


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:	383
<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:	01/02/2025

Task 6.2C
Table of Content

Part A.....	04
Installation Jpert.....	06
Installation (LOIC).....	09
DDoS Attack.....	11

Table of Figures

Figure 01.....	04
Figure 02.....	04
Figure 03.....	04
Figure 04.....	05
Figure 05.....	06
Figure 06.....	07
Figure 07.....	07
Figure 08.....	08
Figure 09.....	09
Figure 10.....	09
Figure 11.....	10
Figure 12.....	11
Figure 13.....	11
Figure 14.....	12
Figure 15.....	12
Figure 16.....	13
Figure 17.....	14
Figure 18.....	14
Figure 19.....	15
Figure 20.....	15
Figure 21.....	16

Part A

- I was able to successfully run onos in the Docker container first.

```
root@kenisha-virtual-machine:/home/kenisha/Desktop/6.2C# docker run -t -d -p 8182:8181 -p 8182:8181 -p 5005:5005 -p 838:838 --name onos_new22 onosproject/onos:2.7.8
36176886c08b55210ae788fa2c88f806babfd6077b7d4e67b14547dc30cadae
root@kenisha-virtual-machine:/home/kenisha/Desktop/6.2C#
```

Figure 01: Using Docker to operate onos

- I used the browser to log into the Onos after that.



Figure 02: Accessing Onos

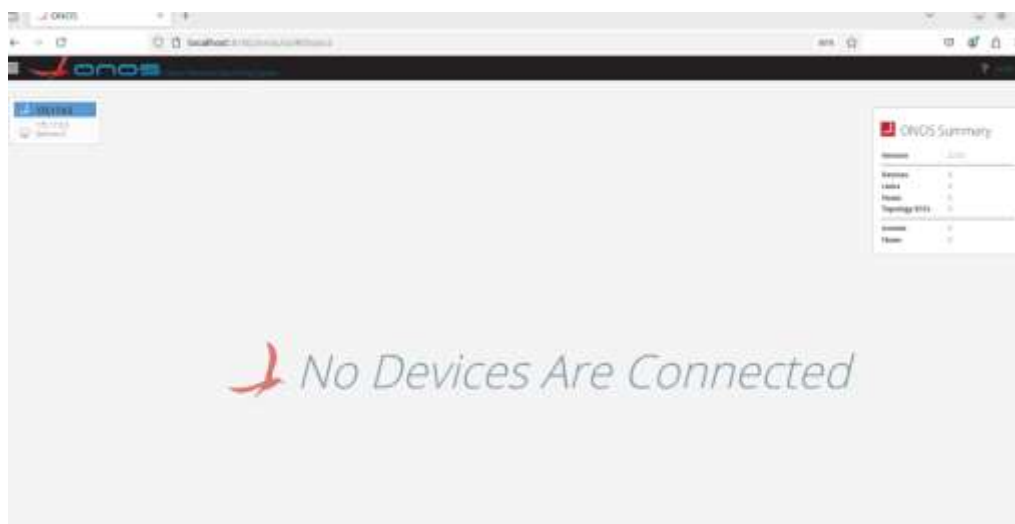


Figure 03: The main page of Onos

```

kensha@kensha-virtual-machine:~/Desktop/k.20$ sudo mn --custom C_Topology.py --topo=C_Topology --controller=remote,ip=172.17.0.2,port=6653 --switch=ovs,protocols=OpenFlow13 -
-mac
*** 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 controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet> pingall
*** 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)

```

Figure 04: Mininet operating successfully

- After testing mininet execution I performed connectivity tests through the ping command.

