Faculdade de Engenharia da Universidade do Porto

Computer Networks

2st Lab Work of RCOM

Supervisor:

Eduardo Almeida enalmeida@fe.up.pt

Authors:

Bruno Huang up202207517@fe.up.pt

Ricardo Yang up202208465@fe.up.pt

Summary

This project was developed as part of the Redes e Computadores (RCOM) course to create an FTP download application and explore the configuration and operation of computer networks.

Through this project, we gained a better understanding of network communication and the *RFC 959* protocol.

Introduction

This project aims to develop a FTP download application and configure a network using the lab computers, also known as TUXs, following the protocol and the provided configuration guide. This report is structured as follows:

The report concludes with the Conclusion, References, Annexes (including images and the source code of the FTP Download Application).

Part 1 - Download application

Architecture

The download application is an FTP client that retrieves files from FTP servers by following the RFC 959 protocol. It is organized into two modules:

- **app_download**: Handles the main functionalities, including URL parsing, establishing connections, sending FTP commands, and processing server responses.
- **main**: Controls the overall application workflow by coordinating the functions in app_download to ensure smooth operation.

Execution Flow

1. **URL Parsing**

   The `parseURL` function extracts the user, password, host, and file path from the given URL. It also resolves the hostname to an IP address using the `gethostbyname` function. If the user and password are not provided in the URL, it uses the default credentials (anonymous:anonymous).

2. **Establishing Connections**
   The `openConnection` function establishes a connection to the FTP server or the passive mode data port.
   The `connectFTP` function logs in to the server using the credentials.
3. **Passive Mode Activation**
   The `enterPassiveMode` function configures the application to use passive mode, creating a secondary connection specifically for file transfer.
4. **Requesting Files**
   The `requestResource` function sends the appropriate FTP command to request the file from the server.
5. **Receiving and Saving Files**
   The `receiveData` function handles downloading the requested file through the data connection. It writes the received file data to a local directory.
6. **Closing Connections**
   The `closeConnection` function ensures that both the control and data connections are cleanly terminated after the file transfer is complete.

This structured approach ensures a clear division of tasks between modules, making the application easy to maintain. Each function handles a specific aspect of the FTP download process, from initiating the connection to saving the file locally.

Download Report

To execute the application, use the Makefile command: `$ make run` (image 1).

The URL can be updated by modifying the Makefile with the following structure: `URL=ftp://[<user>:<password>@]<host>/<url-path>`

Wireshark logs capture details of the initial FTP packets and FTP-DATA frames during the download process (image 2). The application was tested with various files, all of which were successfully downloaded (image 3). Downloaded files are stored in the **/downloads** directory.

Part 2 - Network configuration and analysis

Exp 1 - Configure an IP Network

1.1. Network Architecture

In this experiment, tux3 and tux4 are connected via the Mikrotik switch (image 4).

1.2. Experiment Objectives

The objective is to learn how to configure an IP address on a computer, link two computers through a switch, and understand the concepts of forwarding tables and the ARP protocol.

1.3. Main Configuration Commands

The following commands were used for configuration on bench 4:

```
$ ifconfig eth1 up                        // tux3
$ ifconfig eth1 172.16.40.1/24            // tux3
$ ifconfig eth1 up                        // tux4
$ ifconfig eth1 172.16.41.254/24          // tux4
```

Both computers were connected to the Mikrotik switch. The port used for each computer was not relevant.

1.4. Relevant Logs

ARP (Address Resolution Protocol) is essential for translating IP addresses into MAC addresses within a local network. When tux3 sends a ping to tux4, it initially broadcasts an ARP request to discover the MAC address of tux4. This request includes tux3's MAC address (00:01:02:9f:81:2e) as the source and uses ff:ff:ff:ff:ff:ff as the destination (image 6). The tux4 responds with its MAC address (00:c0:df:02:55:95) which tux3 stores in its ARP table (image 7). This eliminates the need for further ARP exchanges in subsequent communications. ARP packets include both the source's MAC and IP addresses and request the target's MAC address to facilitate communication over Ethernet.

When the ping command is executed, ICMP Echo Request packets are sent to tux4, which replies with ICMP Echo reply packets (image 5). These packets are encapsulated within IP frames for transfer. The source of the ping packets is tux3, which uses its IP and MAC address, and the destination is tux4, which similarly uses its own IP and MAC. If tux3 does not initially know tux4's MAC address, the ARP process ensures the necessary resolution.

Ethernet frames have a field that indicates their type, allowing the identification of ARP or IP frames. A "Type" value of 0x0806 specifies an ARP frame (image 8), while 0x0800 indicates an IP frame (image 9). For IP frames, the "Protocol" field helps identify the ICMP frames (image 10). The "Frame Length" field allows us to determine the length of a receiving frame (image 11).

The loopback interface, which uses the IP address 127.0.0.1, is a virtual network device used for testing on the same machine. It allows software and network configurations to be debugged without the need to connect to an external network, making it an important tool for local development and troubleshooting.

Exp 2 - Implement two bridges in a switch

2.1. Network Architecture

In this experiment, tux3 and tux4 are connected through bridge40, and tux2 is connected through bridge41, all via the Mikrotik switch (image 12). Tux2, tux3, and tux4 are connected to switch ports ether1, ether2, and ether3, respectively.

2.2. Experiment Objectives

The objective is to learn how to configure bridges using the Mikrotik switch GTK terminal and connect computers through the created bridges.

2.3. Main Configuration Commands

The following commands were used for configuration on bench 4:

```
$ ifconfig eth1 up                                    // tux2
$ ifconfig eth1 172.16.41.1/24                        // tux2
```

In the Mikrotik switch GTKterm:

```
> /interface bridge port remove [find interface = ether1]   // tux2
> /interface bridge port remove [find interface = ether2]   // tux3
> /interface bridge port remove [find interface = ether3]   // tux4
> /interface bridge add name=bridge40
> /interface bridge add name=bridge41
> /interface bridge port add bridge=bridge41 interface=ether1
> /interface bridge port add bridge=bridge40 interface=ether2
> /interface bridge port add bridge=bridge40 interface=ether3
```

2.4. Relevant Logs

After configuring the network, a broadcast ping was sent from tux3. The logs confirm that tux3 and tux4 are connected, as both are part of the same bridge (bridge40). However, no ICMP responses were received due to the *net.ipv4.icmp_echo_ignore_broadcasts* parameter being enabled by default. Despite this, the logs indicate that tux4 received the packets (image 13).

On the other hand, tux2 did not receive any ICMP packets from tux3, as it belongs to a separate bridge (bridge41). This was verified by the absence of logs on tux2 (image 14).

From this experiment, it can be concluded that there are two broadcast domains: one for each bridge. This is evident from the logs, as tux3's packets were only received by tux4, which shares the same bridge, while tux2, on a different bridge, did not receive the packets.

Exp 3 - Configure a Router in Linux

3.1. Network Architecture

In this experiment, tux3 and tux4 are connected through bridge40, while tux2 and tux4 are connected through bridge41 (image 15). Tux4 is configured as a router, connecting both bridges. The eth2 of tux4 is connected via switch port ether6.

3.2. Experiment Objectives

The objective is to learn how to configure tux4 as a router, set up network routes, analyze IP and MAC addresses in ARP and ICMP packets, and observe entries in the forwarding table.

3.3. Main Configuration Commands

The following commands were used for configuration on bench 4:

```
$ ifconfig eth2 up                                  // tux4
$ ifconfig eth2 172.16.41.253/24                    // tux4
```

In Mikrotik GTKterm:

```
> /interface bridge port remove [find interface=ether6]
> /interface bridge port add bridge=41 interface=ether6
```

In terminal:

```
$ sysctl net.ipv4.ip_forward=1                      // tux4
$ sysctl net.ipv4.icmp_echo_ignore_broadcasts=0      // tux4
$ route add -net 172.16.41.0/24 gw 172.16.40.254    // tux3
$ route add -net 172.16.40.0/24 gw 172.16.41.253    // tux2
```

3.4. Relevant Logs

After configuring the network, we tested the connection by pinging from tux3 to the IP addresses 172.16.40.254 (tux4's eth1), 172.16.41.253 (tux4's eth2), and 172.16.41.1 (tux2). All pings were successful, confirming that the routes were set up correctly. On tux3, a route was added to send data to the 172.16.41.0/24 network through the gateway 172.16.40.254 (tux4's eth1). Similarly, tux2 was configured with a route to reach the 172.16.40.0/24 network via the gateway 172.16.41.253 (tux4's eth2). These routes allow tux3 and tux2 to communicate through tux4, which acts as a router.

When pinging tux2 from tux3, the logs on tux4 showed the sequence of ARP and ICMP packet flow through its eth1 and eth2 interfaces. Initially, tux3 needed to resolve the MAC address of tux4's eth1 (172.16.40.254) to send the ICMP packet (image 19). After receiving the MAC address, tux3 forwarded the ICMP Echo Request to tux4's eth1. Once tux4 received the ICMP packet, it had to resolve the MAC address of tux2 (172.16.41.1) on its eth2 interface (image 20). After obtaining the MAC address of tux2, tux4 forwarded the ICMP packet to tux2.

The forwarding table on tux4 specified which interface (eth1 or eth2) should be used for each network. The IP addresses stayed the same throughout the communication, while the MAC addresses changed depending on the network segment the packet was traversing. Tux4 played a key role in forwarding the packet from tux3 to tux2 by resolving MAC addresses and directing the traffic based on the configured routes.

Exp 4 - Configure a Commercial Router and Implement NAT

4.1. Network Architecture

In this experiment, tux3 and tux4 are connected through bridge40, while tux2, tux4 and Mikrotik router (Rc) are connected through bridge41 via Mikrotik switch. The Mikrotik router is connected to the Lab Network with NAT enabled (image 21). The ether2 of Rc is connected via switch port ether12.

4.2. Experiment Objectives

The objective is to learn how to configure a Mikrotik router, set up IP addresses and static routes, and understand the functionality and importance of NAT

4.3. Main Configuration Commands

The following commands were used for configuration on bench 4:

In switch GTKterm:

```
> /interface bridge remove port [find interface=ether12]
> /interface bridge add port bridge=bridge41 interface=ether12
```

In Rc GTKterm:

```
> /ip address add address=172.16.1.41/24 interface=ether1
> /ip address add address=172.16.41.254/24 interface=ether2
> /ip route add dst-address=172.16.40.0/24 gateway=172.16.41.253
```

In terminal:

```
$ route add -net 172.16.41.0/24 gw 172.16.40.254          // tux3
$ route add -net 172.16.1.0/24 gw 172.16.40.254           // tux3
$ route add -net 172.16.1.0/24 gw 172.16.41.254           // tux4
$ route add -net 172.16.40.0/24 gw 172.16.41.253          // tux2
$ route add -net 172.16.1.0/24 gw 172.16.41.254           // tux2
```

4.4. Relevant Logs

After configuring the network, we tested connectivity by pinging from tux3 the tux2, tux4, the Mikrotik router (Rc), and the FTP server (172.16.1.10) with NAT enabled. All tests were successful.

Next, the routing configuration was updated to use Rc as the gateway to the bridge40 subnet instead of tux4, and ICMP redirect was disabled. During the test, ICMP packets from tux2 to tux3 followed the path tux2 → Rc → tux4 → tux3, with the return path being tux3 → tux4 → tux2 (image 26). Since ICMP redirect was disabled, the router did not signal the faster route, which would have been tux2 → tux4 → tux3.

When ICMP redirect was enabled, the system detected the shorter path (tux2 → tux4 → tux3) and dynamically updated the routing to follow this more efficient route. This demonstrated ICMP redirect's ability to optimize traffic by informing devices of more direct paths (image 27).

NAT was tested by pinging the lab network. With NAT enabled, the pings worked because NAT converted private IPs into the router's public IP, allowing access to external networks (image 28). Without NAT, the pings failed since private IPs cannot communicate directly on public networks (image 29).

Static routes on the Mikrotik router were set using the previous commands, directing traffic between subnets. NAT was enabled using `/ip firewall nat enable 0`, which hid private IPs behind the router's public IP for external connectivity.

Exp 5 - DNS

5.1. Network Architecture

In this experiment, the network architecture is the same as the Exp 4, with the addition of configuring the Lab Network DNS on tux2, tux3, and tux4.

5.2. Experiment Objectives

The objective is to learn how to configure a DNS service and analyze the information exchanged in DNS packets.

5.3. Main Configuration Commands

Execute the following command in tux2, tux3 and tux4:

```
$ echo $'search netlab.fe.up.pt\nnameserver 10.227.20.3' > /etc/resolv.conf
```

5.4. Relevant Logs

We tested the DNS by pinging google.com from tux3. The DNS query returned a Type A record with the IPv4 address of google.com (142.250.200.142).

The DNS communication involved two types of packets: the DNS Query packet (image 30), which contains the hostname (google.com) being queried and specifies the query type (Type A for IPv4), and the DNS Query Response packet (image 31), which includes the resolved IP address (the answer to the query), along with additional information such as the TTL (Time To Live) value that determines how long the response is valid.

In addition, the response includes other information, such as authority and additional records, depending on the type of query and the server's records. This exchange of packets ensures that the hostname is properly translated into its corresponding address and that all relevant details for further use are communicated efficiently.

Exp 6 - TCP connections

6.1. Network Architecture

The network architecture for this experiment remains unchanged from the previous setup (image 32).

6.2. Experiment Objectives

The objective is to test the integration of the previously developed download application by downloading a file from the web using the FTP protocol.

6.3. Main Configuration Commands

No additional configuration commands were necessary for this experiment.

6.4. Relevant Logs

Since the Netlab FTP server was unavailable, we downloaded the file from *ftp://ftp.up.pt/pub/gnu/emacs/elisp-manual-21-2.8.tar.gz*. This was successful, as shown in Part 1 of the report.

The logs show that two TCP connections were opened by the FTP application (Image 33). The first connection was for FTP control, which exchanged [SYN] and [SYN, ACK] packets to establish the connection. The control information was sent through port 49566 on our machine and port 21 on the FTP server (Image 34). The second connection was used for data transfer. The file data was sent between port 54624 on our machine and port 58150 on the server (Image 35). Once the file transfer was complete, the connection was closed with [FIN, ACK] packets (Image 36).

We also examined how the ARQ (Automatic Repeat reQuest) mechanism works in TCP. ARQ ensures that data is sent reliably by retransmitting lost or duplicate packets. Important fields in this process include the sequence number, which identifies each TCP packet, and the acknowledgment number, which tells the sender the next packet expected. We noticed that the acknowledgment number of one packet matched the sequence number of the next packet, as we can see in Images 37, 38 and 39. This confirms that the ARQ mechanism was working as expected. The window size field, which controls how much data can be sent before waiting for acknowledgment, also played a key role.

Due to problems with the lab machines, we couldn't complete Step 5, where we would start a second file download from tux2 while the first download was ongoing. If this step had been performed, we would expect to see TCP's congestion control mechanism in action. The throughput of the first download would decrease temporarily as the second connection started, as both would share the available bandwidth. Over time, both downloads would stabilize and share the network resources fairly (image 40 and 41). Wireshark would show how throughput changes during this process, especially during the slow start and congestion avoidance phases of TCP.

Conclusions

In this project, we successfully developed an FTP download application and configured a functional network. This allowed us to gain hands-on experience with key network protocols like FTP, NAT, and DNS. We also learned how to set up and manage network devices, including bridges, Linux systems, and commercial routers. Using Wireshark, we analyzed network traffic and deepened our understanding of TCP mechanisms, such as error handling and retransmissions. Overall, this project provided practical insights into network configuration and troubleshooting, enhancing our understanding of how the network layer works in real-world applications.

