Communication Networks

Spring 2017/2018

Dr. Mohammad HAWA

By:

Mohammad
Abu Hashya.

* * *

By Mohammad Abu Hashya.

Communication-Networks

"Routing" Summary.

* Distinction between Forwarding & Routing:

- **Forwarding:**
  - taking a packet, looking at its destination address, consulting a table and sending the packet in a direction determined by the table.
  - it is simple, well-defined process performed locally @ a node.
- **Routing:**
  - it is the process by which forwarding tables are built.
  - it depends on complex distributed algorithms.

* Forwarding Table vs. Routing Table:

- **Forwarding Table:** ① used when a packet is being forwarded & so must contain enough info. to accomplish the forwarding function.

  ⇒ "This means that a row in the forwarding table contains the mapping from a network prefix to an outgoing interface & some MAC info., such as Ethernet address of the next hop."

  ② needs to be structured to optimize the process of looking up an address when forwarding a packet.

  ③ it may even be implemented in specialized hardware.

- **Routing Table:**

  ① it is a table that is built up by the routing algorithms as a precursor to building the forwarding table.

  ⇒ "it contains mapping from network prefixes to next hops, it may also contain info. about how this info was learned, so that the router will be able to decide when it should discard some info."

  ② Needs to be optimized for the purpose of calculating changes in topology.

  ③ it is rarely to be implemented in specialized hardware.

* All abbreviations in this handout provided as follows:

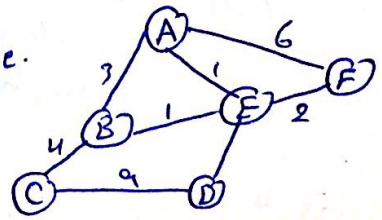
- IGP ≡ Interior Gateway Protocols.
- TTL ≡ Time To Live.
- RIP ≡ Routing Information Protocol.
- BSD ≡ BerKeley Software Distribution.
- LSP ≡ LinK State PacKet.
- OSPF ≡ Open Shortest Path First.
- IETF ≡ Internet Engineering TasK Force.
- LSA ≡ LinK state Advertisement.
- TOS ≡ Type Of Service.

* The protocols studied in this handout called:
Intradomain Routing Protocols or IGPs.

* NetworK as a Graph:

* the basic problem of routing: to find the lowest cost path between two nodes, where the cost of apath = the sum of the costs of all the edges that make up the path.

* Consider the following NetworK:



one can calculate the shortest paths & loading them into non volatile storage.
⇒ such a static approach has several shortcomings:

- It doesn't deal with node or link failures.
- It doesn't consider the addition of new nodes or links.
- It implies that edge cost can't change.

⇒ for these reasons: routing is achieved in most practical networKs by running routing protocols amoung the nodes.

* Why this has been such a rich field of research & development?
one of the main reasons is the distributed nature of routing algorithms.

Example:
2-protocols @ one instant have different ideas about the shortest path to some dest. each one will think that the other one is closer, so it decide to send packets & won't stop until the discrepancy is resolved.

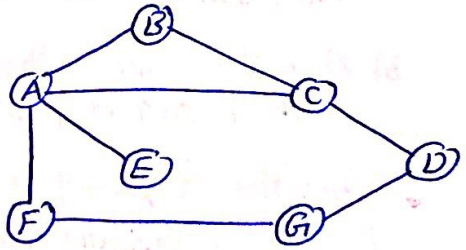
* Two main classes of routing protocols: ① distance vector. ② Link state.

* Distance-Vector (RIP):
* another common name for this class: "Bellman-Ford".

* procedure:
- Each node constructs a one-dimensional array (a vector) containing the "distances" (costs) to all other nodes & distributes that vector to its immediate neighbors.
- The starting assumption for distance-vector routing is that each node knows the cost of the link to each of its directly connected neighbors.

Example:

- in this ex. the cost of each link is set to 1, so that a least-cost path is simply the one with the fewest hops.



info. stored @ 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

- Now each node send msg to its neighbors contain the list of distances (direct connected neighbor).

