarrow B \rightarrow A \rightarrow \dots$ in which A was an intermediate node



→ Possible soln: Track down and distribute the entire path

⇒ Problem: lot more info needs to be xmtted

→ Other solns: Ensure better sequence in update

- Still depends on the sequence, however, the extent of dependency is reduced.
- ① Propagate bad news faster! Triggered update for receiving
 - ② cost and periodic update otherwise (increases prob of convergence)
 - ③ make infinity smaller finite # (ex: 100)

Link State Routing (LSR)

- Basic difference with DUR:

DUR: Sends info of all nodes to neighbors

LSR: Sends info of neighbors to all nodes

- Five operational steps in LSR:

① Discover neighbors ⇒ Measure cost to each neighbor ⇒ construct a pkt incorporating the cost to neighbors

① Discover neighbors:

⇒ Identification of neighbors through exchanging special HELLO packets

② Measuring Line cost? The ECHO pkt is sent immediately. ($\text{delay} = \frac{\text{RTT}}{2}$)

⇒ Two different techniques for starting the RTT timer.

(1) When the ECHO pkt is queued: Considers load balancing

(2) When the ECHO pkt arrives in front of Q: Does NOT consider load balancing

load balancing:

⇒ A-B, C-D are of same BW,

however, experience different traffic loads ($A-B > C-D$ for ex.).

Then, the link with less traffic (C-D) will be preferred for balancing traffic load.



⇒ Problem of considering LB: Oscillation

$(A-B > C-D) \Rightarrow$ choose C-D $\xrightarrow{\text{result}} (C-D > A-B) \Rightarrow$ choose A-B in new update

→ Soln: fraction of traffic is split over multiple lines in fractions rather choosing only a single line (such split inhibits drastic change in traffic load over a link just after one update)

A-B	C-D
.5	.5
.4	.6
new traffic cannot be removed (must be .5, .5 again) (it disrupts balance in prev. case)	

Project :

after 10 classes

- Simulation of different types of networks in ns-2 with already-available & modified protocols and produce ^{values of} different metrics

Different types of networks

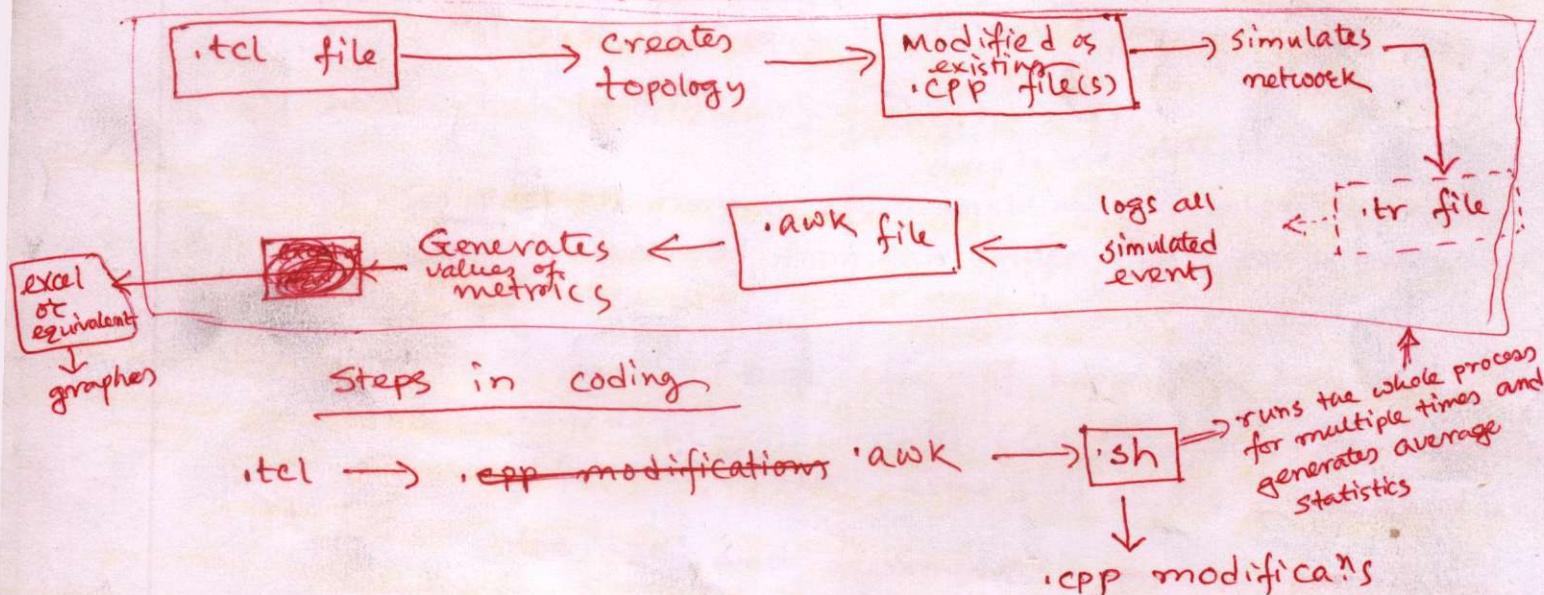
Wired, ^{optical} Wireless, Mobile, Sensor, Satellite
(static)