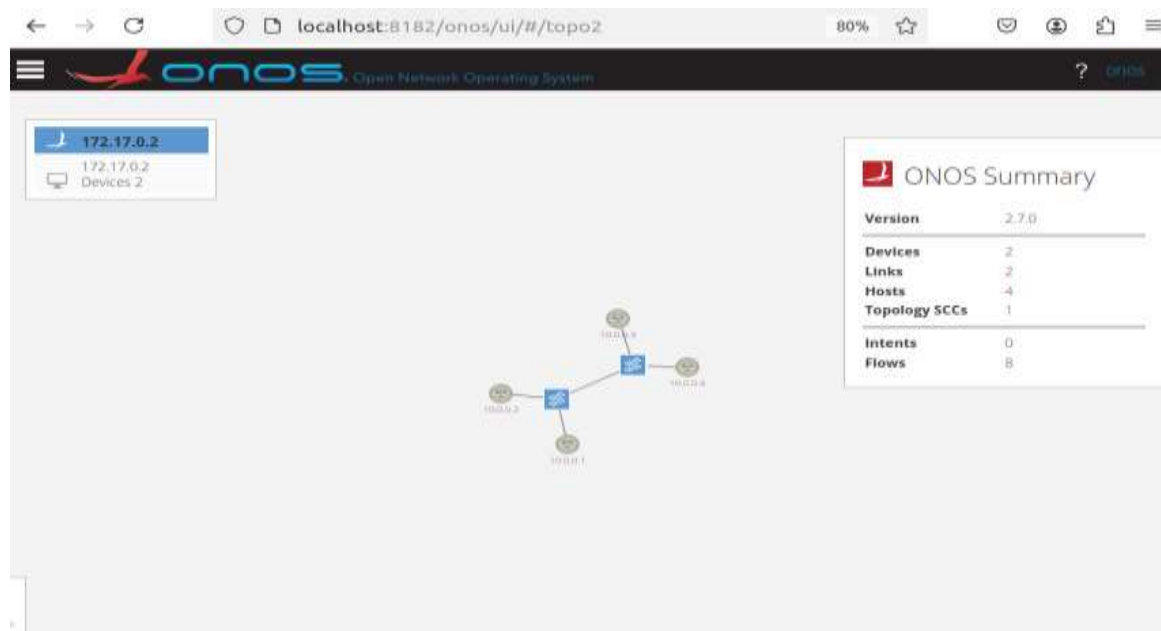


Figure 05: linked network topology

- I used the `pingall` command to build network connections between my hosts and the onos devices.

