Session 5B

DVR: Solutions to Count to Infinity

Mouli Sankaran

Session 5B: Focus

- Distance Vector Routing Algorithm
- **Problem 1**: Recovery from a Link Failure
- Routing Loops and Remedies
  - Count to infinity
  - Split horizon
  - Poison reverse
  - Holddown timers

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

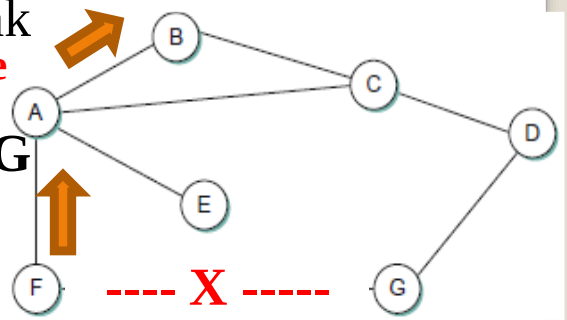
Distance Vector Routing
Link Failure

Problem 1: Link Between G and F has failed
(Reconstruct convergence using DVA)

1. Which are the routers come to know of the failure of the link shown below? **ANS1: Both F and G**

Assume, both take corrective action simultaneously.

2. What are the **stable configurations** on the **Routers F** and **G prior** to this **link Failure**?



1. Original Converged RT at node F

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

2. Original Converged RT at node G

Destination	Cost	Next Hop
A	2	F
B	3	F
C	2	D
D	1	D
E	3	F
F	1	F

Q3: Are there **any paths to any node** which are at **equal distance** from the **Routers F** and **G, prior** to **link failure**?

ANS3: Distance between G and B, is 3, both via F and D, from G.

Let us assume path chosen by both B and G is via A and F.

Q4: **F passes** the **update** to **which node**?

ANS4: To node A. Now, give the new RT at node A.

3. Assume, **F detects** the **link failure**. **New RT** at **F** will be:

3. Changed RT at node F after Link down

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

4. Updated RT at node A based on F

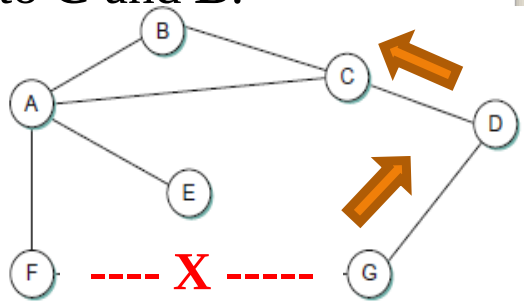
Destination	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2 to ∞	F to --

5. Updated RT at node B based on A

Destination	Cost	Next Hop
A	1	A
C	1	C
D	2	C
E	2	A
F	2	A
G	3 to ∞	A to --

Problem 1: Link to G from F has failed
(Reconstruct convergence using DVA) … contd.

6. Let us **follow** the **updates** from the **node G** to **D** and **then** to **C** and **B**:



6. Original Converged RT at node G

Destination	Cost	Next Hop
A	2	F
B	3	F
C	2	D
D	1	D
E	3	F
F	1	F

7. Changed RT at node G after Link down

Destination	Cost	Next Hop
A	2 to ∞	F to --
B	3 to ∞	F to --
C	2	D
D	1	D
E	3 to ∞	F to --
F	1 to ∞	F to --

5. Updated RT at node B based on A

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

8. Updated RT at node D based on G

Destination	Cost	Next Hop
A	2	C
B	2	C
C	1	C
E	3	C
F	2 to ∞	G to --
G	1	G

9. Updated RT at node C based on D (no changes)

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

10. Updated RT at node B based on C

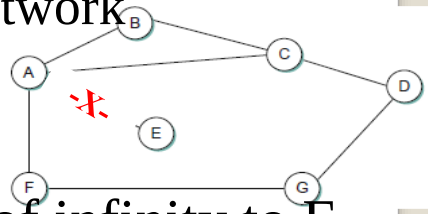
Destination	Cost	Next Hop
A	1	A
C	1	C
D	2	C
E	2	A
F	2	A
G	∞ to 3	-- to C

Note: You can see now that node B is connected to G via C, instead of via A. Similarly, A will also learn that it can reach G via C. All the nodes converge, once the updates from F and G reach all of them. All the nodes will be connected again.