=> Combination of at least two types as mentioned above

(only exception: Satellite)
Different names => might be in flat or hierarchical topology
Backbone networks, ad-hoc networks, mesh networks,
mobile ad-hoc networks, vehicular ad-hoc networks, ...
⇒ 15 combinations

ns-2

Steps in coding of execution



'sh'

define network attr → vary # of nodes, data rate (pps, payload size, ...), speed, protocol, etc.

loop start

change attr

run .tcl and simulate network with changed attr,

run .awk and generate metric values of the simulation run

extract averaged value of the generated metrics

end loop

write averaged value to a file

protocols:

MAC or NL or TL [or even PL models!] ← change any of them with some intuitive explanations

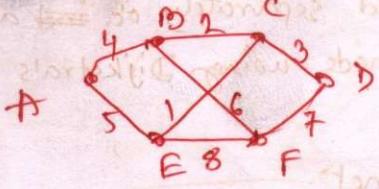
Metrics : Network throughput, end-to-end delay, jitter, per-node throughput, energy consumption, ...

=> Any brand new thinking will get BONUS !!!

3) Constructing Link State Packets

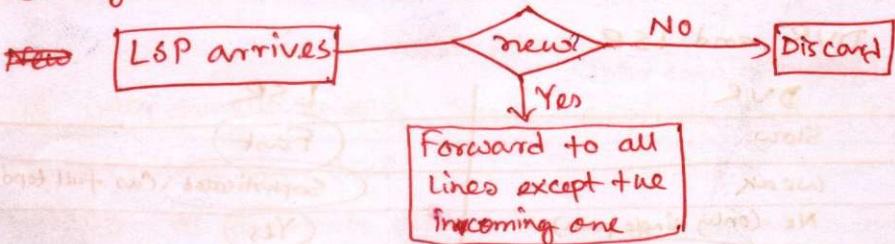
(11)

- LSP contains: own identity, a seq# (decreases during each sending), age (decreased in each sec), list of neighbors



A	B
seq	age
8	6
7	5
6	4
5	3

4) Distributing LSPs (Trickiest part of the algo)



=> Problems with the algo (mainly with seq#):

- ① Wrapping up of the seq#: (0, 1, ..., max#, 0, 1, ...) seems to be obsolete
 - 32-bit seq#, 1 LSP per sec \Rightarrow 137 years to wrap around [so, wrap around may happen]

② Router crash: (0, 1, ..., n) crash (0, 1, ...) seems to be obsolete

- (\Rightarrow after crash - seq# starts from 0 after enabling a crashed router. So, all LSPs generated from this router (up to prevly sent max#) will be discarded.)

- ③ Corrupted seq#: (huge effect if happens in MSB side)

- Suppose 16th bit gets corrupted (0 to 1) and 65,540 is received instead of 4. All LSPs with seq# 5 to 65,540 will be rejected in this case.

Sol.: Include "age", and decrement it once per sec, by each router when the "age" hits zero, info from that router is discarded. [So, if any of the prev. problem would happen, we will end up with only a few wrong discarding events (up to the time age remains > 0)]

"Age" is also decremented during initial flooding process (to reduce the extent of flooding)

=> Increase robustness of the algo:

- An LSP, is not immediately queued for transmission. Instead, it is first put in a holding area to wait a short while.
- The info in holding area helps in discarding duplicate LSPs, determining where to acknowledge (each LSP is acked) and where to send (excluding the incoming line). The holding area is periodically scanned in round-robin order to select a pkt orack to send.

