SDN implementation report: Mininet and Floodlight

Name: Luthiano Trarbach

Student ID: 23273521

Email: luthiano.trarbach2@mail.dcu.ie

Address: Hazelgrove, First Sea Rd, Sligo.

Course: MSc in Blockchain - Distributed Ledger Technologies

Module: 2024_CACA687I

Appendix I: Student Declaration of Academic Integrity

Students may be required to submit work for assessment in a variety of means, for example physical submission or electronic submission as per the lecturer's instructions. In all cases students must make a declaration of academic integrity, either by physically completing such a declaration and submitting it with their assignment or engaging appropriately with the electronic version of the declaration. Assignments submitted such that the form has not been included, or the electronic equivalent has been circumvented, will not be accepted.

Declaration

NAME:	Luthiano Trarbach
STUDENT ID NUMBER(S)	Luthiano : 23273521
PROGRAMME	MSc in Blockchain
MODULE CODE	CA687I Cloud Systems
ASSIGNMENT TITLE	ASSIGNMENT 2 :SDN
SUBMISSION DATE	April 28th / 2024

I understand that the University regards breaches of academic integrity and plagiarism as grave and serious.

I have read and understood the DCU Academic Integrity and Plagiarism Policy. I accept the penalties that may be imposed should I engage in practice or practices that breach this policy.

I have identified and included the source of all facts, ideas, opinions and viewpoints of others in the assignment references. Direct quotations, paraphrasing, discussion of ideas from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the sources cited are identified in the assignment references.

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

I have used the DCU library referencing guidelines (available at https://www4.dcu.ie/library/classes_and_tutorials/citingreferencing.shtml **and/or** the appropriate referencing system recommended in the assignment guidelines and/or programme documentation.

By signing this form or by submitting material online I confirm that this assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

By signing this form or by submitting material for assessment online I confirm that I have read and understood the DCU Academic Integrity and Plagiarism Policy (available at http://www.dcu.ie/registry/examinations/index.shtml).

Scenario:

Leverage SDN configuration to test network and node distribution impacts on decentralized systems. Create a simulation of network partitions to test the reliability of distributed algorithms.

I've started the report looking at using Ryu with Mininet, after a few attempts to set Ryu I've found a few issues with the libraries. Seems that many of the libraries of Ryu and Mininet has discrepancies in versions. As for example the mininet library has its links pointing to a version of openvswitch installation that has an old path for the db.sock. After a few days of working on the setup of virtual machine with Ryu and Mininet, I've decided to make a pivot in

my strategy and look for Floodlight and Mininet. As a fellow mate from the same course pointed out that has been able to set up I decided to follow that direction.
The setup of Floodlight on a VM was less painful than Ryu but it show me that would be a problem to reproduce and demonstrate a working solution. As my set of skills is focused on solving this kind of scenario I decided to work a bit on creating a reproducible set to make it easy to demonstrate the solution. The solution uses docker-compose and docker to create a Floodlight instance and a mininet instance.
To install docker-compose and docker, Digital Ocean has a great set of articles. For my solution, I used Ubuntu 22.04, Composer installation guide
Is important to understand that Mininet requires a few Linux Kernell capabilities that generally are not used in containers by default. Mininet requires privileged users and capabilities to manage network configurations and bridges. The best approach that I found was to mount the docker host machine to openvswitch sock to the container. This is defined on the flag:

```
    volumes:
      - /var/run/openvswitch/db.sock:/var/run/openvswitch/db.sock
```

That was required for a failure at the start of the mininet container.

So apart from having docker-compose installed we also need openvswitch-switch installed on the machine. For Ubuntu we can install it by executing:

```
apt-get install -y --no-install-recommends openvswitch-switch
```

