The Knowledge Hub Universities | partnered with Coventry University

School of Computing

Networking Coursework 1 Report

WORD COUNT: 1414 (EXCLUDING REFERENCES AND SOURCE CODE)

Hassan ElGamal
CU1900037

Table of Contents

1. Part One

1.1 Company Network Structure

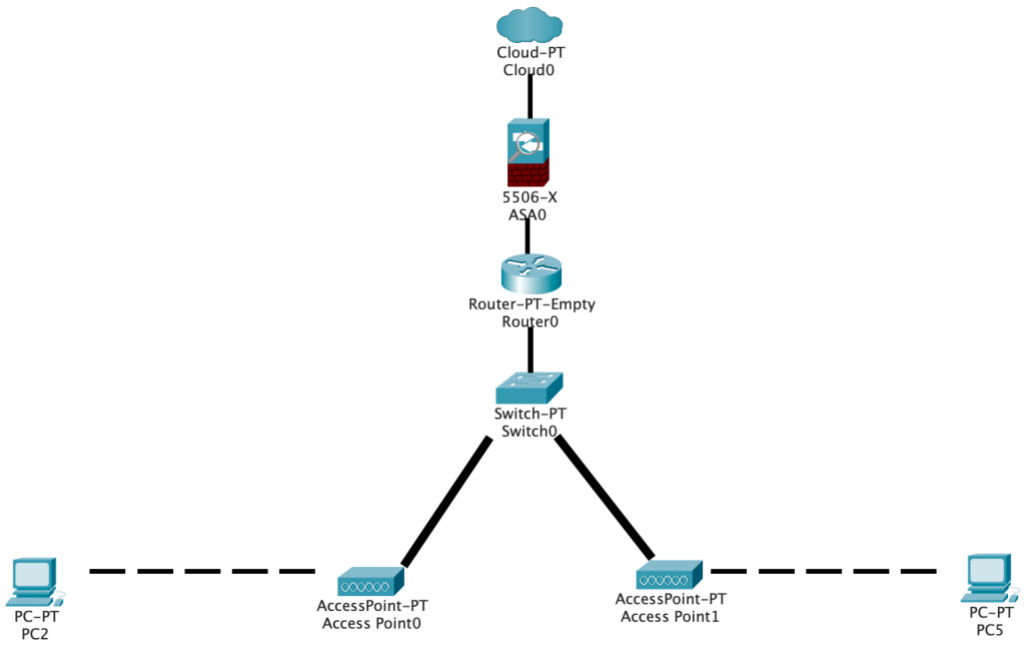

Figure 1 – Primary Company Network Structure

The preliminary structure portrayed in Figure 1 is a very basic working network topology used by the finance advisory setup, myFinTech. The network consists of a web application server, a firewall, a router, a switch, two access points and two personal computers as an example of the connection between the access points to the company devices. The network offers very minimal security support to the entire infrastructure as well as limited contribution to confidentiality, integrity, and availability of the systems. The topology above demonstrates a tree topology, which is a very inefficient method of protecting and offering uninterrupted access to the main data center in any finance advisory headquarters.

Although the tree topology has various advantages such as the ability to easily upgrade it due to its straightforward hierarchy pattern, its security, reliability, cost-efficiency, and its maintenance are its biggest downfalls. Firstly, the security of a tree topology is always at risk of exposing all devices in the topology due to its cabling structure which makes the entire network compromised if an attacker manages to get a hold of one device. This could also apply to the reliability of the network; this is because the main cable of the entire network leads to the connection of the rest of the devices. Hence, if the main cable becomes unstable or damaged, the entire topology will be impacted negatively. In terms of cost-efficiency, the network is very pricey due to the wiring requirements. Some topologies such as the mesh and the star topologies are very costly, however, their wiring requirements include stability and backup wiring routes which makes their cost worth-while. Whereas the tree topology cost in very high but inefficient due to its weak functionality and maintenance. Moreover, the maintenance

of such a network is very difficult due to its large size and direction of configuration; meaning, the maintenance must begin at the very top while it makes its way down. Therefore, a lot of time is wasted on configuring individual point-to-point connections even if they are working perfectly well.

