Redes de Computadores

2nd Pratical Work

Guilherme Almeida - 202008866
Tiago Barbosa - 202004926
Nicolae Munteanu - 202202842

December 20, 2022

Index

1 Summary

This work was done in the scope of the Computer Networks Curricular Unit of the Informatics and Computer Engineering course at FEUP. In this second laboratorial work the two objectives were to build an application capable of transferring files using the FTP protocol and the configuration and study of a computer network.

All the objectives of the work were achieved, since it was possible to use this application to transfer several files without any loss of data both in anonymous mode and in servers with credentials and it was possible to create the network between the 3 computers and connect them to the Internet.

2 Introduction

The main objectives of this work were the creation of an application capable of using the FTP protocol for file transfer and the configuration and study of a network in the FEUP laboratories.

The purpose of this report is to describe the execution of this work in order to show the strategies adopted, choices made and difficulties encountered, as well as to provide a general review of the work done. The report is thus divided into the following sections :

- **Part 1 - Download application**
  - Architecture of the download application
  - Report of a successful download
- **Part 2 – Network configuration and analysis**
  - For each experiment:
    - Network architecture, experiment objectives, main configuration commands
    - Analysis of the logs captured that are relevant for the learning objectives
- **Conclusion**
- **References**
- **Annexes: code of the download application, configuration commands, logs captured**

3 Part 1 - Download application

The first part of this work was the development of an application to download a file from an FTP server that received as argument a link in the following format :

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

This application implements FTP application protocol, as described in RFC959 and adopts URL syntax, as described in RFC1738.

This download application focused on the client – server concept and its peculiarities in TCP/IP to implement the FTP protocol to download a single file. The client-server model relies on the following points:

- A device or application (the client) sends requests to another device or program (the server), which processes those requests and may return a response. This is known as the client-server model of network communication.
- A protocol like TCP/IP is used by the client and server to create a connection, guarantee the data is transmitted reliably, and control how long the session will last.

3.1 Architecture of the download application

We can divide our application on three parts: parsing of the url, logic of the FTP protocol and logic of the download overall process.

3.1.1 Parsing the URL

```
typedef struct{
    char user[128];
    char password[128];
    char host[128];
    char path[128];
    char ip[128];
    char filename[256];
    char host_name[128];
} URL;

int getFileName(char* path,char* filename);
int getIp(char* adress,URL *url);
int parseUrl(char* text, URL *url);
```

Listing 1: url.h

We created an URL struct to better represent all the parts of the input url.

The function getFileName given the path part of the url gives the filename so we can create a file with the same name to save the content of the download.

The function getIp given the host part of the url uses the function gethostbyname(char * address) that using DNS transforms the human readable host to the correspondent IP so that we can use it in the Berkeley sockets.

The function parseUrl given the input url breaks it into all its parts using strtok function and the getFileName and getIp functions. When not prompted with user and password the user filed will be "anonymous" and the password will be "something". After this steps we will have the URL struct completed and ready to be used.

3.1.2 Logic of the FTP protocol

```
int startSocket(int * sockfd, char* ip, int port);
int readResponse(FILE* socketResponse ,char* response,size_t size);
int readIp_Port(FILE* socketResponse ,char* response,size_t size,char* ip,int * port);
int sendCommand(int sockfd, char* command);
int saveFile(char * filename, int sockfd,int fileSize);
void printProgressBar(float current, float total);
int getFileSize(char * response);
```

Listing 2: ftp.h

The startSocket function creates a TCP socket and connects it to the server with the input IP in the input port using the socket() and connect() function.

The readResponse function reads the server response to the commands given by the user and displays it in the terminal to allow a better understanding of what is happening during the FTP server connection phase. If the response code is 550 or 530 (error) it finishes the program indicating the error.

The readIp_Port function behaves in the same way as the readResponse function but is only used to read the response from the pasv command so it can parse that response into and IP and port to be used to open another socket to make the transfer of the file.

The sendCommand function is used to send FTP commands to the server and prints them to simulate a real input from the user.

The saveFile function is used after the retr FTP command to create a file with the same name as the downloaded one and save the content received from the socket connected to the server to this new file thus completing the transfer.

The printProgressBar function is used by the saveFile function to print a progress bar to represent the transfer of the file to allow the user to better know how long the transfer will take.

The getFileSize function is used to know the size in bytes of the transfer file ,that is presented in brackets after the retr FTP command, to be used by the printProgressBar function to know the percentage already downloaded.

3.1.3 Logic of the download process

```
1 int download(char* ftp_link);
```

Listing 3: download.h

The download function is the one called by main with the url string. This function starts by creating an URL struct and then parse the url string into this struct. Then it starts a connection to FTP server with sockets. Then it reads the initial message from the server and sends the user and password reading the respective response from the server to this commands. If the login is successful, it sends the pasv command and reads it response storing the new IP and port. Using this data it opens another socket where the file content will be dumped during the transfer. In the first socket, it send the retr FTP command with the file path and after creating the new file it starts copying the data in the second socket into this file while printing a progress bar to represent the transfer status. When the transfer is complete it sends the quit FTP command and closes both sockets ending the program.

If in any of the above steps there is an error the program terminates earlier indicating an error to the user.

3.2 Report of a successful download

First starting by a transfer with both user and password required for the transfer we tried to download a video from the ftp server "netlab1.fe.up.pt". We will be downloading the crab.mp4 file from files directory and use the following credentials:

- user: rcom
- password: rcom

```
tiagobarbosa05@MBP-de-Tiago-2 bin % ./download ftp://rcom:rcom@netlab1.fe.up.pt/files/crab.mp4
220 Welcome to netlab-FTP server
user rcom
331 Please specify the password.
pass rcom
230 Login successful.
pasv
227 Entering Passive Mode (192,168,109,136,166,74).
ip: 192.168.109.136    port: 42570
retr files/crab.mp4
150 Opening BINARY mode data connection for files/crab.mp4 (88123184 bytes).
Press ENTER key to Continue
```

After the user press the ENTER key the download will proceed:

```
34.06% [#################                                  ]
```

```
100.00% [##################################################]
226 Transfer complete.
quit%
tiagobarbosa05@MBP-de-Tiago-2 bin %
```

Even if the internet connection is interrupted the download will also stop , which can be seen by the progress bar not moving, and will continue when the connection is resumed.

If the login credentials are wrong the program will prompt an error and prematurely end.

```
tiagobarbosa05@MBP-de-Tiago-2 bin % ./download ftp://rcom:rcom2@netlab1.fe.up.pt/files/crab.mp4
220 Welcome to netlab-FTP server
user rcom
331 Please specify the password.
pass rcom2
530 Login incorrect.
Command error
tiagobarbosa05@MBP-de-Tiago-2 bin %
```

Then we to test the anonymous transferring we tried to download a txt file from the ftp server "ftp.up.pt". We will be downloading the timestamp.txt file without giving any credentials:

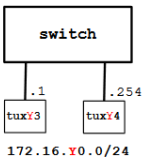
```
tiagobarbosa05@MBP-de-Tiago-2 bin % ./download ftp://ftp.up.pt/pub/kodi/timestamp.txt
220-Welcome to the University of Porto's mirror archive (mirrors.up.pt)
220----------------------------------------------------------------------
220-
220-All connections and transfers are logged. The max number of connections is 200.
220-
220-For more information please visit our website: http://mirrors.up.pt/
220-Questions and comments can be sent to mirrors@uporto.pt
220-
220-
220
user anonymous
331 Please specify the password.
pass something
230 Login successful.
pasv
227 Entering Passive Mode (193,137,29,15,199,163).
ip: 193.137.29.15    port: 51107
retr pub/kodi/timestamp.txt
150 Opening BINARY mode data connection for pub/kodi/timestamp.txt (11 bytes).
Press ENTER key to Continue
```

```
100.00% [##################################################]
226 Transfer complete.
quit%
tiagobarbosa05@MBP-de-Tiago-2 bin %
```

4 Part 2 – Network configuration and analysis

You can see the LINUX and MIKROTIK configuration commands by going to the annexes part 2 where all the commands are listed for each one of the experiences.

4.1 Experience 1 - Configuring an IP Network



This first experience had the objective of teaching what ARP packets are and what they're used for. This was done by connecting tux 3 and tux 4 to a switch and then analyzing what happens when a ping command is executed, before and after deleting the ARP table.

1) What are the ARP packets and what are they used for?

Address Resolution Protocol packets are messages used to obtain the MAC address of a device that corresponds to a specific IP address in a network.

2) What are the MAC and IP addresses of ARP packets and why?

ARP packets have the MAC and IP addresses of both the sender and the target. When sending an ARP Request, all fields are filled out except the one containing the MAC address of the target, which is ignored and left all zeros. As a response, an ARP Reply will arrive, having as target the device that sent the request and providing them the MAC address that was requested.

1	0.000000000	HewlettP_61:2c:54	Broadcast	ARP	42 Who has 172.16.50.254? Tell 172.16.50.1
2	0.000101269	HewlettP_19:09:5c	HewlettP_61:2c:54	ARP	60 172.16.50.254 is at 00:22:64:19:09:5c
3	0.000108672	172.16.50.1	172.16.50.254	ICMP	98 Echo (ping) request id=0x321c, seq=1/256,
4	0.000195973	172.16.50.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x321c, seq=1/256,

In the lab experience, 172.16.50.1 looked for the MAC address of 172.16.50.254. The ARP Reply returned the MAC address 00:22:64:19:09:5c.

3) What packets does the ping command generate?

The ping command is used to test connectivity between two devices in a network. It generates ICMP(*Internet Control Message Protocol*) packets - more specifically it sends ICMP requests and waits for ICMP replies.

4) What are the MAC and IP addresses of the ping packets?

The MAC and IP addresses in the ping packets contain the MAC and IP of the sender and the MAC and IP of the receiver.

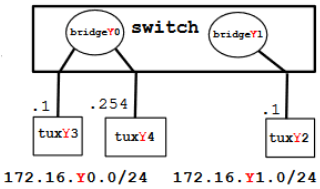
5) How to determine if a receiving Ethernet frame is ARP, IP, ICMP?

One can detect what a receiving Ethernet frame is by looking at the EtherType field. This field has 16 bits and if it's set to 0x0806, the frame contains an ARP packet; if it's set to 0x0800, it's for an IP packet, and if it's set to 0x01 it's for an ICMP packet

6) How to determine the length of a receiving frame? The length of a receiving frame is stored in a field of the header, and it indicates the number of bytes that the frame will be made up of.