References

- RFC 959 - File Transfer Protocol
- Lab 2 Guide available on Moodle

Annexes

Image 1 - Part 1 : Running the download application

Return to Part 1

```
chill-ry@Chill RCOM-2 % make run
gcc -Wall -o bin//main main.c src//app_download.c -Iinclude/
./bin//main ftp://ftp.up.pt/pub/gnu/emacs/elisp-manual-21-2.8.tar.gz
Received argument: ftp://ftp.up.pt/pub/gnu/emacs/elisp-manual-21-2.8.tar.gz
Starting Download Application
  - User: anonymous
  - Password: anonymous
  - Host: ftp.up.pt
  - Host Name: mirrors.up.pt
  - IP Address: 193.137.29.15
  - URL Path: pub/gnu/emacs/elisp-manual-21-2.8.tar.gz
  - File Name: elisp-manual-21-2.8.tar.gz

Successfully connected to 193.137.29.15 on port 21

[220] Welcome to the University of Porto's mirror archive (mirrors.up.pt)
----------------------------------------------------------------


All connections and transfers are logged. The max number of connections is 200.

For more information please visit our website: http://mirrors.up.pt/
Questions and comments can be sent to mirrors@uporto.pt



[331] Please specify the password.
[230] Login successful.
[227] Entering Passive Mode (193,137,29,15,231,106).
[150] Opening BINARY mode data connection for pub/gnu/emacs/elisp-manual-21-2.8.tar.gz (2455995 bytes).
[INFO] File transfer completed, total bytes: 2455995
[226] Transfer complete.
[221] Goodbye.
```

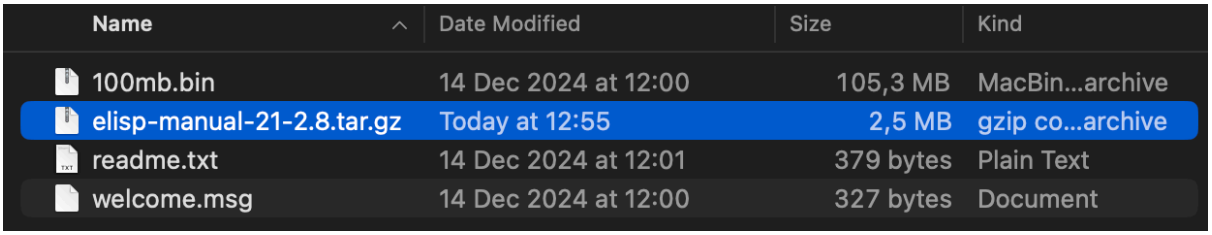
Image 2 - Part 1 : Wireshark FTP

Return to Part 1

No.	Time	Source	Destination	Protocol	Length	Info
136	32.322549997	127.0.0.1	127.0.0.53	DNS	82	Standard query 0x3083 A ftp.up.pt OPT
137	32.322902557	10.227.155.55	10.227.244.110	DNS	82	Standard query 0x0be7 A ftp.up.pt OPT
138	32.345074694	10.227.244.110	10.227.155.55	DNS	120	Standard query response 0x0be7 A ftp.up.pt CNAME mirrors.up.pt A 193.137.29.15 OPT
139	32.345301171	127.0.0.53	127.0.0.1	DNS	120	Standard query response 0x3083 A ftp.up.pt CNAME mirrors.up.pt A 193.137.29.15 OPT
140	32.345514964	10.227.155.55	193.137.29.15	TCP	76	49566 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2366728846 TSecr=0 WS=128
141	32.351500807	193.137.29.15	10.227.155.55	TCP	76	21 → 49566 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1380 SACK_PERM TSval=3256368339 TSecr=2366728
142	32.351559865	10.227.155.55	193.137.29.15	TCP	68	49566 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2366728852 TSecr=3256368339
143	32.358983695	193.137.29.15	10.227.155.55	FTP	141	Response: 220-Welcome to the University of Porto's mirror archive (mirrors.up.pt)
144	32.359012679	10.227.155.55	193.137.29.15	TCP	68	49566 → 21 [ACK] Seq=1 Ack=74 Win=64256 Len=0 TSval=2366728860 TSecr=3256368346
145	32.360039600	193.137.29.15	10.227.155.55	FTP	143	Response: 220---
146	32.360065147	10.227.155.55	193.137.29.15	TCP	68	49566 → 21 [ACK] Seq=1 Ack=149 Win=64256 Len=0 TSval=2366728861 TSecr=3256368346
147	32.361108158	193.137.29.15	10.227.155.55	FTP	233	Response: 220-All connections and transfers are logged. The max number of connections is 200.
148	32.361130759	10.227.155.55	193.137.29.15	TCP	68	49566 → 21 [ACK] Seq=1 Ack=314 Win=64128 Len=0 TSval=2366728862 TSecr=3256368346
149	32.362188737	193.137.29.15	10.227.155.55	FTP	147	Response: 220-Questions and comments can be sent to mirrors@uporto.pt
150	32.362218772	10.227.155.55	193.137.29.15	TCP	68	49566 → 21 [ACK] Seq=1 Ack=393 Win=64128 Len=0 TSval=2366728863 TSecr=3256368346
151	32.362441583	10.227.155.55	193.137.29.15	FTP	84	Request: USER anonymous
152	32.367956358	193.137.29.15	10.227.155.55	TCP	68	21 → 49566 [ACK] Seq=393 Ack=17 Win=65280 Len=0 TSval=3256368355 TSecr=2366728863
153	32.368829255	193.137.29.15	10.227.155.55	FTP	102	Response: 331 Please specify the password.
154	32.368989199	10.227.155.55	193.137.29.15	FTP	84	Request: PASS anonymous
155	32.375643573	193.137.29.15	10.227.155.55	FTP	91	Response: 230 Login successful.
156	32.375764075	10.227.155.55	193.137.29.15	FTP	74	Request: PASV
157	32.381416062	193.137.29.15	10.227.155.55	FTP	119	Response: 227 Entering Passive Mode (193,137,29,15,227,38).
158	32.381633493	10.227.155.55	193.137.29.15	TCP	76	54624 → 58150 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2366728883 TSecr=0 WS=128
159	32.384567155	193.137.29.15	10.227.155.55	TCP	76	58150 → 54624 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1380 SACK_PERM TSval=3256368373 TSecr=2366
160	32.384608551	10.227.155.55	193.137.29.15	TCP	68	54624 → 58150 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2366728886 TSecr=3256368373
161	32.384647723	10.227.155.55	193.137.29.15	FTP	115	Request: RETR pub/gnu/emacs/elisp-manual-21-2.8.tar.gz
162	32.390783200	193.137.29.15	10.227.155.55	FTP-DATA	1436	FTP Data: 1368 bytes (PASV) (RETR pub/gnu/emacs/elisp-manual-21-2.8.tar.gz)
163	32.390840165	10.227.155.55	193.137.29.15	TCP	68	54624 → 58150 [ACK] Seq=1 Ack=1369 Win=64128 Len=0 TSval=2366728892 TSecr=3256368378
164	32.390866093	193.137.29.15	10.227.155.55	FTP-DATA	1436	FTP Data: 1368 bytes (PASV) (RETR pub/gnu/emacs/elisp-manual-21-2.8.tar.gz)
165	32.390883535	10.227.155.55	193.137.29.15	TCP	68	54624 → 58150 [ACK] Seq=1 Ack=2737 Win=64128 Len=0 TSval=2366728892 TSecr=3256368378
166	32.391688738	193.137.29.15	10.227.155.55	FTP-DATA	2804	FTP Data: 2736 bytes (PASV) (RETR pub/gnu/emacs/elisp-manual-21-2.8.tar.gz)
167	32.391772703	10.227.155.55	193.137.29.15	TCP	68	54624 → 58150 [ACK] Seq=1 Ack=5473 Win=64128 Len=0 TSval=2366728893 TSecr=3256368378
168	32.391903554	193.137.29.15	10.227.155.55	FTP-DATA	1436	FTP Data: 1368 bytes (PASV) (RETR pub/gnu/emacs/elisp-manual-21-2.8.tar.gz)
169	32.391922929	10.227.155.55	193.137.29.15	TCP	68	54624 → 58150 [ACK] Seq=1 Ack=6841 Win=64128 Len=0 TSval=2366728893 TSecr=3256368378
170	32.392646321	193.137.29.15	10.227.155.55	FTP-DATA	1436	FTP Data: 1368 bytes (PASV) (RETR pub/gnu/emacs/elisp-manual-21-2.8.tar.gz)
171	32.392656490	10.227.155.55	193.137.29.15	TCP	68	54624 → 58150 [ACK] Seq=1 Ack=8209 Win=64128 Len=0 TSval=2366728894 TSecr=3256368378
172	32.393080270	193.137.29.15	10.227.155.55	FTP-DATA	1436	FTP Data: 1368 bytes (PASV) (RETR pub/gnu/emacs/elisp-manual-21-2.8.tar.gz)
173	32.393099626	10.227.155.55	193.137.29.15	TCP	68	54624 → 58150 [ACK] Seq=1 Ack=9577 Win=64128 Len=0 TSval=2366728894 TSecr=3256368378
174	32.393642796	193.137.29.15	10.227.155.55	FTP-DATA	1436	FTP Data: 1368 bytes (PASV) (RETR pub/gnu/emacs/elisp-manual-21-2.8.tar.gz)

Image 3 - Part 1 : Files Downloaded

Return to Part 1



Name	Date Modified	Size	Kind
100mb.bin	14 Dec 2024 at 12:00	105,3 MB	MacBin...archive
elisp-manual-21-2.8.tar.gz	Today at 12:55	2,5 MB	gzip co...archive
readme.txt	14 Dec 2024 at 12:01	379 bytes	Plain Text
welcome.msg	14 Dec 2024 at 12:00	327 bytes	Document

Image 4 - Part 2 Exp 1

Return to 1.1

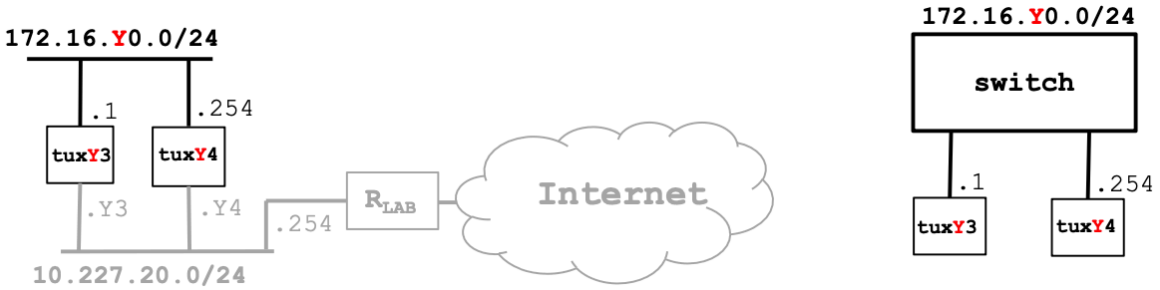


Image 5 - Ping Packets

Return to 1.4

	Time	Source	Destination	Protocol	Length	Info
7	11.484017445	3Com_9f:81:2e	Broadcast	ARP	42	Who has 172.16.40.254? Tell 172.16.40.1
8	11.484173890	KYE_02:55:95	3Com_9f:81:2e	ARP	60	172.16.40.254 is at 00:c0:df:02:55:95
9	11.484195680	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=1/256, ttl=64 (reply in 10)
10	11.484315807	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=1/256, ttl=64 (request in 9)
11	12.013029035	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
12	12.511382542	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=2/512, ttl=64 (reply in 13)
13	12.511512238	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=2/512, ttl=64 (request in 12)
14	13.535386947	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=3/768, ttl=64 (reply in 15)
15	13.535546884	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=3/768, ttl=64 (request in 14)
16	14.015220156	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
17	14.559403295	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=4/1024, ttl=64 (reply in 18)
18	14.559533339	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=4/1024, ttl=64 (request in 17)
19	15.583385910	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=5/1280, ttl=64 (reply in 20)
20	15.583514068	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=5/1280, ttl=64 (request in 19)

Image 6 - Part 2 Exp 1 : ARP packet sent by tux3

Return to 1.4

No.	Time	Source	Destination	Protocol	Length	Info
7	11.484017445	3Com_9f:81:2e	Broadcast	ARP	42	Who has 172.16.40.254? Tell 172.16.40.1
8	11.484173890	KYE_02:55:95	3Com_9f:81:2e	ARP	60	172.16.40.254 is at 00:c0:df:02:55:95
9	11.484195680	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=1/256, ttl=64 (reply in 10)
10	11.484315807	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=1/256, ttl=64 (request in 9)
11	12.013029035	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
12	12.511382542	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=2/512, ttl=64 (reply in 13)
13	12.511512238	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=2/512, ttl=64 (request in 12)
14	13.535386947	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=3/768, ttl=64 (reply in 15)
15	13.535546884	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=3/768, ttl=64 (request in 14)
16	14.015220156	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
17	14.559403295	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=4/1024, ttl=64 (reply in 18)
18	14.559533339	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=4/1024, ttl=64 (request in 17)
19	15.583385910	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=5/1280, ttl=64 (reply in 20)
20	15.583514068	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=5/1280, ttl=64 (request in 19)
21	16.017391092	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001

```
> Frame 7: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface eth1, id 0
v Ethernet II, Src: 3Com_9f:81:2e (00:01:02:9f:81:2e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  > Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  > Source: 3Com_9f:81:2e (00:01:02:9f:81:2e)
    Type: ARP (0x0806)
    [Stream index: 1]
> Address Resolution Protocol (request)
```

Image 7 - Part 2 Exp 1 : ARP packet replied by tux4

Return to 1.4

No.	Time	Source	Destination	Protocol	Length	Info
7	11.484017445	3Com_9f:81:2e	Broadcast	ARP	42	Who has 172.16.40.254? Tell 172.16.40.1
8	11.484173890	KYE_02:55:95	3Com_9f:81:2e	ARP	60	172.16.40.254 is at 00:c0:df:02:55:95
9	11.484195680	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=1/256, ttl=64 (reply in 10)
10	11.484315807	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=1/256, ttl=64 (request in 9)
11	12.013029035	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
12	12.511382542	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=2/512, ttl=64 (reply in 13)
13	12.511512238	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=2/512, ttl=64 (request in 12)
14	13.535386947	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=3/768, ttl=64 (reply in 15)
15	13.535546884	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=3/768, ttl=64 (request in 14)
16	14.015220156	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
17	14.559403295	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=4/1024, ttl=64 (reply in 18)
18	14.559533339	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=4/1024, ttl=64 (request in 17)
19	15.583385910	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=5/1280, ttl=64 (reply in 20)
20	15.583514068	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=5/1280, ttl=64 (request in 19)
21	16.017391092	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001

```
> Frame 8: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth1, id 0
v Ethernet II, Src: KYE_02:55:95 (00:c0:df:02:55:95), Dst: 3Com_9f:81:2e (00:01:02:9f:81:2e)
  > Destination: 3Com_9f:81:2e (00:01:02:9f:81:2e)
  > Source: KYE_02:55:95 (00:c0:df:02:55:95)
    Type: ARP (0x0806)
    [Stream index: 2]
    Padding: 000000000000000000000000000000000000
> Address Resolution Protocol (reply)
```

Image 8 - Part 2 Exp 1 : Header of ARP type packets

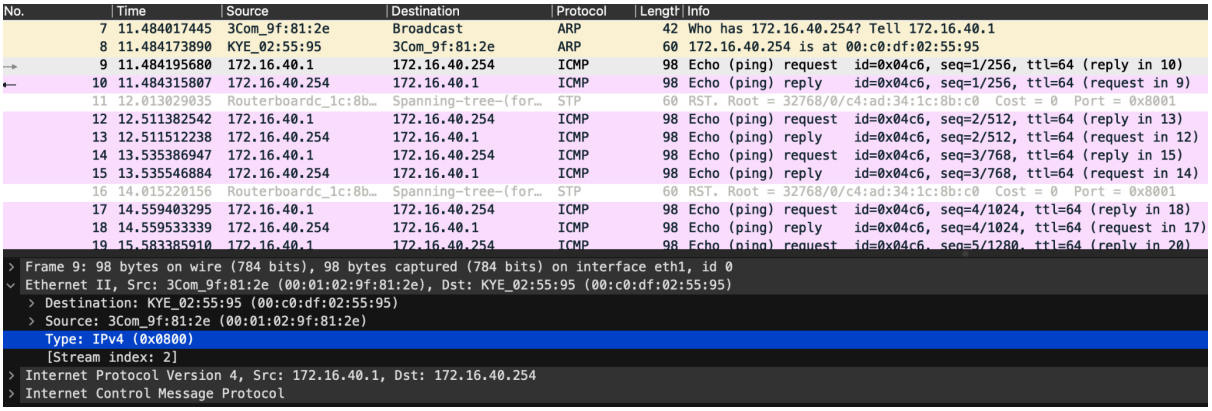
Return to 1.4

No.	Time	Source	Destination	Protocol	Length	Info
7	11.484017445	3Com_9f:81:2e	Broadcast	ARP	42	Who has 172.16.40.254? Tell 172.16.40.1
8	11.484173890	KYE_02:55:95	3Com_9f:81:2e	ARP	60	172.16.40.254 is at 00:c0:df:02:55:95
9	11.484195680	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=1/256, ttl=64 (reply in 10)
10	11.484315807	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=1/256, ttl=64 (request in 9)
11	12.013029035	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
12	12.511382542	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=2/512, ttl=64 (reply in 13)
13	12.511512238	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=2/512, ttl=64 (request in 12)
14	13.535386947	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=3/768, ttl=64 (reply in 15)
15	13.535546884	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=3/768, ttl=64 (request in 14)
16	14.015220156	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
17	14.559403295	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=4/1024, ttl=64 (reply in 18)
18	14.559533339	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=4/1024, ttl=64 (request in 17)
19	15.583385910	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=5/1280, ttl=64 (reply in 20)

```
> Frame 7: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface eth1, id 0
v Ethernet II, Src: 3Com_9f:81:2e (00:01:02:9f:81:2e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  > Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  > Source: 3Com_9f:81:2e (00:01:02:9f:81:2e)
    Type: ARP (0x0806)
    [Stream index: 1]
> Address Resolution Protocol (request)
```

