

Report on File Transfer Protocol Client and Network Setup

Computer Networks 2025/26

Class 04

Casemiro Melo Jorge de Medeiros 202301897
Mykyta Melnykovych 202309183

Summary

This report analyzes the development of a FTP download client application and also analyzes the necessary steps for fully configuring a network that uses one MikroTik router, one MikroTik switch and three debian-based computers. By the end of this process, it is possible to use the developed client to download a file from a FTP server, it can be from an outside server or from Netlab server, from any of the three computers. The client can also be executed from any computer to download files from public FTP servers.

Table of Contents

Introduction	5
Part 1 - Download Application	5
Architecture	5
What happens in a successful download?	5
Part 2 - Network configuration and analysis	6
Experiment 1 - Configure an IP Network	6
Questions	6
Experiment 2 - Implement two bridges in a switch	7
Questions	7
Experiment 3 - Configure a Route in Linux	7
Questions	7
Experiment 4 - Configure a Commercial Router and Implement NAT	8
Questions	8
Experiment 5 - DNS	9
Questions	9
Experiment 6 - TCP Connections	10
Questions	10
Conclusions	11
References	11
Annexes	12
Annex 1.1 - FTP Client successful download	12
Annex 2.1 - Experiment 1: Log	12
Annex 2.2 - Experiment 1: Network	12
Annex 3.1 - Experiment 2: Step 5	13
Annex 3.2 - Experiment 2: Pinging 172.16.90.255 broadcast in tux93	13
Annex 3.3 - Experiment 2: Pinging 172.16.90.255 broadcast in tux94	13
Annex 3.4 - Experiment 2: Pinging 172.16.90.255 broadcast in tux92	13
Annex 3.5 - Experiment 2: Pinging 172.16.91.255 broadcast in tux92	14
Annex 3.6 - Experiment 2: Pinging 172.16.91.255 broadcast in tux93	14
Annex 3.7 - Experiment 2: Pinging 172.16.91.255 broadcast in tux94	15
Annex 3.8 - Experiment 2: Network	15
Annex 4.1 - Experiment 3: Network	15
Annex 4.2 - Experiment 3: Routes on tux92	16
Annex 4.3 - Experiment 3: Routes on tux93	16
Annex 4.4 - Experiment 3: Routes on tux94	17
Annex 4.5 - Experiment 3: Tux93 pinging all network interfaces	17
Annex 4.6 - Experiment 3: Tux94 (if_e1) log when tux93 pings tux92 (post ARP table clean)	18
Annex 4.7 - Experiment 3: Tux94 (if_e2) log when tux93 pings tux92 (post ARP table clean)	18
Annex 5.1 - Experiment 4: Network	19
Annex 5.2 - Experiment 4: Tux93 pinging all interfaces	19

Annex 5.3 - Experiment 4: Tux92 pinging tux93 using router as gateway and no redirect	20
Annex 5.4 - Experiment 4: Tux92 pinging tux93 using tux94 as gateway and redirects	20
Annex 5.5 - Experiment 4: Traceroute from tux92 to tux93 using router as gateway	21
Annex 5.6 - Experiment 4: Traceroute from tux92 to tux93 using tux94 as gateway	22
Annex 5.7 - Experiment 4: Tux93 pinging Netlab FTP Server with NAT on	22
Annex 5.8 - Experiment 4: Tux93 pinging Netlab FTP Server with NAT off	23
Annex 6.1 - Experiment 5: Tux93 pinging google.com	23
Annex 7.1 - Experiment 6: Successful download with all packets	24
Annex 7.2 - Experiment 6: Simultaneous download on tux92 and on tux93	24
Annex 7.3 - Experiment 6: Simultaneous download on tux92 and on tux93	25
Annex 8.1 - download.c	25
Annex 8.2 - parser.h	35
Annex 8.3 - parser.c	36
Annex 8.4 - utils.h	37
Annex 8.5 - utils.c	37
Annex 8.6 - url.h	38
Annex 8.7 - ftp_response.h	39
Annex 8.8 - Makefile	39
Annex 9.1 - Configuration commands	40

Introduction

During the second part of the semester, efforts were directed towards the second practical assignment. The project was divided into two distinct phases. The initial phase was the development of a download application, while the second was the setup of a network in the Netlab room. This network was configured using three computers (tux92, tux93, and tux94) one router and one switch. The final product consists of a tux computer running the application, accessing the Netlab FTP server through the said network, and downloading a file.

Part 1 - Download Application

As previously mentioned, the initial phase of the project consisted in developing a simple program in the C language that implements the File Transfer Protocol (FTP). In summary, the application receives an URL (in the format `ftp://[<user>:<pass>@]<host>/<url-path>`) as an argument, then it parses it, extracting all necessary information. Finally, it establishes a binary, passive-mode connection (authenticated if needed) with the desired host. After the file is successfully transferred, both TCP sockets that were previously opened, one for control and the other for data, are closed. In other words, the connection is closed.

Architecture

The download client is composed of three files, one being the main file and the other two auxiliary files, plus two header files to better organize the FTP response codes and define a struct called *Url* for storing the information extracted from the URL. The three said files are *download.c*, *parser.c* and *utils.c*.

The two auxiliary files are *parser.c* and *utils.c*. The *utils* file contains only one function that is the function used by *parser* to retrieve the IP address of the target server by its host name. Meanwhile, the *parser* file has the logic that reads the URL passed and stores it in the previously said *Url* struct.

The main file is *download.c*. This file contains the *main function*, which handles the TCP socket creation, managing the connection, communication with the server, and file transfer. It also contains the definition of five auxiliary functions, *is_msg_complete*, *get_message*, *send_message*, *parse_ip*, and *get_file_size*. The *get_message* and *send_message* functions are used for communication between client and server, and *is_msg_complete* is used inside *get_message* to check if the message was fully received. The function *parse_ip* is used to compute the new IP address that will be used in the data TCP socket. Lastly, the *get_file_size* function is used before the file transfer itself starts to retrieve the size of the desired file.

What happens in a successful download?

In the event of a successful download, the user can observe the messages exchanged between the client and the server in the terminal. This means that the user can see the messages “transfer complete” and “goodbye”, indicating that the transfer was

successful and that the connection was closed. The received file will be available in the same directory as the project. The FTP packets can be verified in [annex 1](#).

Part 2 - Network configuration and analysis

The subsequent phase of the project consisted in performing several experiments. Each experiment was associated with a specific component of the network setup. For each of these experiments extra steps were done, including capturing Wireshark logs, to better understand how the network operates.

Experiment 1 - Configure an IP Network

The main objective of the experiment was to set IPs of two different computers, namely tux93 and tux 94, and test the communication between them. The log of this experiment is available in [annex 2.1](#) and the network diagram is available in [annex 2.2](#).

Questions

» **What are the ARP packets and what are they used for?**

The Address Resolution Protocol (ARP) packets are used for associating MAC addresses to IP addresses. A request is broadcasted asking who has a certain IP address and a reply is a response containing the MAC address that will be assigned to the IP.

» **What are the MAC and IP addresses of ARP packets and why?**

First the tux93, with MAC ec:75:0c:c2:17:7a and IP 172.16.90.1, broadcasts a request to find out the MAC address of a device with IP 172.16.90.254. Then tux94 sends a response, saying that 172.16.90.254 (tux94) has the following MAC address: ec:75:0c:c2:26:42.

» **What packets does the ping command generate?**

The ping command generates Internet Control Message Protocol (ICMP) packets by default, but it can also generate ARP packets if the IP address is not related to a cached MAC address.

» **What are the MAC and IP addresses of the ping packets?**

Requests are sent from tux93 (MAC ec:75:0c:c2:17:7a IP 172.16.90.1) to tux94. On the other hand, responses are sent from tux94 (MAC ec:75:0c:c2:26:42 IP 172.16.90.254) to tux93.

» **How to determine if a receiving Ethernet frame is ARP, IP, ICMP?**

Using Wireshark, it is possible to check the "Protocol" column. However, it is possible to inspect the Ethernet header

» **How to determine the length of a receiving frame?**

Using Wireshark, it is possible to check the "Length" column. However, similar to determining the protocol, it is possible to inspect the frame header and find the frame length.

» What is the loopback interface and why is it important?

The loopback interface is an interface for locally testing applications that interact with the network (e.g. php servers), inter-process communication

Experiment 2 - Implement two bridges in a switch

The main objective of this experiment was to add a new device, tux92, and set two distinct bridges in the switch, therefore creating two different subnets inside the main network. The final communication diagram can be seen in [annex 3.8](#) and verified in the log available in [annex 3.1](#).

Questions

» How to configure bridgeY0?

In our case, to configure the bridge90, we first created the bridge with */interface bridge add name=bridge90*. After that, considering that tux93 was connected to switch port 16 and tux94 to port 14, we needed to remove the default bridge configuration with */interface bridge port remove [find interface=etherXX]*. Finally, with the ports not belonging to any bridges, we could add them to our new bridge90 with */interface bridge port add bridge=bridge90 interface=etherXX*. It is important to highlight that these commands need to be performed in GTK while being connected to SW CONS.

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

Having two different bridges, bridge90 and bridge91, we have two different broadcast domains. Those two bridges work like two subnets inside the general network we want to set up. Through the logs we can conclude this because when pinging 172.16.90.255 broadcast, tux93 and tux94 can see the ICMP packets, but tux92 cannot see any. However, when pinging 172.16.91.255 broadcast, it occurs the opposite, only tux92 sees ICMP packets. Detailed logs in [annexes](#), also the lack of ICMP packets are easily concluded by the lack of the pink color on the scroll.

Experiment 3 - Configure a Route in Linux

The main objective of this experiment is to transform tux94 into a router. This way, there will be a way to subnet 90 and subnet 91 interact. Network diagram is available in [annex 4.1](#).

Questions

» What routes are there in the tuxes? What are their meaning?

After setting the new routes, we observe that tux92 has a route to network 172.16.90.0/24 using 172.16.91.253 (tux94 if_e2) as gateway, as well as a route to 172.16.91.0/24 through 0.0.0.0, that means that any traffic destined to that LAN, stays within. On tux93 we observe almost the same, but destined to 172.16.91.0/24 through 172.16.90.254 (tux94 if_e1). Lastly, on tux94 we observe that it only contains routes to itself since it is responsible for doing the redirect between networks. All routes are available in [annexes](#).

» What information does an entry of the forwarding table contain?

A forwarding table contains the destination, gateway, genmask, flags, metric and iface. *Destination* is the target network/host, *gateway* is the next hop to reach the *destination* (if it is 0.0.0.0 the packet does not need to change network or it is a direct route), *genmask* is the subnet mask, *flags* is the status/type of a certain route, *metric* is a value assigned to a network path that helps the operating system/router choose the best route when multiple options exist to the same destination, lower metric values indicates a higher preference, *iface* is the interface associated to that route.

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

When inspecting the tux94 (if_e1), we observe that there is a broadcast message from tux93 (ec:75:0c:c2:17:7a) asking who owns IP 172.16.90.254 (ec:75:0c:c2:26:42). After that, when inspecting the tux (if_e2), we observe that tux94 (ec:75:0c:c2:31:93) broadcast an ARP message asking who has IP 172.16.91.1 (a2:05:02:f6:c6:5c). Finally, it is possible to see ICMP packets from tux93 (172.16.90.1) reaching tux92 (172.16.91.1) and it being replied.

Those ARP packets occur because the ARP tables were cleaned. All related logs can be found in [annexes](#).

» What ICMP packets are observed and why?

As previously mentioned, we observe ICMP packets going from tux93 and reaching tux92 on both tux94 network interfaces.

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

In an ICMP packet we have the Source IP address, who is pinging, and the Destination IP address, who is being pinged. Similarly, it happens the same with MAC addresses. There is the Source MAC address, the address of the sending network interface, and the Destination MAC address, the address of the receiving network interface.

The IP addresses ensure communication between different devices in a more general way. In contrast, the MAC addresses are used for local communication, that is, inside the same subnetwork. This can also be observed after each hop, since the MAC addresses change, but not the IP addresses.

Experiment 4 - Configure a Commercial Router and Implement NAT

The main goal of this experiment is to connect our current network to the lab network, so that our computers can access the Netlab FTP server. The network diagram can be found in [annexes](#).

Questions

» How to configure a static route in a commercial router?

For configuring a static route in our router, we first needed to do the router configuration. First, add the router (switch port ether10) to bridge91. Then, set IPs for the router interfaces with commands `/ip address add address=172.16.1.91/24 interface=ether1` and `/ip address add address=172.16.91.254/24 interface=ether2`. Finally, we can run `/ip`

`route add dst-address=172.16.90.0/24 gateway=172.16.91.253`. All these commands must be executed in GTK, the bridge related commands on SW CONS mode and then on ROUTER CONS.

» **What are the paths followed by the packets, with and without ICMP redirect enabled, in the experiments carried out and why?**

Inspecting the packets captured by Wireshark and traceroutes (all available in [annexes](#)), we observe that with ICMP redirect disabled (they appear on Wireshark because they are being rejected) the packets need to travel a longer (tux92 -> router -> tux94 -> tux93) path than with ICMP redirect enabled (tux92 -> tux94 -> tux93). In conclusion, ICMP redirect messages are responsible for informing the source about a shorter path to the destination.

» **How to configure NAT in a commercial router?**

By default it comes activated, but for certainty we can run `/ip firewall nat disable 0` for turning it off and `/ip firewall nat enable 0` for turning it on. Both commands need to be run in GTK.

» **What does NAT do?**

NAT (Network Address Translation) is responsible for translating IP addresses from a local network to a single public IP address, and the opposite. For example, when tux93 (172.16.90.1) tries to reach the FTP server (172.16.1.10), the router changes the Source IP address to its own (172.16.1.91). When the FTP sends a response, it is destined to the router IP address, but then NAT translates it in order to be properly sent to tux93.

This way, there are less public IP addresses in a network and artificially more available IPs, since one single public IP can receive messages on behalf of a lot more.

» **What happens when tuxY3 pings the FTP server with the NAT disabled? Why?**

When the NAT is disabled, the Source IP of an ICMP packet does not get translated to the router's IP address. Also, even if the packet got to the FTP server, it would not reach tux93, since NAT is not there to translate the IP address to the correct recipient.

Experiment 5 - DNS

The main purpose of this experiment is to configure a DNS (Domain Name System) so that we can use human-readable names instead of IP addresses.

Questions

» **How to configure the DNS service in a host?**

For setting the DNS service, it is necessary to check the file `/etc/resolv.conf`. Once we opened the file, we added `nameserver 10.227.20.3`. After that, we are able to use hostname instead of IPs.

» **What packets are exchanged by DNS and what information is transported?**

When pinging a server that our device has never seen before, the DNS message is seen in wireshark. In this message we verify that the device is asking the DNS server for the

IP address associated with the domain google.com. Then, it responds with the IP used in ICMP packets. This log can be accessed in the [annexes](#).

Experiment 6 - TCP Connections

The main objective of this last experiment is to test our developed application and to test the network setup.

Questions

» How many TCP connections are opened by your FTP application?

Two connections are opened - one for exchanging messages with the FTP server, and one for downloading the file in passive mode.

» In what connection is transported the FTP control information?

In the first one, which is used for exchanging messages with the server.

» What are the phases of a TCP connection?

First, there is the connection establishment (3-way handshake), then data transfer (client and server sending packages to each other). Finally, there is connection termination.

» How does the ARQ TCP mechanism work? What are the relevant TCP fields?

What relevant information can be observed in the logs?

ARQ mechanism ensures that the data is delivered by retransmitting lost/corrupted packets. When data is sent the receiver responds with an acknowledgment (ACK). If the sender fails to receive an ACK in time, he retransmits the packet. TCP packets include the sequence number and the acknowledgment number.

In [logs](#) we can see how the connection is being established for both the control and data connections, how messages are sent and connections are closed.

» 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 to the TCP congestion control mechanism?

TCP congestion control mechanism ensures that the sender does not overwhelm the network with too much data. Sender starts with sending a small amount of data, gradually increasing the rate. At a certain threshold the sender starts to increase the rate more slowly to avoid sudden spikes in network congestion. If a packet is lost (e.g. sender receives duplicate ACKs), sender retransmits the lost packet quickly, without waiting for a timeout. After the loss the congestion window is reduced. After a loss, the sending rate is still being slowly increased.

Relevant fields are:

Congestion window (Win) - determines how much data can be sent by sender before waiting for ACKs. It is adjusted based on the current network congestion.

Acknowledgment number(Ack) - is used to confirmed received packets

Sequence number (Seq) - helps track the order of the packets to make sure nothing is lost.

» Is the throughput of a TCP data connections disturbed by the appearance of a

second TCP connection? How?

Yes, it is. We can observe with the help of a [graph](#) that when a second download client is executed in tux92, it makes the total number of packets go up since it was not on the channel limit. However, it is not possible to view in the graph the total speed going down because the second download stayed up for not enough time.

Conclusions

In conclusion, we successfully developed a robust FTP client for Linux-based systems capable of downloading files from both Netlab server and public FTP repositories. Additionally, we were able to successfully configure a complete network infrastructure that uses a router and a switch to manage traffic. Lastly, we used Wireshark to capture packets and inspect what was happening during our setup and see how our FTP client behaved under different circumstances.