7) What is the loopback interface and why is it important? The loopback interface is a virtual network interface used to send data to the same device that the data was sent form. It's very important for testing and debugging.

4.2 Experience 2 - Implement two bridges in a switch



The objective of this experience was to create two bridges in a switch and observe the differences after the change, especially regarding the broadcast domain.

1) How to configure bridgeY0?

In order to configure Bridge50, we created the bridge in the Mikrotik Switch interface; we then removed the interested ports from the default bridge, and added them to Bridge50, being careful to assign the correct interfaces. The specific code is in the annexes.

2) How many broadcast domains are there? How can you conclude it from the logs?

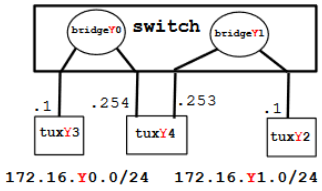
There are two broadcast domains. The first is made up of tux3 and tux4, and the second is made up of just tux2. This is clear when looking at the logs because, when pinging broadcast in tux3, there is only a reply from tux4.

```
23 26.043792121  172.16.50.1      172.16.50.255     ICMP     98 Echo (ping) request
24 26.043937180  172.16.50.254    172.16.50.1       ICMP     98 Echo (ping) reply
25 27.067797867  172.16.50.1      172.16.50.255     ICMP     98 Echo (ping) request
26 27.067942577  172.16.50.254    172.16.50.1       ICMP     98 Echo (ping) reply
```

But, when pinging broadcast in tux2, there is no reply, so that's another broadcast domain.

```
18 16.451383024  172.16.51.0      172.16.51.255     ICMP     98 Echo (ping) request
19 17.475366723  172.16.51.0      172.16.51.255     ICMP     98 Echo (ping) request
```

4.3 Experience 3 - Configure a Router in Linux



This experience had the objective of transforming one of the tuxes into a router, to then see what happens in the rest of the network as a consequence.

1) What routes are there in the tuxes? What are their meaning?

In tux3, whenever we want to go to an address in the 172.16.51.0/24 group, we'll use tux4 as a gateway, which has an IP of 172.16.50.254.

In tux2, whenever we want to go to an address in the 172.15.50.0/24 group, we'll also use tux4 as a gateway, which in this case has an IP of 172.16.51.253.

2) What information does an entry of the forwarding table contain?

Destination address: This is the address that the packet is intended for.

Next hop: This is the address of the next device that the packet should be forwarded to in order to reach its destination.

Interface: This is the physical port (or logical interface) on the router or switch that the packet should be sent out of in order to reach the next hop.

Metric: This is a value that is used to determine the best path to the destination. It may be based on factors such as the number of hops, the bandwidth of the link, or the cost of using the link.

3) What ARP messages, and associated MAC addresses, are observed and why?

After cleaning the ARP tables and trying to ping tux2 from tux3, passing through tux4 as a router, we can see a series of ARP messages.

Tux3 gets the MAC address of the router.

```
 7 11.167349953  HewlettP_61:2c:54   Broadcast           ARP         60 Who has 172.16.50.254? Tell 172.16.50.1
 8 11.167378448  HewlettP_19:09:5c   HewlettP_61:2c:54   ARP         42 172.16.50.254 is at 00:22:64:19:09:5c
```

The router gets the MAC address of tux2.

```
 7 10.166398536  KYE_25:21:9e        Broadcast           ARP         42 Who has 172.16.51.1? Tell 172.16.51.253
 8 10.166539198  HewlettP_5a:7c:e7   KYE_25:21:9e        ARP         60 172.16.51.1 is at 00:21:5a:5a:7c:e7
```

Tux2 gets the MAC address of the router.

```
24 16.420539979  HewlettP_19:09:5c   HewlettP_61:2c:54   ARP         42 Who has 172.16.50.1? Tell 172.16.50.254
25 16.420678546  HewlettP_61:2c:54   HewlettP_19:09:5c   ARP         60 172.16.50.1 is at 00:21:5a:61:2c:54
```

The router gets the MAC address of tux3.

```
21 15.234400552  HewlettP_5a:7c:e7   KYE_25:21:9e        ARP         60 Who has 172.16.51.253? Tell 172.16.51.1
22 15.234422622  KYE_25:21:9e        HewlettP_5a:7c:e7   ARP         42 172.16.51.253 is at 00:c0:df:25:21:9e
```

This is so the ping can reach destination and then come back.

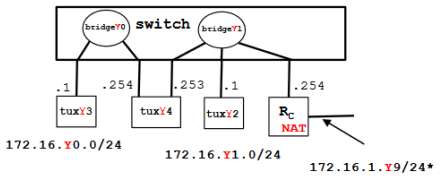
4) What ICMP packets are observed and why?

There are ICMP Reply and ICMP Request packets observed in the tux of the router; this is because the tux redirects the ping message so that it can reach destination.

5) What are the IP and MAC addresses associated to ICMP packets and why?

The IP and MAC addresses are the ones corresponding to the Sender and to the Destination; that's because the router does not change the structure of the ICMP packet while routing it to the destination, so it will just be IP and MAC of tux2 and tux3.

4.4 Experience 4 - Configure a Commercial Router and Implement NAT



The main goal of this experience was to use and understand a Commercial Router and explore the functionality of NAT. In order to accomplish this experience firstly we needed to connect the router to the lab's network and to bridge51.

1) How to configure a static route in a commercial router?

A static route can be used to specify a particular path for traffic to take when commuting with a specific destination. To configure a static route one needs to use the following command:

```
/ip route add dst-address=172.16.50.0/24 gateway=172.16.51.253
```

Specifying the destination's address and the used gateway.

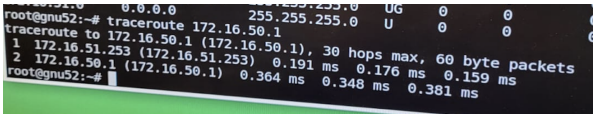
2) What are the paths followed by the packets in the experiments carried out and why?

In the first experience we needed to understand the path of the packets in different scenarios.

Firstly it was asked to remove the route of tux52 to 172.16.50.0/24 via tux54. When pinging tux53 from tux52 we observe that the packets go through the commercial router and then to tux53, due to removing the route beforehand. After re-adding the removed route and pinging tux53 from tux52, the packets don't need to go through the commercial router and take the default path to tux54.

With the command traceroute we can check the path in both scenarios:

```
root@gnu52:~# traceroute 172.16.50.1
traceroute to 172.16.50.1 (172.16.50.1), 30 hops max, 60 byte packets
 1  172.16.51.254 (172.16.51.254)  0.201 ms  0.183 ms  0.201 ms
 2  172.16.51.253 (172.16.51.253)  0.332 ms  0.318 ms  0.327 ms
 3  172.16.50.1 (172.16.50.1)  0.570 ms  0.558 ms  0.536 ms
```



In a third scenario, which consisted in enabling the ICMP redirects and once again removing the route from tux52 to tux53, we observed that only one initial hop to 172.16.51.254 (commercial router) is done and afterwards it knows where to go because of the router.

In a second experience, with the initial setup active, we pinged the router from tux53 successfully, but after disabling NAT the connection failed because tux53 doesn't have any way to connect to the internet.

3) How to configure NAT in a commercial router?

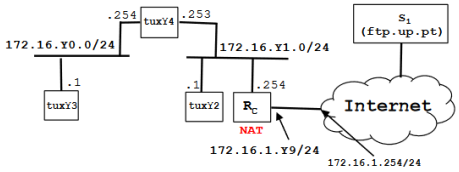
In order to configure NAT in a commercial router one needs to use the following command:

```
/ip firewall nat add chain=srcnat action=masquerade out-interface=ether1
```

4) What does NAT do?

NAT (Network Address Translation) is a method used by routers to allow devices on a private network to access the internet using a single, public IP address. The use of NAT helps to conserve the limited number of available IP addresses and allows devices on a private network to communicate with devices on the internet without the need for a unique, publicly-routable IP address for each device.

4.5 Experience 5 - DNS



In this experience it was asked to configure the DNS service in each of the tuxes and to analyze the DNS related packets when pinging a hostname in the internet.

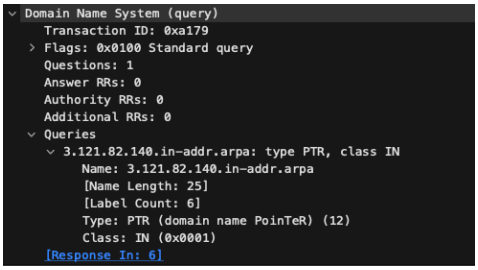
1) How to configure the DNS service at an host?

In order to configure the DNS service we need to access the file */etc/resolv/conf.* and add the IP address and server name. In this experience we used the *services.netlab.fe.up.pt* server and *172.16.2.1 address.*

2)What packets are exchanged by DNS and what information is transported?

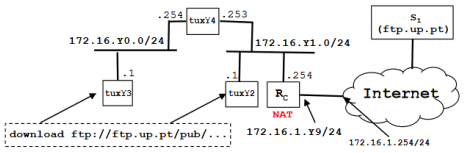
A DNS (Domain Name System) is a distributed network of servers that translates human-readable domain names into numerical IP addresses that computers can understand.

With this in mind the packets exchanged by DNS are query and reply messages, both with the same messages format. The message itself has an header section that includes an identification, flags and the number of questions and resource records. The query contains the domain name that is being queried and the reply contains the answers to the query along with additional information such as the IP address of the server.



```
Domain Name System (response)
    Transaction ID: 0xa179
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  Queries
    3.121.82.140.in-addr.arpa: type PTR, class IN
        Name: 3.121.82.140.in-addr.arpa
        [Name Length: 25]
        [Label Count: 6]
        Type: PTR (domain name PoinTeR) (12)
        Class: IN (0x0001)
  Answers
    > 3.121.82.140.in-addr.arpa: type PTR, class IN, lb-140-82-121-3-fra.github.com
```

4.6 Experience 6 - TCP Connections



In this experience the main goal was to run and test our FTP application, thus understanding the functionalities of a FTP protocol and TCP connections and the respective mechanisms.

1) How many TCP connections are opened by your ftp application?

Two TCP connections are opened in our FTP application: one for control purposes and another for transferring data.

The control connection is used to send commands and receive responses between the client and the server. The data connection is used to transfer files between the client and the server.

