L-1

①

CSE 321: Computer Networks

Text: Computer Networks (4th edition)
- Andrew S. Tanenbaum

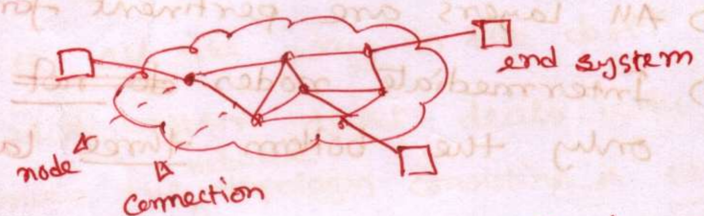
Supplimentary texts:

① Computer Networks: A Systems Approach (4th ed^n)
- Larry L Peterson, Bruce S. Davie

② Data and Computer Communications (7th ed^n)
- William Stallings

③ Computer Networking: A Top-Down Approach (4th ed^n)
- 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 other depends on each other. Even a protocol may use another protocol in its execution.

→ The form of dependency in ~~between~~ a set of 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

Sender (Customer) [Letter] ⟶ Post box [Letter in Envelop having address] ⟶ Post office (local) [Sealed envelop] → ... → Post office... (City) ⟶ Post office (local) → Receiver . . .

Open Protocols and Systems

→ A set of protocols is <u>open</u> if
- Protocol details are publicly available
- Changes are managed by an organizaⁿ whose membership and transactions are open to the public

→ A system that implements open protocols is called <u>open</u> 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

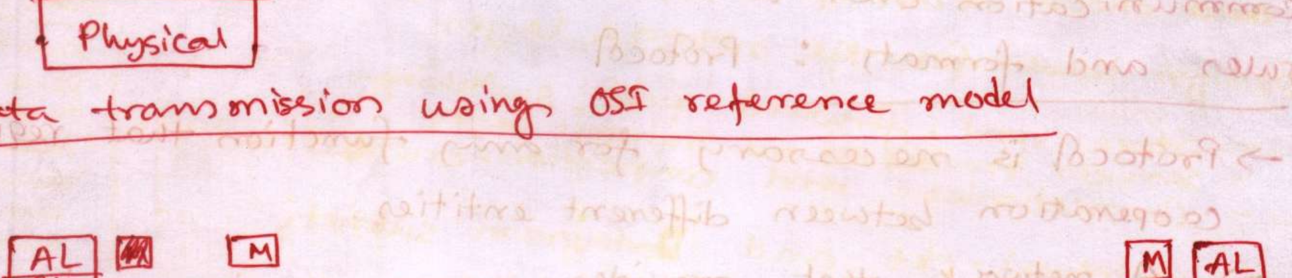
Application
Presentaⁿ
Session
Transport
Network
Data Link
Physical

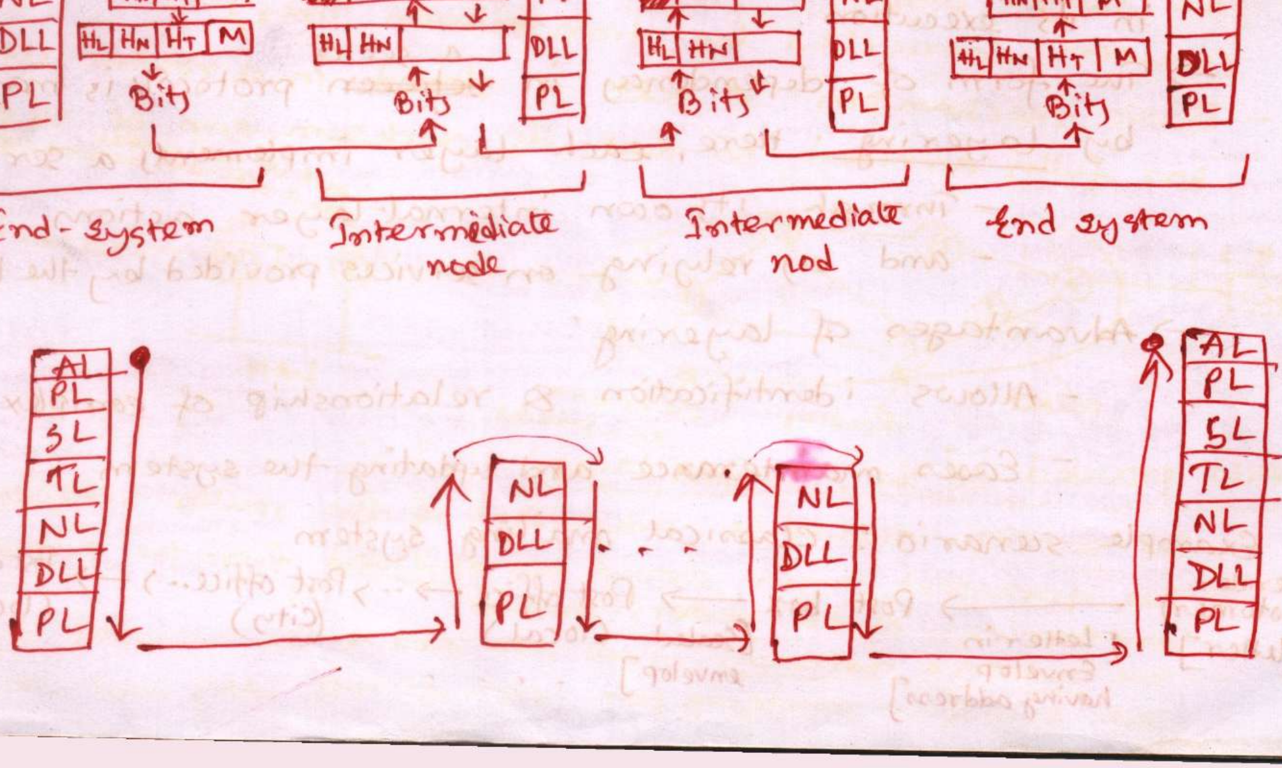
Not present in Internet protocol stack (Presentaⁿ, Session)

