



Statement and Confirmation of Own Work

***A signed copy of this form must be submitted with every assignment.
If the statement is missing your work may not be marked.***

Student Declaration

I confirm the following details:

Student Name:	Johnson Kenisha Corera
Student ID Number:	c23020001@cicra.edu.lk
Qualification:	Bachelor in Cyber Security
Unit:	SIT325 Advanced Network Security
Centre:	CICRA Campus
Word Count:	<Word Count>
<p>I have read and understood both <i>Deakin Academic Misconduct Policy</i> and the <i>Referencing and Bibliographies</i> document. To the best of my knowledge my work has been accurately referenced and all sources cited correctly.</p> <p>I confirm that I have not exceeded the stipulated word limit by more than 10%.</p> <p>I confirm that this is my own work and that I have not colluded or plagiarized any part of it.</p>	
Candidate Signature:	J.
Date:	03/12/2024

Task 3.2C

Table of Content

• Introduction.....	05
• Part A.....	06
• Part B.....	08
• Part C.....	10
• Part D	
✓ Q1.....	11
✓ Q2.....	12
✓ Q3.....	13
✓ Q4.....	14
• Conclusion.....	15
• References	16

Table of Figures

Figure 1.....	06
Figure 2.....	07
Figure 3.....	08
Figure 4.....	08
Figure 5.....	09
Figure 6.....	09
Figure 7.....	09
Figure 8.....	10

Table of Acronyms

SDN	Software Define Network
API	Application Program Interface
WAN	Wide Area Network
SNMP	Simple Network Management Protocol
CLI	Command line Interface
EEM	Embedded Event Manager
KPI	Keep Performance Indicator

Introduction

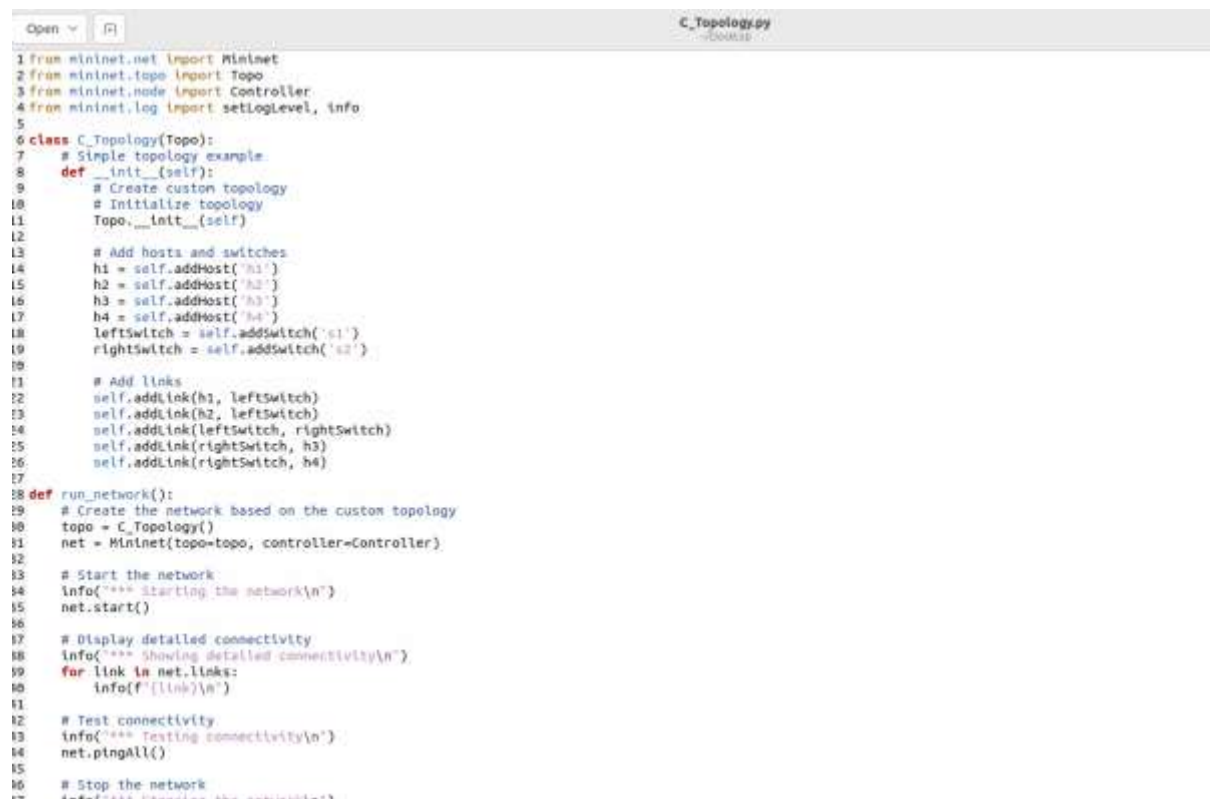
Doing multiple network operations across the different topologies in Mininet, I aimed at enhancing my understanding of network automation using Python and Mininet specifically. I started with the analysis and automation of the custom topology used in a previous work. Includes operations of its creation, identification, and termination processes. To the end of accomplishing these activities, a Python script using the Mininet's API has to be developed. After that, to study the tree like topologies which are predominantly used in data centers and to know how they are built. I instantiated a simple tree topology using Mininet with the help of its high level interfaces. Last of all, I examined how the network operates in several scenarios and tested the connectiveness. Besides, I wrote about statistics of my entries and presented the responses to questions related to network performance, SDN environments, and API usage. As this all-encompassing strategy was proved useful to differentiate the topology automation and management problems,