2) In what connection is transported the FTP control information?

The control information is transported in the connection first mentioned in the previous question.

3) What are the phases of a TCP connection?

The three phases of a TCP connection are the following:

- Connection Establishment - this phase involves an exchange of messages between the two hosts to establish a connection.
- Data Transfer - once the connections is established, the two hosts can transfer data, this data is divided into segments and sent using a series of packets.
- Connection Termination - when either host wants to terminate the connection, there's an exchange of messages similar to the connection establishment, and the connection is terminated.

4) How does the ARQ TCP mechanism work? What are the relevant TCP fields? What relevant information can be observed in the logs?

ARQ (Automatic Repeat Request) is a type of error control mechanism that is used to ensure reliable data transmission over a network. It works by detecting errors in the data that is being transmitted, and re-transmitting any data that is not received correctly.

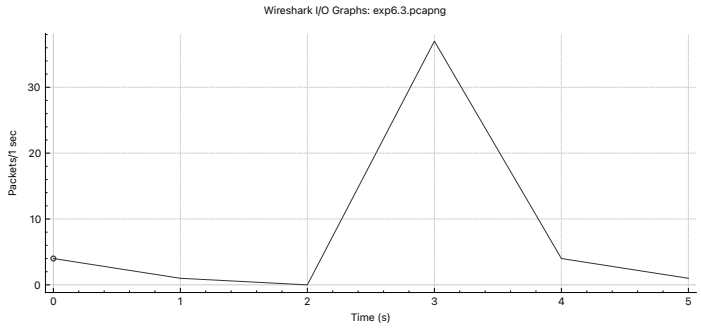
When a sender wants to transmit data to a receiver, it sends a packet to the receiver. The packet includes a sequence number that identifies the order in which the data was transmitted, and the acknowledgment number, that represents the sequence number of the next packet that the receiver is hoping to receive. The receiver receives the packet and checks for errors. If it's received correctly a ACK messages is sent to the sender. If the receiver detects an error it does not send a ACK, instead it waits for the sender to re-transmit the packet. If the sender does not receive the ACK message within a certain timeout, it re-transmits the packet. The process continues until all of the data has been transmitted and acknowledge by the receiver.

5) How does the TCP congestion control mechanism work? What are the relevant fields. How did the throughput of the data connection evolve along the time? Is it according the TCP congestion control mechanism?

The congestion mechanism of TCP helps prevent network overload and ensures that data is transmitted at a rate that the network can handle. It accomplishes this by adjusting the size of the congestion window, which is a measure of the amount of data that can be transmitted before receiving an acknowledgement from the receiving device.

The main field in the TCP header is the *window size*, this field specifies the maximum amount of data that can be transmitted before an acknowledgment is received. If it's receiving information faster than it can process, a *window update* message is sent.

When analysing the graph generated with the Wireshark log, we observe a notable increase in the throughput as expected. As the congestion control mechanism is triggered and the sender reduces its transmission rate, the throughput also decreases. This is to allow the congestion in the network to dissipate, so that the connection can once again operate at a higher transmission rate.



6) Is the throughput of a TCP data connections disturbed by the appearance of a second TCP connection? How?

When analysing the graphs generated with the Wireshark log, we observe a slightly decrease in the throughput while the second TCP connection was active. Preferably, the two connections should have downloaded files with similar sizes, which was not the case. With this, it would be easier to check how the network capacity would split the network in two.

Basically we can see how tux3 has to decrease the throughput (around the 15 seconds mark) in order for tux2 to increase his throughput and complete the download.

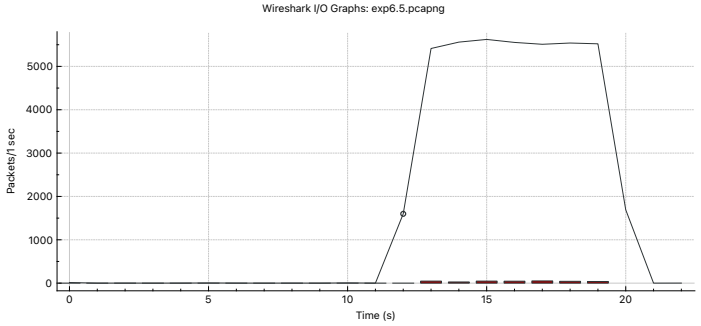


Figure 1: Tux3

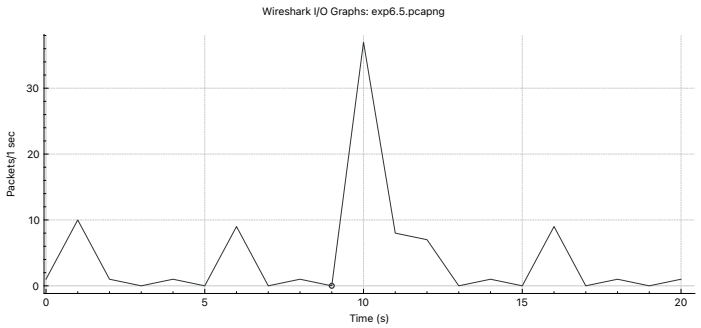


Figure 2: Tux2

5 Conclusion

The practical experience done in the laboratory managed to concretely solidify our understanding of the Computer Networks course. The first part of the practical work gave us insight in how protocols work and how they were written and designed. It also served to deepen our knowledge of Berkeley socket handling in C, FTP - File Transfer Protocol and Client-Server Communication Mode (RFC 959). The second part permitted us to experience the process of setting up a network, testing for errors and getting a real feedback on how the network architecture is built.

We believe the experience was very formative and we are satisfied with our work because we were able to achieve all the proposed objectives.

Note: The maximum number of pages allowed in the report statement was slightly exceeded due to the size and positioning of the inserted images. These images could have been placed only in the annexes, but for visual reasons we understood that having the images in the place of the text to which they refer would facilitate the reading of the report. Sometimes it was also necessary to make page breaks so as not to ruin the formatting of the content. Thus, in terms of text, this report does not exceed the maximum 8 pages allowed.

6 References

[1] Slides from the RC moodle page. https://moodle.up.pt/course/view.php?id=2092

[2] https://www.heavy.ai/technical-glossary/client-server

[3] https://www.rfc-editor.org/rfc/rfc959

[4] https://www.rfc-editor.org/rfc/rfc1738

[5] https://www.geeksforgeeks.org/socket-programming-cc/

7 Annexes Part 1

main.c

```c
#include "download.h"


int main(int argc, char** argv){
    if(argc != 2){
        printf("usage : download ftp://[<user>:<password>@]<host>/<url-path>");
        return -1;
    }

    download(argv[1]);

    return 0;

}
```

download.c

```c
#include "download.h"

int download(char* ftp_link){

    URL url;
    char urlcpy[256];
    char command[256];
    int sockfd;
    int sockfd_b;

    strcpy(urlcpy,ftp_link);

    if(parseUrl(urlcpy,&url)!=0){
        printf("Error parsing URL\n");
        return -1;
    }

    //printf("user:%s\npassword:%s\nhost:%s\npath:%s\nip:%s\nfilename:%s\nhost_name:%s
    \n",url.user,url.password,url.host,url.path,url.ip,url.filename,url.host_name);

    if(startSocket(&sockfd,url.ip,21) !=0){
        printf("Error starting socket\n");
        return -1;
    }

    FILE* socketResponse = fdopen(sockfd,"r");
    char response[512];

    if(readResponse(socketResponse,response,512)!=0){
        return -1;
    }

    sprintf(command, "user %s\n",url.user); //pode faltar \r em windows
    if(sendCommand(sockfd,command)!=0){
        return -1;
    }
    if(readResponse(socketResponse,response,512)!=0){
        return -1;
    }
    sprintf(command, "pass %s\n",url.password);
    if(sendCommand(sockfd,command)!=0){
        return -1;
    }
    if(readResponse(socketResponse,response,512)!=0){
        return -1;
    }

    char ip[32];
    int port;

    sprintf(command, "pasv\n");
    if(sendCommand(sockfd,command)!=0){
        return -1;
    }
    if(readIp_Port(socketResponse,response,512,ip,&port)!=0){
        return -1;
    }

    printf("ip: %s    port: %d\n",ip,port);

    if(startSocket(&sockfd_b,ip,port) !=0){
        printf("Error starting socket\n");
        return -1;
    }

    sprintf(command,"retr %s\n",url.path);
    if(sendCommand(sockfd,command)!=0){
        return -1;
    }
    if(readResponse(socketResponse,response,512)!=0){
```

```
70          return -1;
71      }
72
73      int fileSize = getFileSize(response);
74
75      printf("Press ENTER key to Continue\n");
76      getchar();
77
78      if(saveFile(url.filename,sockfd_b,fileSize)!=0){
79          return -1;
80      }
81
82      sprintf(command,"quit");
83      if(readResponse(socketResponse,response,512)!=0){
84          return -1;
85      }
86
87      if(sendCommand(sockfd,command)!=0){
88          return -1;
89      }
90
91      if (close(sockfd)<0) {
92          perror("close()");
93          exit(-1);
94      }
95
96      if (close(sockfd_b)<0) {
97          perror("close()");
98          exit(-1);
99      }
100
101
102     return 0;
103
104 }
```

ftp.c

```c
#include "ftp.h"

int startSocket(int * sockfd, char* ip, int port){

    struct sockaddr_in server_addr;

    /*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) {
        printf("Error opening TCP socket!\n");
        return -1;
    }
    /*connect to the server*/
    if (connect(*sockfd,
                (struct sockaddr *) &server_addr,
                sizeof(server_addr)) < 0) {
        printf("Error connecting to the server!\n");
        return -1;
    }

    return 0;
}

int readResponse(FILE* socketResponse ,char* response,size_t size){

    while(fgets(response,size,socketResponse)){
        printf("%s",response);
        if(response[3] == ' '){
            int code = atoi(response);
            if (code == 550 || code == 530)
            {
                printf("Command error\n");
                return -1;
            }
            break;
        }
    }
    return 0;
}

int readIp_Port(FILE* socketResponse ,char* response,size_t size,char* ip,int * port){

    while(fgets(response,size,socketResponse)){
        printf("%s",response);
        if(response[3] == ' '){
            break;
        }
    }

    strtok(response,"(");
    char* ip_1 = strtok(NULL, ",");
    char* ip_2 = strtok(NULL, ",");
    char* ip_3 = strtok(NULL, ",");
    char* ip_4 = strtok(NULL, ",");
    char* port_1 = strtok(NULL, ",");
    char* port_2 = strtok(NULL, ")");

    sprintf(ip,"%s.%s.%s.%s",ip_1,ip_2,ip_3,ip_4);
    *port = atoi(port_1)*256 + atoi(port_2);

    return 0;
}