1.2 Structure Upgrades

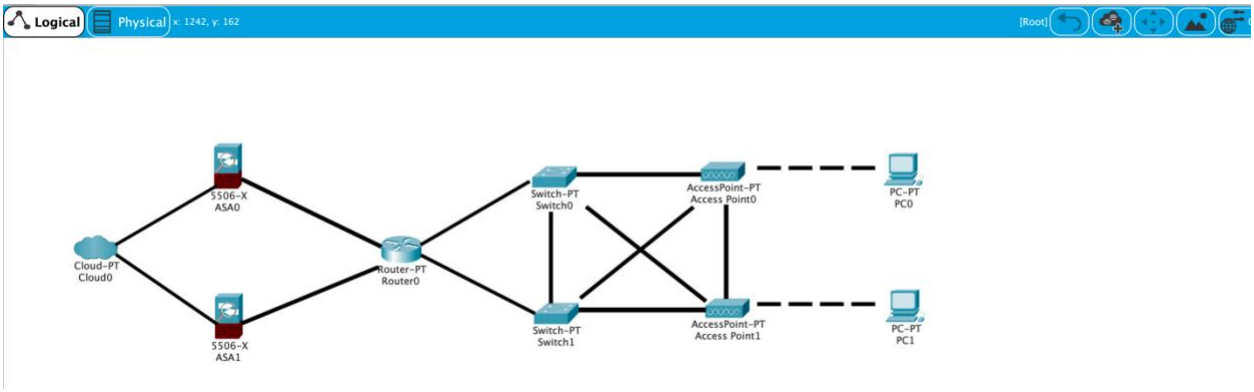

Figure 2 – Upgraded Network Structure to a Mesh Topology

Seeing as the tree topology used in the company's network infrastructure is very inefficient and unstable, a thoroughly thought decision was implemented to upgrade the network to a mesh topology in order to maintain confidentiality, integrity, and availability of the network (Figure 2). In terms of confidentiality, the mesh topology ensures data privacy seeing as there are two firewalls connected which increase security and adds a form of combination between systems that acts as chains to protect the entire system collectively. Additionally, reliability could never be an issue for such a structure that provides backups for core sectors of the network as well as the ability to manage enormous amounts of data due to its high level of security. Lastly, the availability of the system is also one of the vastest advantages of a topology which includes various routes that guarantees the availability of the network if devices or wires are damaged and compromised.

1.3 Budget Estimation and Device Models

Mesh topologies are tremendously pricey, and this scenario includes web application server, two firewall systems, a router, two switches, two access points, two personal computers, and twelve wires which connect all systems. The budget is expected to cost approximately £9940.84 split into:

1. Internet Services Plan: £360 per year (bOnline Business Package)
2. Router: £73 (TP-Link TL-MR6400)
3. Firewalls: £1980 each x2 per year (Logicalis UK Limited Managed Firewall Service 2021)
4. Switches: £1,874.67 each x2 (Cisco Catalyst 9300 24-port data)
5. Wireless Access Points: £521.31 each x2 (Cisco Aironet 1852E Radio Access Point 802.11ac)
6. Wires: £32.99 each x12 (MutecPower 50m CAT6 Outdoor waterproof Direct Burial Ethernet Network Cabl)

Although the topology is costly, the cost-efficiency is well worth the money in comparison to the services provided.

1.4 Stakeholders Involved

The stakeholders include the IT team which carry out constant maintenance, the employees of the company that use the systems daily, the administrators responsible for financing the network infrastructure, and the clients of the company which rely on myFinTech to protect and manage their sensitive data.

2. Part Two

2.1 Multi-User Packet Sniffer Description

The multi-user packet sniffing tools were created using Python language using TCP and UDP protocol. There are four different files:

a. TCPServer.py
b. TCPClient.py
c. UDPServer.py
d. UDPClient.py

This gives the end-user of the tool the ability to pick the preferred type of protocol to use as well as the ability to extract the MAC address, IP address, and port number from the connected clients using the server consoles on both TCP and UDP server architectures. Any types of input from the clients will also be recorded and displayed on the server consoles.

2.2 Pre-requisites

Both TCP and UDP servers use the 'uuid' module in order to define the 'getnode()' function which extracts the MAC addresses and displays them in an unformatted manner to the server console user. All files use the 'socket' module to be able to create a socket-programming tool. All modules are portrayed in the first few lines of the programs.

Table 1.1 – Pre-requisites used in the code

	Function	**Example in the code**
UUID Module	Extracts MAC addresses by generating a UUID for a given host.	print ("Source MAC address: ", hex(uuid.getnode()))
Socket Module	Provides access to a socket interface to simulate how a server and client machines can	client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

	communicate using socket endpoints	
Time Module	Displays the live time per millisecond.	print("Date and time: " + str(datetime.datetime.now()))

Table 1 - Pre-requisites used in the code

2.3 Instructions to use the Software

Firstly, the python files must be located before opening a terminal window. Once the terminal window is opened, the python files must be located using “cd [Path]”. In this scenario, the path is the /Desktop/Coursework/, hence “cd Desktop/Coursework/”. Next, the TCP server will be opened using “sudo python3 TCPServer.py”, the device password will then be required to run the command. This is to get around the permission issues found on Apple’s Macbooks. Once that is done, the terminal must not display any errors and must be completely empty. If that is true, a new terminal window should be opened, and the python files must be located once again using “cd Desktop/Coursework/”. Afterwards, the TCP client command should begin running using the “sudo python3 TCPClient.py” with another password requirement. This will then connect the client to the server using the transmission control protocol. After the client is done writing inputs, the word “end” should be used to disconnect the client from the server which will terminate both processes as mentioned in the console output. The server console will display the client’s IP address, MAC address, port number, and real-time data whenever the user inputs anything as well as disconnecting when the user disconnects. The same execution instructions and methods portrayed in the scenario apply to both the UDPServer.py and the UDPClient.py files.