References

The only materials we used were the material available in RCOM's moodle.

Annexes

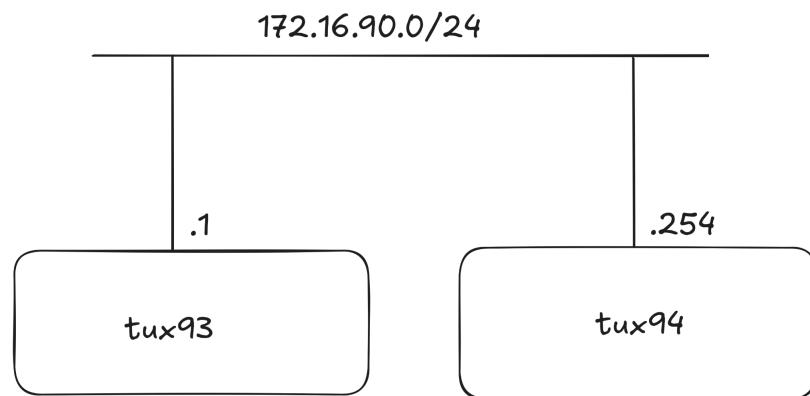
Annex 1.1 - FTP Client successful download

No.	Time	Source	Destination	Protocol	Length	Info
4	0.005244724	172.16.1.10	172.16.90.1	FTP	86	Response: 220 (vsFTPD 3.0.5)
6	0.005332539	172.16.90.1	172.16.1.10	FTP	82	Request: USER anonymous
8	0.005810158	172.16.1.10	172.16.90.1	FTP	100	Response: 331 Please specify the password.
9	0.005865569	172.16.90.1	172.16.1.10	FTP	81	Request: PASS password
10	0.011262290	172.16.1.10	172.16.90.1	FTP	89	Response: 230 Login successful.
11	0.011309666	172.16.90.1	172.16.1.10	FTP	74	Request: TYPE I
12	0.011898997	172.16.1.10	172.16.90.1	FTP	97	Response: 200 Switching to Binary mode.
13	0.011961227	172.16.90.1	172.16.1.10	FTP	82	Request: SIZE /pipe.txt
14	0.012514301	172.16.1.10	172.16.90.1	FTP	75	Response: 213 191
15	0.012576927	172.16.90.1	172.16.1.10	FTP	72	Request: PASV
16	0.013431235	172.16.1.10	172.16.90.1	FTP	115	Response: 227 Entering Passive Mode (172,16,1,10,165,69).
20	0.013998700	172.16.90.1	172.16.1.10	FTP	82	Request: RETR /pipe.txt
21	0.014929987	172.16.1.10	172.16.90.1	FTP	134	Response: 150 Opening BINARY mode data connection for /pipe.t
27	0.058758991	172.16.1.10	172.16.90.1	FTP	90	Response: 226 Transfer complete.
29	0.058882913	172.16.90.1	172.16.1.10	FTP	72	Request: QUIT
30	0.059344481	172.16.1.10	172.16.90.1	FTP	80	Response: 221 Goodbye.

Annex 2.1 - Experiment 1: Log

7	10.62637...	TPLink_c2:17:7a Broadcast	ARP	42 Who has 172.16.90.254? Tell 172.16.90.1
8	10.62654...	TPLink_c2:26:42 TPLink_c2:17:7a	ARP	60 172.16.90.254 is at ec:75:0c:c2:26:42
9	10.62656...	172.16.90.1	172.16.90.254	ICMP 98 Echo (ping) request id=0x127b, seq=1/256, ttl=64 (reply in 10)
10	10.62672...	172.16.90.254	172.16.90.1	ICMP 98 Echo (ping) reply id=0x127b, seq=1/256, ttl=64 (request in 9)
11	11.65234...	172.16.90.1	172.16.90.254	ICMP 98 Echo (ping) request id=0x127b, seq=2/512, ttl=64 (reply in 12)
12	11.65250...	172.16.90.254	172.16.90.1	ICMP 98 Echo (ping) reply id=0x127b, seq=2/512, ttl=64 (request in 11)

Annex 2.2 - Experiment 1: Network



Annex 3.1 - Experiment 2: Step 5

60 28.046264107	172.16.90.1	172.16.90.254	ICMP	98 Echo (ping) request id=0x1ebc, seq=21/5376, ttl=64 (reply in 61)
61 28.0494094647	172.16.90.254	172.16.90.1	ICMP	98 Echo (ping) reply id=0x1ebc, seq=21/5376, ttl=64 (request in 60)
62 29.0492426408	172.16.90.1	172.16.90.254	ICMP	98 Echo (ping) request id=0x1ebc, seq=22/5632, ttl=64 (reply in 63)
63 30.0492454553	172.16.90.254	172.16.90.1	ICMP	98 Echo (ping) reply id=0x1ebc, seq=22/5632, ttl=64 (request in 62)
64 30.0230909051	Routerboardc_3c:95:ca	Spanning-tree-(for-bridges..	STP	00 Echo (ping) request id=0x1ebc, seq=22/5632, ttl=64 (request in 62)
65 31.584237820	0.0.0.0	255.255.255.255	MNDP	159 5678 - 5678 Len=117
66 31.584238359	Routerboardc_3c:95:ca	COP/FTP/DTP/PGP/UDL	COP	93 Device ID: MikroTik Port ID: bridge#9
67 32.025177782	Routerboardc_3c:95:ca	LLDP	110 MAV</4>ad34:1c:95:cb In/bridge#92 Sys=MikroTik Sys=MikroTik RouterOS 6.43.16 (long-term) CRS326-240-25...	
68 32.025177782	Routerboardc_3c:95:ca	Spanning-tree-(for-bridges..	STP	00 RST Root = 32768/0/c4/ad34:1c:95:ca Cost = 0 Port = 0x8001
69 34.029276919	Routerboardc_3c:95:ca	Spanning-tree-(for-bridges..	STP	00 RST Root = 32768/0/c4/ad34:1c:95:ca Cost = 0 Port = 0x8001
70 36.036751634	Routerboardc_3c:95:ca	Spanning-tree-(for-bridges..	STP	00 RST Root = 32768/0/c4/ad34:1c:95:ca Cost = 0 Port = 0x8001
71 38.036946075	Routerboardc_3c:95:ca	Spanning-tree-(for-bridges..	STP	00 RST Root = 32768/0/c4/ad34:1c:95:ca Cost = 0 Port = 0x8001
72 38.036946075	Routerboardc_3c:95:ca	Spanning-tree-(for-bridges..	STP	00 RST Root = 32768/0/c4/ad34:1c:95:ca Cost = 0 Port = 0x8001
73 42.037616026	Routerboardc_3c:95:ca	Spanning-tree-(for-bridges..	STP	00 RST Root = 32768/0/c4/ad34:1c:95:ca Cost = 0 Port = 0x8001
74 44.039901398	Routerboardc_3c:95:ca	Spanning-tree-(for-bridges..	STP	00 RST Root = 32768/0/c4/ad34:1c:95:ca Cost = 0 Port = 0x8001
75 46.042184902	Routerboardc_3c:95:ca	Spanning-tree-(for-bridges..	STP	00 RST Root = 32768/0/c4/ad34:1c:95:ca Cost = 0 Port = 0x8001
76 47.005973091	172.16.90.1	172.16.90.1	ICMP	98 Echo (ping) request id=0x1ebd, seq=1/256, ttl=64 (no response found)
77 47.005973091	Routerboardc_3c:95:ca	Spanning-tree-(for-bridges..	STP	00 Echo (ping) reply id=0x1ebd, seq=1/256, ttl=64 (no response found)
78 48.0114261564	172.16.90.1	172.16.90.1	ICMP	98 Echo (ping) request id=0x1ebd, seq=2/512, ttl=64 (no response found!)
79 49.036270812	172.16.90.1	172.16.90.1	ICMP	98 Echo (ping) request id=0x1ebd, seq=3/768, ttl=64 (no response found!)
80 50.0467471751	Routerboards_1c:95:ca	Spanning-tree-(for-bridges..	STP	00 RST Root = 32768/0/c4/ad34:1c:95:ca Cost = 0 Port = 0x8001
81 50.0467471751	172.16.90.1	172.16.90.1	ICMP	159 5678 - 5678 Len=117
82 50.0467471751	172.16.90.254	172.16.90.1	ICMP	126 Destination unreachable (Host unreachable)
83 50.047370669	172.16.90.254	172.16.90.1	ICMP	126 Destination unreachable (Host unreachable)
84 50.047571721	172.16.90.1	172.16.90.1	ICMP	98 Echo (ping) request id=0x1ebd, seq=4/1024, ttl=64 (no response found!)
85 51.054270328	172.16.90.1	172.16.90.1	ICMP	98 Echo (ping) request id=0x1ebd, seq=5/1280, ttl=64 (no response found!)

Annex 3.2 - Experiment 2: Pinging 172.16.90.255 broadcast in tux93

64 76.508239855	172.16.90.1	172.16.90.255	ICMP	88 Echo (ping) Request id=0x1fee, seq=1/256, ttl=64 (no response found!)
65 70.560492129	172.16.90.254	172.16.90.1	ICMP	88 Echo (ping) reply id=0x1fee, seq=1/256, ttl=64
66 71.519548834	172.16.90.1	172.16.90.255	ICMP	88 Echo (ping) request id=0x1fee, seq=2/512, ttl=64 (no response found!)
67 71.519518853	172.16.90.254	172.16.90.1	ICMP	88 Echo (ping) request id=0x1fee, seq=2/512, ttl=64
68 72.072293799	Routerboardc_3c:95:ca	Spanning-tree-(for-bridges..	STP	00 RST Root = 32768/0/c4/ad34:1c:95:ca Cost = 0 Port = 0x8001
69 72.072293799	172.16.90.1	172.16.90.255	ICMP	88 Echo (ping) request id=0x1fee, seq=3/768, ttl=64 (no response found!)
70 72.072293799	172.16.90.254	172.16.90.1	ICMP	88 Echo (ping) reply id=0x1fee, seq=3/768, ttl=64
71 73.057238191	172.16.90.1	172.16.90.255	ICMP	88 Echo (ping) request id=0x1fee, seq=4/1024, ttl=64 (no response found!)
72 73.057238191	172.16.90.254	172.16.90.1	ICMP	88 Echo (ping) reply id=0x1fee, seq=4/1024, ttl=64
73 74.074421998	Routerboardc_3c:95:ca	Spanning-tree-(for-bridges..	STP	00 RST Root = 32768/0/c4/ad34:1c:95:ca Cost = 0 Port = 0x8001
74 74.501225883	172.16.90.1	172.16.90.255	ICMP	88 Echo (ping) request id=0x1fee, seq=5/1280, ttl=64 (no response found!)
75 74.501473982	172.16.90.254	172.16.90.1	ICMP	88 Echo (ping) reply id=0x1fee, seq=5/1280, ttl=64

Annex 3.3 - Experiment 2: Pinging 172.16.90.255 broadcast in tux94

43 53.4800007386	172.16.90.1	172.16.90.255	ICMP	98 Echo (ping) request id=0x1f0e, seq=1/256, ttl=64 (no response found!)
44 53.4899497272	172.16.90.254	172.16.90.1	ICMP	98 Echo (ping) reply id=0x1f0e, seq=1/256, ttl=64
45 53.4899497272	172.16.90.1	172.16.90.255	ICMP	98 Echo (ping) request id=0x1f0e, seq=2/512, ttl=64
46 54.0499914575	172.16.90.1	172.16.90.255	ICMP	98 Echo (ping) request id=0x1f0e, seq=2/512, ttl=64 (no response found!)
47 54.4999949498	172.16.90.254	172.16.90.1	ICMP	98 Echo (ping) reply id=0x1f0e, seq=2/512, ttl=64
48 55.5239045451	172.16.90.1	172.16.90.255	ICMP	98 Echo (ping) request id=0x1f0e, seq=3/768, ttl=64 (no response found!)
49 55.523914172	172.16.90.254	172.16.90.1	ICMP	98 Echo (ping) reply id=0x1f0e, seq=3/768, ttl=64

Annex 3.4 - Experiment 2: Pinging 172.16.90.255 broadcast in tux92

No.	Time	Source	Destination	Protocol	Length Info
76 139.053690788		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
77 139.053690788		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
78 139.053690788		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
79 139.053690788		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
80 139.053690788		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
81 139.053690788		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
82 139.053690788		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
83 139.053690788		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
84 139.053690788		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
85 140.104130573		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
86 140.104130573		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
87 140.104130573		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
88 140.104130573		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
89 140.104130573		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
90 140.104130573		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
91 154.119616065		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
92 156.121873313		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
93 156.121873313		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
94 156.122674848		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
95 162.128528794		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
96 164.139729898		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
97 164.139729898		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
98 166.135216978		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
99 178.137379477		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
100 178.137379477		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
101 178.137379477		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
102 178.137379477		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
103 178.137379477		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
104 178.137379477		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
105 178.137379477		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
106 184.1352658489		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
107 188.154620652		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
108 188.156247756		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
109 188.156247756		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
110 188.156247756		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
111 194.160679085		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
112 194.160679085		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
113 194.160679085		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
114 199.372922627		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
115 199.372923943		Routerboardc_3c:95:cb	COP/FTP/DTP/PGP/UDL	COP	93 Device ID: MikroTik Port ID: bridge#91
116 199.372923943		Routerboardc_3c:95:cb	LLDP_Multicast	LLDP	110 MAV</4>ad34:1c:95:cb In/bridge#91 128 Sys=MikroTik Sys=MikroTik RouterOS 6.43.16 (long-term) CRS326-240-25...
117 199.372923943		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
118 204.165212247		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
119 204.165212247		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
120 208.165237048		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8001
121 208.165237048		Routerboardc_3c:95:cb	Spanning-tree-(for-bridges..	STP	68 RST Root = 32768/0/c4/ad34:1c:95:cb Cost = 0 Port = 0x8

Annex 3.5 - Experiment 2: Pinging 172.16.91.255 broadcast in tux92

No.	Time	Source	Destination	Protocol	Length	Info
34	47.323951832	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
35	49.641386893	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
36	49.643591141	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
37	49.643604778	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
38	44.648638355	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
39	46.659241293	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
40	46.319235799	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=1/256, ttl=64 (no response found!)
41	47.333958193	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=2/252, ttl=64 (no response found!)
42	48.333958193	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
43	48.357819169	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=3/768, ttl=64 (no response found!)
44	49.381054934	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=4/1024, ttl=64 (no response found!)
45	50.349946060	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
46	50.405033448	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=5/1280, ttl=64 (no response found!)
47	51.429857264	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=6/1536, ttl=64 (no response found!)
48	51.429857264	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
49	52.459839669	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=7/1792, ttl=64 (no response found!)
50	53.477667282	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=8/2048, ttl=64 (no response found!)
51	54.643816689	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
52	54.501835931	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=9/2304, ttl=64 (no response found!)
53	55.522684444	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=10/2560, ttl=64 (no response found!)
54	56.522684444	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
55	56.549946060	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=11/2816, ttl=64 (no response found!)
56	57.537042087	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=12/3072, ttl=64 (no response found!)
57	58.53703922	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
58	59.53703922	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=13/3228, ttl=64 (no response found!)
59	59.621053975	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=14/3584, ttl=64 (no response found!)
60	60.649959376	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
61	61.649959376	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=15/3840, ttl=64 (no response found!)
62	61.649959376	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
63	62.657053337	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=16/4608, ttl=64 (no response found!)
64	62.693856326	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=17/4382, ttl=64 (no response found!)
65	63.717958345	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
66	64.741053596	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=18/5120, ttl=64 (no response found!)
67	64.741053596	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
68	65.765957766	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=19/4864, ttl=64 (no response found!)
69	66.789965017	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=20/5120, ttl=64 (no response found!)
70	70.868505572	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=21/5376, ttl=64 (no response found!)
71	71.813946235	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=22/5632, ttl=64 (no response found!)
72	72.863960829	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
73	73.864074977	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=23/5888, ttl=64 (no response found!)
74	74.861949370	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=24/6144, ttl=64 (no response found!)
75	75.865163518	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
76	76.885055772	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=25/6400, ttl=64 (no response found!)
77	77.909941913	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=26/6656, ttl=64 (no response found!)
78	78.921053777	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
79	79.933941996	172.16.91.1	172.16.91.255	ICMP	98	Echo (ping) request id=0x3e05, seq=27/6912, ttl=64 (no response found!)
80	84.969660882	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
81	85.971014314	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001

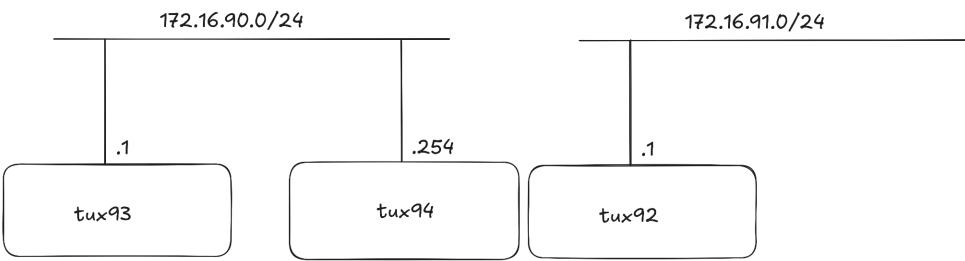
Annex 3.6 - Experiment 2: Pinging 172.16.91.255 broadcast in tux93