```

```
int sendCommand(int sockfd, char* command){

    size_t bytes = write(sockfd, command, strlen(command));
    if (bytes > 0){
        printf("%s", command);
    }
    else {
        perror("write()");
        return -1;
    }

    return 0;
}


void printProgressBar(float current, float total)
{
    usleep(500); // to better visualize progress bar
    float percentage = 100.0 * current / total;

    printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
    printf("\n%.2f%% [", percentage);

    int len = 50;
    int pos = percentage * len / 100.0;
    for (int i = 0; i < len; i++)
        i <= pos ? printf("#") : printf(" ");

    printf("]\n");
}

int getFileSize(char * response){
    strtok(response,"(");
    char* size = strtok(NULL, " ");
    return atoi(size);
}

int saveFile(char * filename, int sockfd,int fileSize){
    static int currentsize = 0;
    int filefd = open(filename,O_WRONLY| O_CREAT,0777);
    if(filefd < 0){
        perror("open file");
        return -1;
    }

    size_t bytes;
    char buffer[2048];

    do{
        bytes = read(sockfd,buffer,2048);
        if(bytes > 0){
            currentsize+=bytes;
            printProgressBar(currentsize,fileSize);
            write(filefd,buffer,bytes);
        }
    }while(bytes!=0);

    close(filefd);

    return 0;
}
```

url.c

```c
#include "url.h"

int getIp(char* adress,URL *url){
    struct hostent *h;

    if ((h = gethostbyname(adress)) == NULL) {
        printf("Error getting host by name!\n");
        return -1;
    }

    strcpy(url->ip,inet_ntoa(*((struct in_addr *) h->h_addr)));
    strcpy(url->host_name,h->h_name);

    return 0;
}

int getFileName(char* path,char* filename){
  char strtokpath[256];
  strcpy(strtokpath, path);
  char* token = strtok(strtokpath, "/");
  while( token != NULL ) {
    strcpy(filename, token);
    token = strtok(NULL, "/");
  }
  return 0;
}

int parseUrl(char* text, URL *url){


    char* ftp = strtok(text,"/");  //ftp
    char* usrpasshost = strtok(NULL,"/"); //[<user>:<password>@]<host>
    char* path = strtok(NULL,""); //<url-path>
    strcpy(url->path,path);

    if(strcmp(ftp,"ftp:")!=0){
        printf("Error: not using ftp!\n");
        return 1;
    }


    char* user = strtok(usrpasshost,":");
    char* pass = strtok(NULL,"@");

    if (pass == NULL ){
        user = "anonymous";
        pass = "something";
        strcpy(url->host,usrpasshost);
    }
    else{
        strcpy(url->host,strtok(NULL,""));
    }

    strcpy(url->user,user);
    strcpy(url->password,pass);

    if(getIp(url->host,url)!=0){
        printf("Error getting ip\n");
        return -1;
    }

    if (getFileName(url->path,url->filename) != 0){
        printf("Error: getFileName()\n");
        return -1;
  }

    return 0;
}
```

8 Annexes Part 2

Experience 1

```
GNU3:
ifconfig eth0 down
ifconfig eth0 up
ifconfig eth0 172.16.50.1/24
IP - 172.16.50.1
MAC - 00:21:5a:61:2c:54

arp -a


GNU4:
ifconfig eth0 down
ifconfig eth0 up
ifconfig eth0 172.16.50.254/24
IP - 172.16.50.254
MAC - 00:22:64:19:09:5c
```

Experience 2

```
GNU 2:
ifconfig eth0 down
ifconfig eth0 up
ifconfig eth0 172.16.51.1/24
ifconfig para ver IP MAC

IP - 172.16.51.1
MAC - 00:21:5a:5a:7c:e7

MIKROTIK SWITCH:
/interface bridge add name=bridge50
/interface bridge add name=bridge51


/interface bridge port remove [find interface=ether8]
/interface bridge port remove [find interface=ether12]
/interface bridge port remove [find interface=ether18]

/interface bridge port add bridge=bridge50 interface=ether8
/interface bridge port add bridge=bridge50 interface=ether18
/interface bridge port add bridge=bridge51 interface=ether12
```

Experience 3

```
GNU4:

ifconfig eth1 down
ifconfig eth1 up
ifconfig eth1 172.16.51.253/24

Eth1
IP - 172.16.51.253
MAC - 00:c0:df:25:21:9e

MIKROTIK SWITCH:

/interface bridge port remove [find interface=ether22]
/interface bridge port add bridge=bridge51 interface=ether22

GNU4:

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

GNU2:
route add -net 172.16.50.0/24 gw 172.16.51.253

GNU3:
route add -net 172.16.51.0/24 gw 172.16.50.254
```

Experience 4

```
MIKROTIK ROUTER:

ip address add address=172.16.2.59/24 interface=ether1
ip address add address=172.16.51.254/24 interface=ether2

MIKROTIK SWITCH:

/interface bridge port remove [find interface=ether24]
/interface bridge port add bridge=bridge51 interface=ether24

GNU2:
route add default gw 172.16.51.254

GNU3:
route add default gw 172.16.50.254

GNU4:
route add default gw 172.16.51.254

MIKROTIK ROUTER:

/ip route add dst-address=172.16.50.0/24 gateway=172.16.51.253

GNU2:
echo 0 > /proc/sys/net/ipv4/conf/eth0/accept_redirects
echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects

route del -net 172.16.50.0/24

traceroute 172.16.50.1
route add -net 172.16.50.0/24 gw 172.16.51.253
traceroute 172.16.50.1

route del -net 172.16.50.0/24

to activate acceptance of icmp:
echo 1 > /proc/sys/net/ipv4/conf/eth0/accept_redirects
echo 1 > /proc/sys/net/ipv4/conf/all/accept_redirects
conclusion: we just do one initial hop to 172.16.51.254 and then it will know where to
     go (the router answers where to go)


GNU3:
ping 172.16.2.254 (to get to the network, tux 3 just knows that he needs to send to
    the default gateway tux4, the tux4 only knows he has to send to the rc)

MIKROTIK ROUTER:

/ip firewall nat disable 0

GNU3:
ping 172.16.2.254 (an error occured because rc doesnt know how to get to the network
    becuase nat is disabled so it doesnt convert private to public ip)
```

Experience 5

```
MIKORTIK ROUTER:

add default gateway to have internet:
/ip route add dst-address=0.0.0.0/0 gateway=172.16.2.254

enable nat:
/ip firewall nat add chain=srcnat action=masquerade out-interface=ether1

EACH GNU:

configure DNS:
edit the resolv.conf in the text editor:
nameserver 172.16.2.1

ping google.com
```

9 Wireshark Logs

Experience 1

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	HewlettP_61:2c:54	Broadcast	ARP	42	Who has 172.16.50.254? Tell 172.16.50.1
2	0.000101269	HewlettP_19:09:5c	HewlettP_61:2c:54	ARP	60	172.16.50.254 is at 00:22:64:19:09:5c
3	0.000108672	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x321c, seq=1/256,
4	0.000195973	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x321c, seq=1/256,
5	0.901646500	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost
6	1.004981831	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x321c, seq=2/512,
7	1.005098675	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x321c, seq=2/512,
8	2.029033331	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x321c, seq=3/768,
9	2.029156809	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x321c, seq=3/768,
10	2.903818519	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost
11	3.052983841	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x321c, seq=4/1024,
12	3.053085739	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x321c, seq=4/1024,
13	4.076987081	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x321c, seq=5/1280,
14	4.077094705	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x321c, seq=5/1280,
15	4.905978944	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost
16	5.100984384	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x321c, seq=6/1536,
17	5.101087679	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x321c, seq=6/1536,
18	6.124981268	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x321c, seq=7/1792,
19	6.125104328	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x321c, seq=7/1792,
20	6.908138321	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost
21	7.148982553	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x321c, seq=8/2048,
22	7.149085428	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x321c, seq=8/2048,
23	8.172983697	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x321c, seq=9/2304,
24	8.173090414	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x321c, seq=9/2304,
25	8.910302308	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost
26	9.196978696	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x321c, seq=10/2560
27	9.197083946	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x321c, seq=10/2560

Experience 2.6
Tux3

No.	Time	Source	Destination	Protocol	Length	Info
18	26.972517798	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x531b, seq=1/256, ttl=64
19	26.972687721	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x531b, seq=1/256, ttl=64
20	28.002164531	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x531b, seq=2/512, ttl=64
21	28.002297088	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x531b, seq=2/512, ttl=64
22	28.004855562	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
23	29.026164015	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x531b, seq=3/768, ttl=64
24	29.026294128	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x531b, seq=3/768, ttl=64
25	29.999380523	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
26	30.050162172	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x531b, seq=4/1024, ttl=64
27	30.050321339	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x531b, seq=4/1024, ttl=64
28	31.074167523	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x531b, seq=5/1280, ttl=64
29	31.074303153	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x531b, seq=5/1280, ttl=64
30	32.004788407	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
31	32.096167049	HewlettP_19:09:5c	HewlettP_61:2c:54	ARP	60	Who has 172.16.50.1? Tell 172.16.50.254
32	32.096183112	HewlettP_61:2c:54	HewlettP_19:09:5c	ARP	42	172.16.50.1 is at 00:21:5a:61:2c:54
33	32.098149477	HewlettP_61:2c:54	HewlettP_19:09:5c	ARP	42	Who has 172.16.50.254? Tell 172.16.50.1
34	32.098159254	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x531b, seq=6/1536, ttl=64
35	32.098249698	HewlettP_19:09:5c	HewlettP_61:2c:54	ARP	60	172.16.50.254 is at 00:22:64:19:09:5c
36	32.098267717	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x531b, seq=6/1536, ttl=64
37	33.122163417	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x531b, seq=7/1792, ttl=64
38	33.122293181	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x531b, seq=7/1792, ttl=64
39	33.998933783	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
40	34.146161783	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x531b, seq=8/2048, ttl=64
41	34.146287217	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x531b, seq=8/2048, ttl=64