Image 9 - Part 2 Exp 1 : Header of IP type packets

Return to 1.4



No.	Time	Source	Destination	Protocol	Length	Info
7	11.484017445	3Com_9f:81:2e	Broadcast	ARP	42	Who has 172.16.40.254? Tell 172.16.40.1
8	11.484173890	KYE_02:55:95	3Com_9f:81:2e	ARP	60	172.16.40.254 is at 00:c0:df:02:55:95
9	11.484195680	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=1/256, ttl=64 (reply in 10)
10	11.484315807	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=1/256, ttl=64 (request in 9)
11	12.013029035	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
12	12.511382542	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=2/512, ttl=64 (reply in 13)
13	12.511512238	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=2/512, ttl=64 (request in 12)
14	13.535386947	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=3/768, ttl=64 (reply in 15)
15	13.535546884	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=3/768, ttl=64 (request in 14)
16	14.015220156	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
17	14.559403295	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=4/1024, ttl=64 (reply in 18)
18	14.559533339	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=4/1024, ttl=64 (request in 17)
19	15.583385910	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=5/1280, ttl=64 (reply in 20)

```
> Frame 9: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth1, id 0
v Ethernet II, Src: 3Com_9f:81:2e (00:01:02:9f:81:2e), Dst: KYE_02:55:95 (00:c0:df:02:55:95)
  > Destination: KYE_02:55:95 (00:c0:df:02:55:95)
  > Source: 3Com_9f:81:2e (00:01:02:9f:81:2e)
    Type: IPv4 (0x0800)
    [Stream index: 2]
> Internet Protocol Version 4, Src: 172.16.40.1, Dst: 172.16.40.254
> Internet Control Message Protocol
```

Image 10 - Part 2 Exp 1 : Header of ICMP type packets

Return to 1.4

No.	Time	Source	Destination	Protocol	Length	Info
9	11.484195680	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=1/256, ttl=64 (reply in 10)
10	11.484315807	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=1/256, ttl=64 (request in 9)
11	12.013029035	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
12	12.511382542	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=2/512, ttl=64 (reply in 13)
13	12.511512238	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=2/512, ttl=64 (request in 12)
14	13.535386947	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=3/768, ttl=64 (reply in 15)
15	13.535546884	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=3/768, ttl=64 (request in 14)
16	14.015220156	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
17	14.559403295	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=4/1024, ttl=64 (reply in 18)
18	14.559533339	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=4/1024, ttl=64 (request in 17)
19	15.583385910	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=5/1280, ttl=64 (reply in 20)
20	15.583514068	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=5/1280, ttl=64 (request in 19)

```
> Frame 12: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth1, id 0
v Ethernet II, Src: 3Com_9f:81:2e (00:01:02:9f:81:2e), Dst: KYE_02:55:95 (00:c0:df:02:55:95)
  > Destination: KYE_02:55:95 (00:c0:df:02:55:95)
  > Source: 3Com_9f:81:2e (00:01:02:9f:81:2e)
    Type: IPv4 (0x0800)
    [Stream index: 2]
v Internet Protocol Version 4, Src: 172.16.40.1, Dst: 172.16.40.254
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x8a0c (35340)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: ICMP (1)
```

Image 11 - Part 2 Exp 1 : Frame Length

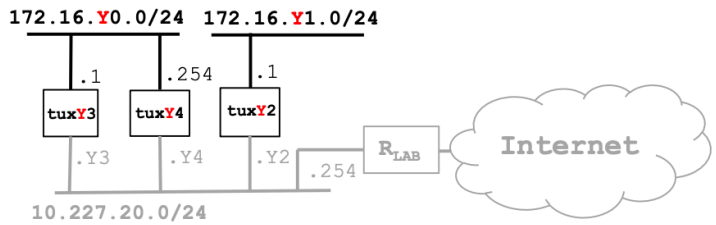
Return to 1.4

No.	Time	Source	Destination	Protocol	Length	Info
9	11.484195680	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=1/256, ttl=64 (reply in 10)
10	11.484315807	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=1/256, ttl=64 (request in 9)
11	12.013029035	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
12	12.511382542	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=2/512, ttl=64 (reply in 13)
13	12.511512238	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=2/512, ttl=64 (request in 12)
14	13.535386947	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=3/768, ttl=64 (reply in 15)
15	13.535546884	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=3/768, ttl=64 (request in 14)
16	14.015220156	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
17	14.559403295	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=4/1024, ttl=64 (reply in 18)
18	14.559533339	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x04c6, seq=4/1024, ttl=64 (request in 17)
19	15.583385910	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x04c6, seq=5/1280, ttl=64 (reply in 20)

```
Frame 12: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth1, id 0
    Section number: 1
  > Interface id: 0 (eth1)
    Encapsulation type: Ethernet (1)
    Arrival Time: Nov 27, 2024 12:10:36.950980057 WET
    UTC Arrival Time: Nov 27, 2024 12:10:36.950980057 UTC
    Epoch Arrival Time: 1732709436.950980057
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 0.498353507 seconds]
    [Time delta from previous displayed frame: 0.498353507 seconds]
    [Time since reference or first frame: 12.511382542 seconds]
    Frame Number: 12
    Frame Length: 98 bytes (784 bits)
    Capture Length: 98 bytes (784 bits)
```

Image 12 - Part 2 Exp 2

Return to 2.1



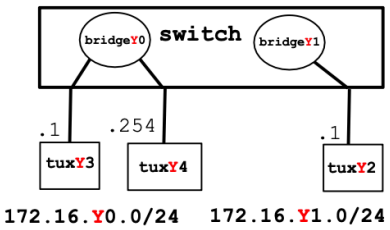


Image 13 - Part 2 Exp 2 : Ping Broadcast from tux3 to tux4

Return to 2.4

No.	Time	Source	Destination	Protocol	Length	Info
13	22.649389228	0.0.0.0	255.255.255.255	MNDP	159	5678 → 5678 Len=117
14	22.649424917	Routerboardc_1c:8b…	CDP/VTP/DTP/PAgP/U…	CDP	93	Device ID: MikroTik Port ID: bridge40
15	22.649475551	Routerboardc_1c:8b…	LLDP_Multicast	LLDP	110	MA/c4:ad:34:1c:8b:c0 IN/bridge40 120 SysN=MikroTik SysD=MikroTik RouterC
16	24.026418807	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8002
17	26.028630478	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8002
18	28.030831952	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8002
19	30.033026722	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8002
20	32.035227917	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8002
21	34.037429740	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8002
22	36.038993983	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8002
23	38.041197832	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8002
24	40.043390297	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8002
25	42.045588629	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8002
26	43.485360084	172.16.40.1	172.16.40.255	ICMP	98	Echo (ping) request id=0x05a5, seq=1/256, ttl=64 (no response found!)
27	44.047795551	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8002
28	44.492571161	172.16.40.1	172.16.40.255	ICMP	98	Echo (ping) request id=0x05a5, seq=2/512, ttl=64 (no response found!)
29	45.516511101	172.16.40.1	172.16.40.255	ICMP	98	Echo (ping) request id=0x05a5, seq=3/768, ttl=64 (no response found!)
30	45.761679896	fe80::2c0:dfff:fe0…	ff02::2	ICMPv6	70	Router Solicitation from 00:c0:df:02:55:95
31	46.049186309	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8002
32	46.540449224	172.16.40.1	172.16.40.255	ICMP	98	Echo (ping) request id=0x05a5, seq=4/1024, ttl=64 (no response found!)
33	47.564396986	172.16.40.1	172.16.40.255	ICMP	98	Echo (ping) request id=0x05a5, seq=5/1280, ttl=64 (no response found!)
34	48.051380869	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8002
35	48.588396221	172.16.40.1	172.16.40.255	ICMP	98	Echo (ping) request id=0x05a5, seq=6/1536, ttl=64 (no response found!)
36	49.612296490	172.16.40.1	172.16.40.255	ICMP	98	Echo (ping) request id=0x05a5, seq=7/1792, ttl=64 (no response found!)
37	50.053583182	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8002
38	50.636248582	172.16.40.1	172.16.40.255	ICMP	98	Echo (ping) request id=0x05a5, seq=8/2048, ttl=64 (no response found!)
39	51.660221906	172.16.40.1	172.16.40.255	ICMP	98	Echo (ping) request id=0x05a5, seq=9/2304, ttl=64 (no response found!)
40	52.055771317	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8002

```
> Frame 26: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth1, id 0
v Ethernet II, Src: 3Com_9f:81:2e (00:01:02:9f:81:2e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
    > Destination: Broadcast (ff:ff:ff:ff:ff:ff)
    > Source: 3Com_9f:81:2e (00:01:02:9f:81:2e)
      Type: IPv4 (0x0800)
      [Stream index: 4]
> Internet Protocol Version 4, Src: 172.16.40.1, Dst: 172.16.40.255
> Internet Control Message Protocol
```

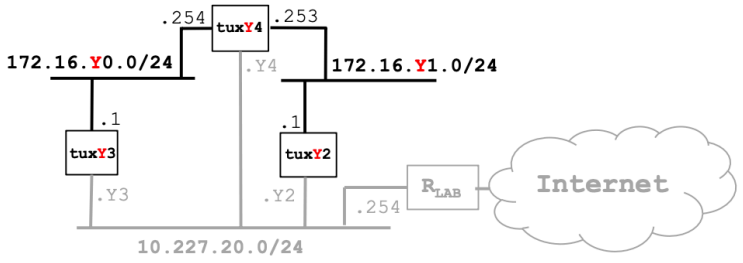
Image 14 - Part 2 Exp 2 : Ping Broadcast from tux3 to tux2

Return to 2.4

No.	Time	Source	Destination	Protocol	Length	Info
22	42.045528629	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:be Cost = 0 Port = 0x8001
23	44.047720702	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:be Cost = 0 Port = 0x8001
24	46.049921993	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:be Cost = 0 Port = 0x8001
25	48.052125450	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:be Cost = 0 Port = 0x8001
26	50.054339104	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:be Cost = 0 Port = 0x8001
27	52.056537184	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:be Cost = 0 Port = 0x8001
28	54.058731353	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:be Cost = 0 Port = 0x8001
29	56.060930760	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:be Cost = 0 Port = 0x8001
30	56.686269309	0.0.0.0	255.255.255.255	MNDP	159	5678 → 5678 Len=117
31	56.686283277	Routerboardc_1c:8b…	CDP/VTP/DTP/PAgP/U…	CDP	93	Device ID: MikroTik Port ID: bridge41
32	56.686319525	Routerboardc_1c:8b…	LLDP_Multicast	LLDP	110	MA/c4:ad:34:1c:8b:be IN/bridge41 120 SysN=MikroTik SysD=MikroTi
33	58.063141203	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:be Cost = 0 Port = 0x8001
34	60.065352624	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:be Cost = 0 Port = 0x8001
35	62.067551124	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:be Cost = 0 Port = 0x8001
36	64.069744806	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:be Cost = 0 Port = 0x8001
37	66.071945332	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:be Cost = 0 Port = 0x8001
38	68.074148373	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:be Cost = 0 Port = 0x8001

Image 15 - Part 2 Exp 3

Return to 3.1



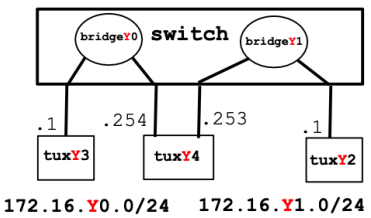


Image 16 - Part 2 Exp 3 : tux3 ping 172.16.40.254 reply

Return to 3.4

No.	Time	Source	Destination	Protocol	Length	Info
18	28.031260029	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
19	30.033488288	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
20	31.129465829	3Com_9f:81:2e	Broadcast	ARP	42	Who has 172.16.40.254? Tell 172.16.40.1
21	31.129622275	KYE_02:55:95	3Com_9f:81:2e	ARP	60	172.16.40.254 is at 00:c0:df:02:55:95
22	31.129645811	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x345b, seq=1/256, ttl=64 (reply in 23)
23	31.129768314	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x345b, seq=1/256, ttl=64 (request in 22)
24	32.035710960	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
25	32.161577290	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x345b, seq=2/512, ttl=64 (reply in 26)
26	32.161704193	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x345b, seq=2/512, ttl=64 (request in 25)
27	33.185573045	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x345b, seq=3/768, ttl=64 (reply in 28)
28	33.185702951	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x345b, seq=3/768, ttl=64 (request in 27)

Image 17 - Part 2 Exp 3 : tux3 ping 172.16.41.253 reply

Return to 3.4

No.	Time	Source	Destination	Protocol	Length	Info
34	36.040182775	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
35	36.257578120	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x345b, seq=6/1536, ttl=64 (reply in 36)
36	36.257724299	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x345b, seq=6/1536, ttl=64 (request in 35)
37	36.287653468	KYE_02:55:95	3Com_9f:81:2e	ARP	60	Who has 172.16.40.1? Tell 172.16.40.254
38	36.287682522	3Com_9f:81:2e	KYE_02:55:95	ARP	42	172.16.40.1 is at 00:01:02:9f:81:2e
39	38.042419974	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
40	40.044660875	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
41	42.046885154	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
42	43.225400883	172.16.40.1	172.16.41.253	ICMP	98	Echo (ping) request id=0x3465, seq=1/256, ttl=64 (reply in 43)
43	43.225554117	172.16.41.253	172.16.40.1	ICMP	98	Echo (ping) reply id=0x3465, seq=1/256, ttl=64 (request in 42)
44	44.049127731	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
45	44.257580231	172.16.40.1	172.16.41.253	ICMP	98	Echo (ping) request id=0x3465, seq=2/512, ttl=64 (reply in 46)
46	44.257710417	172.16.41.253	172.16.40.1	ICMP	98	Echo (ping) reply id=0x3465, seq=2/512, ttl=64 (request in 45)
47	45.281577733	172.16.40.1	172.16.41.253	ICMP	98	Echo (ping) request id=0x3465, seq=3/768, ttl=64 (reply in 48)
48	45.281705334	172.16.41.253	172.16.40.1	ICMP	98	Echo (ping) reply id=0x3465, seq=3/768, ttl=64 (request in 47)

Image 18 - Part 2 Exp 3 : tux3 ping 172.16.41.1 reply

Return to 3.4

No.	Time	Source	Destination	Protocol	Length	Info
51	46.305721692	172.16.41.253	172.16.40.1	ICMP	98	Echo (ping) reply id=0x3465, seq=4/1024, ttl=64 (request in 50)
52	47.329582303	172.16.40.1	172.16.41.253	ICMP	98	Echo (ping) request id=0x3465, seq=5/1280, ttl=64 (reply in 53)
53	47.329728622	172.16.41.253	172.16.40.1	ICMP	98	Echo (ping) reply id=0x3465, seq=5/1280, ttl=64 (request in 52)
54	48.053593958	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
55	48.353575893	172.16.40.1	172.16.41.253	ICMP	98	Echo (ping) request id=0x3465, seq=6/1536, ttl=64 (reply in 56)
56	48.353706358	172.16.41.253	172.16.40.1	ICMP	98	Echo (ping) reply id=0x3465, seq=6/1536, ttl=64 (request in 55)
57	50.055822218	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
58	52.058115570	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
59	54.060370160	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
60	54.345561382	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x346c, seq=1/256, ttl=64 (reply in 61)
61	54.345981202	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) reply id=0x346c, seq=1/256, ttl=63 (request in 60)
62	55.361576924	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x346c, seq=2/512, ttl=64 (reply in 63)
63	55.361813060	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) reply id=0x346c, seq=2/512, ttl=63 (request in 62)
64	56.062603239	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8001
65	56.385554521	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x346c, seq=3/768, ttl=64 (reply in 66)
66	56.385815939	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) reply id=0x346c, seq=3/768, ttl=63 (request in 65)

