Session 5A

Distance Vector Routing

Mouli Sankaran

Session 5A: Focus

- Routing Protocols
  - Network as a Graph
  - Costs of the links
  - Issues with Static Routing
- Distance Vector Routing
  - Explained with an Example
  - Summary
  - Recovery from Node or Link failures
- GATE 2023: Three Networking Questions

Course page where the course materials will be posted as the course progresses:

Routing Protocols

Distance Vector and Link State Routing

- Routing protocols fall into **two main categories**
  - **Distance Vector** and
  - **Link State**
- Distance Vector protocols determine best path on how far the destination is.
- While Link State protocols are capable of using more sophisticated methods taking into consideration **link variables,** such as:
  - Bandwidth, delay, reliability, load, etc.

Distance Vector Routing

- Distance Vector protocols judge best path on how far it is.
- Distance can be hops or a combination of metrics calculated to represent a distance value.
- The following lists the IP Distance Vector routing protocols still in use today
- And the metric that they use for determining the best path:
  - RIP v1 (**hops**)
  - RIP v2 (**hops**)
  - IGRP (**combined metric, bandwidth** and **delay**)

RIP: Routing Information Protocol

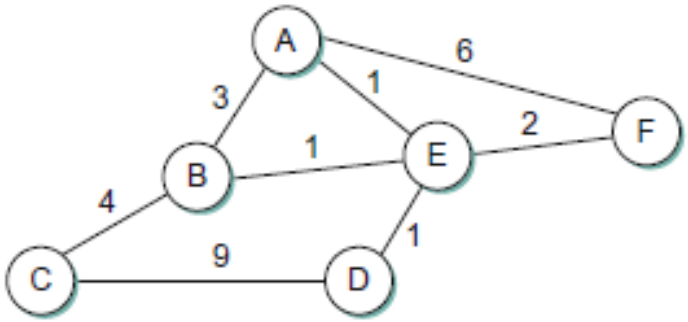
v1 and **v2** are **versions 1** and **2**

- The best known and most popular Distance Vector protocol to date is IP RIP.
- **IGRP** was originally developed by Cisco Systems and also supports the routing of IP traffic

IGRP: Interior Gateway Routing Protocols

Network as a graph

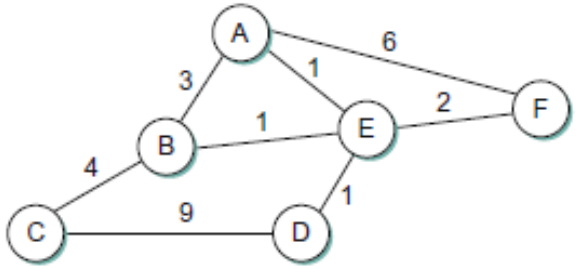
- Routing is, in essence, a problem of graph theory.
- The figure shows a graph representing a network
- The nodes of the graph, labeled A through F, may be:
  - **Hosts, switches, routers, or networks** **What** do the **nodes** represent?



- What do the nodes represent?
- Switches could also be represented by the nodes, because switches could also be IP aware or L3 switches
- As you are aware, telnet into L3 switches are also possible, which means that a L3 switch could also be similar to a host with its own (VLAN) IP.
- Routers are of course considered as nodes, interconnected with links

Cost of the links

- For **our** initial **discussion**, we will **focus** on the case where the **nodes** are **only routers**.
- The edges of the graph correspond to the network links between the routers.
- Each edge has an associated *cost*, which gives some indication of the desirability of sending traffic over that link.



Note: Undirected edges with each edge assigned a single cost are assumed here. Though edges could be directed with pair of edges between two nodes, each with its own edge costs.

- Cost of a link can be derived based on various metrics that we had discussed in the last session (bandwidth, delay, reliability, load, etc.)
- The basic problem of routing is to find the lowest-cost path between any two nodes.
- Where the cost of a path equals the sum of the costs of all the edges that make up the path.

Issues with the Static Routing (To find the Shortest Paths)

- We could imagine just calculating all the shortest paths and loading them into some nonvolatile storage on each node.
- Such a static approach has several **shortcomings**:

1. It does not deal with node or link failures.
2. It does not consider the addition of new nodes or links.
3. It implies that edge costs cannot change, even though we might reasonably wish to have link costs change over time.
   - **Example**: assigning high cost to a link that is heavily loaded.

- For these reasons, **routing** is achieved in most practical networks **by** running **routing protocols** among the nodes.
- These protocols provide a **distributed**, dynamic way to solve the problem of finding the lowest-cost path in the presence of link and node failures and changing edge costs.

Distributed Routing Protocols: Explained

- The distributed nature of routing algorithms is one of the main reasons why this has been such a rich field of research and development.
- There are a lot of **challenges** in making **distributed algorithms** work well.
- For example, distributed algorithms raise the possibility that **two routers** will **at one instant** have **different ideas** about the **shortest path** to **some destination**.
- In fact, each one may think that the other one is closer to the destination and decide to send packets to the other one.
- Clearly, such **packets** will be **stuck in a loop** until the discrepancy between the two routers is resolved, and it would be good to resolve it as soon as possible.
- This is just one example of the type of problem routing protocols must address.

