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:	924

I have read and understood both *Deakin Academic Misconduct Policy* and the *Referencing and Bibliographies* document. To the best of my knowledge my work has been accurately referenced and all sources cited correctly.

I confirm that I have not exceeded the stipulated word limit by more than 10%.

I confirm that this is my own work and that I have not colluded or plagiarized any part of it.

Candidate Signature:	J.
Date:	03/12/2024

2.1P Task

Table of Content

Table of Acronyms

Acronyms	Full Form
VM	Virtual Machine
CRM	Customer Relationship Management
BGP	Border Gateway Protocol
ISP	Internet Service Provider
RIP	Routing Information Protocol
SDN	Software Define Network

Table of Figures

Introduction

Task 2.1P on SIT325: The topic of Advanced Network Security is presented in detail in this Assignment. To complete the challenge, Mininet must be used to create a network topology; host connectivity must be verified; and, there are questions about aspects of network theory. The relationship between Software-Defined Networking (SDN) and traffic.

Network Topology

Network diagram

I built the network topology plan that incorporate MAC and IP addresses of switch and hosts. I labeled the switch ports to which the controller and the hosts are connected to. From the result, it was easy to understand the network setups and connections if there was a visualization like this one.

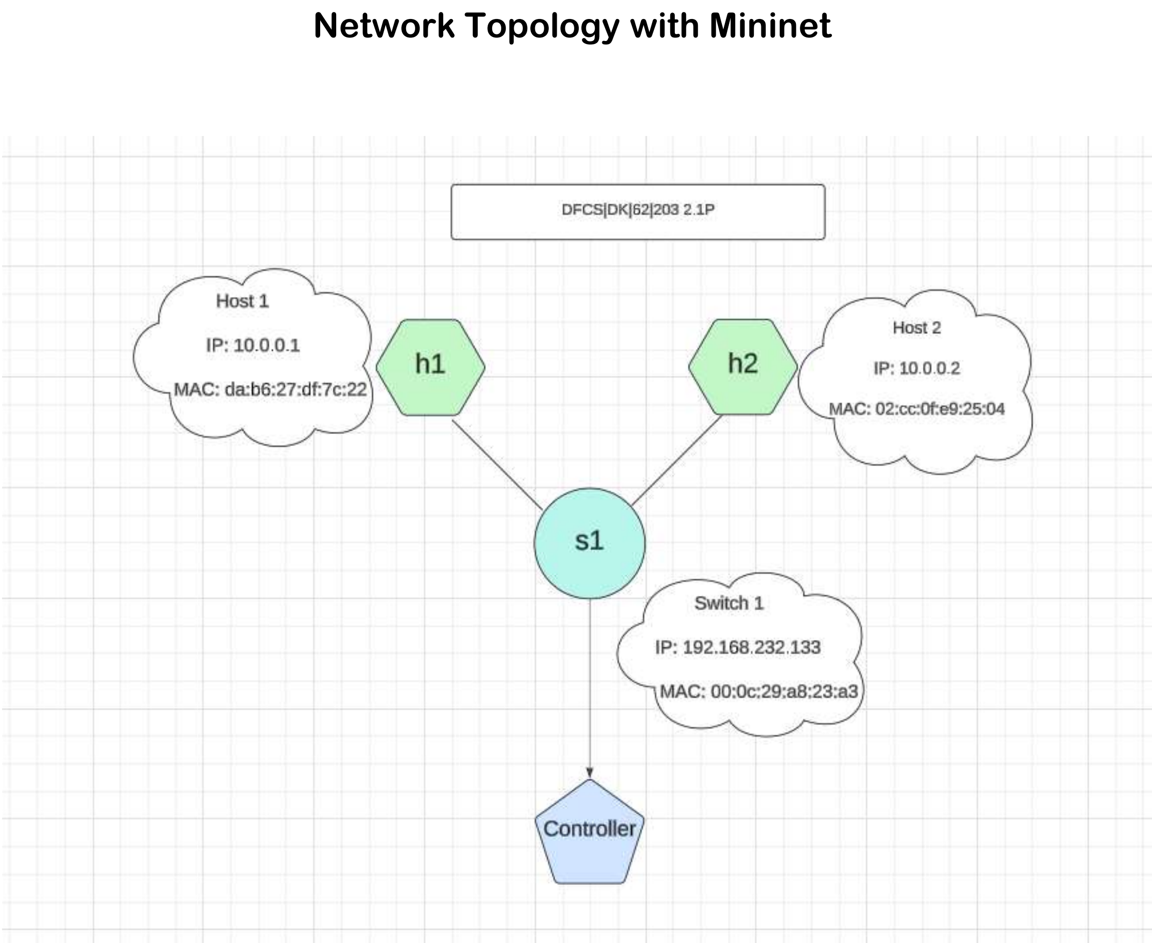
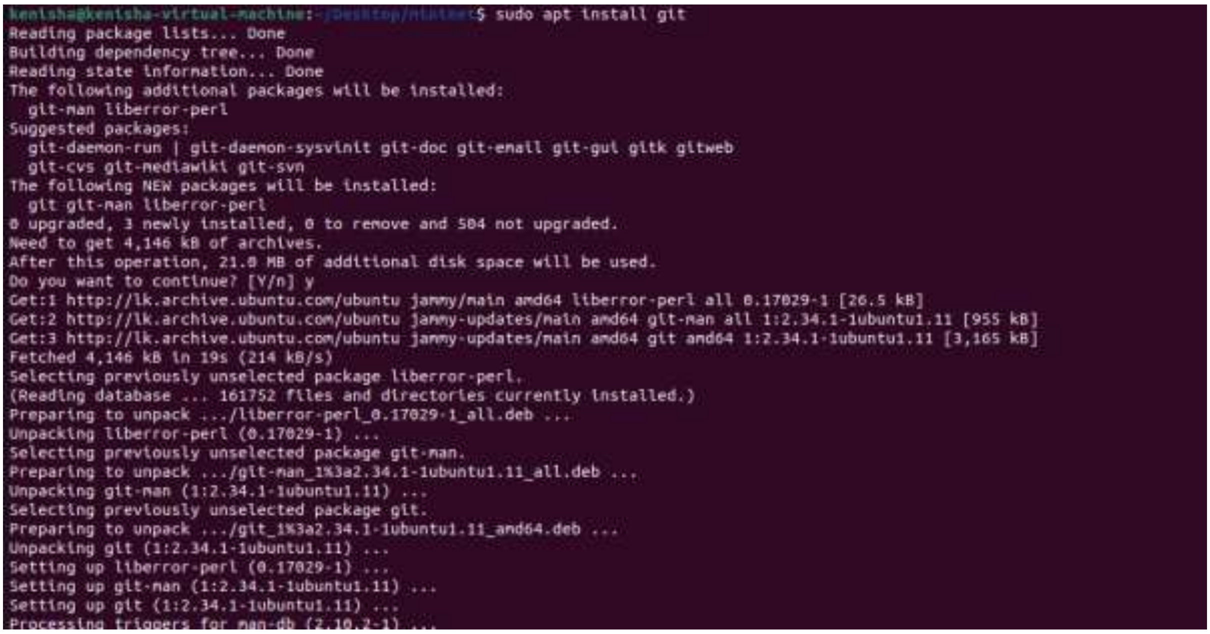
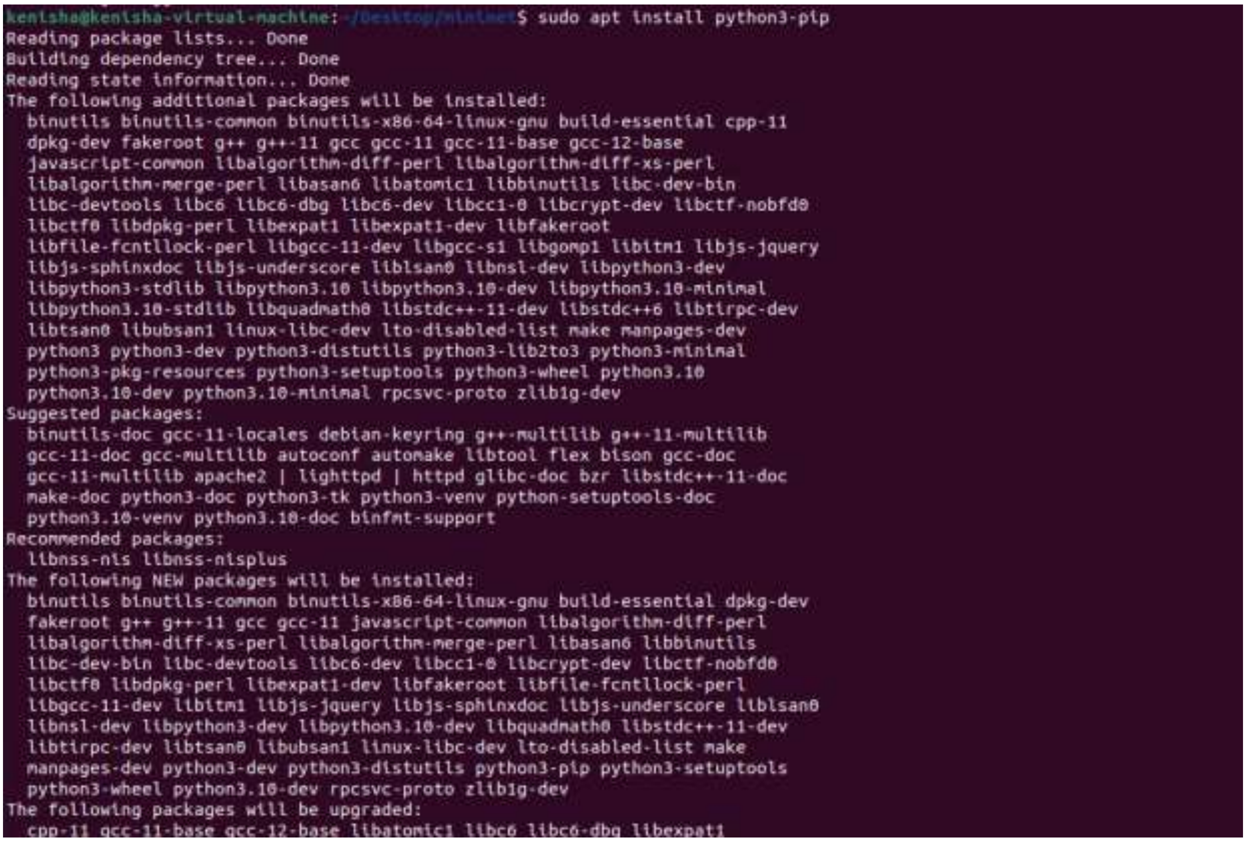


