DCU School of Computing
Assignment Submission

Student Name(s):	Liam Waters
Student Number(s):	19214078
Email Address:	liam.waters3@mail.dcu.ie

Program of Study:	M.Sc. in Computing (Blockchain)
Module Code:	CA687 Cloud Systems
Assignment:	Software Defined Network Application
Date of Submission:	20/04/2020

An assignment submitted to Dublin City University, School of Computing for module CA687 Cloud Systems.

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, 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. By signing this form or by submitting this 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 DCU Academic Integrity and Plagiarism Policy available here.

- Name: Liam Waters
- Date: 20/04/2020

Assignment Submission Information

Source code can be found at: https://github.com/LiamWaters/2020_CA2

Demo video can be found at: https://youtu.be/5ho5dbhxlr4

Introduction and motivation for the app

Delivering critical network resources in the appropriate manner is key to the successfulness of an organization. It is challenging is to do this in a cost-effective manner, reserving costly faster, more redundant equipment for revenue generating activity and less resilient equipment for non-mission critical tasks. SDN is an ideal technology to help address this challenge due to its core benefits of:

- Centralized network provisioning speeding up delivery and increasing enterprise agility
- Granular security control
- Enabling management of an entire network as a single unit
- Reduction of operating costs and hardware expenses by virtualising the control planes

This assignment will show how these benefits can be realised. The app designed segments external data traffic placing non-critical network traffic on lower capacity (Wi-Fi) links while revenue generating (web) traffic is placed on high capacity/low latency (Fibre) links.

This design is relevant not only to web traffic but all data traffic including financial transactions using technologies such as Blockchain.

The created SDN network topology and details of the core functions used in the SDN controller design

To simulate a real-world example a representative network was created in Mininet containing five elements – external network resources, NAT, Traffic Prioritisation network, Internal network resources, Web services.

Section	Purpose	Appliance
External Resource Network	Demonstrate external access to internal critical and non-critical resources	External User (extu1) and External Switch (e0)
NAT Bridge	Convert external IP addresses to internal IP using Network Address Translation	NAT Router – r0
Traffic Prioritisation	Segments traffic to pass over appropriate network link	Prioritisation Switches – s0, s1, s2, s3
Internal Resources	Non-critical internal network	Internal User (intu1)
Web Services	Critical internal network	Core switch (core0), Load balancer (lb0), Load balanced switches(lb1, lb2), Load balanced Web Servers (web1, web2, web3, web4) and Non-Load Balanced Web Server (web0)

This infrastructure requires three POX controllers

- Standard routing forwarding controller – CFWD0
- Load balancer controller – CLB0
- Custom Priority Routing controller – CPT0

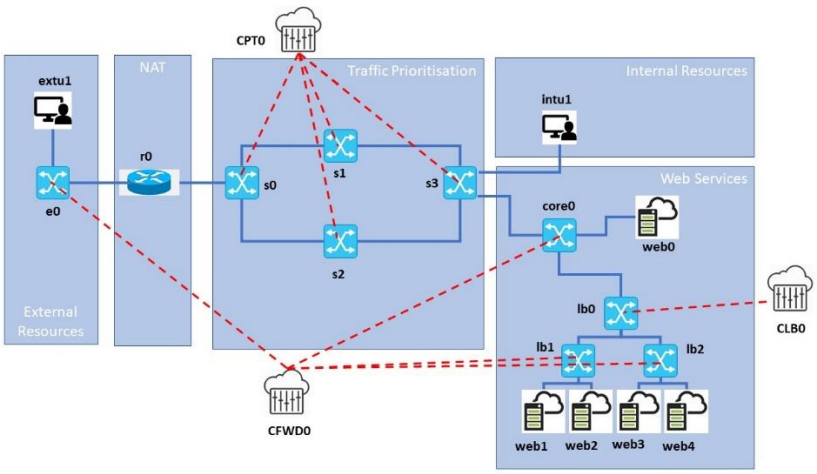


Figure 1 - SDN Topology

The SDN network presented in Fig 1 was designed in Mininet with the controllers created in POX. It represents an external user with the IP address 192.168.1.100 accessing an internal network of the 10.0.x.x range.

This is facilitated by a router r0 performing NAT on the external user changing the IP address from 192.168.1.100 to 10.0.0.1.

Traffic from the router will follow one of two paths when it hits the Prioritisation switches s0 and s3.

If it is bound for the internal nonpriority resources flagged in the figure as internal user 1 (intu1 – 10.0.0.10) it is sent over low bandwidth (10MB), high latency (100ms) links which are subject to 50% packets loss. This simulates a wireless connection.

If it is bound for the priority web resources (web0, web1, web2, web3, web4) it is sent over high bandwidth (1000MB), low latency (1ms) links which are subject to 0% packets loss. This simulates a Fibre connection.

Traffic to the IP address 10.0.1.1 is load balanced using the Controller CLB0 across four web servers – web1, web2, web3 and web4. This simulates high availability services.

