Programming Assignment: Programming Assignment 4 Submission

You have not submitted. You must earn 7/10 points to pass.

Deadline Pass this assignment by January 22, 11:59 PM PST

Instructions

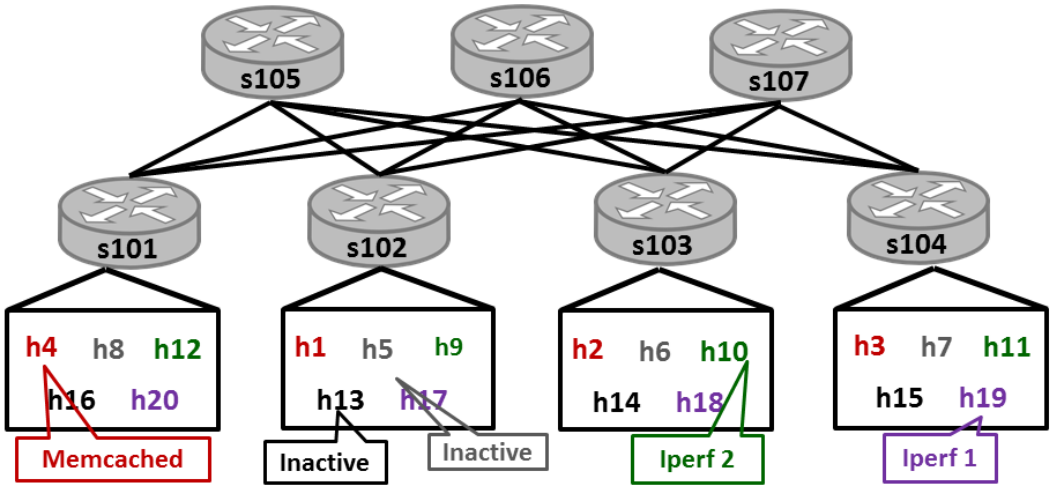
My submission

Discussions

Overview

While the routing policy in the previous assignment utilizes all core switches, statically assigning core switches to route traffic for each VLAN can be problematic. Not all tenants use the same amount of bandwidth, so the load on the core switches may not be evenly distributed. As a result, one core switch may handle traffic for many tenants using large amounts of bandwidth while the other core switches are under utilized. In this assignment, we will observe this scenario and update our routing policy to account for different traffic demands between tenants.

We will use a larger topology for this assignment, shown below, with 3 core switches, 4 edge switches, and 5 hosts per edge switch. There will be 3 different types of tenants, each with 4 hosts: Iperf, Inactive, and Memcached. The 2 Iperf tenants consist of hosts running iperf to simulate bulk transfers. Hosts that are members of the 2 Inactive tenants will not transfer any data. The Memcached tenant will consist of one client retrieving memcached objects, in parallel, from 3 different memcached servers. This will simulate a client loading a web page of multiple objects.



Rather than pre-assigning VLANs to particular core switches, your new routing policy will create a flow scheduling policy that balances flows across core switches, regardless of their VLAN/tenant association. You will implement this improved policy in the file *~/cloudmooc/minidc/controller/policy.py*. Specifically, you will add code to the function *minUtilization()* in the class *AdaptivePolicy*. Your code should use the dictionary *self.utilization* to find the least utilized core switch. This dictionary stores switch names as keys and utilization (in bytes) as the value. Remember: since we are balancing the utilization among core switches, the dictionary *self.utilization* will only contain keys for core switches. Your code should not need to reference any other modules or objects aside from *self.utilization*.

Instructions

We will explore this scenario using memcached, a caching system commonly used for caching objects in a web page. A client will issue requests to memcached servers and record the response time to load all the objects. The client will perform 20 page loads and record the median and 95th percentile from these 20 trials. The results will be reported at the bottom of the dashboard at http://127.0.0.1.

To validate your code, perform the following steps:

1. In terminal 1, cd to *~/cloudnetmooc* and run: *sudo ./mdc --adp*. **Note:** use the parameter --adp for this assignment, not --vid.

2. In terminal 2, cd to *~/cloudnetmooc/minidc/controller.*

- Start Ryu: *ryu-manager controller.py*.
- The default (naive) routing policy will be loaded automatically.

3. Now that Ryu has started, press <enter> in **terminal 1**

4. Open a Chrome instance from the menu bar and navigate to http://127.0.0.1.

- Wait several seconds and observe the graph of memcached response times updates.
- Load the routing policy from Assignment 2 by selecting the radio button labeled **Static** and pressing the **Update Policy** button.
- Observe that the response time does not significantly decrease.
- Load the newest routing policy by selecting the radio button labeled **Adaptive** and pressing the **Update Policy** button.
- If coded correctly, your newest routing policy should significantly decrease the response time.

5. End the experiment:

- In terminal 1, type *exit<enter>* (**do not** Ctrl-c to exit, this will interrupt the teardown process. If you accidentally Ctrl-c and interrupt the process, run *sudo mn -c*).
- In terminal 2, Ctrl-c to stop Ryu.

Expected Result

If your code is working properly, you should see the memcached response time improve when the adaptive policy is loaded. In particular, you should see a noticeable improvement in the 95th percentile compared to the default and static policies

What to Submit

Submit the file *policy.py* containing your changes.

How to submit

When you're ready to submit, you can upload files for each part of the assignment on the "My submission" tab.