Computer Network Configuration
RCOM Lab 2 - Report

Turma 7

- Rui Alexandre Costa Andrade - up202306128@fe.up.pt
- Tiago Miguel Veiga Monteiro - up202305179@fe.up.pt

Summary

This lab was conducted to design and implement a complete file download system. The project had three primary goals: to develop a download application, to configure a functional computer network enabling file transfer from a server and to analyze the entire data exchange process.

The work was structured into two main phases. First, the download application's architecture was created and tested, with a successful file transfer documented using Wireshark to capture the underlying FTP protocol exchange. Second, a series of six targeted network experiments were configured and executed. Each experiment involved setting up specific network architectures, applying configuration commands, and analyzing packet logs to explore core networking concepts like TCP behavior and congestion control.

The project successfully achieved its objectives. A working application and network were built, facilitating a practical file transfer. Most importantly, through hands-on configuration and detailed analysis of network traffic, the lab provided direct insight into the fundamental mechanisms of data exchange, flow control, and protocol interaction, thereby reinforcing key theoretical concepts with empirical evidence.

Introduction

This report explores the process of developing a file download application using the FTP and TCP/IP protocols, together with the design and implementation of a computer network where the application is tested. The main objectives of this project are to understand the correct use of the FTP protocol for file transfer, to study the functionality of network devices such as switches and routers, and to learn how to configure a simple computer network. By combining application development with network setup, this project provides practical insight into data communication and network operation.

Part 1 – Download application

The download application is a command-line FTP file downloader application written in C that retrieves files from FTP servers through a sequence of FTP protocol commands. The application follows a modular, step-by-step architecture designed to parse URLs, resolve hostnames, establish network connections and download files. The main modules of the application are the Main Controller, URL Parser, DNS Resolver, FTP Client and File Creator.

The Main controller (defined in download.c) is responsible for orchestrating the entire download process and coordinating the interaction between all modules.

The URL Parser (defined in parser.c and parser.h) parses FTP URLs into structured components. It validates the URL format and protocol, extracts the protocol, username, password, host, and file path, and defaults to anonymous authentication when no user credentials are provided.

The DNS Resolver (defined in getip.c and getip.h) resolves hostnames into IP addresses and populates the corresponding IP field in the URL structure.

The FTP Client (defined in clientTCP.c and clientTCP.h) manages all FTP protocol communication by using two sockets: one for the control connection (termA) and another for the data connection (termB). Initially, termA establishes the control connection and authenticates the user. Subsequently, termB sets up a passive-mode data connection. The

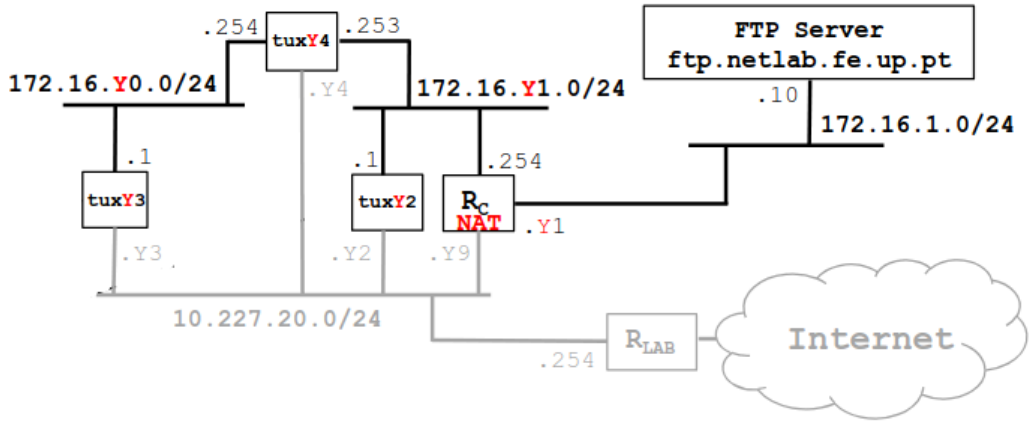
control socket then issues the RETR command to request the file specified in the URL path, after which termB receives the file content through the data connection.

Finally, the File Creator module (defined in create_file.c and create_file.h) is responsible for storing the downloaded content in the local file system.

Our application was successful at retrieving and downloading a file from a FTP server, as is shown below on the logs **A. Successful Download**, specifically the file present in **/put/teste.txt**, from the netlab server. The FTP Protocol proves this with the “Response: Transfer Complete.”.

Part 2 – Network configuration and analysis

In this part of the lab, we will configure a computer network over desk 5 in order for the download application to be able to download files via the ftp.netlab.fe.up.pt FTP Server. By the end of all experiments, the network will resemble something like this:



The **Y,** in our case, stands for 5. We followed the exact protocol and steps detailed in the guide.

Exp 1 - Configure an IP Network

This experiment established a direct link between two hosts, tux53 and tux54, by connecting them to a switch and configuring their network interfaces. After verifying connectivity with a ping command and recording the initial state of routing and ARP tables, the ARP cache on tux53 was deliberately cleared. A Wireshark capture was then initiated on tux53 to record all network traffic while a series of ICMP echo requests (ping) were sent to tux54, capturing the complete process of ARP resolution followed by the ICMP exchange.

Exp 1 - Questions/Logs

The ARP (Address Resolution Protocol) is used to map an IP address to a physical MAC address on a local network, allowing devices to interact with each other. Therefore, the ARP packets allow a device to discover the MAC address associated with a known IP address.

When device A tries to send a packet to device B but it doesn’t have its MAC mapped to the IP, device A will send an ARP request via broadcast MAC address (FF:FF:FF:FF:FF:FF) to all devices on the local network, specifying the target IP address to

find the MAC address of device B. Device B recognizes its own IP address as the target, therefore creating an ARP packet containing its MAC address and sending it directly to device A, using the MAC address of device A provided in the original request. Device A receives the reply and then maps the MAC of device B to the IP of device B, being able to properly communicate with it.

The ping command first generates ARP packets (the message via Broadcast and the reply) and then the ICMP (Internet Control Message Protocol) packets.

Through this experiment, we can see through the log **B.Experiment 1** the MAC and IP addresses of ping packets.

Wireshark allows its user to identify the type of data packets sent, either by displaying its name or a colour, and also the length of a receiving frame.

Wireshark also allows its user to identify the length of a receiving frame in the "Length" column. Although ARP is a fixed-length protocol, the IP has a Total Length field in its header and ICMP's length depends on the IP's Total Length field.

The loopback interface is a visual interface that allows a computer to communicate with itself. It's important since it allows the user to test network configurations.

Exp 2 - Implement two bridges in a switch

This experiment established a controlled network topology by creating and assigning custom bridges (bridge50 and bridge51) on a switch, moving the connections for hosts tux52, tux53, and tux54 from the default bridge to these isolated segments through GTK.

After initially recording the interface state of tux52, connectivity was evaluated by sending ICMP echo requests from tux53 to tux54 and to tux52 while capturing traffic on tux53 (C. Exp 2 Ping from tux53 to tux54 and D. Exp 2 Ping from tux53 to tux52). Subsequently, simultaneous packet captures were started on all three hosts to analyze broadcast behavior, first during a broadcast ping from tux53 to the 172.16.50.255 subnet (E. Exp 2 Ping Broadcast from tux53, capture at tux52; F. Exp 2 Ping Broadcast from tux53, capture at tux53; G. Exp 2 Ping Broadcast from tux53, capture at tux54) and then repeated during a broadcast ping from tux52 to the 172.16.51.255 subnet, with all traffic logged for comparative analysis (H. Exp 2 Ping Broadcast from tux52, capture at tux52;I. Exp 2 Ping Broadcast from tux52, capture at tux53; J. Exp 2 Ping Broadcast from tux52, capture at tux54).

Exp 2 - Questions

The following commands allow the configuration of bridge50 by just inserting them into the Mikrotik terminal:

- /interface bridge add name=bridge50
- /interface bridge port remove [find interface=ether3]
- /interface bridge port remove [find interface=ether4]
- /interface bridge port add bridge=bridge50 interface=ether3
- /interface bridge port add bridge=bridge50 interface=ether4

Through Logs **E through J**, we can verify that there are two broadcast domains: 172.16.50.0/24 and 172.16.51.0/24, since tux53 can broadcast to itself and to tux54, but it does not send any ICMP packages to tux52 and tux52 can only broadcast to itself.

Exp 3 - Configure a Router in Linux

This experiment configured host tux54 as a functional router by setting up its second network interface, adding it to bridge51, enabling IP forwarding, and adjusting broadcast handling, with its interface addresses recorded. After reconfiguring tux53 and tux52 for routed connectivity, their routing tables were inspected. Packet capture on tux53 monitored ICMP echo requests sent to key IP addresses across the subnets to verify routing functionality. Finally, a fresh address resolution test was conducted by clearing all ARP caches and initiating a sustained ICMP exchange from tux53 to tux52 while capturing traffic on both interfaces of the tux54 router, with all logs saved for analysis.

Exp 3 - Questions

Tux53 has one route on top of the ones it uses for the loopback interface, network 172.16.51.0/24 via gateway: 172.16.50.254.. Tux54 has only its loopback interface routes. Tux52 has one route on top of the ones it uses for the loopback interface, network 172.16.50.0/24 via gateway 172.16.51.253. Routes indicate to an interface how to reach a network. They are routed to the correct interface whenever they want to access any other of that specific network.

An entry of the forwarding table contains a route with the following attributes: Destination (network), Gateway, Genmask, Flags, Metric, Ref, Use and Interface;

Through logs **M and N**, it is visible to see the ARP messages and their associated MAC addresses through the interface if_e1 and if_e2 of tux54, respectively. The ARP messages show that tuxY3 successfully resolved its gateway's MAC, but the gateway could not reach tux52. No ARP reply came from tux52, so the gateway sent ICMP Host Unreachable messages back, causing the ping to fail.

Through the observation of log **N**, it is verified that ICMP Echo Requests and Destination Unreachable packets are observed in this experiment. This happens because tux53 sends ping requests to tux52 through its gateway, but the gateway cannot reach tux52.

The IP and MAC addresses associated with ICMP packets in this experiment are:

Packet Type	Source IP	Destination IP	Source MAC	Destination MAC	Why
ICMP Echo Request	172.16.50.1	172.16.51.1	TPLink_c2:51:4b	TPLink_c2:10:59	tuxY3 sending ping to tuxY2 via gateway (ARP resolved gateway MAC).
ICMP Dest. Unreachable	172.16.50.254	172.16.50.1	TPLink_c2:10:59	TPLink_c2:51:4b	Gateway informing tuxY3 that tuxY2 is unreachable.

Exp 4 - Configure a Commercial Router and Implement NAT

This experiment integrated router RC into the existing network, connecting one interface to the lab network with default NAT and another to bridge51, and configuring its IP addresses via console. Static routes were added across all hosts (tux53, tux54, tux52) and RC to ensure full subnet reachability, which was verified using ICMP and packet captures.