Part A

In order to automate these operations, I have now built the code for this topology:

- Get the network going.
- Show connectivity profile of each node of a network.
- In order to verify connectivity, one must ping all nodes.

To do the job I had to extend the prior code from the job 3.1p.



```

1 from mininet.net import Mininet
2 from mininet.topo import Topo
3 from mininet.node import Controller
4 from mininet.log import setLogLevel, info
5
6 class C_Topology(Topo):
7     # Simple topology example
8     def __init__(self):
9         # Create custom topology
10        # Initialize topology
11        Topo.__init__(self)
12
13        # Add hosts and switches
14        h1 = self.addHost('h1')
15        h2 = self.addHost('h2')
16        h3 = self.addHost('h3')
17        h4 = self.addHost('h4')
18        leftSwitch = self.addSwitch('s1')
19        rightSwitch = self.addSwitch('s2')
20
21        # Add links
22        self.addLink(h1, leftSwitch)
23        self.addLink(h2, leftSwitch)
24        self.addLink(leftSwitch, rightSwitch)
25        self.addLink(rightSwitch, h3)
26        self.addLink(rightSwitch, h4)
27
28 def run_network():
29     # Create the network based on the custom topology
30     topo = C_Topology()
31     net = Mininet(topo=topo, controller=Controller)
32
33     # Start the network
34     info('*** Starting the network\n')
35     net.start()
36
37     # Display detailed connectivity
38     info('*** Showing detailed connectivity\n')
39     for link in net.links:
40         info(f'{link}\n')
41
42     # Test connectivity
43     info('*** Testing connectivity\n')
44     net.pingall()
45
46     # Stop the network
47     net.stop()

