COMP3721 Week Thirteen Lab Synopsis

Routing Protocols

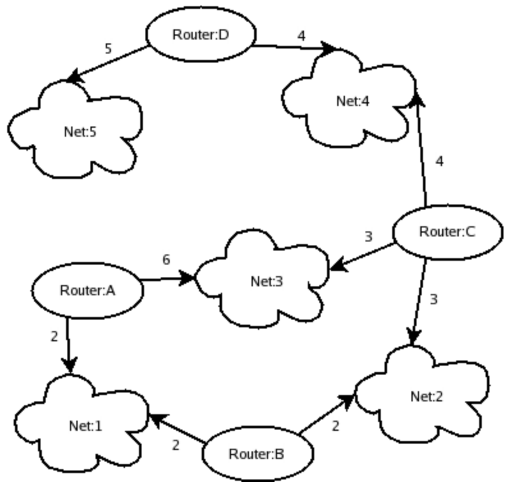
- Determine which one, of many links, a router will forward a packet via to move it closer to its destination within the internetwork
- Routing protocol allows a router to build a routing table
- Routing table consists of three columns:
  - Network (the destination network id) – a destination network address is part of a particular destination network (see week twelve)
  - Cost to get a packet to the destination network
    - in complex networks, multiple paths often exist to a particular destination
    - routing then tries to minimize one or more of the following when picking a routing path:
      - financial cost
      - delay
      - number of intermediate hops
      - ...
    - the particular costing criteria used is called the routing metric
  - Next hop – if the router is not directly connected to the destination network, then it must send the packet forward indirectly via another router.

Protocol Classes

- There are two major classes of routing protocol, distance vector and link state

Distance Vector Protocols

- Example protocol: RIP (Routing Information Protocol)
- General principle: every router shares all its information about the network with its neighbouring routers (the heresay principle)
- Eventually, each router will learn about all other networks via this intercommunication



- consider the network above, the initial routing tables (before any router intercommunication) for routers A, B and C are:

Router A Initial Table

Network	*Cost*	*Next Hop*
3	6	-
1	2	-

Router B Initial Table

Network	*Cost*	*Next Hop*
1	2	-
2	2	-

Router C Initial Table

Network	*Cost*	*Next Hop*
2	3	-
3	3	-
4	4	-

- notes on the diagram:
  - Net:3 would refer to a network with a network id of 3. If IPv4 addressing were being used, each subnet addresses would be used in each case, i.e. Net:1 == 192.168.4.0/24, Net:2 == 192.168.3.0/24, ...
  - the routers are members of each connected network despite being drawn as independent nodes
    - for example, Net:1 might be an fast-Ethernet network where the devices are interconnected by a switch. Then both router A and router B have fast-Ethernet cards and which are connected to that switch. Similarly, they have network addresses within the Net:1 address range.
    - The cost to send something from router A to anyone in Net:1 is shown to be 2. Then the cost to send something from router A to router B is 2 (not 4) as B is part of Net:1.
      - The graph can be supplemented to reflect this by adding 0-cost edges connecting networks to (connected) routers. This is shown in the section on link-state protocols
  - The cost to send into a given network may not be symetrical –

for example, router C has a cost of 3 to send to Net:3 whereas router A has a cost of 6.

    - this difference can arise for a variety of reasons. For example, if the cost metric being used were expected delay and router C had experienced less collisions (than router A) when trying to send on the Net:1 Ethernet network, then it could have a lesser expected delay

- Each of these routers would regularly share their existing routing table with its neighbouring routers
  - for now consider B sharing its table with A and C (not D as D is not a neighbour)
  - the distance vector sent out by B does not include the next hop column

B: Distance Vector

Network	*Cost*
1	2
2	2

  - when A receives this vector, it can now incorporate it into its own routing table
    - that is, packet can be sent to networks 1 and 2 via router B
    - but router A must first consider the cost of getting a packet to router B
    - the router B distance vector would have arrived via Net:1 and A's expected cost through Net:1 is 2, so A must add to to each of the costs in the router B distance vector:

B: Distance Vector

Network	*Cost*
1	2
2	2

+

A to B
2
2

=

A's cost via B

Network	*Cost*
1	4
2	4

  - Router A already has a cheaper path to Net:1 – it costs 2 to send directly to a destination address in Net:1 versus the alternative of sending the packet through Net:1 to router B who would then send the packet back into Net:1 to the destination (for a total cost of 4)
    - So the Net:1 entry in A's table is not replaced – the router is only looking for cheaper paths to networks
  - Router A currently does not have a router to Net:2 and so adds the cost 4 path to its table

Router A Updated Table

Network	Cost	Next Hop
3	6	-
1	2	-
2	4	B

- through a series of distance vector exchanges involving all the routers, routers A, B and C should eventually end up with stable tables as follows:

Router A Stable Table

Network	Cost	Next Hop
3	6	-
1	2	-
2	4	B
4	8	B
5	13	B

Router B Stable Table

Network	Cost	Next Hop
1	2	-
2	2	-
3	5	C
4	6	C
5	11	C

Router C Stable Table

Network	Cost	Next Hop
2	3	-
3	3	-
4	4	-
1	5	A
5	9	D

  - Note that C could alternately route to Net:1 through router B at an equal cost of 5 – the choice is arbitrary (as the metric says the cost is the same)
- full knowledge of the internetwork spreads fairly quickly in distance vector protocols (good news travels fast)
- distance vectors are easily transmitted to all neighbouring routers through broadcasts to each router link
- unfortunately, changes in (inter-)network topology are slow to propagate (bad news travels slowly)
  - consider if router C’s link into network 4 goes down
  - router C changes it’s cost to Net:4 to ∞ (infinity)
  - router C sends out it’s updated distance vector to both B and A
  - unfortunately, B simultaneously sends its (as of yet unchanged) distance vector out to A and C
    - based on B’s distance vector, C now believes it can route packets to network 4 via B
    - if A will also start to route packets to network 4 via B
    - B will receive C’s new distance vector and update its table, but will also receive subsequent updates from A (for example) which will lead it to believe Net:4 is still reachable
    - Generally, a routing loop will emerge, as each router

believes one of the other routers has a path to the inaccessible network – the cost to that network will slowly increment as each router adds the cost of the intermediate hop. This problem is thus called the count-to-infinity problem.

- This is just one of the problems with distance vector protocols
  - this type of problem is generally unavoidable where the protocol relies on second-hand information to build a routing table (as do distance vector protocols)

Link State Protocols

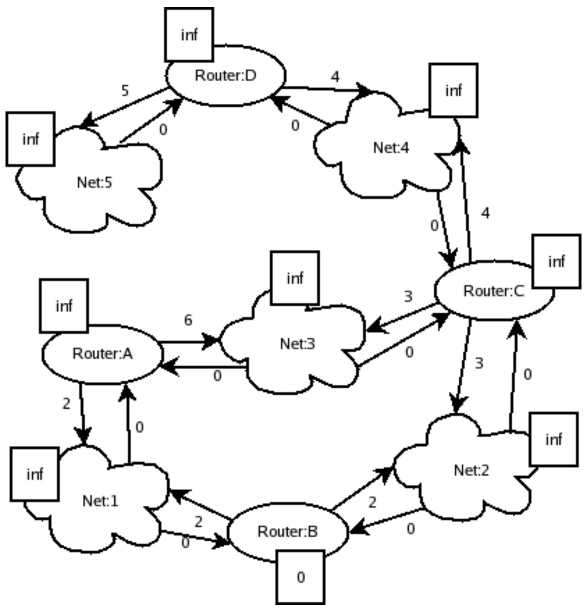
- Example protocol: OSPF (open shortest path first)
- General principle: every router shares the cost to/state of its direct links with all other routers
  - Broadcasting capability does not help to distribute link-state information – it must cross multiple routers
  - Link-state advertisements must then be distributed through more complex multi-casting techniques
- Each router can build a complete view of the inter-network – that is it knows all links and all networks from all of the link-state advertisements
- Given a complete view of the inter-network, the shortest path from the router to any other location can be solved
  - this problem is known as the single source shortest path problem
  - the Dijkstra algorithm is the best known solution to this problem