Installation Jpert

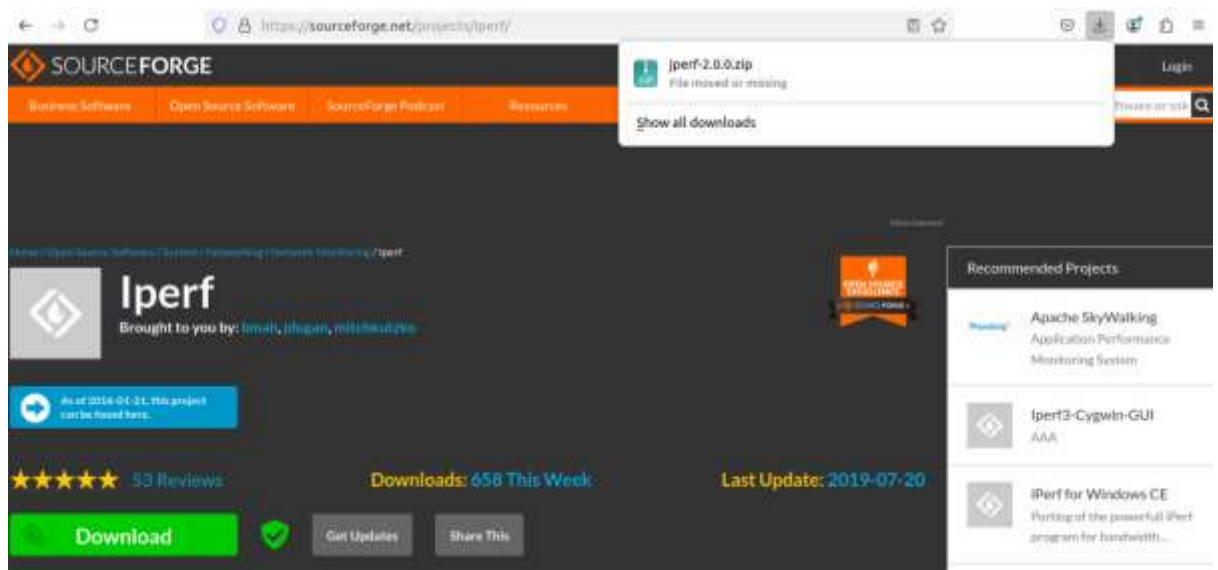


Figure 06: JPerf installation via SourceForge

- Setting up JPERF and releasing the files.

```
kentsha@kentsha-virtual-machine: ~/Desktop/9.0 $ unzip jperf-2.0.0.zip
Archive: jperf-2.0.0.zip
  creating: jperf-2.0.0/
  creating: jperf-2.0.0/bin/
  creating: jperf-2.0.0/lib/
  inflating: jperf-2.0.0/bin/jperf.exe
  inflating: jperf-2.0.0/jperf-2.0.0.jar
  inflating: jperf-2.0.0/jperf.bat
  inflating: jperf-2.0.0/jperf.sh
  inflating: jperf-2.0.0/lib/forms-1.1.0.jar
  inflating: jperf-2.0.0/lib/jcommon-1.0.10.jar
  inflating: jperf-2.0.0/lib/jfreechart-1.0.6.jar
  inflating: jperf-2.0.0/lib/swingx-2000_02_03.jar
kentsha@kentsha-virtual-machine: ~/Desktop/9.0 $ ls
C_Topology.py  jperf-2.0.0  jperf-2.0.0.zip  jperf-writer
kentsha@kentsha-virtual-machine: ~/Desktop/9.0 $ unzip jperf-2.0.0
Archive: jperf-2.0.0.zip
replace jperf-2.0.0/bin/jperf.exe? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: jperf-2.0.0/bin/jperf.exe
replace jperf-2.0.0/jperf-2.0.0.jar? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: jperf-2.0.0/jperf-2.0.0.jar
replace jperf-2.0.0/jperf.bat? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: jperf-2.0.0/jperf.bat
replace jperf-2.0.0/jperf.sh? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: jperf-2.0.0/jperf.sh
replace jperf-2.0.0/lib/forms-1.1.0.jar? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: jperf-2.0.0/lib/forms-1.1.0.jar
replace jperf-2.0.0/lib/jcommon-1.0.10.jar? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: jperf-2.0.0/lib/jcommon-1.0.10.jar
replace jperf-2.0.0/lib/jfreechart-1.0.6.jar? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: jperf-2.0.0/lib/jfreechart-1.0.6.jar
replace jperf-2.0.0/lib/swingx-2000_02_03.jar? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: jperf-2.0.0/lib/swingx-2000_02_03.jar
kentsha@kentsha-virtual-machine: ~/Desktop/9.0 $ cd jperf-2.0.0
kentsha@kentsha-virtual-machine: ~/Desktop/9.0/jperf-2.0.0 $ ls
bin  jperf-2.0.0.jar  jperf.bat  jperf.sh  lib
kentsha@kentsha-virtual-machine: ~/Desktop/9.0/jperf-2.0.0 $ sudo chmod +x jperf.sh
[sudo] password for kentsha:
kentsha@kentsha-virtual-machine: ~/Desktop/9.0/jperf-2.0.0 $ ./jperf.sh
```

Figure 07: launching the program after unzipping it

- Executive process launch of the JPerf utility succeeded once I ran the program.

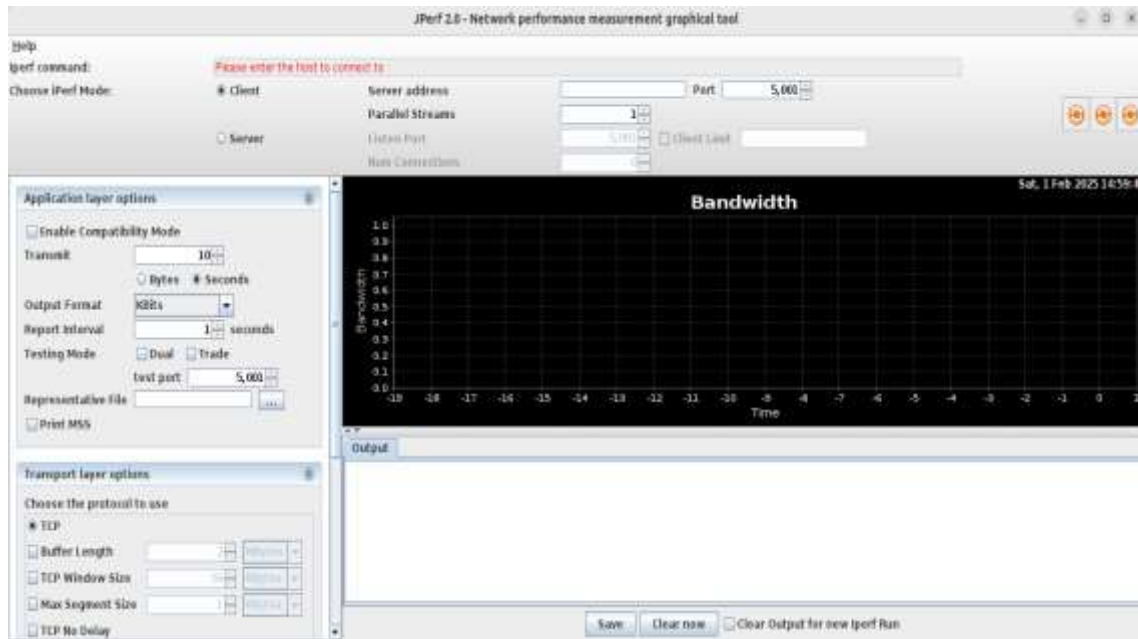


Figure 08: Jperf had a successful launch.