Figure 1: Network diagram



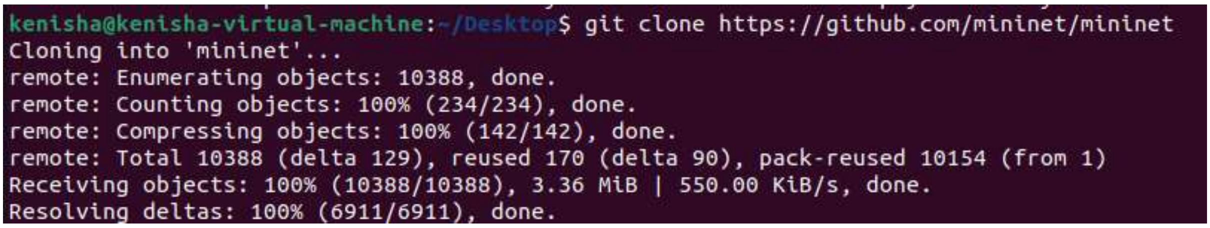
```
kenisha@kenisha-virtual-machine:~/Desktop/mininet$ sudo apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb
  git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 504 not upgraded.
Need to get 4,146 kB of archives.
After this operation, 21.0 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://lk.archive.ubuntu.com/ubuntu jammy/main amd64 liberror-perl all 0.17029-1 [26.5 kB]
Get:2 http://lk.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all 1:2.34.1-1ubuntu1.11 [955 kB]
Get:3 http://lk.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:2.34.1-1ubuntu1.11 [3,165 kB]
Fetched 4,146 kB in 19s (214 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 161752 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17029-1_all.deb ...
Unpacking liberror-perl (0.17029-1) ...
Selecting previously unselected package git-man.
Preparing to unpack .../git-man_1%3a2.34.1-1ubuntu1.11_all.deb ...
Unpacking git-man (1:2.34.1-1ubuntu1.11) ...
Selecting previously unselected package git.
Preparing to unpack .../git_1%3a2.34.1-1ubuntu1.11_amd64.deb ...
Unpacking git (1:2.34.1-1ubuntu1.11) ...
Setting up liberror-perl (0.17029-1) ...
Setting up git-man (1:2.34.1-1ubuntu1.11) ...
Setting up git (1:2.34.1-1ubuntu1.11) ...
Processing triggers for man-db (2.10.2-1) ...
```