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
2	1.653795310	172.16.99.254	239.255.255.256	UDP	649	48786 - 3762 Len=567
3	1.653867084	172.16.99.254	224.0..22	IGMPv3	60	Membership Report / Leave group 239.255.255.256
4	1.749375744	172.16.99.254	239.255.255.256	IGMPv3	649	48786 - 3762 Len=567
5	1.757437744	172.16.99.1	224.0..22	IGMPv3	54	Membership Report / Leave group 239.255.255.256
6	2.692210777	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
7	2.708876734	172.16.99.1	224.0..22	IGMPv3	2240	56725 - 3762 Len=567
8	2.713376734	172.16.99.1	239.255.255.256	IGMPv3	649	48786 - 3762 Len=567
9	2.734985403	172.16.99.254	224.0..22	IGMPv3	60	Membership Report / Leave group 239.255.255.256
10	2.744549784	172.16.99.1	224.0..22	IGMPv3	54	Membership Report / Leave group 239.255.255.256
11	2.745088192	172.16.99.254	224.0..22	IGMPv3	60	Membership Report / Leave group 239.255.255.256
12	2.745088192	172.16.99.254	239.255.255.256	IGMPv3	649	48786 - 3762 Len=567
13	3.262658589	172.16.99.254	224.0..22	IGMPv3	60	Membership Report / Leave group 239.255.255.256
14	4.084423227	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
15	4.110886607	172.16.99.1	224.0..22	IGMPv3	60	Membership Report / Leave group 239.255.255.256
16	4.110886607	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
17	8.068854671	172.16.99.1	224.0..22	IGMPv3	60	Membership Report / Leave group 239.255.255.256
18	10.811864380	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
19	12.821508080	172.16.99.1	224.0..22	IGMPv3	60	Membership Report / Leave group 239.255.255.256
20	22.022150808	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
21	22.350896495	172.16.99.254	224.0..22	IGMPv3	60	Membership Report / Leave group 239.255.255.256
22	22.350896495	172.16.99.254	239.255.255.256	IGMPv3	649	48786 - 3762 Len=567
23	22.350896495	172.16.99.254	224.0..22	IGMPv3	60	Membership Report / Leave group 239.255.255.256
24	22.350896495	172.16.99.254	239.255.255.256	IGMPv3	649	48786 - 3762 Len=567
25	24.054274907	0.0..0.0	255.255.255.256	MNDP	150	5678 - 5678 Len=117
25	25.15.054274907	Routerboarddc_1c:95:c9	CDP/VT/PagU/WLDD	CDP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
26	25.15.05436334	Routerboarddc_1c:95:c9	LLDP_Multicast	LLDP	118	MAC/C4:ad:34:1c:95:c9 In/bridge09 129 Sys=MikroTik RouterOS 0.43..16 (long-term) CRS326-240-25-26
27	26.617722943	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
28	26.617722943	172.16.99.1	224.0..22	IGMPv3	60	Membership Report / Leave group 239.255.255.256
29	26.617722943	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
30	26.617722943	172.16.99.1	239.255.255.256	IGMPv3	649	48786 - 3762 Len=567
31	26.622436659	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
32	26.622436659	172.16.99.1	224.0..22	IGMPv3	60	Membership Report / Leave group 239.255.255.256
33	28.630275617	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
34	34.832482775	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
35	35.324767216	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
36	36.368391984	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
38	38.041345469	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
39	39.4053957275	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
40	40.4053957275	172.16.99.1	224.0..22	IGMPv3	60	Membership Report / Leave group 239.255.255.256
41	41.404308859	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
42	44.019891257	172.16.99.254	224.0..251	MDNS	168	Standard query 0x0000 PTR _ftp._tcp.local, "QM" question PTR _nfs._tcp.local, "QM" question PTR _afpovertcp..
43	45.049900135	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
44	45.049900135	172.16.99.1	224.0..22	IGMPv3	60	Membership Report / Leave group 239.255.255.256
45	45.049900135	Routerboarddc_1c:95:c9	Spanning-tree-(for-bridges..	STP	60	RST. Root = 32768/0/c4/ad:34:1c:95:c9 Cost = 0 Port = 0x8001
46	45.049900135	172.16.99.1	239.255.255.256	IGMP		

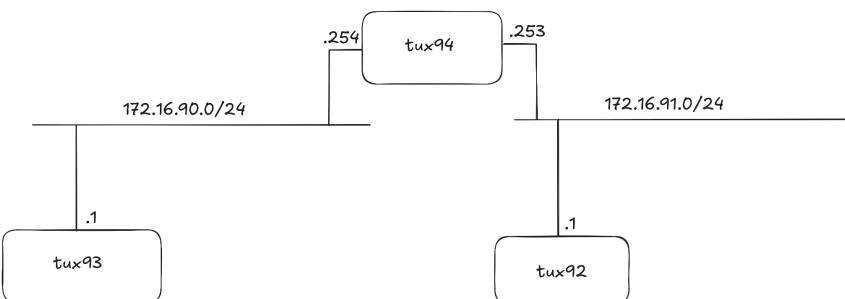
Annex 3.7 - Experiment 2: Pinging 172.16.91.255 broadcast in tux94

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
2	0.002113393	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
3	0.002113393	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
4	0.277215818	172.16.90.254	239.255.255.255	IGMPv3	649	58725 - 3782 Len=687
5	5.439056737	172.16.90.254	224.0.0.22	IGMPv3	54	Membership Report / Leave group 239.255.255.255
6	5.831046934	172.16.90.254	224.0.0.22	IGMPv3	54	Membership Report / Leave group 239.255.255.255
7	6.080047570	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
8	6.080047570	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
9	8.046535010	0.0.0.0	255.255.255.255	MNDP	159	5878 - 5678 Len=117
10	8.046535251	Routerboardc_1c:95:ca	CDP/VT/PTP/Agp/UDL	CDP	93	Device ID: MikroTik Port ID: bridge#0
11	8.046564195	Routerboardc_1c:95:ca	LLDP_Multicast	LLDP	110	MAC:4:8d:34:1c:95:ca In:bridge#0 129 Sys=MikroTik Sys=MikroTik RouterOS 6.43.16 (long-term) CRS326-240-25
12	8.046564195	Routerboardc_1c:95:ca	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
13	12.013292299	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
14	14.015594917	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
15	16.017722994	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
16	18.020002299	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
17	20.021412299	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
18	22.023613698	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
19	24.025831857	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
20	26.028051857	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
21	28.030271336	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
22	30.032477247	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
23	32.034695292	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
24	34.036915292	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
25	36.039374043	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
26	37.612025114	172.16.90.254	224.0.0.251	MNDP	166	Standard query 0x0000 PTR _ftp._tcp.local, "QM" question PTR _ftp._tcp.local, "QM" question PTR _afpovertcp..
27	38.041954716	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
28	40.044274716	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
29	42.045762163	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
30	44.037850757	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
31	46.039270415	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
32	48.041580348	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
33	50.042973979	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
34	52.045180348	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
35	54.047385994	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
36	56.050005994	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
37	58.051088413	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
38	60.054822118	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
39	62.056222814	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
40	64.058522814	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
41	66.060655098	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
42	68.047829177	0.0.0.0	255.255.255.255	MNDP	159	5878 - 5678 Len=117
43	68.047829598	Routerboardc_1c:95:ca	CDP/VT/PTP/Agp/UDL	CDP	93	Device ID: MikroTik Port ID: bridge#0
44	70.050122127	Routerboards_1c:95:cb	LLDP_Multicast	LLDP	110	MAC:4:8d:34:1c:95:ca In:bridge#0 129 Sys=MikroTik Sys=MikroTik RouterOS 6.43.16 (long-term) CRS326-240-25
45	68.062862127	Routerboards_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
46	70.065909214	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
47	72.067361826	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
48	74.069969385	Routerboardc_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002
49	76.071725021	Routerboards_1c:95:cb	Spanning-tree-(for-bridges_	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0xb002

Annex 3.8 - Experiment 2: Network



Annex 4.1 - Experiment 3: Network



Annex 4.2 - Experiment 3: Routes on tux92

The screenshot shows a terminal window titled "netedu@tux92: ~". It has three tabs open: "netedu@tux92: ~", "netedu@tux93: ~", and "netedu@tux94: ~". The current tab is "netedu@tux92: ~". The terminal displays the following commands and output:

```
netedu@tux92: $ ifconfig if_e1 172.16.91.1/24
[sudo] password for netedu:
SIOCSIFADDR: No such device
if_e1: ERROR while getting interface flags: No such device
SIOCSIFNETMASK: No such device
netedu@tux92: $ ifconfig enp2s0 172.16.91.1/24
netedu@tux92: $ route add -net 172.16.90.0/24 gw 172.16.91.253
netedu@tux92: $ route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref  Use Iface
0.0.0.0          10.227.20.254  0.0.0.0        UG    100   0      0 if_mng
10.227.20.0     0.0.0.0        255.255.255.0  U     100   0      0 if_mng
172.16.90.0     172.16.91.253  255.255.255.0  UG    0     0      0 enp2s0
172.16.91.0     0.0.0.0        255.255.255.0  U     0     0      0 enp2s0
netedu@tux92: $
```

Annex 4.3 - Experiment 3: Routes on tux93

The screenshot shows a terminal window titled "netedu@tux93: ~". It has three tabs open: "netedu@tux92: ~", "netedu@tux93: ~", and "netedu@tux94: ~". The current tab is "netedu@tux93: ~". The terminal displays the following commands and output:

```
netedu@tux93: $ ifconfig if_e1 172.16.90.1/24
[sudo] password for netedu:
netedu@tux93: $ route add -net 172.16.91.0/24 gw 172.16.90.254
netedu@tux93: $ route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref  Use Iface
0.0.0.0          10.227.20.254  0.0.0.0        UG    100   0      0 if_mng
10.227.20.0     0.0.0.0        255.255.255.0  U     100   0      0 if_mng
172.16.90.0     0.0.0.0        255.255.255.0  U     0     0      0 if_e1
172.16.91.0     172.16.90.254  255.255.255.0  UG    0     0      0 if_e1
netedu@tux93: $
```

Annex 4.4 - Experiment 3: Routes on tux94

```

netedu@tux94:~$ ifconfig if_e1 172.16.90.254/24
[sudo] password for netedu:
netedu@tux94:~$ ifconfig if_e2 172.16.91.253/24
netedu@tux94:~$ sudo sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
netedu@tux94:~$ sudo sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=0
net.ipv4.icmp_echo_ignore_broadcasts = 0
netedu@tux94:~$ route -n
Kernel IP routing table
Destination     Gateway      Genmask      Flags Metric Ref    Use Iface
0.0.0.0         10.227.20.254  0.0.0.0      UG   100    0        0 if_mng
10.227.20.0     0.0.0.0      255.255.255.0 U     100    0        0 if_mng
172.16.90.0     0.0.0.0      255.255.255.0 U     0       0        0 if_e1
172.16.91.0     0.0.0.0      255.255.255.0 U     0       0        0 if_e2
netedu@tux94:~$ 

```

Annex 4.5 - Experiment 3: Tux93 pinging all network interfaces

No.	Time	Source	Destination	Protocol	Length	Info
24	33.370847924	netedu@tux93:~	172.16.90.254	ICMP	80	Echo (ping) reply id=0x219a, seq=7/192, ttl=64 (request in 23)
25	34.481538216	netedu@tux93:~	172.16.90.254	ICMP	80	Echo (ping) request id=0x219a, seq=8/192, ttl=64 (reply in 27)
27	14.4820316494	172.16.90.254	172.16.90.1	ICMP	80	Echo (ping) reply id=0x219a, seq=9/192, ttl=64 (request in 26)
28	16.811798449	Routerboard_3c:95:ca	Spanning-tree-(For-bridges).00	STP	60	Routing table cost = 0 port = 0x0001
29	17.811798449	Routerboard_3c:95:ca	Spanning-tree-(For-bridges).00	STP	60	Routing table cost = 0 port = 0x0001
30	18.722488267	172.16.90.253	172.16.90.1	ICMP	80	Echo (ping) request id=0x219b, seq=1/192, ttl=64 (reply in 31)
31	19.722488267	172.16.90.253	172.16.90.1	ICMP	80	Echo (ping) reply id=0x219b, seq=2/192, ttl=64 (request in 30)
32	19.745924653	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219b, seq=3/192, ttl=64 (reply in 33)
33	19.746016439	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219b, seq=4/192, ttl=64 (request in 32)
34	20.746016439	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219b, seq=5/192, ttl=64 (reply in 35)
35	20.769848699	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219b, seq=6/192, ttl=64 (request in 34)
36	20.770867341	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219b, seq=7/192, ttl=64 (reply in 37)
37	21.748626871	172.16.91.253	172.16.90.1	ICMP	80	Echo (ping) reply id=0x219b, seq=8/192, ttl=64 (request in 36)
38	21.748626871	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219b, seq=9/192, ttl=64 (reply in 41)
39	22.748626871	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219b, seq=10/192, ttl=64 (request in 40)
40	23.748626871	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219b, seq=11/192, ttl=64 (reply in 43)
41	23.842669776	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219b, seq=12/192, ttl=64 (request in 42)
42	24.842669776	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219b, seq=13/192, ttl=64 (reply in 45)
43	24.866883138	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219b, seq=14/192, ttl=64 (request in 44)
44	25.889848618	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219b, seq=15/192, ttl=64 (reply in 47)
45	26.889848618	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219b, seq=16/192, ttl=64 (request in 46)
46	26.890833348	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219b, seq=17/192, ttl=64 (reply in 45)
47	26.890833348	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219b, seq=18/192, ttl=64 (request in 47)
48	26.890833348	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219b, seq=19/192, ttl=64 (reply in 46)
49	26.890833348	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219b, seq=20/192, ttl=64 (request in 48)
50	26.890833348	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219b, seq=21/192, ttl=64 (reply in 47)
51	26.914666539	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219b, seq=22/192, ttl=64 (request in 50)
52	27.937363657	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219b, seq=23/192, ttl=64 (reply in 53)
53	27.938034451	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219b, seq=24/192, ttl=64 (request in 52)
54	28.938034451	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219b, seq=25/192, ttl=64 (reply in 54)
55	28.961832557	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219b, seq=26/192, ttl=64 (request in 55)
56	28.961832557	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219b, seq=27/192, ttl=64 (reply in 56)
57	29.969835708	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219b, seq=28/192, ttl=64 (request in 58)
58	29.969835713	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219b, seq=29/192, ttl=64 (reply in 57)
59	29.969835713	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219b, seq=30/192, ttl=64 (request in 59)
60	31.009832259	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219b, seq=31/192, ttl=64 (reply in 61)
61	31.010977698	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219b, seq=32/192, ttl=64 (request in 60)
62	31.010977698	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219b, seq=33/192, ttl=64 (reply in 63)
63	31.010977698	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219b, seq=34/192, ttl=64 (request in 62)
64	34.138417332	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219c, seq=5/192, ttl=64 (reply in 65)
65	34.13876208	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219c, seq=6/192, ttl=64 (request in 64)
66	35.137844696	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219c, seq=7/192, ttl=64 (reply in 67)
67	35.137844696	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219c, seq=8/192, ttl=64 (request in 66)
68	35.187694414	ec:75:0c:c2:26:42	ec:75:0c:c2:26:42	ARP	64	Who has 172.16.90.1 Tell 172.16.90.254
69	35.187715496	ec:75:0c:c2:26:42	ec:75:0c:c2:26:42	ARP	64	172.16.90.1 is at ec:75:0c:c2:26:42
70	36.180533853	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219d, seq=5/192, ttl=64 (reply in 71)
71	36.180533853	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219d, seq=6/192, ttl=64 (request in 70)
72	36.180533853	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219d, seq=7/192, ttl=64 (reply in 72)
73	37.185831173	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219d, seq=8/192, ttl=64 (request in 74)
74	37.186291389	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219d, seq=9/192, ttl=64 (reply in 73)
75	38.209842223	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219d, seq=10/192, ttl=64 (request in 76)
76	38.209842223	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219d, seq=11/192, ttl=64 (reply in 77)
77	38.210281463	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219d, seq=12/192, ttl=64 (request in 78)
78	38.210281463	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219d, seq=13/192, ttl=64 (reply in 79)
79	39.234286734	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219d, seq=14/192, ttl=64 (request in 80)
80	40.257333589	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219e, seq=5/192, ttl=64 (reply in 82)
81	40.257333589	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219e, seq=6/192, ttl=64 (request in 81)
82	40.258270263	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219e, seq=7/192, ttl=64 (reply in 83)
83	40.258270263	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219e, seq=8/192, ttl=64 (request in 84)
84	40.385058418	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219e, seq=9/192, ttl=64 (reply in 85)
85	41.281832687	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) reply id=0x219e, seq=10/192, ttl=64 (request in 86)
86	41.281832687	172.16.90.1	172.16.91.253	ICMP	80	Echo (ping) request id=0x219e, seq=11/192, ttl=64 (reply in 85)
87	42.454655854	Routerboard_3c:95:ca	Spanning-tree-(For-bridges).00	STP	60	Routing table cost = 0 port = 0x0001
88	42.454655854	Routerboard_3c:95:ca	Spanning-tree-(For-bridges).00	STP	60	Routing table cost = 0 port = 0x0001