⇒ All layers are pertinent for end systems

⇒ Intermediate nodes ~~do not~~ deal with only the bottom <u>three</u> layers

Data transmission using OSI reference model





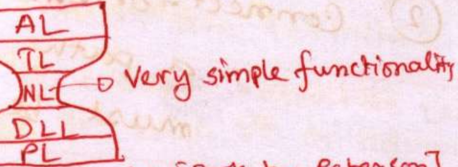
Approach of studying Computer networks

1. Bottom up approach (PL to AL) : Tanenbaum
2. Top down approach (AL to PL) : Kurose

⇒ Our approach : Hybrid of ~~tem~~ them !

PL to DLL } Simultaneously
NL to TL }
→ by Saifur Sir
→ by me

Chapter 5: Network Layer



Hour-glass model [Book by Peterson]

- ~~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

define "frame" and "pckt" over here

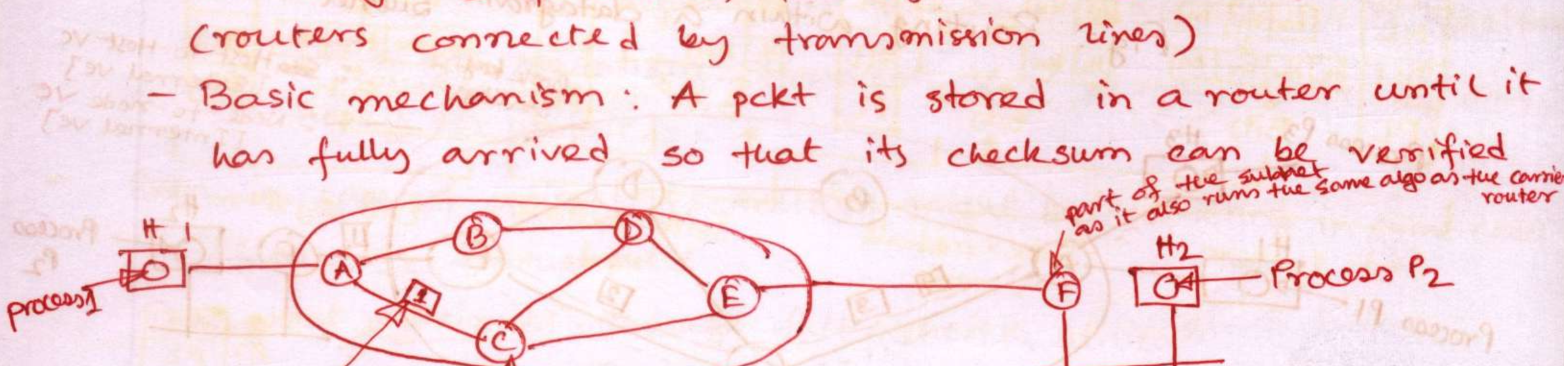
- Must know the network topology consisting a set of routers (called subnet) to ~~and~~ choose ~~a~~ appropriate paths through it.

→ 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

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

Store-and-forward pckt switching

- 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 accross LANs and WANs.

Connectionless VS Connection oriented service

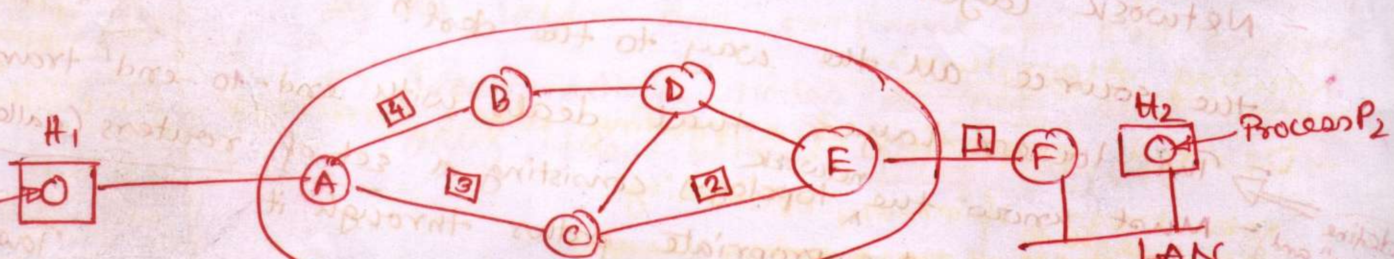
① Connectionless service:

- packets are injected into to subnet individually and <u>routed independently</u> of each other.
- ~~subnet →~~ packets are termed as ~~ckt~~ Datagram
  the subnet is " " datagram subnet

~~⇒ In case of conne~~

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

- a path from the source router to the destⁿ router must be established before any data pckts can be sent.
- the connection is termed as VC (virtual ~~circ~~ ckt)
  the subnet " " " virtual-ckt subnet



A's table

Initially for packets 1, 2, 3

A	–
B	B
C	C
D	B
E	C
F	C

Later for pckt 4

A	–
B	B
C	C
D	B
E	B
F	B

Dest Line

C's table

A	A
B	A
C	–
D	D
E	E
F	E

E's table

A	C
B	D
C	C
D	D
E	–
F	F

An internal table telling it where to send packets for each possible destⁿ.