The customised Controller (CPT0) was based on a design from Chih-Heng Ke (2013) to inspect the incoming events to the openflow switches and direct them through the appropriate port onto the next switch. A topology of the prioritisation switches and their ports can be found in Fig 2.

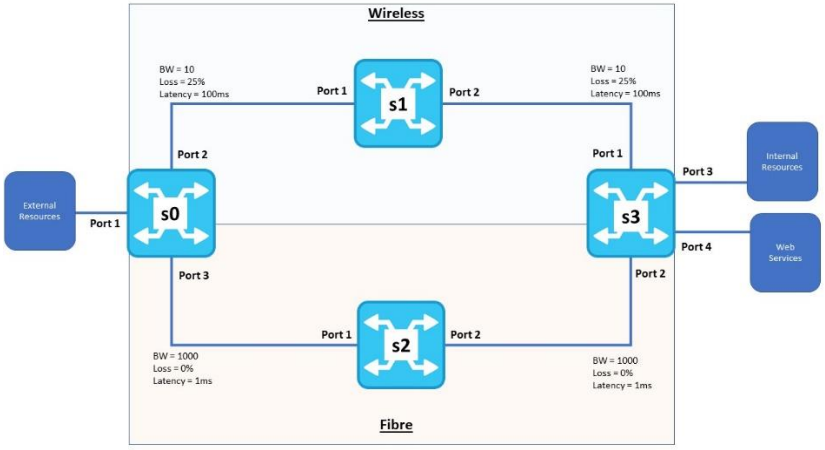


Figure 2 - Controller CPT0

For example, a packet coming from 192.168.1.100 on port 1 - If the network destination (nw_dst) was for the web service 10.0.1.1 the openflow action (ofp_action_output) directed the packet out of port 3 through the high-quality link. Non prioritised traffic for internal network resoures was directed out port 2. Switches s1 and s2 just forwarded traffic onwards to the next switch in the chain. Switch s3 took the incoming traffic and routed it to the internal resources on port 3 or to the Core Web Services switch core0. This redirection was performed in reverse for traffic directed externally.

By specifying the IP addresses which were to be directed on s0 and s3 without wildcards meant that this prioritisation controller added an additional layer of security to the network like a basic firewall.

Evaluation of design and test results

Traffic from external resource extu1 can be seen to be directed over different routes depending on destination IP:

Priority Traffic - Web Services (10.0.1.1 or 10.0.1.11)

Latency & Packet Loss

```
mininet> extu1 ping -c 10 web0
PING 10.0.1.11 (10.0.1.11) 56(84) bytes of data.
64 bytes from 10.0.1.11: icmp_seq=1 ttl=63 time=139 ms
64 bytes from 10.0.1.11: icmp_seq=2 ttl=63 time=28.5 ms
64 bytes from 10.0.1.11: icmp_seq=3 ttl=63 time=27.5 ms
64 bytes from 10.0.1.11: icmp_seq=4 ttl=63 time=28.5 ms
64 bytes from 10.0.1.11: icmp_seq=5 ttl=63 time=27.0 ms
64 bytes from 10.0.1.11: icmp_seq=6 ttl=63 time=26.8 ms
64 bytes from 10.0.1.11: icmp_seq=7 ttl=63 time=26.3 ms
64 bytes from 10.0.1.11: icmp_seq=8 ttl=63 time=27.4 ms
64 bytes from 10.0.1.11: icmp_seq=9 ttl=63 time=26.6 ms
64 bytes from 10.0.1.11: icmp_seq=10 ttl=63 time=26.8 ms

--- 10.0.1.11 ping statistics ---
10 packets transmitted, 10 receive[illegible] 0% packet loss, [illegible]ime 9014ms
rtt min/avg/max/mdev = 26.315/38.427/[illegible] ms
```

Bandwidth

```
"Node: extu1"
root@osboxes:/SDNProject# iperf -c 10.0.1.11
------------------------------------------------------------
Client connecting to 10.0.1.11, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 45] local 192.168.1.100 port 39726 connected with 10.0.1.11 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 45]  0.0-10.1 sec  14.0 MBytes  11.6 Mbits/sec
root@osboxes:/SDNProject#
```

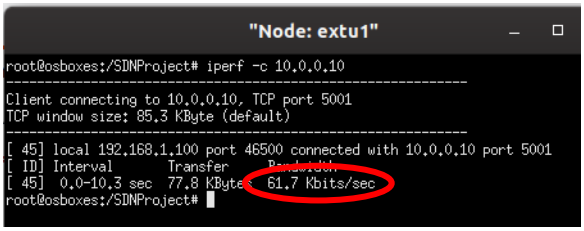
Nonpriority Traffic – Internal Resources (10.0.0.10)