The experiment then explored routing behavior by first disabling ICMP redirects on tux52 (**Log O**) and changing its gateway for the 172.16.50.0/24 subnet to RC (**Log P**), analyzing packet paths and performing traceroutes. This configuration was reverted to use tux54 as the gateway, and traceroute was repeated before re-enabling redirects under specific conditions to observe the resulting behavior (**Log Q**). Finally, connectivity to an external FTP server was tested, and the impact of disabling NAT on router RC was analyzed through subsequent pings (**Logs R and S**).

Exp 4 - Questions

Using GTKTerminal, connected to the commercial router, we use the command "/ip route add" to add a static route specifying the destination network and the IP address so that traffic is forwarded correctly between networks.

Observing logs **O through Q**, it's visible the paths followed by the packets, with and without ICMP redirect enabled, in the experiments carried out on this experiment.

Those were:

- Without ICMP Redirects (using RC as gateway):
  - tux52 → RC → tux54 → tux53 → tux54 → tux52
  - tux52 uses RC as its default gateway, so all traffic to 172.16.50.1 goes via RC. RC knows tux54 is the next hop to reach 172.16.50.0/24, so it forwards it there. The reply path is symmetric because tux53 replies via tux54 directly to tux52.
- Without ICMP Redirects (using tux54 as gateway):
  - tux52 → tux54 → tux53 → tux54 → tux52
  - tux52 is configured to use tux54 as its gateway for 172.16.50.1, so it sends traffic directly to tux54. tux54 routes it to tux53. Replies go back via tux54 because tux53 uses tux54 as its gateway for 172.16.51.0/24.
- With ICMP Redirects (using tux54 as gateway):
  - tux52 → RC → tux54 → tux53 → tux54 → tux52
  - tux52 → tux54 → tux53 → tux54 → tux52
  - ICMP Redirect is triggered when a router (RC) receives a packet that it must forward out the same interface it came in on — indicating a suboptimal route. RC informs tux52 to send future packets directly to tux54. This optimizes the path by removing the unnecessary hop through RC.

To configure NAT in a commercial router, we used **/ip firewall nat add chain=srcnat action=masquerade out-interface=ether1**, but the command to make the changes is in GTK "/ip firewall nat …".

(NAT) is a method used to map a private, internal IP address space to a single public IP address, allowing multiple devices on a local network to access external networks. By acting as an intermediary, NAT enhances security by hiding internal network topology from the public internet; however, because it typically only allows outbound-initiated connections, any external client wishing to reach a specific server inside the NAT-protected network must use Port Forwarding to map a specific public port to the server's internal destination.

By observing the log **S**, it's verified that with NAT disabled, pings from tux53 to the FTP server do not respond. Meaning that there is no ICMP reply from the FTP server to the tux53 ping. This happens because the FTP server cannot route replies back to the private IP 172.16.50.1.

Exp 5 - DNS

This experiment configured DNS services on hosts tux53, tux54, and tux52 using the netlab.fe.up.pt DNS server. Name resolution was verified by pinging hostnames and testing via a web browser. Finally, DNS packet analysis was performed by sending ICMP echo requests to a new Internet hostname while using Wireshark to capture and examine the associated DNS query and response traffic.

Exp 5 - Questions

To configure the DNS service in a host, write the below lines into the opened file after using the command **sudo nano /etc/resolv.conf**:

- search netlab.fe.up.pt
- nameserver 10.227.20.3

DNS Queries (Requests) and DNS responses are exchanged by DNS. These packets follow the DNS Message Format (Identification (16bit), Flags, Questions, Answers, Authority, Additional information) and can be observed in log **T**.

Exp 6 - TCP connections

This experiment compiled and executed a file download application on host tux53 while capturing network traffic to analyze the complete FTP transaction. The analysis focused on the separate TCP control and data connections, detailing their establishment, data transfer with ARQ mechanisms, termination phases, and TCP congestion control behavior, supported by Wireshark's statistical tools. Finally, the experiment observed the impact of concurrent traffic by initiating a second download from tux52 during an ongoing transfer, using statistical analysis to evaluate how TCP throughput varies under shared network conditions. To do this last part of the experiment, the mirrors.up.pt server was used due to a constraint in traffic in the netlab.fe.up.pt server. Thus, the TCP and FTP packages will show accordingly.

Exp 6 - Questions

The FTP application opens two TCP connections: one Control Connection (Persistent connection on port 21 for FTP commands and responses) and a Data Connection (Temporary connection on a dynamically assigned port (in passive mode) for actual file transfer).

FTP control information is transported exclusively through the Control Connection.

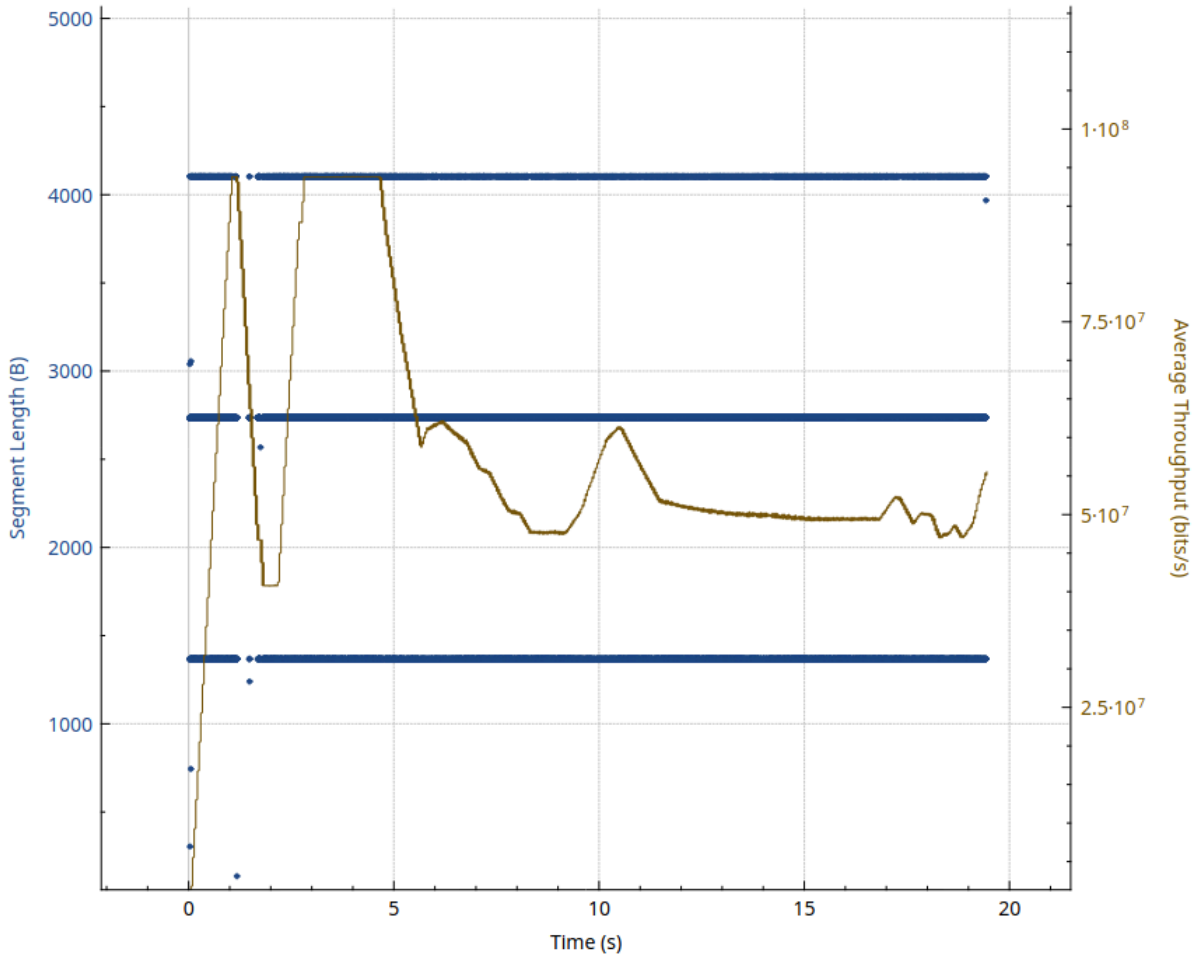
A TCP connection goes through three main phases: Connection establishment (3-way handshake), Data transfer phase and Connection termination (4-way handshake).

TCP uses Selective Repeat ARQ with cumulative acknowledgments, retransmitting segments when acknowledgments are not received within a calculated timeout, ensuring reliable data delivery. Sequence Number, Acknowledgement Number, ACK Flag, Window Size, Checksum are relevant fields on it. This can be verified through both log **A** and log **U**.

In the congestion control, the receiver's advertised TCP window tells the sender how much data it can accept. If the window becomes full, the receiver sends a zero-window ACK,

pausing transmission. Ignoring this causes packet loss and retransmissions. When the receiver is ready, a window update resumes sending. This is part of TCP's flow control.

The throughput of the first TCP connection is significantly disturbed by the start of a second connection, as shown by the immediate drop in speed at the five-second mark in the captured graph. Before the second download begins, the first connection occupies the full bandwidth at nearly 100 Mbps, but the sudden competition causes the TCP congestion control mechanism to reduce the first sender's congestion window to prevent network collapse. This behavior follows the principle of fairness, where the two flows eventually stabilize at roughly 50 Mbps each, effectively splitting the total available capacity of the network link as they both utilize the additive increase multiplicative decrease algorithm to share resources.



a. Throughput graph of download.

Conclusions

In conclusion, this lab successfully developed a functional download application and configured the necessary network to transfer a file from a server. The analysis, structured around the application architecture and six key network experiments, provided practical insight into protocols like FTP and TCP. Through configuration, packet capture, and log analysis, the core objectives of understanding data exchange, network behavior, and control mechanisms were effectively achieved.

Annexes:

Below follows all the annexes requested for the protocol, including the code of the download application, all the configuration commands and finally the logs captured and used for analysis.