⇒ The algorithm that manges the tables and makes the routing decisions is called the routing algorithm.

Fig: Routing within a datagram subnet

Book key stallings
- ...... : ~~Sta~~ Host-to-Host VC [External VC]
- — — — : Node-to-node VC [Internal VC]

A's table

In		Out	
H1	1	C	1
H3	1	C	2

Any packet from H1 having connection id 1, will be xfered to C with connⁿ id 1

C's table

device	connⁿ id		
A	1	E	1
A	2	E	2

E's table

C	1	F	1
C	2	F	2

A changes the connⁿ id ~~s~~ of the pckts from H3, to distinguish them from the packets arrived from H1. This is done as even though A can distinguish packets from H1 and packets from H3, C <u>cannot</u>.

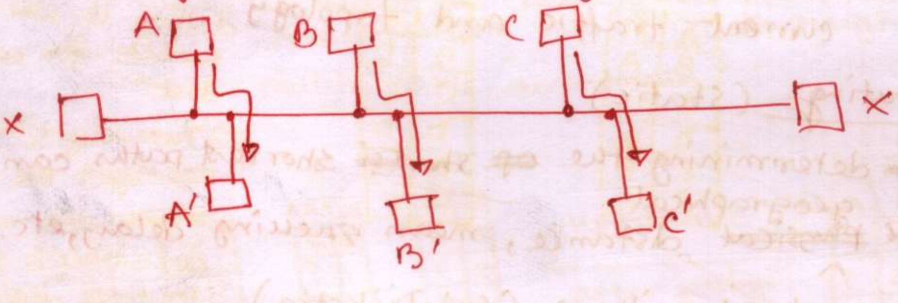
Fig: Routing over a virtual-ckt subnet.

Comparison between virtual-ckt and datagram subnets

Issue	Datagram subnet	Virtual-ckt subnet
Ckt setup req?	No	Yes
Control info in each pckt	Full src and dest.^n addresses	A short VC number
Routing of each pckt	Independent	Following the route chosen ~~wile~~ while VC setup
Effect of route failure	None, only the pckts transmitted during the crash will be lost	All VCs that passed through the failed router are terminated
QoS 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
  (a) should converge quickly
  (b) should work over a long period of time
  (b) should be able to adapt all types of topology changes)
- Conflicting metrics in routing: Fairness vs Optimality



⇒ 4 flows (A-A', B-B', C-C', X-X')
⇒ Enough traffic in the first 3 flows to saturate the horizontal communica.^n line

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

- Even in case of optimality, metrics might be conflicting in some cases.

Delay vs throughput:

Throughput maximiza.^n through full network utiliza.^n
⇓
network queues are close to their capacities
⇓
Queuing delay is increased
⇓
Total delay is increased

Conflicting

→ However, in some cases delay and througput might ~~not~~ be conflicting. Ex: # of hops ↓ (delay ↓ BW Consump^n ↓) Throughput ↑

Consistent

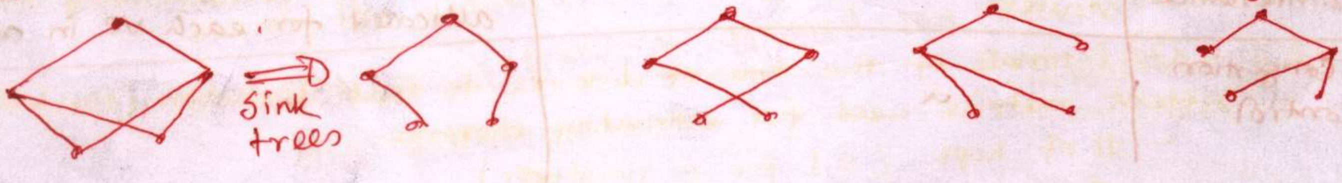
Optimality principle:

(I) ...... (J) ---- (K) optimal implies (J) --- (K) to be optimal.

Dotted line indicates that the path might be multi-hop

⇒ 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 destina^n: Sink tree (may not unique)



Routing algos

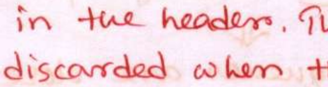
- Adaptive: Dynamic routing decisions based on estimates on current traffic & topology
- Static/Non-adaptive: ~~Rely~~ Do not base their ~~deci~~ routing decisions on measurements or estimates of the current traffic and topology

Shortest ~~P~~ Path Routing (static)

- Metrics for ~~deter~~ determining the ~~sp~~ ~~shortes~~ shortest paths can be: # of hops, ~~phys~~ ~~physical~~ 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 ~~without any~~ measure)
- Avoidance:
  1. Use a ~~header~~ hop count in the headers. 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 ~~b~~ list of seq # ~~per~~ for each source routers is maintained in each routers. An incoming pckt 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 seq #s up to K have been seen.