Dijkstra's Algorithm

- The inter-network is represented as a graph:
  - networks and routers are treated as vertices (or nodes)
  - routers are connected to their associated networks via edges
- The algorithm can be expressed in three steps:
  1. make the cheapest non-permanent node permanent
  2. relax the distance to each neighbour of the the above node
  3. repeat until all nodes are permanent
- Permanent nodes are those whose shortest path is already

known

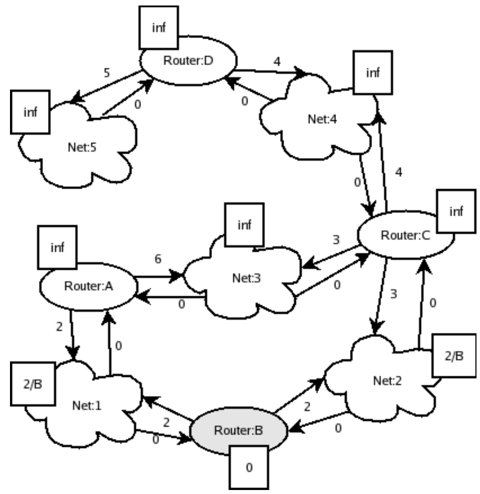
- Relaxation involves looking at the cumulative cost to get to a neighbouring node – if a cheaper route is found, the path to a node is updated
- Initially, the only known distance is that to the source node (0). All other nodes start out at an infinite distance. For the example below, we will assume router B is solving the shortest path from it to every other network



- For the above network, there are nine nodes – four routers and five networks
  - therefore, the Dijkstra algorithm will require nine passes to solve the shortest path problem
  - once done, a routing table can easily be generated

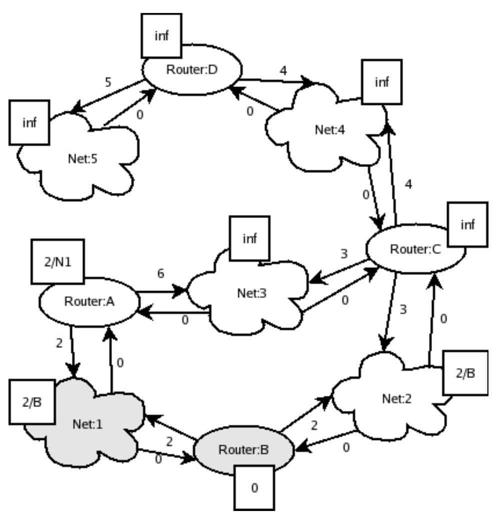
1. B becomes permanent
   1. B has a cost of 0, every other node has an infinite cost
   2. the total cost to Net:1 is 0 (to B) + 2 (from B to Net:1) = 2; this is better than infinity so the node is updated. Similarly, Net:2 is updated with a cost of 2 (via B).

Pass	***Node***	***Cost***
1	B	0



2. Net:1 becomes permanent
    1. the choice between Net:1 or Net:2 is arbitrary here, both are equally cheap
    2. Router:A has an updated cost of 2 (to Net:1) + 0 (from Net:1 to A) = 2

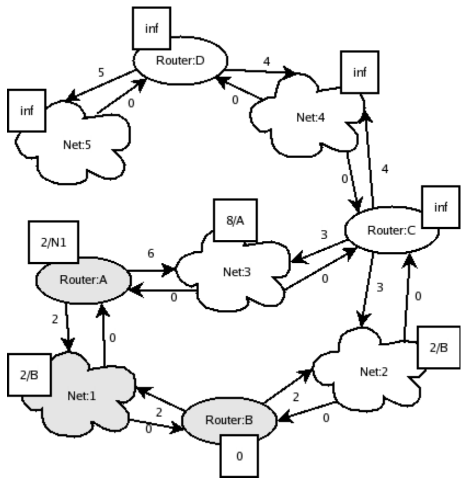
Pass	***Node***	***Cost***
1	B	0
2	Net:1	2/B