Logs

```
 1 0.000000000   Routerboardc_1c:a3:… Nearest-Customer-Br… STP       60 RST. Root = 32768/0/c4:ad:34:1c:a3:2c  Cost = 0  Port = 0x8001
 2 1.441740529   172.16.120.1        172.16.1.10         TCP       74 44338 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3102652546 TSecr=0 WS=128
 3 1.442238717   172.16.1.10         172.16.120.1        TCP       74 21 → 44338 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2023103222 TSe
 4 1.442262666   172.16.120.1        172.16.1.10         TCP       66 44338 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3102652546 TSecr=2023103222
 5 1.446490946   172.16.1.10         172.16.120.1        FTP       86 Response: 220 (vsFTPd 3.0.5)
 6 1.446517568   172.16.120.1        172.16.1.10         TCP       66 44338 → 21 [ACK] Seq=1 Ack=21 Win=64256 Len=0 TSval=3102652550 TSecr=2023103226
 7 1.446598878   172.16.120.1        172.16.1.10         FTP       82 Request: USER anonymous
 8 1.446959254   172.16.1.10         172.16.120.1        TCP       66 21 → 44338 [ACK] Seq=21 Ack=17 Win=65280 Len=0 TSval=2023103226 TSecr=3102652550
 9 1.447053517   172.16.1.10         172.16.120.1        FTP      100 Response: 331 Please specify the password.
10 1.447330602   172.16.120.1        172.16.1.10         FTP       82 Request: PASS anonymous
11 1.452708325   172.16.1.10         172.16.120.1        FTP       89 Response: 230 Login successful.
12 1.452869898   172.16.120.1        172.16.1.10         FTP       72 Request: PASV
13 1.453648545   172.16.1.10         172.16.120.1        FTP      116 Response: 227 Entering Passive Mode (172,16,1,10,178,106).
14 1.453759081   172.16.120.1        172.16.1.10         TCP       74 57372 → 45674 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3102652558 TSecr=0 WS=
15 1.454195734   172.16.1.10         172.16.120.1        TCP       74 45674 → 57372 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2023103234
16 1.454216139   172.16.120.1        172.16.1.10         TCP       66 57372 → 45674 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3102652558 TSecr=2023103234
17 1.454459995   172.16.120.1        172.16.1.10         FTP       87 Request: RETR /pub/teste.txt
18 1.455374974   172.16.1.10         172.16.120.1        FTP      137 Response: 150 Opening BINARY mode data connection for /pub/teste.txt (6 bytes).
19 1.455375230   172.16.1.10         172.16.120.1        FTP-DA…   72 FTP Data: 6 bytes (PASV) (RETR /pub/teste.txt)
20 1.455375280   172.16.1.10         172.16.120.1        TCP       66 45674 → 57372 [FIN, ACK] Seq=7 Ack=1 Win=65280 Len=0 TSval=2023103235 TSecr=3102652558
21 1.455403946   172.16.120.1        172.16.1.10         TCP       66 57372 → 45674 [ACK] Seq=1 Ack=7 Win=64256 Len=0 TSval=3102652559 TSecr=2023103235
22 1.455441153   172.16.120.1        172.16.1.10         TCP       66 57372 → 45674 [FIN, ACK] Seq=1 Ack=8 Win=64256 Len=0 TSval=3102652559 TSecr=2023103235
23 1.455841870   172.16.1.10         172.16.120.1        TCP       66 45674 → 57372 [ACK] Seq=8 Ack=2 Win=65280 Len=0 TSval=2023103235 TSecr=3102652559
24 1.455954085   172.16.1.10         172.16.120.1        FTP       90 Response: 226 Transfer complete.
25 1.455983473   172.16.120.1        172.16.1.10         TCP       66 44338 → 21 [ACK] Seq=60 Ack=223 Win=64256 Len=0 TSval=3102652560 TSecr=2023103235
26 1.456168244   172.16.120.1        172.16.1.10         TCP       66 44338 → 21 [FIN, ACK] Seq=60 Ack=223 Win=64256 Len=0 TSval=3102652560 TSecr=2023103235
27 1.456597941   172.16.1.10         172.16.120.1        TCP       66 21 → 44338 [FIN, ACK] Seq=223 Ack=61 Win=65280 Len=0 TSval=2023103236 TSecr=3102652560
28 1.456618769   172.16.120.1        172.16.1.10         TCP       66 44338 → 21 [ACK] Seq=61 Ack=224 Win=64256 Len=0 TSval=3102652560 TSecr=2023103236
```

A. Part 1 Successful Download Logs

```
10 16.959540081  TPLink_c2:51:4b      Broadcast           ARP       42 Who has 172.16.50.254? Tell 172.16.50.1
11 16.959717828  TPLink_c2:10:59      TPLink_c2:51:4b     ARP       60 172.16.50.254 is at ec:75:0c:c2:10:59
12 16.959718349  TPLink_c2:17:3e      TPLink_c2:51:4b     ARP       60 172.16.50.254 is at ec:75:0c:c2:17:3e
13 16.959732789  172.16.50.1          172.16.50.254       ICMP      98 Echo (ping) request  id=0x3ff9, seq=1/256, ttl=64 (reply in 14)
14 16.959915583  172.16.50.254        172.16.50.1         ICMP      98 Echo (ping) reply    id=0x3ff9, seq=1/256, ttl=64 (request in 13)
15 17.979391769  172.16.50.1          172.16.50.254       ICMP      98 Echo (ping) request  id=0x3ff9, seq=2/512, ttl=64 (reply in 16)
16 17.979606249  172.16.50.254        172.16.50.1         ICMP      98 Echo (ping) reply    id=0x3ff9, seq=2/512, ttl=64 (request in 15)
```

B. Experiment 1

```
14 25.615824095  172.16.50.1          172.16.50.254       ICMP      98 Echo (ping) request  id=0x40a4, seq=1/256, ttl=64 (reply in 15)
15 25.616075683  172.16.50.254        172.16.50.1         ICMP      98 Echo (ping) reply    id=0x40a4, seq=1/256, ttl=64 (request in 14)
16 26.019900219  Routerboardc_1c:8e:… Nearest-Customer-Br… STP       60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
17 26.632906469  172.16.50.1          172.16.50.254       ICMP      98 Echo (ping) request  id=0x40a4, seq=2/512, ttl=64 (reply in 18)
18 26.633154122  172.16.50.254        172.16.50.1         ICMP      98 Echo (ping) reply    id=0x40a4, seq=2/512, ttl=64 (request in 17)
19 27.656905925  172.16.50.1          172.16.50.254       ICMP      98 Echo (ping) request  id=0x40a4, seq=3/768, ttl=64 (reply in 20)
20 27.657160526  172.16.50.254        172.16.50.1         ICMP      98 Echo (ping) reply    id=0x40a4, seq=3/768, ttl=64 (request in 19)
21 28.022189417  Routerboardc_1c:8e:… Nearest-Customer-Br… STP       60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
22 28.684896329  172.16.50.1          172.16.50.254       ICMP      98 Echo (ping) request  id=0x40a4, seq=4/1024, ttl=64 (reply in 23)
23 28.685148343  172.16.50.254        172.16.50.1         ICMP      98 Echo (ping) reply    id=0x40a4, seq=4/1024, ttl=64 (request in 22)
24 29.704902649  172.16.50.1          172.16.50.254       ICMP      98 Echo (ping) request  id=0x40a4, seq=5/1280, ttl=64 (reply in 25)
25 29.705147434  172.16.50.254        172.16.50.1         ICMP      98 Echo (ping) reply    id=0x40a4, seq=5/1280, ttl=64 (request in 24)
```

C. Exp 2 Ping from tux53 to tux54

```
60 58.732651938  172.16.50.254        172.16.50.1         ICMP     126 Destination unreachable (Host unreachable)
61 58.732652293  172.16.50.254        172.16.50.1         ICMP     126 Destination unreachable (Host unreachable)
62 58.732652366  172.16.50.254        172.16.50.1         ICMP     126 Destination unreachable (Host unreachable)
63 58.732652436  172.16.50.254        172.16.50.1         ICMP     126 Destination unreachable (Host unreachable)
64 59.730937011  172.16.50.1          172.16.51.1         ICMP      98 Echo (ping) request  id=0x40a5, seq=5/1280, ttl=64 (no response found!)
```

D. Exp 2 Ping from tux53 to tux52

```
 2 2.002328866   Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
 3 4.004681269   Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
 4 6.007004009   Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
 5 8.009335826   Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
 6 10.011662305  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
 7 12.013992265  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
 8 14.016318424  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
 9 16.008638300  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
10 18.010966450  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
11 20.013256112  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
12 22.015618086  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
13 24.007923627  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
14 26.010267298  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
15 28.012589646  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
```

E. Exp 2 Ping Broadcast from tux53, capture at tux52

```
35 61.208640194  172.16.50.1      172.16.50.255    ICMP     98 Echo (ping) request  id=0x410f, seq=1/256, ttl=64 (no response found!)
36 61.208919632  172.16.50.254    172.16.50.1      ICMP     98 Echo (ping) reply    id=0x410f, seq=1/256, ttl=64
37 62.031801936  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
38 62.240342736  172.16.50.1      172.16.50.255    ICMP     98 Echo (ping) request  id=0x410f, seq=2/512, ttl=64 (no response found!)
39 62.240587620  172.16.50.254    172.16.50.1      ICMP     98 Echo (ping) reply    id=0x410f, seq=2/512, ttl=64
40 63.264347924  172.16.50.1      172.16.50.255    ICMP     98 Echo (ping) request  id=0x410f, seq=3/768, ttl=64 (no response found!)
41 63.264621473  172.16.50.254    172.16.50.1      ICMP     98 Echo (ping) reply    id=0x410f, seq=3/768, ttl=64
42 64.034117668  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
43 64.288341606  172.16.50.1      172.16.50.255    ICMP     98 Echo (ping) request  id=0x410f, seq=4/1024, ttl=64 (no response found!)
44 64.288574056  172.16.50.254    172.16.50.1      ICMP     98 Echo (ping) reply    id=0x410f, seq=4/1024, ttl=64
45 65.312344899  172.16.50.1      172.16.50.255    ICMP     98 Echo (ping) request  id=0x410f, seq=5/1280, ttl=64 (no response found!)
46 65.312619249  172.16.50.254    172.16.50.1      ICMP     98 Echo (ping) reply    id=0x410f, seq=5/1280, ttl=64
```

F. Exp 2 Ping Broadcast from tux53, capture at tux53

```
13 22.193843141  172.16.50.1      172.16.50.255    ICMP     98 Echo (ping) request  id=0x410f, seq=1/256, ttl=64 (no response found!)
14 22.193880585  172.16.50.254    172.16.50.1      ICMP     98 Echo (ping) reply    id=0x410f, seq=1/256, ttl=64
15 23.225515500  172.16.50.1      172.16.50.255    ICMP     98 Echo (ping) request  id=0x410f, seq=2/512, ttl=64 (no response found!)
16 23.225556100  172.16.50.254    172.16.50.1      ICMP     98 Echo (ping) reply    id=0x410f, seq=2/512, ttl=64
17 24.017991342  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8002
18 24.249550204  172.16.50.1      172.16.50.255    ICMP     98 Echo (ping) request  id=0x410f, seq=3/768, ttl=64 (no response found!)
19 24.249591732  172.16.50.254    172.16.50.1      ICMP     98 Echo (ping) reply    id=0x410f, seq=3/768, ttl=64
20 25.273514893  172.16.50.1      172.16.50.255    ICMP     98 Echo (ping) request  id=0x410f, seq=4/1024, ttl=64 (no response found!)
21 25.273553512  172.16.50.254    172.16.50.1      ICMP     98 Echo (ping) reply    id=0x410f, seq=4/1024, ttl=64
22 26.010318760  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8002
23 26.297561860  172.16.50.1      172.16.50.255    ICMP     98 Echo (ping) request  id=0x410f, seq=5/1280, ttl=64 (no response found!)
24 26.297601385  172.16.50.254    172.16.50.1      ICMP     98 Echo (ping) reply    id=0x410f, seq=5/1280, ttl=64
25 27.321530043  172.16.50.1      172.16.50.255    ICMP     98 Echo (ping) request  id=0x410f, seq=6/1536, ttl=64 (no response found!)
26 27.321570791  172.16.50.254    172.16.50.1      ICMP     98 Echo (ping) reply    id=0x410f, seq=6/1536, ttl=64
```