```

Figure 1: The Python Code

- I now automated the tasks listed in the mininet by running the code, as shown

```

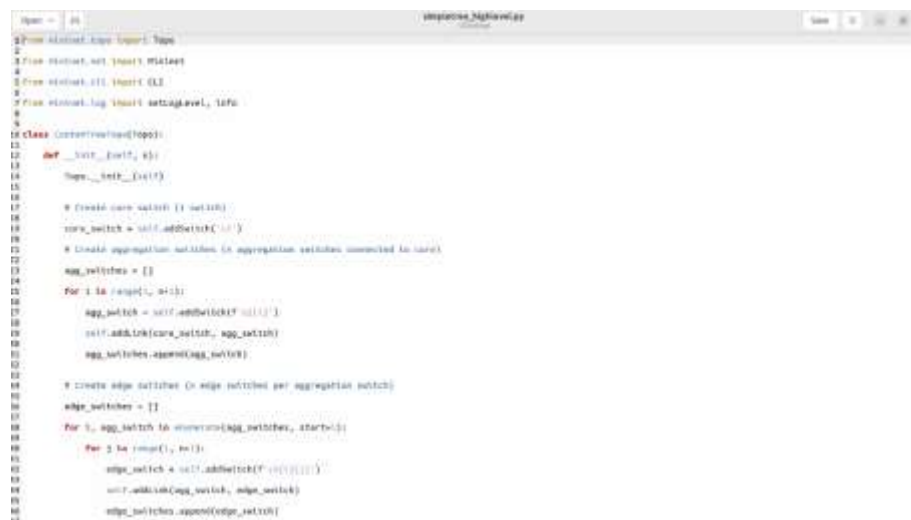
kentisha@kentisha-virtual-machine:~/Desktop$ sudo python3 C_Topology.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (s1, s2) (s2, h3) (s2, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting the network
*** Starting controller
t0
*** Starting 2 switches
s1 s2 ...
*** Showing detailed connectivity
h1-eth0<->s1-eth1
h2-eth0<->s1-eth2
s1-eth3<->s2-eth1
s2-eth2<->h3-eth0
s2-eth3<->h4-eth0
*** Testing connectivity
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
*** Stopping the network
*** Stopping 1 controllers
t0
*** Stopping 5 links
.....
*** Stopping 2 switches
s1 s2
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done

```

Figure2: Run Automate tasks

Part B

- The following simple-tree topology needs to be created and this is the reason why the designing of the Python program using high level API has to be done. Providing an input value of n, the software finds out the number of host connected to each edge switch, the number of aggregation switch connected to the core switch and the number of edge switch connected to each of the aggregation switch. Once I will make sure that the code is adhering itself to the standard naming conventions then I will compile all the code into a single source file having the name `simpletree_highlevel.py`. I will include this file in the folder of my submission.

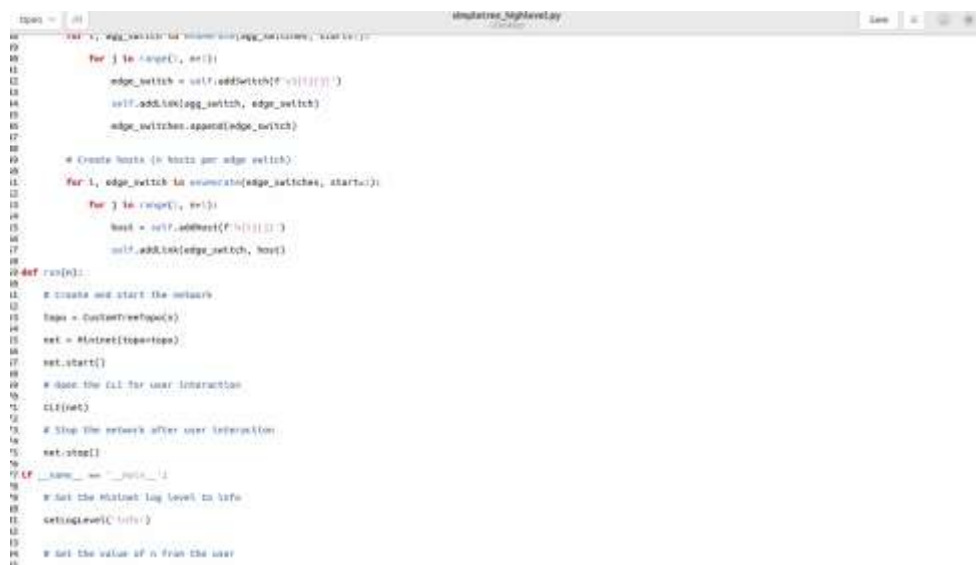


```

1 from mininet.topo import Topo
2
3 from mininet.net import Mininet
4
5 from mininet.cli import CLI
6
7 from mininet.log import setLogLevel, info
8
9
10 class CustomTreeTopo(Topo):
11     def __init__(self, n):
12         Topo.__init__(self)
13
14         # Create core switch (1 switch)
15         core_switch = self.addSwitch('c1')
16
17         # Create aggregation switches (n aggregation switches connected to core)
18         agg_switches = []
19         for i in range(1, n+1):
20             agg_switch = self.addSwitch('a'+str(i))
21             self.addLink(core_switch, agg_switch)
22             agg_switches.append(agg_switch)
23
24         # Create edge switches (n edge switches per aggregation switch)
25         edge_switches = []
26         for i, agg_switch in enumerate(agg_switches, start=1):
27             for j in range(1, n+1):
28                 edge_switch = self.addSwitch('e'+str(i)+str(j))
29                 self.addLink(agg_switch, edge_switch)
30                 edge_switches.append(edge_switch)