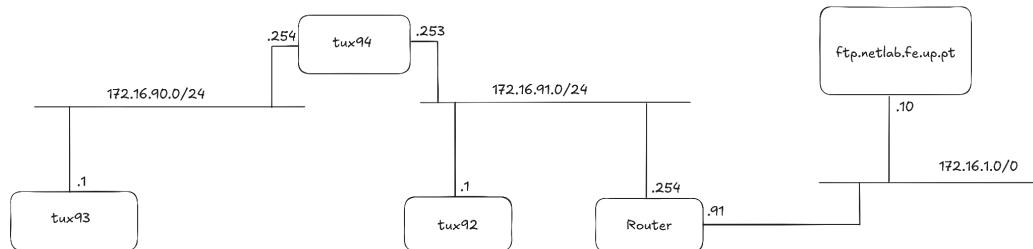
Annex 4.6 - Experiment 3: Tux94 (if_e1) log when tux93 pings tux92 (post ARP table clean)

No.	Time	Source	Destination	Protocol	Length	Info
77	11:22:53.7238381	Routerboarddc_1c:95:05	Spanning-tree-(for-brides...	STP	60	RST_Root = 32768/0/c4:ad:34:1c:95:05 Cost = 0 Port = 0x8002
78	11:41:01941482	Routerboarddc_1c:95:05	Spanning-tree-(for-brides...	STP	60	RST_Root = 32768/0/c4:ad:34:1c:95:05 Cost = 0 Port = 0x8002
79	11:41:01941482	Routerboarddc_1c:95:05	Spanning-tree-(for-brides...	STP	60	RST_Root = 32768/0/c4:ad:34:1c:95:05 Cost = 0 Port = 0x8002
80	11:41:01941482	ec:75:0c:c2:17:78	Routerboarddc_1c:95:05	ARP	60	Who has 172.16.99.1 is at ec:75:0c:c2:17:78
81	117.2739606868	ec:75:0c:c2:26:42	ec:75:0c:c2:17:78	ARP	42	172.16.99.1 is at ec:75:0c:c2:17:78
82	117.274898481	172.16.99.1	172.16.99.1	ICMP	88	Echo (ping) request id=0x0c75, seq=1/256, ttl=64 (reply in 83)
83	117.274577677	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0c75, seq=1/256, ttl=64 (request in 82)
84	118.296958984	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) request id=0x0c75, seq=2/252, ttl=64 (reply in 88)
85	118.296958984	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0c75, seq=2/252, ttl=64 (request in 85)
86	118.297215324	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) request id=0x0c75, seq=3/256, ttl=64 (reply in 87)
87	118.320958948	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0c75, seq=3/256, ttl=64 (request in 87)
88	118.321194546	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) request id=0x0c75, seq=4/256, ttl=64 (reply in 89)
89	118.344951441	Routerboarddc_1c:95:05	Spanning-tree-(for-brides...	STP	60	RST_Root = 32768/0/c4:ad:34:1c:95:05 Cost = 0 Port = 0x8002
90	118.344951441	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) request id=0x0c75, seq=4/1024, ttl=64 (reply in 91)
91	120.345167160	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0c75, seq=4/1024, ttl=64 (request in 90)
92	121.369225961	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) request id=0x0c75, seq=5/1280, ttl=64 (reply in 93)
93	122.368330337	Routerboarddc_1c:95:05	Spanning-tree-(for-brides...	STP	60	RST_Root = 32768/0/c4:ad:34:1c:95:05 Cost = 0 Port = 0x8002
94	122.368330337	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) request id=0x0c75, seq=5/1280, ttl=64 (request in 92)
95	122.405292196	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0c75, seq=9/2384, ttl=64 (request in 95)
96	122.405292196	ec:75:0c:c2:26:42	ec:75:0c:c2:17:78	ARP	42	Who has 172.16.99.1 is at ec:75:0c:c2:17:78
97	122.405292196	ec:75:0c:c2:17:78	172.16.99.1	ARP	60	172.16.99.1 is at ec:75:0c:c2:17:78
98	122.41686718	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) request id=0x0c75, seq=7/1792, ttl=64 (reply in 99)
99	122.41686718	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0c75, seq=7/1792, ttl=64 (request in 99)
100	123.417104431	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) request id=0x0c75, seq=8/1792, ttl=64 (reply in 100)
101	123.417104431	Routerboarddc_1c:95:05	Spanning-tree-(for-brides...	STP	60	RST_Root = 32768/0/c4:ad:34:1c:95:05 Cost = 0 Port = 0x8002
102	124.440933299	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) request id=0x0c75, seq=8/2048, ttl=64 (reply in 103)
103	124.441188186	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0c75, seq=8/2048, ttl=64 (request in 102)
104	124.441188186	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) request id=0x0c75, seq=9/2384, ttl=64 (reply in 105)
105	125.465292196	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0c75, seq=9/2384, ttl=64 (request in 104)
106	126.4532798977	Routerboarddc_1c:95:05	Spanning-tree-(for-brides...	STP	60	RST_Root = 32768/0/c4:ad:34:1c:95:05 Cost = 0 Port = 0x8002
107	126.488917933	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) request id=0x0c75, seq=10/2560, ttl=64 (reply in 108)
108	126.488917933	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0c75, seq=10/2560, ttl=64 (request in 107)
109	127.512923522	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) request id=0x0c75, seq=11/2816, ttl=64 (reply in 109)
110	127.513199491	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0c75, seq=11/2816, ttl=64 (request in 109)
111	127.513199491	Routerboarddc_1c:95:05	Spanning-tree-(for-brides...	STP	60	RST_Root = 32768/0/c4:ad:34:1c:95:05 Cost = 0 Port = 0x8002
112	128.512923522	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) request id=0x0c75, seq=12/3072, ttl=64 (reply in 113)
113	128.537292181	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0c75, seq=12/3072, ttl=64 (request in 112)
114	129.566987464	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) request id=0x0c75, seq=13/3236, ttl=64 (reply in 115)
115	129.566987464	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0c75, seq=13/3236, ttl=64 (request in 114)
116	129.566987464	Routerboarddc_1c:95:05	Spanning-tree-(for-brides...	STP	60	RST_Root = 32768/0/c4:ad:34:1c:95:05 Cost = 0 Port = 0x8002
117	132.639456526	Routerboarddc_1c:95:05	Spanning-tree-(for-brides...	STP	60	RST_Root = 32768/0/c4:ad:34:1c:95:05 Cost = 0 Port = 0x8002
118	133.321421204	172.16.99.254	239.255.255.250	UDP	640	44722 - 3782 Len=607
119	133.343681269	172.16.99.254	239.255.255.250	UDPMV3	54	Membership Report / Leave group 239.255.255.250
120	133.343681269	172.16.99.254	239.255.255.250	UDPMV3	54	Membership Report / Leave group 239.255.255.250
121	133.469886153	172.16.99.1	239.255.255.250	UDPMV3	54	Membership Report / Leave group 239.255.255.250
122	133.469886153	172.16.99.1	239.255.255.250	UDPMV3	54	Membership Report / Leave group 239.255.255.250
123	133.886830296	172.16.99.1	239.255.255.250	UDPMV3	54	Membership Report / Leave group 239.255.255.250
124	134.039986456	Routerboarddc_1c:95:05	Spanning-tree-(for-brides...	STP	60	RST_Root = 32768/0/c4:ad:34:1c:95:05 Cost = 0 Port = 0x8002
125	134.042873897	172.16.99.254	239.255.255.250	UDPMV3	54	Membership Report / Leave group 239.255.255.250
126	134.042873897	172.16.99.254	239.255.255.250	UDPMV3	54	Membership Report / Leave group 239.255.255.250
127	134.042873897	172.16.99.254	239.255.255.250	UDPMV3	54	Membership Report / Leave group 239.255.255.250
128	134.042827179	Routerboarddc_1c:95:05	CDP/VT/DT/PoP/UDLD	CDP	93	Device ID: MikroTik Port: bridge0
129	134.042827179	Routerboarddc_1c:95:05	LLDP_Multicast	LLDP	110	HA:c4:ad:34:1c:95:05/ln/bridge00 129 Sys=MikroTik RouterOS 6.43.16 (long-term) CRS326-24G-2S+2T
130	134.042827179	172.16.99.254	239.255.255.250	UDP	640	5266 - 3782 Len=607

Annex 4.7 - Experiment 3: Tux94 (if_e2) log when tux93 pings tux92 (post ARP table clean)

No.	Time	Source	Destination	Protocol	Length	Info
77	11:22:53.7238381	Routerboarddc_1c:95:05	Spanning-tree-(for-brides...	STP	60	RST_Root = 32768/0/c4:ad:34:1c:95:05 Cost = 0 Port = 0x8002
78	11:41:01941482	Routerboarddc_1c:95:05	Spanning-tree-(for-brides...	STP	60	RST_Root = 32768/0/c4:ad:34:1c:95:05 Cost = 0 Port = 0x8002
79	11:41:01941482	Routerboarddc_1c:95:05	Spanning-tree-(for-brides...	STP	60	RST_Root = 32768/0/c4:ad:34:1c:95:05 Cost = 0 Port = 0x8002
80	11:41:01941482	ec:75:0c:c2:31:93	Routerboarddc_1c:95:05	ARP	42	Who has 172.16.99.1 is at ec:75:0c:c2:31:93
81	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
82	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
83	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
84	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
85	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
86	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
87	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
88	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
89	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
90	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
91	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
92	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
93	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
94	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
95	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
96	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
97	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
98	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
99	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
100	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
101	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
102	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
103	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
104	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
105	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
106	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
107	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
108	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
109	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
110	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
111	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
112	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	60	172.16.99.1 is at a8:05:02:f8:c6:5c
113	115.2728260496	ec:75:0c:c2:31:93	ec:05:02:f8:c6:5c	ARP	6	

Annex 5.1 - Experiment 4: Network



Annex 5.2 - Experiment 4: Tux93 pinging all interfaces

No.	Time	Source	Destination	Protocol	Length	Info
9	4.084141191	172.16.99.1	172.16.99.1	ICMP	88	Echo (ping) request id=0x0de5, seq=9/1280, ttl=64 (reply in 10)
10	4.084141201	172.16.99.1	172.16.99.1	ICMP	88	Echo (ping) reply id=0x0de5, seq=5/1280, ttl=63 (request in 9)
11	4.084141232	172.16.99.1	172.16.99.1	ICMP	88	Echo (ping) request id=0x0de5, seq=6/1536, ttl=64 (reply in 12)
12	5.108959480	172.16.99.1	172.16.99.1	ICMP	88	Echo (ping) reply id=0x0de5, seq=6/1536, ttl=63 (request in 11)
13	5.108639124	ec:75:0c:c2:26:42	ec:75:0c:c2:17:78	ARP	60	Who has 172.16.99.1? Tell 172.16.99.25
14	5.110654434	ec:75:0c:c2:17:78	ec:75:0c:c2:26:42	ARP	42	172.16.09.1 is at ec:75:0c:c2:17:78
15	5.110654437	ec:75:0c:c2:26:42	ec:75:0c:c2:17:78	ARP	42	172.16.09.1 is at ec:75:0c:c2:17:78
16	5.284274999	ec:75:0c:c2:26:42	ec:75:0c:c2:17:78	ARP	60	172.16.99.254 is at ec:75:0c:c2:26:42
17	6.132144221	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) request id=0x0de5, seq=7/1702, ttl=64 (reply in 18)
18	6.132062722	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0de5, seq=7/1702, ttl=63 (request in 17)
19	6.132062729	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) request id=0x0de5, seq=8/2048, ttl=64 (reply in 19)
20	7.156612599	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0de5, seq=8/2048, ttl=63 (request in 19)
21	12.059954359	0.0.0.0	255.255.255.255	MDNP	159	5078 - 5078 Len=17 Port ID: bridge00
22	12.059954729	RouteBoard_dc_1c:95:d7	CDF/VT/DP/PGP/JOLD	CDP	93	Device ID: Mikrotik Port ID: bridge00
23	13.174614229	172.16.99.1	172.16.99.254	ICMP	98	Echo (ping) request id=0x0de5, seq=1/256, ttl=64 (reply in 24)
24	17.360169013	172.16.99.254	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0de5, seq=1/256, ttl=64 (request in 23)
25	18.388119158	172.16.99.1	172.16.99.254	ICMP	98	Echo (ping) request id=0x0de5, seq=2/512, ttl=64 (reply in 26)
26	19.388119158	172.16.99.254	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0de5, seq=2/512, ttl=64 (request in 25)
27	19.421482968	172.16.99.1	172.16.99.254	ICMP	98	Echo (ping) request id=0x0de5, seq=3/768, ttl=64 (reply in 26)
28	19.421482968	172.16.99.254	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0de5, seq=3/768, ttl=64 (request in 27)
29	20.436152589	172.16.99.1	172.16.99.254	ICMP	98	Echo (ping) request id=0x0de5, seq=4/1024, ttl=64 (reply in 38)
30	20.436152589	172.16.99.254	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0de5, seq=4/1024, ttl=64 (request in 39)
31	21.405152589	172.16.99.1	172.16.99.254	ICMP	98	Echo (ping) request id=0x0de5, seq=5/1280, ttl=64 (reply in 52)
32	21.405152589	172.16.99.254	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0de5, seq=5/1280, ttl=64 (request in 51)
33	22.484137489	172.16.99.1	172.16.99.254	ICMP	98	Echo (ping) request id=0x0de5, seq=6/1536, ttl=64 (reply in 34)
34	22.484137489	172.16.99.254	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0de5, seq=6/1536, ttl=64 (request in 33)
35	33.765163849	172.16.99.1	172.16.99.253	ICMP	98	Echo (ping) request id=0x0de5, seq=7/256, ttl=64 (reply in 35)
36	31.768373848	172.16.99.253	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0de5, seq=7/256, ttl=64 (request in 35)
37	32.788138892	172.16.99.1	172.16.99.253	ICMP	98	Echo (ping) request id=0x0de5, seq=8/512, ttl=64 (reply in 38)
38	32.788138892	172.16.99.253	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0de5, seq=8/512, ttl=64 (request in 38)
39	33.802147467	172.16.99.1	172.16.99.253	ICMP	98	Echo (ping) request id=0x0de5, seq=9/768, ttl=64 (reply in 40)
40	33.812375815	172.16.99.1	172.16.99.253	ICMP	98	Echo (ping) reply id=0x0de5, seq=9/768, ttl=64 (request in 39)
41	34.836141468	172.16.99.1	172.16.99.254	ICMP	98	Echo (ping) request id=0x0de5, seq=10/1024, ttl=64 (reply in 42)
42	34.836141468	172.16.99.254	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0de5, seq=10/1024, ttl=64 (request in 41)
43	35.860140319	172.16.99.1	172.16.99.253	ICMP	98	Echo (ping) request id=0x0de5, seq=11/1280, ttl=64 (reply in 41)
44	35.860140319	172.16.99.253	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0de5, seq=11/1280, ttl=64 (request in 43)
45	36.854590712	172.16.99.1	172.16.99.254	ICMP	98	Echo (ping) request id=0x0de5, seq=12/1536, ttl=64 (reply in 43)
46	36.854590712	172.16.99.254	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0de5, seq=12/1536, ttl=64 (request in 43)
47	36.854590712	172.16.99.1	172.16.99.253	ICMP	98	Echo (ping) request id=0x0de5, seq=13/256, ttl=64 (reply in 48)
48	36.884344894	172.16.99.1	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0de5, seq=13/256, ttl=64 (request in 47)
49	36.948088899	ec:75:0c:c2:17:78	ec:75:0c:c2:26:42	ARP	60	Who has 172.16.99.1? Tell 172.16.99.1
50	36.948088899	ec:75:0c:c2:26:42	ec:75:0c:c2:17:78	ARP	42	172.16.99.1 is at ec:75:0c:c2:17:78
51	37.988314766	172.16.99.1	172.16.99.253	ICMP	98	Echo (ping) request id=0x0de5, seq=7/1702, ttl=64 (reply in 52)
52	37.988311887	172.16.99.253	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0de5, seq=7/1702, ttl=64 (request in 51)
53	38.932163117	172.16.99.1	172.16.99.253	ICMP	98	Echo (ping) request id=0x0de5, seq=8/2048, ttl=64 (reply in 54)
54	38.932163117	172.16.99.253	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0de5, seq=8/2048, ttl=64 (request in 53)
55	44.848117201	172.16.99.1	172.16.99.254	ICMP	98	Echo (ping) request id=0x0de5, seq=9/256, ttl=64 (reply in 56)
56	44.848117201	172.16.99.254	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0de5, seq=9/256, ttl=64 (request in 55)
57	45.876139499	172.16.99.1	172.16.99.254	ICMP	98	Echo (ping) request id=0x0de5, seq=10/512, ttl=64 (reply in 58)
58	45.876139499	172.16.99.254	172.16.99.1	ICMP	98	Echo (ping) reply id=0x0de5, seq=10/512, ttl=64 (request in 57)
59	46.900954324	172.16.99.1	172.16.99.254	ICMP	98	Echo (ping) request id=0x0de5, seq=11/768, ttl=64 (reply in 60)
60	46.9009545679	172.16.99.1	172.16.99.254	ICMP	98	Echo (ping) reply id=0x0de5, seq=11/768, ttl=64 (request in 59)
61	47.924140432	172.16.99.1	172.16.99.254	ICMP	98	Echo (ping) request id=0x0de5, seq=12/1024, ttl=64 (reply in 62)
62	47.924476015	172.16.99.1	172.16.99.254	ICMP	98	Echo (ping) reply id=0x0de5, seq=12/1024, ttl=63 (request in 61)
63	48.924140432	172.16.99.1	172.16.99.254	ICMP	98	Echo (ping) request id=0x0de5, seq=13/2048, ttl=64 (reply in A4)