F tells A it can reach G by cost 1. A knows that it reah F by cost 1 so it adds these costs to get the cost of reaching G1 by means of F. (cost=2) which is less than ∞ (better) any higher cost will be ignored if a new low cost discovered.

⇒ Initial routing table at Node A:

Dest.	cost	Next Hop
B	1	B
C	1	C
D	∞	—
E	1	E
F	1	F
G	∞	—

*Definition:

• Convergence: The process of getting consistent routing information to all the nodes.

⇒ Final routing Table @ node A:

Dest.	cost	Next Hop.
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

⇒ final Distances stored @ each node:

info. stored @ Node	Dist. 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

*We Note that there is two different circumstances under which a given node decides to send a routing update to its neighbors:

1 Periodic Update:

a) 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.

b) It makes sure that they Keep getting info. that they may need if their current routes become unviable.

•Note: the frequancy of these periodic updates varies from protocol to protocol (typically several seconds to several minutes).

2 Triggered Update:

occur when a node notices a link failure or receives an update from one of its neighbors, causing a change in one of the routes in routing table, this change lead to send an update to it neighbors which may lead to a change in their table, causing them to send an update to their neighbors.

* What happens when a link or node fails?

⇒ the nodes which notice first, send new lists of distances to their neighbors & normally the system settles down fairly quickly to a new state.

* How a node detects a failure?

first approach:

① A node continually tests the link to another node by sending a control PKt & seeing if it receives an acknowledgment.

second approach:

② A node determines that the Link (or the node at the other end of the link) is down if it doesn't receive the expected periodic routing update for the last few update cycles.

• Example on failure: 

⇒ if F detects that its link to G has failed:

- F sets its new distance to G (∞) & passes that info. to A.
- A knows that its 2-hop path to G through F, so A set its distance to G (∞).
- the next update from C, A would learn that C has a 2-hop path to G. , so A would know that it could reach G in 3 hops through C. (<∞).
- A update its table.
- when it advertises this to F, F learn that it can reach G through A at a cost of 4. (<∞).
- the system again would be stable.

* Count to ∞ problem:

⇒ Example: 

if the link A to E goes down:

- A advertises a distance (∞) to E, B&C advertise a distance (2) to E.
- Node B, upon hearing that E can be reached in 2-hops from C, concludes that it can reach E in 3-hops & advertises this to A.
- A concludes that it can reach E in 4 hops & advertises this to C, concludes that it can reach E in 5-hops & so on.

"this will reach ∞ without knowing E is unreachable".

⇒ Solution: use small # as an approximation of ∞.

Ex. Decide that the MAX # of Hops ≤ 16 "here 16 represent the ∞".

* Techniques to improve the time to stabilize routing:

1. Split Horizon:

Ex. if B has the route (E, 2, A) in its table, then it knows it must have learned this route from A, whenever B sends routing update to A, it does NOT include the route (E, 2) in the update.

2. Split Horizon with poiso reverse:

Ex. B sends that route to A, but it puts -ve info in the route to ensure that A won't use B to get to E, B sends (E, ∞) to A.

** The problem of both techniques: They only work for routing loops which involve 2-nodes.

※ Implementation:

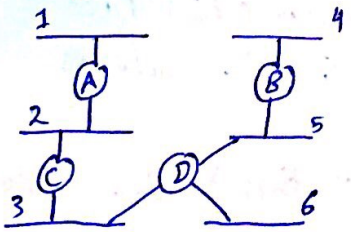
- The routine that updates the local node's routing table based on a new route is given by "MergeRoute".

* RIP:

- RIP is the canonical example of a routing protocol built on the distance-vector algorithm.
- RIP is fact a fairly straightforward implementation of distance-vector routing.
- RIP it always tries to find the minimum hop route.

Note: routers running RIP send their advertisements every 30 seconds.

Example: "Network running RIP"



here C would advertise to A that it can reach Networks 2 & 3 (0 cost), & to 5&6 in (1 cost), & to 4 in (2 cost).

* RIPv2:

for example: A learns from B that Network "X" can be reached at a lower cost via B than via the existing next hop in the routing table. , A updats the cost & next hop info for the Network Number.

*what is the reason for the Family part of advertisements?
since it supports multiple address families.