```

Figure 3: `simpletree_highlevel.py`



```

31
32         # Create hosts (n hosts per edge switch)
33         for i, edge_switch in enumerate(edge_switches, start=1):
34             for j in range(1, n+1):
35                 host = self.addHost('h'+str(i)+str(j))
36                 self.addLink(edge_switch, host)
37
38     def run(self):
39         # Create and start the network
40         net = Mininet(self)
41         net.start()
42
43         # Open the CLI for user interaction
44         CLI(net)
45
46         # Stop the network after user interaction
47         net.stop()
48
49 if __name__ == '__main__':
50     # Set the Mininet log level to info
51     setLogLevel('info')
52
53     # Set the value of n from the user

```

Figure 4: `simpletree_highlevel.py`

- I then ran the code to create a tree topology with the value n being equal to 2.

```
mininet@mininet:~/mininet$ sudo python3 singletree_highlevel.py
Enter the value of n (n represents the branching factor): 2
*** Creating network
*** Adding controller
*** Adding hosts:
h11 h12 h21 h22 h31 h32 h41 h42
*** Adding switches:
s1 s21 s22 s311 s312 s321 s322
*** Adding links:
(c1, s21) (c1, s22) (s21, s311) (s22, s312) (s21, s321) (s22, s322) (s311, h11) (s312, h12) (s312, h21) (s312, h22) (s321, h31) (s322, h32) (s322, h41) (s322, h42)
*** Configuring hosts
h11 h12 h21 h22 h31 h32 h41 h42
*** Starting controller
c0
*** Starting 7 switches
s1 s21 s22 s311 s312 s321 s322 ...
*** Starting 14:
```

Figure 5: Tree Topology

- Next, I checked the hosts' connectivity.

```
mininet> net
h11 h11-eth0:s311-eth2
h12 h12-eth0:s311-eth3
h21 h21-eth0:s312-eth2
h22 h22-eth0:s312-eth3
h31 h31-eth0:s321-eth2
h32 h32-eth0:s321-eth3
h41 h41-eth0:s322-eth2
h42 h42-eth0:s322-eth3
c1 lo: c1-eth1:s21-eth1 c1-eth2:s22-eth1
s21 lo: s21-eth1:c1-eth1 s21-eth2:s311-eth1 s21-eth3:s312-eth1
s22 lo: s22-eth1:c1-eth2 s22-eth2:s321-eth1 s22-eth3:s322-eth1
s311 lo: s311-eth1:s21-eth2 s311-eth2:h11-eth0 s311-eth3:h12-eth0
s312 lo: s312-eth1:s21-eth3 s312-eth2:h21-eth0 s312-eth3:h22-eth0
s321 lo: s321-eth1:s22-eth2 s321-eth2:h31-eth0 s321-eth3:h32-eth0
s322 lo: s322-eth1:s22-eth3 s322-eth2:h41-eth0 s322-eth3:h42-eth0
c0
```