Periodic update

Routing Loops

Distance Vector Algorithm: Count to infinity Problem (Example: Link between E and A fails) - Explained



- Unfortunately, slightly different circumstances can prevent the network from stabilizing.
- Suppose, for example, that the link between A and E goes down.
- In the next round of updates, **A** advertises to B, a distance of infinity to E, since B was reaching E via A, it updates its distance also to infinity, to E.
- Depending on the exact timing of events, the following might happen:
  - Before A's new update reaches C, C might give its periodic update to B a distance of 2 to E via A
  - Node B, upon hearing that E can be reached in 2 hops from C, it concludes that it can reach E in 3 hops and
  - B advertises this to A; node A concludes that it can reach E in 4 hops and advertises this latest update 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. **Note**: C accepts the increased cost from A because it was originally reaching E via A only.
- 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**.

Routing Loops: Description

- One of the main problems inherent with Distance Vector routing protocols is routing loops (route updates going into a loop).
- Routing loops occur in networks when old (bad) route information exists in a route table.
- The problem stems primarily from the inter-mixing of periodic route updates, sharing stale routes before receiving updates, triggered by link failures.
- Similar to what happened when the link to E failed from A.

Explained in the Previous Slide

- Because of the intervals between the periodic route updates, and the delay in the updates based on link failures, reaching all the nodes, routers may not learn about topology changes in a timely manner.
- They may be relying on outdated or incorrect route information.
- Slow convergence can result in routing loops, causing datagrams to bounce between routers endlessly if not detected, causing the routers to start a count to infinity.

Routing Loops: Remedies

Routing Loops: Remedies

- Routing protocols can take advantage of one or more loop avoidance mechanisms to minimize the impact a loop has on the network.
- These techniques, or combinations of these techniques, can minimize routing loops passing on incorrect routing information:

1. **Count to infinity**
2. **Split horizon**
3. **Poison reverse**
4. **Holddown timers**

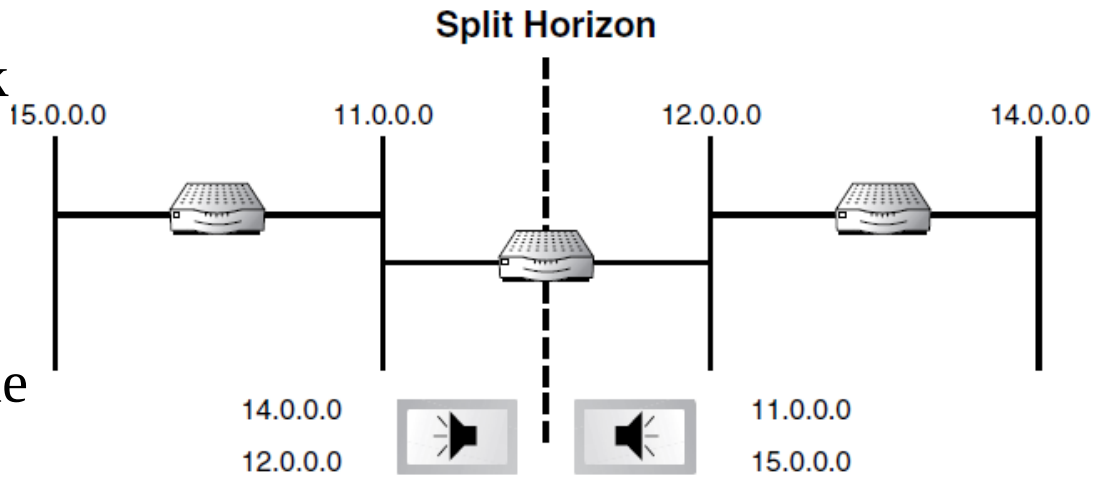
Note: Implementations vary based on vendor support and routing protocol.

