Computer Networks

Network Setup Project Report

João Baltazar[1] and Nuno Costa[1]

[1]Faculty of Engineering of the University of Porto

December 5, 2022

Abstract

The goal of the project is to create a small, local computer network through which to download a file from an FTP server.

This goal was achieved - both the network and the download application have met the specifications and the hands-on approach and further reflection on what was happening allowed for deeper learning opportunities.

1 Introduction

The second lab assignment is composed of 2 parts that constitute the final presentation: An FTP download application and a network configuration through which to download. They will be explored with more detail below.

2 Download Application

Part 1 consists in the development of a simple FTP application that retrieves a specified file using the FTP protocol as described in RFC959. It should take an argument that adopts the URL syntax, as described in RFC1738. Example:

```
download ftp://ftp.up.pt/pub/hello.txt
```

where the URL path is

```
ftp://[<user>:<password>@]<host>/<url-path>
```

It seeks to teach through practice the following points:

- Describe client-server concept and its peculiarities in TCP/IP
- Characterize application protocols in general, characterize URL, describe in detail the behaviour of FTP
- Locate and read RFCs
- Implement a simple FTP client in C language
- Use sockets and TCP in C language
- Understand the service provided DNS and use it within a client program

2.1 Architecture of the Download Application

The application can be defined by a series of consecutive steps:

1. Input validation and extraction of valid information from the program's argument
2. Establishment of the connection on the FTP control port
3. Login
4. Activation of passive mode and listening on the provided port
5. Request of the file and download
6. Closing the connection

All communication is performed using TCP sockets, using the service provided DNS through *gethostbyname()* to get the IP, and ports 21 (FTP standard control port) and whatever is supplied by the server for the download later. Login is done with the provided fields (if none, an anonymous connection is established). The FTP server replies are fed through a state machine which exhausts the input line by line as per the standard and interprets the 3-digit codes at the start. *227 Entering Passive Mode (IP1, IP2, IP3, IP4, PORT1, PORT2)* is handled specially, as the 6 sent bytes need to be parsed and used by the application to open the second socket and receive the file. The data is written to a local file named after the downloaded file. All connections are adequately closed.

2.2 Report of a Successful Download

The log of a successful download is attached as annex.

3 Network Configuration and Analysis

The main purpose of this sequence of experiments is to successfully build a small local network serving two computers, with two VLAN's and two routers - one of which is a regular computer serving as a routing mechanism between VLAN's -, which then connect to the internet through the Lab's router.

All experiments were executed in Bench 3.

3.1 Experiment I - IP Config

3.1.1 Network Architecture

At the end of this experiment, the network configuration that we seek to match is the following:

3.1.2 Experiment Objectives

The main objectives are to successfully set up the IP Addresses of tux33 and tux34, as well as connecting both of them to the Cisco Switch in order to allow communication between the two computers.

3.1.3 Main Configuration Commands

For this experiment, the only commands that had to be executed were the following:

```
Tux3:

root# ifconfig eth0 up
root# ifconfig eth0 172.16.30.1/24

Tux4:
```

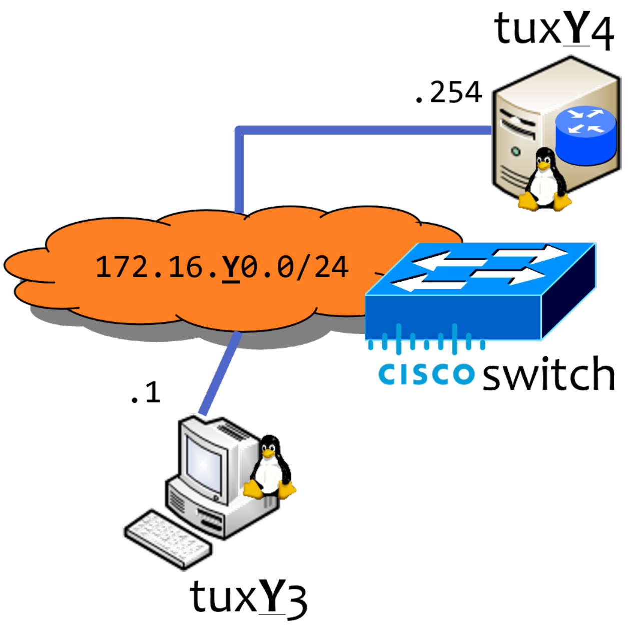

Figure 1: Network Configuration for Experiment I

```
root# ifconfig eth0 up
root# ifconfig eth0 172.16.30.254/24
```