3. Router:A becomes permanent
    1. again, the choice between Router:A and Net:2 is arbitrary here, both are equally cheap

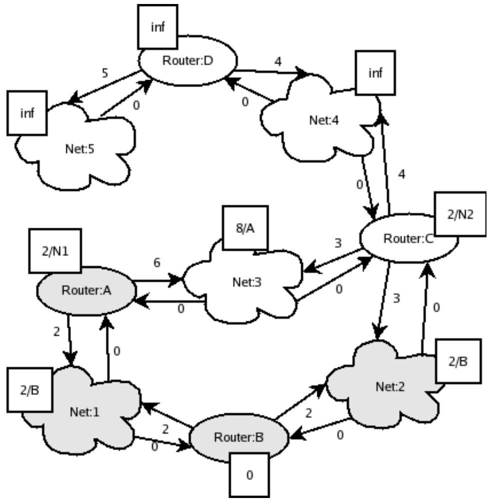
2. Net:3 has an updated cost of 2 (to Net:1) + 0 (from Net:1 to A) = 2

Pass	*Node*	*Cost*
1	B	0
2	Net:1	2/B
3	A	2/Net:1



4. Net:2 becomes permanent
    1. Net:2, at a cost of 2 is the cheapest non-permanent node
    2. Router:C has an updated cost of 2 (to Net:2) + 0 (from Net:2 to C) = 2

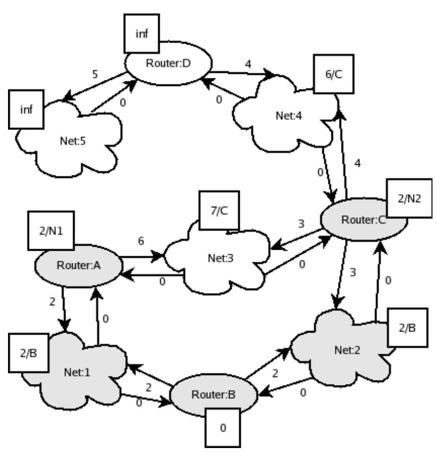
Pass	*Node*	*Cost*
1	B	0
2	Net:1	2/B
3	A	2/Net:1
4	Net:2	2/B



5. Router:C becomes permanent
    1. Router:C, at a cost of 2 is the cheapest non-permanent node
    2. Net:4 has an updated cost of 2 (to Router:C) + 4 (from Router:C to Net:4) = 6.

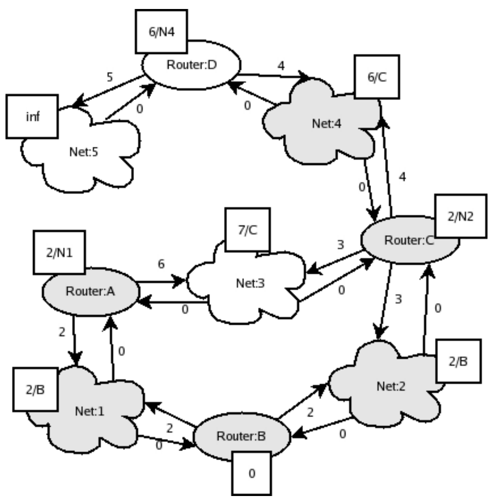
       Net:3 has an updated cost of 2 (to Router:C) + 3 (from Router:C to Net:3) = 7. If the edge from C to Net:3 had a cost of 5, then the path to Net:3 would have remained unchanged.