Source	Seq#	Age	Holding area in B		
			A	C	F
A	21	60	0	1	1
F	21	60	1	1	0
E	21	59	0	1	0
C	20	60	1	0	1
D	21	59	1	0	0

Holding area in B

\Rightarrow Arrived directly from A; so, ack to A send to rest

\Rightarrow u u u u F; so, ack to F send to rest

\Rightarrow If a copy of C arrived through F before performing corresponding forwarding, this entry will become 100 011

→ If we would work only with SEND flag (assuming ACK flag as its complement) then we will have trouble at the beginning when all SEND & ACK flags should be 0 (Not complement).

- ⑤ Compute shortest path to every other routers
- Constructs entire subnet graph after accumulating all LSPs. Each link is represented twice, which can be used separately or averaged.
 - Shortest path determination in each node using Dijkstra's algo

⇒ Problem with the algo

→ Computational time in solving large subnet.

→ Memory requirement (~~# of router * # of neighbors~~)

⇒ Comparison between DVR and LSR

Metric Attribute	DVR	LSR	Or better
Convergence rate	Slow	Fast	
Metric	Weak	Sophisticated (Can full topo is available)	
Incorporate multiple paths	No (only single path)	Yes	
Complexity	Simple	Complex & source of bugs	
Distributivity	Full	Partial	

⇒ Examples of LSR: OSPF (Open Shortest Path First), IS-IS (Intermediate System - Intermediate System).

OSPF :

→ Open standard ; SPF : Another name of Dijkstra's algo

⇒ Flooding of LSA (Link State Advertisement)

→ Received seq # vs stored seq # (in case of receiving an old LSA)

> : Update & propagate to all other lines

< : Send back to sender telling notifying it about its stale info (↔)

⇒ When flooding is performed? : Periodically & triggered (when changes happen)

⇒ What happens when Router goes down & back up?

- Seq # set to 0 ; Router will send LSAs with number 0

→ Will get LSAs with last time sent max #

- Router sets the seq # to (max # + 1) and resends

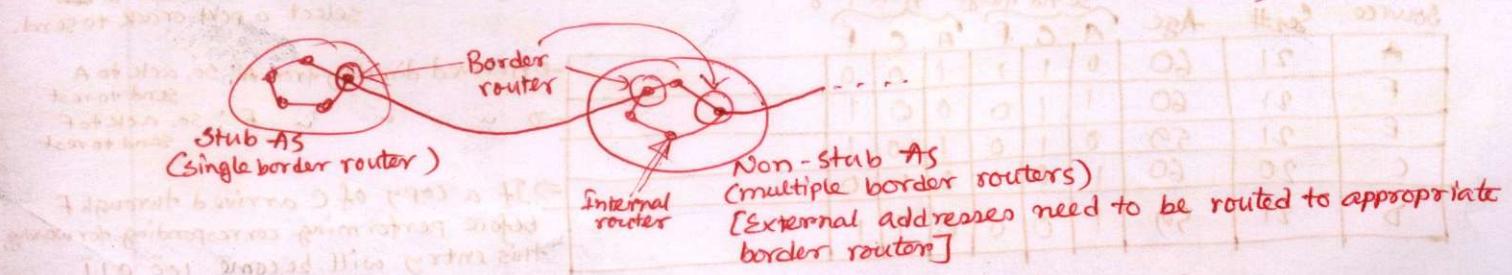
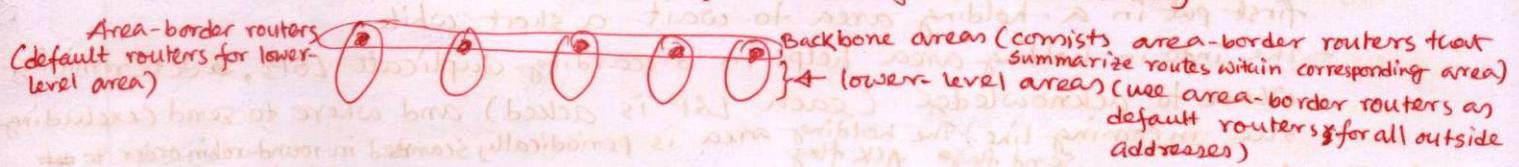
⇒ Seq #, age, load balancing → Similar as discussed above