Installation (LOIC) Low Orbit Ion Cannon

- Through the LOIC application I am currently conducting a DDoS attack.

```
kenisha@kenisha-virtual-machine:~/Desktop/LOIC$ sudo apt-get install mono-complete
[sudo] password for kenisha:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
mono-complete is already the newest version (6.8.0.105+dfsg-3.2).
The following packages were automatically installed and are no longer required:
  libreoffice-ogltrans linux-headers-6.8.0-49-generic
  linux-hwe-6.8-headers-6.8.0-49 linux-hwe-6.8-tools-6.8.0-49
  linux-image-6.8.0-49-generic linux-modules-6.8.0-49-generic
  linux-modules-extra-6.8.0-49-generic linux-tools-6.8.0-49-generic
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 211 not upgraded.
kenisha@kenisha-virtual-machine:~/Desktop/LOIC$ ls
```

Figure 09: Setting up LOIC

- The tool extraction process took place immediately following installation.

```
kenisha@kenisha-virtual-machine:~/Desktop/LOIC$ ls
LOIC
kenisha@kenisha-virtual-machine:~/Desktop/LOIC$ cd LOIC
kenisha@kenisha-virtual-machine:~/Desktop/LOIC/LOIC$ ls
CONTRIBUTING.md  LICENSE.md      loic.sh         README.md      WORK_IN_PROGRESS
help             loic-net4.0.sh  README.BeSquare src
kenisha@kenisha-virtual-machine:~/Desktop/LOIC/LOIC$ chmod +x loic.sh
kenisha@kenisha-virtual-machine:~/Desktop/LOIC/LOIC$ ./loic.sh run
/usr/bin/mono
Gtk-Message: 16:26:20.358: Failed to load module "canberra-gtk-module"
```

Figure 10: conducted with success LOIC.

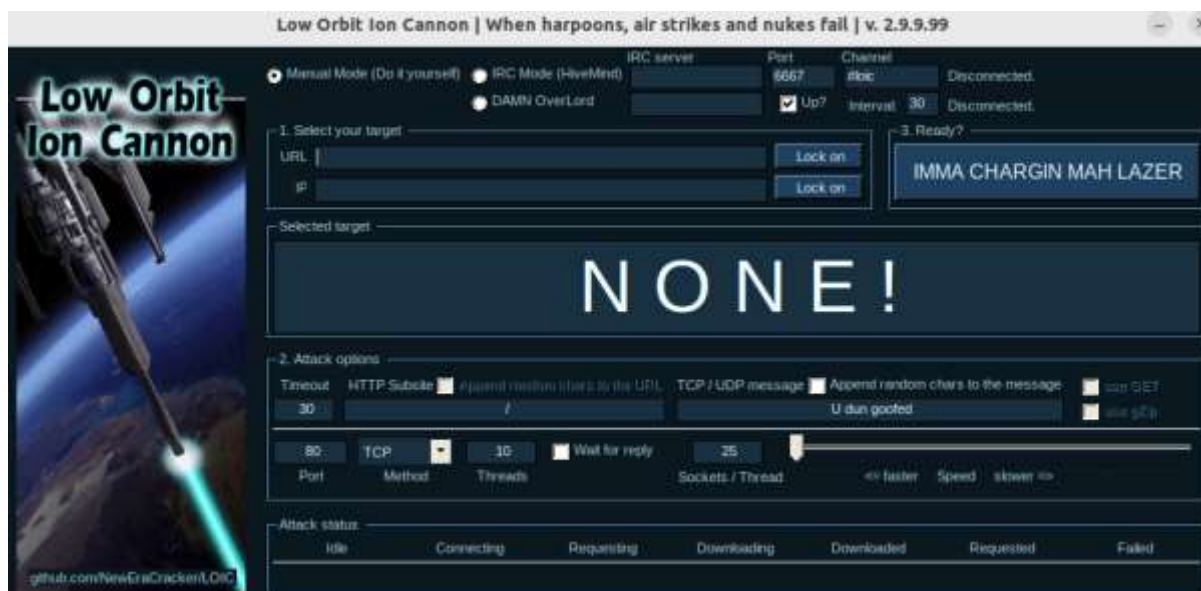


Figure 11: operating LOIC

- I managed to acquire the LOIC GUI interface successfully.