2.4 GitHub Repo Link

https://github.com/hassannhanyyy/Networking.git

2.5 Execution Snapshots

2.5.1 Server Side (TCP)

```
hassan@hassans-MacBook-Pro-2 Coursework % sudo python3 TCPServer.py
Connection Protocol:  TCP
Source MAC address:   0x2af8bc6cb6a5
Connected by   ('127.0.0.1', 54665)
Date and time:  2021-12-18 13:00:48.111243
```

Figure 3 – TCP Server Console

2.5.2 Client Side (TCP)

```
hassan@hassans-MacBook-Pro-2 Coursework % sudo python3 TCPClient.py
* What's your name? Hassan
* Hello Hassan(Write 'end' to disconnect)
* end
* Disconnected
hassan@hassans-MacBook-Pro-2 Coursework %
```

Figure 4 – TCP Client Console

2.5.3 Server Side (UDP)

```
hassan@hassans-MacBook-Pro-2 Coursework % sudo python3 UDPserver.py
Password:
test
Connection Protocol:  UDP
Source MAC address:   0x2af8bc6cb6a5
IP address:   127.0.0.1
Port number:   8080
Date and time:  2021-12-18 13:10:35.276219
Client Disconnected
hassan@hassans-MacBook-Pro-2 Coursework %
```

Figure 5 – UDP Server Console

2.5.4 Client Side (UDP)

```
[hassan@hassans-MacBook-Pro-2 Coursework % sudo python3 UDPClient.py
* Input (Write 'end' to disconnect)
* test
* Output: TEST
* Input (Write 'end' to disconnect)
* end
* Disconneted
```

Figure 6 – UDP Client Console

2.5.5 Wireshark Comparison

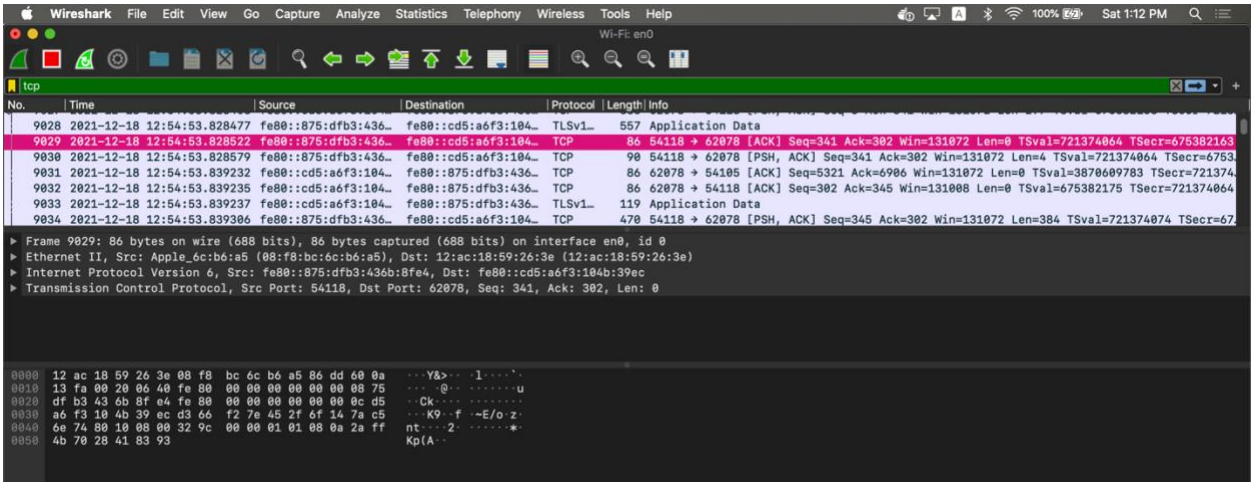

Figure 7 – Wireshark TCP Search Output

Figure 5 demonstrates the Wireshark TCP search output which in comparison to the client-server architecture built, is not that different. In terms of the GUI, Wireshark includes a more relaxing output that makes it easier for the end-user to read. More options are also offered by Wireshark such as the Frame numbers, the captured bytes on the interfaces, the internet protocol version, written data regarding the protocol, and many more. However, in terms of functionality, both the TCP server created in python and Wireshark offer the same key options such as the real-live time, the MAC addresses, the IP addresses, the port numbers, and the protocol type. The python program offers the IP addresses and port numbers on-display instantly, whereas the user must look for them on Wireshark under the "Ethernet II" and the "TCP" drop-down menu options in Figure 7