Experience 2.8 Tux2

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Port = 0x8001
2	2.009800684	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Port = 0x8001
3	4.019589984	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Port = 0x8001
4	6.025817299	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Port = 0x8001
5	8.029359925	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Port = 0x8001
6	10.039641546	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Port = 0x8001
7	12.049114177	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Port = 0x8001
8	14.044176765	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Port = 0x8001
9	16.048989133	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Port = 0x8001
10	18.048696338	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Port = 0x8001
11	20.049007125	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Port = 0x8001
12	22.048372170	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Port = 0x8001
13	24.044225377	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Port = 0x8001
14	26.048560019	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Port = 0x8001
15	28.043101018	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Port = 0x8001
16	30.048260712	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Port = 0x8001
17	32.047981395	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Port = 0x8001
18	34.043925257	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Port = 0x8001

Tux3

No.	Time	Source	Destination	Protocol	Length	Info
11	20.028157774	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
12	21.965059659	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x53b5, seq=1/256, ttl=64
13	21.965254724	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x53b5, seq=1/256, ttl=64
14	22.032974344	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
15	22.971788289	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x53b5, seq=2/512, ttl=64
16	22.971933767	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x53b5, seq=2/512, ttl=64
17	23.995807934	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x53b5, seq=3/768, ttl=64
18	23.995995736	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x53b5, seq=3/768, ttl=64
19	24.032676481	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
20	25.019794824	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x53b5, seq=4/1024, ttl=64
21	25.019974943	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x53b5, seq=4/1024, ttl=64
22	26.032999221	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
23	26.043792121	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x53b5, seq=5/1280, ttl=64
24	26.043937180	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x53b5, seq=5/1280, ttl=64
25	27.067797867	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x53b5, seq=6/1536, ttl=64
26	27.067942577	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x53b5, seq=6/1536, ttl=64
27	27.129808151	HewlettP_19:09:5c	HewlettP_61:2c:54	ARP	60	Who has 172.16.50.1? Tell 172.16.50.254
28	27.129829802	HewlettP_61:2c:54	HewlettP_19:09:5c	ARP	42	172.16.50.1 is at 00:21:5a:61:2c:54
29	28.032357810	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
30	28.091799214	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x53b5, seq=7/1792, ttl=64
31	28.091935124	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x53b5, seq=7/1792, ttl=64
32	29.115793646	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x53b5, seq=8/2048, ttl=64
33	29.115969575	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x53b5, seq=8/2048, ttl=64

Tux4

No.	Time	Source	Destination	Protocol	Length	Info
13	24.033027810	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
14	25.969993433	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x53b5, seq=1/256, ttl=64
15	25.970028284	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x53b5, seq=1/256, ttl=64
16	26.037837749	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
17	26.976701757	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x53b5, seq=2/512, ttl=64
18	26.976711116	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x53b5, seq=2/512, ttl=64
19	28.000731475	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x53b5, seq=3/768, ttl=64
20	28.000769190	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x53b5, seq=3/768, ttl=64
21	28.037531336	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
22	29.024715447	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x53b5, seq=4/1024, ttl=64
23	29.024749739	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x53b5, seq=4/1024, ttl=64
24	30.037855388	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
25	30.048695227	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x53b5, seq=5/1280, ttl=64
26	30.048713456	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x53b5, seq=5/1280, ttl=64
27	31.072708253	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x53b5, seq=6/1536, ttl=64
28	31.072718310	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x53b5, seq=6/1536, ttl=64
29	31.134576531	HewlettP_19:09:5c	HewlettP_61:2c:54	ARP	42	Who has 172.16.50.1? Tell 172.16.50.254
30	31.134711116	HewlettP_61:2c:54	HewlettP_19:09:5c	ARP	60	172.16.50.1 is at 00:21:5a:61:2c:54
31	32.037215618	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
32	32.096704307	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x53b5, seq=7/1792, ttl=64
33	32.096712548	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x53b5, seq=7/1792, ttl=64
34	33.120712164	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x53b5, seq=8/2048, ttl=64
35	33.120743802	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x53b5, seq=8/2048, ttl=64

Experience 2.10
Tux2

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Po
2	2.003132004	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Po
3	4.002532515	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Po
4	6.002752282	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Po
5	8.002949699	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Po
6	8.271221741	172.16.51.0	172.16.51.255	ICMP	98	Echo (ping) request id=0x5141, seq=1/256, ttl=64
7	9.283360795	172.16.51.0	172.16.51.255	ICMP	98	Echo (ping) request id=0x5141, seq=2/512, ttl=64
8	10.009010491	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Po
9	10.307359650	172.16.51.0	172.16.51.255	ICMP	98	Echo (ping) request id=0x5141, seq=3/768, ttl=64
10	11.331386652	172.16.51.0	172.16.51.255	ICMP	98	Echo (ping) request id=0x5141, seq=4/1024, ttl=64
11	12.012363407	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Po
12	12.355359037	172.16.51.0	172.16.51.255	ICMP	98	Echo (ping) request id=0x5141, seq=5/1280, ttl=64
13	13.379357753	172.16.51.0	172.16.51.255	ICMP	98	Echo (ping) request id=0x5141, seq=6/1536, ttl=64
14	14.012893695	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Po
15	14.403352068	172.16.51.0	172.16.51.255	ICMP	98	Echo (ping) request id=0x5141, seq=7/1792, ttl=64
16	15.427358047	172.16.51.0	172.16.51.255	ICMP	98	Echo (ping) request id=0x5141, seq=8/2048, ttl=64
17	16.018257385	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Po
18	16.451383024	172.16.51.0	172.16.51.255	ICMP	98	Echo (ping) request id=0x5141, seq=9/2304, ttl=64
19	17.475366723	172.16.51.0	172.16.51.255	ICMP	98	Echo (ping) request id=0x5141, seq=10/2560, ttl=64
20	18.018262174	Routerbo_1c:95:c9	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Po
21	18.503354791	172.16.51.0	172.16.51.255	ICMP	98	Echo (ping) request id=0x5141, seq=11/2816, ttl=64
22	19.523376585	172.16.51.0	172.16.51.255	ICMP	98	Echo (ping) request id=0x5141, seq=12/3072, ttl=64

Tux3

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8001
2	2.009503551	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8001
3	4.011746464	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8001
4	6.014876353	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8001
5	8.014288972	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8001
6	10.014502384	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8001
7	12.014711604	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8001
8	14.020772147	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8001
9	16.024116724	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8001
10	18.024651401	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8001
11	20.030013326	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8001
12	22.030102698	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8001
13	24.033921562	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8001
14	26.033576314	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8001
15	28.040126716	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8001

Tux4

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8002
2	1.999739740	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8002
3	4.003190678	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8002
4	6.012700068	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8002
5	8.014939525	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8002
6	10.018067304	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8002
7	12.017480462	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8002
8	14.017700365	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8002
9	16.017904065	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8002
10	18.023965000	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8002
11	20.027309568	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8002
12	22.027844947	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8002
13	24.033221709	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8002
14	26.033295431	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8002
15	28.037122257	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8002
16	30.036766032	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8002
17	32.043318444	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8002
18	34.046603435	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8002
19	36.046493402	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8002

Experience 3.7

No.	Time	Source	Destination	Protocol	Length	Info
10	6.022347157	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
11	6.845775440	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x59b5, seq=4/1024, ttl=64
12	6.845942219	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x59b5, seq=4/1024, ttl=64
13	7.869769593	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x59b5, seq=5/1280, ttl=64
14	7.869901661	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x59b5, seq=5/1280, ttl=64
15	8.032661307	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
16	8.795882317	HewlettP_19:09:5c	HewlettP_61:2c:54	ARP	60	Who has 172.16.50.1? Tell 172.16.50.254
17	8.795902501	HewlettP_61:2c:54	HewlettP_19:09:5c	ARP	42	172.16.50.1 is at 00:21:5a:61:2c:54
18	8.797745945	HewlettP_61:2c:54	HewlettP_19:09:5c	ARP	42	Who has 172.16.50.254? Tell 172.16.50.1
19	8.797838414	HewlettP_19:09:5c	HewlettP_61:2c:54	ARP	60	172.16.50.254 is at 00:22:64:19:09:5c
20	10.042319793	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
21	12.049207699	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
22	12.359369655	172.16.50.1	172.16.51.253	ICMP	98	Echo (ping) request id=0x59b9, seq=1/256, ttl=64 (
23	12.359534339	172.16.51.253	172.16.50.1	ICMP	98	Echo (ping) reply id=0x59b9, seq=1/256, ttl=64 (
24	13.373807578	172.16.50.1	172.16.51.253	ICMP	98	Echo (ping) request id=0x59b9, seq=2/512, ttl=64 (
25	13.373937552	172.16.51.253	172.16.50.1	ICMP	98	Echo (ping) reply id=0x59b9, seq=2/512, ttl=64 (
26	14.049357167	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
27	14.397827084	172.16.50.1	172.16.51.253	ICMP	98	Echo (ping) request id=0x59b9, seq=3/768, ttl=64 (
28	14.397962854	172.16.51.253	172.16.50.1	ICMP	98	Echo (ping) reply id=0x59b9, seq=3/768, ttl=64 (
29	15.421791484	172.16.50.1	172.16.51.253	ICMP	98	Echo (ping) request id=0x59b9, seq=4/1024, ttl=64
30	15.421922645	172.16.51.253	172.16.50.1	ICMP	98	Echo (ping) reply id=0x59b9, seq=4/1024, ttl=64
31	16.051934018	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
32	18.051970134	Routerbo_1c:95:ca	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
33	18.599272825	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x59c0, seq=1/256, ttl=64 (