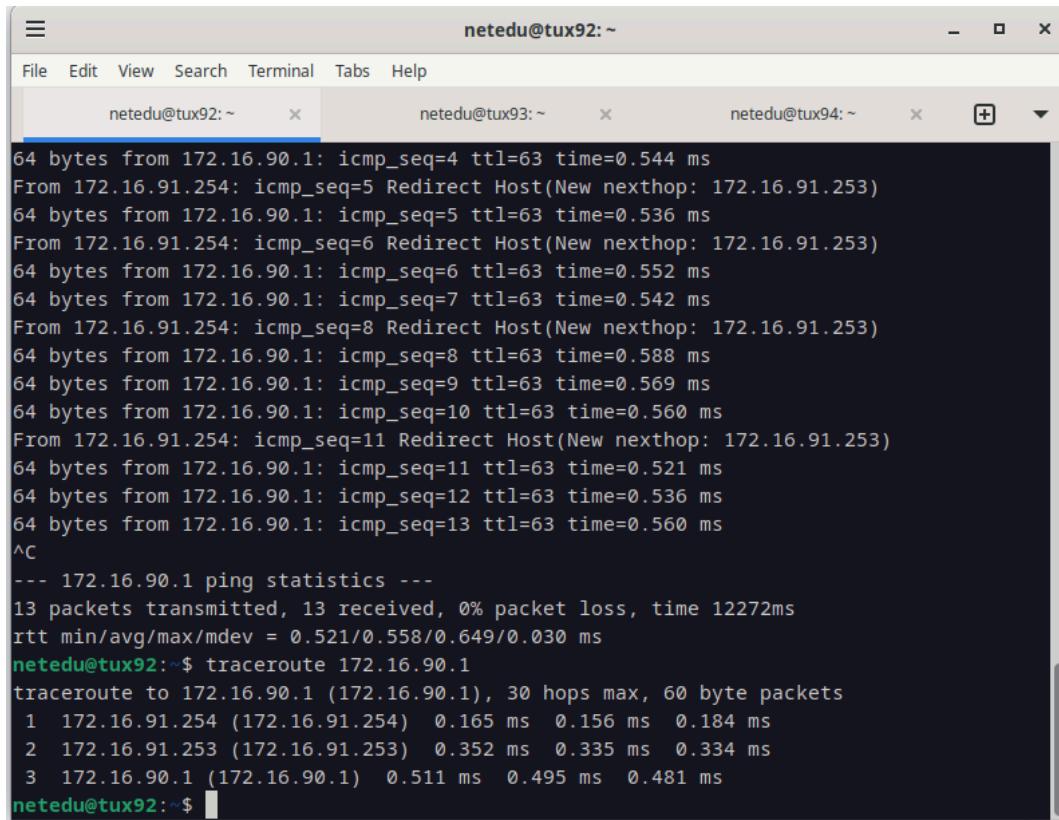
Annex 5.3 - Experiment 4: Tux92 pinging tux93 using router as gateway and no redirect

No.	Time	Source	Destination	Protocol	Length	Info
18	12:56:26.77484	Routerboardc_1c:95:d9	Spanning-tree-(for-brdg..	STP	60	RST Root = 32768/0/74:4d:28:eb:24..1d Cost = 10 Port = 0x8001
11	14:00:46:93740	Routerboardc_1c:95:d9	Spanning-tree-(for-brdg..	STP	60	RST Root = 32768/0/74:4d:28:eb:24..1d Cost = 10 Port = 0x8001
12	16:00:58:50823	Routerboardc_1c:95:d9	Spanning-tree-(for-brdg..	STP	60	RST Root = 32768/0/74:4d:28:eb:24..1d Cost = 10 Port = 0x8001
13	16:57:44:05147	a2:05:02:f6:c5:5c	Broadcast	ARP	42	Who has 172.16.91.254 Tell 172.16.91.1 256? 4d:28:eb:24..1d
15	16:57:44:05147	a2:05:02:f6:c5:5c	172.16.90.1	ICMP	98	Echo (ping) request id=0x0049, seq=1/256, ttl=64 (request in 16)
16	16:57:44:05147	a2:05:02:f6:c5:5c	172.16.90.1	ICMP	98	Echo (ping) reply id=0x0049, seq=1/256, ttl=63 (request in 15)
17	16:57:44:05147	a2:05:02:f6:c5:5c	172.16.90.1	ICMP	98	Echo (ping) request id=0x0049, seq=2/512, ttl=64 (reply in 19)
18	17:56:52:56191	172.16.91.254	172.16.91.1	ICMP	128	Redirect (Redirect for host)
19	17:56:52:56191	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) reply id=0x0049, seq=2/512, ttl=63 (request in 17)
20	17:56:52:56191	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) request id=0x0049, seq=3/768, ttl=64 (reply in 23)
21	18:11:06:63555	172.16.91.1	172.16.91.1	ICMP	128	Redirect (Redirect for host)
22	18:11:06:63555	172.16.91.254	172.16.91.1	ICMP	128	Redirect (Redirect for host)
23	18:11:06:63555	172.16.90.1	172.16.90.1	ICMP	98	Echo (ping) reply id=0x0049, seq=3/768, ttl=63 (request in 21)
24	18:11:06:63555	172.16.90.1	172.16.90.1	ICMP	98	Echo (ping) request id=0x0049, seq=4/1024, ttl=64 (reply in 26)
25	19:11:06:63555	172.16.91.254	172.16.91.1	ICMP	128	Redirect (Redirect for host)
26	19:11:06:63555	172.16.90.1	172.16.91.1	ICMP	98	Echo (ping) reply id=0x0049, seq=4/1024, ttl=63 (request in 24)
27	20:01:16:77212	Routerboardc_1c:95:d9	Spanning-tree-(for-brdg..	STP	60	RST Root = 32768/0/74:4d:28:eb:24..1d Cost = 10 Port = 0x8001
28	20:01:16:77212	Routerboardc_1c:95:d9	172.16.90.1	ICMP	98	Echo (ping) request id=0x0049, seq=5/1280, ttl=64 (reply in 30)
29	20:01:16:77212	Routerboardc_1c:95:d9	172.16.90.1	ICMP	98	Echo (ping) reply id=0x0049, seq=5/1280, ttl=63 (request in 29)
30	20:01:16:77212	Routerboardc_1c:95:d9	172.16.90.1	ICMP	98	Echo (ping) request id=0x0049, seq=6/1536, ttl=64 (reply in 33)
31	21:01:22:35516	172.16.91.254	172.16.91.1	ICMP	128	Redirect (Redirect for host)
32	21:01:22:35516	172.16.91.254	172.16.91.1	ICMP	128	Redirect (Redirect for host)
33	21:01:22:35516	172.16.90.1	172.16.90.1	ICMP	98	Echo (ping) reply id=0x0049, seq=6/1536, ttl=63 (request in 31)
34	21:01:22:35516	172.16.90.1	172.16.90.1	ICMP	98	Echo (ping) request id=0x0049, seq=7/1280, ttl=64 (reply in 34)
35	21:01:22:35516	172.16.90.1	172.16.90.1	ICMP	98	Echo (ping) reply id=0x0049, seq=7/1280, ttl=63 (request in 39)
36	21:01:22:35516	172.16.90.1	172.16.90.1	ICMP	98	Echo (ping) request id=0x0049, seq=8/1536, ttl=64 (reply in 43)
37	21:01:22:35516	172.16.90.1	172.16.90.1	ICMP	98	Echo (ping) reply id=0x0049, seq=8/1536, ttl=63 (request in 47)
38	22:05:08:81110	a2:05:02:f6:c5:5c	Routerboardc_eb:24:id	ARP	60	Who has 172.16.91.254 Tell 172.16.91.1 256? 4d:05:02:f6:c5:5c
39	22:05:08:81110	a2:05:02:f6:c5:5c	172.16.90.1	ICMP	98	Echo (ping) request id=0x0049, seq=9/1280, ttl=64 (reply in 48)
40	22:05:08:81110	a2:05:02:f6:c5:5c	172.16.90.1	ICMP	98	Echo (ping) reply id=0x0049, seq=9/1280, ttl=63 (request in 49)
41	23:09:06:62734	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) request id=0x0049, seq=10/1920, ttl=64 (reply in 43)
42	23:09:06:62734	172.16.91.254	172.16.91.1	ICMP	128	Redirect (Redirect for host)
43	23:09:06:62734	172.16.90.1	172.16.91.1	ICMP	98	Echo (ping) reply id=0x0049, seq=8/2048, ttl=63 (request in 41)
44	23:09:06:62734	172.16.90.1	172.16.90.1	ICMP	98	Echo (ping) request id=0x0049, seq=9/2048, ttl=64 (reply in 45)
45	24:54:09:87872	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) request id=0x0049, seq=9/2284, ttl=64 (reply in 46)
46	24:54:09:87872	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) reply id=0x0049, seq=9/2284, ttl=63 (request in 54)
47	24:54:09:87872	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) request id=0x0049, seq=10/2560, ttl=64 (reply in 48)
48	25:07:58:92491	172.16.90.1	172.16.90.1	ICMP	98	Echo (ping) reply id=0x0049, seq=10/2560, ttl=63 (request in 47)
49	26:01:17:44139	Routerboardc_1c:95:d9	Spanning-tree-(for-brdg..	STP	60	RST Root = 32768/0/74:4d:28:eb:24..1d Cost = 10 Port = 0x8001
50	26:01:17:44139	Routerboardc_1c:95:d9	172.16.91.1	ICMP	98	Echo (ping) request id=0x0049, seq=11/2816, ttl=64 (reply in 52)
51	26:01:17:44139	Routerboardc_1c:95:d9	172.16.91.1	ICMP	128	Redirect (Redirect for host)
52	26:01:17:44139	172.16.90.1	172.16.91.1	ICMP	98	Echo (ping) reply id=0x0049, seq=11/2816, ttl=63 (request in 50)
53	27:02:06:65268	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) request id=0x0049, seq=12/3072, ttl=64 (reply in 54)
54	27:02:06:65268	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) reply id=0x0049, seq=12/3072, ttl=63 (request in 53)
55	27:02:06:65268	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) request id=0x0049, seq=13/3232, ttl=64 (reply in 57)
56	28:05:08:76066	172.16.91.1	172.16.90.1	ICMP	98	Echo (ping) reply id=0x0049, seq=13/3232, ttl=63 (request in 56)
57	28:05:08:76066	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) request id=0x0049, seq=13/3232, ttl=64 (reply in 56)
58	30:02:17:21243	Routerboardc_1c:95:d9	Spanning-tree-(for-brdg..	STP	60	RST Root = 32768/0/74:4d:28:eb:24..1d Cost = 10 Port = 0x8001
59	32:02:38:08408	Routerboardc_1c:95:d9	Spanning-tree-(for-brdg..	STP	60	RST Root = 32768/0/74:4d:28:eb:24..1d Cost = 10 Port = 0x8001
60	34:01:54:05272	Routerboardc_1c:95:d9	Spanning-tree-(for-brdg..	STP	60	RST Root = 32768/0/74:4d:28:eb:24..1d Cost = 10 Port = 0x8001

Annex 5.4 - Experiment 4: Tux92 pinging tux93 using tux94 as gateway and redirects

No.	Time	Source	Destination	Protocol	Length	Info
15	22:00:59:79105	RouterBoardc_1c:95:d9	Spanning-tree-(for-brdg..	STP	60	RST Root = 32768/0/74:4d:28:eb:24..1d Cost = 10 Port = 0x8001
16	22:00:59:80885	RouterBoardc_1c:95:d9	Spanning-tree-(for-brdg..	STP	60	RST Root = 32768/0/74:4d:28:eb:24..1d Cost = 10 Port = 0x8001
17	26:00:58:04263	RouterBoardc_1c:95:d9	Spanning-tree-(for-brdg..	STP	60	RST Root = 32768/0/74:4d:28:eb:24..1d Cost = 10 Port = 0x8001
18	28:01:01:71646	RouterBoardc_1c:95:d9	Spanning-tree-(for-brdg..	STP	60	RST Root = 32768/0/74:4d:28:eb:24..1d Cost = 10 Port = 0x8001
19	28:01:01:71646	RouterBoardc_1c:95:d9	Spanning-tree-(for-brdg..	STP	60	RST Root = 32768/0/74:4d:28:eb:24..1d Cost = 10 Port = 0x8001
20	30:09:44:07385	172.16.91.1	172.16.90.1	ICMP	98	Echo (ping) request id=0x0049, seq=1/256, ttl=64 (reply in 21)
21	30:09:44:07385	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) reply id=0x0049, seq=1/256, ttl=63 (request in 20)
22	31:09:44:07385	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) request id=0x0049, seq=2/512, ttl=64 (reply in 23)
23	31:09:44:07385	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) reply id=0x0049, seq=2/512, ttl=63 (request in 22)
24	32:01:44:36598	Routerboardc_1c:95:d9	Spanning-tree-(for-brdg..	STP	60	RST Root = 32768/0/74:4d:28:eb:24..1d Cost = 10 Port = 0x8001
25	32:01:44:36598	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) request id=0x0049, seq=3/768, ttl=64 (reply in 29)
26	32:01:44:36598	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) reply id=0x0049, seq=3/768, ttl=63 (request in 28)
27	33:09:44:63711	172.16.91.1	172.16.90.1	ICMP	98	Echo (ping) request id=0x0049, seq=4/1024, ttl=64 (reply in 28)
28	33:09:44:63711	172.16.90.1	172.16.91.1	ICMP	98	Echo (ping) reply id=0x0049, seq=4/1024, ttl=63 (request in 27)
29	34:01:05:75221	Routerboardc_1c:95:d9	Spanning-tree-(for-brdg..	STP	60	RST Root = 32768/0/74:4d:28:eb:24..1d Cost = 10 Port = 0x8001
30	34:01:05:75221	172.16.91.1	172.16.90.1	ICMP	98	Echo (ping) request id=0x0049, seq=5/1280, ttl=64 (reply in 31)
31	34:01:05:75221	172.16.90.1	172.16.91.1	ICMP	98	Echo (ping) reply id=0x0049, seq=5/1280, ttl=63 (request in 30)
32	35:09:44:87823	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) request id=0x0049, seq=6/1536, ttl=64 (reply in 35)
33	35:09:44:87823	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) reply id=0x0049, seq=6/1536, ttl=63 (request in 34)
34	35:09:44:87823	172.16.91.1	172.16.90.1	ICMP	98	Echo (ping) request id=0x0049, seq=7/1280, ttl=64 (reply in 36)
35	35:09:44:87823	172.16.90.1	172.16.91.1	ICMP	98	Echo (ping) reply id=0x0049, seq=7/1280, ttl=63 (request in 35)
36	35:09:44:87823	172.16.90.1	172.16.90.1	ICMP	98	Echo (ping) request id=0x0049, seq=8/1536, ttl=64 (reply in 37)
37	36:09:54:05494	172.16.91.1	172.16.90.1	ICMP	98	Echo (ping) reply id=0x0049, seq=8/1536, ttl=63 (request in 36)
38	36:09:54:05494	172.16.90.1	172.16.91.1	ICMP	98	Echo (ping) request id=0x0049, seq=9/1280, ttl=64 (reply in 38)
39	36:09:54:05494	172.16.90.1	172.16.90.1	ICMP	98	Echo (ping) reply id=0x0049, seq=9/1280, ttl=63 (request in 39)
40	36:09:54:05494	172.16.90.1	172.16.90.1	ICMP	98	Echo (ping) request id=0x0049, seq=10/1920, ttl=64 (reply in 40)
41	36:09:54:05494	172.16.90.1	172.16.90.1	ICMP	98	Echo (ping) reply id=0x0049, seq=10/1920, ttl=63 (request in 41)
42	36:21:85:64324	Routerboardc_1c:95:d9	Spanning-tree-(for-brdg..	STP	60	RST Root = 32768/0/74:4d:28:eb:24..1d Cost = 10 Port = 0x8001
43	36:21:85:64324	172.16.91.1	172.16.90.1	ICMP	98	Echo (ping) request id=0x0049, seq=11/2816, ttl=64 (reply in 45)
44	36:21:85:64324	172.16.90.1	172.16.91.1	ICMP	98	Echo (ping) reply id=0x0049, seq=11/2816, ttl=63 (request in 44)
45	36:78:14:13100	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) request id=0x0049, seq=12/3072, ttl=64 (reply in 47)
46	37:08:49:78818	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) reply id=0x0049, seq=12/3072, ttl=63 (request in 45)
47	37:08:49:78818	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) request id=0x0049, seq=13/3232, ttl=64 (reply in 49)
48	37:08:49:78818	172.16.91.1	172.16.91.1	ICMP	98	Echo (ping) reply id=0x0049, seq=13/3232, ttl=63 (request in 48)
49	40:02:2795957	Routerboardc_1c:95:d9	Spanning-tree-(for-brdg..	STP	60	RST Root = 32768/0/74:4d:28:eb:24..1d Cost = 10 Port = 0x8001
50	40:02:2795957	172.16.91.1	172.16.90.1	ICMP	98	Echo (ping) request id=0x0049, seq=14/2048, ttl=64 (reply in 50)
51	40:02:2795957	172.16.90.1	172.16.91.1	ICMP	98	Echo (ping) reply id=0x0049, seq=14/2048, ttl=63 (request in 49)
52	45:14:2416718	172.16.91.1	172.16.91.1	ICMP	299	259.255.259
53	45:14:2416718	172.16.91.1	172.16.91.1	ICMP	244	0.0.22
54	45:14:2416718	172.16.91.1	172.16.91.1	ICMP	244	0.0.22
55	45:14:2416718	172.16.91.1	172.16.91.1	ICMP	244	0.0.22
56	45:14:2416718	172.16.91.1	172.16.91.1	ICMP	244	0.0.22
57	45:14:2416718	172.16.91.1	172.16.91.1	ICMP	244	0.0.22
58	46:76:49:26949	172				