Figure 2: git clone Installation



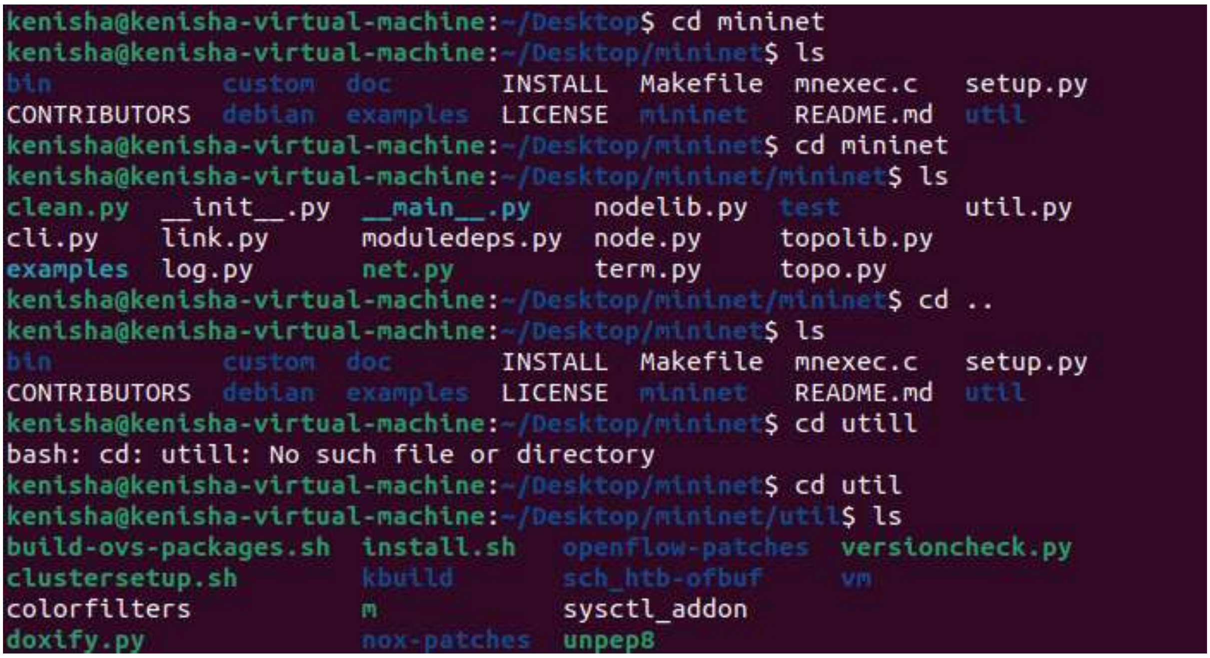
```
kenisha@kenisha-virtual-machine:~/Desktop/mininet$ sudo apt install python3-pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential cpp-11
  dpkg-dev fakeroot g++ g++-11 gcc gcc-11 gcc-11-base gcc-12-base
  javascript-common libalgorithm-diff-perl libalgorithm-diff-xs-perl
  libalgorithm-merge-perl libasan6 libatomic1 libbinutils libc-dev-bin
  libc-devtools libc6 libc6-dbg libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0
  libctf0 libdpkg-perl libexpat1 libexpat1-dev libfakeroot
  libfile-fcntllock-perl libgcc-11-dev libgcc-s1 libgomp1 libitm1 libjs-jquery
  libjs-sphinxdoc libjs-underscore liblsan0 libnsl-dev libpython3-dev
  libpython3-stdlib libpython3.10 libpython3.10-dev libpython3.10-minimal
  libpython3.10-stdlib libquadmath0 libstdc++-11-dev libstdc++6 libtirpc-dev
  libtsan0 libubsan1 linux-libc-dev lto-disabled-list make manpages-dev
  python3 python3-dev python3-distutils python3-lib2to3 python3-minimal
  python3-pkg-resources python3-setuptools python3-wheel python3.10
  python3.10-dev python3.10-minimal rpcsvc-proto zlib1g-dev
Suggested packages:
  binutils-doc gcc-11-locales debian-keyring g++-multilib g++-11-multilib
  gcc-11-doc gcc-multilib autoconf automake libtool flex bison gcc-doc
  gcc-11-multilib apache2 | lighttpd | httpd glibc-doc bzr libstdc++-11-doc
  make-doc python3-doc python3-tk python3-venv python-setuptools-doc
  python3.10-venv python3.10-doc binfmt-support
Recommended packages:
  libnss-nis libnss-nisplus
The following NEW packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential dpkg-dev
  fakeroot g++ g++-11 gcc gcc-11 javascript-common libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan6 libbinutils
  libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0
  libctf0 libdpkg-perl libexpat1-dev libfakeroot libfile-fcntllock-perl
  libgcc-11-dev libitm1 libjs-jquery libjs-sphinxdoc libjs-underscore liblsan0
  libnsl-dev libpython3-dev libpython3.10-dev libquadmath0 libstdc++-11-dev
  libtirpc-dev libtsan0 libubsan1 linux-libc-dev lto-disabled-list make
  manpages-dev python3-dev python3-distutils python3-pip python3-setuptools
  python3-wheel python3.10-dev rpcsvc-proto zlib1g-dev
The following packages will be upgraded:
  cpp-11 gcc-11-base gcc-12-base libatomic1 libc6 libc6-dbg libexpat1
```