※ Link State (OSPF):

⇒ The basic idea behind it: Every node knows how to reach its directly connected neighbors, and if we make sure that the totality of this knowledge is disseminated to every node, then every node will have enough knowledge of the network to build a complete map of the network.

⇒ Link-state rely on 2-mechanisms:
1. Reliable dissemination of link-state info.
2. The Calculation of routes from the sum of all the accumulated link-state knowledge.

※※ Reliable Flooding:

Def.: is the process of making sure that all the nodes participating in the routing protocol get a copy of the link-state info. from all other nodes.

- LSP: it is the updated PKt created by each node.
- LSP contain the following info.:
  1) The ID of the node that created the LSP.
  2) A list of directly connected neighbors of that node, with cost of the link to each one.
  3) A sequance Number.
  4) A time to Live for this PKt.

⇒ the first 2-items: needed to enable route calculation.

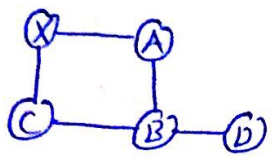
⇒ the last 2-items: used for the process of flooding the PKt to all nodes reliable.

※ How flooding works?

- the transmission of LSPs between adjacent routers is made reliable using acknowledgments & retransmissions just as in the reliable link-layer protocol. However several more steps necessary to reliably flood an LSP to all nodes in a network.

※ Consider a node X, receives a copy of LSP organized at node Y
↳ X check if it has a copy of LSP from Y. if not, it stores the LSP
if Yes, it compares the seq. #'s, if the new LSP has larger seq.# it is assumed to be the most recent, & that LSP stored, replacing the old one.
if it is smaller or equal seq.# the new LSP would be discarded.
Now X send the new LSP to all neighbors except the one that it received from
& same process for all other nodes.

• Example:



⇒ LSP arrived @ node X, it send LSP to A & C then B will receive one copy & ignore the other one (will receive which arrive first), then D will receive the LSP & the process will be completed.

* Each node generates LSPs under 2-circumstances:

[1] The expiry of a periodic timer. [2] A change in topology can cause a node to generate a new LSP.

* Design Goals of link-state protocols flooding mechanism:

① the newest info must be flooded to all nodes as quickly as possible.
② old info. must be removed from the network & not allowed to circulate.
③ Minimize the total amount of routing traffic that is sent around the network.

* LSPs carry a time to live, used to ensure that old link-state info. is eventually removed from the Nework.

** Route Calculation:

* Dijkstra's Algorithm:

$L(i,j)$ ⇒ non negative cost.
$L(i,j) = \infty$ if No edge connects i & j.
M ⇒ is the set of nodes incorprated by the algorithm.
$C(n)$ ⇒ cost of the path from s to n.

* Algorithm defined as follows:

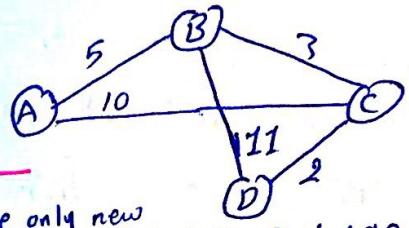
$M = \{s\}$
for each n in $N - \{s\}$
$C(n) = L(s,n)$
while $(N \neq M)$
$M = M \cup \{w\}$ such that $C(w)$ is the minimum for all w in $(N-M)$.
for each n in $(N-M)$.
$C(n) = MIN(C(n), C(w) + L(w,n))$

* Forward Search Algorithm:

⇒ each switch compute its routing table directly from LSPs it has collected using a realization of Dijkstra's Algorithm.

* each switch contain 2-lists: [1] Tentative. [2] Confirmed.
each of these lists contain a set of entries of the form (Dest., Cost, NextHop).

