Ben Blau

CS 455

11/13/14

Programming Assignment 3 – Routing Protocols

*Testing and Compilation*:

First compile all of the required files including the primary file modified for this assignment, “Node.java.” After compiling all of the necessary files in the same directory, run “Project3.java” to test the code. Alternatively you can type “java Project3” to run on the Boston University “csa2.bu.edu” or “csa2.bu.edu” server. You can modify the “LINKCHANGES” in “NetworkSimlator.java” to test the link changing function (0 to disable, 1 to enable) and upon running you can modify the trace and seed for different desired outputs. To make sure the code is working you can compare the tables representing the distances for every node after there are no routing packets left in the network. The table for each node should contain the shortest path to all other nodes before both link changes and when the program finishes running. In addition, the tables prior to the first link change should be the same as those after the second link change.

*General Design*:

The concept behind this programming assignment is to simulate a connected network of routers. We implemented a split-horizon with poison reverse heuristic. The simulated network of routers sends packets to each other containing the shortest path between them. When one node is out of reach of another the distance is poisoned. This sets the distance to “INFINITY” which was defined as “9999.” When one node communicates with the next it will send the information of whether or not a path is reachable to the other nodes in the network, if there is no path then the receiving node will cut the link to that node and create a loop with the initial sending node. The poison protocol we implemented will poison the link in the situation where the node will loop back with the original sender.

At the start of the program all four nodes will be initialized in the “NetworkSimulator” by the method “rtinit().” The starting distance is set for all connected nodes. At this point the nodes send packets to each other containing distances to all other nodes (method “transferMinimumCosts()”) and then each node updates its distance table if a shorter distance is available. Upon updating (method “rtupdate()”), packets will be sent out again with the new information. This process repeats until no shorter paths are available and the tables finish updating. In the process of running there will be two link changes (when the feature is toggled on). The first will change the link between node 0 and node 1 from 1 to 20 at t = 10000. The second will occur at t = 20000 and will return the link back to 1 from 20. The distances are updated upon link changes.