Latency & Packet Loss

```
mininet> extu1 ping -c 10 intu1
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data.
64 bytes from 10.0.0.10: icmp_seq=7 ttl=63 time=453 ms
64 bytes from 10.0.0.10: icmp_seq=9 ttl=63 time=443 ms
64 bytes from 10.0.0.10: icmp_seq=10 ttl=63 time=442 ms

--- 10.0.0.10 ping statistics ---
10 packets transmitted, 3 received, 70% packet loss, time 9158ms
rtt min/avg/max/mdev = 442.156/445.929/453.703/4.803 ms
```

Bandwidth

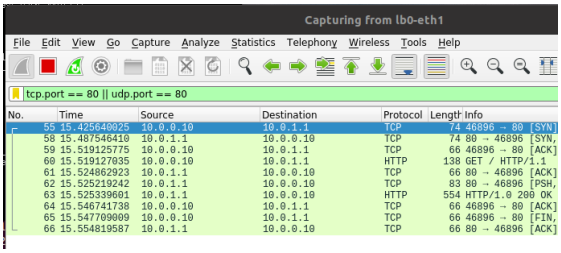


Network Load Balancing for calls to Web Server IP 10.0.1.1

Web server response from external user

```
mininet> extu1 curl 10.0.1.1
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html>
<title>Directory listing for /</title>
<body>
<h2>Directory listing for /</h2>
<hr>
<ul>
<li><a href="cgi-bin/">cgi-bin/</a>
<li><a href="old/">old/</a>
<li><a href="SDN_CA2_CA687.py">SDN_CA2_CA687.py</a>
<li><a href="shortest_path_app/">shortest_path_app/</a>
</ul>
<hr>
</body>
</html>
```

Wireshark capture of request to 10.0.1.1 load balancer



Load balancing across Web Servers

```
DEBUG:iplb.76-16-70-0a-37-4f:Directing traffic to 10.0.1.5
DEBUG:iplb.76-16-70-0a-37-4f:Directing traffic to 10.0.1.4
DEBUG:iplb.76-16-70-0a-37-4f:Directing traffic to 10.0.1.3
DEBUG:iplb.76-16-70-0a-37-4f:Directing traffic to 10.0.1.4
DEBUG:iplb.76-16-70-0a-37-4f:Directing traffic to 10.0.1.2
DEBUG:iplb.76-16-70-0a-37-4f:Directing traffic to 10.0.1.3
```

Challenges and lessons learned

Development Environmental

The default Mininet Virtual Box which came from Mininet.org (2020) had several missing elements and was using an older version of Ubuntu. This was resolved by downloading Ubuntu 19.10 and installing all Mininet dependencies.

MiniNAM worked has a nice graphical display for the routing of traffic on small topologies. Once these became more complex, for example multiple controllers, MiniNAM ran into issues. For this reason, its use was discarded during the assignment and Mininet was used. Building out individual elements of the topology was beneficial and joining them at a later point. This allowed for effective troubleshooting of design issues.

Customer Controller

A challenge which arose during the implementation of a more complex network was an issue with the LinuxRouter Class. Using the TCPlink argument in initiation the Mininet network meant that no external traffic could pass through the router. It could not act as a bridge to the external network or act as a gateway for internal devices. By removing this argument NAT worked but meant applying custom latency, packet loss and bandwidth became an issue. This was overcome by adding cls = TCLink parameter when creating links between topology elements.

Conclusion

The Topology created I believe represents a real-world use case and the Controller CPT0 App addresses an organisational challenge in segmenting high value and low value data traffic to maximise the value of investments to an organisation. The assignment demonstrates the core benefits of SDN in the following ways:

Benefit	Demonstration
Centralized network provisioning speeding up delivery and increasing enterprise agility – using	Provision of dynamic network topology using Mininet to perform use cases in the creation of assignment which would not have been possible in a physical environment
Granular security control	Restriction of data traffic by IP using NAT Router
Enabling management of an entire network as a single unit	Using Mininet and network tools to verify and trouble shoot network end to end
Reduction of operating costs and hardware expenses by virtualising the control planes	Demonstrate the effectiveness of virtual appliances in the effective delivery of a network without needing the capital expense of physical switches, controllers and routers

Bibliography

Ke, C., 2013. Software Defined Network (SDN) Experiment Using Mininet And POX Controller. [online] Csie.nqu.edu.tw. Available at: <http://csie.nqu.edu.tw/smallko/sdn/mySDN.pdf> [Accessed 30 March 2020].

Mininet.org. 2020. Mininet: An Instant Virtual Network On Your Laptop (Or Other PC) - Mininet. [online] Available at: <http://mininet.org/> [Accessed 15 April 2020].

Quinlan, J., 2017. ICIN 2017 | University College Cork. [online] University College Cork. Available at: <https://www.ucc.ie/en/misl/research/software/mininam/> [Accessed 25 March 2020].

Scott, L., 2016. Custom Mininet Topologies And Introducing Atom - Inside Openflow. [online] Inside OpenFlow. Available at: <https://inside-openflow.com/2016/06/29/custom-mininet-topologies-and-introducing-atom/> [Accessed 31 March 2020].