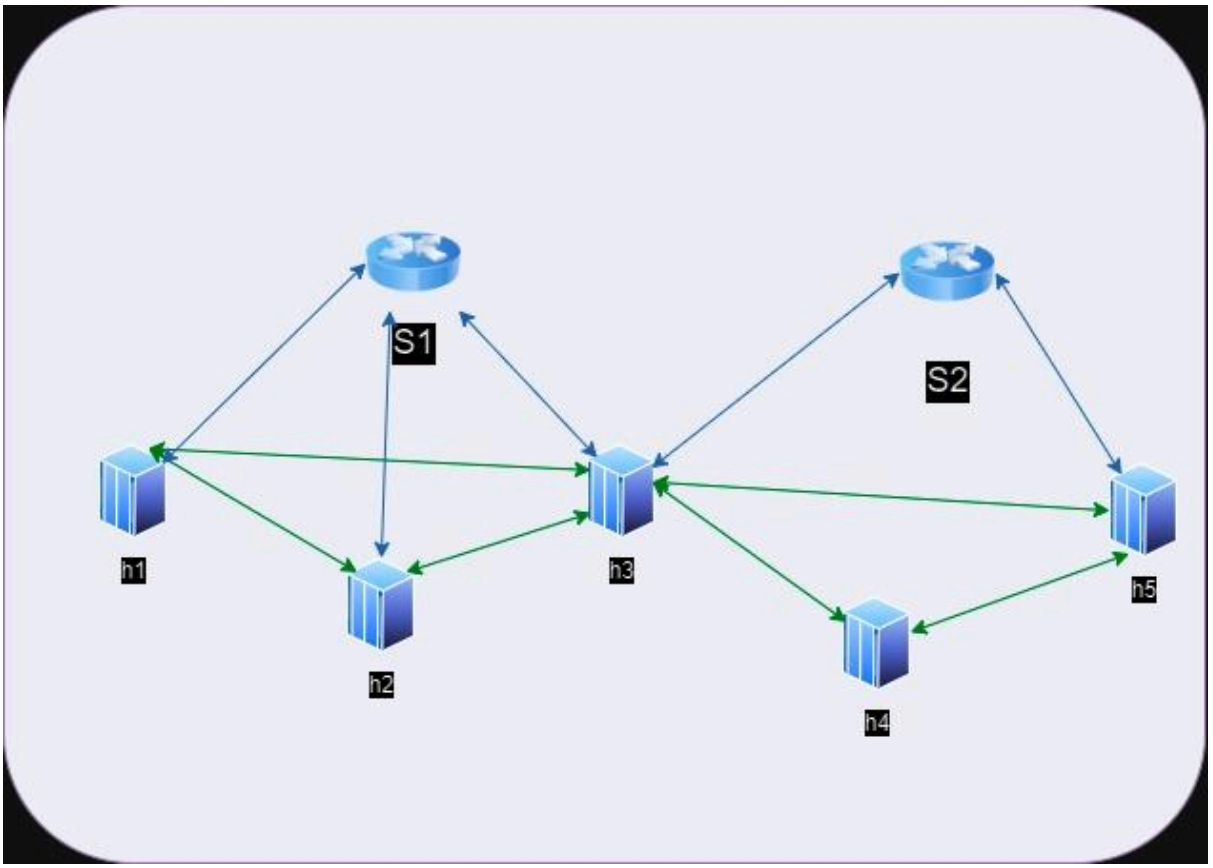
After that, all the requirements for running the solutions should be satisfied.
We can then use:

 docker-compose build --no-cache

That will build the image without caching.

And:
docker-compose up --force-recreate
To force docker to recreate the required containers.

The solution:



The idea was to have python scripts running a raft cluster simulation so that we could create partitions on the network to test. It unfortunately create a few drawbacks as the custom Python scripts don't seem to be executed or got hanging on an unfinished state with the following error on Floodlight:
floodlight_controller_1 | Exception in thread "debugserver" Traceback (most recent call last):
floodlight_controller_1 | File "<string>", line 1, in <module>

My guess and by some article readings is that Floodlight was not been able to create the proper channels for these connections.
So the main idea was to create a use case for SDN in the evaluation of decentralized systems I moved to simple Python httpd servers as it was a simple implementation.

The code creates the hosts using the "h3" host as the main node that is connected to the two switches so once this host "h3" loses the connectivity we could play with two versions of the network creating partitions, and we could abstract this to run our application codes and network number of nodes to test our consensus. As at the moment, I could not implement more reliable scripts in Python I could not implement the main objective which was making a simple socket and syncing server that we could change versions of its running states similar to a Merkle tree hash on the page that we could validate on the end.

The full code is stored on https://github.com/lmtrarbach/Msc-CloudSystems-ca2.git

Few major errors noticed during the development:

Had to shift from a more stable iptables rule setting as when Mininet faced some connectivity with the Floodlight we got the packages dropped and the subnets get dropped the packers.

Also was noticed that sometimes mininet is not able to delete the s2 switch and had to manually delete using ovs-vsctl with the commands:

To get the bridges:

```
ovs-vsctl  list-br
```

To delete the bridges:

```
ovs-vsctl  del-br
```

Conclusions:

The mininet and floodlight are interesting SDN applications to test decentralized networks, we could use the solution to emulate the Blockchain network's consensus algorithms network partition tolerance and scalability. The main issue faced here is the lack of support for the libraries which often has issues in libraries and a lack of documentation. That is of course an issue on open source that is our responsibility to support and provide resources. Also, a possible idea is also implement genomic algorithms to improve the networks using crossover mutations on the best setups.
It's important to note that even do I did my best to create a reproducible environment I reached my self in many situations were I lost complete access to the machines that I had worked.

Bibliography and resources:

1. Chris Jensen, Heidi Howard, and Richard Mortier. 2021. Examining Raft's behaviour during partial network failures. In 1st Workshop on High Availability and Observability of Cloud Systems (HAOC'21), April 26, 2021, Online, United Kingdom. ACM, New Y
2. https://github.com/surfer2047/cheatsheet/blob/master/mininet_cheatsheet.md
3. https://github.com/latarc/dockerfiles
4. https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-compose-on-ubuntu-22-04