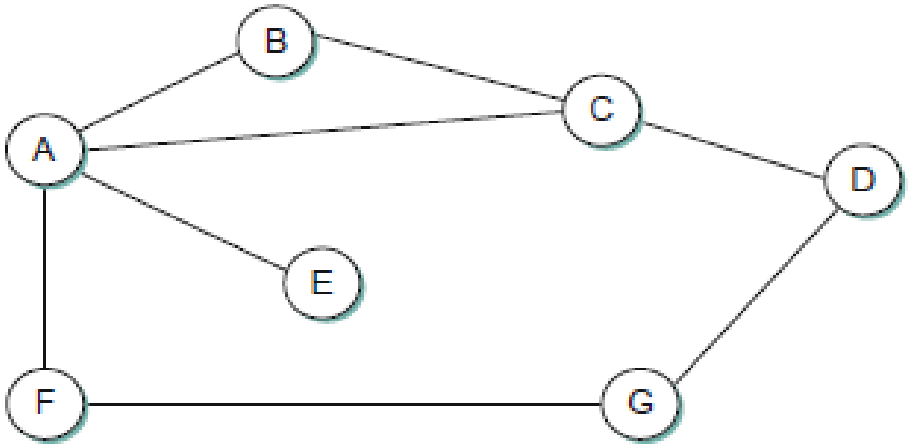
Distance Vector Routing

Distance Vector Routing

- The idea behind the distance-vector algorithm is suggested by its name.
- Each node constructs a one-dimensional array (a vector) containing the "distances" (costs) to all other nodes and distributes that vector to its immediate neighbors.
- The starting assumption for distance-vector routing is that each node knows the cost of the links to each of its directly connected neighbors.
- These costs may be provided when the router is configured by a network manager, along with the IP address of each interface of the routers connected to the network.
- A link that is down is assigned an infinite cost.

1. Distance Vector Algorithm

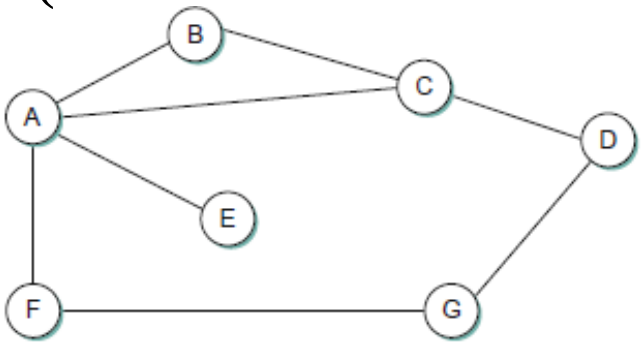
- To see how a distance-vector routing algorithm works, it is easier to consider an example like the one depicted in the figure.



- In this example, the cost of each link is set to 1, so that a **least-cost path** is simply the **one with the fewest hops**.
- Since **all edges** have the **same cost**, costs are **not shown** in the picture above.

2. Distance Vector Algorithm

- We can represent each node's knowledge about the distances to all other nodes as a table.
- Each node knows only the information in one row or column of the table (the one that bears its name).



Initial Distances Stored at Each Node (Global view)

Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

Note:. The **global view** that is presented here is **not available** at any **single point** in the **Network**, they are available on each router in the network.

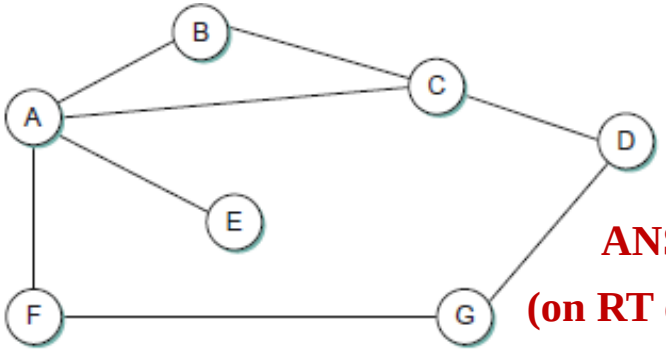
Q1. Each row or **column** is **stored** in each **router?** **ANS1:**

Entries in the row and column of each router are the same!!!

3. Distance Vector Algorithm

- Each **row** in **the routing table** is the current beliefs of that node on the distance to all other nodes.
- Thus, A initially believes that it can reach B in one hop and the Node A does not even know the existence of nodes D & G.
- The routing table stored at A reflects this set of beliefs and includes the name of the next hop that A would use to reach any reachable node.

Note: Though entries are shown, cost with ∞ are not present in RT.



ANS1:

(on RT of nodeA)

Initial Routing Table at Node A

Destination	Cost	NextHop
B	1	B
C	1	C
D	∞	—
E	1	E
F	1	F
G	∞	—

Q1: How many entries on power on?

4

1.There is **no entry** for itself (**node A**) because this is the **routing table** stored in **Node A**. Remember we are considering only routers as nodes here.

2. Next hop entries in **RT** are **actually** the **IP addresses** of the **Next hop Router Interfaces,** directly connected to the Router.