Apart from this, we only had to connect both tuxes to the switch. In our setup, tux3's eth0 was connected to the Switch's port 1, and tux4's eth0 was connected to the Switch's port 4.

3.1.4 Log Analysis

The corresponding Wireshark log for the ping from tux3 to tux4 is present in the Appendix II, figure I. The IP Address of tux3 is 172.16.30.1, with a MAC address of 00:21:5a:5a:7d:b7, while the IP Address of tux4 is 172.16.30.254, with a MAC address of 00:21:5a:5a:74:3e.

The ARP (Address Resolution Protocol) packets that are present in this log seek to resolve the unknown location - in this case, the MAC Address - of the ping destination. Since we've attempted to ping tux4 using its IP Address, the ping needs to translate that IP Address to the corresponding MAC Address.

As such, it sends out an ARP Broadcast Packet (ff:ff:ff:ff:ff:ff) from tux3 (00:21:5a:5a:7d:b7), *"asking who's"* 172.16.30.254 (IP Address of tux4), and tux4, upon receiving the broadcast request, sends to tux3 a response (from 00:21:5a:5a:74:3e). After that, tux3 saves this information in its ARP Table, where all associations IP Address - MAC

Address are stored for future use. With this newly stored information, tux3 no longer needs to make a broadcast request for tux4's MAC Address when it needs it, since it's already stored in its ARP Table.

After resolving tux4's MAC Address, tux3 can now ping tux4 using a ICMP (Internet Control Message Protocol) packet, and receive the corresponding answer. Each packet uses the previously mentioned IP Addresses and MAC Addresses.

Alongside the experiment itself, other relevant information could be found in the Wireshark log. Every packet has its type defined in the sent bytes 12 and 13. For IPv4 Packets, the type value corresponds to 0x0800, while the ARP Packets have a value of 0x0806. The length of the IPv4 Packets can also be calculated through bytes 2 and 3 of the IPv4 header, the Total Length field. On the other hand, the length of the ARP Packets can be obtained according to the type of ARP Packet: the request packet's size is equal to the sum of the Ethernet II and ARP layers' size, while the response packet's size is equal to the request size plus an additional padding.

Another relevant frame was detected in the Wireshark log: the LOOP packet. That packet has two main purposes. Firstly, it is meant as a **keep-alive** message: it seeks to verify that the network is operational by sending a frame from and to the MAC of the Cisco Switch. Secondly, this same message checks and prevents self-looped port, which are ports that receive the same frames that themselves have sent. This is done by checking if the port receives the same LOOP frame itself sent, and if is the case, putting the port in an error state.

3.2 Experiment II

3.2.1 Network Architecture

At the end of this experiment, the network configuration that we seek to match is the following:

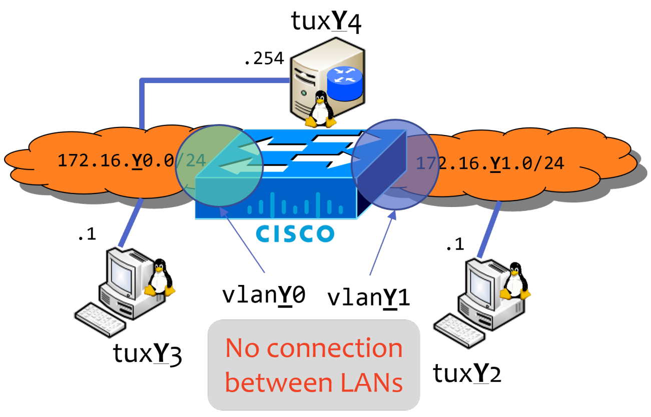

Figure 2: Network Configuration for Experiment II

3.2.2 Experiment Objectives

The main objective is to successfully set up the two VLANs represented in the schema above in the Cisco Switch.

