CSE 160 – Computer Networks Spring 2021 Section: 02L

Daniel Alejandro Maciel Manzo

Project 2: Distance Vector Routing

In project 1 we got to implement on of the simplest ways of message propagation in a network with different nodes. In this project we implement a much more efficient way of routing messages trough a network. The idea consists of having a link state packet layer that will use the information from neighbor discovery along with Dijkstra's algorithm (to find the shortest path) to create link state packets. These link state packets represent the state of the network. Each node will have its own link state packet collection that will help the node know where to redirect a packet that has a specific node destination.

The routing table design consists of a Hashmap<int> key: TOS_NODE_ID value: next_hop. Each node owns its own routing table, and it gives the node a direct route of where to send the packet trough to get to a destination. This table is first initialized after the NEIGHBOR_DISCOVERY routine ends. A timer is fired to populate a routing table first with only the direct neighbors. Then another timer is fired for a period subroutine that will trigger Dijkstra's algorithm for calculating the shortest path to get from A to B and save the next hop in the routing table. After an updated routing table has been created it is then copied into a message as the payload. Then it is sent from the node that updated the table to all of it immediate neighbors. The neighbors check in the cache to see if they already have the information up to date in their tables. If the new routing table is new, then we replace it in our routing table and send the updated routing table to immediate neighbors until all neighbors are updated or the TTL reaches 0.

When a new ping message is sent by A to D, and received by B. B will check it routing table to see where the next hop is. If there is no routing table then we will flood and along the flooding Link State Updates will be created by each node that receives the package effectively giving all nodes trough the path a routing table to get to the next hop. When a message is sent again from A to D and B received it. B will know to route the message to the next hop in the routing table and will not start a flooding sequence.

1. What are the pros and cons of using distance vector routing compared to link state routing?
    a. The pros of vector distance routing include the fact that it needs less configuration than link state routing, which is more difficult. However, distance vector routing is both expensive and sluggish. Unless optimizations like split horizon and poison reverse are in place, it is vulnerable to routing loops like the count to infinity problem.
2. Does your routing algorithm produce symmetric routes (that follow the same path from X to Y in reverse when going from Y to X)? Why or why not?
    a. Since the routing algorithm implements Dijkstra's shortest path for graph. seeks the shortest path between nodes, the path from X to Y should be the shortest path from Y to X.
3. What if a node advertised itself as having a route to some nodes, but never forwards packets to those nodes? Is there anything you can do in your implementation to deal with this case?
    a. I implement a node Aging method such as the one implemented for neighbor discovery. The nodes age is increased every time there is no update from it. Once it reaches MAX_AGE we drop it from the routing table.
4. What happens if a distance vector packet is lost or corrupted?
    a. If a packet is lost, the other packets are updated, but if a corrupted packet is sent with corrupted route information, poison reverse kicks in to avoid a loop.
5. What would happen if a node alternated between advertising and withdrawing a route to a node every few milliseconds? How might you modify your implementation to deal with this case?

    a. There will be too many collisions if the node alternated between advertising and removing a node, preventing any packets from being routed efficiently and correctly. The routing table will struggle to keep track of the nodes are available to receive packets.