Experience 3.11
Tux 4 eth0

No.	Time	Source	Destination	Protocol	Length	Info
7	11.167349953	HewlettP_61:2c:54	Broadcast	ARP	60	Who has 172.16.50.254? Tell 172.16.50.1
8	11.167378448	HewlettP_19:09:5c	HewlettP_61:2c:54	ARP	42	172.16.50.254 is at 00:22:64:19:09:5c
9	11.167491942	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5b56, seq=1/256, ttl=64 (
10	11.167795196	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5b56, seq=1/256, ttl=63 (
11	12.005027708	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
12	12.198575116	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5b56, seq=2/512, ttl=64 (
13	12.198739384	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5b56, seq=2/512, ttl=63 (
14	13.222568830	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5b56, seq=3/768, ttl=64 (
15	13.222734495	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5b56, seq=3/768, ttl=63 (
16	14.014745241	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
17	14.246566595	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5b56, seq=4/1024, ttl=64
18	14.246758870	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5b56, seq=4/1024, ttl=63
19	15.270571484	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5b56, seq=5/1280, ttl=64
20	15.270741270	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5b56, seq=5/1280, ttl=63
21	16.024230479	Routerbo_1c:95:cb	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Po
22	16.294564640	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5b56, seq=6/1536, ttl=64
23	16.294710400	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5b56, seq=6/1536, ttl=63
24	16.420539979	HewlettP_19:09:5c	HewlettP_61:2c:54	ARP	42	Who has 172.16.50.1? Tell 172.16.50.254
25	16.420678546	HewlettP_61:2c:54	HewlettP_19:09:5c	ARP	60	172.16.50.1 is at 00:21:5a:61:2c:54
26	17.318564779	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5b56, seq=7/1792, ttl=64
27	17.318734356	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5b56, seq=7/1792, ttl=63

Tux 4 eth1

No.	Time	Source	Destination	Protocol	Length	Info
7	10.166398536	KYE_25:21:9e	Broadcast	ARP	42	Who has 172.16.51.1? Tell 172.16.51.253
8	10.166539198	HewlettP_5a:7c:e7	KYE_25:21:9e	ARP	60	172.16.51.1 is at 00:21:5a:5a:7c:e7
9	10.166554005	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5b56, seq=1/256, ttl=63 (
10	10.166672457	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5b56, seq=1/256, ttl=64 (
11	11.197485691	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5b56, seq=2/512, ttl=63 (
12	11.197608753	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5b56, seq=2/512, ttl=64 (
13	12.005041187	Routerbo_1c:95:cf	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Po
14	12.221482339	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5b56, seq=3/768, ttl=63 (
15	12.221603096	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5b56, seq=3/768, ttl=64 (
16	13.245481221	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5b56, seq=4/1024, ttl=63
17	13.245627471	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5b56, seq=4/1024, ttl=64
18	14.014750758	Routerbo_1c:95:cf	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Po
19	14.269485412	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5b56, seq=5/1280, ttl=63
20	14.269610918	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5b56, seq=5/1280, ttl=64
21	15.234400552	HewlettP_5a:7c:e7	KYE_25:21:9e	ARP	60	Who has 172.16.51.253? Tell 172.16.51.1
22	15.234422622	KYE_25:21:9e	HewlettP_5a:7c:e7	ARP	42	172.16.51.253 is at 00:c0:df:25:21:9e
23	15.293474377	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5b56, seq=6/1536, ttl=63
24	15.293591223	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5b56, seq=6/1536, ttl=64
25	16.024233342	Routerbo_1c:95:cf	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c9 Cost = 0 Po
26	16.317480802	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5b56, seq=7/1792, ttl=63
27	16.317603585	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5b56, seq=7/1792, ttl=64

Experience 4.3
Tux 2

No.	Time	Source	Destination	Protocol	Length	Info
17	30.258710293	KYE_25:21:9e	Broadcast	ARP	60	Who has 172.16.51.1? Tell 172.16.51.253
18	30.258737113	HewlettP_5a:7c:e7	KYE_25:21:9e	ARP	42	172.16.51.1 is at 00:21:5a:5a:7c:e7
19	30.258871561	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5220, seq=1/256, ttl=63 (
20	30.258892444	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5220, seq=1/256, ttl=64 (
21	31.288003053	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5220, seq=2/512, ttl=63 (
22	31.288012133	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5220, seq=2/512, ttl=64 (
23	32.033898660	Routerbo_1c:95:d3	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/74:4d:28:eb:24:1d Cost = 10 P
24	32.311983003	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5220, seq=3/768, ttl=63 (
25	32.312005841	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5220, seq=3/768, ttl=64 (
26	33.339975008	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5220, seq=4/1024, ttl=63
27	33.339982970	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5220, seq=4/1024, ttl=64
28	34.035924418	Routerbo_1c:95:d3	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/74:4d:28:eb:24:1d Cost = 10 P
29	34.359977684	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5220, seq=5/1280, ttl=63
30	34.359987811	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5220, seq=5/1280, ttl=64
31	35.383977958	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x5220, seq=6/1536, ttl=63
32	35.383987387	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5220, seq=6/1536, ttl=64
33	35.473005800	HewlettP_5a:7c:e7	KYE_25:21:9e	ARP	42	Who has 172.16.51.253? Tell 172.16.51.1
34	35.473118876	KYE_25:21:9e	HewlettP_5a:7c:e7	ARP	60	172.16.51.253 is at 00:c0:df:25:21:9e

Tux 3

No.	Time	Source	Destination	Protocol	Length	Info
18	10.923153884	HewlettP_19:09:5c	HewlettP_61:2c:54	ARP	60	172.16.50.254 is at 00:22:64:19:09:5c
19	10.966490304	HewlettP_19:09:5c	HewlettP_61:2c:54	ARP	60	Who has 172.16.50.1? Tell 172.16.50.254
20	10.966498895	HewlettP_61:2c:54	HewlettP_19:09:5c	ARP	42	172.16.50.1 is at 00:21:5a:61:2c:54
21	12.000440913	172.16.50.1	172.16.2.1	DNS	84	Standard query 0xb3f1 A detectportal.firefox.com
22	12.000450760	172.16.50.1	172.16.2.1	DNS	84	Standard query 0x0502 AAAA detectportal.firefox.com
23	12.001764394	172.16.2.1	172.16.50.1	DNS	198	Standard query response 0xb3f1 A detectportal.firefox.com CNAME
24	12.001779130	172.16.2.1	172.16.50.1	DNS	210	Standard query response 0x0502 AAAA detectportal.firefox.com CNA
25	12.002156409	172.16.50.1	34.107.221.82	TCP	74	46870 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=
26	12.002391284	172.16.51.254	172.16.50.1	ICMP	102	Destination unreachable (Network unreachable)
27	12.003541071	172.16.50.1	172.16.2.1	DNS	84	Standard query 0xb6b3 A detectportal.firefox.com
28	12.003548334	172.16.50.1	172.16.2.1	DNS	84	Standard query 0xbdbb AAAA detectportal.firefox.com
29	12.004227884	172.16.2.1	172.16.50.1	DNS	198	Standard query response 0xb6b3 A detectportal.firefox.com CNAME
30	12.004241992	172.16.2.1	172.16.50.1	DNS	210	Standard query response 0xbdbb AAAA detectportal.firefox.com CNA
31	12.004516675	172.16.50.1	34.107.221.82	TCP	74	46886 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=
32	12.004730527	172.16.51.254	172.16.50.1	ICMP	102	Destination unreachable (Network unreachable)
33	12.005770176	172.16.50.1	172.16.2.1	DNS	84	Standard query 0x93cc A detectportal.firefox.com
34	12.005776461	172.16.50.1	172.16.2.1	DNS	84	Standard query 0x7bd5 AAAA detectportal.firefox.com
35	12.006465859	172.16.2.1	172.16.50.1	DNS	198	Standard query response 0x93cc A detectportal.firefox.com CNAME
36	12.006478709	172.16.2.1	172.16.50.1	DNS	210	Standard query response 0x7bd5 AAAA detectportal.firefox.com CNA
37	12.006746059	172.16.50.1	34.107.221.82	TCP	74	46892 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=
38	12.006960400	172.16.51.254	172.16.50.1	ICMP	102	Destination unreachable (Network unreachable)
39	12.007980144	172.16.50.1	172.16.2.1	DNS	84	Standard query 0xf51d A detectportal.firefox.com

Tux 3

No.	Time	Source	Destination	Protocol	Length	Info
19	14.907265590	Routerbo_eb:24:1d	KYE_25:21:9e	ARP	60	Who has 172.16.51.253? Tell 172.16.51.254
20	14.907286683	KYE_25:21:9e	Routerbo_eb:24:1d	ARP	42	172.16.51.253 is at 00:c0:df:25:21:9e
21	14.970747982	KYE_25:21:9e	Routerbo_eb:24:1d	ARP	42	Who has 172.16.51.254? Tell 172.16.51.253
22	14.970845621	Routerbo_eb:24:1d	KYE_25:21:9e	ARP	60	172.16.51.254 is at 74:4d:28:eb:24:1d
23	16.004831704	172.16.50.1	172.16.2.1	DNS	84	Standard query 0xb3f1 A detectportal.firefox.com
24	16.004839107	172.16.50.1	172.16.2.1	DNS	84	Standard query 0x0502 AAAA detectportal.firefox.com
25	16.005994295	172.16.2.1	172.16.50.1	DNS	198	Standard query response 0xb3f1 A detectportal.firefox.com
26	16.006022162	172.16.2.1	172.16.50.1	DNS	210	Standard query response 0x0502 AAAA detectportal.firefox.
27	16.006526981	172.16.50.1	34.107.221.82	TCP	74	46870 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
28	16.006651858	172.16.51.254	172.16.50.1	ICMP	102	Destination unreachable (Network unreachable)
29	16.007916000	172.16.50.1	172.16.2.1	DNS	84	Standard query 0xb6b3 A detectportal.firefox.com
30	16.007921797	172.16.50.1	172.16.2.1	DNS	84	Standard query 0xbdbb AAAA detectportal.firefox.com
31	16.008461746	172.16.2.1	172.16.50.1	DNS	198	Standard query response 0xb6b3 A detectportal.firefox.com
32	16.008481092	172.16.2.1	172.16.50.1	DNS	210	Standard query response 0xbdbb AAAA detectportal.firefox.
33	16.008885408	172.16.50.1	34.107.221.82	TCP	74	46886 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
34	16.008993104	172.16.51.254	172.16.50.1	ICMP	102	Destination unreachable (Network unreachable)
35	16.010142635	172.16.50.1	172.16.2.1	DNS	84	Standard query 0x93cc A detectportal.firefox.com
36	16.010148432	172.16.50.1	172.16.2.1	DNS	84	Standard query 0x7bd5 AAAA detectportal.firefox.com
37	16.010701092	172.16.2.1	172.16.50.1	DNS	198	Standard query response 0x93cc A detectportal.firefox.com
38	16.010720508	172.16.2.1	172.16.50.1	DNS	210	Standard query response 0x7bd5 AAAA detectportal.firefox.
39	16.011115046	172.16.50.1	34.107.221.82	TCP	74	46892 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
40	16.011223231	172.16.51.254	172.16.50.1	ICMP	102	Destination unreachable (Network unreachable)
41	16.012352578	172.16.50.1	172.16.2.1	DNS	84	Standard query 0xf51d A detectportal.firefox.com
42	16.012358305	172.16.50.1	172.16.2.1	DNS	84	Standard query 0x9028 AAAA detectportal.firefox.com