Figure 6: Topology with nodes

```
mininet> pingall
*** Ping: testing ping reachability
h11 -> h12 h21 h22 h31 h32 h41 h42
h12 -> h11 h21 h22 h31 h32 h41 h42
h21 -> h11 h12 h22 h31 h32 h41 h42
h22 -> h11 h12 h21 h31 h32 h41 h42
h31 -> h11 h12 h21 h22 h32 h41 h42
h32 -> h11 h12 h21 h22 h31 h41 h42
h41 -> h11 h12 h21 h22 h31 h32 h42
h42 -> h11 h12 h21 h22 h31 h32 h41
*** Results: 0% dropped (56/56 received)
```

Figure 7: Pinging Hosts

- Pinging and connectivity are both successful because of no packet loss and control switch is still connected.

Part C

- I will now shut down the topology in an attempt to be able to ping it and see if there is a connection.

```

mininet> switch c1 stop
mininet> net
h11 h11-eth0:s311-eth2
h12 h12-eth0:s311-eth3
h21 h21-eth0:s312-eth2
h22 h22-eth0:s312-eth3
h31 h31-eth0:s321-eth2
h32 h32-eth0:s321-eth3
h41 h41-eth0:s322-eth2
h42 h42-eth0:s322-eth3
c1 lo: c1-eth1:s21-eth1 c1-eth2:s22-eth1
s21 lo: s21-eth1:c1-eth1 s21-eth2:s311-eth1 s21-eth3:s312-eth1
s22 lo: s22-eth1:c1-eth2 s22-eth2:s321-eth1 s22-eth3:s322-eth1
s311 lo: s311-eth1:s21-eth2 s311-eth2:h11-eth0 s311-eth3:h12-eth0
s312 lo: s312-eth1:s21-eth3 s312-eth2:h21-eth0 s312-eth3:h22-eth0
s321 lo: s321-eth1:s22-eth2 s321-eth2:h31-eth0 s321-eth3:h32-eth0
s322 lo: s322-eth1:s22-eth3 s322-eth2:h41-eth0 s322-eth3:h42-eth0
c0
mininet> pingall
*** Ping: testing ping reachability
h11 -> h12 h21 h22 X X X X
h12 -> h11 h21 h22 X X X X
h21 -> h11 h12 h22 X X X X
h22 -> h11 h12 h21 X X X X
h31 -> X X X X h32 h41 h42
h32 -> X X X X h31 h41 h42
h41 -> X X X X h31 h32 h42
h42 -> X X X X h31 h32 h41

```

Figure 8: Unsuccessful Ping

- Because the control switch did not continue and the percent of the packets which were lost is rather high, the ping attempt fails.

Part D

Q1. Are the SDN needs purely related to the data centre and virtualization?

- It is important to bear in mind that what it in fact takes to implement is SDN is not strictly speaking confined to the prerequisites of virtualization and data centers only. Data centres initially used SDN to enable the orchestration of large virtually set environments, but this is not the only use it has. Moreover SDN has been applied in telecommunications, especially in service provider networks, enterprise networks, and wide area networks.
- Centralized network control, as well as the management of resources and traffic control were also made possible by SDN especially for WANs. This especially applies to regulating and directing, traffic especially over broad areas for services. With the policies and auto control of SDN centralized, SDN is able to provide the enterprise networks extend, scalability and more secured network policy and control than setting up complex network policies. (Astuto, 2014)
- SDN has made it possible for telcos to integrate from complicated wired, hardware-based network to simpler software-based ones currently. Further, it influences future services, including 5G, through the provision of techniques for its quick delivery and for achieving lower operating costs. However, in addition to data centres and virtualization, it becomes clear that SDN's programmability, centralization and efficiency improvement characteristics cut across numerous networking domains.

Q2. What type of APIs are available on legacy switches and routers? What is their level of capability?

