Project 2

Cpts/EE 455 Carl Hauser

November 16, 2015

Daiane Torres Feitosa
Gregory Joseph Taranto
Feross Salameh

The Routing Problem

In this project, the problem was to create a network of simulated routers implementing a distance-vector routing algorithm. Each simulated router is a separate process which will communicate to other router processes through connected datagram sockets.

The user can communicate to router processes via L and P messages (Link update and Print Table messages respectively). These force the router to change a link distance cost or print to the user a routers current distance vector table.

Routers communicate to each other through U messages (update messages) either routinely every 30 seconds, or on triggered updates which is when a link changes or a neighbor has a link change.

Each router sits in a state of listening for its neighbor's status to change or it's own status to be forcibly changed through a L message. The router sits in a select function call which will timeout every 30 seconds or receive an update. Regardless of how long it sits waiting the response is the same: send an update message to each neighbor of its current distance vector table. Then continue waiting for a neighbor response for at most 30 seconds.

Each router is governed through the simulation through another program which can start and stop processes, and send messages to individual routers. This program is what is used to execute test scripts to test the simulated network as a whole.

The Solution

User Interface:

The user interface provided is written in C# and takes place of the suggested scripts in the project assignment. It allows the user to send messages to specific routers, start test cases, start individual router programs, and terminate one or all router programs operating.

Loop:

The loop sequence, or the code that listens for incoming messages via the 'select' function call, is an infinite loop that cycles through a select sequence every 10 seconds. 10 seconds vice 30 seconds was chosen so that the routers will move react to input from the user or other routers at a rate that will expediently execute a test case. The main idea of the loop is that the FD_SETs used to represent the ports a router is listening to are reset, a timeout is set, select is called (and the program waits for a response), and the router takes appropriate action depending on the response (if any). If no stimuli is sent to the router during the select-wait, the router will send a routine, update message. If an error is received during the select-wait, the router will announce it to the user. Lastly if any message is received the router will take appropriate Print or Link-Change actions, or send a response to an Update message.

Algorithm:

The distance vector algorithm works as follows: Two nested loops will increment through the neighbors distance tables and compare against the current distance vector table of the router. Every entry in the distance vector table needs an update. A shorter or equivalent distance route satisfies this requirement.

After cycling through each neighbor's distance table for a shorter or equivalent route, the router will see if all routing entries have been updated, if not, it means the routing table entry has actually increased in distance cost. The router will set those distances to infinity and search again for the shortest distance for only those entries.

Lastly if a path shorter than infinity still isn't found it means one of two cases: either the path increase is a neighbor who's routing distance isn't set by anyone else or the router has been removed from the network. Either case means the distance is set to the default distance. The default distance for a neighbor is the value last loaded from the test case setup or manually adjusted by the user. If it is not a neighbor, the default distance is infinite, which is the same as a router being removed from the network.

Now once all routes have been updated, messages are generated and sent to each neighbor as an update message.

Message Handling:

All messages begin by tokenizing the incoming message, performing some basic error checking, and taking the required action for the message. The appropriate action by message type are:

Print:

A print message will print either the entire distance vector table or a single entry if a parameter is passed in with the message. This is done by accessing the distance vector table for all or one of the entries and printing to the user the router name, distance, and next hop.