DDoS Attack

- The experimental setup consisted of H1 functioning as the host machine with H2 running as the client system supported by H3 running the LOIC program.

```

"Node: h1"
root@kenisha-virtual-machine:/home/kenisha/Desktop/6.2C# ls
C_Topology.py  jperf-2.0.0  jperf-2.0.0.zip  jperf-master
root@kenisha-virtual-machine:/home/kenisha/Desktop/6.2C# cd jperf-2.0.0
root@kenisha-virtual-machine:/home/kenisha/Desktop/6.2C/jperf-2.0.0# ls
bin  jperf-2.0.0.jar  jperf.bat  jperf.sh  lib
root@kenisha-virtual-machine:/home/kenisha/Desktop/6.2C/jperf-2.0.0# ./jperf.sh

```

Figure 12: Using H1 as the operating system

- JPerf functionality on H1 required suitable parameter settings to become an operational server. The system required setup of a listening port along with other necessary consumption parameters. The server currently operated in a state where it could receive client data. The client believed these configurations delivered precise performance measurement results. Figure 1 displays how the H1 needs to be configured to serve as a server system.

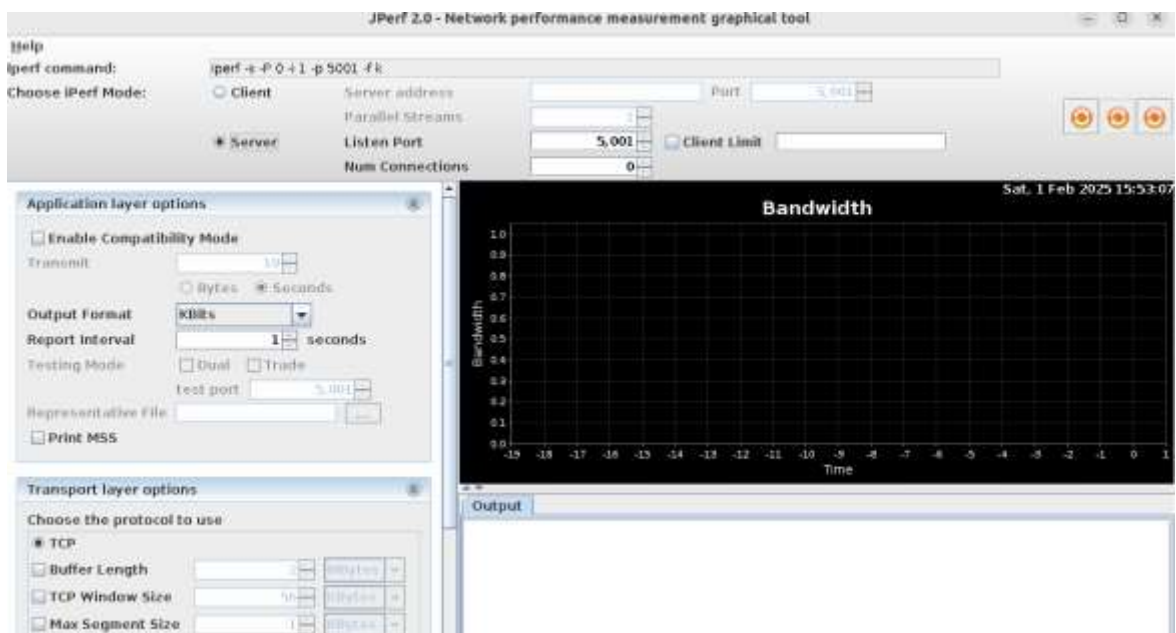


Figure 13: H1 JPerf

- I provided JPerf all necessary server information including the IP address and test parameters to allow it operate as a client on h2. The client had the willingness to share performance measurement data with the server on h1. Precise data transfer and measurement required these setup configurations. The host H2 could start testing.

```

"Node: h2"
root@kenisha-virtual-machine:/home/kenisha/Desktop/6.2C# cd jperf-2.0.0
root@kenisha-virtual-machine:/home/kenisha/Desktop/6.2C/jperf-2.0.0# ls
bin  jperf-2.0.0.jar  jperf.bat  jperf.sh  lib
root@kenisha-virtual-machine:/home/kenisha/Desktop/6.2C/jperf-2.0.0# ./jperf.sh