Annex 5.5 - Experiment 4: Traceroute from tux92 to tux93 using router as gateway



The screenshot shows a terminal window titled "netedu@tux92: ~". It has three tabs open: "netedu@tux92: ~", "netedu@tux93: ~", and "netedu@tux94: ~". The "netedu@tux92: ~" tab is active and displays the following command-line session:

```
64 bytes from 172.16.90.1: icmp_seq=4 ttl=63 time=0.544 ms
From 172.16.91.254: icmp_seq=5 Redirect Host(New nexthop: 172.16.91.253)
64 bytes from 172.16.90.1: icmp_seq=5 ttl=63 time=0.536 ms
From 172.16.91.254: icmp_seq=6 Redirect Host(New nexthop: 172.16.91.253)
64 bytes from 172.16.90.1: icmp_seq=6 ttl=63 time=0.552 ms
64 bytes from 172.16.90.1: icmp_seq=7 ttl=63 time=0.542 ms
From 172.16.91.254: icmp_seq=8 Redirect Host(New nexthop: 172.16.91.253)
64 bytes from 172.16.90.1: icmp_seq=8 ttl=63 time=0.588 ms
64 bytes from 172.16.90.1: icmp_seq=9 ttl=63 time=0.569 ms
64 bytes from 172.16.90.1: icmp_seq=10 ttl=63 time=0.560 ms
From 172.16.91.254: icmp_seq=11 Redirect Host(New nexthop: 172.16.91.253)
64 bytes from 172.16.90.1: icmp_seq=11 ttl=63 time=0.521 ms
64 bytes from 172.16.90.1: icmp_seq=12 ttl=63 time=0.536 ms
64 bytes from 172.16.90.1: icmp_seq=13 ttl=63 time=0.560 ms
^C
--- 172.16.90.1 ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 12272ms
rtt min/avg/max/mdev = 0.521/0.558/0.649/0.030 ms
netedu@tux92:~$ traceroute 172.16.90.1
traceroute to 172.16.90.1 (172.16.90.1), 30 hops max, 60 byte packets
 1  172.16.91.254 (172.16.91.254)  0.165 ms  0.156 ms  0.184 ms
 2  172.16.91.253 (172.16.91.253)  0.352 ms  0.335 ms  0.334 ms
 3  172.16.90.1 (172.16.90.1)  0.511 ms  0.495 ms  0.481 ms
netedu@tux92:~$
```

Annex 5.6 - Experiment 4: Traceroute from tux92 to tux93 using tux94 as gateway

```

netedu@tux92:~$ traceroute 172.16.90.1
traceroute to 172.16.90.1 (172.16.90.1), 30 hops max, 60 byte packets
 1  172.16.91.254 (172.16.91.254)  0.165 ms  0.156 ms  0.184 ms
  2  172.16.91.253 (172.16.91.253)  0.352 ms  0.335 ms  0.334 ms
  3  172.16.90.1 (172.16.90.1)  0.511 ms  0.495 ms  0.481 ms
netedu@tux92:~$ route del -net 172.16.90.0/24 gw 172.16.91.254
netedu@tux92:~$ route add -net 172.16.90.0/24 gw 172.16.91.253
netedu@tux92:~$ traceroute 172.16.90.1
traceroute to 172.16.90.1 (172.16.90.1), 30 hops max, 60 byte packets
  1  172.16.91.253 (172.16.91.253)  0.201 ms  0.182 ms  0.168 ms
  2  172.16.90.1 (172.16.90.1)  0.311 ms  0.335 ms  0.315 ms
netedu@tux92:~$ 

```

Annex 5.7 - Experiment 4: Tux93 pinging Netlab FTP Server with NAT on

No.	Time	Source	Destination	Protocol	Length Info
1	0.0000000000	172.16.90.1	172.16.1.10	ICMP	98 Echo (ping) request id=0x1001, seq=1/256, ttl=64 (reply in 2)
2	0.000003239	172.16.1.10	172.16.90.1	ICMP	98 Echo (ping) reply id=0x1001, seq=1/256, ttl=64 (request in 1)
3	0.1000000000	172.16.90.1	172.16.90.1	ICMP	98 Echo (ping) request id=0x1001, seq=2/256, ttl=64 (reply in 4)
4	1.615651698	172.16.1.10	172.16.90.1	ICMP	98 Echo (ping) reply id=0x1001, seq=2/256, ttl=64 (request in 3)
5	2.039122531	172.16.90.1	172.16.1.10	ICMP	98 Echo (ping) request id=0x1001, seq=3/256, ttl=64 (reply in 6)
6	2.639666604	172.16.1.10	172.16.90.1	ICMP	98 Echo (ping) reply id=0x1001, seq=3/256, ttl=64 (request in 5)
7	3.665127886	172.16.90.1	172.16.1.10	ICMP	98 Echo (ping) request id=0x1001, seq=4/256, ttl=64 (reply in 8)
8	4.0000000000	172.16.1.10	172.16.90.1	ICMP	98 Echo (ping) reply id=0x1001, seq=4/256, ttl=64 (request in 7)
9	4.087889196	172.16.90.1	172.16.1.10	ICMP	98 Echo (ping) request id=0x1001, seq=5/256, ttl=64 (reply in 10)
10	4.087592831	172.16.1.10	172.16.90.1	ICMP	98 Echo (ping) reply id=0x1001, seq=5/256, ttl=64 (request in 9)
11	5.111128370	172.16.90.1	172.16.1.10	ICMP	98 Echo (ping) request id=0x1001, seq=6/256, ttl=64 (reply in 12)
12	5.111129245	172.16.1.10	172.16.90.1	ICMP	98 Echo (ping) reply id=0x1001, seq=6/256, ttl=64 (request in 11)
13	5.111129245	172.16.90.1	172.16.1.10	ICMP	98 Echo (ping) request id=0x1001, seq=7/256, ttl=64 (reply in 13)
14	5.1111318199	ec:75:0c:c2:17:7a	ec:75:0c:c2:26:42	ARP	42 172.16.90.1 is at ec:75:0c:c2:26:42
15	5.207080322	ec:75:0c:c2:17:7a	ec:75:0c:c2:26:42	ARP	42 Who has 172.16.90.1? Tell 172.16.90.1
16	5.207269285	ec:75:0c:c2:26:42	ec:75:0c:c2:17:7a	ARP	60 172.16.90.254 is at ec:75:0c:c2:26:42
17	6.135126524	172.16.90.1	172.16.1.10	ICMP	98 Echo (ping) request id=0x1001, seq=7/256, ttl=64 (reply in 18)
18	6.135126524	172.16.1.10	172.16.90.1	ICMP	98 Echo (ping) reply id=0x1001, seq=7/256, ttl=64 (request in 17)
19	7.159126163	172.16.90.1	172.16.1.10	ICMP	98 Echo (ping) request id=0x1001, seq=8/256, ttl=64 (reply in 20)
20	7.159656981	172.16.1.10	172.16.90.1	ICMP	98 Echo (ping) reply id=0x1001, seq=8/256, ttl=64 (request in 19)
21	8.183131275	172.16.90.1	172.16.1.10	ICMP	98 Echo (ping) request id=0x1001, seq=9/256, ttl=64 (reply in 22)
22	8.1836669292	172.16.1.10	172.16.90.1	ICMP	98 Echo (ping) reply id=0x1001, seq=9/256, ttl=64 (request in 21)
23	8.2071580000	172.16.90.1	172.16.1.10	ICMP	98 Echo (ping) request id=0x1001, seq=10/256, ttl=64 (reply in 24)
24	9.2070804813	172.16.1.10	172.16.90.1	ICMP	98 Echo (ping) reply id=0x1001, seq=10/256, ttl=64 (request in 23)

Annex 5.8 - Experiment 4: Tux93 pinging Netlab FTP Server with NAT off

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	172.16.99.1	172.16.1.19	ICMP	68	Echo (ping) request id=0x100d, seq=1/256, ttl=64 (no response found!)
2	1.030993279	172.16.99.1	172.16.1.19	ICMP	68	Echo (ping) request id=0x100d, seq=2/512, ttl=64 (no response found!)
3	2.054996698	172.16.99.1	172.16.1.19	ICMP	68	Echo (ping) request id=0x100d, seq=3/768, ttl=64 (no response found!)
4	3.082397971	172.16.99.1	172.16.1.19	ICMP	68	Echo (ping) request id=0x100d, seq=4/1024, ttl=64 (no response found!)
5	4.010880903	172.16.99.1	172.16.1.19	ICMP	68	Echo (ping) request id=0x100d, seq=5/1280, ttl=64 (no response found!)
6	5.0494837623	ec:75:0c:c2:17:7a	ec:75:0c:c2:26:42	ARP	42	Who has 172.16.99.254? Tell 172.16.99.1
7	5.099593196	ec:75:0c:c2:17:7a	ec:75:0c:c2:17:7a	ARP	60	172.16.99.254 is at ec:75:0c:c2:26:42
8	5.126885253	172.16.99.1	172.16.1.19	ICMP	68	Echo (ping) request id=0x100d, seq=6/1536, ttl=64 (no response found!)
9	6.150891978	172.16.99.1	172.16.1.19	ICMP	68	Echo (ping) request id=0x100d, seq=7/1792, ttl=64 (no response found!)
10	7.179891978	172.16.99.1	172.16.1.19	ICMP	68	Echo (ping) request id=0x100d, seq=8/2048, ttl=64 (no response found!)
11	8.198891915	172.16.99.1	172.16.1.19	ICMP	68	Echo (ping) request id=0x100d, seq=9/2304, ttl=64 (no response found!)
12	9.222888685	172.16.99.1	172.16.1.19	ICMP	68	Echo (ping) request id=0x100d, seq=10/2560, ttl=64 (no response found!)
13	10.137270596	0.0.0.0	255.255.255.255	MNDP	159	5678 - 5678 Len=117
14	10.137270596	Routerboardc_1c:95:d7	CDP/VT/OTP/PagP/UDLD	CDP	93	Device ID: MikroTik Port ID: bridge80
15	10.137270596	172.16.99.1	172.16.1.19	ICMP	68	Echo (ping) request id=0x100d, seq=11/2816, ttl=64 (no response found!)
16	11.278892746	172.16.99.1	172.16.1.19	ICMP	68	Echo (ping) request id=0x100d, seq=12/3072, ttl=64 (no response found!)
17	12.29497669	172.16.99.1	172.16.1.19	ICMP	68	Echo (ping) request id=0x100d, seq=13/3328, ttl=64 (no response found!)
18	13.318889886	172.16.99.1	172.16.1.19	ICMP	68	Echo (ping) request id=0x100d, seq=14/3584, ttl=64 (no response found!)
19	14.342891117	172.16.99.1	172.16.1.19	ICMP	68	Echo (ping) request id=0x100d, seq=15/3840, ttl=64 (no response found!)

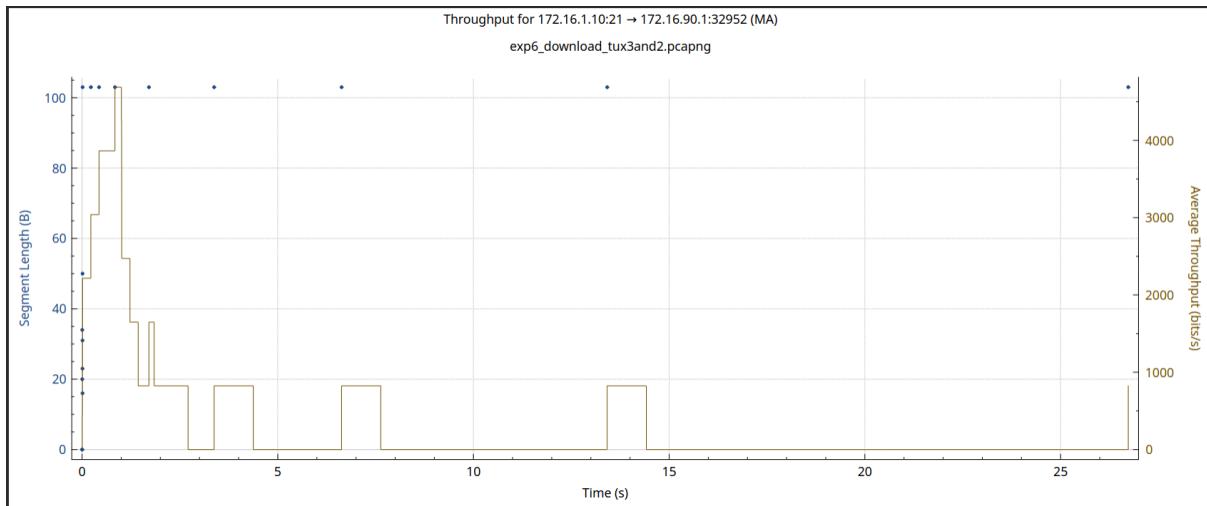
Annex 6.1 - Experiment 5: Tux93 pinging google.com