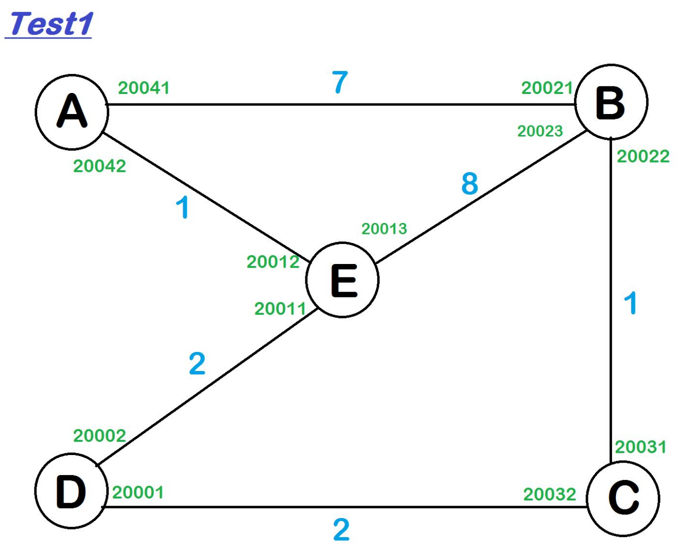
Link Change:

This manually changes the default distance for a router. It will use this cost in the distance vector routing calculations.

Update:

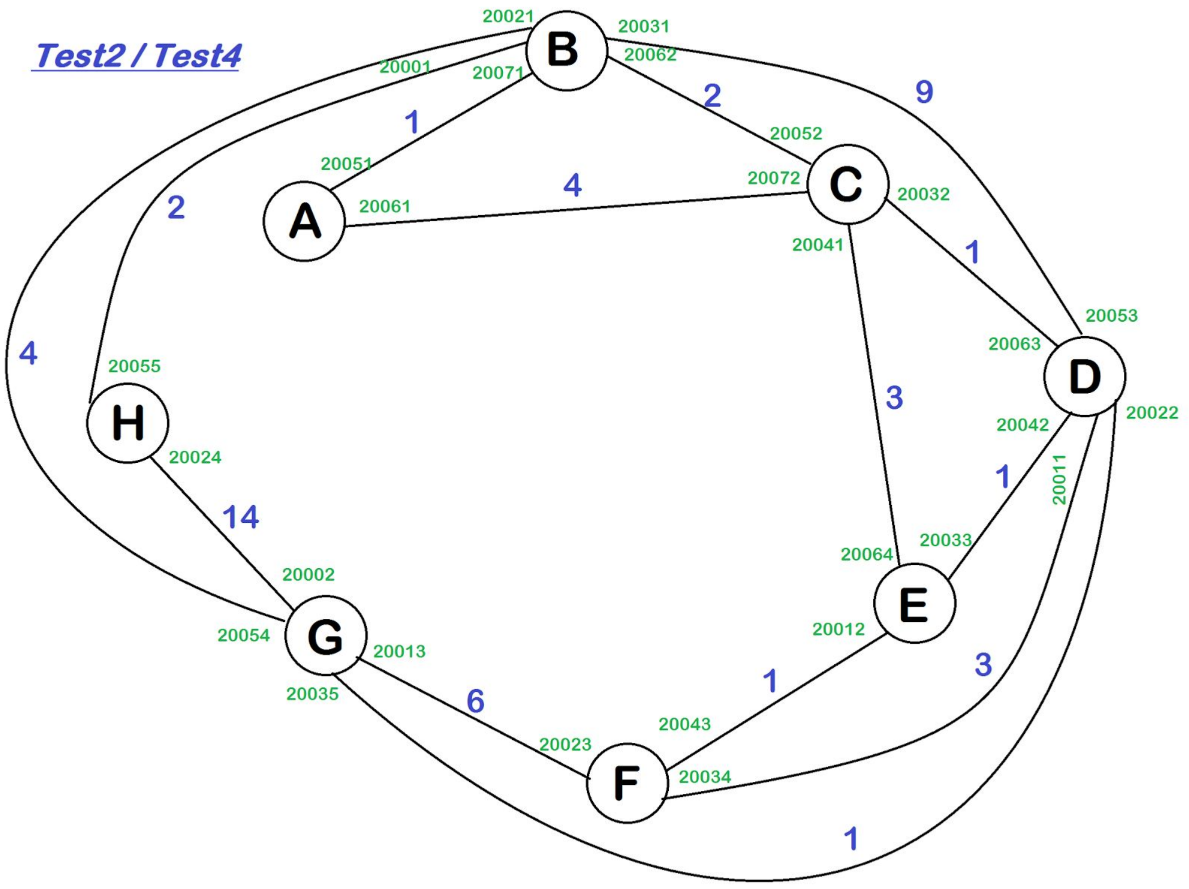
An update is an incoming message from a neighbor and sets the distance cost information to non-neighboring, more distant routers. Following update and link change messages, the algorithm is ran to look for new routes. This is the follow-up of a triggered update.