Figure 3: python3-pip Installation



```
kenisha@kenisha-virtual-machine:~/Desktop$ git clone https://github.com/mininet/mininet
Cloning into 'mininet'...
remote: Enumerating objects: 10388, done.
remote: Counting objects: 100% (234/234), done.
remote: Compressing objects: 100% (142/142), done.
remote: Total 10388 (delta 129), reused 170 (delta 90), pack-reused 10154 (from 1)
Receiving objects: 100% (10388/10388), 3.36 MiB | 550.00 KiB/s, done.
Resolving deltas: 100% (6911/6911), done.
```

Figure 4: git clone



```
kenisha@kenisha-virtual-machine:~/Desktop$ cd mininet
kenisha@kenisha-virtual-machine:~/Desktop/mininet$ ls
bin           custom  doc       INSTALL  Makefile  mnexec.c   setup.py
CONTRIBUTORS  debian  examples  LICENSE  mininet   README.md  util
kenisha@kenisha-virtual-machine:~/Desktop/mininet$ cd mininet
kenisha@kenisha-virtual-machine:~/Desktop/mininet/mininet$ ls
clean.py  __init__.py  __main__.py    nodelib.py  test        util.py
cli.py    link.py      moduledeps.py  node.py     topolib.py
examples  log.py       net.py         term.py     topo.py
kenisha@kenisha-virtual-machine:~/Desktop/mininet/mininet$ cd ..
kenisha@kenisha-virtual-machine:~/Desktop/mininet$ ls
bin           custom  doc       INSTALL  Makefile  mnexec.c   setup.py
CONTRIBUTORS  debian  examples  LICENSE  mininet   README.md  util
kenisha@kenisha-virtual-machine:~/Desktop/mininet$ cd utill
bash: cd: utill: No such file or directory
kenisha@kenisha-virtual-machine:~/Desktop/mininet$ cd util
kenisha@kenisha-virtual-machine:~/Desktop/mininet/util$ ls
build-ovs-packages.sh  install.sh   openflow-patches  versioncheck.py
clustersetup.sh        kbuild       sch_htb-ofbuf     vm
colorfilters           m            sysctl_addon
doxify.py              nox-patches  unpep8
```

Figure 5: Navigating Through mininet

```
kenisha@kenisha-virtual-machine:~/Desktop/mininet/util$ sudo ./install.sh -a
Detected Linux distribution: Ubuntu 22.04 jammy amd64
sys.version_info(major=3, minor=10, micro=12, releaselevel='final', serial=0)
Detected Python (python3) version 3
Installing all packages except for -eix (doxypy, ivs, nox-classic)...
Install Mininet-compatible kernel if necessary
Hit:1 http://lk.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:3 http://lk.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://lk.archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
Reading package lists...
Building dependency tree...
Reading state information...
linux-image-5.19.0-32-generic is already the newest version (5.19.0-32.33~22.04.1).
linux-image-5.19.0-32-generic set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 484 not upgraded.
Installing Mininet dependencies
Reading package lists...
Building dependency tree...
Reading state information...
gcc is already the newest version (4:11.2.0-1ubuntu1).
gcc set to manually installed.
make is already the newest version (4.3-4.1build1).
make set to manually installed.
net-tools is already the newest version (1.60+git20181103.0eebece-1ubuntu5).
net-tools set to manually installed.
psmisc is already the newest version (23.4-2build3).
psmisc set to manually installed.
socat is already the newest version (1.7.4.1-3ubuntu4).
socat set to manually installed.
telnet is already the newest version (0.17-44build1).
telnet set to manually installed.
iperf is already the newest version (2.1.5+dfsg1-1).
iperf set to manually installed.
The following additional packages will be installed:
  blt libtk8.6 libutempter0 ncurses-term openssh-client openssh-server
  openssh-sftp-server pycodestyle python3-astroid python3-isort
  python3-lazy-object-proxy python3-logilab-common python3-mccabe
  python3-mypy-extensions python3-platformdirs python3-pycodestyle
```

Figure 6: Running an Executable

```
kenisha@kenisha-virtual-machine:~/Desktop/mininet$ sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches

s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 1.675 seconds
```