```

Figure 14: utilizing H2 as a client

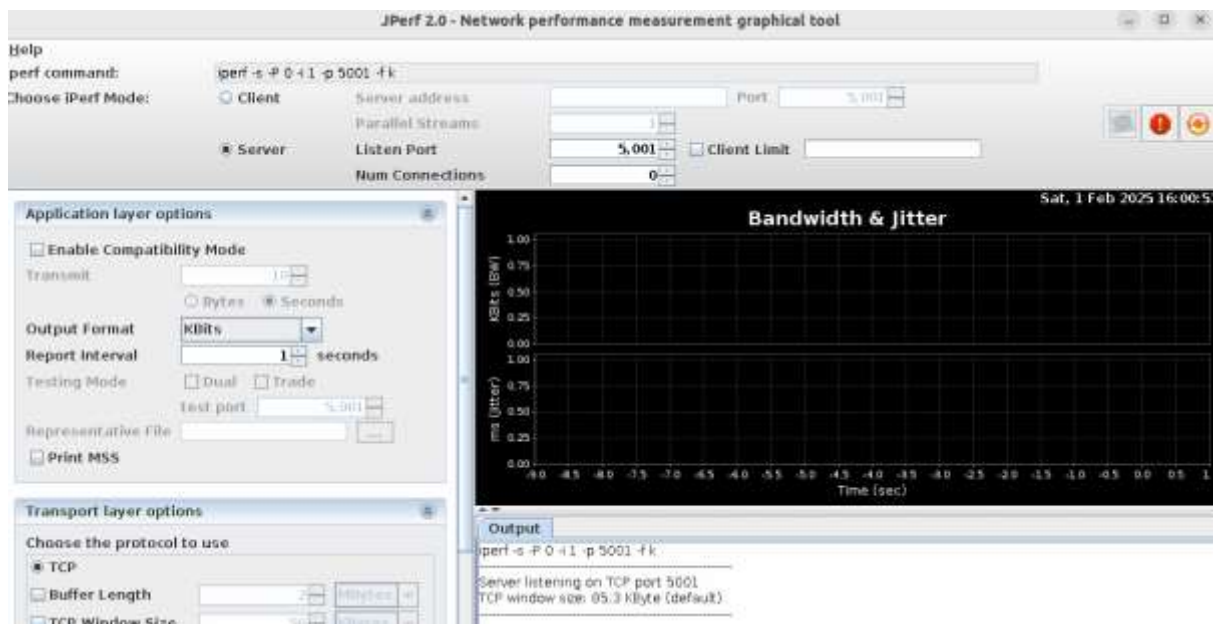


Figure 15: Using H2 as a client, Jperf

- The client computer is on H2.

- The H3 will serve as my trigger to start the DDoS attack.

```

"Node: h3"

root@kenisha-virtual-machine:/home/kenisha/Desktop/6.2C# ls
C_Topology.py  jperf-2.0.0  jperf-2.0.0.zip  jperf-master  'TCP client.png'  'TCP server.png'
root@kenisha-virtual-machine:/home/kenisha/Desktop/6.2C# cd ..
root@kenisha-virtual-machine:/home/kenisha/Desktop# ls
6.2C  bazel  C_Topology.py  LOIC  mininet  oflops  oftest  openflow  pox  simletree_highlevel.py
root@kenisha-virtual-machine:/home/kenisha/Desktop# cd LOIC
root@kenisha-virtual-machine:/home/kenisha/Desktop/LOIC# ls
LOIC
root@kenisha-virtual-machine:/home/kenisha/Desktop/LOIC# cd LOIC
root@kenisha-virtual-machine:/home/kenisha/Desktop/LOIC/LOIC# ls
CONTRIBUTING.md  Help  LICENSE.md  loic-net4.0.sh  loic.sh  README.BeSquare  README.md  src  WORK_IN_PROGRESS
root@kenisha-virtual-machine:/home/kenisha/Desktop/LOIC/LOIC# ./loic.sh
Usage: ./loic.sh <install|update|run>
root@kenisha-virtual-machine:/home/kenisha/Desktop/LOIC/LOIC# ./loic.sh run
/usr/bin/mono
Gtk-Message: 17:01:55.955: Failed to load module "canberra-gtk-module"