G. Exp 2 Ping Broadcast from tux53, capture at tux54

```
20 31.586081012  172.16.51.1      172.16.51.255    ICMP     98 Echo (ping) request  id=0x2d86, seq=1/256, ttl=64 (no response found!)
21 32.036564481  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
22 32.606911740  172.16.51.1      172.16.51.255    ICMP     98 Echo (ping) request  id=0x2d86, seq=2/512, ttl=64 (no response found!)
23 33.630896332  172.16.51.1      172.16.51.255    ICMP     98 Echo (ping) request  id=0x2d86, seq=3/768, ttl=64 (no response found!)
24 34.038876181  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
25 34.654886682  172.16.51.1      172.16.51.255    ICMP     98 Echo (ping) request  id=0x2d86, seq=4/1024, ttl=64 (no response found!)
26 35.678893844  172.16.51.1      172.16.51.255    ICMP     98 Echo (ping) request  id=0x2d86, seq=5/1280, ttl=64 (no response found!)
27 36.041201185  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
28 36.702887294  172.16.51.1      172.16.51.255    ICMP     98 Echo (ping) request  id=0x2d86, seq=6/1536, ttl=64 (no response found!)
29 37.726888595  172.16.51.1      172.16.51.255    ICMP     98 Echo (ping) request  id=0x2d86, seq=7/1792, ttl=64 (no response found!)
30 38.043532679  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
31 38.750890706  172.16.51.1      172.16.51.255    ICMP     98 Echo (ping) request  id=0x2d86, seq=8/2048, ttl=64 (no response found!)
32 39.774881279  172.16.51.1      172.16.51.255    ICMP     98 Echo (ping) request  id=0x2d86, seq=9/2304, ttl=64 (no response found!)
33 40.045851525  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
34 40.798891650  172.16.51.1      172.16.51.255    ICMP     98 Echo (ping) request  id=0x2d86, seq=10/2560, ttl=64 (no response found!)
35 41.822889745  172.16.51.1      172.16.51.255    ICMP     98 Echo (ping) request  id=0x2d86, seq=11/2816, ttl=64 (no response found!)
36 42.048191014  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
37 42.846887420  172.16.51.1      172.16.51.255    ICMP     98 Echo (ping) request  id=0x2d86, seq=12/3072, ttl=64 (no response found!)
38 43.870883815  172.16.51.1      172.16.51.255    ICMP     98 Echo (ping) request  id=0x2d86, seq=13/3328, ttl=64 (no response found!)
39 44.040494350  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:14  Cost = 0  Port = 0x8001
40 44.894894505  172.16.51.1      172.16.51.255    ICMP     98 Echo (ping) request  id=0x2d86, seq=14/3584, ttl=64 (no response found!)
41 45.918919494  172.16.51.1      172.16.51.255    ICMP     98 Echo (ping) request  id=0x2d86, seq=15/3840, ttl=64 (no response found!)
```

H. Exp 2 Ping Broadcast from tux52, capture at tux52

```
 7 12.003846846  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
 8 14.006208072  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
 9 16.008533165  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
10 18.010859934  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
11 20.013160394  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
12 22.015508880  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
13 24.017790763  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
14 26.020144372  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
15 28.022120553  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
16 30.024447009  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
```

I. Exp 2 Ping Broadcast from tux52, capture at tux53

```
 8 8.008996653   Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8002
 9 10.011006968  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8002
10 12.013332679  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8002
11 14.015666889  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8002
12 16.018001080  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8002
13 18.020326918  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8002
14 20.022651237  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8002
15 22.024950184  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8002
16 24.027262204  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8002
17 26.029629067  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8002
18 28.031963791  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8002
```

J. Exp 2 Ping Broadcast from tux52, capture at tux54

```
14 24.764176785  172.16.50.1      172.16.50.254    ICMP      98 Echo (ping) request  id=0x444f, seq=1/256, ttl=64 (reply in 17)
15 24.764433494  TPLink_c2:10:59   Broadcast        ARP       60 Who has 172.16.50.1? Tell 172.16.50.254
16 24.764462427  TPLink_c2:51:4b   TPLink_c2:10:59  ARP       42 172.16.50.1 is at ec:75:0c:c2:51:4b
17 24.764620761  172.16.50.254    172.16.50.1      ICMP      98 Echo (ping) reply    id=0x444f, seq=1/256, ttl=64 (request in 14)
18 25.781404792  172.16.50.1      172.16.50.254    ICMP      98 Echo (ping) request  id=0x444f, seq=2/512, ttl=64 (reply in 19)
19 25.781588521  172.16.50.254    172.16.50.1      ICMP      98 Echo (ping) reply    id=0x444f, seq=2/512, ttl=64 (request in 18)
20 26.029234625  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
21 26.805381166  172.16.50.1      172.16.50.254    ICMP      98 Echo (ping) request  id=0x444f, seq=3/768, ttl=64 (reply in 22)
22 26.805598609  172.16.50.254    172.16.50.1      ICMP      98 Echo (ping) reply    id=0x444f, seq=3/768, ttl=64 (request in 21)
23 27.829385579  172.16.50.1      172.16.50.254    ICMP      98 Echo (ping) request  id=0x444f, seq=4/1024, ttl=64 (reply in 24)
24 27.829639593  172.16.50.254    172.16.50.1      ICMP      98 Echo (ping) reply    id=0x444f, seq=4/1024, ttl=64 (request in 23)
```

K. Exp 3 Ping from tux53 to 172.16.50.254

```
 5 7.648602141   172.16.50.1      172.16.51.253    ICMP      98 Echo (ping) request  id=0x44ba, seq=1/256, ttl=64 (reply in 6)
 6 7.648830170   172.16.51.253    172.16.50.1      ICMP      98 Echo (ping) reply    id=0x44ba, seq=1/256, ttl=64 (request in 5)
 7 8.009020545   Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
 8 8.649718184   172.16.50.1      172.16.51.253    ICMP      98 Echo (ping) request  id=0x44ba, seq=2/512, ttl=64 (reply in 9)
 9 8.649972300   172.16.51.253    172.16.50.1      ICMP      98 Echo (ping) reply    id=0x44ba, seq=2/512, ttl=64 (request in 8)
10 9.673688261   172.16.50.1      172.16.51.253    ICMP      98 Echo (ping) request  id=0x44ba, seq=3/768, ttl=64 (reply in 11)
11 9.673911046   172.16.51.253    172.16.50.1      ICMP      98 Echo (ping) reply    id=0x44ba, seq=3/768, ttl=64 (request in 10)
12 10.011258825  Routerboardc_1c:8e:… Nearest-Customer-Br… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
13 10.697662927  172.16.50.1      172.16.51.253    ICMP      98 Echo (ping) request  id=0x44ba, seq=4/1024, ttl=64 (reply in 14)
14 10.697886850  172.16.51.253    172.16.50.1      ICMP      98 Echo (ping) reply    id=0x44ba, seq=4/1024, ttl=64 (request in 13)
15 11.721693455  172.16.50.1      172.16.51.253    ICMP      98 Echo (ping) request  id=0x44ba, seq=5/1280, ttl=64 (reply in 16)
16 11.721923269  172.16.51.253    172.16.50.1      ICMP      98 Echo (ping) reply    id=0x44ba, seq=5/1280, ttl=64 (request in 15)
```

L. Exp 3 Ping from tux53 to 172.16.51.253

```
88 150.618431793 172.16.50.1      172.16.51.1      ICMP      98 Echo (ping) request  id=0x4576, seq=4/1024, ttl=64 (no response found!)
89 150.618533301 172.16.50.254    172.16.50.1      ICMP     126 Destination unreachable (Host unreachable)
90 150.618552670 172.16.50.254    172.16.50.1      ICMP     126 Destination unreachable (Host unreachable)
91 150.618561558 172.16.50.254    172.16.50.1      ICMP     126 Destination unreachable (Host unreachable)
92 150.618568594 172.16.50.254    172.16.50.1      ICMP     126 Destination unreachable (Host unreachable)
93 151.642498333 172.16.50.1      172.16.51.1      ICMP      98 Echo (ping) request  id=0x4576, seq=5/1280, ttl=64 (no response found!)
```

M. Exp 3 Ping from tux53 to tux52, interface e1

```
78 145.552600936 TPLink_c2:17:3e      Broadcast              ARP          42 Who has 172.16.51.1? Tell 172.16.51.253
79 146.145006444 Routerboardc_1c:8e:… Nearest-Customer-Br… STP          60 RST. Root = 32768/0/c4:ad:34:1c:8e:13  Co
80 146.568144689 TPLink_c2:17:3e      Broadcast              ARP          42 Who has 172.16.51.1? Tell 172.16.51.253
81 147.592160602 TPLink_c2:17:3e      Broadcast              ARP          42 Who has 172.16.51.1? Tell 172.16.51.253
82 148.147260032 Routerboardc_1c:8e:… Nearest-Customer-Br… STP          60 RST. Root = 32768/0/c4:ad:34:1c:8e:13  Co
83 149.640154252 TPLink_c2:17:3e      Broadcast              ARP          42 Who has 172.16.51.1? Tell 172.16.51.253
84 150.149519259 Routerboardc_1c:8e:… Nearest-Customer-Br… STP          60 RST. Root = 32768/0/c4:ad:34:1c:8e:13  Co
85 150.664141951 TPLink_c2:17:3e      Broadcast              ARP          42 Who has 172.16.51.1? Tell 172.16.51.253
86 151.688141332 TPLink_c2:17:3e      Broadcast              ARP          42 Who has 172.16.51.1? Tell 172.16.51.253
87 152.151757095 Routerboardc_1c:8e:… Nearest-Customer-Br… STP          60 RST. Root = 32768/0/c4:ad:34:1c:8e:13  Co
88 153.736149491 TPLink_c2:17:3e      Broadcast              ARP          42 Who has 172.16.51.1? Tell 172.16.51.253
89 154.154009183 Routerboardc_1c:8e:… Nearest-Customer-Br… STP          60 RST. Root = 32768/0/c4:ad:34:1c:8e:13  Co
90 154.760142181 TPLink_c2:17:3e      Broadcast              ARP          42 Who has 172.16.51.1? Tell 172.16.51.253
91 155.784184511 TPLink_c2:17:3e      Broadcast              ARP          42 Who has 172.16.51.1? Tell 172.16.51.253
92 156.156247642 Routerboardc_1c:8e:… Nearest-Customer-Br… STP          60 RST. Root = 32768/0/c4:ad:34:1c:8e:13  Co
93 157.832098388 TPLink_c2:17:3e      Broadcast              ARP          42 Who has 172.16.51.1? Tell 172.16.51.253
94 158.158500239 Routerboardc_1c:8e:… Nearest-Customer-Br… STP          60 RST. Root = 32768/0/c4:ad:34:1c:8e:13  Co
95 158.856146638 TPLink_c2:17:3e      Broadcast              ARP          42 Who has 172.16.51.1? Tell 172.16.51.253
96 159.880149058 TPLink_c2:17:3e      Broadcast              ARP          42 Who has 172.16.51.1? Tell 172.16.51.253
```