3. List of Figures

4. List of Tables

5. References

Work Cited (APA)

Business Broadband. (n.d.). Retrieved from USwitch: https://www.uswitch.com/broadband/business-broadband/

Cisco Catalyst 9300 24-port data only, Network Essentials. (n.d.). Retrieved from Business Direct: https://www.businessdirect.bt.com/products/cisco-catalyst-9300-24-port-data-only--network-essentials-c9300-24t-e-G2QJ.html?fb=363

Extracting MAC Addresses using Python. (n.d.). Retrieved from Geeks for Geeks: https://www.geeksforgeeks.org/extracting-mac-address-using-python/

Logicalis UK Limited Managed Firewall Service. (n.d.). Retrieved from Digital Market Place: https://www.digitalmarketplace.service.gov.uk/g-cloud/services/739718354344693

Mesh Network. (n.d.). Retrieved from GetInternet: https://getinternet.com/mesh-network/

Mesh Topology Advantages and Disadvantages . (n.d.). Retrieved from AplusTopper: https://www.aplustopper.com/mesh-topology-advantages-and-disadvantages/

MutecPower 50m CAT6 Outdoor waterproof Direct Burial Ethernet Network. (n.d.). Retrieved from Amazon UK: https://www.amazon.co.uk/MutecPower-Outdoor-waterproof-Ethernet-Network/dp/B00EOTHDWQ/ref=sr_1_3?keywords=50+metre+ethernet+cable&qid=1639998925&sr=8-3

Tree Topology Advantages and Disadvantages. (n.d.). Retrieved from AplusStopper: https://www.aplustopper.com/tree-topology-advantages-and-disadvantages/

TP-Link TL-MR6400. (n.d.). Retrieved from Amazon UK: https://www.amazon.co.uk/TL-MR6400-Unlocked-Configuration-Required-External/dp/B016ZWXYXG/ref=sr_1_8?_encoding=UTF8&c=ts&keywords=Routers&qid=1639998454&s=computers&sr=1-8&ts_id=430579031

6. Source Code

6.1 TCPServer.py

```
# TCP Server

import socket
import uuid
import time
import datetime

host = socket.gethostname()

port_number = 3030

with socket.socket(socket.AF_INET,socket.SOCK_STREAM) as s:
    s.bind((host,port_number))
    s.listen()

    connect,address = s.accept()

    with connect:

        print("Connection Protocol:  TCP")
        print("Source MAC address:  ", hex(uuid.getnode()))
        print("Connected by  ", address)
        print("Date and time:  " + str(datetime.datetime.now()))

        while True:

            data = connect.recv(1024)

            if not data:

                break
```

```
        connect.sendall(data)
```

6.2 TCPClient.py

```
# TCP Client

import socket

message = ""

host = socket.gethostname()

port_number = 3030

with socket.socket(socket.AF_INET,socket.SOCK_STREAM) as s:

    s.connect((host,port_number))

    name = input("* What's your name? ")

    welcome = input("* Hello " + name.capitalize() + "(Write 'end' to disconnect)\n* ")

    while (message.lower().strip() == "end"):

        message = input("* ")

print("* Disconnected")
```

6.3 UDPServer.py

```
# UDP Server

import socket
import uuid
import time
```

```
import datetime

if __name__ == "__main__":
    host = "127.0.0.1"
    port = 8080

    server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    server.bind((host, port))

    while True:
        data, address = server.recvfrom(1024)
        data = data.decode("utf-8")

        if data == "end":
            print("Client Disconnected")
            break

        print(data)
        print("Connection Protocol:  UDP")
        print("Source MAC address:  ", hex(uuid.getnode()))
        print("IP address:  ", host)
        print("Port number:  ", port)
        print("Date and time:  " + str(datetime.datetime.now()))



        data = data.upper()
        data = data.encode("utf-8")
        server.sendto(data, address)


host = socket.gethostname()
```

6.4 UDPClient.py

```
# UDP Client

import socket

if __name__ == "__main__":
    host = "127.0.0.1"
    port = 8080
    address = (host, port)

    client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    while True:
        data = input("* Input (Write 'end' to disconnect) \n* ")


        if data == "end":
            data = data.encode("utf-8")
            client.sendto(data, address)

            print("* Disconneted")

            break

        data = data.encode("utf-8")
        client.sendto(data, address)

        data, address = client.recvfrom(1024)
        data = data.decode("utf-8")
        print(f"* Output: {data}")
```