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

App's of flooding:

① Military: for extreme robustness
② Distributed DB: To update data concurrently
③ Wireless networks: All messages Xmitted by a sta? can be received by all sta's
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 ~~an~~ one entry for each routers in the subnet
- Each entry contains two info:
  ① Preferred outgoing line/next-hop node for corresponding destination routers
  ② Estimate of the ~~time of~~ distance to that destination (the best-known distance)
- Diffent metrics used for estimating distance:
  - # of hops (= 1 for a neighbor)
  - Time delay (by sending ECHO pckts, which initiates timestamping and sending back from the ~~peer~~ neighbors)
  - Queue length (by simply examining queus of other nodes)
- Algo:
  ① ~~The best known distance~~ Each router exchanges ~~its~~ own routing table info with its neighbors
  (only the distance part (excluding the prefferred next-hop) suffices)
  ② After receiving an info from a neighbor, a router updates its own routing table accordingly

A B C D E

Initially (only by ECHO exchanges) — full routing table

A

A	0	A
B	1	B
C	∞	-
D	∞	-
E	∞	-

B

A	1
B	0
C	1
D	∞
E	∞

C

A	∞
B	1
C	0
D	1
E	∞

D

A	∞
B	∞
C	1
D	0
E	1

E

A	∞
B	∞
C	∞
D	1
E	0

Update in C after getting routing table info from B

Dest	dist	next-hop
A	2	B
B	1	B
C	0	C
D	1	D
E	∞	-

Update in C after getting routing-table info from D

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 ~~ten~~ hop length subnet → Everyone will know good news (presence of a router in subnet) in N exchanges
(Longest path in N hop)
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)

A — B — C — D — E

distances to A

A	B	C	D	E	
	1	2	3	4	Initially
	∞	2	3	4	After B detets it has no direct conn.ⁿ to A anymore
	3	2	3	4	B thinks it can go to A through C
	3	4	3	4	C has to update the distance to A as both of its neighbors have ~~path to~~ 3-hop paths to A
	5	4	5	4	
	5	6	5	6	((will be better understandable below while discussing Algorithmic perspective of DVR))
	7	6	7	6	

Updates are Sequence dependent

or   or

⋮

⟹ Bad news travels slowly !!!   Count-to-infinity problem

Can be set as (longest path + 1)

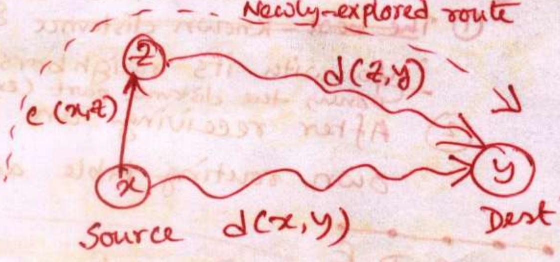
<u>Reason</u>: No idea about whether I am ~~in~~ the ~~next~~ next-hop ~~node~~ node in the neighbors path (only know its own next-hop)

→ Elaboraⁿ & Solⁿ ~~in~~ from algorithmic perspective [book from Patterson]

<u>Algorithmic perspective of DVR</u> (from the book of Patterson)

- Initially routing table in each node will have ~~only~~ finite distance for only directly-connected nodes
    - distances might be # of ~~of~~ hops or anything else that acts as weights in a graph

- Algo for update



Update (x, y, z)

    d ← c(x,z) + d(z,y) #cost of path from x to y with first hop z

    if d < d(x,y)

        #found better route

        return d, z # update cost, next hop

    else #existing cost, next hop

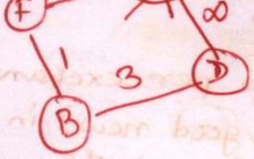
        return d(x,y), ~~next~~ hop (x,y)

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

[Also only <u>info from neighbors</u> is received]

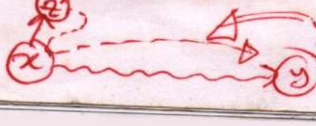
- Repeat the "Update" until convergence (Bellman-Ford Algo)
    - Time of convergence : length of the longest "shortest path", which is termed as diameter of the graph

- Impact of node failure:



⟹ | c | ∞ | – | in ~~both~~ F and D's table ⟹ | c | 2 | F | B's table arrives at D and F ⟹ | c | 5 | B | Updated entry in D's table | c | 3 | B | Updated entry in F's table (WRONG!!)

→ Solⁿ: ~~An~~ While updating, a node has to make sure that it is <u>not</u> the next hop on the path from its first-hop node



Such circular route will <u>not</u> be explored

Update (x, y, z)

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

if $d < d(x,y)$ & $x \neq nexthop(z,y)$ } "Split Horizon Rule"

#found better route

return d, z

else

return $d(x,y)$, $nexthop(x,y)$

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

→ Sol^n: Always accommodate the update from next-hop node

Update (x, y, z)

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

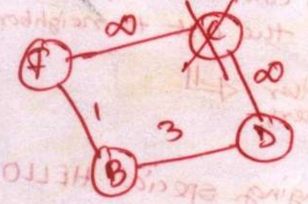
if $nexthop(x,y) = z$ | $(d < d(x,y)$ & $x \neq nexthop(z,y))$

#forced update

return d, z

else

return $d(x,y)$, $nexthop(x,y)$



⇒ | c | 5 | B | ⇒ | c | ∞ | - | ⇒ | c | ∞ | - |

in D   in B   in D   OK

→ Ques: All problems solved? Let's see...