4. Distance Vector Algorithm (C to A)

Initial RT at node A

Destination	Cost	NextHop
B	1	B
C	1	C
D	∞	—
E	1	E
F	1	F
G	∞	—

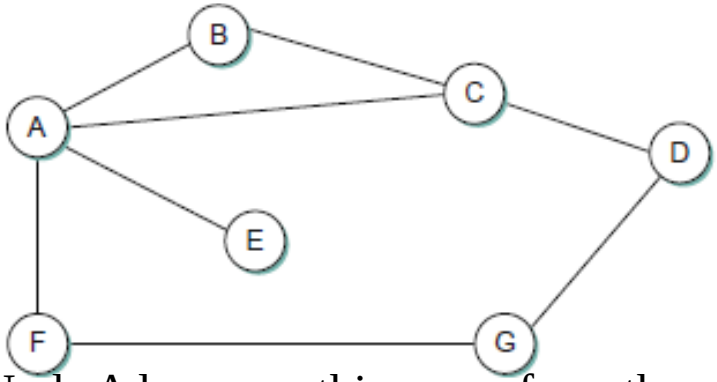
From Node C to Node A

Initial RT at node C

Destination	Cost	Next Hop
A	1	A
B	1	B
D	1	D
E	∞	--
F	∞	--
G	∞	--

RT of Node A after Receiving update from Node C

Destination	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	∞	--



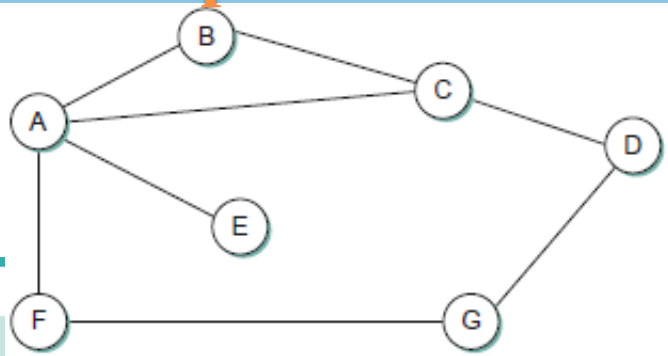
1. Does the Node A learn anything new from the update it receives from Node C? **ANS1: Yes.**

2. What changes does Node A make, to its RT based on the update from Node C? **ANS2: Node A adds a new entry for Node D, which it didn't know about.**

5. Explanation on Node A learning from the Initial update from Node C

Initial RT at node A

Destination	Cost	NextHop
B	1	B
C	1	C
D	∞	—
E	1	E
F	1	F
G	∞	—



Changes to RT at Node A from is initial state based on the first update from Node C

Destination	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	∞	--

- For example, A learns from C that D can be reached from C at a cost of 1; it adds this to the cost of reaching C (1) and decides that D can be reached via C at a cost of 2, which was not in the RT earlier on Node A's RT.
- At the same time, A learns from C that B can be reached from C at a cost of 1, so it concludes that the cost of reaching B via C is 2.
- Since this is worse than the current cost of reaching B (1), this new information is ignored, no update is done related to its entry about Node B.

6. Distance Vector Algorithm (B to A)

RT at Node A after the update from C was received

Destination	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	∞	--

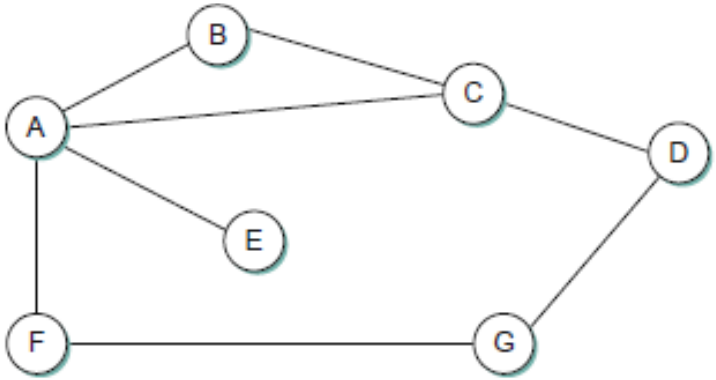
From Node B to Node A

Initial RT at node B

Destination	Cost	Next Hop
A	1	A
C	1	C
D	∞	--
E	∞	--
F	∞	--
G	∞	--

RT of node A after Receiving Update from node B

Destination	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	∞	--



1. Does the Node A learn anything new from the update it receives from Node B? **ANS1: No.**

2. What is the change Node A makes to its RT?

ANS2: No changes.

7. Distance Vector Algorithm (E to A)

RT at Node A after updates from B & C were received

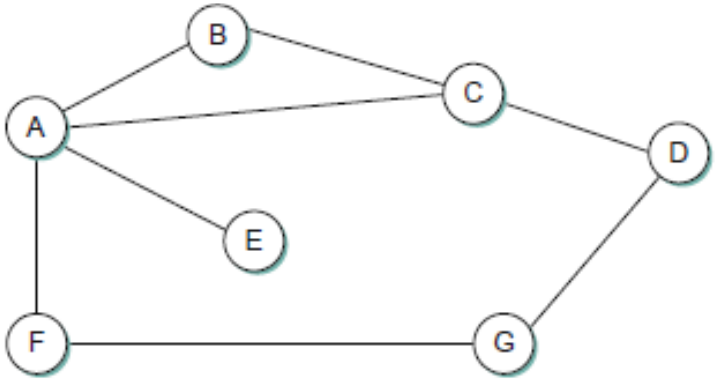
Destination	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	∞	--