⇒ Types of Service (TOS) metrics : Link characteristic varies in multiple dimensions such as latency, throughput, cost, reliability

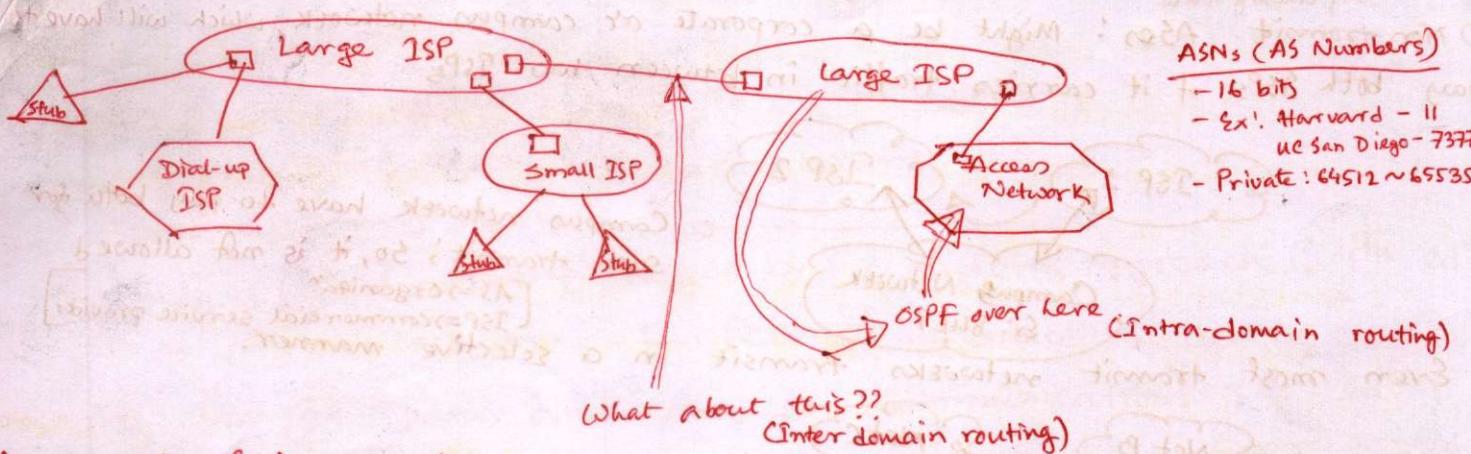
- Example: Satellite link provides high throughput long latency link, whereas fiber optic link provides lower throughput low latency link

⇒ OSPF routing hierarchy :

- Why: Many Autonomous Systems (ASes) are themselves large and non-trivial to manage. Therefore, they can be organized in hierarchy



A larger view: Internet structure



An example of inter-domain routing: Border Gateway Protocol (BGP) [Patterson's book]

⇒ Key considerations in BGP:

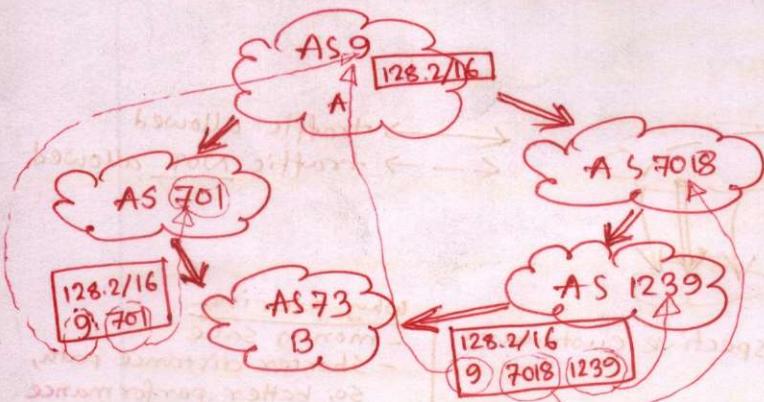
- Scalability: As forwards pkts to any address in Internet
- Domains are autonomous: No idea what interior protocol/metrics used within each
- Dominated by policy, business considerations: An AS may not believe advertisement of another AS, **a non-AS** may not carry traffic between two other ASes