Image 19 - Part 2 Exp 3 : eth1 of tux4 ARP packets

Return to 3.4

No.	Time	Source	Destination	Protocol	Length	Info
68	115.2643500…	3Com_9f:81:2e	Broadcast	ARP	60	Who has 172.16.40.254? Tell 172.16.40.1
69	115.2643723…	KYE_02:55:95	3Com_9f:81:2e	ARP	42	172.16.40.254 is at 00:c0:df:02:55:95
70	115.2644795…	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x35fa, seq=1/256, ttl=64 (reply in 71)
71	115.2647838…	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) reply id=0x35fa, seq=1/256, ttl=63 (request in 70)
72	116.1153099…	0.0.0.0	255.255.255.255	MNDP	159	5678 → 5678 Len=117
73	116.1153437…	Routerboardc_1c:8b…	CDP/VTP/DTP/PAgP/U…	CDP	93	Device ID: MikroTik Port ID: bridge40
74	116.1153928…	Routerboardc_1c:8b…	LLDP_Multicast	LLDP	110	MA/c4:ad:34:1c:8b:c0 IN/bridge40 120 SysN=MikroTik SysD=MikroTik Rc
75	116.1288766…	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8002
76	116.2794025…	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x35fa, seq=2/512, ttl=64 (reply in 77)
77	116.2795364…	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) reply id=0x35fa, seq=2/512, ttl=63 (request in 76)
78	117.3034082…	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x35fa, seq=3/768, ttl=64 (reply in 79)
79	117.3035488…	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) reply id=0x35fa, seq=3/768, ttl=63 (request in 78)
80	118.1311385…	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8002
81	118.3274135…	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x35fa, seq=4/1024, ttl=64 (reply in 82)
82	118.3275799…	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) reply id=0x35fa, seq=4/1024, ttl=63 (request in 81)
83	119.3514166…	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x35fa, seq=5/1280, ttl=64 (reply in 84)
84	119.3515723…	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) reply id=0x35fa, seq=5/1280, ttl=63 (request in 83)
85	120.1333851…	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c0 Cost = 0 Port = 0x8002
86	120.3701935…	KYE_02:55:95	3Com_9f:81:2e	ARP	42	Who has 172.16.40.1? Tell 172.16.40.254
87	120.3703102…	3Com_9f:81:2e	KYE_02:55:95	ARP	60	172.16.40.1 is at 00:01:02:9f:81:2e
88	120.3754096…	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x35fa, seq=6/1536, ttl=64 (reply in 89)
89	120.3755569…	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) reply id=0x35fa, seq=6/1536, ttl=63 (request in 88)

Image 20 - Part 2 Exp 3 : eth2 of tux4 ARP packets

Return to 3.4

No.	Time	Source	Destination	Protocol	Length	Info
56	103.2510366…	3Com_a0:ad:91	Broadcast	ARP	42	Who has 172.16.41.1? Tell 172.16.41.253
57	103.2511703…	Netronix_b5:8c:8f	3Com_a0:ad:91	ARP	60	172.16.41.1 is at 00:e0:7d:b5:8c:8f
58	103.2511916…	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x35fa, seq=1/256, ttl=63 (reply in 59)
59	103.2513110…	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) reply id=0x35fa, seq=1/256, ttl=64 (request in 58)
60	104.1020432…	0.0.0.0	255.255.255.255	MNDP	159	5678 → 5678 Len=117
61	104.1020765…	Routerboardc_1c:8b…	CDP/VTP/DTP/PAgP/U…	CDP	93	Device ID: MikroTik Port ID: bridge41
62	104.1021241…	Routerboardc_1c:8b…	LLDP_Multicast	LLDP	110	MA/c4:ad:34:1c:8b:bc IN/bridge41 120 SysN=MikroTik SysD=MikroTik Rc
63	104.1153932…	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8002
64	104.2659595…	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x35fa, seq=2/512, ttl=63 (reply in 65)
65	104.2660720…	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) reply id=0x35fa, seq=2/512, ttl=64 (request in 64)
66	105.2899689…	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x35fa, seq=3/768, ttl=63 (reply in 67)
67	105.2900838…	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) reply id=0x35fa, seq=3/768, ttl=64 (request in 66)
68	106.1176525…	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8002
69	106.3139772…	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x35fa, seq=4/1024, ttl=63 (reply in 70)
70	106.3141154…	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) reply id=0x35fa, seq=4/1024, ttl=64 (request in 69)
71	107.3379806…	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x35fa, seq=5/1280, ttl=63 (reply in 72)
72	107.3380960…	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) reply id=0x35fa, seq=5/1280, ttl=64 (request in 71)
73	108.1199033…	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8002
74	108.3619740…	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x35fa, seq=6/1536, ttl=63 (reply in 75)
75	108.3620875…	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) reply id=0x35fa, seq=6/1536, ttl=64 (request in 74)
76	108.3985671…	Netronix_b5:8c:8f	3Com_a0:ad:91	ARP	60	Who has 172.16.41.253? Tell 172.16.41.1
77	108.3985775…	3Com_a0:ad:91	Netronix_b5:8c:8f	ARP	42	172.16.41.253 is at 00:01:02:a0:ad:91
78	109.3859872…	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x35fa, seq=7/1792, ttl=63 (reply in 79)

Image 21 - Part 2 Exp 4

Return to 4.1

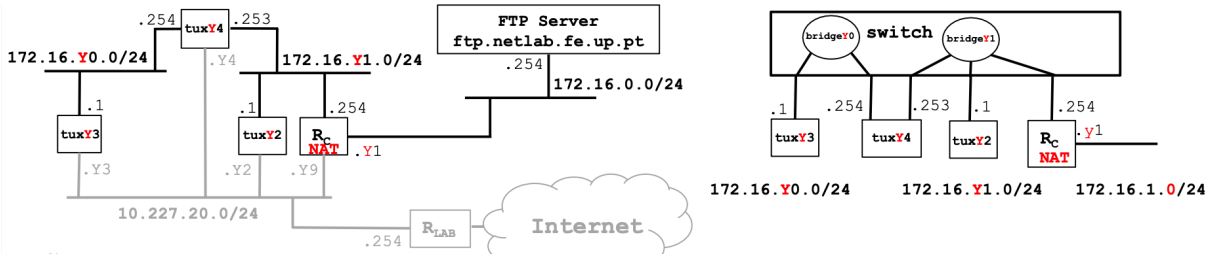


Image 22 - Part 2 Exp 4 : tux3 ping tux2

Return to 4.4

No.	Time	Source	Destination	Protocol	Length	Info
9	16.006146383	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
10	18.008310866	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
11	18.104156046	3Com_9f:81:2e	Broadcast	ARP	42	Who has 172.16.40.254? Tell 172.16.40.1
12	18.104310119	KYE_02:55:95	3Com_9f:81:2e	ARP	60	172.16.40.254 is at 00:c0:df:02:55:95
→ 13	18.104331142	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x1fe7, seq=1/256, ttl=64 (reply in 14)
← 14	18.104630348	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) reply id=0x1fe7, seq=1/256, ttl=63 (request in 13)
15	19.105102409	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x1fe7, seq=2/512, ttl=64 (reply in 16)
16	19.105341900	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) reply id=0x1fe7, seq=2/512, ttl=63 (request in 15)
17	20.010501960	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
18	20.129100542	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x1fe7, seq=3/768, ttl=64 (reply in 19)
19	20.129337728	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) reply id=0x1fe7, seq=3/768, ttl=63 (request in 18)
20	21.153103355	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x1fe7, seq=4/1024, ttl=64 (reply in 21)
21	21.153398929	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) reply id=0x1fe7, seq=4/1024, ttl=63 (request in 20)
22	22.012681042	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
23	22.177104282	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) request id=0x1fe7, seq=5/1280, ttl=64 (reply in 24)

Image 23 - Part 2 Exp 4 : tux3 ping tux4

Return to 4.4

No.	Time	Source	Destination	Protocol	Length	Info
35	36.017867599	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
36	38.020040892	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
→ 37	38.472788617	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x1ff4, seq=1/256, ttl=64 (reply in 38)
← 38	38.472940385	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x1ff4, seq=1/256, ttl=64 (request in 37)
39	39.489107138	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x1ff4, seq=2/512, ttl=64 (reply in 40)
40	39.489232646	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x1ff4, seq=2/512, ttl=64 (request in 39)
41	40.022206085	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
42	40.513105905	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x1ff4, seq=3/768, ttl=64 (reply in 43)
43	40.513229526	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x1ff4, seq=3/768, ttl=64 (request in 42)
44	41.537104742	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x1ff4, seq=4/1024, ttl=64 (reply in 45)
45	41.537258675	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x1ff4, seq=4/1024, ttl=64 (request in 44)
46	42.024255130	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
47	42.561105534	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x1ff4, seq=5/1280, ttl=64 (reply in 48)

Image 24 - Part 2 Exp 4 : tux3 ping Rc

Return to 4.4

No.	Time	Source	Destination	Protocol	Length	Info
54	46.028598790	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
55	48.030569336	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
56	50.032937077	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
→ 57	51.225195193	172.16.40.1	172.16.41.254	ICMP	98	Echo (ping) request id=0x1ffb, seq=1/256, ttl=64 (reply in 58)
← 58	51.225678434	172.16.41.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x1ffb, seq=1/256, ttl=63 (request in 57)
59	52.035121831	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
60	52.257104339	172.16.40.1	172.16.41.254	ICMP	98	Echo (ping) request id=0x1ffb, seq=2/512, ttl=64 (reply in 61)
61	52.257368274	172.16.41.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x1ffb, seq=2/512, ttl=63 (request in 60)
62	53.281105134	172.16.40.1	172.16.41.254	ICMP	98	Echo (ping) request id=0x1ffb, seq=3/768, ttl=64 (reply in 63)
63	53.281374796	172.16.41.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x1ffb, seq=3/768, ttl=63 (request in 62)
64	53.394258995	0.0.0.0	255.255.255.255	MNDP	159	5678 → 5678 Len=117
65	53.394291612	Routerboardc_1c:8b…	CDP/VTP/DTP/PAgP/U…	CDP	93	Device ID: MikroTik Port ID: bridge40
66	53.394341060	Routerboardc_1c:8b…	LLDP_Multicast	LLDP	110	MA/c4:ad:34:1c:8b:bd IN/bridge40 120 SysN=MikroTik SysD=MikroTik R
67	54.027281445	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
68	54.305107466	172.16.40.1	172.16.41.254	ICMP	98	Echo (ping) request id=0x1ffb, seq=4/1024, ttl=64 (reply in 69)
69	54.305381808	172.16.41.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x1ffb, seq=4/1024, ttl=63 (request in 68)

Image 25 - Part 2 Exp 4 : tux3 ping FTP server (Rc NAT enabled)

Return to 4.4

No.	Time	Source	Destination	Protocol	Length	Info
14	20.020616614	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
15	21.944890572	3Com_9f:81:2e	Broadcast	ARP	42	Who has 172.16.40.254? Tell 172.16.40.1
16	21.945041083	KYE_02:55:95	3Com_9f:81:2e	ARP	60	172.16.40.254 is at 00:c0:df:02:55:95
→ 17	21.945060988	172.16.40.1	172.16.1.10	ICMP	98	Echo (ping) request id=0x224c, seq=1/256, ttl=64 (reply in 18)
← 18	21.945586274	172.16.1.10	172.16.40.1	ICMP	98	Echo (ping) reply id=0x224c, seq=1/256, ttl=62 (request in 17)
19	22.022788719	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
20	22.950473687	172.16.40.1	172.16.1.10	ICMP	98	Echo (ping) request id=0x224c, seq=2/512, ttl=64 (reply in 21)
21	22.950916978	172.16.1.10	172.16.40.1	ICMP	98	Echo (ping) reply id=0x224c, seq=2/512, ttl=62 (request in 20)
22	23.974319883	172.16.40.1	172.16.1.10	ICMP	98	Echo (ping) request id=0x224c, seq=3/768, ttl=64 (reply in 23)
23	23.974752139	172.16.1.10	172.16.40.1	ICMP	98	Echo (ping) reply id=0x224c, seq=3/768, ttl=62 (request in 22)
24	24.024964945	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
25	24.998272310	172.16.40.1	172.16.1.10	ICMP	98	Echo (ping) request id=0x224c, seq=4/1024, ttl=64 (reply in 26)
26	24.998712529	172.16.1.10	172.16.40.1	ICMP	98	Echo (ping) reply id=0x224c, seq=4/1024, ttl=62 (request in 25)
27	26.022274954	172.16.40.1	172.16.1.10	ICMP	98	Echo (ping) request id=0x224c, seq=5/1280, ttl=64 (reply in 28)

Image 26 - Part 2 Exp 4 :tux2 ping tux3 (ICMP redirect disabled)

Return to 4.4

No.	Time	Source	Destination	Protocol	Length	Info
3	2.573729943	Netronix_b5:8c:8f	Broadcast	ARP	42	Who has 172.16.41.254? Tell 172.16.41.1
4	2.573868928	Routerboardc_eb:18…	Netronix_b5:8c:8f	ARP	60	172.16.41.254 is at 74:4d:28:eb:18:d0
5	2.573887017	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) request id=0x1ce5, seq=1/256, ttl=64 (reply in 6)
6	2.574272123	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) reply id=0x1ce5, seq=1/256, ttl=63 (request in 5)
7	3.577400909	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) request id=0x1ce5, seq=2/512, ttl=64 (reply in 9)
8	3.577558890	172.16.41.254	172.16.41.1	ICMP	126	Redirect (Redirect for host)
9	3.577769392	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) reply id=0x1ce5, seq=2/512, ttl=63 (request in 7)
10	4.004288231	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/74:4d:28:eb:18:d0 Cost = 10 Port = 0x8001
11	4.601410765	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) request id=0x1ce5, seq=3/768, ttl=64 (reply in 13)
12	4.601565743	172.16.41.254	172.16.41.1	ICMP	126	Redirect (Redirect for host)
13	4.601779039	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) reply id=0x1ce5, seq=3/768, ttl=63 (request in 11)
14	5.625408400	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) request id=0x1ce5, seq=4/1024, ttl=64 (reply in 16)
15	5.625561562	172.16.41.254	172.16.41.1	ICMP	126	Redirect (Redirect for host)
16	5.625767734	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) reply id=0x1ce5, seq=4/1024, ttl=63 (request in 14)
17	6.006410423	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/74:4d:28:eb:18:d0 Cost = 10 Port = 0x8001
18	6.649407433	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) request id=0x1ce5, seq=5/1280, ttl=64 (reply in 20)
19	6.649562760	172.16.41.254	172.16.41.1	ICMP	126	Redirect (Redirect for host)
20	6.649803923	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) reply id=0x1ce5, seq=5/1280, ttl=63 (request in 18)
21	7.673410797	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) request id=0x1ce5, seq=6/1536, ttl=64 (reply in 23)
22	7.673571502	172.16.41.254	172.16.41.1	ICMP	126	Redirect (Redirect for host)
23	7.673787522	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) reply id=0x1ce5, seq=6/1536, ttl=63 (request in 21)
24	7.776604603	3Com_a0:ad:91	Netronix_b5:8c:8f	ARP	60	Who has 172.16.41.1? Tell 172.16.41.253
25	7.776619828	Netronix_b5:8c:8f	3Com_a0:ad:91	ARP	42	172.16.41.1 is at 00:e0:7d:b5:8c:8f
26	8.008560625	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/74:4d:28:eb:18:d0 Cost = 10 Port = 0x8001
27	8.577486964	Routerboardc_eb:18…	Netronix_b5:8c:8f	ARP	60	Who has 172.16.41.1? Tell 172.16.41.254
28	8.577493878	Netronix_b5:8c:8f	Routerboardc_eb:18…	ARP	42	172.16.41.1 is at 00:e0:7d:b5:8c:8f
29	8.697388251	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) request id=0x1ce5, seq=7/1792, ttl=64 (reply in 30)
30	8.697720976	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) reply id=0x1ce5, seq=7/1792, ttl=63 (request in 29)

Image 27 - Part 2 Exp 4 :tux2 ping tux3 (ICMP redirect enabled)

