COSC 364 – Assignment 1

Isaac Foster – 64555890

Zachary Sanson – 58520526

# Percentage of contribution of each partner

50% both ways.

# Which aspects of your program do you consider well done?

The main properties of the program we feel are very well done. This ranges from multi-threading operations which was a rather challenging problem at first, but as soon as we got a handle of how threading interacts with other threads and objects the program really started to work well; Packet handling, the whole way that we deal with packing and unpacking data to and from byte streams is implemented well and is simple to use and understand; logical decisions within the routers classes, which we feel were very challenging at first but now looking at our functions, works quite well; All the docstrings and comments which allow for easy references and understanding.

# Which aspects of your program could be improved?

The converting between different class types, lists and dictionaries. At the moment there is a few non-straight forward or unlinked functions. The interaction between the packet classes and the router classes could do with a small change to allow for better accessing and interchanging of data. As an example take the packet class’s RipEntries field and the routers dictionaries; there was a need for conversion to interact between the two functions when they could just contain the same data types and be access interchangeably.

# How have you ensured atomicity of event processing?

Due to running processes involving iteration of, and modification of dictionaries on parallel threads, we have implemented a system where the dictionary is copied to a local working dictionary to allow other threads to run without issue. Once the working dictionary has been successfully modified it is copied back into place. This minimises the chances of threads overwriting each other’s changes. Threads that receive information are blocking with select() until all data is received into the buffer, and sending threads use a global pause boolean designed to skip the send loop until other send loops are finished with their operations.

# Testing we have performed as a group:

All throughout the creation of this program we have run a plentiful amount of testing. We originally started of trying to get the router to display to correct information fields and update/refresh the display with information. And after we implementing more functions we would test each one of them along the way, running multiple tests to check that the program was functioning the way we expected it to. There were several times where the testing of our program did not show us our expected results, although in these times we would just group together and discuss how to fix these issues and get the program functioning again. Originally when we were working on separate files we had to run separate tests to check if our own change worked before we merged it with the other person. Making sure the merge resulted in a functioning program and fixing the errors that occurred.

A summary of items we tested for consistency was:

Start up config (making sure the program unpacks the config properly), router start-up & port bindings, packet formation, consistent timer calls, multithreading safety, dictionary/list contents, interaction between routers (sending and receiving), route finding, metrics, garbage collections, timeouts and much more. We also purposefully constructed routing loops and tested the split horizon / poison reverse functionality. Each time we would try to run on several routers each with slightly different configs to allow for differing topologies. Another edge case we tried to test for was where thread functions could technically collide. This seemed to test well and when we managed to get a collision the router still managed to get back into sync and an error message was displayed.