3.2.3 Main Configuration Commands

For this experiment, the commands that had to be executed were the following:

```
Tux2:

root# ifconfig eth0 up
root# ifconfig eth0 172.16.31.1/24

Switch:

Switch enable
Switch# configure terminal
Switch(config)# vlan 30
Switch(config)# end

Switch# configure terminal
Switch(config)# interface fastethernet 0/1
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 30
Switch(config-if)# end

Switch# configure terminal
Switch(config)# interface fastethernet 0/4
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 30
Switch(config-if)# end

Switch# configure terminal
Switch(config)# vlan 31
Switch(config)# end

Switch# configure terminal
Switch(config)# interface fastethernet 0/2
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan
                                        31
Switch(config-if)# end
```

We also connected tux2's eth0 to the Switch's port 2.

3.2.4 Log Analysis

After the VLAN configuration on the switch and properly connecting the ports to each VLAN, we can ponder why the logs turned out as they did.

First of all, it is clear that tux3 can ping tux4, as it was already possible in the previous experiment. It must noted, however, that neither tux3 nor tux4 can ping tux2 - and vice-versa. That does happen since no connection between vlans and/or machines was made, so no possible routing exists for that connection. This thesis is reinforced by the logs.

Alongside all of this, this experiment made clear the existence of two distinct broadcast domains. When sending a broadcast frame from tux 3, it reached tux4, and vice-versa. However, when sending a broadcast frame from tux2, no other computer received that same ping, and neither did tux2 receive any ping from other computers. As such, it is clear the existence of these two broadcast domains: one which includes tux3 and tux4, and another with tux2 only. This is also reinforced by the log: 172.16.30.X is one of the broadcast domains, while the other is 172.16.31.X.

3.3 Experiment III

3.3.1 Network Architecture

As this was a remote experiment, there's no network architecture to strive for.

3.3.2 Experiment Objectives

The main objectives are analysing the configuration file of a Cisco router, testing DNS entries and configuring routes on the local machine.

3.3.3 Log Analysis

Concerning the Cisco router configuration, the router name can be found on line 7 - *hostname gnu-rtr1*; there are two available fast-ethernet Ethernet ports - *interface FastEthernet0/0* and *interface FastEthernet0/1*; port 0 is used for the "inside" IP address of the NAT configuration, as 172.16.30.1/24 and port 1 is used for the "outside" IP address as 172.16.254.45/24; the default gateway for outbound traffic goes through 172.16.254.1 (*ip route 0.0.0.0 0.0.0.0 172.16.254.1*), and any traffic with network 172.16.40.0/24 must go through port 2 on network 172.16.30.0/24 (*ip route 172.16.40.0 255.255.255.0 172.16.30.2*). The outside interface is connected to the Internet (*ip nat pool ovrld 172.16.254.45 172.16.254.45*) with $2^{32-24} = 256$ minus the reserved .0 and .255 ports (*prefix-length 24*). The router is using overloading.

Concerning the DNS configs, adding the entry *142.250.200.142 youtubas* to */etc/hosts* will basically work as a DNS entry, translating the hostname into an IP address. This way, we have local storage of the association and thus do not ask a DNS service. Later, editing */etc/resolv.conf*, we can confirm that it is also possible to change (or delete) the nameserver. In any case, the alias for youtubas would work because of the local rule.

Concerning Linux routing, deleting the default gateway obviously locks us off from accessing the outside. Adding a route for a specific IP address to the now-deleted gateway lets us reach that IP address. However, that does not work if we add a hostname, as we have no way of translating it without a DNS service. Adding just the DNS service (and not the IP the DNS will give us) still isn't enough to let us reach that host, as we'll receive the IP address from the DNS service and not have a route to it. Restoring the default gateway fixes everything.