Return to 4.4

No.	Time	Source	Destination	Protocol	Length	Info
3	3.255278304	Netronix_b5:8c:8f	Broadcast	ARP	42	Who has 172.16.41.254? Tell 172.16.41.1
4	3.255417149	Routerboardc_eb:18…	Netronix_b5:8c:8f	ARP	60	172.16.41.254 is at 74:4d:28:eb:18:d0
5	3.255437054	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) request id=0x1dcb, seq=1/256, ttl=64 (reply in 6)
6	3.255817899	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) reply id=0x1dcb, seq=1/256, ttl=63 (request in 5)
7	4.004365666	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/74:4d:28:eb:18:d0 Cost = 10 Port = 0x8001
8	4.274150339	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) request id=0x1dcb, seq=2/512, ttl=64 (reply in 12)
9	4.274303641	172.16.41.254	172.16.41.1	ICMP	126	Redirect (Redirect for host)
10	4.274337863	Netronix_b5:8c:8f	Broadcast	ARP	42	Who has 172.16.41.253? Tell 172.16.41.1
11	4.274452194	3Com_a0:ad:91	Netronix_b5:8c:8f	ARP	60	172.16.41.253 is at 00:01:02:a0:ad:91
12	4.274511629	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) reply id=0x1dcb, seq=2/512, ttl=63 (request in 8)
13	5.298158518	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) request id=0x1dcb, seq=3/768, ttl=64 (reply in 15)
14	5.298327884	172.16.41.254	172.16.41.1	ICMP	126	Redirect (Redirect for host)
15	5.298563669	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) reply id=0x1dcb, seq=3/768, ttl=63 (request in 13)
16	6.005662459	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/74:4d:28:eb:18:d0 Cost = 10 Port = 0x8001
17	6.322119206	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) request id=0x1dcb, seq=4/1024, ttl=64 (reply in 18)
18	6.322390261	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) reply id=0x1dcb, seq=4/1024, ttl=63 (request in 17)
19	7.346135000	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) request id=0x1dcb, seq=5/1280, ttl=64 (reply in 20)
20	7.346404030	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) reply id=0x1dcb, seq=5/1280, ttl=63 (request in 19)
21	8.007972446	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/74:4d:28:eb:18:d0 Cost = 10 Port = 0x8001
22	8.370137106	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) request id=0x1dcb, seq=6/1536, ttl=64 (reply in 23)
23	8.370385951	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) reply id=0x1dcb, seq=6/1536, ttl=63 (request in 22)
24	8.377322260	3Com_a0:ad:91	Netronix_b5:8c:8f	ARP	60	Who has 172.16.41.1? Tell 172.16.41.253
25	8.377329942	Netronix_b5:8c:8f	3Com_a0:ad:91	ARP	42	172.16.41.1 is at 00:e0:7d:b5:8c:8f
26	9.394144729	172.16.41.1	172.16.40.1	ICMP	98	Echo (ping) request id=0x1dcb, seq=7/1792, ttl=64 (reply in 27)
27	9.394413549	172.16.40.1	172.16.41.1	ICMP	98	Echo (ping) reply id=0x1dcb, seq=7/1792, ttl=63 (request in 26)
28	10.000200185	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/74:4d:28:eb:18:d0 Cost = 10 Port = 0x8001
29	10.296307842	Routerboardc_eb:18…	Netronix_b5:8c:8f	ARP	60	Who has 172.16.41.1? Tell 172.16.41.254

Image 28 - Part 2 Exp 4 : tux3 ping FTP server (Rc NAT enabled)

Return to 4.4

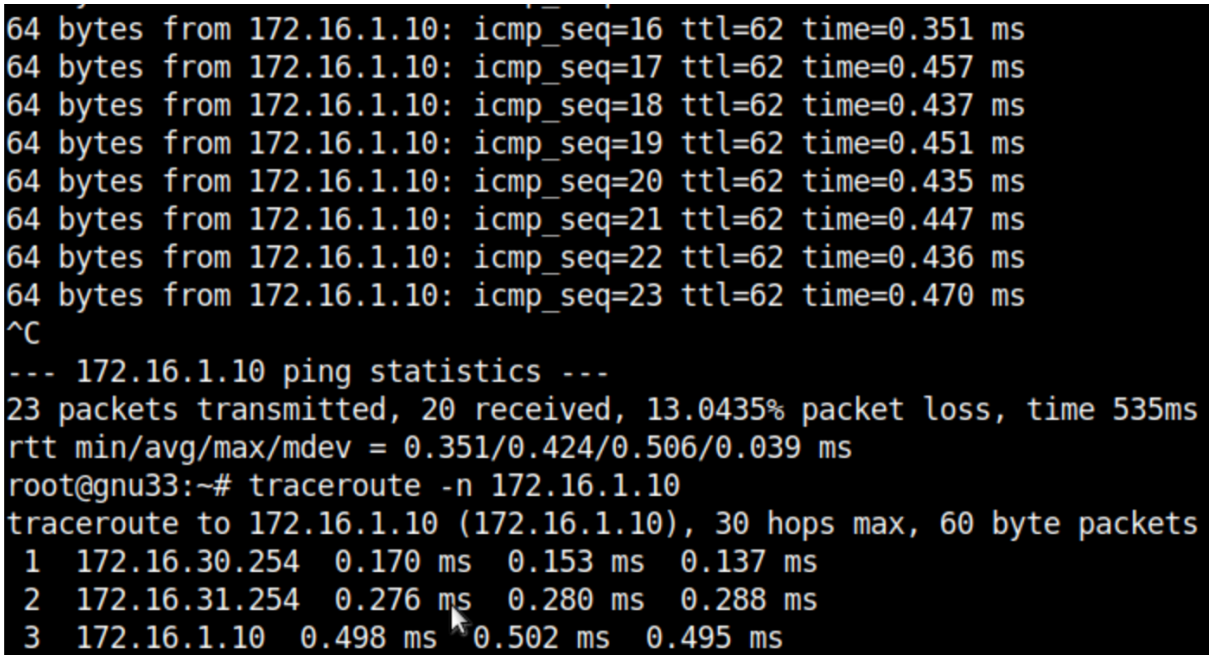


Image 29 - Part 2 Exp 4 : tux3 ping FTP server (Rc NAT disabled)

Return to 4.4

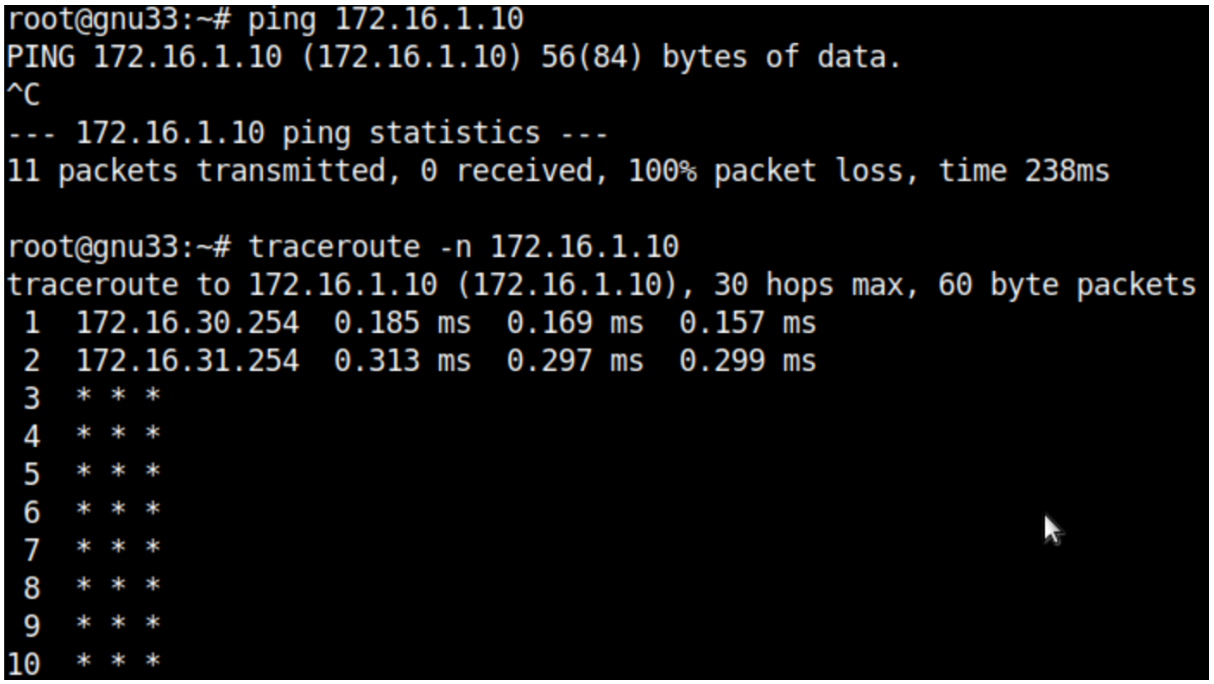
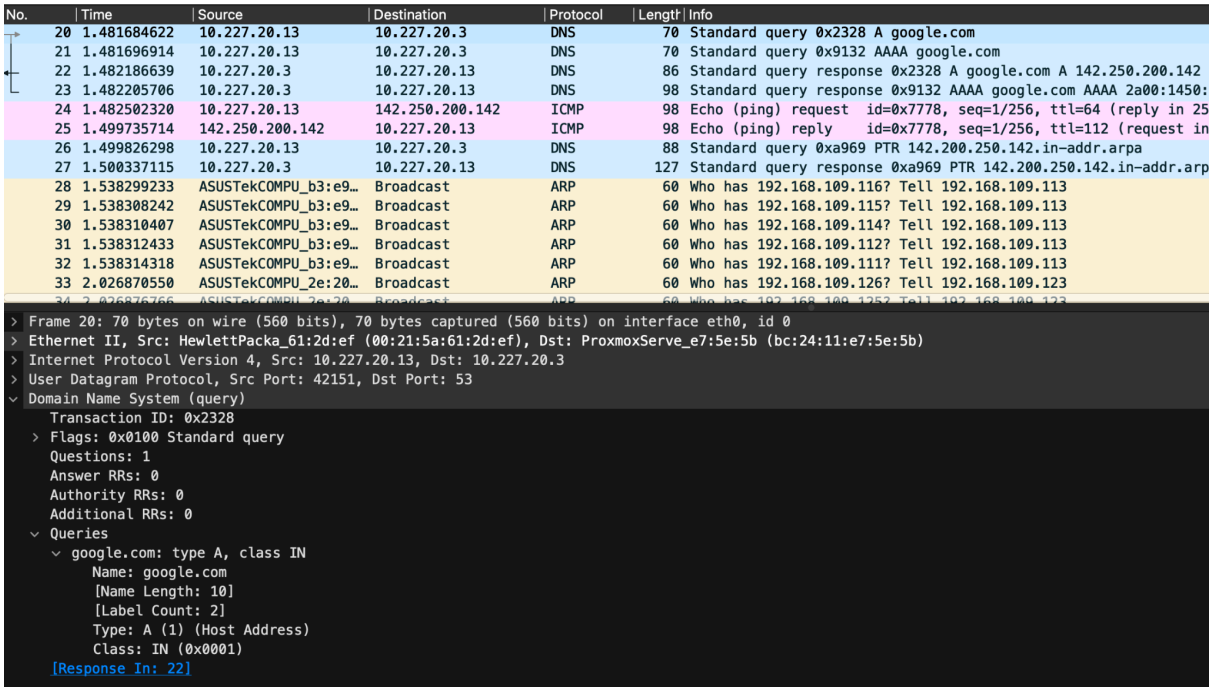


Image 30 - Part 2 Exp 5 : tux3 DNS query

Return to 5.4

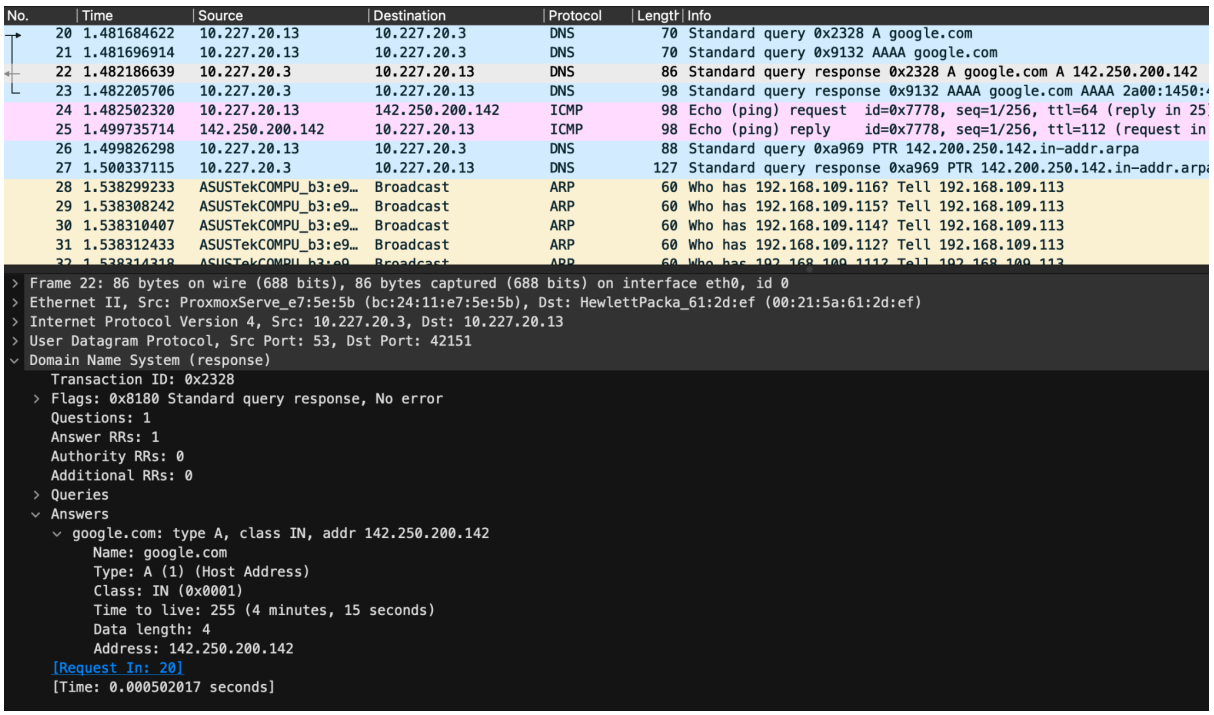


No.	Time	Source	Destination	Protocol	Length	Info
20	1.481684622	10.227.20.13	10.227.20.3	DNS	70	Standard query 0x2328 A google.com
21	1.481696914	10.227.20.13	10.227.20.3	DNS	70	Standard query 0x9132 AAAA google.com
22	1.482186639	10.227.20.3	10.227.20.13	DNS	86	Standard query response 0x2328 A google.com A 142.250.200.142
23	1.482205706	10.227.20.3	10.227.20.13	DNS	98	Standard query response 0x9132 AAAA google.com AAAA 2a00:1450:
24	1.482502320	10.227.20.13	142.250.200.142	ICMP	98	Echo (ping) request id=0x7778, seq=1/256, ttl=64 (reply in 25
25	1.499735714	142.250.200.142	10.227.20.13	ICMP	98	Echo (ping) reply id=0x7778, seq=1/256, ttl=112 (request in
26	1.499826298	10.227.20.13	10.227.20.3	DNS	88	Standard query 0xa969 PTR 142.200.250.142.in-addr.arpa
27	1.500337115	10.227.20.3	10.227.20.13	DNS	127	Standard query response 0xa969 PTR 142.200.250.142.in-addr.arp
28	1.538299233	ASUSTekCOMPU_b3:e9…	Broadcast	ARP	60	Who has 192.168.109.116? Tell 192.168.109.113
29	1.538308242	ASUSTekCOMPU_b3:e9…	Broadcast	ARP	60	Who has 192.168.109.115? Tell 192.168.109.113
30	1.538310407	ASUSTekCOMPU_b3:e9…	Broadcast	ARP	60	Who has 192.168.109.114? Tell 192.168.109.113
31	1.538312433	ASUSTekCOMPU_b3:e9…	Broadcast	ARP	60	Who has 192.168.109.112? Tell 192.168.109.113
32	1.538314318	ASUSTekCOMPU_b3:e9…	Broadcast	ARP	60	Who has 192.168.109.111? Tell 192.168.109.113
33	2.026870550	ASUSTekCOMPU_2e:20…	Broadcast	ARP	60	Who has 192.168.109.126? Tell 192.168.109.123

```
> Frame 20: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface eth0, id 0
> Ethernet II, Src: HewlettPacka_61:2d:ef (00:21:5a:61:2d:ef), Dst: ProxmoxServe_e7:5e:5b (bc:24:11:e7:5e:5b)
> Internet Protocol Version 4, Src: 10.227.20.13, Dst: 10.227.20.3
> User Datagram Protocol, Src Port: 42151, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0x2328
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v google.com: type A, class IN
        Name: google.com
        [Name Length: 10]
        [Label Count: 2]
        Type: A (1) (Host Address)
        Class: IN (0x0001)
    [Response In: 22]
```