Initial RT at node E

Destination	Cost	Next Hop
A	1	A
B	∞	--
C	∞	--
D	∞	--
F	∞	--
G	∞	--

RT of Node A after Receiving update from Node E

Destination	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	∞	--



1. Does the node A learn anything new from the Update it receives from node E? **ANS1: No.**

2. What is the change does node A make? **ANS2: None.**

8. Distance Vector Algorithm (F to A)

RT at node A after updates from B, C & E were received

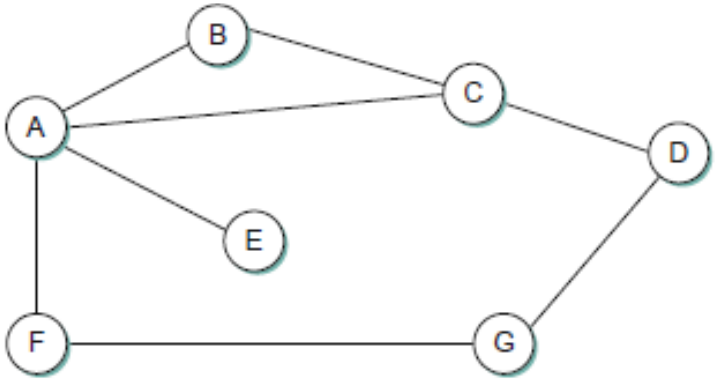
Destination	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	∞	--

Initial RT at node F

Destination	Cost	Next Hop
A	1	A
B	∞	--
C	∞	--
D	∞	--
F	∞	--
G	1	G

RT of node A after Receiving Update from node F

Destination	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F



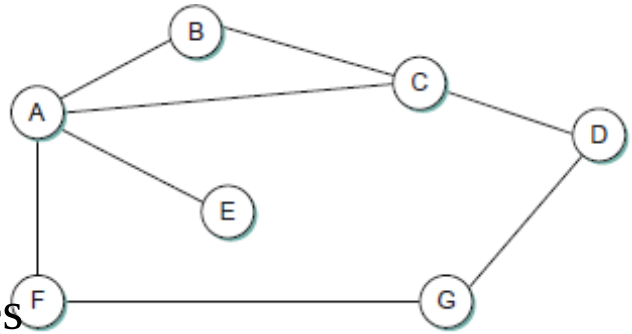
1. Does the node A learn anything new from the Update it receives from node E? **ANS1: Yes.**

2. What is the change does node A make?

ANS2: It updates the entry for G with a hop distance of 2 through node F.

9. Distance Vector Algorithm (Global view and local view at node A)

- In the absence of any topology changes, it takes only a few exchanges of information between neighbors before each node has a complete routing table.
- The process of getting consistent routing information to all the nodes is called ***convergence.***
- The final set of costs from each node to all other nodes when routing has converged is shown below.



RT at node A receiving all the updates
All the nodes in the network

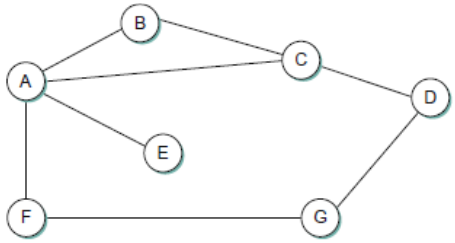
Destination	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

Individual view at node A is the same as the global view, after convergence

Information stored at Node	Distance to Reach Node A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

10. Distance Vector Algorithm: Summary

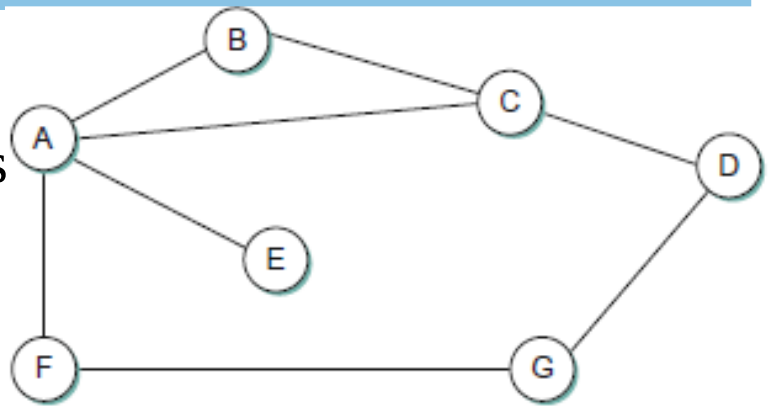
- Remember that there is no one node in the network that has the global view of all the nodes in the network, each node only knows about the contents of its own routing table.



- The beauty of a distributed algorithm like this is that it enables all nodes to achieve a consistent view of the network in the absence of any centralized authority.
- There are **two different circumstances** under which a given node decides to send a routing update to its neighbors.
- One of these circumstances is the ***periodic*** **update**.
- In this case, each node automatically sends an update message every so often, even if nothing has changed.
- This serves to let the other nodes know that this node is still running. It also makes sure that they keep getting information that they may need if their current routes become unviable.

11. Distance Vector Algorithm: Summary

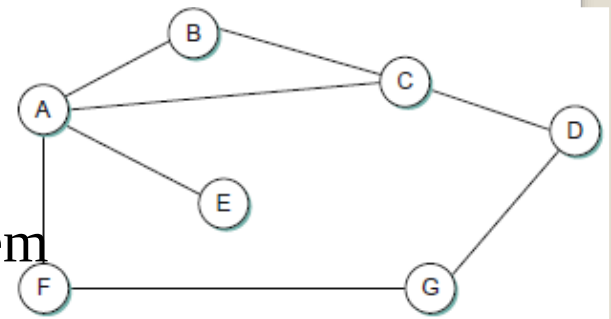
- The frequency of these periodic updates varies from protocol to protocol, but it is typically on the order of several seconds to several minutes.