3.4 Experiment IV

3.4.1 Network Architecture

At the end of this experiment, the network configuration that we seek to match is the following:

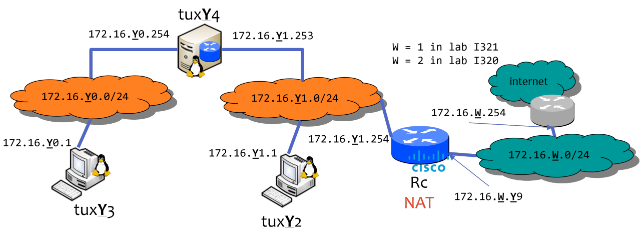

Figure 3: Network Configuration for Experiment IV

3.4.2 Experiment Objectives

The main objectives are to successfully apply the router configuration tested in the experiment III, having experiment II as a basis, and successfully configure the Cisco Router and the Linux Router.

3.4.3 Main Configuration Commands

For this experiment, the commands that had to be executed were the following:

```
Tux2:
root# route add -net
      172.16.30.0/24 gw 172.16.31.253
root# route add default
      gw 172.16.31.254

Tux3:
root# route add default
      gw 172.16.30.254


Tux4:

root# ifconfig eth1 up
root# ifconfig eth1 172.16.30.253/24

root# route add default
      gw 172.16.31.254

root# echo 1 > /proc/sys/net/ipv4/
          ip_forward
root# echo 0 > /proc/sys/net/ipv4/
          icmp_echo_ignore_broadcasts


Switch:

Switch enable
Switch# configure terminal
Switch(config)# interface fastethernet 0/3
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan
                                      31
Switch(config-if)# end

Switch# configure terminal
Switch(config)# interface fastethernet 0/5
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access
                   vlan 31
Switch(config-if)# end

Router:

router enable
router# configure terminal
router# {--In Annex--}
router# end
```

Alongside these commands, we also connected FE0/0 to port 5 of the Switch, and FE0/1 to the Port of the Lab Router.

3.4.4 Log Analysis

After this experiment, tux2 and tux3 were successfully connected to the Internet.

For that to happen, we had to set up a routing system.

To connect tux2 and tux3, a route was added to tux3 to forward all traffic to the port 254 of VLAN 30 (defining the default gateway as tux4), as well as a route on tux2 to forward all traffic to VLAN 30 through the port 253 of VLAN 31 (also tux4). By adding these routes to the forwarding table, both tuxes can now ping each other, as they now know how to route their traffic - by having tux4 as their router , which forwards the traffic both ways, accordingly. This can be confirmed by the Wireshark captures. When pinging tux2 from tux3, and capturing on tux4, the requests and replies are captured, with tux2 and tux3 as sender and destination (interchangeably), clarifying that tux4 serves as router for this connection.

As for the connection to the Cisco Router, the process was a little bit different. The NAT (Network Address Translation) had to be properly set up to ensure that the local IP Addresses were properly converted into the public one, ensuring proper routing to the outside of the local network. Both the inside and outside interfaces were defined (inside - 172.16.31.254; outside - 172.16.2.39) to achieve this goal, as well as the proper NAT configuration (in this case, nat pool ovrld to the outside route). Alongside that, the router's default gateway was set as 172.16.2.254, which is the Lab's Router, and the route to VLAN 30 was set to the port 253 of VLAN 31 (tux4). This ensures that the traffic to VLAN 30 is forwarded to tux4, which then handles the routing. As a final note, both VLAN 30 and 31 were added to the router's access-list (with only a 7 bit mask, which is why tux4 cannot access the internet, since both tux4 routes are 172.16.30.254 and 172.16.31.253, therefore not whitelisted).

After setting the default gateway of both tux2 and tux4 to the inside IP address of the router, the network configuration is complete, and the access to the internet is available on both tuxes.

4 Conclusions

All of the project's goals were achieved - all experiments were executed with success, and the required download application is complete, well designed and working as intended; the analysis of the logs matches and reinforces what was discussed in class on a theoretical level; and the development of the project as a whole was a solid practical learning experience.