Image 31 - Part 2 Exp 5 : DNS response

Return to 5.4



No.	Time	Source	Destination	Protocol	Length	Info
20	1.481684622	10.227.20.13	10.227.20.3	DNS	70	Standard query 0x2328 A google.com
21	1.481696914	10.227.20.13	10.227.20.3	DNS	70	Standard query 0x9132 AAAA google.com
22	1.482186639	10.227.20.3	10.227.20.13	DNS	86	Standard query response 0x2328 A google.com A 142.250.200.142
23	1.482205706	10.227.20.3	10.227.20.13	DNS	98	Standard query response 0x9132 AAAA google.com AAAA 2a00:1450:
24	1.482502320	10.227.20.13	142.250.200.142	ICMP	98	Echo (ping) request id=0x7778, seq=1/256, ttl=64 (reply in 25
25	1.499735714	142.250.200.142	10.227.20.13	ICMP	98	Echo (ping) reply id=0x7778, seq=1/256, ttl=112 (request in
26	1.499826298	10.227.20.13	10.227.20.3	DNS	88	Standard query 0xa969 PTR 142.200.250.142.in-addr.arpa
27	1.500337115	10.227.20.3	10.227.20.13	DNS	127	Standard query response 0xa969 PTR 142.200.250.142.in-addr.arp
28	1.538299233	ASUSTekCOMPU_b3:e9…	Broadcast	ARP	60	Who has 192.168.109.116? Tell 192.168.109.113
29	1.538308242	ASUSTekCOMPU_b3:e9…	Broadcast	ARP	60	Who has 192.168.109.115? Tell 192.168.109.113
30	1.538310407	ASUSTekCOMPU_b3:e9…	Broadcast	ARP	60	Who has 192.168.109.114? Tell 192.168.109.113
31	1.538312433	ASUSTekCOMPU_b3:e9…	Broadcast	ARP	60	Who has 192.168.109.112? Tell 192.168.109.113

```
> Frame 22: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface eth0, id 0
> Ethernet II, Src: ProxmoxServe_e7:5e:5b (bc:24:11:e7:5e:5b), Dst: HewlettPacka_61:2d:ef (00:21:5a:61:2d:ef)
> Internet Protocol Version 4, Src: 10.227.20.3, Dst: 10.227.20.13
> User Datagram Protocol, Src Port: 53, Dst Port: 42151
v Domain Name System (response)
    Transaction ID: 0x2328
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  > Queries
  v Answers
    v google.com: type A, class IN, addr 142.250.200.142
        Name: google.com
        Type: A (1) (Host Address)
        Class: IN (0x0001)
        Time to live: 255 (4 minutes, 15 seconds)
        Data length: 4
        Address: 142.250.200.142
    [Request In: 20]
    [Time: 0.000502017 seconds]
```

Image 32 - Part 2 Exp 6

Return to 6.1

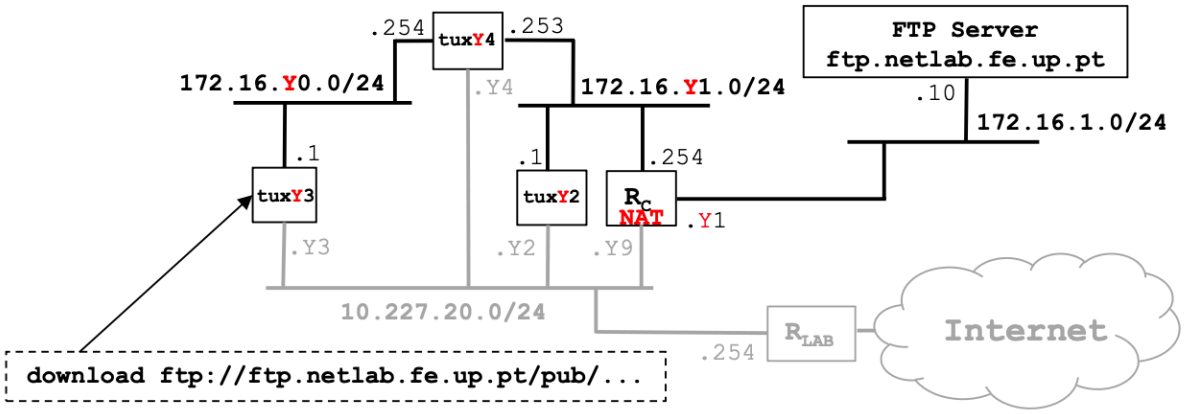


Image 33 - Part 2 Exp 6 : Two TCP connections are opened by Download Application

Return to 6.4

No.	Time	Source	Destination	Protocol	Length	Info
140	32.345514964	10.227.155.55	193.137.29.15	TCP	76	49566 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2366728846 TSecr=0 WS=128
141	32.351500807	193.137.29.15	10.227.155.55	TCP	76	21 → 49566 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1380 SACK_PERM TSval=3256368339 TSecr=2366728846 W
142	32.351559865	10.227.155.55	193.137.29.15	TCP	68	49566 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2366728852 TSecr=3256368339
143	32.358983695	193.137.29.15	10.227.155.55	FTP	141	Response: 220-Welcome to the University of Porto's mirror archive (mirrors.up.pt)
144	32.359012679	10.227.155.55	193.137.29.15	TCP	68	49566 → 21 [ACK] Seq=1 Ack=74 Win=64256 Len=0 TSval=2366728860 TSecr=3256368346
145	32.360039600	193.137.29.15	10.227.155.55	FTP	143	Response: 220----------
146	32.360065147	10.227.155.55	193.137.29.15	TCP	68	49566 → 21 [ACK] Seq=1 Ack=149 Win=64256 Len=0 TSval=2366728861 TSecr=3256368346
147	32.361108158	193.137.29.15	10.227.155.55	FTP	233	Response: 220-All connections and transfers are logged. The max number of connections is 200.
148	32.361130759	10.227.155.55	193.137.29.15	TCP	68	49566 → 21 [ACK] Seq=1 Ack=314 Win=64128 Len=0 TSval=2366728862 TSecr=3256368346
149	32.362188737	193.137.29.15	10.227.155.55	FTP	147	Response: 220-Questions and comments can be sent to mirrors@uporto.pt
150	32.362218772	10.227.155.55	193.137.29.15	TCP	68	49566 → 21 [ACK] Seq=1 Ack=393 Win=64128 Len=0 TSval=2366728863 TSecr=3256368346
151	32.362441583	10.227.155.55	193.137.29.15	FTP	84	Request: USER anonymous
152	32.367956358	193.137.29.15	10.227.155.55	TCP	68	21 → 49566 [ACK] Seq=393 Ack=17 Win=65280 Len=0 TSval=3256368355 TSecr=2366728863
153	32.368829255	193.137.29.15	10.227.155.55	FTP	102	Response: 331 Please specify the password.
154	32.368989199	10.227.155.55	193.137.29.15	FTP	84	Request: PASS anonymous
155	32.375643573	193.137.29.15	10.227.155.55	FTP	91	Response: 230 Login successful.
156	32.375764075	10.227.155.55	193.137.29.15	FTP	74	Request: PASV
157	32.381416062	193.137.29.15	10.227.155.55	FTP	119	Response: 227 Entering Passive Mode (193,137,29,15,227,38).
158	32.381633493	10.227.155.55	193.137.29.15	TCP	76	54624 → 58150 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2366728883 TSecr=0 WS=128
159	32.384567155	193.137.29.15	10.227.155.55	TCP	76	58150 → 54624 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1380 SACK_PERM TSval=3256368373 TSecr=236672888
160	32.384608551	10.227.155.55	193.137.29.15	TCP	68	54624 → 58150 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2366728886 TSecr=3256368373
161	32.384647723	10.227.155.55	193.137.29.15	FTP	115	Request: RETR pub/gnu/emacs/elisp-manual-21-2.8.tar.gz

Image 34 - Part 2 Exp 6 : Control Packet

Return to 6.4

No.	Time	Source	Destination	Protocol	Length	Info
146	32.360065147	10.227.155.55	193.137.29.15	TCP	68	49566 → 21 [ACK] Seq=1 Ack=149 Win=64256 Len=0 TSval=2366728861 TSecr=3256368346
147	32.361108158	193.137.29.15	10.227.155.55	FTP	233	Response: 220-All connections and transfers are logged. The max number of connections is 200.
148	32.361130759	10.227.155.55	193.137.29.15	TCP	68	49566 → 21 [ACK] Seq=1 Ack=314 Win=64128 Len=0 TSval=2366728862 TSecr=3256368346
149	32.362188737	193.137.29.15	10.227.155.55	FTP	147	Response: 220-Questions and comments can be sent to mirrors@uporto.pt
150	32.362218772	10.227.155.55	193.137.29.15	TCP	68	49566 → 21 [ACK] Seq=1 Ack=393 Win=64128 Len=0 TSval=2366728863 TSecr=3256368346
151	32.362441583	10.227.155.55	193.137.29.15	FTP	84	Request: USER anonymous
152	32.367956358	193.137.29.15	10.227.155.55	TCP	68	21 → 49566 [ACK] Seq=393 Ack=17 Win=65280 Len=0 TSval=3256368355 TSecr=2366728863
153	32.368829255	193.137.29.15	10.227.155.55	FTP	102	Response: 331 Please specify the password.
154	32.368989199	10.227.155.55	193.137.29.15	FTP	84	Request: PASS anonymous
155	32.375643573	193.137.29.15	10.227.155.55	FTP	91	Response: 230 Login successful.

```
> Frame 151: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 10.227.155.55, Dst: 193.137.29.15
v Transmission Control Protocol, Src Port: 49566, Dst Port: 21, Seq: 1, Ack: 393, Len: 16
    Source Port: 49566
    Destination Port: 21
    [Stream index: 29]
  > [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 16]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 2414124319
    [Next Sequence Number: 17    (relative sequence number)]
    Acknowledgment Number: 393    (relative ack number)
    Acknowledgment number (raw): 2062035566
    1000 .... = Header Length: 32 bytes (8)
```

Image 35 - Part 2 Exp 6 : Data Packet

Return to 6.4

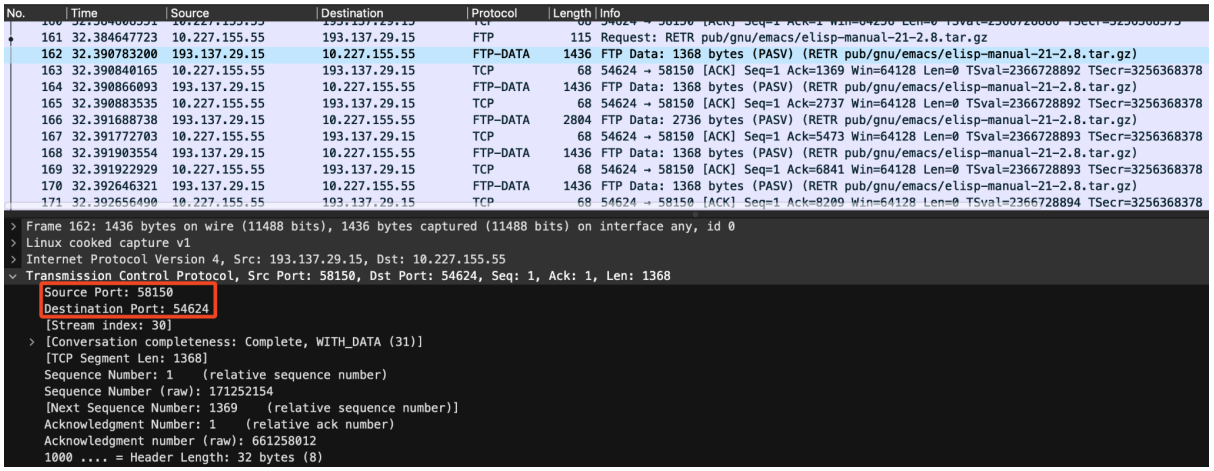


Image 36 - Part 2 Exp 6 : Close Connection

Return to 6.4

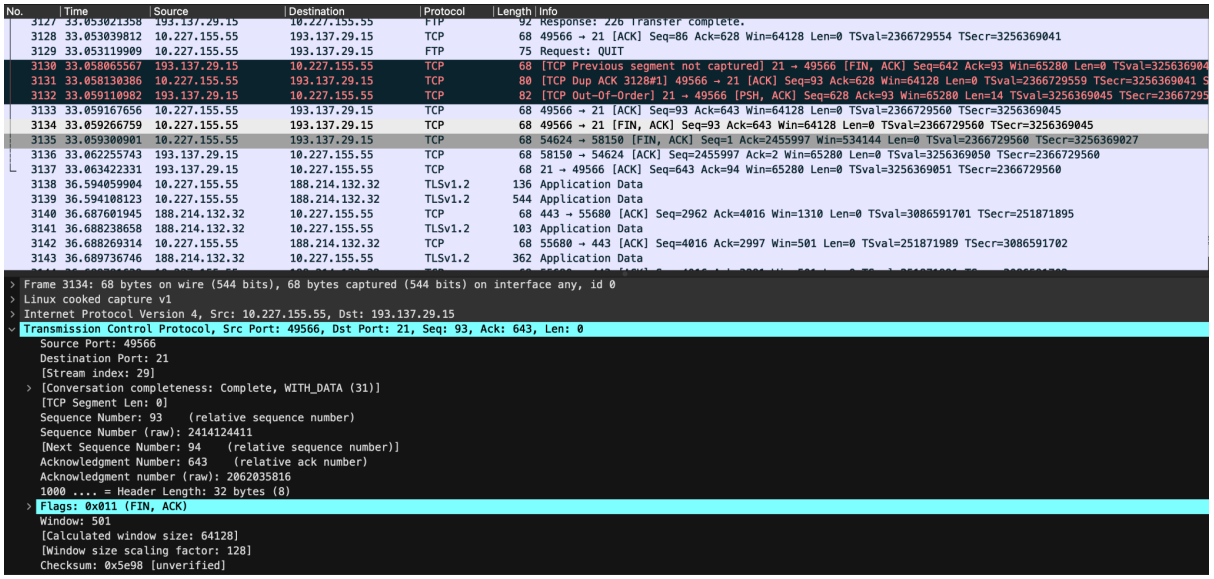
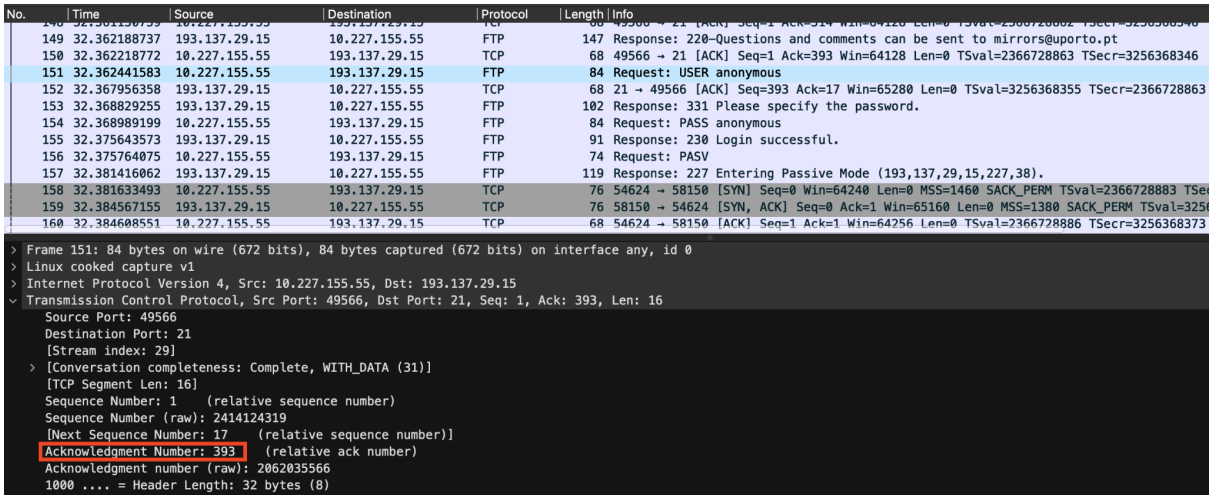