⇒ Goal of BGP

- Simply finds a path between two nodes
- Does NOT try to "optimize" path

⇒ Path Vector Protocol

- Distance vector algorithm with extra information
- Extra info: for each route, store the complete path (ASes)
 - No extra computation, just extra storage
- Advantages:
 - Can make policy choices based on set of ASes in path
 - Can easily avoid loops



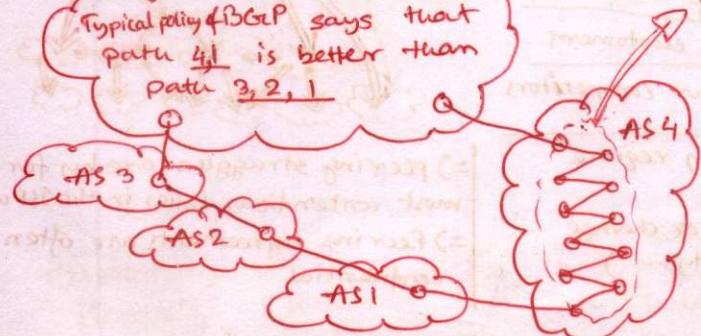
Typical policy: Prefer path with minimum AS hops

Limitation:

Typical policy of BGP says that path 4,1 is better than path 3,2,1

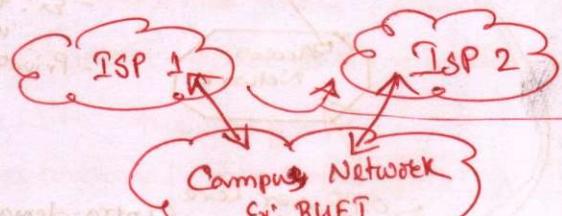
This info is completely ignored in the typical policy

⇒ Exporting internal state would dramatically increase global instability and amount of routing states



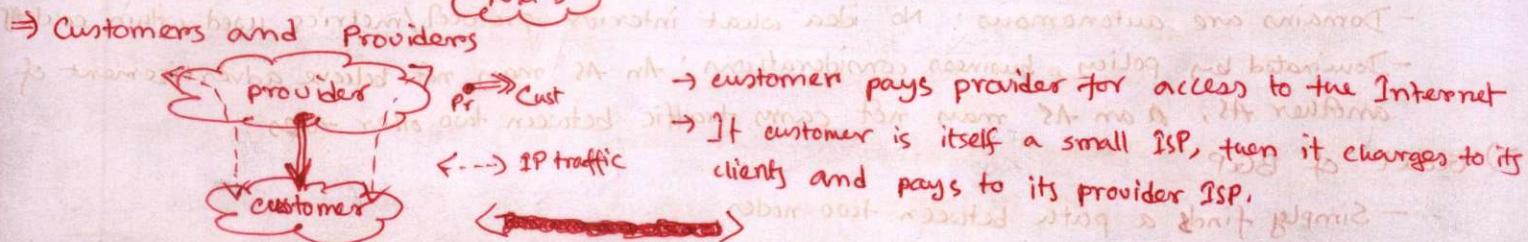
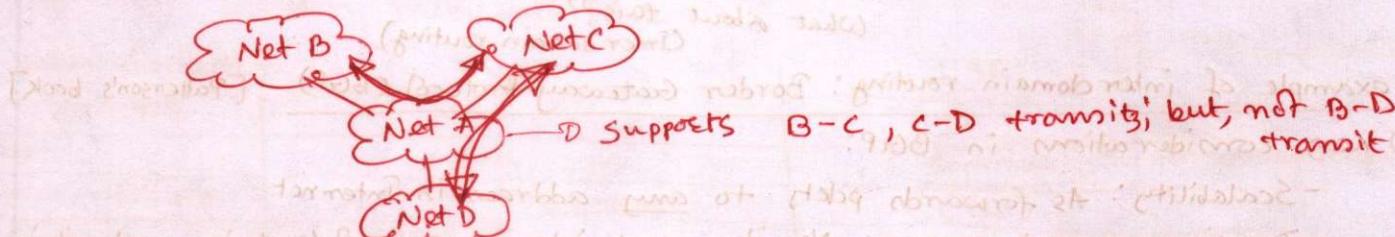
there might be different metrics in different ASes
- How to make them consistent??
for ex: delay vs # of hops