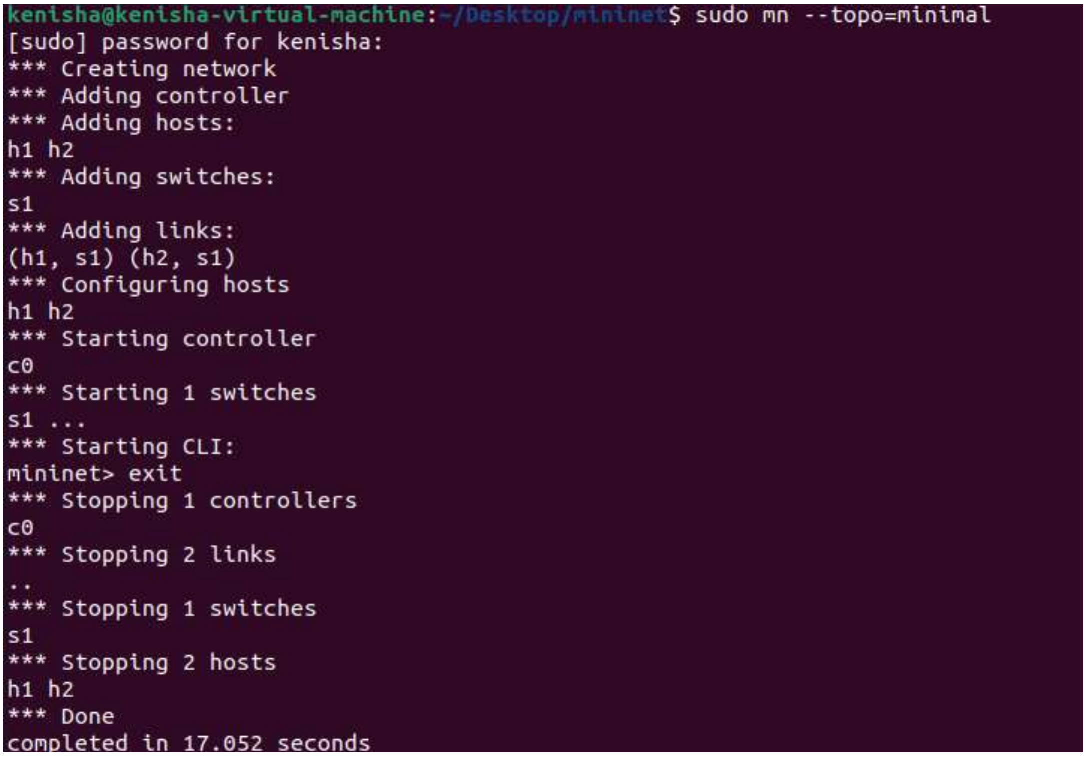
Figure 7: Creating a Network Topology

```
$ sudo mn -c
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-controller ovs-testcontroller udpbwtest mnexec ivs ryu-manager  2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-controller ovs-testcontroller udpbwtest mnexec ivs ryu-manager  2> /dev/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
***  Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-_.[:alnum:]]+-eth[[:digit:]]+)'
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
```

Figure 8: Cleaning the Topology

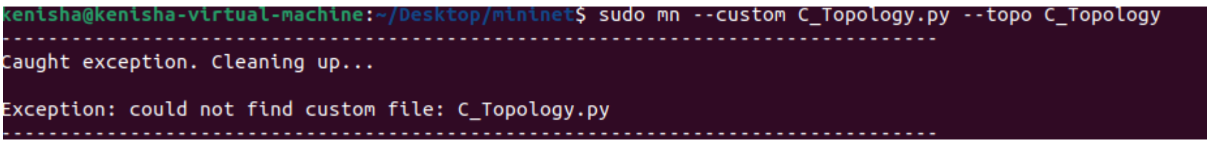
Node and interface detail

✓ First, I set up a simple network scenario of Mininet. By typing the command **sudo mn –topo=minimal,** the topology comprising of one OpenFlow kernel switch connected to two hosts and the OpenFlow reference controller was created. However, the clear design gave me an opportunity to think about the basics of the networks and their setups.



```
kenisha@kenisha-virtual-machine:~/Desktop/mininet$ sudo mn --topo=minimal
[sudo] password for kenisha:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 17.052 seconds
```

Figure 9: Started the Topology



```
kenisha@kenisha-virtual-machine:~/Desktop/mininet$ sudo mn --custom C_Topology.py --topo C_Topology
--------------------------------------------------------------------------------
Caught exception. Cleaning up...

Exception: could not find custom file: C_Topology.py
--------------------------------------------------------------------------------
```

Figure 10: Cleaning the Topology

- ✓ In order to get a list of nodes in my network, I used the **mininet> nodes** command that gives details of each node that was present. I implemented the aspects of the network topology using **mininet> net**. With such instructions, I was able to confirm the details of the topology configuration. I copied and pasted the above outputs so that you can have a visual feel of the actual network topology.

```
mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet>
```

Figure 11: nodes in the mininet

- ✓ Hosting mininet, using **mininet>h1 ifconfig -a**, I confirmed that the network interfaces of the hosts and the switch was **mininet>s1 ifconfig -a**. In addition, by typing this command, **mininet> h1 ps –a**. I was able to review the processes running within the host h1. The instructions also provided detailed description of the network settings and host processes. It is why I included related screen grabs to enhance the message's comprehension.

```
mininet> h1 ifconfig -a
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::d8b6:27ff:fedf:7c22  prefixlen 64  scopeid 0x20<link>
        ether da:b6:27:df:7c:22  txqueuelen 1000  (Ethernet)
        RX packets 31  bytes 3542 (3.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9  bytes 726 (726.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 12: IP configuration of Host 1