Image 37 - Part 2 Exp 6 : ARQ 1

Return to 6.4

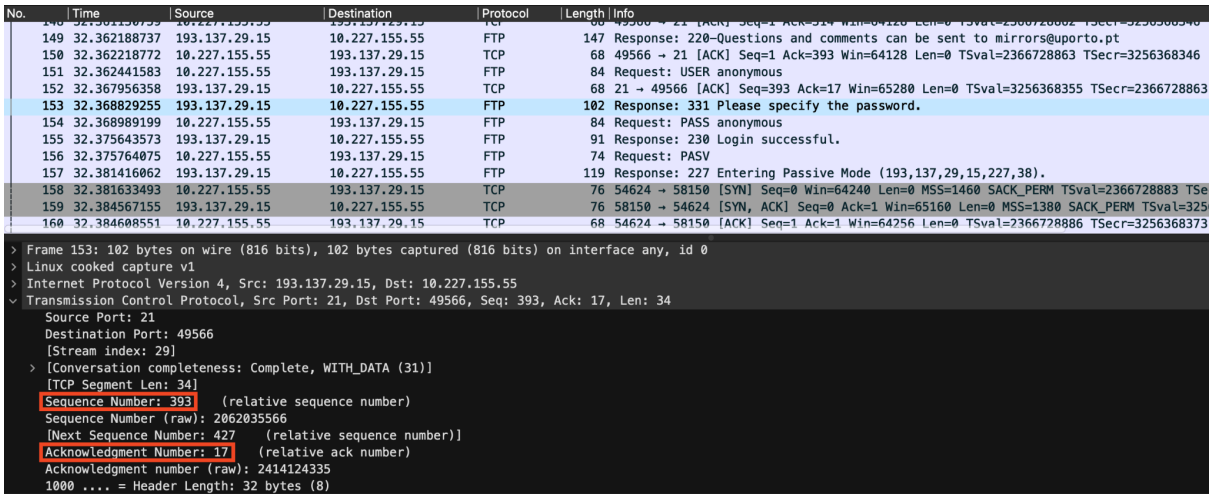


No.	Time	Source	Destination	Protocol	Length	Info
149	32.362188737	193.137.29.15	10.227.155.55	FTP	147	Response: 220-Questions and comments can be sent to mirrors@uporto.pt
150	32.362218772	10.227.155.55	193.137.29.15	TCP	68	49566 → 21 [ACK] Seq=1 Ack=393 Win=64128 Len=0 TSval=2366728863 TSecr=3256368346
151	32.362441583	10.227.155.55	193.137.29.15	FTP	84	Request: USER anonymous
152	32.367956358	193.137.29.15	10.227.155.55	TCP	68	21 → 49566 [ACK] Seq=393 Ack=17 Win=65280 Len=0 TSval=3256368355 TSecr=2366728863
153	32.368829255	193.137.29.15	10.227.155.55	FTP	102	Response: 331 Please specify the password.
154	32.368989199	10.227.155.55	193.137.29.15	FTP	84	Request: PASS anonymous
155	32.375643573	193.137.29.15	10.227.155.55	FTP	91	Response: 230 Login successful.
156	32.375764075	10.227.155.55	193.137.29.15	FTP	74	Request: PASV
157	32.381416062	193.137.29.15	10.227.155.55	FTP	119	Response: 227 Entering Passive Mode (193,137,29,15,227,38).
158	32.381633493	10.227.155.55	193.137.29.15	TCP	76	54624 → 58150 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2366728883 TSe
159	32.384567155	193.137.29.15	10.227.155.55	TCP	76	58150 → 54624 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1380 SACK_PERM TSval=325
160	32.384608551	10.227.155.55	193.137.29.15	TCP	68	54624 → 58150 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2366728886 TSecr=3256368373

```
> Frame 151: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 10.227.155.55, Dst: 193.137.29.15
v Transmission Control Protocol, Src Port: 49566, Dst Port: 21, Seq: 1, Ack: 393, Len: 16
    Source Port: 49566
    Destination Port: 21
    [Stream index: 29]
  > [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 16]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 2414124319
    [Next Sequence Number: 17    (relative sequence number)]
    Acknowledgment Number: 393    (relative ack number)
    Acknowledgment number (raw): 2062035566
    1000 .... = Header Length: 32 bytes (8)
```

Image 38 - Part 2 Exp 6 : ARQ 2

Return to 6.4



No.	Time	Source	Destination	Protocol	Length	Info
149	32.362188737	193.137.29.15	10.227.155.55	FTP	147	Response: 220-Questions and comments can be sent to mirrors@uporto.pt
150	32.362218772	10.227.155.55	193.137.29.15	TCP	68	49566 → 21 [ACK] Seq=1 Ack=393 Win=64128 Len=0 TSval=2366728863 TSecr=3256368346
151	32.362441583	10.227.155.55	193.137.29.15	FTP	84	Request: USER anonymous
152	32.367956358	193.137.29.15	10.227.155.55	TCP	68	21 → 49566 [ACK] Seq=393 Ack=17 Win=65280 Len=0 TSval=3256368355 TSecr=2366728863
153	32.368829255	193.137.29.15	10.227.155.55	FTP	102	Response: 331 Please specify the password.
154	32.368989199	10.227.155.55	193.137.29.15	FTP	84	Request: PASS anonymous
155	32.375643573	193.137.29.15	10.227.155.55	FTP	91	Response: 230 Login successful.
156	32.375764075	10.227.155.55	193.137.29.15	FTP	74	Request: PASV
157	32.381416062	193.137.29.15	10.227.155.55	FTP	119	Response: 227 Entering Passive Mode (193,137,29,15,227,38).
158	32.381633493	10.227.155.55	193.137.29.15	TCP	76	54624 → 58150 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2366728883 TSe
159	32.384567155	193.137.29.15	10.227.155.55	TCP	76	58150 → 54624 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1380 SACK_PERM TSval=325
160	32.384608551	10.227.155.55	193.137.29.15	TCP	68	54624 → 58150 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2366728886 TSecr=3256368373

```
> Frame 153: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 193.137.29.15, Dst: 10.227.155.55
v Transmission Control Protocol, Src Port: 21, Dst Port: 49566, Seq: 393, Ack: 17, Len: 34
    Source Port: 21
    Destination Port: 49566
    [Stream index: 29]
  > [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 34]
    Sequence Number: 393    (relative sequence number)
    Sequence Number (raw): 2062035566
    [Next Sequence Number: 427    (relative sequence number)]
    Acknowledgment Number: 17    (relative ack number)
    Acknowledgment number (raw): 2414124335
    1000 .... = Header Length: 32 bytes (8)
```

Image 39 - Part 2 Exp 6 : ARQ 3

Return to 6.4

No.	Time	Source	Destination	Protocol	Length	Info
147	32.361108158	193.137.29.15	10.227.155.55	FTP	233	Response: 220-All connections and transfers are logged. The max number of connect
148	32.361130759	10.227.155.55	193.137.29.15	TCP	68	49566 → 21 [ACK] Seq=1 Ack=314 Win=64128 Len=0 TSval=2366728862 TSecr=3256368346
149	32.362188737	193.137.29.15	10.227.155.55	FTP	147	Response: 220-Questions and comments can be sent to mirrors@uporto.pt
150	32.362218772	10.227.155.55	193.137.29.15	TCP	68	49566 → 21 [ACK] Seq=1 Ack=393 Win=64128 Len=0 TSval=2366728863 TSecr=3256368346
151	32.362441583	10.227.155.55	193.137.29.15	FTP	84	Request: USER anonymous
152	32.367956358	193.137.29.15	10.227.155.55	TCP	68	21 → 49566 [ACK] Seq=393 Ack=17 Win=65280 Len=0 TSval=3256368355 TSecr=2366728863
153	32.368829255	193.137.29.15	10.227.155.55	FTP	102	Response: 331 Please specify the password.
154	32.368989199	10.227.155.55	193.137.29.15	FTP	84	Request: PASS anonymous
155	32.375643573	193.137.29.15	10.227.155.55	FTP	91	Response: 230 Login successful.
156	32.375764075	10.227.155.55	193.137.29.15	FTP	74	Request: PASV
157	32.381416062	193.137.29.15	10.227.155.55	FTP	119	Response: 227 Entering Passive Mode (193,137,29,15,227,38).
158	32.381633493	10.227.155.55	193.137.29.15	TCP	76	54624 → 58150 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2366728883 TSe
159	32.384567155	193.137.29.15	10.227.155.55	TCP	76	58150 → 54624 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1380 SACK_PERM TSval=325
160	32.384608551	10.227.155.55	193.137.29.15	TCP	68	54624 → 58150 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2366728886 TSecr=3256368373

```
> Frame 154: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 10.227.155.55, Dst: 193.137.29.15
v Transmission Control Protocol, Src Port: 49566, Dst Port: 21, Seq: 17, Ack: 427, Len: 16
    Source Port: 49566
    Destination Port: 21
    [Stream index: 29]
  > [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 16]
    Sequence Number: 17    (relative sequence number)
    Sequence Number (raw): 2414124335
    [Next Sequence Number: 33    (relative sequence number)]
    Acknowledgment Number: 427    (relative ack number)
    Acknowledgment number (raw): 2062035600
    1000 .... = Header Length: 32 bytes (8)
```

Image 40 - Part 2 Exp 6 : Expected graph of only tux3 downloading

Return to 6.4

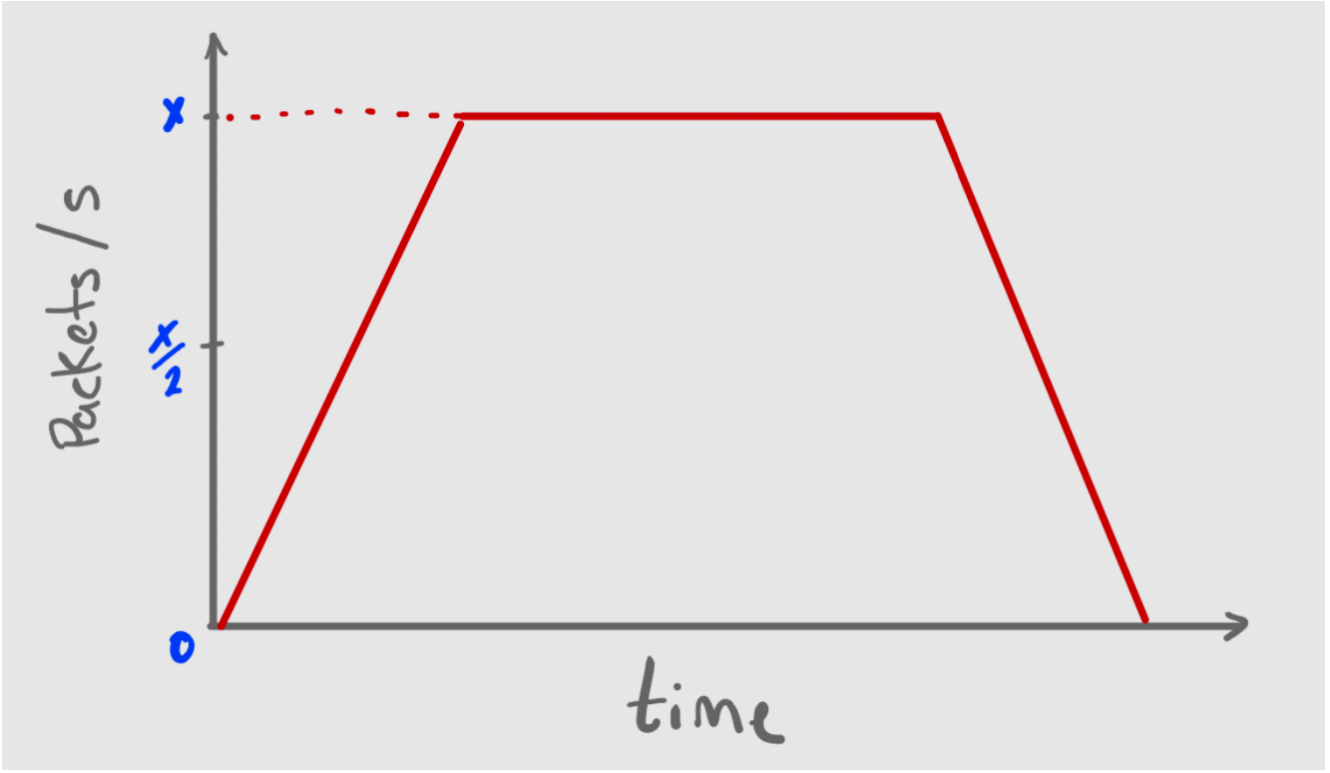
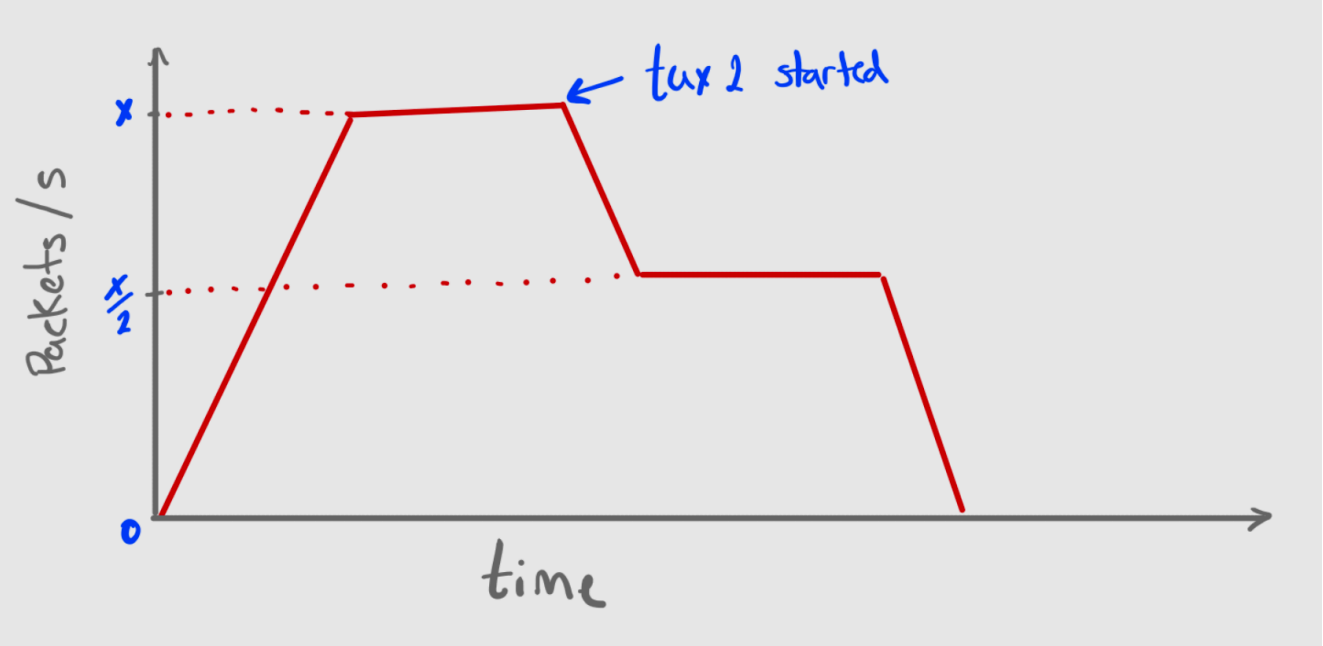


Image 41 - Part 2 Exp 6 : Expected graph of tux2 start a download in the middle of the tux3

⏪Return to 6.4



Download Application Directory structure

```
.
├── Makefile
├── bin
│   └── main
├── downloads
├── include
│   └── app_download.h
├── main.c
└── src
    └── app_download.c
```