A	B	D	F
c | 7 | F	c | 8 | A	c | 9 | B	c | 1 | C
-	-	-	c | ∞ | -
c | ∞ | -	-	-	-
-	c | ∞ | -	-	-
-	-	c | ∞ | -	-

arrival of D's entry to A

c | 13 D | ~~13~~	-	-	-
-	c | 14 | A	-	-
-	-	c | 15 | B	-
-	-	-	-
c | 19 D | ~~19~~ (forced update)			

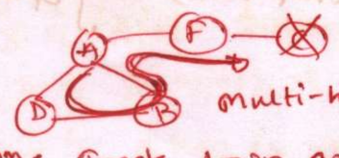
⇑ Okay

⇑ Count to infinity!!

(So, convergence depends on the sequence of update)

⇒ Reason behind sequence-dependent convergence: Only nexthop is known, entire path is <u>NOT</u> known

"A" adopts the path D → B → A → .... in which A was an intermediate node



multi-hop circular route!

→ Possible sol^n: Track down and distribute the entire path

⇒ Problem: lot more info needs to be xmitted.

→ Other sol^ns: Ensure better sequence in update

① Propagate bad news faster: Triggered update for receiving ∞ cost and periodic update otherwise (increases prob of convergence)

② Make infinity smaller finite # (ex: 100)

Still depends on the sequence, however, the extent of dependency is reduced.

<u>Link State Routing (LSR)</u>

- Basic differene with DVR:

DVR: Sends info of <u>all nodes</u> to <u>neighbors</u>

LSR: Sends info of <u>neighbors</u> to <u>all nodes</u>

- Five operational steps in LSR:

Discover neighbors ⇒ Measure cost to each neighbors ⇒ Construct a pckt incorporating the cost to neighbors ⇓ Send the info to all other routers ⇐ Compute shortest path to every other routers

① <u>Discover neighbors</u>:

→ Identification of neighbors through exchanging special HELLO packets

→ Measurement of delay/cost to neighbors through exchanging special ECHO pckts.

② <u>Measuring Line cost</u>: The ECHO pckt is sent immediately. (delay $= \frac{RTT}{2}$)

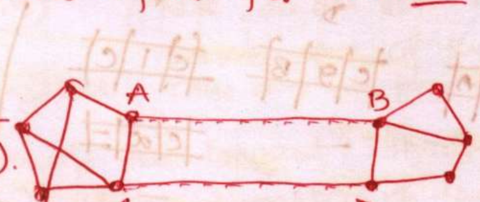
⇒ Two different techniques for starting the RTT timer.

① When the ECHO pckt is queued: Considers load balancing

② When the ECHO pckt arrives in front of Q: Does <u>NOT</u> consider load balancing

<u>load balancing</u>:

⇒ 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

$(A\text{-}B \underset{\text{traffic}}{>} C\text{-}D) \Longrightarrow$ choose C-D in new update $\xrightarrow[\text{in}]{\text{results}}$ $(C\text{-}D \underset{\text{traffic}}{>} A\text{-}B) \Longrightarrow$ choose A-B in new update ↩ ... continues

→ Sol^n: ~~Fracs of~~ traffic ~~are~~ is splitted 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 (must be .5, .5 again) as balanced in prev. case

Project : after 10 classes ~~//~~ ⊗

- Simulation of <u>different types of networks</u> in <u>ns-2</u> with already-available & modified <u>protocols</u> and produce values of <u>different metrics</u>

<u>Different types of networks</u>

Wired, optical, Wireless (static), Mobile, Sensor, Satellite

⇒ Combination of at least two types as mentioned above → 15 combinations

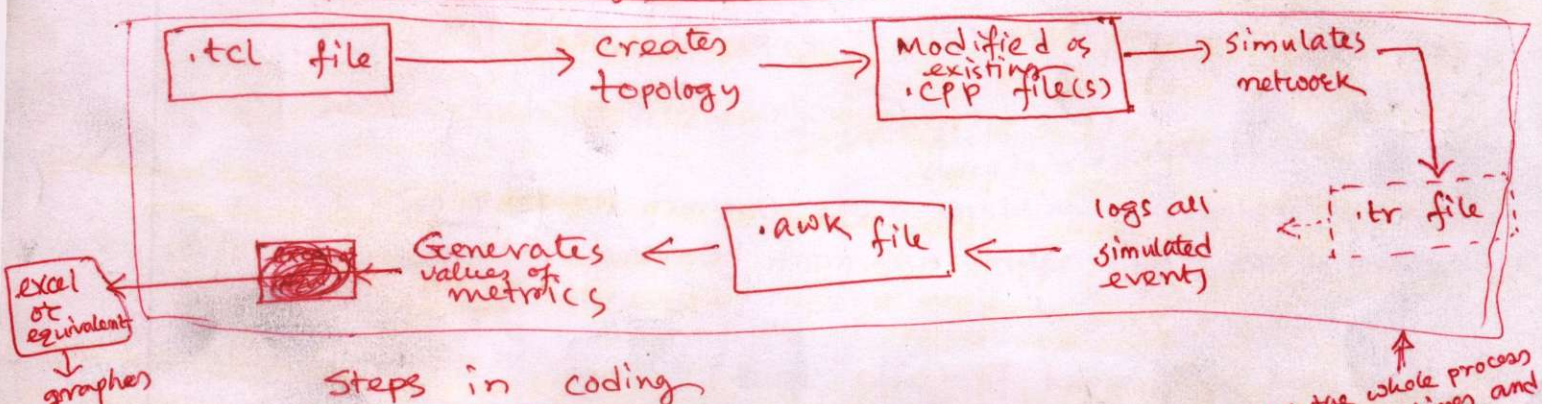
(only exception : Satellite)