1. Count to Infinity (max limit)

- Count to infinity is a loop avoidance mechanism that sets a maximum hop count value, that, when exceeded, equates to infinity hops (or destination unreachable).
- The maximum hop count for **RIP** is **15** and **IGRP** is **255**.
- Any value above this is considered **infinity** (**unreachable**).
- Distance-vector protocols limit the distance (hops) a datagram may traverse.
- If a route loop exists within the topology, the router automatically trashes the datagram when it exceeds the maximum hop set by the routing protocol running on the router.
- If for some reason datagrams continue to be forwarded after the maximum hop has been exceeded (infinity has been reached), then a fall back method, the TTL timer.
- Infinity value and TTL timer combination provides a viable remedy to most routing loops.

2. Split Horizon

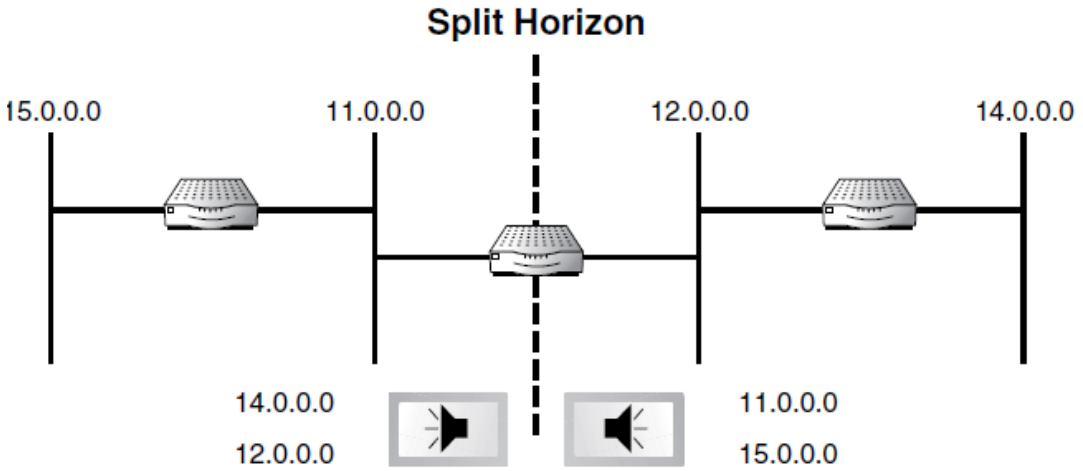
- Split horizon prevents information being sent back in the direction from which that information was received.
- When a change occurs in the network, routers only advertise that change in one direction.



- That means that they send the updates out to all other ports except the one from which it was learned.
- With split horizon, any router is the starting point.
- Split horizon sends only information learned from other ports.
- Split horizon never sends information out the same port it learned it from.

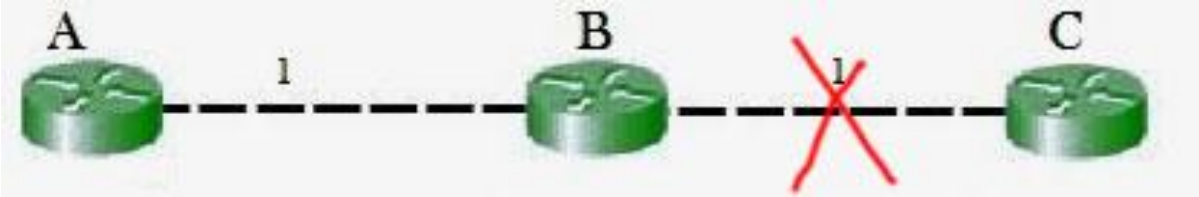
Split Horizon: Rules Explained

- The split horizon rule states that “route information learned through an interface may not be transmitted out that same interface”
- The middle router learns about networks 12.0.0.0 and 14.0.0.0 from the interface on the right.