N. Exp 3 Ping from tux53 to tux52, interface e2

```
15 6.894417682  TPLink_c2:17:3e      TPLink_c2:10:4b      ARP          60 Who has 172.16.51.1? Tell 172.16.51.253
16 6.894436763  TPLink_c2:10:4b      TPLink_c2:17:3e      ARP          42 172.16.51.1 is at ec:75:0c:c2:10:4b
17 6.985333623  172.16.51.1          172.16.50.1          ICMP         98 Echo (ping) request  id=0x0fad, seq=6/1536, ttl=64 (reply in 18)
18 6.985748343  172.16.50.1          172.16.51.1          ICMP         98 Echo (ping) reply    id=0x0fad, seq=6/1536, ttl=63 (request in 17)
19 7.081302262  TPLink_c2:10:4b      TPLink_c2:17:3e      ARP          42 Who has 172.16.51.253? Tell 172.16.51.1
20 7.081522294  TPLink_c2:17:3e      TPLink_c2:10:4b      ARP          60 172.16.51.253 is at ec:75:0c:c2:17:3e
21 8.007687479  Routerboardc_1c:8e:… Nearest-Customer-Br… STP          60 RST. Root = 32768/0/74:4d:28:ea:74:f5  Cost = 10  Port = 0x8001
22 8.009326489  172.16.51.1          172.16.50.1          ICMP         98 Echo (ping) request  id=0x0fad, seq=7/1792, ttl=64 (reply in 23)
23 8.009736383  172.16.50.1          172.16.51.1          ICMP         98 Echo (ping) reply    id=0x0fad, seq=7/1792, ttl=63 (request in 22)
24 9.033343619  172.16.51.1          172.16.50.1          ICMP         98 Echo (ping) request  id=0x0fad, seq=8/2048, ttl=64 (reply in 25)
25 9.033740022  172.16.50.1          172.16.51.1          ICMP         98 Echo (ping) reply    id=0x0fad, seq=8/2048, ttl=63 (request in 24)
26 10.010507819 Routerboardc_1c:8e:… Nearest-Customer-Br… STP          60 RST. Root = 32768/0/74:4d:28:ea:74:f5  Cost = 10  Port = 0x8001
27 10.057319880 172.16.51.1          172.16.50.1          ICMP         98 Echo (ping) request  id=0x0fad, seq=9/2304, ttl=64 (reply in 28)
28 10.057705380 172.16.50.1          172.16.51.1          ICMP         98 Echo (ping) reply    id=0x0fad, seq=9/2304, ttl=63 (request in 27)
29 11.081373871 172.16.51.1          172.16.50.1          ICMP         98 Echo (ping) request  id=0x0fad, seq=10/2560, ttl=64 (reply in 30)
30 11.081770655 172.16.50.1          172.16.51.1          ICMP         98 Echo (ping) reply    id=0x0fad, seq=10/2560, ttl=63 (request in 29)
```

O. Exp 4 Ping from tux52 to tux53, with no redirects

```
17 18.092591971 Routerboardc_ea:74:… Broadcast            ARP          60 Who has 172.16.51.1? Tell 172.16.51.254
18 18.092606902 TPLink_c2:10:4b      Routerboardc_ea:74:… ARP          42 172.16.51.1 is at ec:75:0c:c2:10:4b
19 18.092717810 172.16.51.254        172.16.51.1          ICMP        126 Redirect             (Redirect for host)
20 18.092910116 172.16.50.1          172.16.51.1          ICMP         98 Echo (ping) reply    id=0x0f50, seq=2/512, ttl=63 (request in 16)
21 19.098291665 172.16.51.1          172.16.50.1          ICMP         98 Echo (ping) request  id=0x0f50, seq=3/768, ttl=64 (reply in 23)
22 19.098468886 172.16.51.254        172.16.51.1          ICMP        126 Redirect             (Redirect for host)
23 19.098800130 172.16.50.1          172.16.51.1          ICMP         98 Echo (ping) reply    id=0x0f50, seq=3/768, ttl=63 (request in 21)
24 20.011589026 Routerboardc_1c:8e:… Nearest-Customer-Br… STP          60 RST. Root = 32768/0/74:4d:28:ea:74:f5  Cost = 10  Port = 0x8001
25 20.122303131 172.16.51.1          172.16.50.1          ICMP         98 Echo (ping) request  id=0x0f50, seq=4/1024, ttl=64 (reply in 27)
26 20.122451100 172.16.51.254        172.16.51.1          ICMP        126 Redirect             (Redirect for host)
27 20.122771989 172.16.50.1          172.16.51.1          ICMP         98 Echo (ping) reply    id=0x0f50, seq=4/1024, ttl=63 (request in 25)
28 21.146291224 172.16.51.1          172.16.50.1          ICMP         98 Echo (ping) request  id=0x0f50, seq=5/1280, ttl=64 (reply in 30)
29 21.146482682 172.16.51.254        172.16.51.1          ICMP        126 Redirect             (Redirect for host)
30 21.146729926 172.16.50.1          172.16.51.1          ICMP         98 Echo (ping) reply    id=0x0f50, seq=5/1280, ttl=63 (request in 28)
31 22.013692137 Routerboardc_1c:8e:… Nearest-Customer-Br… STP          60 RST. Root = 32768/0/74:4d:28:ea:74:f5  Cost = 10  Port = 0x8001
32 22.138244005 TPLink_c2:10:4b      Routerboardc_ea:74:… ARP          42 Who has 172.16.51.254? Tell 172.16.51.1
33 22.138388214 Routerboardc_ea:74:… TPLink_c2:10:4b      ARP          60 172.16.51.254 is at 74:4d:28:ea:74:f5
34 22.170288473 172.16.51.1          172.16.50.1          ICMP         98 Echo (ping) request  id=0x0f50, seq=6/1536, ttl=64 (reply in 36)
35 22.170440878 172.16.51.254        172.16.51.1          ICMP        126 Redirect             (Redirect for host)
36 22.170714060 172.16.50.1          172.16.51.1          ICMP         98 Echo (ping) reply    id=0x0f50, seq=6/1536, ttl=63 (request in 34)
37 23.194291680 172.16.51.1          172.16.50.1          ICMP         98 Echo (ping) request  id=0x0f50, seq=7/1792, ttl=64 (reply in 38)
38 23.194747410 172.16.50.1          172.16.51.1          ICMP         98 Echo (ping) reply    id=0x0f50, seq=7/1792, ttl=63 (request in 37)
```

P. Exp 4 Ping from tux52 to tux53 through the router, with no redirects

```
31 48.211169050 172.16.51.1          172.16.50.1          ICMP         98 Echo (ping) request  id=0x0ff7, seq=1/256, ttl=64 (reply in 32)
32 48.211580909 172.16.50.1          172.16.51.1          ICMP         98 Echo (ping) reply    id=0x0ff7, seq=1/256, ttl=63 (request in 31)
33 49.233881855 172.16.51.1          172.16.50.1          ICMP         98 Echo (ping) request  id=0x0ff7, seq=2/512, ttl=64 (reply in 34)
34 49.234249447 172.16.50.1          172.16.51.1          ICMP         98 Echo (ping) reply    id=0x0ff7, seq=2/512, ttl=63 (request in 33)
35 50.031777755 Routerboardc_1c:8e:… Nearest-Customer-Br… STP          60 RST. Root = 32768/0/74:4d:28:ea:74:f5  Cost = 10  Port = 0x8001
36 50.257928453 172.16.51.1          172.16.50.1          ICMP         98 Echo (ping) request  id=0x0ff7, seq=3/768, ttl=64 (reply in 37)
37 50.258304962 172.16.50.1          172.16.51.1          ICMP         98 Echo (ping) reply    id=0x0ff7, seq=3/768, ttl=63 (request in 36)
38 51.281910245 172.16.51.1          172.16.50.1          ICMP         98 Echo (ping) request  id=0x0ff7, seq=4/1024, ttl=64 (reply in 39)
39 51.282273705 172.16.50.1          172.16.51.1          ICMP         98 Echo (ping) reply    id=0x0ff7, seq=4/1024, ttl=63 (request in 38)
40 52.033562142 Routerboardc_1c:8e:… Nearest-Customer-Br… STP          60 RST. Root = 32768/0/74:4d:28:ea:74:f5  Cost = 10  Port = 0x8001
41 52.305895847 172.16.51.1          172.16.50.1          ICMP         98 Echo (ping) request  id=0x0ff7, seq=5/1280, ttl=64 (reply in 42)
42 52.306241878 172.16.50.1          172.16.51.1          ICMP         98 Echo (ping) reply    id=0x0ff7, seq=5/1280, ttl=63 (request in 41)
43 53.329933744 172.16.51.1          172.16.50.1          ICMP         98 Echo (ping) request  id=0x0ff7, seq=6/1536, ttl=64 (reply in 44)
44 53.330335334 172.16.50.1          172.16.51.1          ICMP         98 Echo (ping) reply    id=0x0ff7, seq=6/1536, ttl=63 (request in 43)
```

Q. Exp 4 Ping from tux52 to tux53, with redirects

```
  3 2.784954764   172.16.50.1          172.16.1.10          ICMP      98 Echo (ping) request  id=0x2644, seq=1/256, ttl=64 (reply in 4)
  4 2.785852442   172.16.1.10          172.16.50.1          ICMP      98 Echo (ping) reply    id=0x2644, seq=1/256, ttl=62 (request in 3)
  5 3.786003992   172.16.50.1          172.16.1.10          ICMP      98 Echo (ping) request  id=0x2644, seq=2/512, ttl=64 (reply in 6)
  6 3.786516156   172.16.1.10          172.16.50.1          ICMP      98 Echo (ping) reply    id=0x2644, seq=2/512, ttl=62 (request in 5)
  7 4.004274088   Routerboardc_1c:8e:… Nearest-Customer-Br… STP       60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
  8 4.803610423   172.16.50.1          172.16.1.10          ICMP      98 Echo (ping) request  id=0x2644, seq=3/768, ttl=64 (reply in 9)
  9 4.804159965   172.16.1.10          172.16.50.1          ICMP      98 Echo (ping) reply    id=0x2644, seq=3/768, ttl=62 (request in 8)
 10 5.827611895   172.16.50.1          172.16.1.10          ICMP      98 Echo (ping) request  id=0x2644, seq=4/1024, ttl=64 (reply in 11)
 11 5.828158335   172.16.1.10          172.16.50.1          ICMP      98 Echo (ping) reply    id=0x2644, seq=4/1024, ttl=62 (request in 10)
 12 6.006402793   Routerboardc_1c:8e:… Nearest-Customer-Br… STP       60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
 13 6.851610068   172.16.50.1          172.16.1.10          ICMP      98 Echo (ping) request  id=0x2644, seq=5/1280, ttl=64 (reply in 14)
 14 6.852159154   172.16.1.10          172.16.50.1          ICMP      98 Echo (ping) reply    id=0x2644, seq=5/1280, ttl=62 (request in 13)
```