Experience 4.4

No.	Time	Source	Destination	Protocol	Length	Info
4	3.791220212	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x4b94, seq=1/256, ttl=63 (request in 3)
5	4.004327325	Routerbo_1c:95:d3	Spanning-tree-(for-bridges)_00	STP	60	RST. Root = 32768/0/74:4d:28:eb:24:1d Cost = 10 Port = 0x8001
6	4.813677656	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x4b94, seq=2/512, ttl=64 (reply in 8)
7	4.813838783	172.16.51.254	172.16.51.1	ICMP	126	Redirect (Redirect for host)
8	4.814026870	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x4b94, seq=2/512, ttl=63 (request in 6)
9	5.392415935	fe80::2c0:dfff:fe25:219e	ff02::fb	MDNS	180	Standard query 0x0000 PTR _ftp._tcp.local, "QM" question PTR _nfs._tcp.local,
10	5.392457701	172.16.51.253	224.0.0.251	MDNS	160	Standard query 0x0000 PTR _ftp._tcp.local, "QM" question PTR _nfs._tcp.local,
11	5.837692806	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x4b94, seq=3/768, ttl=64 (reply in 13)
12	5.837837171	172.16.51.254	172.16.51.1	ICMP	126	Redirect (Redirect for host)
13	5.838023582	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x4b94, seq=3/768, ttl=63 (request in 11)
14	6.006497448	Routerbo_1c:95:d3	Spanning-tree-(for-bridges)_00	STP	60	RST. Root = 32768/0/74:4d:28:eb:24:1d Cost = 10 Port = 0x8001
15	6.861692871	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x4b94, seq=4/1024, ttl=64 (reply in 17)
16	6.861841915	172.16.51.254	172.16.51.1	ICMP	126	Redirect (Redirect for host)
17	6.862068765	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x4b94, seq=4/1024, ttl=63 (request in 15)
18	7.885690560	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x4b94, seq=5/1280, ttl=64 (reply in 20)
19	7.885857764	172.16.51.254	172.16.51.1	ICMP	126	Redirect (Redirect for host)
20	7.886094532	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x4b94, seq=5/1280, ttl=63 (request in 18)
21	8.008661635	Routerbo_1c:95:d3	Spanning-tree-(for-bridges)_00	STP	60	RST. Root = 32768/0/74:4d:28:eb:24:1d Cost = 10 Port = 0x8001
22	8.498242514	0.0.0.0	255.255.255.255	MNDP	159	5678 → 5678 Len=117
23	8.498278413	Routerbo_1c:95:d3	CDP/VTP/DTP/PAgP/UDLD	CDP	93	Device ID: MikroTik Port ID: bridge51
24	8.498328560	Routerbo_1c:95:d3	LLDP_Multicast	LLDP	110	MA/c4:ad:34:1c:95:d3 IN/bridge51 120 SysN=MikroTik SysD=MikroTik RouterOS 6.4
25	8.800577570	KYE_25:21:9e	HewlettP_5a:7c:e7	ARP	60	Who has 172.16.51.1? Tell 172.16.51.253
26	8.800597475	HewlettP_5a:7c:e7	KYE_25:21:9e	ARP	42	172.16.51.1 is at 00:21:5a:5a:7c:e7
27	8.909686783	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x4b94, seq=6/1536, ttl=64 (reply in 29)
28	8.909828495	172.16.51.254	172.16.51.1	ICMP	126	Redirect (Redirect for host)
29	8.910010854	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x4b94, seq=6/1536, ttl=63 (request in 27)
30	9.005666229	HewlettP_5a:7c:e7	Routerbo_eb:24:1d	ARP	42	Who has 172.16.51.254? Tell 172.16.51.1
31	9.005766244	Routerbo_eb:24:1d	HewlettP_5a:7c:e7	ARP	60	172.16.51.254 is at 74:4d:28:eb:24:1d

Experience 5.3

No.	Time	Source	Destination	Protocol	Length	Info
23	2.961519761	172.16.50.1	142.250.200.78	ICMP	98	Echo (ping) request id=0x5ae3, seq=3/768, ttl=64 (reply in 24)
24	2.977806887	142.250.200.78	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5ae3, seq=3/768, ttl=112 (request in 2
25	3.962875934	172.16.50.1	142.250.200.78	ICMP	98	Echo (ping) request id=0x5ae3, seq=4/1024, ttl=64 (reply in 26)
26	3.979232132	142.250.200.78	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5ae3, seq=4/1024, ttl=112 (request in
27	4.758638468	Routerbo_1c:95:cf	Spanning-tree-(for-bridge…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:cf Cost = 0 Port = 0x8001
28	4.964307954	172.16.50.1	142.250.200.78	ICMP	98	Echo (ping) request id=0x5ae3, seq=5/1280, ttl=64 (reply in 29)
29	4.980529988	142.250.200.78	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5ae3, seq=5/1280, ttl=112 (request in
30	5.008091425	172.16.50.1	172.16.2.1	DNS	85	Standard query 0xf596 PTR 5.121.82.140.in-addr.arpa
31	5.008991741	172.16.2.1	172.16.50.1	DNS	129	Standard query response 0xf596 PTR 5.121.82.140.in-addr.arpa PTR
32	5.009220469	172.16.50.1	172.16.2.1	DNS	88	Standard query 0xb415 PTR 154.110.199.185.in-addr.arpa
33	5.010063586	172.16.2.1	172.16.50.1	DNS	132	Standard query response 0xb415 PTR 154.110.199.185.in-addr.arpa
34	5.010196982	172.16.50.1	172.16.2.1	DNS	85	Standard query 0xbcfd PTR 3.121.82.140.in-addr.arpa
35	5.011038701	172.16.2.1	172.16.50.1	DNS	129	Standard query response 0xbcfd PTR 3.121.82.140.in-addr.arpa PTR
36	5.011253601	172.16.50.1	172.16.2.1	DNS	88	Standard query 0x91ae PTR 154.109.199.185.in-addr.arpa
37	5.011994262	172.16.2.1	172.16.50.1	DNS	132	Standard query response 0x91ae PTR 154.109.199.185.in-addr.arpa
38	5.012162159	172.16.50.1	172.16.2.1	DNS	86	Standard query 0x6852 PTR 29.220.184.93.in-addr.arpa
39	5.012953314	172.16.2.1	172.16.50.1	DNS	157	Standard query response 0x6852 No such name PTR 29.220.184.93.in
40	5.419525869	172.16.50.1	93.184.220.29	TCP	66	41660 → 80 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=646488780 TSecr
41	5.465927667	93.184.220.29	172.16.50.1	TCP	66	[TCP ACKed unseen segment] 80 → 41660 [ACK] Seq=1 Ack=2 Win=135
42	5.931524676	172.16.50.1	34.107.221.82	TCP	66	56912 → 80 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=2415159127 TSec
43	5.931532498	172.16.50.1	34.107.221.82	TCP	66	56900 → 80 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=2415159127 TSec
44	5.945354387	34.107.221.82	172.16.50.1	TCP	66	[TCP ACKed unseen segment] 80 → 56900 [ACK] Seq=1 Ack=2 Win=265
45	5.947108855	34.107.221.82	172.16.50.1	TCP	66	[TCP ACKed unseen segment] 80 → 56912 [ACK] Seq=1 Ack=2 Win=265
46	5.965605740	172.16.50.1	142.250.200.78	ICMP	98	Echo (ping) request id=0x5ae3, seq=6/1536, ttl=64 (reply in 47)
47	5.982242977	142.250.200.78	172.16.50.1	ICMP	98	Echo (ping) reply id=0x5ae3, seq=6/1536, ttl=112 (request in

Experience 6.3

No.	Time	Source	Destination	Protocol	Length	Info
7	3.324209811	172.16.50.1	172.16.2.1	DNS	76	Standard query 0x69a9 A netlab1.fe.up.pt
8	3.325160622	172.16.2.1	172.16.50.1	DNS	92	Standard query response 0x69a9 A netlab1.fe.up.pt A 192.168.109.136
9	3.325296043	172.16.50.1	192.168.109.136	TCP	74	48054 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=4226833796 TSecr
10	3.326316415	192.168.109.136	172.16.50.1	TCP	74	21 → 48054 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=14288
11	3.326334364	172.16.50.1	192.168.109.136	TCP	66	48054 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=4226833797 TSecr=142886269
12	3.328345287	192.168.109.136	172.16.50.1	FTP	100	Response: 220 Welcome to netlab-FTP server
13	3.328355134	172.16.50.1	192.168.109.136	TCP	66	48054 → 21 [ACK] Seq=1 Ack=35 Win=64256 Len=0 TSval=4226833799 TSecr=142886271
14	3.328392220	172.16.50.1	192.168.109.136	FTP	76	Request: user rcom
15	3.328965821	192.168.109.136	172.16.50.1	TCP	66	21 → 48054 [ACK] Seq=35 Ack=11 Win=65280 Len=0 TSval=142886272 TSecr=4226833799
16	3.329096703	192.168.109.136	172.16.50.1	FTP	100	Response: 331 Please specify the password.
17	3.329120239	172.16.50.1	192.168.109.136	FTP	76	Request: pass rcom
18	3.329922429	192.168.109.136	172.16.50.1	TCP	66	21 → 48054 [ACK] Seq=69 Ack=21 Win=65280 Len=0 TSval=142886273 TSecr=4226833800
19	3.339220319	192.168.109.136	172.16.50.1	FTP	89	Response: 230 Login successful.
20	3.339244065	172.16.50.1	192.168.109.136	FTP	71	Request: pasv
21	3.339895119	192.168.109.136	172.16.50.1	TCP	66	21 → 48054 [ACK] Seq=92 Ack=26 Win=65280 Len=0 TSval=142886283 TSecr=4226833810
22	3.340053797	192.168.109.136	172.16.50.1	FTP	120	Response: 227 Entering Passive Mode (192,168,109,136,159,236).
23	3.340096330	172.16.50.1	192.168.109.136	TCP	74	46050 → 40940 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=4226833811 TS
24	3.340805004	192.168.109.136	172.16.50.1	TCP	74	40940 → 46050 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=14
25	3.340815899	172.16.50.1	192.168.109.136	TCP	66	46050 → 40940 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=4226833811 TSecr=142886283
26	3.340831264	172.16.50.1	192.168.109.136	FTP	80	Request: retr pipe.txt
27	3.341433989	192.168.109.136	172.16.50.1	TCP	66	21 → 48054 [ACK] Seq=146 Ack=40 Win=65280 Len=0 TSval=142886284 TSecr=4226833811
28	3.341606705	192.168.109.136	172.16.50.1	FTP	134	Response: 150 Opening BINARY mode data connection for pipe.txt (1863 bytes).
29	3.341963102	192.168.109.136	172.16.50.1	FTP-DATA	1929	FTP Data: 1863 bytes (PASV) (retr pipe.txt)
30	3.341973578	172.16.50.1	192.168.109.136	TCP	66	46050 → 40940 [ACK] Seq=1 Ack=1864 Win=63488 Len=0 TSval=4226833812 TSecr=142886
31	3.341977489	192.168.109.136	172.16.50.1	TCP	66	40940 → 46050 [FIN, ACK] Seq=1864 Ack=1 Win=65280 Len=0 TSval=142886284 TSecr=42
32	3.383472141	172.16.50.1	192.168.109.136	TCP	66	46050 → 40940 [ACK] Seq=1 Ack=1865 Win=64128 Len=0 TSval=4226833854 TSecr=142886
33	3.383476611	172.16.50.1	192.168.109.136	TCP	66	48054 → 21 [ACK] Seq=40 Ack=214 Win=64256 Len=0 TSval=4226833854 TSecr=142886284
34	3.384508926	192.168.109.136	172.16.50.1	FTP	90	Response: 226 Transfer complete.