- The **second mechanism**, sometimes called a ***triggered*** update, happens whenever a node notices a link failure or receives an update from one of its neighbors that causes it to change one of the routes in its routing table.
- Whenever a node's routing table changes, it sends an update to its neighbors, which may lead to a change in their tables, causing them to send an update to their neighbors.

12. Distance Vector Algorithm: (when a node or link fails)

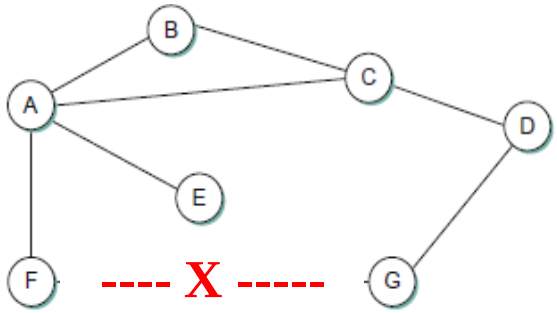
- Now consider what happens when a link or node fails.
- The nodes that notice first, send new lists of distances to their neighbors, and normally the system settles down fairly quickly to a new state



- As to the question of how a node detects a failure, there are a couple of different answers.
- In one approach, a node continually tests the link to another node by sending a control packet and seeing if it receives an acknowledgment.
- In another approach, a node determines that the link (or the node at the other end of the link) is down if it does not receive the expected periodic routing update for the last few update cycles.

13. Distance Vector Algorithm: (Example: Link to G from F has failed)

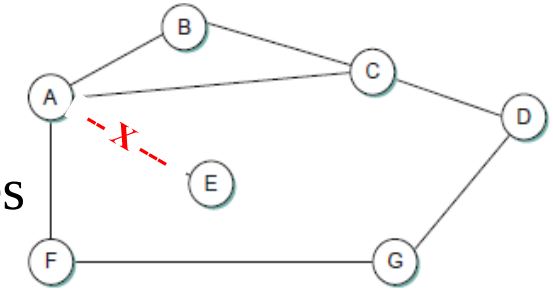
- To understand what happens when a node detects a link failure, consider what happens when F detects that its link to G has failed.
- First, F sets its new distance to G to infinity and passes that information along to A.



- Since A knows that its 2-hop path to G is through F, A would also set its distance to G to infinity. However, with the next update from C, A would learn that C has a 2-hop path to G.
- Thus, A would know that it could reach G in 3 hops through C, which is less than infinity, and so A would update its table accordingly.
- When it advertises this to F, node F would learn that it can reach G at a cost of 4 through A, which is less than infinity, and the system would again become stable.

Distance Vector Algorithm: Count to infinity Problem (Example: Link to E from A fails)

- Unfortunately, slightly different circumstances can prevent the network from stabilizing.
- Suppose, for example, that the link from A to E goes down.



- In the next round of updates, A advertises a distance of infinity to E, but B and C advertise a distance of 2 to E, of course via the node A.
- Depending on the exact timing of events, the following might happen: Node B, upon hearing that E can be reached in 2 hops from C, concludes that it can reach E in 3 hops and advertises this to A; node A concludes that it can reach E in 4 hops and advertises this to C; node C concludes that it can reach E in 5 hops; and so on.
- This cycle stops only when the distances reach some number that is large enough to be considered infinite.
- In the meantime, none of the nodes actually knows that E is unreachable, and the routing tables for the network do not stabilize.
- This situation is known as the ***count to infinity*** **problem**.

GATE 2023 Questions on Networking

Stop-and-wait Protocol (Layer 2)

- The Stop-and-Wait protocol is a simple data link layer (Layer 2) used in point-to-point communication.
- It ensures reliable transmission of data by following an ACK-based mechanism where the sender waits for an ACK from the receiver before sending the next data frame.
  1. Sender transmits a data frame (Packet #1).
  2. Sender waits for an acknowledgment (ACK) from the receiver.
  3. Receiver receives the frame, processes it, and sends an ACK.
  4. Upon receiving the ACK, the sender transmits the next frame (Packet #2).
  5. If no ACK is received (due to loss or error), the sender retransmits the same frame after a timeout.
- If the **link** is **long**, **propagation delay** is **large**, because the sender has to wait longer before receiving an ACK.
- A **higher transmission rate** means **transmission delay is small**, leading to a longer idle periods between data transmissions.

CS: GATE 2023 : Q17 — 1 mark

- Suppose two hosts are connected by a **point-to-point link** and they are configured to use **Stop-and-Wait protocol** for reliable data transfer. Identify in which one of the following scenarios, the **utilization of the link is the lowest**:

A. Longer link length and lower transmission rate
B. Longer link length and higher transmission rate
C. Shorter link length and lower transmission rate
D. Shorter link length and higher transmission rate

ANS: B

Note: Utilization will be higher when the idle time of the link is lower than the time when there is transmission of data over it.

CS: GATE 2023 : Q50

2 marks

Multiple Correct Options

Suppose you are asked to design a new reliable byte-stream transport protocol like TCP. This protocol, named **myTCP,** runs over a **100 Mbps** network with **Round Trip Time** of **150 milliseconds** and the **Maximum Segment Lifetime (MSL)** of **2 minutes**.

Which of the following is/are **valid lengths** of the **Sequence Number field** in the **myTCP** header? **ANS: B, C and D** **Note**: 31 or more bits needed

A. 30 bits
B. 32 bits
C. 34 bits
D. 36 bits

MSL: It refers to the maximum amount of time a TCP segment can exist in the network before being discarded.