- The router in the middle can only propagate this information out the opposite interface.
- It learns about networks 11.0.0.0 and 15.0.0.0 from the interface on the left and can only propagate this information out the opposite interface

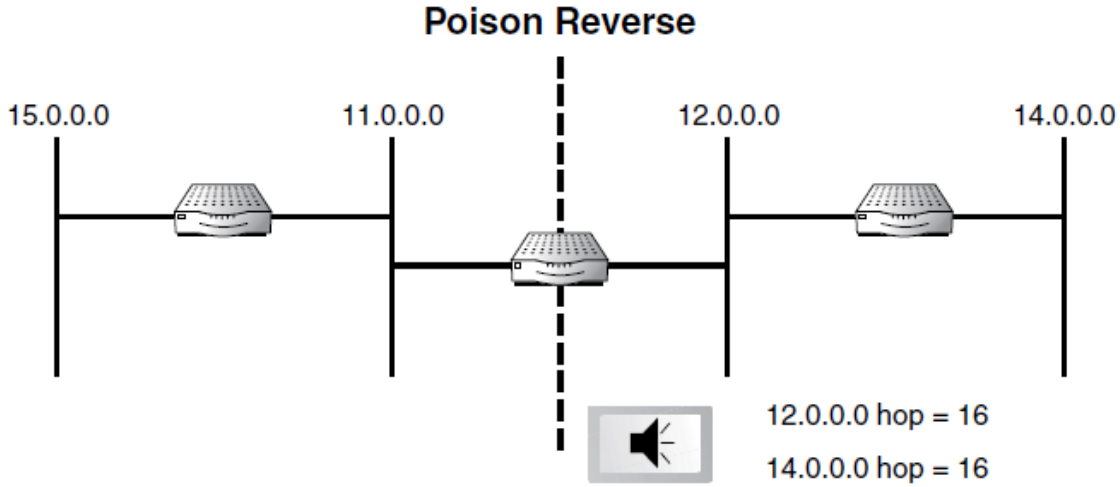
Split Horizon: Solution



- If the link between B and C goes down, and B had received a route from A , B could end up using that route via A.
- **A** would send the packet right back to B, creating a loop.
- But according to Split horizon Rule, Node A does not advertise its route for C (namely A to B to C) back to B.
- On the surface, this seems redundant since B will never route via node A because the route costs more than the direct route from B to C.
- But when the link between B and C fails, the update from A would have caused B to believe there is another longer route via A to C
- But, split horizon prevents such a confusion and C is considered unreachable quickly preventing routing loops.

3. Poison Reverse

- Poison reverse allows routers to break the split horizon rule by advertising information learned from an interface out the same interface



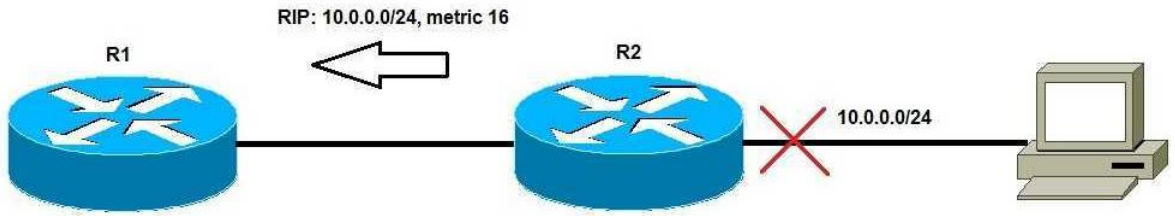
- However, it can advertise routes learned from an interface out the same interface with a 16-hop count, which indicates a destination unreachable, "**poisoning**" the route
- While the other routers slowly converge, the router maintains the poisoned route in its table and ignores updates from other routers about better routes to the poisoned network.
- Poison reverse prevents updates with inconsistencies from spreading.
- Poison reverse when implemented takes precedence over split horizon.