```
mininet> h2 ifconfig -a
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.2  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::cc:fff:fee9:2504  prefixlen 64  scopeid 0x20<link>
        ether 02:cc:0f:e9:25:04  txqueuelen 1000  (Ethernet)
        RX packets 36  bytes 3946 (3.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 11  bytes 866 (866.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 13: IP configuration of Host 2

```
mininet> s1 ifconfig -a
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.232.133  netmask 255.255.255.0  broadcast 192.168.232.255
        inet6 fe80::bfb:32a1:7e07:a361  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:a8:23:a3  txqueuelen 1000  (Ethernet)
        RX packets 1503238  bytes 2201010391 (2.2 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 453624  bytes 27488964 (27.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 660  bytes 71330 (71.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 660  bytes 71330 (71.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ovs-system: flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether b6:bc:8a:e4:26:1b  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

s1: flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether c2:6a:a3:ad:17:4c  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 17  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

s1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::100f:68ff:fe03:1ea6  prefixlen 64  scopeid 0x20<link>
        ether 12:0f:68:03:1e:a6  txqueuelen 1000  (Ethernet)
        RX packets 10  bytes 796 (796.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 34  bytes 3789 (3.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

s1-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::ac2f:45ff:fee1:6dfa  prefixlen 64  scopeid 0x20<link>
```

Figure 14: IP configuration of switch 1

```
mininet> h1 ps -a
    PID TTY          TIME CMD
   1540 tty2     00:00:00 gnome-session-b
  25295 pts/0    00:00:00 python3
  25390 pts/0    00:00:00 sudo
  25392 pts/1    00:00:00 mn
 147222 pts/0    00:00:00 sudo
 147224 pts/2    00:00:00 mn
 147272 pts/3    00:00:00 controller
 147297 pts/4    00:00:00 ps
```

Figure 15: Processes running in the host

Connectivity Testing

Ping Testing

- ✓ The connectivity between the two hosts was as determined by the **'mininet > h1 ping -c5 h2'** command. I entered this command to send five ping requests to host h2. On each try, I noted the ping values in the table below and captured the screenshots of the output results. The results of the test showed the ability of the hosts to interact with each other.

```
mininet> h1 ping -c 5 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=5.47 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.847 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.094 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.177 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.078 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4065ms
rtt min/avg/max/mdev = 0.078/1.332/5.468/2.087 ms
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 2 links
```

Figure 16: Pinging test from Host 1 to Host 2

✓ It is possible to observe variability in ping times in the figure due to a network delay. Network delay refers to the time taken for a data packet to go from one device and back. It could arise from a number of factors, including:

- In a network with high traffic, it can be filled up with too many traffic that will slow down transfer of data packets. The more cars are on the road the longer time each car they take to get to the intended or required destination.
- Of course, a data packet travelling a longer distance will do this in a longer time. Signal has to travel a longer physical distance and this is what causes this.
- Transmitted data must be understandable by routers and other network devices during the data packets' travel. Depending on duration of time taken in this process it might add on to total time taken.
- In addition, the dynamics of the selected network equipment may also influence communication latency. This might be arising from hardware that is either overworked, or the computers which are outdated.

Answer the following Questions

Question 01

01. Provide three examples of east-west traffic and three examples of north-south traffic in a modern data centre.

- **East – West Traffic**

  The definition also shows that data transport operating in the east-west direction concerns traffic that transverses within the same data centre. For instance, take into account:

  Data Replication: Data replication in storage devices within a particular data centre ensures redundant data and availability is achieved while data consistency, and disaster recovery is promoted.

  Inter- Service Communication: In a micro services architecture, a large number of services often interact with other services. For example if a payment processing company is using an authentication service to confirm the transaction data there could be lot of internal east-west traffic.

  VM-to-VM Communication: Communication between VMs is the transfer of information between Virtual machines which are living on the same physical host. An application server virtual machine may talk to a database server VM in order to acquire data.

- **North-South Traffic**

  The movement of data into or out of a data centre is called north-south traffic. Some examples are as follows:

  Cloud Data Backup: It is much common already for data centres to contract out from external service providers in the cloud to contain copies of crucial data. This is a type of traffic known as North-South traffic where data is shifted from the data centre to cloud.

  User Access to Cloud Service: Employees contribute to the formation of north-south traffic if they apply their smart mobile devices to obtain data from cloud applications oriented to the company. An employee working from home having accessed a company's cloud stored CRM database.