The length of Sequence number field should be such that it is large enough to accommodate all the bytes transmitted are valid within an MSL period, so that a delayed or duplicate packet is not misinterpreted as a valid new packet. **Seq no. should not wrap around with in a MSL period.**

Need to find the **number of bytes** that could be **transmitted** at the given **link rate** within an **MSL period of 2 minutes (120 seconds).**

Link rate = 100 Mbps = 100 x 10^6 bits/sec = (100 x 10^6) /8 Bytes/sec = 12.5 x10^6 Bytes/sec

Total bytes can be sent during MSL = 12.5 x 10^6 x 120 Bytes/sec = 1.5 x 10^9 Bytes/sec

32 bits wide → 2^{32} → 4.3 x 10^9 31 bits → 2^{31} → 2.1 x 10^9 30 bits → 2^{30} → 1.07 x 10^9

CS: GATE 2023 : Q65

2 marks

The forwarding table of a router is shown below.

Subnet Number	Subnet Mask	Interface ID
200.150.0.0	255.255.0.0	1
200.150.64.0	255.255.224.0	2
200.150.68.0	255.255.255.0	3
200.150.68.64	255.255.255.224	4
Default		0

A packet addressed to a destination address 200.150.68.118 arrives at the router. It will be forwarded to the interface with ID _____.

ANS: 3

Note: Remember router does the **longest-prefix matching** of the destination IP address with the entries in the Routing table, to find the interface on which the packet needs to be routed. If none of the entries match with the destination IP, default route will be chosen.

Here, **200.150.68.0** has the match of **24 bits** with the destination IP **200.150.68.118**.

The last entry, after applying the subnet mask (**/27**) on the dest IP → 200.150.68.96, and not 200.150.68.64 as shown in the Routing table. So, the 4th entry does not match.