* Best illustration for the algorithm by an example:



step	Confirmed	Tentative	comments
①	(D,0,-)	—	since D is the only new member of the confirmed list, look @ its LSP.
②	(D,0,-)	(B,11,B) (C,2,C)	D's LSP says we can reach B through B at cost 11, which is better, put it on the tentative list, same for C.
③	(D,0,-) (C,2,C)	(B,11,B)	put lowest cost member of Tentative(C) onto Confirmed list, Next, examine LSP of newly confirmed member(C).
④	(D,0,-) (C,2,C)	(B,5,C) (A,12,C)	cost to reach B through C is 5 so replace (B,11,B). C's LSP tells us that we can reach A at cost 12.
⑤	(D,0,-) (C,2,C) (B,5,C)	(A,12,C)	Move lowest-cost member of Tentative (B) to confirmed, then look @ its LSP.
⑥	(D,0,-) (C,2,C) (B,5,C)	(A,10,C)	since we can reach A @ cost 5 through B, replace the Tentative entry
⑦	(D,0,-) (C,2,C) (B,5,C) (A,10,C)	—	move lowest-cost member of Tentative (A) to Confirmed, and we are all done.

* Properties of Link-state routing algorithm:

1. It has been proven to stabilize quickly.
2. It does not generate much traffic.
3. It responds rapidly to topology changes or node failures.

* A summary for the difference between distance-vector & link-state algorithms:

⇒ In distance-vector: each node talks only to its directly connected neighbors, but it tells them everything it has learned (distance to all nodes).

⇒ In link state: each node talks to all other nodes, but it tells them only what it knows for sure (only the state of its directly connected links).

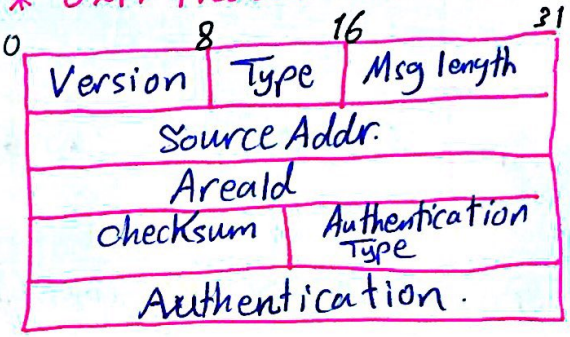
※ The Open Shortest Path First Protocol (OSPF):

* "Open" ⇒ refers to the fact that is open, nonproprietary standard, created under the auspices of the IETF.
* "SPF" ⇒ comes from an alternative name for Link-state routing.

* Features of OSPF:

1. Authentication of routing messages.
2. Additional Hierarchy.
3. Load Balancing.

* OSPF Header Format:



* OSPF msg types:

- Type 1 : is the "Hello" msg ⇒ a router send to its peers to notify them that is still alive.
- Remaining Types used to : request, send, and acknowledge the receipt of Link-state msgs.

* Two Types of Advertisements:

[1] Advertise one or more of the networks that are directly connected to that router.

[2] Router that is connected to another router by some link must advertise the cost of reaching that rater over the link.

* Types of LSA :

- Type 1 LSAs: advertise the cost of links between routers.
- Type 2 LSAs: used to advertise networks to which the advertising router is connected
- Other Types : Used to support additional hierarchy.

* Continue ... OSPF link-state advertisement:

* In a type 1 LSA: Link state ID & the advertising router field are identical. each carries 32-bit identifier for the router that created this LSA.

* Requirements while a number of assignment strategies may be used to assign ID:

[1] It is essential to be unique in the routing domain.

[2] Given router consistently uses the same router ID.

⇒ one way to pick a router ID that meets these requirements would be to pick the lowest IP address among all the IP addresses assigned to the router.

* The LS Checksum: used to verify that data has NOT been corrupted. It covers all fields in the PKt except LS AGE.

* Link Data: used to disambiguate among multiple parallel links.

* The Metric: it is the cost of the link.

※ Metrics:

⇒ The considered approach which "assign a cost of 1 to all links – the least-cost route will then be the one with the fewest hops", has several Drawbacks:

1. It doesn't distinguish between links on a latency basis. a satellite link with 250-ms latency looks just as attractive to the routing protocol as a terrestrial link with 1-ms latency.
2. It doesn't distinguish between routes on a capacity basis, making a 9.6 Kbps link look like just as good as a 45-Mbps link.
3. It doesn't distinguish between links based on their current load, making it impossible to route around overloaded links.

* * * * * * * * *

Best of Luck 11 Best of Luck.