4. Holddown Timers

- Another technique typically used in combination with route poisoning is holddown timers.
- Holddown timers start as soon as a router receives an update from a neighbor indicating that an attached network has gone down.
- Until the timer elapses, the router ignores updates regarding this route from other routers unless it receives an update from the neighboring router that initially informed the network of the downed link.
- The timer stops if it receives a message from the neighboring router.
- At that point, the network is marked as reachable again and the route table is updated.
- Routers use holddown timers after they have learned that a route is unavailable to ensure that this route will not be mistakenly reinstated by an advertisement received from another router that has not yet learned about this route being unavailable.

4. Holddown Timers … Example

- Typically, this timer is greater than the total convergence time, providing time for accurate information to be learned, consolidated, and propagated through the network by all routers.



Note: **Default** value of **Holddown timer** is **180 seconds** in **RIP**

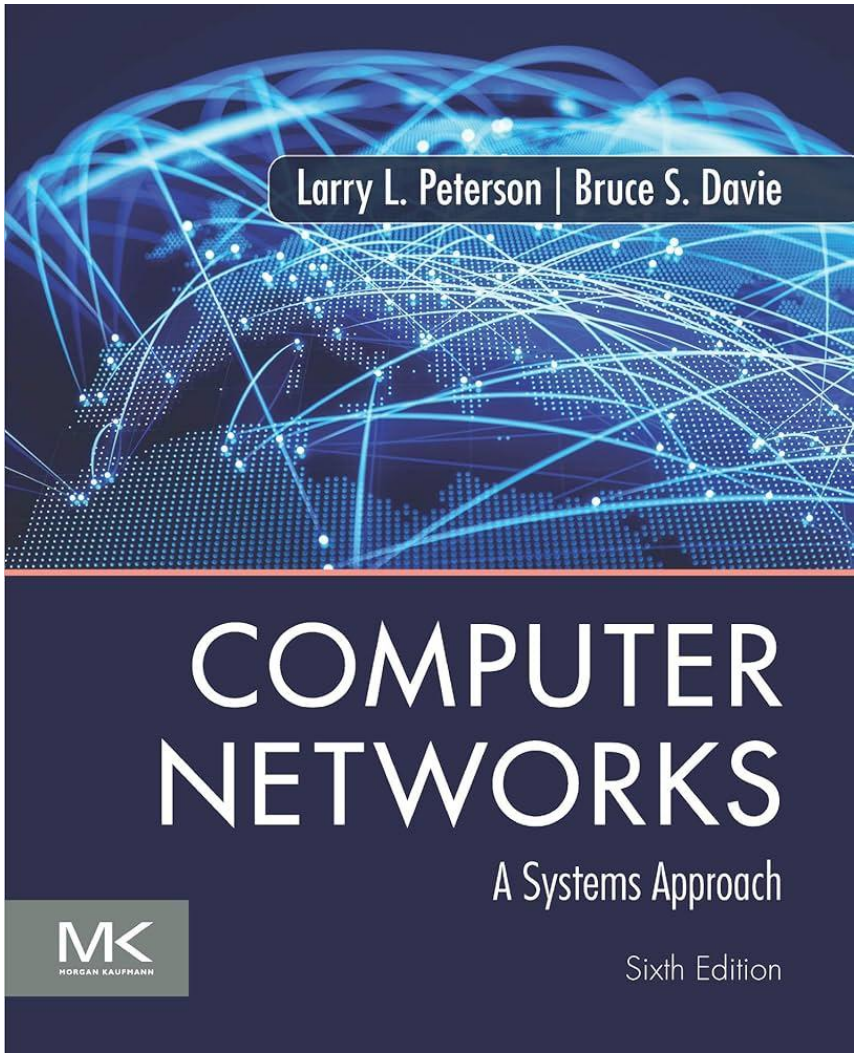
- We have a network of two routers. Both routers are running RIP and R2 has advertised the **10.0.0.0/24** network to R1. Consider what happens if the network fails:
  1. R2 advertises the **10.0.0.0/24** network with the infinitive metric (**16**) to R1, indicating that the network is no longer accessible.
  2. R1 receives the routing update, marks the route as unreachable, and starts the **holddown timer**.
  3. During the holddown period, R1 will not process any routing update about that route received from other routers. Only updates from R2 will be processed.