```

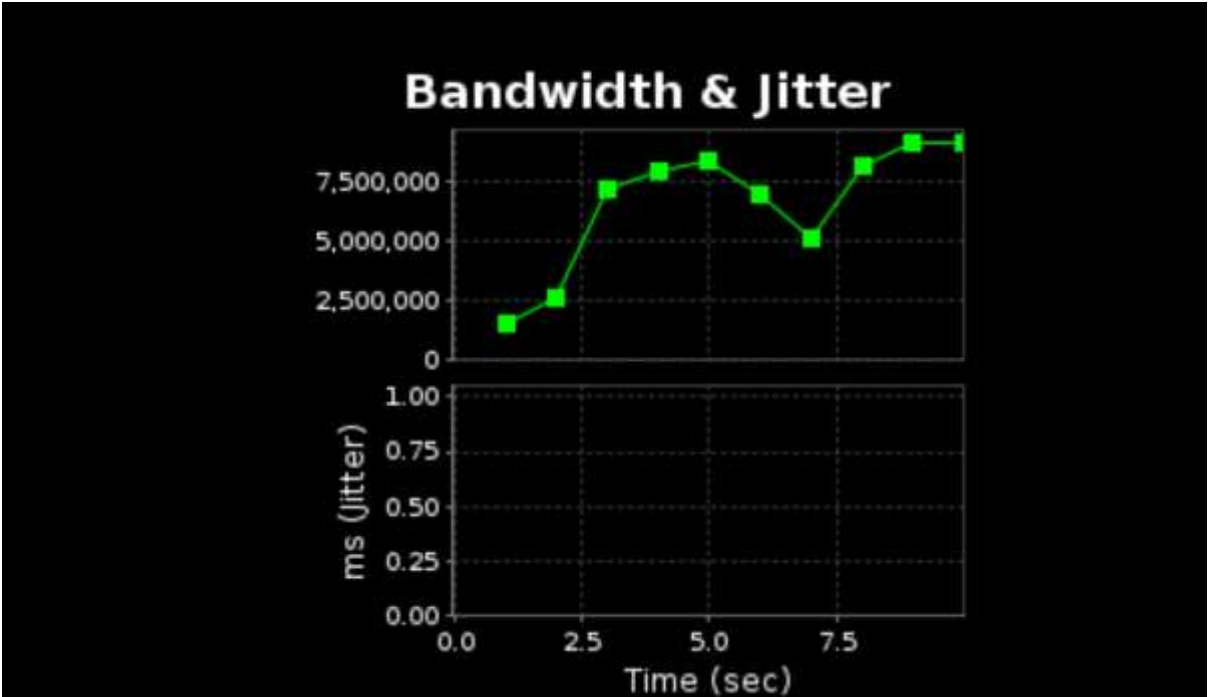
Figure 16: H3 operating as LOIC

- Server targeting becomes an essential step when working with the LOIC tool from H3.



Figure 17: LOIC began to run.

- During H3 I deployed the attack before implementing the LOIC protocol. Users must construct an H2 client application which connects to a server system to initiate the monitoring process.