Experience 6.5
Tux 2

No.	Time	Source	Destination	Protocol	Length	Info
18	6.738441377	172.16.2.1	172.16.51.1	DNS	145	Standard query response 0x7800 PTR 60.20.89.52.in-addr.arpa PTR ec2-52-89-20-60
19	6.738569538	172.16.51.1	172.16.2.1	DNS	86	Standard query 0xaab2 PTR 21.103.201.35.in-addr.arpa
20	6.739226132	172.16.2.1	172.16.51.1	DNS	138	Standard query response 0xaab2 PTR 21.103.201.35.in-addr.arpa PTR 21.103.201.35
21	6.739352408	172.16.51.1	172.16.2.1	DNS	86	Standard query 0x0191 PTR 26.112.82.140.in-addr.arpa
22	6.739841936	172.16.2.1	172.16.51.1	DNS	131	Standard query response 0x0191 PTR 26.112.82.140.in-addr.arpa PTR lb-140-82-112
23	8.009853574	Routerbo_1c:95:d3	Spanning-tree-(for-bridges)_00	STP	60	RST. Root = 32768/0/74:4d:28:eb:24:1d Cost = 10 Port = 0x8001
24	10.014261394	Routerbo_1c:95:d3	Spanning-tree-(for-bridges)_00	STP	60	RST. Root = 32768/0/74:4d:28:eb:24:1d Cost = 10 Port = 0x8001
25	10.707133944	172.16.51.1	172.16.2.1	DNS	76	Standard query 0x7ed3 A netlab1.fe.up.pt
26	10.709608967	172.16.2.1	172.16.51.1	DNS	92	Standard query response 0x7ed3 A netlab1.fe.up.pt A 192.168.109.136
27	10.709729027	172.16.51.1	192.168.109.136	TCP	74	55428 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=4198200524 TSec
28	10.717972585	192.168.109.136	172.16.51.1	TCP	74	21 → 55428 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1433
29	10.717990395	172.16.51.1	192.168.109.136	TCP	66	55428 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=4198200532 TSecr=143379326
30	10.724862657	192.168.109.136	172.16.51.1	FTP	100	Response: 220 Welcome to netlab-FTP server
31	10.724872785	172.16.51.1	192.168.109.136	TCP	66	55428 → 21 [ACK] Seq=1 Ack=35 Win=64256 Len=0 TSval=4198200539 TSecr=143379336
32	10.724914411	172.16.51.1	192.168.109.136	FTP	76	Request: user rcom
33	10.731450170	192.168.109.136	172.16.51.1	TCP	66	21 → 55428 [ACK] Seq=35 Ack=11 Win=65280 Len=0 TSval=143379346 TSecr=4198200539
34	10.731456177	192.168.109.136	172.16.51.1	FTP	100	Response: 331 Please specify the password.
35	10.731480412	172.16.51.1	192.168.109.136	FTP	76	Request: pass rcom
36	10.733843826	192.168.109.136	172.16.51.1	TCP	66	21 → 55428 [ACK] Seq=69 Ack=21 Win=65280 Len=0 TSval=143379347 TSecr=4198200546
37	10.742497572	192.168.109.136	172.16.51.1	FTP	89	Response: 230 Login successful.
38	10.742529490	172.16.51.1	192.168.109.136	FTP	71	Request: pasv
39	10.743920062	192.168.109.136	172.16.51.1	TCP	66	21 → 55428 [ACK] Seq=92 Ack=26 Win=65280 Len=0 TSval=143379358 TSecr=4198200557
40	10.743925300	192.168.109.136	172.16.51.1	FTP	120	Response: 227 Entering Passive Mode (192,168,109,136,164,123).
41	10.743967416	172.16.51.1	192.168.109.136	TCP	74	37076 → 42107 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=4198200558 T
42	10.748605124	192.168.109.136	172.16.51.1	TCP	74	42107 → 37076 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=14
43	10.748616858	172.16.51.1	192.168.109.136	TCP	66	37076 → 42107 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=4198200563 TSecr=143379364
44	10.748635017	172.16.51.1	192.168.109.136	FTP	80	Request: retr pipe.txt
45	10.750332409	192.168.109.136	172.16.51.1	TCP	66	21 → 55428 [ACK] Seq=146 Ack=40 Win=65280 Len=0 TSval=143379364 TSecr=4198200563

Tux 3

No.	Time	Source	Destination	Protocol	Length	Info
2400	13.136463464	192.168.109.136	172.16.50.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
2401	13.136513470	192.168.109.136	172.16.50.1	FTP-DATA	7306	FTP Data: 7240 bytes (PASV) (retr files/crab.mp4)
2402	13.136526321	172.16.50.1	192.168.109.136	TCP	66	45688 → 42669 [ACK] Seq=1 Ack=3947249 Win=788736 Len=0 TSval=4227323272 TSecr
2403	13.136574371	192.168.109.136	172.16.50.1	FTP-DATA	7306	FTP Data: 7240 bytes (PASV) (retr files/crab.mp4)
2404	13.136584219	172.16.50.1	192.168.109.136	TCP	66	45688 → 42669 [ACK] Seq=1 Ack=3954489 Win=788736 Len=0 TSval=4227323272 TSecr
2405	13.136633177	192.168.109.136	172.16.50.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
2406	13.136639672	192.168.109.136	172.16.50.1	FTP-DATA	2962	FTP Data: 2896 bytes (PASV) (retr files/crab.mp4)
2407	13.136644631	192.168.109.136	172.16.50.1	FTP-DATA	2962	[TCP Previous segment not captured] FTP Data: 2896 bytes (PASV) (retr files/c
2408	13.136647634	172.16.50.1	192.168.109.136	TCP	66	45688 → 42669 [ACK] Seq=1 Ack=3958833 Win=785792 Len=0 TSval=4227323273 TSecr
2409	13.136654269	172.16.50.1	192.168.109.136	TCP	78	[TCP Dup ACK 2408#1] 45688 → 42669 [ACK] Seq=1 Ack=3958833 Win=785792 Len=0 T
2410	13.136697989	192.168.109.136	172.16.50.1	FTP-DATA	7306	FTP Data: 7240 bytes (PASV) (retr files/crab.mp4)
2411	13.136708535	172.16.50.1	192.168.109.136	TCP	78	[TCP Window Update] 45688 → 42669 [ACK] Seq=1 Ack=3958833 Win=786688 Len=0 TS
2412	13.136726903	192.168.109.136	172.16.50.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
2413	13.136733329	172.16.50.1	192.168.109.136	TCP	78	[TCP Dup ACK 2408#2] 45688 → 42669 [ACK] Seq=1 Ack=3958833 Win=786688 Len=0 T
2414	13.136789201	192.168.109.136	172.16.50.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
2415	13.136825728	172.16.50.1	192.168.109.136	TCP	78	[TCP Dup ACK 2408#3] 45688 → 42669 [ACK] Seq=1 Ack=3958833 Win=786688 Len=0 T
2416	13.136892216	192.168.109.136	172.16.50.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
2417	13.136928324	172.16.50.1	192.168.109.136	TCP	78	[TCP Dup ACK 2408#4] 45688 → 42669 [ACK] Seq=1 Ack=3958833 Win=786688 Len=0 T
2418	13.137012901	192.168.109.136	172.16.50.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
2419	13.137048799	172.16.50.1	192.168.109.136	TCP	78	[TCP Dup ACK 2408#5] 45688 → 42669 [ACK] Seq=1 Ack=3958833 Win=786688 Len=0 T
2420	13.137133516	192.168.109.136	172.16.50.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
2421	13.137169623	172.16.50.1	192.168.109.136	TCP	78	[TCP Dup ACK 2408#6] 45688 → 42669 [ACK] Seq=1 Ack=3958833 Win=786688 Len=0 T
2422	13.137509189	192.168.109.136	172.16.50.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
2423	13.137545157	172.16.50.1	192.168.109.136	TCP	78	[TCP Dup ACK 2408#7] 45688 → 42669 [ACK] Seq=1 Ack=3958833 Win=786688 Len=0 T
2424	13.138740061	192.168.109.136	172.16.50.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
2425	13.138776797	172.16.50.1	192.168.109.136	TCP	78	[TCP Dup ACK 2408#8] 45688 → 42669 [ACK] Seq=1 Ack=3958833 Win=786688 Len=0 T
2426	13.138864168	192.168.109.136	172.16.50.1	FTP-DATA	2962	FTP Data: 2896 bytes (PASV) (retr files/crab.mp4)
2427	13.138900904	172.16.50.1	192.168.109.136	TCP	78	[TCP Dup ACK 2408#9] 45688 → 42669 [ACK] Seq=1 Ack=3958833 Win=786688 Len=0 T