Session 5B: Summary

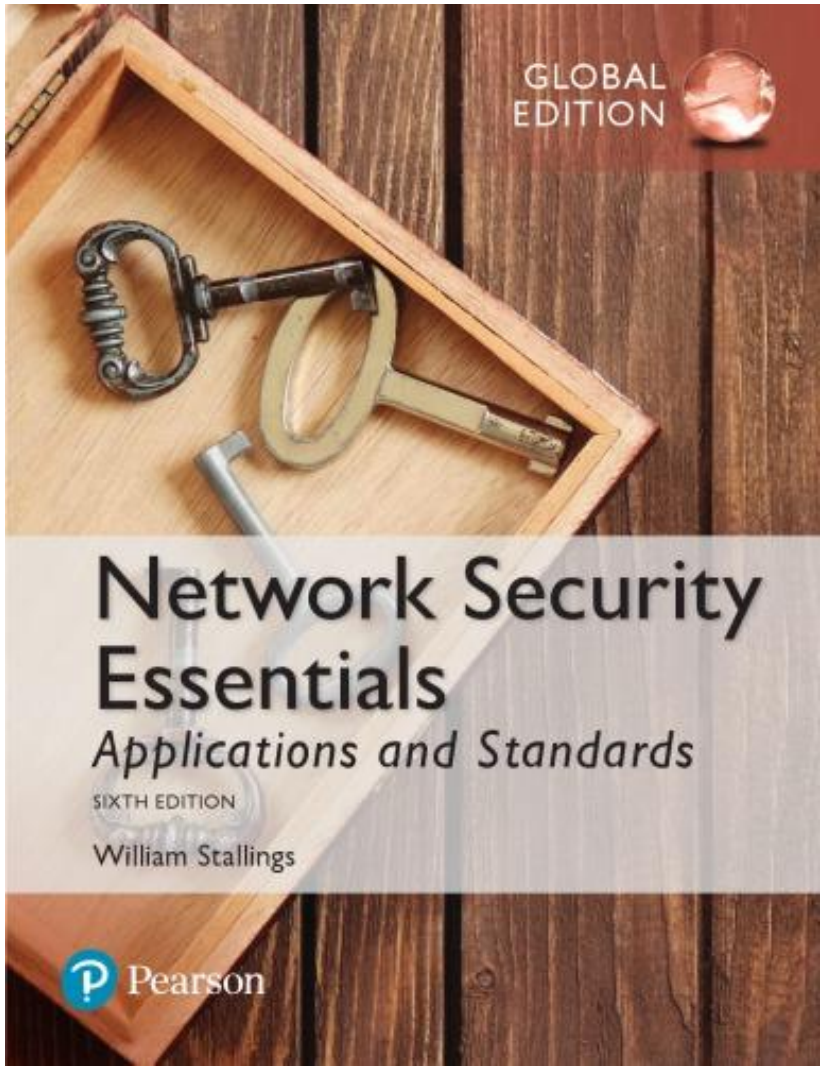
- Distance Vector Routing Algorithm
- **Problem 1**: Recovery from a Link Failure
- Routing Loops and Remedies
  - Count to infinity
  - Split horizon
  - Poison reverse
  - Holddown timers