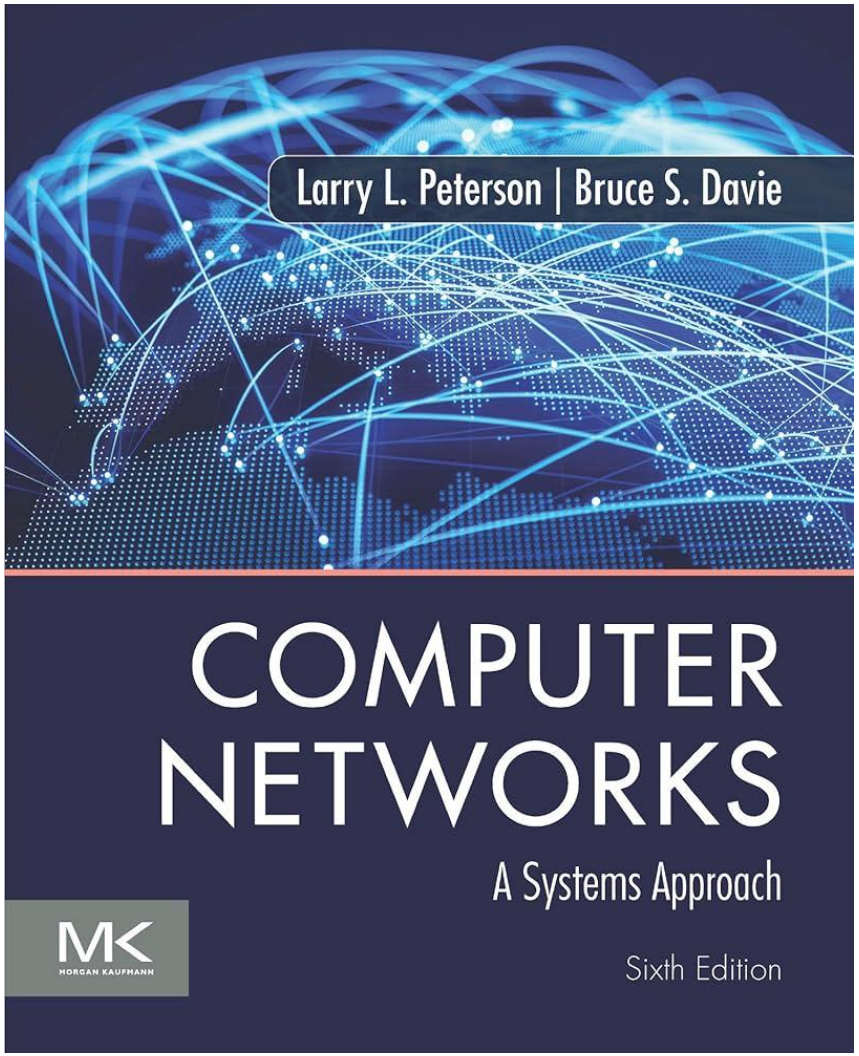
224 → 0xE0 118 → 0x76

Session 5A: Summary

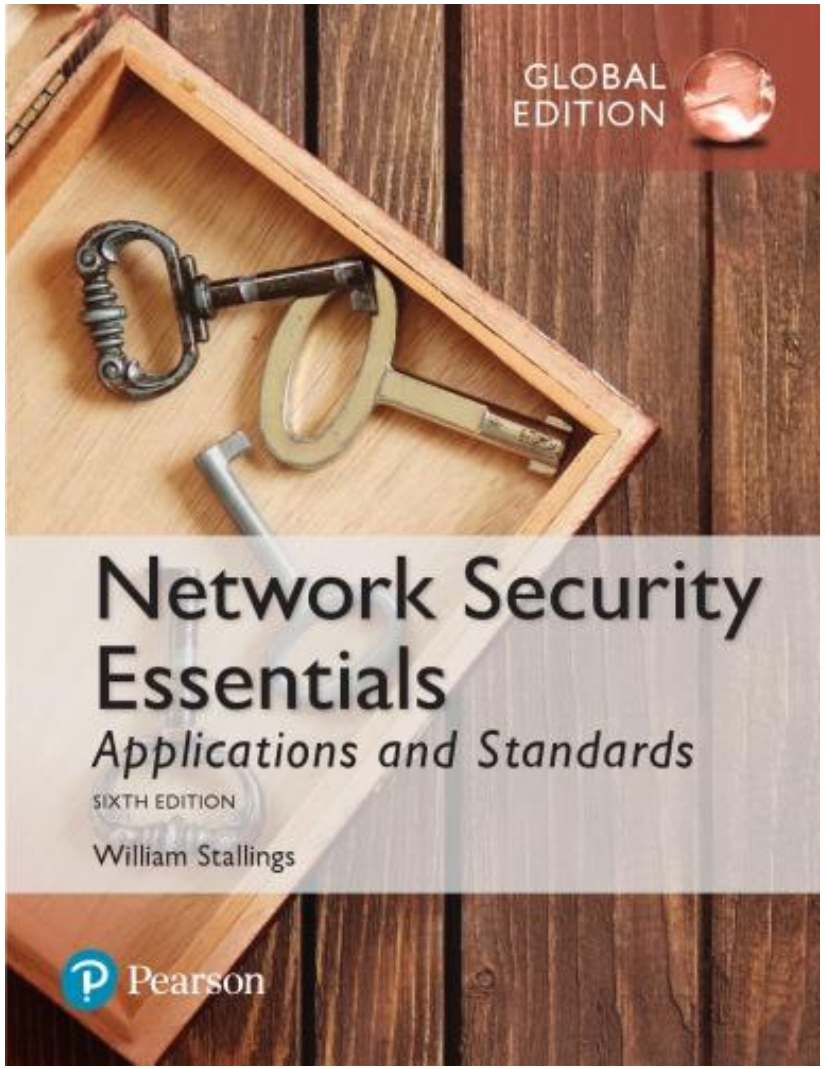
- Routing Protocols
  - Network as a Graph
  - Costs of the links
  - Issues with Static Routing
- Distance Vector Routing
  - Explained with an Example
  - Summary
  - Recovery from Node or Link failures
- GATE 2023: Three Networking Questions