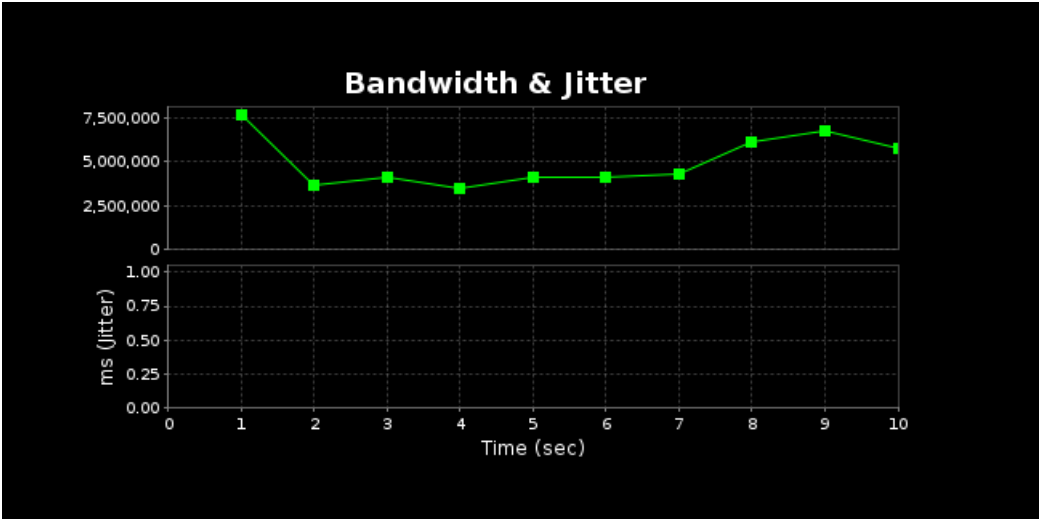


Figure 19: DDoS and TCP

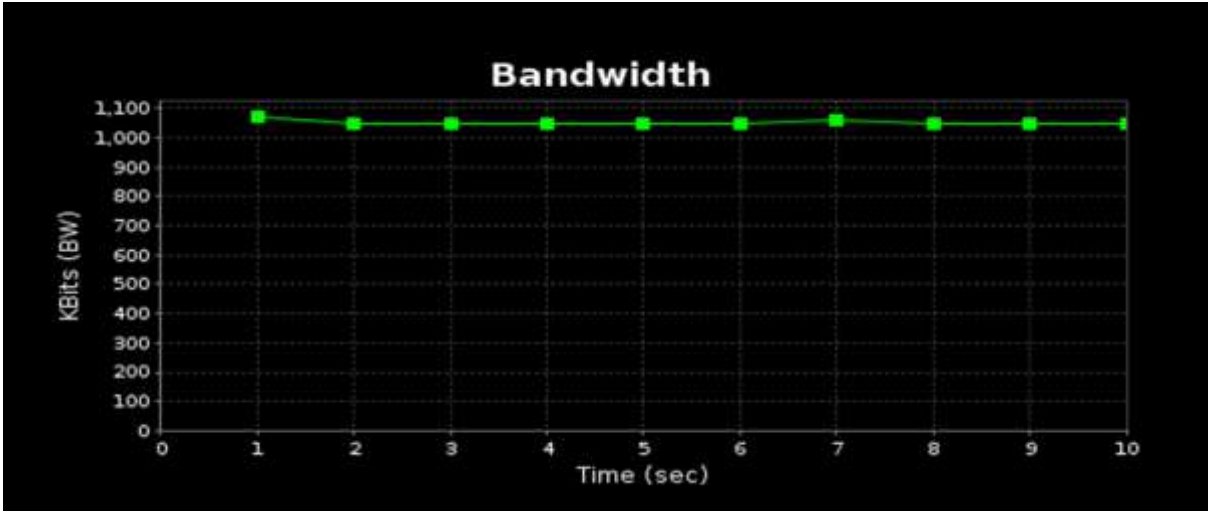


Figure 20: Without DDoS, UDP

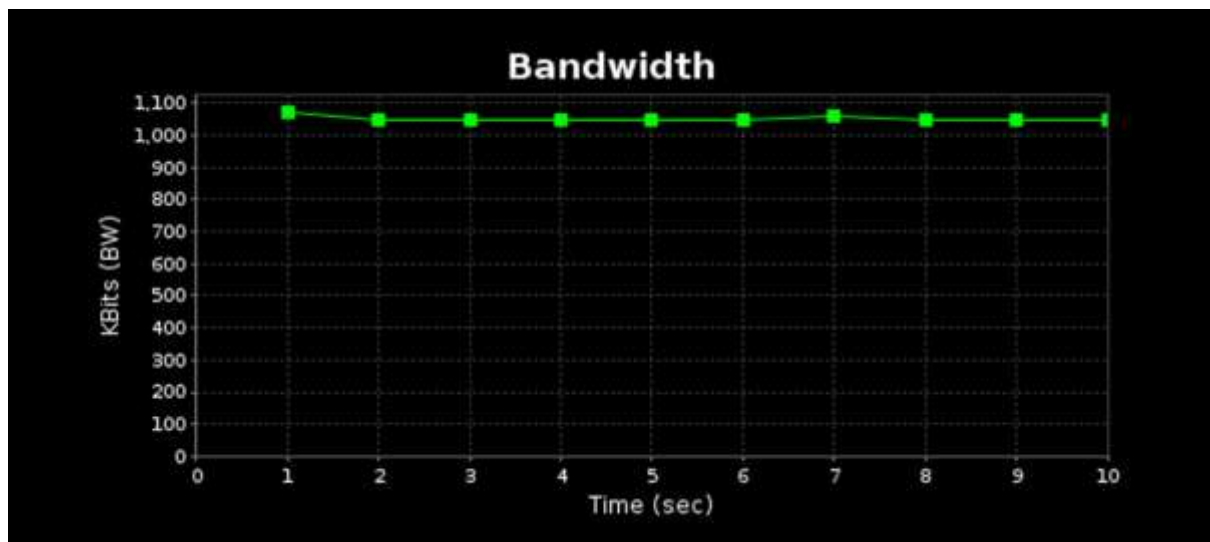


Figure 21: UDP with DDoS