⇒ Might be in flat ~~to~~ or hierarchical topology

Different names ⇒ Backbone networks, ad-hoc networks, ~~..~~ mesh networks, mobile ad-hoc networks, vehicular ad-hoc networks, ...

<u>ns-2</u>

<u>Steps ~~in coding~~ of execution</u>



<u>Steps in coding</u>

.tcl → ~~.cpp modifications~~ .awk → .sh ⇒ runs the whole process for multiple times and generates average statistics

.sh → .cpp modifica^ns

<u>.sh</u>

~~change~~ 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~~ value of ~~of~~ the generated metrics
end loop
write averaged value to a file

<u>protocols</u> :

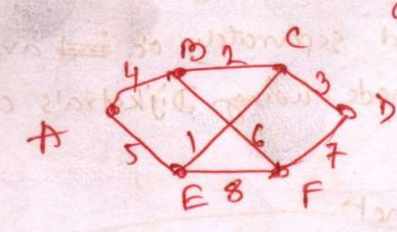
MAC or NL or TL [or even PL models !] ⇐ change any of them with some intuitive explanations

<u>Metrics</u> : Network throughput, end-to-end delay, jitters, per-node throughput, energy consumption, ...

⇒ Any brand new thinking will get ~~to~~ BONUS !!!

③ Constructing Link State Packets

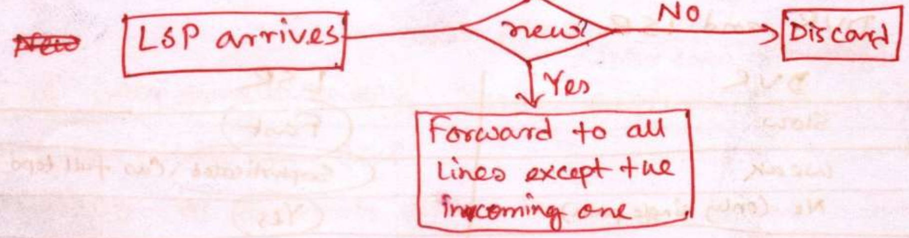
- LSP contains: own identity, a seg# (increases during each sending), age (decreased in each sec), list of neighbors



A	
Seg	
Age	
B	4
E	5

B	
Seg	
Age	
A	4
C	2
F	6

④ Distributing LSPs (~~The~~ Trickiest part of the algo)



[similar to flooding] extent is kept checked using seg#. determines whether a newly-received pckt is duplicate (same#) or obsolete (less # compared to already seen ones)

⇒ Problems with the algo (mainly with seg#):

① Wraping up of the seg#: (0, 1, ... max#, 0, 1, ...) — seems to be obsolete

- 32-bit seg#, 1 LSP per sec ⇒ 137 years to wrap around [So, ~~the~~ wrap around may happen]

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

- Seg # 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 Seg #: (Huge effect if happens in MSB side)

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

Sol.n: Include ~~the~~ "age", and decrement it once per ~~seg~~ sec, ~~at~~ by ~~each router~~ when the "age" hits zero, info from that router is discarded. [So, if any of the prev. problem could happens, 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, after being received, is not immediately queued ~~for~~ 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 pckt or ack to send.

Source	Seg#	Age	Send flags A	Send flags C	Send flags F	ACK flag A	ACK flag C	ACK flag F	
A	21	60	0	1	1	1	0	0	→ Arrived directly from A; so, ack to A send to rest
F	21	60	1	1	0	0	0	1	→ " " " F; so, ack to F send to rest
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

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

Holding area in B

⇒ If we would store only ~~ACK~~ 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)
beginning of each period

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 ~~and~~ 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

O → better

~~metric~~ Attribute	DVR	LSR
Convergence ~~rate~~	Slow	Fast
Metric	Weak	Sophisticated (as 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).
↳ in some IP backbones

OSPF :

→ Open : 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 ~~new~~ LSA)

> : Update & propagate to all other lines

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

= : Ignore

⇒ 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 seq number 0
- Will get LSAs with last time sent max seq # ←
- Router sets the seq # to (max seq # + 1) and resends