R. Exp 4 Ping from tux53 to FTP server, with NAT

```
 5 6.793693928   172.16.50.1          172.16.1.10          ICMP      98 Echo (ping) request  id=0x26ad, seq=1/256, ttl=64 (no response found!)
 6 7.819183852   172.16.50.1          172.16.1.10          ICMP      98 Echo (ping) request  id=0x26ad, seq=2/512, ttl=64 (no response found!)
 7 8.008534775   Routerboardc_1c:8e:… Nearest-Customer-Br… STP       60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
 8 8.843185466   172.16.50.1          172.16.1.10          ICMP      98 Echo (ping) request  id=0x26ad, seq=3/768, ttl=64 (no response found!)
 9 9.867190155   172.16.50.1          172.16.1.10          ICMP      98 Echo (ping) request  id=0x26ad, seq=4/1024, ttl=64 (no response found!)
10 10.010038233  Routerboardc_1c:8e:… Nearest-Customer-Br… STP       60 RST. Root = 32768/0/c4:ad:34:1c:8e:15  Cost = 0  Port = 0x8001
11 10.891187423  172.16.50.1          172.16.1.10          ICMP      98 Echo (ping) request  id=0x26ad, seq=5/1280, ttl=64 (no response found!)
12 11.915176443  172.16.50.1          172.16.1.10          ICMP      98 Echo (ping) request  id=0x26ad, seq=6/1536, ttl=64 (no response found!)
```

S. Exp 4 Ping from tux53 to FTP server, with no NAT

```
 78 13.527672998  10.227.20.53          10.227.20.3          DNS       70 Standard query 0x9843 A google.com
 79 13.527694533  10.227.20.53          10.227.20.3          DNS       70 Standard query 0x4846 AAAA google.com
 80 13.528466986  10.227.20.3           10.227.20.53         DNS       86 Standard query response 0x9843 A google.com A 142.251.209.174
 81 13.528467317  10.227.20.3           10.227.20.53         DNS       98 Standard query response 0x4846 AAAA google.com AAAA 2a00:1450:4006:817::200e
 82 13.528783417  10.227.20.53          142.251.209.174      ICMP      98 Echo (ping) request  id=0x139b, seq=1/256, ttl=64 (reply in 83)
 83 13.559890545  142.251.209.174       10.227.20.53         ICMP      98 Echo (ping) reply    id=0x139b, seq=1/256, ttl=110 (request in 82)
 84 13.560223508  10.227.20.53          10.227.20.3          DNS       88 Standard query 0x2b4e PTR 174.209.251.142.in-addr.arpa
 85 13.560960059  10.227.20.3           10.227.20.53         DNS      156 Standard query response 0x2b4e PTR 174.209.251.142.in-addr.arpa PTR ncmrsa-ah-in-f14.1e100.net
 86 14.308173860  ProxmoxServe_e7:5e:… Broadcast            ARP       60 Who has 10.227.20.83? Tell 10.227.20.3
 87 14.310341885  Routerboardc_20:25:… Broadcast            ARP       60 Who has 10.227.20.83? Tell 10.227.20.254
 88 14.530167796  10.227.20.53          142.251.209.174      ICMP      98 Echo (ping) request  id=0x139b, seq=2/512, ttl=64 (reply in 89)
 89 14.559913461  142.251.209.174       10.227.20.53         ICMP      98 Echo (ping) reply    id=0x139b, seq=2/512, ttl=110 (request in 88)
 90 14.560308438  10.227.20.53          10.227.20.3          DNS       88 Standard query 0xd17e PTR 174.209.251.142.in-addr.arpa
 91 14.561010701  10.227.20.3           10.227.20.53         DNS      156 Standard query response 0xd17e PTR 174.209.251.142.in-addr.arpa PTR ncmrsa-ah-in-f14.1e100.net
 92 14.942013908  fe80::5ef9:ddff:fe7… ff02::2              ICMPv6    62 Router Solicitation
 93 15.185824983  Cisco_3a:f1:07        Nearest-Customer-Br… STP       60 RST. Root = 4096/1/30:37:a6:d4:1c:00  Cost = 200019  Port = 0x8007
 94 15.531185493  10.227.20.53          142.251.209.174      ICMP      98 Echo (ping) request  id=0x139b, seq=3/768, ttl=64 (reply in 95)
 95 15.560936785  142.251.209.174       10.227.20.53         ICMP      98 Echo (ping) reply    id=0x139b, seq=3/768, ttl=110 (request in 94)
 96 15.561308724  10.227.20.53          10.227.20.3          DNS       88 Standard query 0x83d5 PTR 174.209.251.142.in-addr.arpa
 97 15.562087594  10.227.20.3           10.227.20.53         DNS      156 Standard query response 0x83d5 PTR 174.209.251.142.in-addr.arpa PTR ncmrsa-ah-in-f14.1e100.net
 98 15.651177956  10.227.20.53          142.250.200.138      UDP       71 57259 → 443 Len=29
 99 15.667925666  142.250.200.138       10.227.20.53         UDP       71 443 → 57259 Len=29
100 16.531790717  10.227.20.53          142.251.209.174      ICMP      98 Echo (ping) request  id=0x139b, seq=4/1024, ttl=64 (reply in 101)
101 16.561958940  142.251.209.174       10.227.20.53         ICMP      98 Echo (ping) reply    id=0x139b, seq=4/1024, ttl=110 (request in 100)
```

T. Exp 5 Related DNS packets

```
20 2.345914397   10.227.20.53          193.137.29.15        TCP       74 42666 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=4194776040 TSecr=0 WS=128
21 2.405809229   193.137.29.15         10.227.20.53         TCP       74 21 → 42666 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1380 SACK_PERM TSval=108010161 TSecr=419
22 2.405859213   10.227.20.53          193.137.29.15        TCP       66 42666 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=4194776100 TSecr=108010161
23 2.451971752   193.137.29.15         10.227.20.53         FTP      139 Response: 220-Welcome to the University of Porto's mirror archive (mirrors.up.pt)
24 2.451972005   193.137.29.15         10.227.20.53         FTP      141 Response: 220----------------------------------------------------------------------
25 2.451972077   193.137.29.15         10.227.20.53         FTP      151 Response: 220-All connections and transfers are logged. The max number of connections is 200.
26 2.451972147   193.137.29.15         10.227.20.53         FTP      146 Response: 220-
27 2.451972215   193.137.29.15         10.227.20.53         FTP      145 Response: 220-Questions and comments can be sent to mirrors@uporto.pt
28 2.451998036   10.227.20.53          193.137.29.15        TCP       66 42666 → 21 [ACK] Seq=1 Ack=74 Win=64256 Len=0 TSval=4194776146 TSecr=108010236
29 2.452012609   10.227.20.53          193.137.29.15        TCP       66 42666 → 21 [ACK] Seq=1 Ack=149 Win=64256 Len=0 TSval=4194776146 TSecr=108010236
30 2.452022632   10.227.20.53          193.137.29.15        TCP       66 42666 → 21 [ACK] Seq=1 Ack=234 Win=64256 Len=0 TSval=4194776146 TSecr=108010236
31 2.452031096   10.227.20.53          193.137.29.15        TCP       66 42666 → 21 [ACK] Seq=1 Ack=314 Win=64256 Len=0 TSval=4194776146 TSecr=108010236
32 2.452061110   10.227.20.53          193.137.29.15        TCP       66 42666 → 21 [ACK] Seq=1 Ack=393 Win=64256 Len=0 TSval=4194776146 TSecr=108010236
33 2.452114567   10.227.20.53          193.137.29.15        FTP       82 Request: USER anonymous
34 2.515706990   193.137.29.15         10.227.20.53         TCP       66 21 → 42666 [ACK] Seq=393 Ack=17 Win=65280 Len=0 TSval=108010309 TSecr=4194776146
35 2.515707414   193.137.29.15         10.227.20.53         FTP      100 Response: 331 Please specify the password.
36 2.515837328   10.227.20.53          193.137.29.15        FTP       82 Request: PASS anonymous
37 2.585775278   193.137.29.15         10.227.20.53         FTP       89 Response: 230 Login successful.
38 2.585873122   10.227.20.53          193.137.29.15        FTP       72 Request: PASV
39 2.655826049   193.137.29.15         10.227.20.53         FTP      118 Response: 227 Entering Passive Mode (193,137,29,15,208,190).
40 2.656000777   10.227.20.53          193.137.29.15        TCP       74 45686 → 53438 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=4194776350 TSecr=0 WS=128
41 2.657646672   193.137.29.15         10.227.20.53         TCP       74 53438 → 45686 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1380 SACK_PERM TSval=108010469 TSecr=
42 2.657678514   10.227.20.53          193.137.29.15        TCP       66 45686 → 53438 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=4194776351 TSecr=108010469
43 2.657760487   10.227.20.53          193.137.29.15        FTP      121 Request: RETR pub/archlinux/archive/iso/arch-0.8-base-i686.iso
44 2.676998848   193.137.29.15         10.227.20.53         FTP-DA… 2802 FTP Data: 2736 bytes (PASV) (RETR pub/archlinux/archive/iso/arch-0.8-base-i686.iso)
45 2.677021692   10.227.20.53          193.137.29.15        TCP       66 45686 → 53438 [ACK] Seq=1 Ack=2737 Win=63488 Len=0 TSval=4194776371 TSecr=108010488
```

U. Exp 6 ARQ TCP Mechanism

Configuration Commands

```
Tux 53:
sudo systemctl restart networking
sudo ifconfig if_e1 up
sudo ifconfig if_e1 172.16.50.1/24
sudo route add -net 172.16.51.0/24 gw 172.16.50.254
sudo route add -net 172.16.1.0/24 gw 172.16.50.254

Tux 54:
sudo systemctl restart networking
sudo ifconfig if_e1 up
sudo ifconfig if_e1 172.16.50.254/24
sudo ifconfig if_e2 up
sudo ifconfig if_e2 172.16.51.253/24
sudo sysctl net.ipv4.ip_forward=1
sudo sysctl net.ipv4.icmp_echo_ignore_broadcasts=0
sudo route add -net 172.16.1.0/24 gw 172.16.51.254

Tux 52:
sudo systemctl restart networking
sudo ifconfig if_e1 up
sudo ifconfig if_e1 172.16.51.1/24
sudo route add -net 172.16.50.0/24 gw 172.16.51.253
sudo route add -net 172.16.1.0/24 gw 172.16.51.254

GTK:
/system reset-configuration
 - Bridges:
    /interface bridge add name=bridge50
    /interface bridge add name=bridge51
 - Ports:
    /interface bridge port remove [find interface=ether2]
    /interface bridge port add bridge=bridge51 interface=ether2
    /interface bridge port remove [find interface=ether3]
    /interface bridge port remove [find interface=ether4]
    /interface bridge port add bridge=bridge50 interface=ether3
    /interface bridge port add bridge=bridge50 interface=ether4
    /interface bridge port remove [find interface=ether24]
    /interface bridge port add bridge=bridge51 interface=ether24
    /interface bridge port remove [find interface=ether12]
    /interface bridge port add bridge=bridge51 interface=ether12

 - Interface: (Router)
    /ip address add address=172.16.51.254/24 interface=ether2
    /ip address add address=172.16.1.51/24 interface=ether1
    /ip route add dst-address=172.16.50.0/24 gateway=172.16.51.253
```

Code:

The protocol code was divided into multiple files has follows:

download.c

```c
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

#include "parser.h"
#include "getip.h"
#include "clientTCP.h"
#include "create_file.h"

int main(int argc, char *argv[]){

    if (argc != 2){
        return 1;
    }

    struct URL url;

    // Step 1

    if (parse(&url, argv[1])){
        return 1;
    }

    printf("URL Components:\n");
    printf("Protocol: %s\n", url.protocol);
    printf("User: %s\n", url.user);
    printf("Password: %s\n", url.password);
    printf("Host: %s\n", url.host);
    printf("Path: %s\n", url.path);

    // Step 2

    getip(&url);
    printf("IP Address : %s\n", url.ip);

    // Step 3

    if (term_A1(url)){
        return 1;
    }

    // Step 4

    if (term_B1()){
        return 1;
    }

    // Step 5

    if (term_A2(url)){
        return 1;
    }

    // Step 6
    char *content;
    if (term_B2(&content)){
        return 1;
    }

    // Step 7

    if (create_file(url, content)){
        return 1;
    }

    free_url(&url);

    return 0;
}
```

clientTCP.h

```h
#ifndef CLIENTTCP_H
#define CLIENTTCP_H

#include <netinet/in.h>
#include "parser.h"

int read_ftp_response(FILE *sock_file, int *code);
int configure_socket(char *ip, uint16_t port, int *sockfd, FILE **sock_file);
int term_B2(char **content);
int term_A2(struct URL url);
int term_B1();
int term_A1(struct URL url);

#endif /* CLIENTTCP_H */
```

clientTCP.c

```c
/**      (C)2000-2021 FEUP
 *       tidy up some includes and parameters
 * */

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

#include "parser.h"

#define CONTROLLER_PORT 21
#define BUF_SIZE 100000

int sockfd1, sockfd2;
struct sockaddr_in server_addr;
int ip1, ip2, ip3, ip4, port1, port2;
ssize_t bytes;
char line[BUF_SIZE];
int code;
FILE *sock_file1, *sock_file2;

int read_ftp_response(FILE *sock_file, int *code) {

    while (fgets(line, sizeof(line), sock_file) != NULL) {
        // Remove trailing \r\n
        line[strcspn(line, "\r\n")] = '\0';
        //printf("%s\n", line);  // for debugging

        // Parse code from this line
        if (sscanf(line, "%d", code) != 1) {
            return 1;
        }

        // Check if this is a termination line
        if (line[3] == ' ') {
            //printf("%d\n", *code); // for debugging
            return 0;
        }
    }

    return 1;
}
```

```c
int configure_socket(char *ip, uint16_t port, int *sockfd, FILE **sock_file){
    /*server address handling*/
    bzero((char *) &server_addr, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = inet_addr(ip);    /*32 bit Internet address network byte ordered*/
    server_addr.sin_port = htons(port);        /*server TCP port must be network byte ordered */

    /*open a TCP socket*/
    if ((*sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket()");
        return 1;
    }

    /*connect to the server*/
    if (connect(*sockfd,
                (struct sockaddr *) &server_addr,
                sizeof(server_addr)) < 0) {
        perror("connect()");
        return 1;
    }

    /* Create FILE* stream*/
    if (sock_file != NULL) {
        *sock_file = fdopen(*sockfd, "r");
        if (*sock_file == NULL) {
            perror("fdopen failed");
            close(*sockfd);
            return 1;
        }
    }

    if (port == CONTROLLER_PORT){
        char read_buf[BUF_SIZE];
        bytes = read(*sockfd, &read_buf, BUF_SIZE);
        printf("\nControl socket configured with success!\n");
    } else printf("\nData socket configured with success!\n");

    return 0;
}
```

```c
int term_B2(char **content) {
    if (!content || !sock_file2) return 1;

    char buffer[4096];
    size_t total = 0;
    *content = malloc(1);
    if (!*content) return 1;
    (*content)[0] = '\0';

    while (!feof(sock_file2)) {
        size_t bytes = fread(buffer, 1, sizeof(buffer), sock_file2);
        if (bytes > 0) {
            *content = realloc(*content, total + bytes + 1);
            memcpy(*content + total, buffer, bytes);
            total += bytes;
            (*content)[total] = '\0';
        }
    }

    fclose(sock_file1);
    fclose(sock_file2);
    close(sockfd1);
    close(sockfd2);

    return 0;
}

int term_A2(struct URL url){
    char *buf = NULL;
    bytes = asprintf(&buf, "RETR %s\r\n", url.path);

    // Send RETR command
    bytes = write(sockfd1, buf, strlen(buf));
    if (bytes <= 0) return 1;

    // Read and check RETR response
    read_ftp_response(sock_file1, &code);

    if (code != 150 && code != 125) {
        fprintf(stderr, "RETR command failed (expected 150 or 125, got %d)\n", code);
        free(buf);
        return 1;
    }
    printf("\nSent Retrieve request successfully!\n");

    // Cleanup
    free(buf);
    close(sockfd1);
    return 0;
}
```

```c
int term_B1(){
    uint16_t data_port = (256 * port1) + port2;
    char *ip = NULL;
    bytes = asprintf(&ip, "%d.%d.%d.%d", ip1, ip2, ip3, ip4);

    if (configure_socket(ip, data_port, &sockfd2, &sock_file2)) {
        return 1;
    }

    // Cleanup
    free(ip);
    return 0;
}

int term_A1(struct URL url) {
    if (configure_socket(url.ip, CONTROLLER_PORT, &sockfd1, &sock_file1)) {
        return 1;
    }

    char *buf1 = NULL, *buf2 = NULL;
    bytes = asprintf(&buf1, "USER %s\r\n", url.user);
    bytes = asprintf(&buf2, "PASS %s\r\n", url.password);
    char buf3[] = "PASV\r\n";

    // Send USER command
    bytes = write(sockfd1, buf1, strlen(buf1));
    if (bytes <= 0) return 1;

    // Read and check USER response
    read_ftp_response(sock_file1, &code);

    if (code != 331) {
        fprintf(stderr, "USER command failed (expected 331, got %d)\n", code);
        free(buf1);
        free(buf2);
        return 1;
    }

    // Send PASS command
    bytes = write(sockfd1, buf2, strlen(buf2));
    if (bytes <= 0) return 1;

    // Read and check PASS response
    read_ftp_response(sock_file1, &code);

    if (code != 230) {
        fprintf(stderr, "Login failed (expected 230, got %d)\n", code);
        free(buf1);
        free(buf2);
        return 1;
    }
    printf("Login successful!\n");
```

```c
    // Send PASV command
    bytes = write(sockfd1, buf3, strlen(buf3));
    if (bytes <= 0) return 1;

    // Read and check PASV response
    read_ftp_response(sock_file1, &code);

    if (code != 227) {
        fprintf(stderr, "PASV command failed (expected 227, got %d)\n", code);
        free(buf1);
        free(buf2);
        return 1;
    }

    // Parse PASV response
    printf("Entered Passive mode with success!\n");
    if (sscanf(line, "227 Entering Passive Mode (%d,%d,%d,%d,%d,%d)",
               &ip1, &ip2, &ip3, &ip4, &port1, &port2) == 6) {
        printf("Data connection: %d.%d.%d.%d:%d\n",
               ip1, ip2, ip3, ip4, port1 * 256 + port2);
    }

    // Cleanup
    free(buf1);
    free(buf2);

    return 0;
}
```

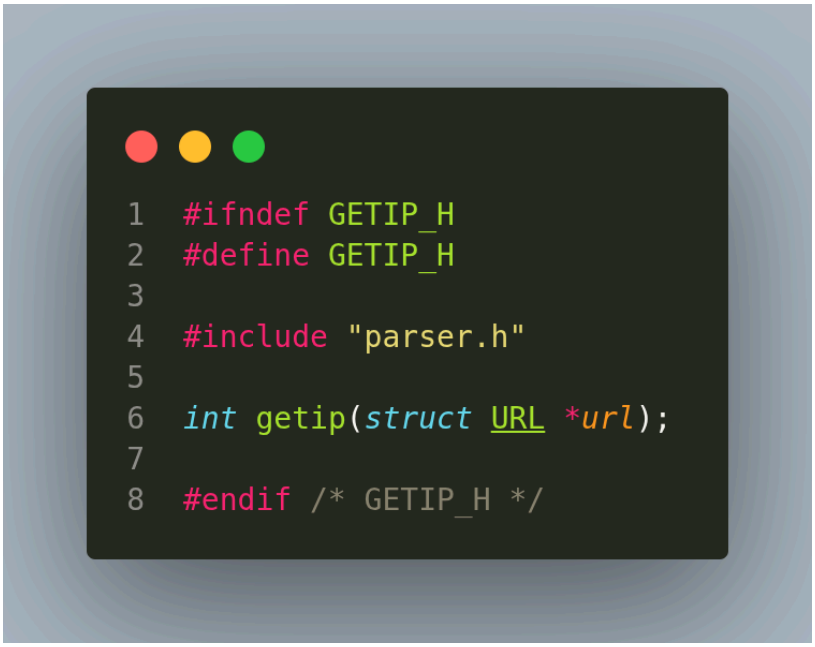
createfile.h

```c
#ifndef CREATE_FILE_H
#define CREATE_FILE_H

#include "parser.h"

int create_file(struct URL url, char *content);

#endif /* CREATE_FILE_H */
```

Createfile.c

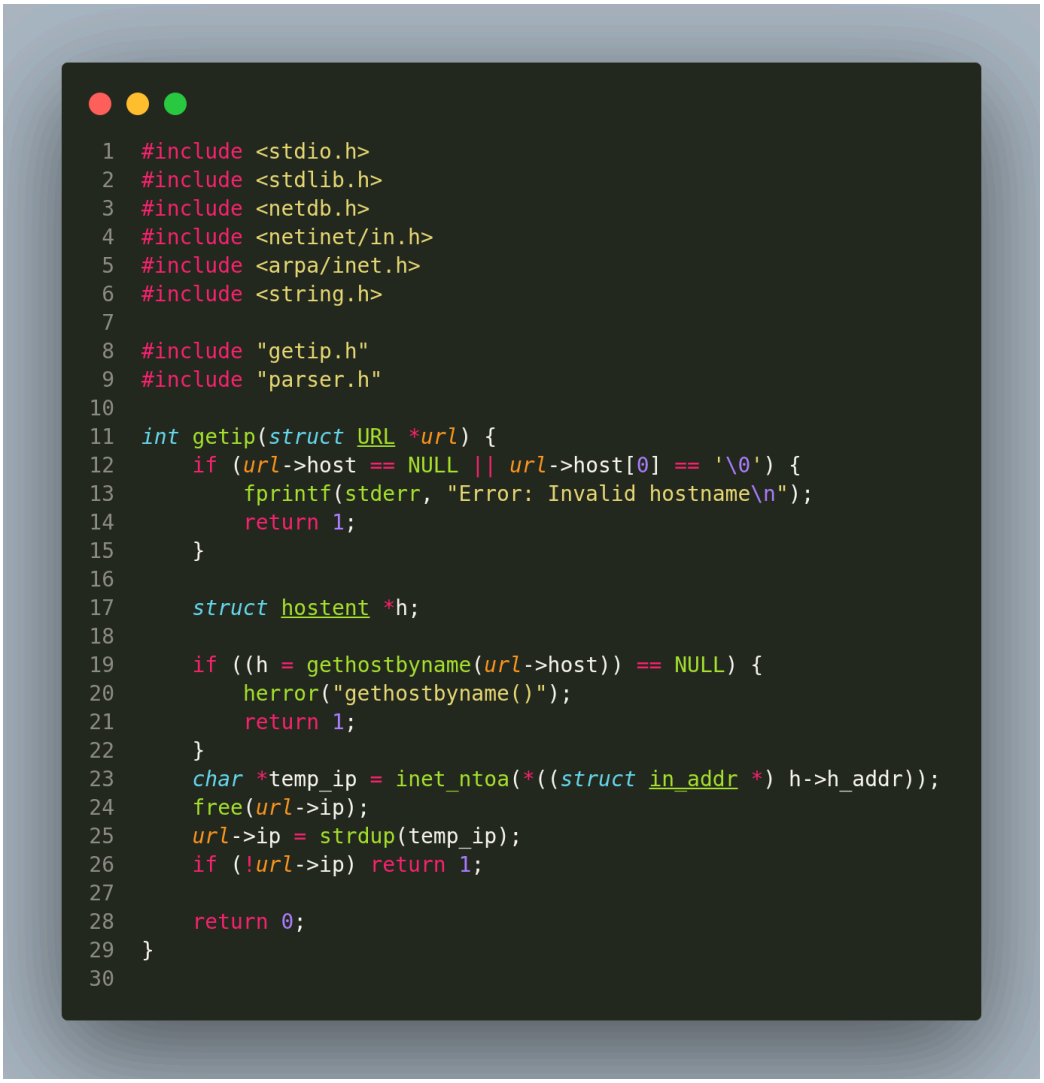
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "parser.h"

