COSC 465 Capstone Project

Alex Greene, John-Michael Chin, Lizeth Mora Guerrero, Luis Ramos

Milestone 1 Reflection

Progress:

Since our proposal, we have created multiple classes and functions to complete the proposed router simulation goal. We have created 6 files, including client.py, packet.py, router.py, simulation.py, test.py, and network_config_test.json. Our client file creates a client class with a client name, IP, and mac address. Our packet file contains different mini functions but is made of sourceIP, src mac address, destination IP, destination mac address, and the packet itself. Two key functions in the packet file are creating an ARP request and an ARP reply. Our router class is more complex where it is made up of a switchyard network object, a routing table, a client table, both of which are empty dictionaries, and a dictionary containing interface information, such as the IP, subnet mask, and mac address. We have four functions that process a packet, handle ARP requests, handle IP packets, and send the packet. Our simulation file works in hand with our JSON file to set up the network as our JSON files have the routers and clients with their respective names, IP addresses, subnet masks, interfaces, IDs, and gateways. Our test file will be utilized to ultimately test our code and see how successful we are in creating a simulation. We are using Switchyard to create a test scenario to see if our network topology works. What this looks like essentially is that there's a topology we have created where Client1 communicates with Router1, Router1 communicates with Router2 and Router2 communicates with Client2. Something we are doing that is different from Switchyard is using the net objects' "interfaces" as our nodes in the network. The nodes in this case will be our 2 clients and routers previously mentioned in our topology example.

Struggles:

A struggle we have encountered that has slowed down our steps is running Swayed. When we try it and run the test file, it sends the message that we are missing the scenario definition; however, we define scenario, so it is as if it is not being acknowledged properly. Understanding Swyard in general has been a huge struggle since we've never actually had to work with coding network topologies using this framework.

To-Do:

As of right now ideally, when we run the Swyard test environment, our network should recognize ARP requests from the router to the client and ARP replies so that it can store its Mac address. We still need to handle IP packets being forwarded across the network from clients using static routing protocols and add dynamic routing protocols for larger networks with multiple routers and clients. The UI aspect also needs to happen, but that will happen after milestone 2.

Timeline:

Once the error defined above is figured out, ideally we should be able to finish our first network where a client sends a packet to another client in another router and they are all connected via "direct links". That should be done by the end of this week. Adding a new JSON file with a network topology with multiple routers so that we can introduce dynamic routing protocols should take around a week as well and we'll have done by Milestone 2. Once that happens we should be able to add a UI aspect that simulates what's happening given the network topology scenario. Maybe even adding a few options where the user can choose which dynamic route/protocol to use so they can see the same network topology with packets going in different paths and coming from different clients would be a neat feature to have.