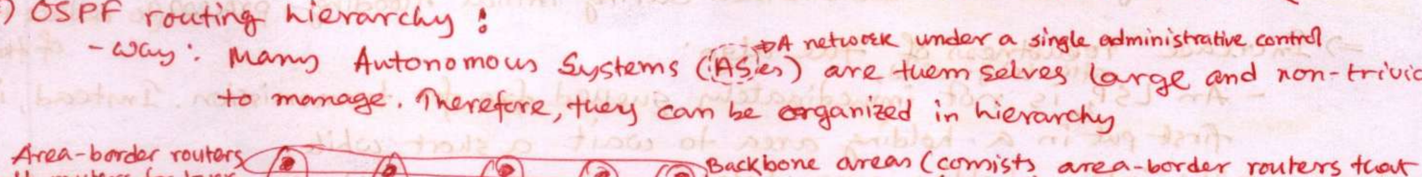
⇒ Seq #, age, Load balancing → Similar as discussed above

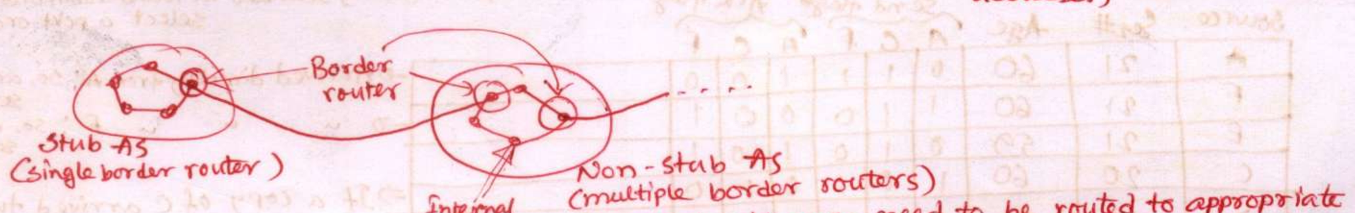
⇒ Types of Service (TOS) metrics : Link characteristics varies in multiple dimensions such as latency, throughput, cost, reliability

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

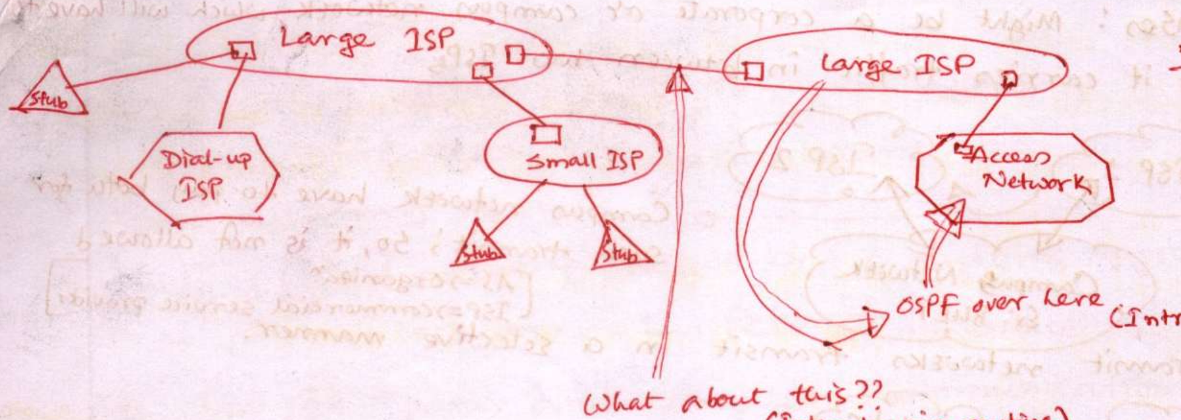
⇒ OSPF routing hierarchy :

- Why : Many Autonomous Systems (ASes) are themselves large and non-trivial to manage. Therefore, they can be organized in hierarchy
→ A network under a single administrative control





⇒ larger view : Internet structure



ASNs (AS Numbers)
- 16 bits
- Ex: Harvard - 11
  UC San Diego - 7377
- Private : 64512 ~ 65535

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

⇒ Key considerations in BGP:

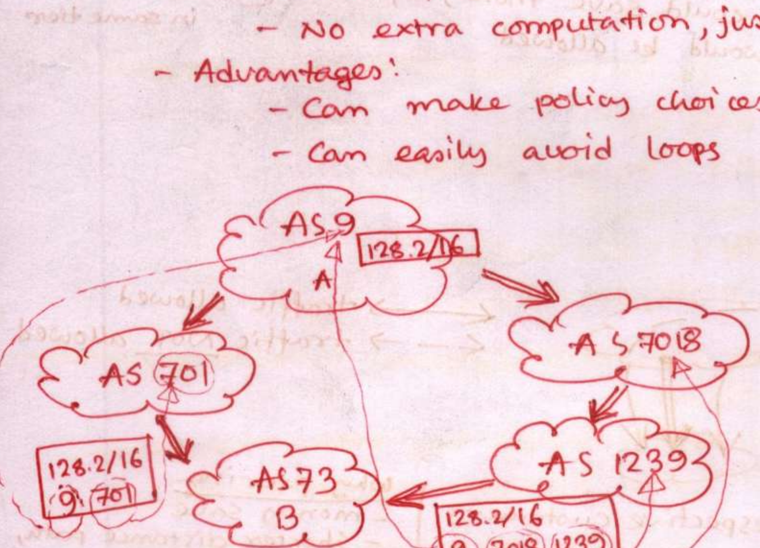
- Scalability : As forwards pckts to <u>any</u> address in Internet
- Domains are autonomous : No idea what interior protocol/metrics used within each AS
- Dominated by policy, business considerations : An AS may not believe advertisement of another AS, ~~A~~ an AS may not carry traffic between two other ~~AS~~s