Textbooks

Textbook 1

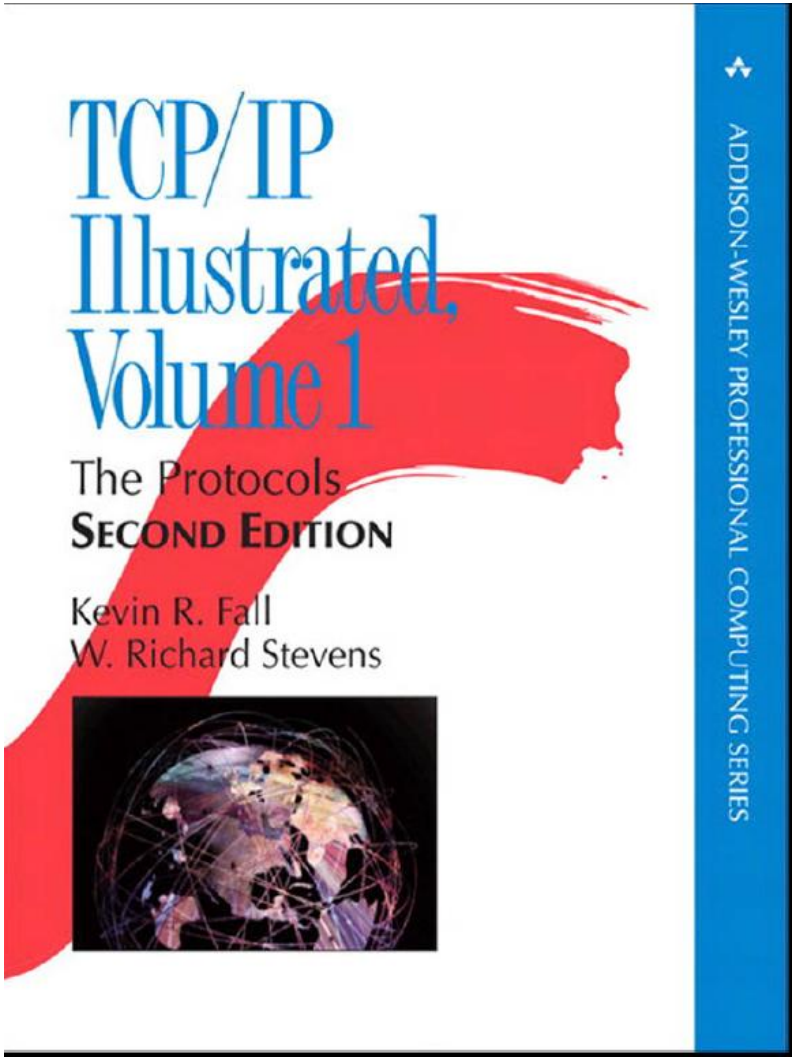


Textbook 2



References

Ref 1



Ref 2

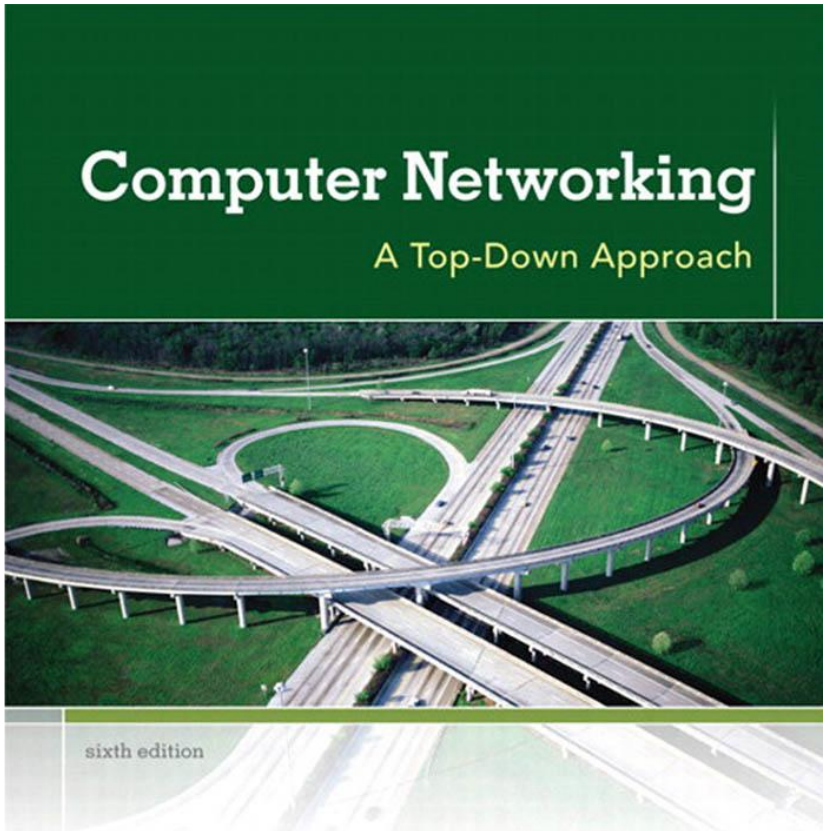
TCP Congestion Control: A Systems Approach