- ⇒ Dependency might be in terms of transit selection
 ↗ choosing route
- ⇒ Non-transit ASes: Might be a corporate or campus network, which will have to pay both ISPs if it carries traffic in between two ISPs

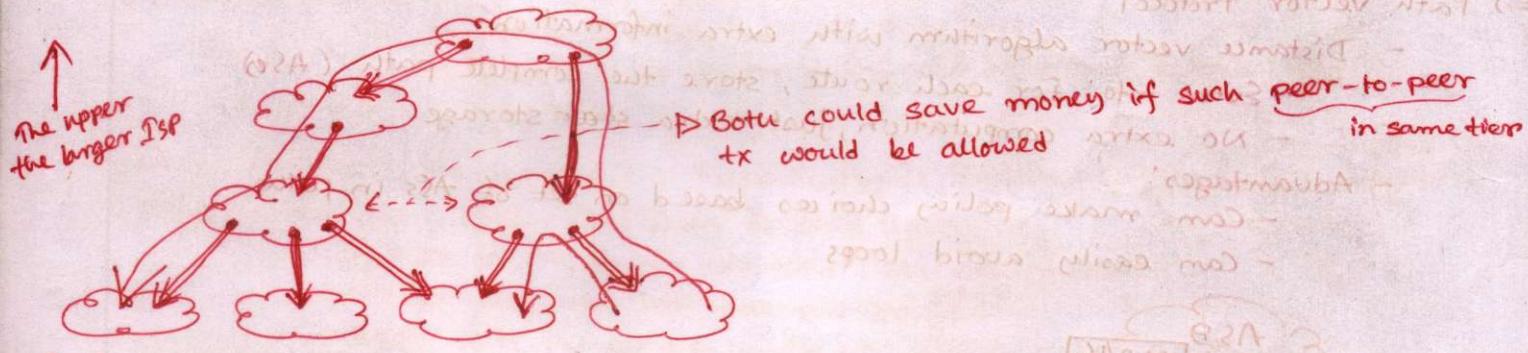


→ Campus network have to pay both for such transit; So, it is not allowed
 [AS ⇒ organization]
 [ISP ⇒ commercial service provider]

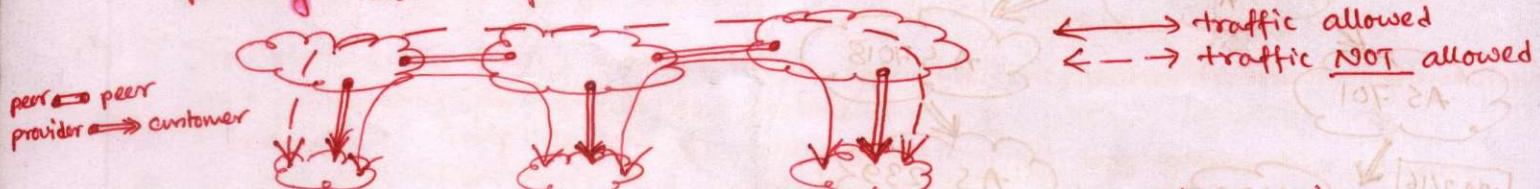
- ⇒ Even most transit networks transit in a selective manner.



- ⇒ Customer-provider hierarchy (in tiers)



- ⇒ The peering relationship:



- Peers provide Xsit between their respective customers
 → Peers do not exchange money
 → Peers do not provide Xsit between peers (***)

- ⇒ Peering wars (Source of conflict! Not clear whether two ISPs should peer or not)

- Why peering
- money save
 - shorter distance path, so, better performance
 - higher redundancy
 - Might be the only path to connect some customers



- ⇒ peering struggles are by far the most contentious issues in the ISP world
 ⇒ Peering agreements are often confidential

Peer	Don't Peer
Reduces upstream Xsit costs	You would rather have customers
Can increase end-to-end performance	Peers are usually your competitors
May be the only way to connect your customers to some part of the Internet (Tier 1)	Peering relationships may require periodic negotiation (due to business policy, or change in network connectivity, ...)