⇒ Goal of BGP

- Simply finds a path between two nodes
- Does <u>NOT</u> ~~typ~~ try to "optimize" path

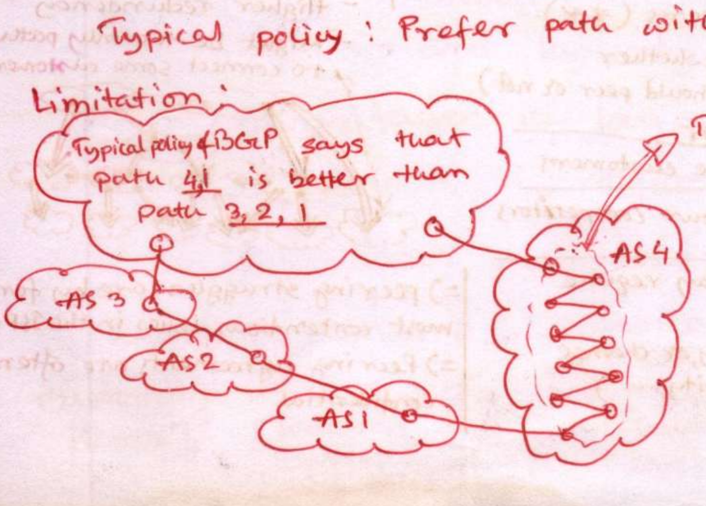
⇒ Path Vector Protocol

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



Typical policy : Prefer path with minimum AS hops

Limitation :



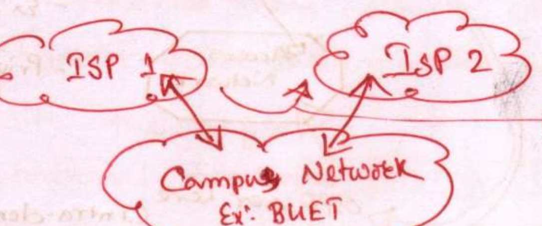
This info is completely ignored in the typical policy

⇒ Exporting internal state would dramatically <u>increase</u> global <u>instability</u> and amount of routing states
(as too much <u>dependency</u>)

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
↳ of choosing route

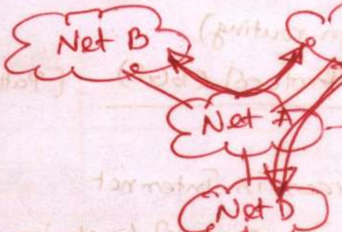
=) Non transit ASes : Might be a corporate or campus network, which will have to pay both ISPs if it carries traffic in between two ISPs



→ Campus network have to pay both for such transit; so, it is not allowed

[AS => organiza^n
ISP => commercial service provider]

=) Even most transit networks transit in a selective manner.



→ supports B–C, C–D transit; but, not B–D transit

=) Customers and Providers

Pr → Cust

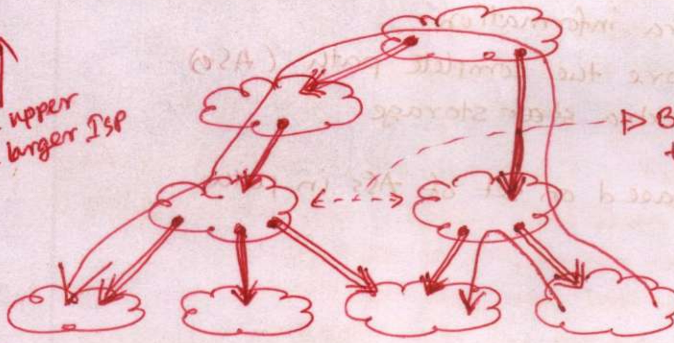
<- - -> IP traffic

→ customer pays provider for access to the Internet

→ If customer is itself a small ISP, then it charges to its clients and pays to its provider ISP.

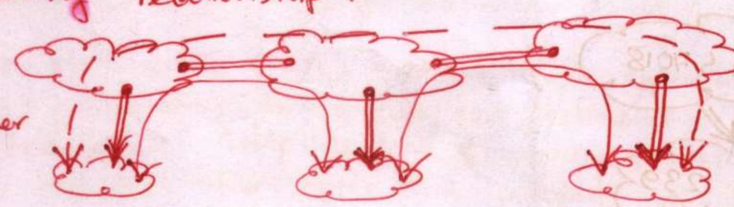
=) customer-provider hierarchy (in tiers)

The upper the larger ISP



→ Both could save money if such peer-to-peer tx would be allowed
in same tier

=) The peering relationship :

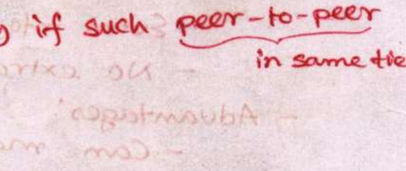


⟷ traffic allowed
<- - -> traffic NOT allowed

peer ⟷ peer
provider ⟹ customer

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

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



=) Peering wars (Source of conflict : Not clear whether two ISPs should peer or not)

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 rel^ships may require periodic negotiation (due to business policy, or change in network connectivity, ...)

=) peering struggles are by far the most contentious issues in the ISP world

=) Peering agreements are often confidential