No.	Time	Source	Destination	Protocol	Length	Info	
97	3.585988776	Dell_0c:02:b4	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.229	
98	4.191372706	Dell_0b:cf:89	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.210	
99	4.544301846	Dell_0b:cf:89	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.210	
100	4.687984018	Dell_0c:02:b4	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.229	
101	4.700000000	Dell_0c:02:b4	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.229	
102	5.590446231	Dell_0b:cf:89	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.210	
103	5.768411546	Dell_0b:cf:89	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.210	
104	5.951098977	Dell_0c:02:b4	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.229	
105	6.000000000	Dell_0c:02:b4	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.229	
106	6.623498568	Dell_0b:cf:89	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.210	
107	6.815331497	Dell_0b:cf:89	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.210	
108	6.952292038	10.227.20.3	DNS	70	Standard query 0x003a A google.com		
109	7.002292038	10.227.20.3	DNS	70	Standard query 0x003a A google.com		
110	8.053182447	10.227.20.3	DNS	88	Standard query response 0x03a A google.com 142.250.184.174		
111	9.053102790	10.227.20.3	DNS	88	Standard query response 0x03e AAAA google.com AAAA 2a00:1450:4003:80c::200e		
112	0.053421705	142.250.184.174	ICMP	68	Echo (ping) request id=0x1003, seq=1/256, ttl=64 (reply in 113)		
113	0.053421705	142.250.184.174	ICMP	68	Echo (ping) reply id=0x1003, seq=1/256, ttl=64 (request in 112)		
114	0.973388130	10.227.20.3	DNS	88	Standard query 0x003e PTR 174.184.250.142.in-addr.arpa		
115	6.072114068	10.227.20.3	DNS	127	Standard query response 0x003c PTR mad07s23-in-f14.e1e100.net		
116	6.975977625	Dell_0c:02:b4	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.229	
117	7.000000000	Dell_0c:02:b4	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.229	
118	7.839412472	Dell_0b:cf:89	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.210	
119	7.955119705	10.227.20.3	ICMP	68	Echo (ping) request id=0x1003, seq=2/512, ttl=64 (reply in 120)		
120	7.979888342	142.250.184.174	ICMP	68	Echo (ping) reply id=0x1003, seq=2/512, ttl=64 (request in 119)		
121	8.000000000	10.227.20.3	DNS	88	Standard query response 0x003d PTR 174.184.250.142.in-addr.arpa		
122	7.972146925	10.227.20.3	DNS	127	Standard query response 0x0017 PTR 174.184.250.142.in-addr.arpa PTR mad07s23-in-f14.e1e100.net		
123	8.169751809	Dell_0b:cf:89	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.210	
124	8.192908074	Dell_0c:02:b4	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.229	
125	8.192908074	Dell_0c:02:b4	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.229	
126	8.125874709	ProxmoXServe_e7:5e:5b	Dell_70:98:0b	ProxmoXServe_e7:5e:5b	ARP	60	Who has 10.227.20.249? Tell 10.227.20.210
127	8.512896681	ProxmoXServe_e7:5e:5b	Dell_70:98:0b	ProxmoXServe_e7:5e:5b	ARP	42	10.227.20.93 is at 5c:f0:dd:70:98:0b
128	8.956296682	10.227.20.93	ICMP	68	Echo (ping) request id=0x1003, seq=3/768, ttl=64 (reply in 129)		
129	9.000000000	142.250.184.174	ICMP	68	Echo (ping) reply id=0x1003, seq=3/768, ttl=64 (request in 128)		
130	8.97383454	10.227.20.93	DNS	88	Standard query response 0xd5a6 PTR 174.184.250.142.in-addr.arpa		
131	8.974183793	10.227.20.93	DNS	127	Standard query response 0xd5d0 PTR 174.184.250.142.in-addr.arpa PTR mad07s23-in-f14.e1e100.net		
132	9.183396175	Dell_0b:cf:89	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.210	
133	9.183396175	Dell_0b:cf:89	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.229	
134	9.379416996	Dell_0b:cf:89	Broadcast	ARP	60	Who has 10.227.20.249? Tell 10.227.20.210	
135	9.45477379	210.58.209.74	UDU	71	000930 - 443 Len=29		
136	10.227.20.93	151.181.193.01	TLSV1.2	105	Application Data		
137	10.227.20.93	34.36.54.88	TLSV1.2	105	Application Data		
138	10.227.20.93	34.36.54.88	TLSV1.2	105	Application Data		
139	10.227.20.93	151.181.129.01	TLSV1.2	105	Application Data		
140	10.227.20.93	10.227.20.93	TCP	66	443 - 36762 [ACK] Seq=1 Ack=40 Win=280 Len=0 TSeq=4123043331 TSecr=3306830050		
141	10.227.20.93	151.181.129.91	TCP	66	443 - 47346 [ACK] Seq=1 Ack=40 Win=280 Len=0 TSeq=2876939870 TSecr=887591929		
142	10.227.20.93	10.227.20.93	TCP	66	443 - 47346 [ACK] Seq=1 Ack=40 Win=280 Len=0 TSeq=2876939870 TSecr=887591929		
143	10.227.20.93	151.181.129.91	TCP	66	443 - 47346 [ACK] Seq=1 Ack=40 Win=280 Len=0 TSeq=2876939870 TSecr=887591929		
144	9.238858384	34.36.54.88	TLSV1.2	105	Application Data		
145	10.227.20.93	10.227.20.93	TLSV1.2	105	Application Data		
146	9.451342465	10.227.20.93	TCP	78	47346 - 443 [PSH, ACK] Seq=448 Win=280 Len=0 TSeq=2876939894 SLE=1 SRE=48		
147	9.45405490	151.181.193.01	TCP	105	[TCP Retransmission] 443 - 36762 [PSH, ACK] Seq=448 Win=280 Len=0 TSeq=4123043347 TSecr=3306830050		
148	9.458417816	10.227.20.93	TCP	78	36762 - 443 [ACK] Seq=448 Win=501 Len=0 TSeq=1386830970 TSecr=4123043347 SLE=1 SRE=48		
149	9.458417816	34.36.137.293	TCP	66	443 - 47346 [ACK] Seq=1 Ack=40 Win=1044 Len=0 TSeq=624771048 TSecr=3191736878		
150	9.45284676	34.36.137.293	TCP	66	443 - 47346 [ACK] Seq=1 Ack=40 Win=501 Len=0 TSeq=1910736714 TSecr=624771048		
151	9.451306833	10.227.20.93	TCP	66	46798 - 443 [ACK] Seq=448 Win=501 Len=0 TSeq=1910736714 TSecr=624771048		

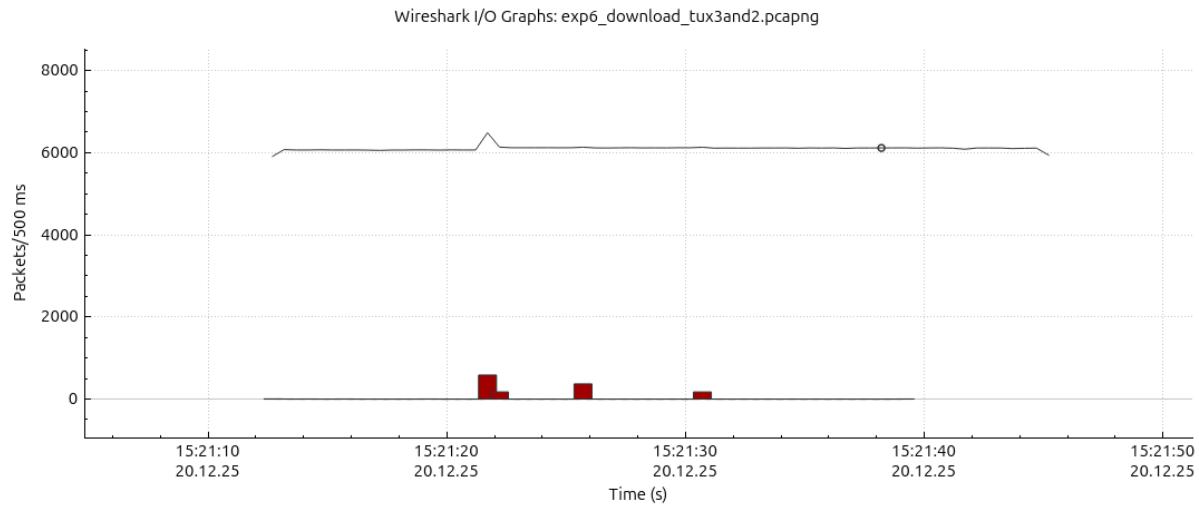
Annex 7.1 - Experiment 6: Successful download with all packets

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	172.16.99.1	172.16.1.10	TCP	74	58702 - 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM Tsvl=206813392 TSecr=0 WS=128
2	0.0008888275	172.16.1.10	172.16.99.1	TCP	74	21 - 58702 [SYN, ACK] Seq=1 Win=65160 Len=0 MSS=1460 SACK_PERM Tsvl=196568788 TSecr=206813392 WS=1
3	0.0008851934	172.16.99.1	172.16.1.10	TCP	66	58702 - 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 Tsvl=206813393 TSecr=196568788
4	0.0008851934	172.16.99.1	172.16.1.10	FTP	80	Response: 230 User anonymous. Tsvl=196568788 TSecr=196568788
5	0.005281389	172.16.99.1	172.16.1.10	FTP	66	58702 - 21 [ACK] Seq=1 Ack=21 Win=64256 Len=0 Tsvl=206813398 TSecr=196568793
6	0.00532539	172.16.99.1	172.16.1.10	FTP	82	Request: USER anonymous.
7	0.005738232	172.16.1.10	172.16.99.1	TCP	66	21 - 58702 [ACK] Seq=21 Ack=17 Win=65280 Len=0 Tsvl=196568793 TSecr=206813398
8	0.005810158	172.16.1.10	172.16.99.1	FTP	80	Request: PASS password.
9	0.005810158	172.16.99.1	172.16.1.10	FTP	80	Response: 230 Logon successful.
10	0.012623299	172.16.99.1	172.16.1.10	FTP	74	Request: TYPE I
11	0.01389666	172.16.99.1	172.16.1.10	FTP	80	Response: 200 Switching to Binary mode.
12	0.011898997	172.16.1.10	172.16.99.1	FTP	97	Request: SIZE /pipe.txt
13	0.011961227	172.16.99.1	172.16.1.10	FTP	75	Response: 213 191
14	0.012543827	172.16.1.10	172.16.99.1	FTP	72	Request: PASV
15	0.012543827	172.16.99.1	172.16.1.10	FTP	115	Response: 227 Entering Passive Mode (172,16,1,10,165,69).
16	0.013431235	172.16.1.10	172.16.99.1	FTP	74	38914 - 42389 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM Tsvl=206813406 TSecr=0 WS=128
17	0.013512931	172.16.99.1	172.16.1.10	TCP	74	42399 - 38914 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM Tsvl=196568802 TSecr=206813406 W=128
18	0.013944669	172.16.1.10	172.16.99.1	TCP	66	38914 - 42389 [ACK] Seq=1 Ack=1 Win=64256 Len=0 Tsvl=206813406 TSecr=196568802
19	0.013977612	172.16.99.1	172.16.1.10	FTP	90	Request: RETR /pipe.txt
20	0.014000000	172.16.1.10	172.16.99.1	FTP	134	Response: 158 Opening BINARY mode data connection for /pipe.txt (191 bytes).
21	0.014923987	172.16.1.10	172.16.99.1	FTP	257	FTP Data: 191 bytes (PASV) (RETR /pipe.txt)
22	0.014930225	172.16.1.10	172.16.99.1	FTP-DATA	66	42399 - 38914 [FIN, ACK] Seq=192 Ack=1 Win=65280 Len=0 Tsvl=196568803 TSecr=206813406
23	0.014930207	172.16.1.10	172.16.99.1	TCP	66	38914 - 42389 [FIN, ACK] Seq=192 Ack=1 Win=65280 Len=0 Tsvl=196568803 TSecr=206813406
24	0.014989000	172.16.1.10	172.16.99.1	TCP	66	38914 - 42389 [ACK] Seq=1 Ack=192 Win=64128 Len=0 Tsvl=206813407 TSecr=196568803
25	0.058898446	172.16.99.1	172.16.1.10	FTP	60	Response: 226 Transfer complete.
26	0.058897251	172.16.99.1	172.16.1.10	FTP	66	58702 - 21 [ACK] Seq=78 Ack=235 Win=64256 Len=0 Tsvl=206813451 TSecr=196568803
27	0.058758991	172.16.1.10	172.16.99.1	FTP	99	Response: 226 Transfer complete.
28	0.058799924	172.16.99.1	172.16.1.10	TCP	66	58702 - 21 [ACK] Seq=78 Ack=259 Win=64256 Len=0 Tsvl=206813451 TSecr=196568846
29	0.0588882913	172.16.99.1	172.16.1.10	FTP	72	Request: QUIT
30	0.0588882913	172.16.1.10	172.16.99.1	FTP	80	Response: 221 Goodbye.
31	0.059434721	172.16.1.10	172.16.99.1	TCP	66	21 - 58702 [FIN, ACK] Seq=273 Ack=84 Win=65280 Len=0 Tsvl=196568847 TSecr=206813451
32	0.059430822	172.16.99.1	172.16.1.10	TCP	66	58702 - 21 [FIN, ACK] Seq=84 Ack=274 Win=64256 Len=0 Tsvl=206813452 TSecr=196568847
33	0.059458888	172.16.99.1	172.16.1.10	TCP	66	38914 - 42389 [FIN, ACK] Seq=1 Ack=193 Win=64128 Len=0 Tsvl=206813452 TSecr=196568803
34	0.0598981711	172.16.1.10	172.16.99.1	TCP	66	21 - 58702 [ACK] Seq=274 Ack=85 Win=65280 Len=0 Tsvl=196568848 TSecr=206813452
35	0.0598981711	172.16.99.1	172.16.1.10	TCP	66	21 - 58702 [ACK] Seq=274 Ack=85 Win=65280 Len=0 Tsvl=196568848 TSecr=206813452
36	9.164624114	ec:75:0c:c2:26:42	ec:75:0c:c2:17:7a	ARP	60	Who has 172.16.99.1? Tell 172.16.99.254
37	9.164642023	ec:75:0c:c2:17:7a	ec:75:0c:c2:26:42	ARP	42	172.16.99.1 is at ec:75:0c:c2:17:7a

Annex 7.2 - Experiment 6: Simultaneous download on tux92 and on tux93



Annex 7.3 - Experiment 6: Simultaneous download on tux92 and on tux93



Annex 8.1 - download.c

C/C++

```
// main entrypoint for the program

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

#include "parser.h"
#include "ftp_response.h"

#define FTP_PORT 21
#define MAX_BUF_SIZE 10000

/** 
 * Checks if the message was complete
 * @param buf Buffer where the message is stored
 * @param buf_size Buffer size
 * @return true if complete, false otherwise
 */
```

```

bool is_msg_complete(const char* buf, int buf_size) {
    int i = 0;
    bool final_line = false;

    while(i+1 < buf_size) {
        // buffer safety check
        if (i+4 >= buf_size && !final_line)
            return false;

        if (!final_line && (i == 0 || buf[i-1] == '\n')
            && isdigit(buf[i]) && isdigit(buf[i+1]) && isdigit(buf[i+2])
            && buf[i+3] == ' ') {
            // final, skip all until CRLF
            final_line = true;
        }

        if (final_line && buf[i] == '\r' && buf[i+1] == '\n') {
            return true;
        }

        i++;
    }

    return false;
}

/**
 * Reads message pieces and writes them to buf until CRLF is encountered
 * @param socket Socket being read
 * @param buf Buffer where the message is stored
 * @param buf_size Buffer size
 * @return the response code or -1 if something's wrong (e.g. buffer too
small)
*/
int get_message(int socket, char* buf, int buf_size) {
    int total_bytes = 0;

    while (total_bytes < buf_size) {
        long int bytes = recv(socket, buf + total_bytes, buf_size -
total_bytes, 0);
        if (bytes > 0) {
            printf("Bytes read %ld\n", bytes);
        }
        else {
            perror("recv() failed");
            return -1;
        }
    }
}

```

```

        total_bytes += bytes;

        if (is_msg_complete(buf, total_bytes)) {
            break;
        }

        if (total_bytes >= buf_size)
            return -1;

        buf[total_bytes] = '\0';

        // get message code
        int res = (buf[0] - '0')*100 + (buf[1] - '0')*10 + (buf[2] - '0');
        return res;
    }

    /**
     * Sends a message via socket
     * @param socket Socket used to send a message
     * @param buf Where message was
     * @param buf_size Buffer size
     * @return 0 if succeeded, -1 otherwise
     */
    int send_message(int socket, char* buf, int buf_size) {
        long int bytes = send(socket, buf, buf_size, 0);
        if (bytes > 0) {
            printf("Bytes sent %ld\n", bytes);
        } else {
            perror("send()");
            return -1;
        }

        return 0;
    }

    /**
     * Parses the 227 message (PASSIVE MODE ON)
     * @param buf Message to be parsed
     * @param ip New ip
     * @return Port for data socket
     */
    int parse_ip(const char* buf, char* ip) {
        // get to opening paren and skip it
        while (*buf != '(') {
            buf++;

```

```

}

buf++;

int ip_index = 0;
// copy first octet to the ip address
ip[ip_index] = *buf;
buf++;
ip_index++;
if (isdigit(*buf)) {
    ip[ip_index] = *buf;
    buf++;
    ip_index++;
    if (isdigit(*buf)) {
        ip[ip_index] = *buf;
        buf++;
        ip_index++;
    }
}

assert(*buf == ',');

ip[ip_index] = '.';
buf++;
ip_index++;

// copy second octet to the ip address
ip[ip_index] = *buf;
buf++;
ip_index++;
if (isdigit(*buf)) {
    ip[ip_index] = *buf;
    buf++;
    ip_index++;
    if (isdigit(*buf)) {
        ip[ip_index] = *buf;
        buf++;
        ip_index++;
    }
}

assert(*buf == ',');

ip[ip_index] = '.';
buf++;
ip_index++;

// copy third octet to the ip address
ip[ip_index] = *buf;
buf++;
ip_index++;

```

```

if (isdigit(*buf)) {
    ip[ip_index] = *buf;
    buf++;
    ip_index++;
    if (isdigit(*buf)) {
        ip[ip_index] = *buf;
        buf++;
        ip_index++;
    }
}

assert(*buf == ',');
ip[ip_index] = '.';
buf++;
ip_index++;
// copy fourth octet to the ip address
ip[ip_index] = *buf;
buf++;
ip_index++;
if (isdigit(*buf)) {
    ip[ip_index] = *buf;
    buf++;
    ip_index++;
    if (isdigit(*buf)) {
        ip[ip_index] = *buf;
        buf++;
        ip_index++;
    }
}
assert(*buf == ',');
buf++;
ip[ip_index] = '\0';

// Now the port
int port_oct1 = *buf - '0';
buf++;
if (isdigit(*buf)) {
    port_oct1 *= 10;
    port_oct1 += *buf - '0';
    buf++;
    if (isdigit(*buf)) {
        port_oct1 *= 10;
        port_oct1 += *buf - '0';
        buf++;
    }
}
assert(*buf == ',');

```

```

buf++;
// second octet
int port_oct2 = *buf - '0';
buf++;
if (isdigit(*buf)) {
    port_oct2 *= 10;
    port_oct2 += *buf - '0';
    buf++;
    if (isdigit(*buf)) {
        port_oct2 *= 10;
        port_oct2 += *buf - '0';
        buf++;
    }
}
assert(*buf == ')');
printf("oct1 %d oct2 %d\n", port_oct1, port_oct2);

return port_oct1 * 256 + port_oct2;
}

size_t get_file_size(const char* buf) {
    // skip code and space
    buf += 4;

    size_t size = *buf - '0';
    buf++;

    while (isdigit(*buf)) {
        size *= 10;
        size += *buf - '0';
        buf++;
    }

    return size;
}

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

    if (argc != 2) {
        printf("Usage: ftp://[<user>:<password>@]<host>/<url-path>\n");
        return -1;
    }

    Url url;

    // get information from param
    if (parse(argv[1], &url) != 0) {