The Test Cases



Under the test 1, the program should run from A to B in a path with cost 7. On the other hand, A looked into their neighbors, A discovered a path with small cost as A, E, D, C and B, with cost 1 + 2 + 2 +1 = 6, respectively. In addition, the program shows that from B to E has a cost of 8. However, there exist a path with under cost as B, C, D and E in total, 1 + 2 + 2 = 5 respectively, and the program follow this new smaller cost in the path.

The results of the test 1 were behaving correctly. Each router converged on the correct route after three rounds of back and forth from the routers. In most routers, the path of E-B was avoided because of the lowers costs between D-C-B. The results are in the output files.



Under the test 2 and 4, the program should run from B to D in a directly path with cost 9. However, B looked into their neighbors, B discovered a path with small cost as B, C and D, with cost 2 + 1 = 3, respectively. In addition, the program shows that from F to G has a cost of 6. On the other hand, there is a path with under cost as F, E, D and G in total, 1 + 1 + 1 = 3 respectively, and the program follow this new smaller cost in the path. Furthermore, the cost of the path through G to H cost 14. The program can find a smaller path as G, B and H, as consecutively 4 + 2 = 6.

Although the router map was more complicated than test1, both test2 and test4 met satisfactory requirements. Each router reached convergence after 5 rounds and back and forth. The results are in the output files.

Additional Experiments

No additional experiments were performed.

Significant Decisions

Based on the guide and test cases, an assumption was made to store the name of every router to be simulated in a single char. The reason for the decision was arbitrary. A string could have been used which would have the possibility of larger networks to be simulated. The most number of routers to be simulated based on this limit is 127 (128 ascii characters available except for the <space> char.

Another assumption made was implementing an upper limit in message size. The reason for this decision to limit message size was to ease coding and error possibilities. However, the implication of a set message size is that the number of simulated routers in a network is limited. For instance, if the distance of between each router and the name length of each router was one char long, then an update message with a set size of 512 can contain at most 127 routers. "U r1 d1 r2 d2 … r127 d127" where each " rn dn" group is 4 chars long (including spaces). Four chars * 127 + 'U' = 509. For every routing distance to a node exceeding one char length, the available space for simulated routers decreases.

Comparison

One deviation from the book and our project was that in the book, the "Good news travels fast, bad news travels slow" phenomena was stressed. However in the project problem statement, we were to implement triggered updates that would send out update messages of our routing table immediately. Triggered update messages were sent immediately regardless of being a "good news" or "bad news" message.

Conclusion

One capability a real router possesses is that our simulated router does not is built in protection for split horizon possibilities. Another capability, there is possible for a real router to possess, is for a router to build a table of information such as distance cost, and next hop for itself. Our system is capable of doing this. On the other hand, we do not implement this data gathering as it is useless information for the assigned problem.

One deviation from RIP that was required by the problem statement was to assign the value infinity (INF) to 64. RIP assigns INF to a value of 16.

Spent Time in Project

Group Meetings:

~3-4hrs depending on meeting. Some had to leave earlier than others. Some could not make a meeting. Most communication was done over Facebook or Github via comments. The purpose of the meetings were status updates: to come up with a game plan on what remains, announce what works, announce what doesn't work and a plan to complete, and some troubleshooting.

Individual Time Spent:

Daiane: 14hrs
Greg: 30hrs
Feross: 25hrs