**CS4283/5283: Computer Networks**

Vanderbilt University, Fall 2022; Instructor: Aniruddha Gokhale

**Programming Assignment #4**

Handed out 11/10/2022; Due 12/01/2022 (11:59 pm Central in Brightspace)

# **Assignment Theme:** SDN-drivenRouting in Docker Overlay Networks

**Overview**

In this assignment, we are going to construct routing tables using Dijkstra’s algorithm and supply these routing tables to individual application-level routers so that messages from client to server are relayed using these routes. To keep things simple and enable us to focus on the primary objective of this assignment, we will be using the test\_client, test\_router and test\_server that is available in our nwclass image that is already created and made available in our private registry.

**Expected Operation:**

We are given the following topology:

Diagram

Description automatically generated

Essentially, there are 9 networks in the above topology as shown below (a somewhat busy and confusing picture, but see below for the explanation).

Diagram

Description automatically generated

For example, the client is connected to R1, R2 and R3 via three different interfaces. Consequently, the client is part of three different networks, which in our case are going to be the overlay networks. Other networks can be explained in a similar manner.

**Supplied Material**

The Docker Swarm comprising one master and ten workers is already created for us in our Chameleon project CH-819381 project. The image needed to run our client/router/server code is also already pulled and installed in our private registry. See the Chameleon GUI screenshot below for all the nodes in our cluster along with their public IP addresses and private IP addresses. You will need the public IP addresses to ssh into these individual machines whereas all the internal communication needed to demonstrate the client->set of routers->server communication should be done using private IP addresses. We have also created the nine overlay networks that we will need for this project. These are named as overlay\_nw1 thru overlay\_nw9.

Table

Description automatically generated

A picture containing text, plaque

Description automatically generated

**Expected Setup**

You will need to create seven services. Please do not give generic names like “client” or “router” to these because these may conflict with the same name chosen by another student. Thus, please use unique names for the services such as <unique\_prefix>\_client, <unique\_prefix>\_router1, ……, <unique\_prefix>\_router7, and <unique\_prefix>\_server. Depending on whether it is a router or server, publish the appropriate port using mode=host approach. By using the mode=host approach, we will not rely on the Please consult the table below for the mapping of the 4444 and 5555 ports that each one of us must use. Each of the services that we create must belong to the appropriate number and distinct networks chosen from among the 9 overlay networks that are available. Use any of the 10 worker nodes upon which you will instantiate your seven services. Make sure that each service is on a distinct worker node. We will use the master to execute control operations/validation.

**Expected Operation**

We will find shortest path from client to server going thru some routers using two approaches:

1. Smallest hop count
2. Shortest end to end delay. For this, use the “tc” trick that we did in class to assign different delays to different interfaces going out of each router. After this, Dijkstra’s algo will be run to compute the shortest delay (even if it may be longer hop count).
3. Compare the results from #2 using smallest delay vs shortest hop count when tc is enabled.
4. Try a different set of delays and see if your routes are different and redo the experiment and get the comparisons. You can plot graphs showing the comparisons.

Note: I am hoping to cover some SDN controller/switch material during the week of Nov 14th, and this will be used to run the Dijkstra’s algo. If this does not materialize, we will just build our own ad hoc external controller and compute the shortest path. The SDN controller is supposed to have visibility into the network i.e., the topology is known.

**Reserved Port per Participant**

Note that because we are all going to use the same set of Swarm cluster nodes and same overlay networks, we all cannot publish the 4444 and 5555 ports used by the router and server, respectively, as 4444 and 5555, respectively because only one instance of the port usage per VM is allowed. As a result, please use the following table for your port number mappings. Please strictly adhere to this and do not use someone else’s port number OR use a completely different port number. Using something else will require additional firewall rules.

|  |  |  |
| --- | --- | --- |
| **Name** | **Published port for 4444** | **Published port for 5555** |
| Instructor (Andy Gokhale) | 30000 | 30001 |
| Jay Barot | 30002 | 30003 |
| Mary Brown | 30004 | 30005 |
| Hao Fu | 30006 | 30007 |
| Xihan Fu | 30008 | 30009 |
| Manda Li | 30010 | 30011 |
| Tianfang Liu | 30012 | 30013 |
| Adit Negi | 30014 | 30015 |
| Zhengyu Shen | 30016 | 30017 |
| Yuheng Shi | 30018 | 30019 |
| Chih-Ting Yeh | 30020 | 30021 |

**Milestones (and hence the deliverables and expectations):**

Please use the Slack Channel for programming assignments discussions/questions.

***Milestone 1:***

* Experiment setup and a manual confirmation that communication with your server is happening from a standalone bash-based test\_client that is run from the Master machine.

***Milestone 2 (final):***

* Shortest path computation using hop count and smallest delay
* Plot graphs
* Create video for the grader
* Submit the code, plots etc to Brightspace

**Rubrics (for grading after the final milestone)**

* Correctness (program works): 40%
* Experiments in all scenarios 20%
* Plots of results 20%
* README file explaining how to run the code: 10%
* Zoom-demo or self-explanatory video to grader: 10%