```

```

        printf("Error parsing url\n");
        return -1;
    }

    printf( "Host: %s\n"
            "IP Address: %s\n"
            "User: %s\n"
            "Password: %s\n"
            "Resource: %s\n"
            "File: %s\n",
            url.hostname,
            url.host_ip,
            url.user,
            url.password,
            url.resource,
            url.file
    );

    // creating the ctrl socket

    // ctrl_socket
    int sockfd;
    struct sockaddr_in server_addr;
    size_t bytes;

    /*server address handling*/
    bzero((char *) &server_addr, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = inet_addr(url.host_ip);      /*32 bit
Internet address network byte ordered*/
    server_addr.sin_port = htons(FTP_PORT);           /*server TCP port must be
network byte ordered */

    /*open a TCP socket*/
    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket()");
        exit(-1);
    }
    /*connect to the server*/
    if (connect(sockfd,
                (struct sockaddr *) &server_addr,
                sizeof(server_addr)) < 0) {
        perror("connect()");
        exit(-1);
    }
}

```

```

char buf[MAX_BUF_SIZE];
// get welcome message
if (get_message(sockfd, buf, MAX_BUF_SIZE) != WELCOME_CODE) {
    printf("Error: did not receive welcome message.\n");
    return -1;
}
printf("%s\n", buf);

// send USER message for authentication
sprintf(buf, "USER %s\r\n", url.user);
if (send_message(sockfd, buf, strlen(buf)) != 0) {
    printf("Failed sending USER message.\n");
    return -1;
}
// (try to) receive 331 (request for password)
if (get_message(sockfd, buf, MAX_BUF_SIZE) != PSSWRD_REQUEST_CODE) {
    printf("Error: did not receive password request\n");
    return -1;
}
printf("%s\n", buf);

// send PASS message
sprintf(buf, "PASS %s\r\n", url.password);
if (send_message(sockfd, buf, strlen(buf)) != 0) {
    printf("Error: could not send the PASS\n");
    return -1;
}
// receive 230 (login successful)
if (get_message(sockfd, buf, MAX_BUF_SIZE) != LOGIN_SUCCESS_CODE) {
    printf("Error: couldn't login\n");
    return -1;
}
printf("%s\n", buf);

// send TYPE message (switching to binary mode)
sprintf(buf, "TYPE I\r\n");
if (send_message(sockfd, buf, strlen(buf)) != 0) {
    printf("Error: couldn't switch to binary mode\n");
    return -1;
}
// receive 200 (switched to binary mode)
if (get_message(sockfd, buf, MAX_BUF_SIZE) != CMD_OK_CODE) {
    printf("Error: couldn't switch to binary mode\n");
    return -1;
}
printf("%s\n", buf);

```

```

// send SIZE message
sprintf(buf, "SIZE %s\r\n", url.resource);
if (send_message(sockfd, buf, strlen(buf)) != 0) {
    printf("Error: couldn't send SIZE message\n");
    return -1;
}
// receive 213 (file size in bytes)
if (get_message(sockfd, buf, MAX_BUF_SIZE) != FILE_STAT_CODE) {
    printf("Error: couldn't get file size\n");
    return -1;
}
printf("%s\n", buf);
// parse the file size
size_t file_size = get_file_size(buf);
printf("file size is %ld\n", file_size);

// send PASV message
sprintf(buf, "PASV\r\n");
if (send_message(sockfd, buf, strlen(buf))) {
    printf("Error: couldn't send PASV message\n");
}
// receive 227 (passive mode entered, plus the new address)
if (get_message(sockfd, buf, MAX_BUF_SIZE) != PASSIVE_MODE_CODE) {
    printf("Error: couldn't enter in PASSIVE mode\n");
    return -1;
}
printf("%s\n", buf);

// compute the new address
char data_ip[] = "255.255.255.255";
int data_port = parse_ip(buf, data_ip);
printf("%s:%d\n", data_ip, data_port);

// create a separate connection to the new address
int data_socket;
struct sockaddr_in data_addr;
bzero((char *) &data_addr, sizeof(data_addr));
data_addr.sin_family = AF_INET;
data_addr.sin_addr.s_addr = inet_addr(data_ip); /*32 bit Internet
address network byte ordered*/
data_addr.sin_port = htons(data_port); /*server TCP port must be
network byte ordered */

/*open a TCP socket*/
if ((data_socket = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    printf("Error: can't create data socket\n");
}

```

```

        return -1;
    }

/*connect to the server*/
if (connect(data_socket,
            (struct sockaddr *) &data_addr,
            sizeof(data_addr)) < 0) {
    perror("connect()");
    exit(-1);
}

// send RETR message
sprintf(buf, "RETR %s\r\n", url.resource);
if (send_message(sockfd, buf, strlen(buf)) != 0) {
    printf("Error: couldn't send RETR message\n");
    return -1;
}
// receive 150 (transfer started)
if (get_message(sockfd, buf, MAX_BUF_SIZE) != OPEN_DATA_CONNECTION_CODE)
{
    printf("Error: couldn't start transfer\n");
    return -1;
}
printf("%s\n", buf);

// read from data socket, write to the file each time
FILE* file = fopen(url.file, "w");
size_t total_bytes = 0;
while (total_bytes < file_size) {
    bytes = recv(data_socket, buf, MAX_BUF_SIZE, 0);
    if (bytes > 0) {
        //printf("Bytes read %ld\n", bytes);
        // comment out for a cleaner download process
    }
    else {
        perror("recv() failed");
        return -1;
    }
    total_bytes += bytes;
    fwrite(buf, 1, bytes, file);
}
if (total_bytes != file_size) {
    printf("Error: transfer was problematic\n");
    return -1;
}

// receive 226 (transfer complete)

```

```

    if (get_message(sockfd, buf, MAX_BUF_SIZE) != TRANSFER_COMPLETE_CODE) {
        printf("Error: transfer was not done\n");
        return -1;
    }
    printf("%s\n", buf);

    // send QUIT message
    sprintf(buf, "QUIT\r\n");
    if (send_message(sockfd, buf, strlen(buf)) != 0) {
        printf("Error: couldn't send QUIT message");
        return -1;
    }
    // receive 221 (bye)
    if (get_message(sockfd, buf, MAX_BUF_SIZE) != BYE_CODE) {
        printf("Error: couldn't get bye\n");
        return -1;
    }
    printf("%s\n", buf);

    // close connections

    if (close(sockfd) < 0) {
        printf("Error: couldn't close ctrl_socket\n");
        return -1;
    }
    if (close(data_socket) < 0) {
        printf("Error: couldn't close data_socket\n");
        return -1;
    }

    return 0;
}

```

Annex 8.2 - parser.h

C/C++

```

#ifndef PARSER_H
#define PARSER_H

#include "url.h"

/**
 * Parses the received url
 * @param raw_url Url in string format

```

```

* @param url Url after being parsed in url format
* @return 0 if succeeded, -1 otherwise
*/
int parse(char *raw_url, Url *url);

#endif

```

Annex 8.3 - parser.c

```

C/C++
#include "parser.h"

#include <string.h>
#include <stdio.h>

#include "utils.h"

int parse(char *raw_url, Url *url) {

    // pointer to update curr url element while preserving original string
    char *ptr = raw_url;

    if (strncmp(ptr, "ftp://", 6) != 0) {
        printf("Invalid protocol in url\n");
        return -1;
    }

    ptr += 6;

    char *at = strchr(ptr, '@');

    if (at != NULL) {
        char *colon = strchr(ptr, ':');

        // get user
        strncpy(url->user, ptr, colon-ptr);
        ptr += colon-ptr+1;

        // get password
        strncpy(url->password, ptr, at-ptr);
        ptr += at-ptr+1;
    } else {
        strcpy(url->user, "anonymous");
    }
}

```

```

        strcpy(url->password, "password");
    }

    char *slash = strchr(ptr, '/');
    strncpy(url->hostname, ptr, slash-ptr);
    ptr += slash-ptr; // dont skip the / char, because it can cause
problem if the resource is the file itself

    strcpy(url->resource, ptr);

    if (getHostIp(url) != 0) return -1;

    // creates a pointer to the last occurrence of /, then copies the file
name to url struct
    char *f = strrchr(ptr, '/');
    strcpy(url->file, f+1);

    return 0;
}

```

Annex 8.4 - utils.h

```

C/C++
#ifndef UTILS_H
#define UTILS_H

#include "url.h"

/**
 * Gets Host IP from a given Url Struct
 * @return 0 if succeeded, -1 otherwise
 */
int getHostIp(Url *url);

#endif

```

Annex 8.5 - utils.c

```
C/C++
#include "utils.h"

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

int getHostIp(Url *url) {
    struct hostent *h;

    if ( (h = gethostbyname(url->hostname)) == NULL) {
        printf("error getting host information\n");
        return -1;
    }

    strcpy(url->host_ip, inet_ntoa(*((struct in_addr *) h->h_addr)) );
    return 0;
}
```

Annex 8.6 - url.h

```
C/C++
#ifndef URL_H
#define URL_H

#define MAX_LENGTH 500

typedef struct Url {
    char user[MAX_LENGTH];
    char password[MAX_LENGTH];
    char hostname[MAX_LENGTH];
    char resource[MAX_LENGTH];

    char host_ip[MAX_LENGTH];
    char file[MAX_LENGTH];
} Url;

#endif
```

Annex 8.7 - ftp_response.h

```
C/C++  
ifndef FTP_RESPONSE_H  
define FTP_RESPONSE_H  
  
define WELCOME_CODE 220  
define PSSWRD_REQUEST_CODE 331  
define LOGIN_SUCCESS_CODE 230  
define CMD_OK_CODE 200  
define FILE_STAT_CODE 213  
define PASSIVE_MODE_CODE 227  
define OPEN_DATA_CONNECTION_CODE 150  
define TRANSFER_COMPLETE_CODE 226  
define BYE_CODE 221  
  
endif
```

Annex 8.8 - Makefile

```
None  
CC = gcc  
CFLAGS = -Wall -Iinclude  
  
SRC = src  
INCLUDE = include  
BIN = bin  
  
# URL for debugging  
#URL1 = ftp://ftp.up.pt/pub/gnu/emacs/elisp-manual-21-2.8.tar.gz  
URL1 = ftp://ftp.up.pt/pub/debian/README.html  
URL2 = ftp://demo:password@test.rebex.net/readme.txt  
URL3 = ftp://anonymous:anonymous@ftp.bit.nl/speedtest/100mb.bin  
  
.PHONY: all  
all: main  
  
main: $(SRC)/*.c  
	$(CC) $(CFLAGS) -o $(BIN)/download $^  
  
# simple command for easier debugging  
.PHONY: run_client_debug  
run_client_debug: main
```

```

@echo ""
@echo "Testing 1st case"
./$(BIN)/download $(URL1)
#
@echo ""
# @echo "Testing 2nd case"
# ./$(BIN)/download $(URL2)
@echo ""
@echo "Testing 3rd case"
./$(BIN)/download $(URL3)

.PHONY: clean
clean:
    rm -rf $(BIN)/download
    rm -rf 100mb.bin
    rm -rf readme.txt
    rm -rf README.html

```

Annex 9.1 - Configuration commands

None

```

# Part 2 - Practical experiments

## Step 0

1. Reset switch and router (in GTK).
   1. Do the router, then move the leftmost cable to SW CON and reset again
   2. Login: admin Password: <blank>

```bash
/system reset-configuration
```

2. Reset networking services

```bash
sudo systemctl restart networking
```

## Exp 1 - Configure an IP Network

### Steps

1. Connect cables from E1_tuxY3 [ether16] and E1_tuxY4 [ether14] to switch
2. Set ip and masks

```

```

```bash
on tux Y3
ifconfig if_e1 up
ifconfig if_e1 <IP> [172.16.Y0.1/24] # this sets the ip and the mask at
the same time

on tux Y4
ifconfig if_e1 up
ifconfig if_e1 <IP> ?? [172.16.Y0.254/24]
```

```

3. Verify connection with ping. i.e, try to ping tuxY3 on tuxY4 and vice versa

Exp 2 - Implement two bridges in a switch

Steps

1. Connect E1_tuxY2 [ether18] to switch and set its ip/mask

```

```bash
on tuxY2; in tux92 if_e1 is enp2s0
ifconfig if_e1 up
ifconfig if_e1 <IP> [172.16.Y1.1/24]
```

```

2. Assure that the cables are correctly connected. Left cable should be in SW CON since we are setting up the switch

3. Add new bridges

```

```bash
/interface bridge add name=bridgeY0
/interface bridge add name=bridgeY1
```

```

4. Remove ports default bridge from tuxY3, tuxY4 and tuxY2.

```

```bash
these ether interface are not fixed. is just to be easier when following
the guide

port for tux93
/interface bridge port remove [find interface=ether16]

port for tux94
/interface bridge port remove [find interface=ether14]
```

```

```
# port for tux92
/interface bridge port remove [find interface=ether18]
``
```

5. Add ports to new bridge

```
```bash
port for tux93
/interface bridge port add bridge=bridge90 interface=ether16

port for tux94
/interface bridge port add bridge=bridge90 interface=ether14

port for tux92
/interface bridge port add bridge=bridge91 interface=ether18
``
```

#### 6. Verify with ping. Pinging from tuxY3 to tuxY4 must succeed, but fail to tuxY2

```
Exp 3 - Configure a Router in Linux
```

#### ### Steps

##### 1. Plug E2\_tux94 [ether15] to switch and set its ip/mask

```
```bash
# on tux Y4
ifconfig if_e2 up
ifconfig if_e2 <IP> ?? [172.16.Y1.253/24]
``
```

2. Add port to tux92 bridge

```
```bash
removing default port
/interface bridge port remove [find interface=ether15]

port for tux94_e2
/interface bridge port add bridge=bridge91 interface=ether15
``
```

##### 3. Enable IP forwarding

```
```bash
sudo sysctl -w net.ipv4.ip_forward=1
```

...

4. Disable ICMP echo-ignore-broadcast

```
```bash
sudo sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=0
```
```

5. Add new routing

```
```bash
on tuxY2; add a "path" to tuxY3 via tuxY4(if_e2)
route add -net 172.16.Y0.0/24 gw 172.16.Y1.253

on tuxY3; add a "path" to tuxY3 via tuxY4(if_e1)
route add -net 172.16.Y1.0/24 gw 172.16.Y0.254
```
```

6. From tuxY3, ping the other network interfaces (172.16.Y0.254, 172.16.Y1.253, 172.16.Y1.1) and verify if there is connectivity

>> clean arp tables with "sudo ip neigh flush all"

Exp 4 - Configure a Commercial Router and Implement NAT

1. Plug Rc ether2 to the switch (ether10)
2. Add Rc port to bridge91

```
```bash
remove default port
/interface bridge port remove [find interface=ether10]

port for router
/interface bridge port add bridge=bridge91 interface=ether10
```
```

3. Change from SW CONS to ROUTER CONS

4. Set router ips via GTK

```
```bash
/ip address add address=172.16.1.Y1/24 interface=ether1

/ip address add address=172.16.Y1.254/24 interface=ether2
```
```

5. Add new routes

```

```bash
on tuxY2
#route add default gw 172.16.Y1.254
route add -net 172.16.1.0/24 gw 172.16.Y1.254

on tuxY3
#route add default gw 172.16.Y0.254
route add -net 172.16.1.0/24 gw 172.16.Y0.254

on tuxY4
#route add default gw 172.16.Y1.254
route add -net 172.16.1.0/24 gw 172.16.Y1.254

on Router serial terminal
/ip route add dst-address=172.16.Y0.0/24 gateway=172.16.Y1.253
```

```

6. Verify with ping on tuxY3. Try to ping 172.16.Y0.254 (tuxY4 e1), 172.16.Y1.253 (tuxY4 e2), 172.16.Y1.1 (tuxY2 e1), 172.16.Y1.254 (Rc)

Exp 5 - DNS

1. In each tux access the file and add nameserver 10.227.20.254

```

```bash
sudo nano /etc/resolv.conf
```

```

2. Some checking and problem resolution

```

```bash
check
ping google.com
ping ftp.netlab.fe.up.pt
```

```

Exp 6 - TCP Connections

Important info

e1_tuxY2

MAC: a2:05:02:f6:c6:5c
IP: 172.16.91.1

ether18

```
### e1_tuxY3

MAC: ec:75:0c:c2:17:7a
IP: 172.16.90.1

ether16

### e1_tuxY4

MAC: ec:75:0c:c2:26:42
IP: 172.16.90.254

ether14

### e2_tuxY4

MAC: ec:75:0c:c2:31:93
IP: 172.16.91.253

ether15

### Rc

IP: 192.168.88.1 (generic)

ip_lab: 172.16.1.99

ip_sw: 172.16.91.254

ether10
```