Textbooks

Textbook 1

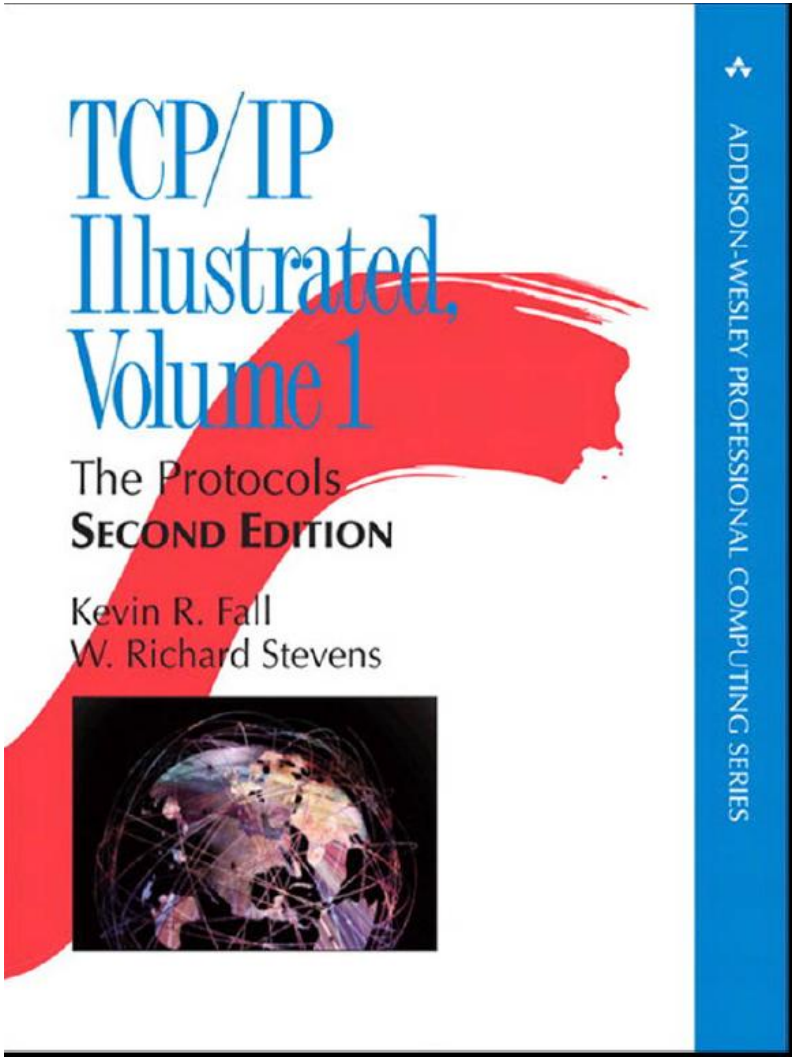


Textbook 2



References

Ref 1



Ref 2

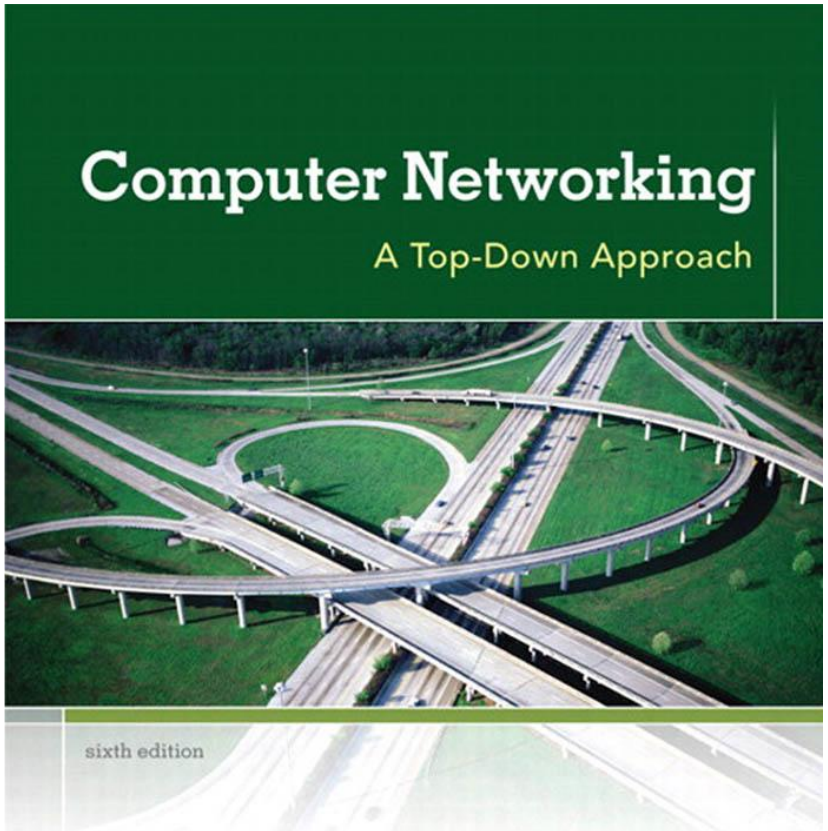
TCP Congestion Control: A Systems Approach

TCP Congestion Control: A Systems Approach

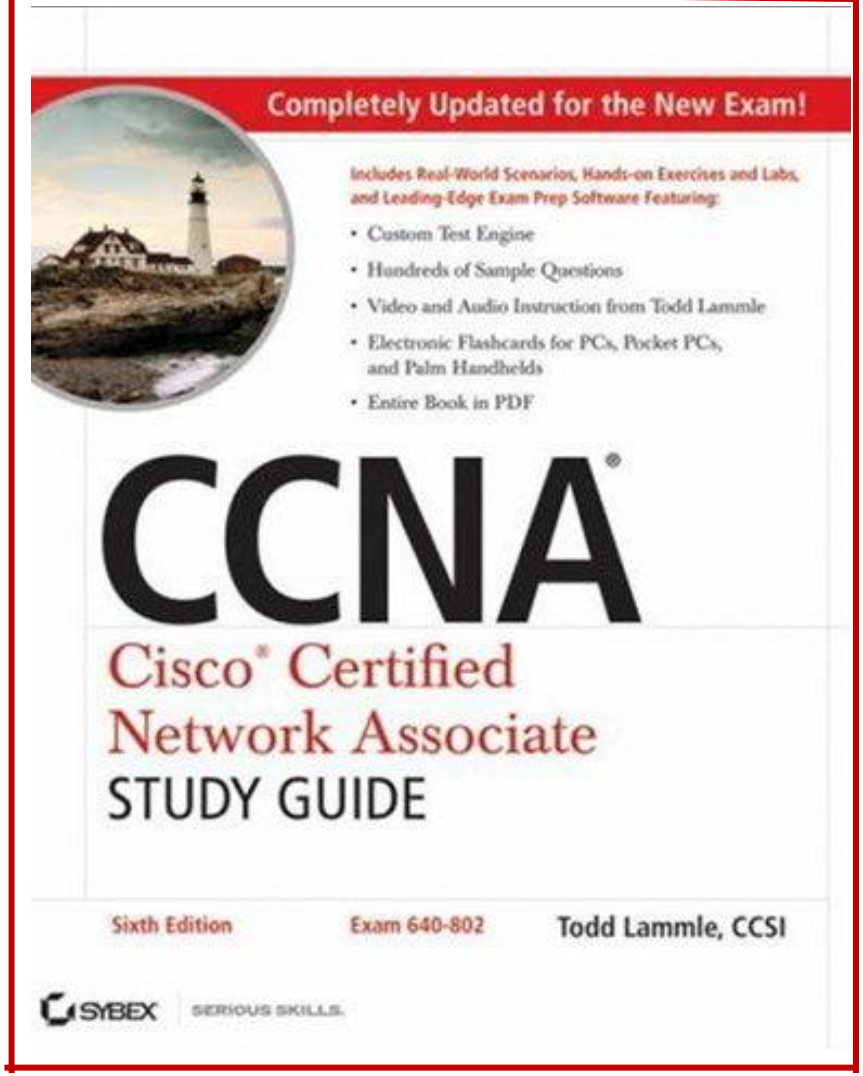
Peterson, Brakmo, and Davie

References

Ref 3



Ref 4



References

Ref 5