TCP Congestion Control: A Systems Approach

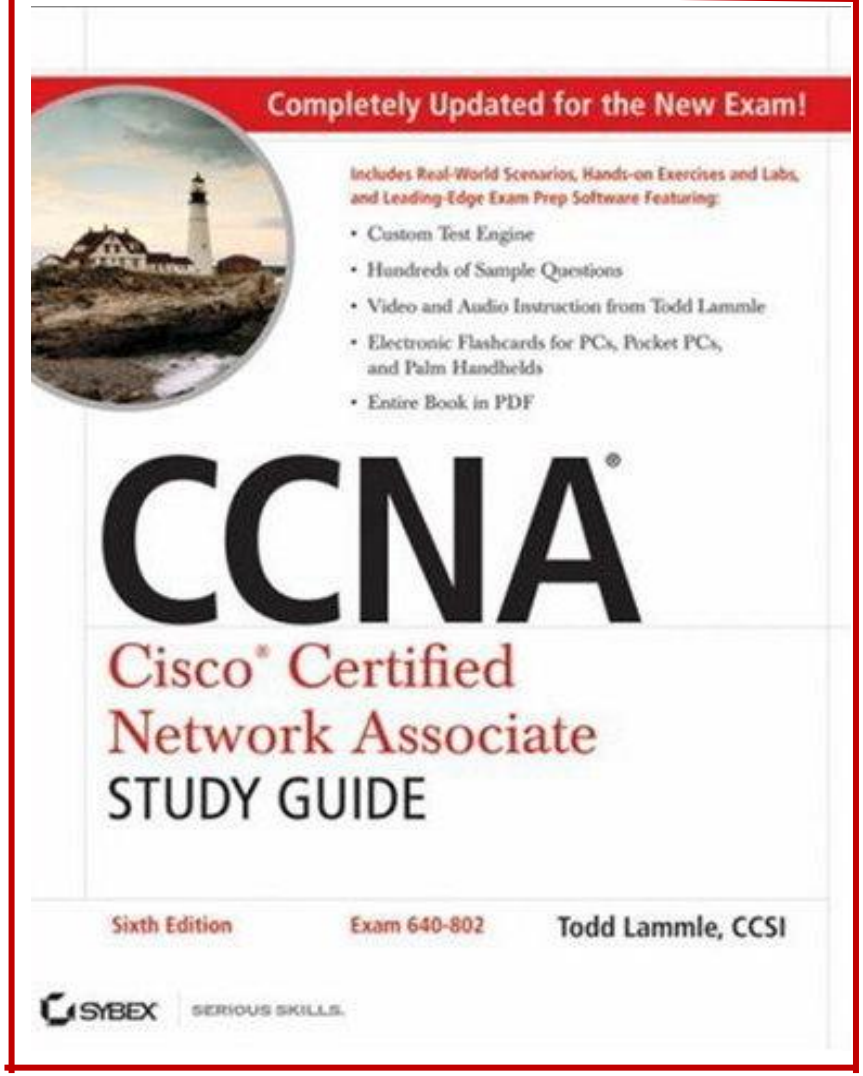
Peterson, Brakmo, and Davie

References

Ref 3



Ref 4



References

Ref 5