CSE 160 Computer Networks

Project 2

Discussion Questions & Report

Abdias Tellez & Renato Millan

REPORT

In Project #2, we use the neighbor discover and some implementations from flooding in our implementations of distance vector routing. Neighbor discovery was a pivotal part of distance vector routing so we can calculate the distance in every node's routing table. The flooding helps us create a receiver and wiring the send from simple send to distance vector routing. Therefore, the sender from flooding would interfere with distance vector routing's receiver during the ping command in Ping Test.

Foremost, we needed to get the neighbor into the Distance vector routing before we can route a packet. In addition, we implemented a timer in Distance vector routing, so it performs periodic updates and advertisements on the network. The neighbor discovery has the knowledge of the networks before sending out routing packets.

Furthermore, we use the book implementation of an struct array so we can create the routing table structure. The array has 4 pieces of information in the element/struct/route, it contains the destination, next hop, distance, and the TLL for the route packet. We decide to use uint8_t because uint8_t allows indexing 255 nodes (2^8 -1=255) for the array struct. This decision allows greater node ids that can be entered in a smaller table. Our implementation can be altered to accommodate different routing table size that and the number of nodes that can be routed. Considering that we only must deal with 256 nodes the length will be defaulted to the length of 256. Since the packet payload allows a maximum of 20 8-bit unsigned integers and size of our route struct is 4 bytes(one for each context), the maximum of routes we can send per packet is 5; the reason for route struct. Our distance vector routing will use the shortest path calculation to transverse to the packet's destination, thus guarantying that the current path is the shortest path.

Moreover, the distance vector algorithm was simple to understand. The most difficult element of building the algorithm was ensuring that the ability to appropriately update the TTL values for the routes in the database was preserved. Many factors were examined to determine if the route received was a new route, an update of an existing route, or a less-than-optimal path. In all except the last example, the TTL was set to renew.

Finally, we implemented both strategies split horizon and poison reverse so we restores tables to prevent infinite loops, incorrect path, and incorrect cost. Therefore, we implemented these strategies every time we call an update to the routing table or adding new packets; if we received a corrupted packet or the packet is loss the routing table is restore to its previous state before it's call to that packet.

1. What are the pros and cons of using distance vector routing compared to link state routing?

Pros:

- Configuration for vector routing is faster to set up, as compared to link state routing

Cons:

- Count to Infinity problem from the routing loops (feeding old information)
- Requires split horizon and poison reverse method to solve count to infinity problem
- Slower compared to link state routing

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?

The distance vector routing(DVR) algorithm finds the shortest path from X to Y. Let's say we have a sub-path in the shortest path between X and Y, let's call them A and B. By the DVR algorithm, the path from A to B should be the shortest path from A to B. Because of this, the algorithm should produce symmetric routes. If our route from nodes X to Y include A and B, the path from X to Y should be X -> A -> B -> Y. By symmetry, the shortest path from Y to X would be Y -> B -> A -> 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?

If a node advertises itself as having a route with some nodes, but never forwards the packet to those nodes, the algorithm will have to know if the packets are sent and received. If the packet is not received by the receiver, the route is dropped from the routing table.

4. What happens if a distance vector packet is lost or corrupted?

Our program handles the corrupted or lost packets and restores the table. Lost and corrupted packets are handled by split horizon and poison reverse, which in turn stop infinite loops.

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?

If nodes are alternating between advertising and withdrawing a node, incorrect information would be put into routing tables because of the collisions. This can result in infinite loops as route costs would be incorrectly calculated. To solve this, we can use a strategy that can acknowledge changing paths. This protocol would allow the advertising or withdrawing to be open for some time, where the node is able to receive a packet.