  Client Request to Web Servers: North-south traffic occurs when a user asks for a website; information goes from the user's device to the web server at the data centre externally.

(278 words)

02. Premise of SDN is that the plethora of control plane protocols (discussed in the week 2's lecture slides 14-18) creates complexity that is not necessary in all networks. Give an example of a network where the complexity is unnecessary and explain why this is so.

- Small office networks employ complex routing protocols such as; BGP that are normally a preserve of ISPs or large organization networks. Small networks, with less number of devices and less complicated topology, can well use RIP routing or static routing in its true sense**.**

- If BGP is employed further cost is incidental to configuration, maintenance, as well as problem solving. BGP-compatible gear costs higher than a standard gear and they need specialized personnel and operational costs are also high. Also, the administration of network is somehow more complex due to the complexity that is incurred by BGP. Small office managers may not have the knowledge on how to solve the BGP problems, which sometimes may result to network downtime.

- Using RIP or static routing and other similar routing, small organization can suffice their network requirement without incurring further expenses or introducing more sophistication.

- This makes it possible for the network to always be both dependable and effective, relatively cheap to maintain, and easy to manage. It is unadvisable to complicate network solutions because it only leads to discouraging complications and coupled with improper management of resources.

(186 words)

03. What is OpenFlow?

- Software Defined Networking calls for the application of a communication protocol known as OpenFlow. In implementation it enables the overall exchange of information between the data and the control plane within switches and routers amongst others. While remaining capable of deterministic forwarding, OpenFlow enables programmability and flexibility by giving a central controller the ability to dynamically manage and direct data paths. The graphical separation of the control and the data plane enables network managers to architect traffic flow patterns to suit unique demands thereby increasing ease of management and efficiency.

- Recognizing that the general techniques provided by OpenFlow can increases costs while providing tremendous benefits, using the ability to control the traffic, set up policies and react to alterations in the network instantaneously may help organizations to cut costs while increasing productivity. Its standardized methodology results in a more adaptable and transparent network environment for device interoperability among numerous suppliers. SDN is only possible with OpenFlow technology, which even in today's complex network structures today enables simple network upgrades, more scalability, and better manageability.

- In summary, OpenFlow means the flexible network control and administration in accordance with key principles and protocols of network management. It assists the businesses to maintain their network operations within the optimum levels of effectiveness and versatility while minimizing the issues of the complexity of a network as well as the cost aspect. Being an important component of SDN, OpenFlow helps to progress from more static network control to a more flexible one.

(247 Words)

04. List essential characteristics of a network to be considered as an SDN network.

- **Centralised Control:** Software-defined networking rely in a centralised controller that act as the control point for the whole system. This unburdens the network management and optimizes the usage of the resources available for other purposes.

- **Flexibility and Scalability:** By implementation of SDN, there is an increase in the abilities of networks to expand and develop. More devices and services can be readily integrated into the network without causing much interruption while the flow of traffic can also change quickly to meet the new demands.

- **Separation of Control and Data Planes**: SDN separates the control plane that is responsible of deciding where traffic should be forwarded and the data plane that is responsible of actually forwarding the traffic. This leads to more obtaining centralised control as well as affording more flexible administration.

- **Programmability**: Being as programmable as they are, it is quite probable for network managers to perform complex tasks, including repetitive ones, and create further complicated regulatory systems. This means that the amount of human configuration is minimized and the flexibility to change the network is enhanced.

- **Use of Standardised Protocols**: Standardised technologies such as OpenFlow and other protocols that makes it easier for end devices across numerous vendors and create more flexible and open environment for the network are used in SDN.

(213 words)

Conclusion

- ✓ In the conclusion, I carefully summarized the principal findings and conclusions derived from the completed assignments, giving particular emphasis to the primary roles that network security and SDN assume in the modern networking environments. I spoke on how SDN makes the network environments more flexible, scalable, and centralized apart from discussing on the importance of enhanced security measures that ensure data security and network sanctity.

- ✓ Furthermore, based on the intended learning outcomes achieved as part of the assignments, there were tasks such as understanding the nature of SDN architectural design, elements, and the protocols inherent in it. I also learned practical applications of networks security for instance identifying the vulnerabilities and coming up with ways to counter them.

References

- East-west Traffic
  https://www.pomerium.com/glossary/east-west-traffic
- North-south Traffic
  https://www.pomerium.com/glossary/north-south-traffic
- Wright, G. (2023) OpenFlow.
  https://www.techtarget.com/whatis/definition/OpenFlow
- Control Plane vs. Data Plane: What Are The Differences?
  https://www.splunk.com/en_us/blog/learn/control-plane-vs-data-plane.html