int create_file(struct URL url, char *content) {
     // Check for valid inputs
    if (content == NULL) {
        fprintf(stderr, "Error: Content is NULL\n");
        return 1;
    }

    char *filename = url.path;
    char *last_slash = strrchr(url.path, '/');

    if (last_slash != NULL) {
        filename = last_slash + 1;
        if (*filename == '\0') {
            filename = url.path;
        }
    }

    char filepath[512];
    snprintf(filepath, sizeof(filepath), "%s", filename);

    FILE *file = fopen(filepath, "w");
    if (file == NULL) {
        fprintf(stderr, "Error: Cannot create file '%s'\n", filepath);
        return 1;
    }

    // Write content to file
    size_t content_len = strlen(content);
    size_t bytes_written = fwrite(content, 1, content_len, file);

    if (bytes_written != content_len) {
        fprintf(stderr, "Error: Failed to write all content. Written %zu of %zu bytes\n",
                bytes_written, content_len);
        fclose(file);
        return 1;
    }

    // Close file
    if (fclose(file) != 0) {
        fprintf(stderr, "Error: Failed to close file '%s'\n", filepath);
        return 1;
    }
    printf("\nFile %s created with success!\n", filename);

    return 0;
}
```

getip.h



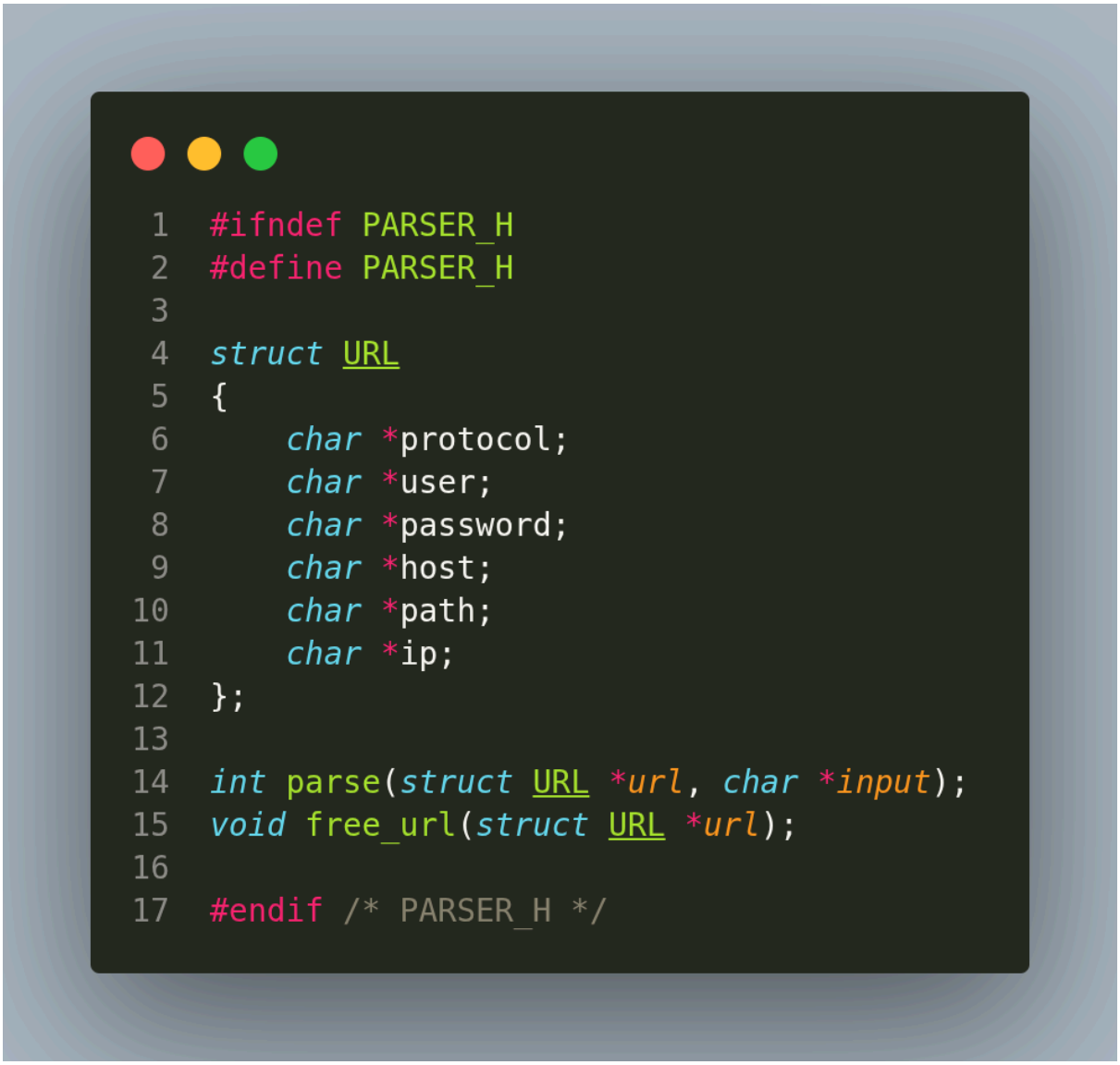
```h
#ifndef GETIP_H
#define GETIP_H

#include "parser.h"

int getip(struct URL *url);

#endif /* GETIP_H */
```

getip.c



```c
#include <stdio.h>
#include <stdlib.h>
#include <netdb.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>

#include "getip.h"
#include "parser.h"

int getip(struct URL *url) {
    if (url->host == NULL || url->host[0] == '\0') {
        fprintf(stderr, "Error: Invalid hostname\n");
        return 1;
    }

    struct hostent *h;

    if ((h = gethostbyname(url->host)) == NULL) {
        herror("gethostbyname()");
        return 1;
    }
    char *temp_ip = inet_ntoa(*((struct in_addr *) h->h_addr));
    free(url->ip);
    url->ip = strdup(temp_ip);
    if (!url->ip) return 1;

    return 0;
}

```

parser.h



```h
#ifndef PARSER_H
#define PARSER_H

struct URL
{
    char *protocol;
    char *user;
    char *password;
    char *host;
    char *path;
    char *ip;
};

int parse(struct URL *url, char *input);
void free_url(struct URL *url);

#endif /* PARSER_H */
```

parser.c

```c
#include "parser.h"

#include <string.h>

#include <stdlib.h>

#include <stdio.h>



int parse(struct URL *url, char *input) {

   url->protocol = NULL;
```

```c
    url->user = NULL;
    url->password = NULL;
    url->host = NULL;
    url->path = NULL;
    url->ip = NULL;

    if (input == NULL || strlen(input) == 0) {
        fprintf(stderr, "Error: Empty or NULL input string\n");
        return 1;
    }

    // Make a copy we can modify
    char *str = strdup(input);
    if (str == NULL) return 1;

    // Parse protocol
    char *proto_end = strstr(str, "://");
    if (proto_end == NULL) {
        fprintf(stderr, "Error: Invalid URL format - missing protocol separator\n");
        free(str);
        return 1;
    }
    *proto_end = '\0';
    url->protocol = strdup(str);
```

```
    // Check protocol
    if (strcmp(url->protocol, "ftp") != 0) {
        fprintf(stderr, "Error: Unsupported protocol\n");
        free(str);
        return 1;
    }

    char *ptr = proto_end + 3;

    // First, check if there's an '@' to determine if we have
credentials
    char *at_ptr = strchr(ptr, '@');

    if (at_ptr != NULL) {
        // We have '@', so parse credentials
        *at_ptr = '\0';  // Split at '@'

        char *credentials = ptr;
        char *colon_ptr = strchr(credentials, ':');

        if (colon_ptr != NULL) {
            // We have "user:password" format
            *colon_ptr = '\0';
            url->user = strdup(credentials);
```

```
            url->password = strdup(colon_ptr + 1);



            // Check if either user or password is empty

            if ((url->user && strlen(url->user) == 0) ||

                (url->password && strlen(url->password) == 0)) {

                fprintf(stderr, "Error: Both user and password must be
specified when using credentials\n");

                free(str);

                return 1;

            }

        } else {

            // We have only user (no password) - ERROR

            fprintf(stderr, "Error: Password missing - both user and
password must be specified\n");

            free(str);

            return 1;

        }



        ptr = at_ptr + 1;  // Move past '@'

    } else {

        // No '@' found, check if there's a colon that might be confused
as credentials

        char *colon_in_path = strchr(ptr, ':');

        if (colon_in_path != NULL) {

            // There's a colon but no '@' - this is an error

            // Example:
"ftp://anonymous:anonymousftp.bit.nl/speedtest/100mb.bin"
```

```
            fprintf(stderr, "Error: Invalid URL format - missing '@'
separator\n");
            free(str);
            return 1;
        }

        // No credentials and no confusing colon - use anonymous
        url->user = strdup("anonymous");
        url->password = strdup("anonymous");
    }

    // Parse host and path
    char *slash_ptr = strchr(ptr, '/');
    if (slash_ptr != NULL) {
        *slash_ptr = '\0';
        url->host = strdup(ptr);
        url->path = strdup(slash_ptr + 1);
    } else {
        url->host = strdup(ptr);
        url->path = strdup("/");
    }

    // Validate host
    if (url->host == NULL || strlen(url->host) == 0) {
        fprintf(stderr, "Error: No host specified\n");
```

```
        free(str);
        return 1;
    }

    // Validate path is not empty (except for root)
    if (url->path && strlen(url->path) == 0) {
        free(url->path);
        url->path = strdup("/");
    }

    free(str);
    return 0;
}

void free_url(struct URL *url) {
    if (url->protocol) free(url->protocol);
    if (url->user) free(url->user);
    if (url->password) free(url->password);
    if (url->host) free(url->host);
    if (url->path) free(url->path);
    if (url->ip) free(url->ip);
}
```