Pass	***Node***	***Cost***
1	B	0
2	Net:1	2/B
3	A	2/Net:1
4	Net:2	2/B
5	C	2/Net:2



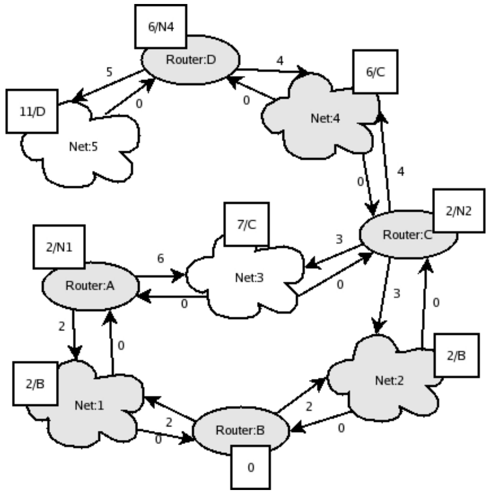
6.

Pass	***Node***	***Cost***
1	B	0
2	Net:1	2/B
3	A	2/Net:1
4	Net:2	2/B
5	C	2/Net:2
6	Net:4	6/C



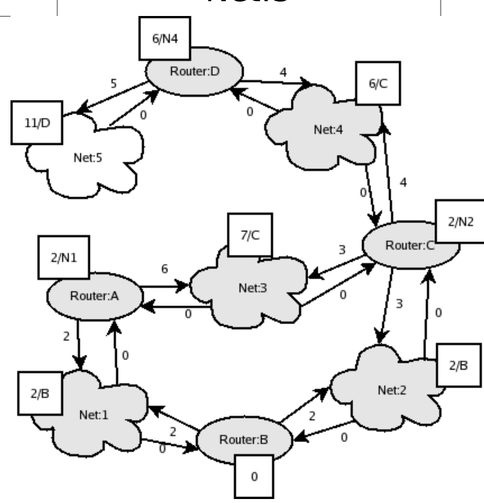
7.

Pass	***Node***	***Cost***
1	B	0
2	Net:1	2/B
3	A	2/Net:1
4	Net:2	2/B
5	C	2/Net:2
6	Net:4	6/C
7	D	6/Net:4



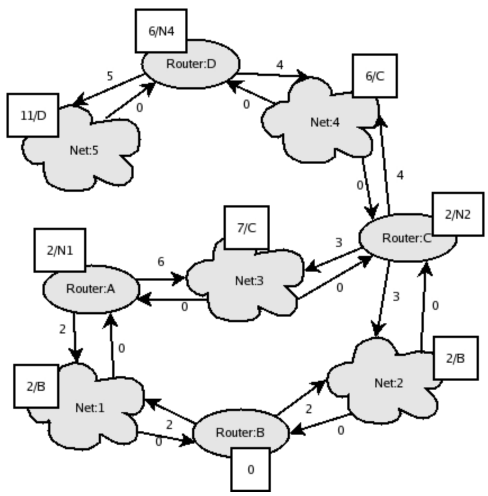
8.

Pass	***Node***	***Cost***
1	B	0
2	Net:1	2/B
3	A	2/Net:1
4	Net:2	2/B
5	C	2/Net:2
6	Net:4	6/C
7	D	6/Net:4
8	Net:3	7/C



9.

Pass	***Node***	***Cost***
1	B	0
2	Net:1	2/B
3	A	2/Net:1
4	Net:2	2/B
5	C	2/Net:2
6	Net:4	6/C
7	D	6/Net:4
8	Net:3	7/C
9	Net:5	11/D



- Then from the final table, the routing table can be extracted

Router B Table

Network	*Cost*	*Next Hop*
1	2	-
2	2	-
3	5	C
4	6	C
5	11	C

- the only tricky entry in the above table is for Net:5 where the next hop must be C, not D (as in the Dijkstra solution). This entry can be found by walking backwards through the Dijkstra solution to the first router after B (Net:5 comes from D, D comes from Net:4, Net:4 comes from C and router B can send directly to C --> therefore C is the first hop)