Source Code (main.c)

```c
#include <stdio.h>
#include "app_download.h"

int main(int argc, char *argv[]) {
    if (argc != 2) {
        printf("Usage: %s <argument>\n", argv[0]);
        exit(-1);
    }
```

```
    const char *argument = argv[1];
    printf("Received argument: %s\n", argument);

    ConnectionSettings settings;
    if (parseURL(argument, &settings) != 0) {
        printf("[ERROR] Could not parse URL\n");
        printf("URL format should be:
ftp://[<user>:<password>@]<host>/<url-path>");
        exit(-1);
    }

    printf("Starting Download Application\n");
    printf("  - User: %s\n", settings.user);
    printf("  - Password: %s\n", settings.password);
    printf("  - Host: %s\n", settings.host);
    printf("  - Host Name: %s\n", settings.host_name);
    printf("  - IP Address: %s\n", settings.ip_address);
    printf("  - URL Path: %s\n", settings.url_path);
    printf("  - File Name: %s\n", settings.file_name);
    printf("\n");

    // Connect to FTP server
    int control_sockfd = openConnection(settings.ip_address, DEFAULT_PORT);
    if (control_sockfd < 0) {
        printf("[ERROR] Failed to connect to %s on port %d\n",
settings.ip_address, DEFAULT_PORT);
        exit(-1);
    }
    printf("Successfully connected to %s on port %d\n\n", settings.ip_address,
DEFAULT_PORT);

    // Perform FTP login
    if (connectFTP(control_sockfd, settings.user, settings.password) != 0) {
        printf("[ERROR] Failed to log in to FTP server\n");
        closeConnection(control_sockfd, -1);
        exit(-1);
    }
    //printf("Successfully logged in to FTP server\n");

    // Enter passive mode
    char data_ip[16];
    int data_port;
    if (enterPassiveMode(control_sockfd, data_ip, &data_port) != 0) {
        printf("[ERROR] Failed to enter passive mode\n");
```

```
        closeConnection(control_sockfd, -1);
        exit(-1);
    }
    //printf("Entered passive mode. Data connection IP: %s, Port: %d\n",
data_ip, data_port);

    // Connect to data connection
    int data_sockfd = openConnection(data_ip, data_port);
    if (data_sockfd < 0) {
        printf("[ERROR] Failed to connect to data connection\n");
        closeConnection(control_sockfd, -1);
        exit(-1);
    }
    //printf("Successfully connected to data connection\n");

    // Request resource from server
    if (requestResource(control_sockfd, settings.url_path) != 0) {
        printf("[ERROR] Failed to request resource from server\n");
        closeConnection(control_sockfd, data_sockfd);
        exit(-1);
    }
    //printf("Successfully requested resource from server\n");

    // Receive data from server and save to file
    //printf("Starting download...\n");
    if (receiveData(control_sockfd, data_sockfd, settings.file_name) != 0) {
        printf("[ERROR] Failed to receive data from server\n");
        closeConnection(control_sockfd, data_sockfd);
        exit(-1);
    }
    //printf("Successfully received data from server and saved to \"%s%s\"\n",
DOWNLOAD_PATH, settings.file_name);
    //printf("\nExiting...\n");

    // Close control and data connection
    if (closeConnection(control_sockfd, data_sockfd) != 0) {
        printf("[ERROR] Failed to close control/data connection\n");
        exit(-1);
    }

    //printf("\nGoodbye!\n");

    return 0;
}
```

app_download.h

```
#ifndef APP_DOWNLOAD_H
#define APP_DOWNLOAD_H

#include <stdio.h>
#include <stdlib.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <regex.h>

#define USER_PASS_HOST_REGEX "ftp://%[^:]:%[^@]@%[^/]/%s"
#define HOST_REGEX "ftp://%[^/]/%s"
#define PASSIVE_REGEX "Entering Passive Mode (%d,%d,%d,%d,%d,%d)"

#define DOWNLOAD_PATH "downloads/"

#define DEFAULT_USER "anonymous"
#define DEFAULT_PASSWORD "anonymous"
#define DEFAULT_PORT 21

#define FTP_RESPONSE_125 125    // Data connection already open; transfer
starting
#define FTP_RESPONSE_150 150    // File status OK, about to open data connection
#define FTP_RESPONSE_220 220    // Service ready for new user
#define FTP_RESPONSE_221 221    // Service closing control connection
#define FTP_RESPONSE_226 226    // Closing data connection, file transfer
successful
#define FTP_RESPONSE_227 227    // Entering Passive Mode
#define FTP_RESPONSE_230 230    // User logged in, proceed
#define FTP_RESPONSE_331 331    // User name OK, need password
#define FTP_RESPONSE_426 426    // Connection closed; transfer aborted

#define MAX_RESPONSE 1024
#define MAX_SIZE 256

#define TRUE 1
#define FALSE 0

typedef struct {
```

```
    char user[MAX_SIZE];
    char password[MAX_SIZE];
    char host[MAX_SIZE];
    char host_name[MAX_SIZE];
    char url_path[MAX_SIZE];
    char file_name[MAX_SIZE];
    char ip_address[16];
} ConnectionSettings;

typedef enum {
    READING_CODE,       // state of receive code
    READING_MESSAGE,    // state of receive message
    MULTILINE_RESPONSE, // state of receive multiline response
    DONE
} State;

// Function to parse URL and extract connection settings
// URL format: ftp://[<user>:<password>@]<host>/<url-path>
int parseURL(const char *url, ConnectionSettings *settings);

// Function to open a connection to the specified IP address and port
int openConnection(const char* ip_address, const int port);

// Function to read response from server
int readResponse(const int socket, char *buffer, int *code);

// Function to connect to FTP server
int connectFTP(const int socket, const char *user, const char *password);

// Function to enter passive mode
int enterPassiveMode(const int socket, char *ip_address, int *port);

// Function to request a resource from the server
int requestResource(const int socket, const char *resource);

// Function to receive data from the server and save it to a file
int receiveData(const int control_socket, const int data_socket, const char
*file_name);

// Function to close the connection
int closeConnection(const int socketControl, const int socketData);

#endif
```

app_download.c

```
#include "app_download.h"

int parseURL(const char *url, ConnectionSettings *settings) {
    regex_t regex;
    int res;

    // Compile regex to find '@' in the URL
    res = regcomp(&regex, "@", 0);
    if (res) {
        printf("[ERROR] Could not compile regex\n");
        return -1;
    }

    // Execute regex to check if '@' is present in the URL
    res = regexec(&regex, url, 0, NULL, 0);
    regfree(&regex);

    if (!res) {
        // '@' found in the URL, extract user, password, host, and path
        if (sscanf(url, USER_PASS_HOST_REGEX, settings->user,
settings->password, settings->host, settings->url_path) != 4) {
            printf("[ERROR] Could not parse URL with user and password\n");
            return -1;
        }
    } else {
        // '@' not found, extract host and path
        if (sscanf(url, HOST_REGEX, settings->host, settings->url_path) != 2) {
            printf("[ERROR] Could not parse URL without user and password\n");
            return -1;
        }
        strcpy(settings->user, DEFAULT_USER);
        strcpy(settings->password, DEFAULT_PASSWORD);
    }

    // Extract file name from URL path
    char *file_name = strrchr(settings->url_path, '/');
    if (file_name) {
        strcpy(settings->file_name, file_name + 1);
    } else {
        strcpy(settings->file_name, settings->url_path);
    }

    // Get IP address from host
```

```
    struct hostent *h;
    if (strlen(settings->host) > 0) {
        if ((h = gethostbyname(settings->host)) == NULL) {
            herror("gethostbyname()");
            return -1;
        }
        strcpy(settings->host_name, h->h_name);
        strcpy(settings->ip_address, inet_ntoa(*((struct in_addr *)
h->h_addr)));
    } else {
        return -1;
    }

    return 0;
}

int openConnection(const char* ip_address, const int port) {
    int sockfd;
    struct sockaddr_in serv_addr;

    // Create socket
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) {
        perror("ERROR opening socket");
        return -1;
    }

    // Set up server address
    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = inet_addr(ip_address);
    serv_addr.sin_port = htons(port);

    // Connect to server
    if (connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
        perror("ERROR connecting");
        close(sockfd);
        return -1;
    }

    return sockfd;
}

int readResponse(const int socket, char *buffer, int *code) {
```

```
    if (buffer == NULL || code == NULL) {
        printf("[ERROR] Invalid buffer or code\n");
        return -1;
    }

    State state = READING_CODE;
    *code = 0;
    int index = 0;
    ssize_t read_bytes;
    int prev_code = 0;

    while (state != DONE) {
        char byte = 0;
        read_bytes = read(socket, &byte, 1);
        if (read_bytes < 0) {
            perror("[ERROR] Could not read byte from socket");
            return -1;
        } else if (read_bytes == 0) {
            printf("[INFO] Server closed the connection\n");
            return -1;
        }

        if (index >= MAX_RESPONSE - 1) {
            printf("[ERROR] Buffer overflow detected\n");
            return -1;
        }

        switch (state) {
            case READING_CODE:
                if (byte >= '0' && byte <= '9') {
                    *code = *code * 10 + (byte - '0');
                } else if (byte == ' ') {
                    state = READING_MESSAGE;
                } else if (byte == '-') {
                    state = MULTILINE_RESPONSE;
                }
                break;
            case READING_MESSAGE:
                if (byte == '\n') {
                    buffer[index] = '\0';
                    state = DONE;
                } else {
                    buffer[index++] = byte;
                }
```

```
                break;
            case MULTILINE_RESPONSE:
                if (byte == '\n') {
                    state = READING_CODE;
                    if (prev_code != 0 && prev_code != *code) {
                        printf("[ERROR] Invalid multiline response\n");
                        return -1;
                    }
                    prev_code = *code;
                    *code = 0;
                }
                buffer[index++] = byte;
                break;
            default:
                printf("[ERROR] Invalid state\n");
                return -1;
        }
    }

    buffer[index] = '\0';
    printf("[%d] %s\n", *code, buffer);
    return 0;
}

int connectFTP(const int socket, const char *user, const char *password) {
    char *buffer = (char *)malloc(MAX_RESPONSE);
    int response = 0;

    do {
        if (readResponse(socket, buffer, &response) != 0) {
            printf("[ERROR] Could not read response from server\n");
            free(buffer);
            buffer = NULL;
            return -1;
        }
    } while (response != FTP_RESPONSE_220);

    // Send USER command
    sprintf(buffer, "USER %s\r\n", user);
    if (write(socket, buffer, strlen(buffer)) < 0) {
        printf("[ERROR] Could not write to socket\n");
        free(buffer);
        buffer = NULL;
        return -1;
```

```
    }

    // Read response from server
    if (readResponse(socket, buffer, &response) != 0) {
        printf("[ERROR] Could not read response from server\n");
        free(buffer);
        buffer = NULL;
        return -1;
    }

    // Check for successful USER command response (code 331)
    // User name OK, need password
    if (response != FTP_RESPONSE_331) {
        printf("[ERROR] Expected response code %d, received %d\n",
FTP_RESPONSE_331, response);
        free(buffer);
        buffer = NULL;
        return -1;
    }

    // Send PASS command
    sprintf(buffer, "PASS %s\r\n", password);
    if (write(socket, buffer, strlen(buffer)) < 0) {
        printf("[ERROR] Could not write to socket\n");
        free(buffer);
        buffer = NULL;
        return -1;
    }

    // Read response from server
    if (readResponse(socket, buffer, &response) != 0) {
        printf("[ERROR] Could not read response from server\n");
        free(buffer);
        buffer = NULL;
        return -1;
    }

    // Check for successful PASS command response (code 230)
    // User logged in, proceed
    if (response != FTP_RESPONSE_230) {
        printf("[ERROR] Expected response code %d, received %d\n",
FTP_RESPONSE_230, response);
        free(buffer);
        buffer = NULL;
```

```
        return -1;
    }

    free(buffer);
    buffer = NULL;
    return 0;
}

int enterPassiveMode(const int socket, char *ip_address, int *port) {
    char *buffer = (char *)malloc(MAX_RESPONSE);
    int response = 0;

    if (send(socket, "PASV\r\n", 6, 0) < 0) {
        perror("[ERROR] Could not send PASV command to server");
        free(buffer);
        buffer = NULL;
        return -1;
    }

    if (readResponse(socket, buffer, &response) != 0) {
        printf("[ERROR] Could not read response from server\n");
        free(buffer);
        buffer = NULL;
        return -1;
    }

    // Entering Passive Mode
    if (response != FTP_RESPONSE_227) {
        printf("[ERROR] Expected response code 227, received %d\n", response);
        free(buffer);
        buffer = NULL;
        return -1;
    }

    // Parse IP address and port from PASV response
    int ip1, ip2, ip3, ip4, p1, p2;
    if (sscanf(buffer, PASSIVE_REGEX, &ip1, &ip2, &ip3, &ip4, &p1, &p2) != 6) {
        printf("[ERROR] Could not parse PASV response\n");
        free(buffer);
        buffer = NULL;
        return -1;
    }

    sprintf(ip_address, "%d.%d.%d.%d", ip1, ip2, ip3, ip4);
```

```
    *port = p1 * 256 + p2;

    free(buffer);
    buffer = NULL;
    return 0;
}

int requestResource(const int socket, const char *resource) {
    char *buffer = (char *)malloc(MAX_RESPONSE);
    int response = 0;

    sprintf(buffer, "RETR %s\r\n", resource);
    if (write(socket, buffer, strlen(buffer)) < 0) {
        perror("[ERROR] Could not write to socket");
        free(buffer);
        buffer = NULL;
        return -1;
    }

    if (readResponse(socket, buffer, &response) != 0) {
        printf("[ERROR] Could not read response from server\n");
        free(buffer);
        buffer = NULL;
        return -1;
    }

    // File status OK, about to open data connection
    if (response != FTP_RESPONSE_150 && response != FTP_RESPONSE_125) {
        printf("[ERROR] Expected response code %d, received %d\n",
FTP_RESPONSE_150, response);
        free(buffer);
        buffer = NULL;
        return -1;
    }

    free(buffer);
    buffer = NULL;
    return 0;
}

int receiveData(const int control_socket, const int data_socket, const char
*file_name) {
    char file_path[MAX_SIZE];
    snprintf(file_path, sizeof(file_path), "%s%s", DOWNLOAD_PATH, file_name);
```

```
    FILE *file = fopen(file_path, "wb");
    if (file == NULL) {
        perror("[ERROR] Could not open file for writing");
        return -1;
    }

    char *buffer = (char *)malloc(MAX_RESPONSE);
    ssize_t read_bytes;
    size_t total_bytes = 0;

    while ((read_bytes = read(data_socket, buffer, MAX_RESPONSE)) > 0) {
        if (fwrite(buffer, 1, read_bytes, file) != read_bytes) {
            perror("[ERROR] Could not write to file");
            free(buffer);
            fclose(file);
            return -1;
        }
        total_bytes += read_bytes;
        //printf("[DEBUG] Wrote %zd bytes to file (total: %zu)\n", read_bytes,
total_bytes);
    }

    if (read_bytes < 0) {
        perror("[ERROR] Could not read from socket");
        free(buffer);
        fclose(file);
        return -1;
    }

    fclose(file);
    printf("[INFO] File transfer completed, total bytes: %zu\n", total_bytes);

    memset(buffer, 0, MAX_RESPONSE);

    int response = 0;
    if (readResponse(control_socket, buffer, &response) != 0) {
        printf("[ERROR] Could not read response from server\n");
        free(buffer);
        buffer = NULL;
        return -1;
    }

    if (response != FTP_RESPONSE_226) {
```

```
        printf("[ERROR] Expected response code %d, received %d\n",
FTP_RESPONSE_226, response);
        free(buffer);
        buffer = NULL;
        return -1;
    }

    free(buffer);
    buffer = NULL;
    return 0;
}

int closeConnection(const int socketControl, const int socketData) {
    char *buffer = (char *)malloc(MAX_RESPONSE);
    int response = 0;

    sprintf(buffer, "QUIT \r\n");
    if (write(socketControl, buffer, strlen(buffer)) < 0) {
        perror("[ERROR] Could not write to socket");
        free(buffer);
        buffer = NULL;
        return -1;
    }

    if (readResponse(socketControl, buffer, &response) != 0) {
        printf("[ERROR] Could not read response from server\n");
        free(buffer);
        buffer = NULL;
        return -1;
    }

    if (response != FTP_RESPONSE_221) {
        printf("[ERROR] Expected response code %d, received %d\n",
FTP_RESPONSE_221, response);
        free(buffer);
        buffer = NULL;
        return -1;
    }

    if (close(socketControl) < 0) {
        perror("[ERROR] Could not close control socket");
        free(buffer);
        buffer = NULL;
        return -1;
```

```
    }

    if (socketData != -1 && close(socketData) < 0) {
        perror("[ERROR] Could not close data socket");
        free(buffer);
        buffer = NULL;
        return -1;
    }

    return 0;
}
```

Makefile

```
CC = gcc
CFLAGS = -Wall

SRC = src/
INCLUDE = include/
BIN = bin/

URL= ftp://ftp.up.pt/pub/gnu/emacs/elisp-manual-21-2.8.tar.gz
#URL= ftp://demo:password@test.rebex.net/readme.txt
#URL= ftp://anonymous:anonymous@ftp.bit.nl/speedtest/100mb.bin

#URL= ftp://netlab1.fe.up.pt/pub.txt
#URL= ftp://rcom:rcom@netlab1.fe.up.pt/pipe.txt
#URL= ftp://rcom:rcom@netlab1.fe.up.pt/files/pic1.jpg
#URL= ftp://rcom:rcom@netlab1.fe.up.pt/files/pic2.png
#URL= ftp://rcom:rcom@netlab1.fe.up.pt/files/crab.mp4
#URL= ftp://rcom:rcom@netlab1.fe.up.pt/debian/ls-lR.gz
#URL=
ftp://rcom:rcom@netlab1.fe.up.pt/pub/parrot/iso/testing/Parrot-architect-5.3_amd
64.iso
#URL= ftp://ftp.up.pt/pub/kodi/timestamp.txt

#URL= ftp://ftp.bit.nl/pub/welcome.msg

.PHONY: clean all run

all: $(BIN)/main

$(BIN)/main: main.c $(SRC)/*.c
	$(CC) $(CFLAGS) -o $@ $^ -I$(INCLUDE)
```

```
run: $(BIN)/main
	./$(BIN)/main $(URL)

clean:
	rm -f $(BIN)/*
```