- Most incumbent switches and routers before SDN implementation have limited APIs; widely used are CLI, SNMP, and sometimes proprietary standards. While the CLI API does not include aspects of automation and programmability, it does permit setting up and command-configuring devices and their protocols by typing commands in as string text. SNMP is for network device monitoring and has features of configurations property and device state settings. However, the functionality is quite limited in SNMP and It is generally only used to monitor and for only basic configuration.
- Although custom APIs provided by third parties such as Juniper Network's Junoscript or Cisco's EEM have more functions, it's normally vendor distinctive and not performed equally among many suppliers. Unfortunately, these APIs lack the flexibility required to maintain today's complex networks and, at times, can be quite complex and cumbersome to manage a batch of automated tasks. Taking everything into consideration, the APIs of the traditional switches and routers are highly limited and suffice only for basic configuration and monitoring with no opportunity to implement further networking, using the programmability now offered by the modern SDN devices.

Q3. What is the level of programming ability present in your development team(s)?

- The following programming skills that I have as a self-employed developer shall be appropriate for managing, and implementing this project. I know many computer languages from different programming languages and for programming I mostly use python which is needed for designing structures of networks that can be seen in the list above. Using related frameworks like Mininet for network simulation is also one of my main skills while this project is yet another one.
- Hence, in terms of creating experience, I have wrapped up a number of jobs that makes me competent in constructing complex systems. These include for example establishing networks, checking connections and developing programs with the right functionality. Also, I have growth orientation, and, therefore, I update knowledge on the newest approaches and tools through attending online classes, tutorials, and practicing in projects. I therefore think that my programming skills are quite suitable for this project's need and that I can come up with a suitable network configuration as well as the right connectivity tests. (Kim, 2013)

Q4. What are various key performance indicators (KPIs) you have noted in this simple set-up and connectivity checks?

- It is observed that when analyzing the connectivity and setup of a basic network there are several KPI that need to be considered in order to evaluate the stability and performance of the network. The connectivity success rate is used to describe the efficiency in the connection of the various nodes in the network in performance of their function transit data between the nodes.
- Delay measures the time between when the data packets start propagating from the source it took for them to reach the destination. to the receiver, a measure of speed of a given network and as well a means of determining the amount of time taken by a given network. The network's throughput defines the level of flow over the specified area of the channel it is capable of receiving. Specific time borders which define if a network can support traffic inside a certain time are highly valuable and prove the capacity. Packet loss measures the fraction of transmitted packets that were never delivered to the recipient and shows network problems and quality. Jitter reflects heights of network coherency and stability and is calculated as the divergence of packets arrival time.
- The bandwidth utilisation statistics, which indicate the distribution of the bandwidth usage, proves useful in Resource usage assessment and Identifying Resource constraints. As the frequency at which errors are generated when relaying information determines the network's efficacy, the Error Rate index identifies hardware or configuration problems that slow it down. Consequently, Response Time evaluates how a network can support suitable interaction as well as how quickly it can respond to queries. For the best performance, these KPIs aggregate information on the status of the networks, and make it easy to note the disruptions that may have affected the flow of performance.

Conclusion

- The case that I have used to explore the various network activities with Mininet through systematic evaluation has helped to enhance my understanding of Topology management and network automation. I have also shown how Python and Mininet API are useful in building and managing networks through the provision of unique and tree like topologies, automation of some important activities and connectivity test.
- Analyzing connectivity and performance provided meaningful understanding of network resilience and efficiency; analyzing key performance metrics including but not limited to latency, throughput, packet loss rate, jitter, and capacity utilization. Moreover, while examining SDN in terms of broader uses, legacy APIs, and the significance of programming skills that have been discovered in today's complex business environment, the critical role of advanced network automation was demonstrated.
- The technical skills needed in the project have been enhanced through this project, and I have enhancing understanding of architecture of the network, automation and management issues and solutions. It also pointed out the need for keeping pace with frameworks and technologies because of the current networking demands' challenges.

References

- A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks
https://www.researchgate.net/publication/260064541_A_Survey_of_SoftwareDefined_Networking_Past_Present_and_Future_of_Programmable_Networks
- Improving Network Management with Software Defined Networking
https://www.researchgate.net/publication/260514929_Improving_Network_Management_with_Software_Defined_Networking