U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Redes de Computadores

Lab 2 - Computer Networks

Filipa Manita Santos Durão, up201606640
Miguel Pereira Duarte, up201606298
Rui Pedro Moutinho Moreira Alves, up201606746

Summary

For the second lab the development of a download application and the configuration/study of a computer network was requested. The download application should download a single file and follow the FTP (**F**ile **T**ransfer **P**rotocol) standard. The configuration of the computer network should follow the six experiments described in the project's handout and should be studied and analysed using the saved logs.

After the download application was fully implemented, a deep understanding about the FTP protocol was acquired. The application was able to connect to any ftp server and download a file. The experiments that consisted in the configuration of the computer network allowed a good understanding of router and switch configurations, as well as the understanding of the ICMP (**I**nternet **C**ontrol **M**essage **P**rotocol), TCP (**T**ransfer **C**ontrol **P**rotocol) and IP (**I**nternet **P**rotocol) standards in computer network communication.

Introduction

The second project has two main objectives: Developing a Download Application (**DA**) and Configuring a Computer Network (**CN**) to be used alongside the DA to download files from any FTP server in the internet. The DA should follow the FTP standard and its input (the ftp server url) should adopt the URL syntax. The CN should be configured following the six experiments described in the project's handout.

The report starts with the analysis of the developed Download Application, starting with its architecture followed by the report of a successful download. Afterwards, an in-depth analysis of the implemented network configuration and the six implemented experiments will take place. For each experiment, the main objectives, the network architecture and the analysis of the obtained results will be specified. In the end of the report, a set of Attachments (which will be mentioned throughout the report) is available.

Part 1 - Download Application

Architecture of the download application

The developed download application connects to the target ftp server using a socket (the control connection). Afterwards, it sends a set of FTP requests in order to obtain the desired file. It starts by logging into the FTP server with the given credentials. Afterwards, it changes to the requested file's directory and sets the data connection to binary mode. Subsequently, in order to start the file's downloading, a request to enter passive mode is sent. If the request is successful, the server answers with an IP address and a port, which correspond to the data port of the server that the client must connect to. The application then proceeds to connect to it using a second socket. Afterwards, it sends a request (via the control connection) to retrieve the file and proceeds to transfer the file (via the data connection). After the file is transferred completely, the server sends a "transfer complete" message via the control connection. When this message is received, connection with the server is closed by sending a QUIT message and closing the control connection socket.

The download application is divided in four main different modules. The **parser** module is responsible for parsing the server URL (the application's input) and the data connection IP/Port sent by the FTP servers when entering passive mode. The **commands** module is responsible for sending requests to and receiving replies from the server. The **connection** module is responsible for managing the control and data connection's logic and flow. Finally, the **file** module is responsible for transferring the file from the FTP server to the client's machine.

Download Application Experimentation

Various download tests to different FTP servers were made (both on servers that used credentials and servers that didn't require authentication). All of the tests were successful.

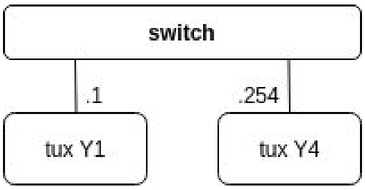
Experimentation also showed that Microsoft FTP servers would reply with different response codes than UPorto's FTP server - for example, UPorto's server replied with code **150 - Opening BINARY mode data connection** and Microsoft servers answered with code **125 - Data connection already open; Transfer starting** after receiving a RETR <file> request. For this reason, compatibility with both kinds of FTP servers was implemented.

The application allows the user to observe the communication process, printing all the requests sent and the responses received, with the respective reply codes. A successful download report can be found in **Attachment D - Successful Download Report** in the end of this report.

Part 2 - Network configuration and analysis

For all of the following experiments, all the addresses and vlan names assume that the considered workstation is station 4. The diagrams consider a generic situation, in which Y represents the station number.

Experiment 1



Experiment Objectives: Configuring an IP Network

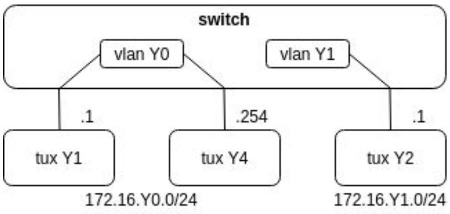
For the first experiment, both tux 41 and tux 44 were connected to the switch (in any port, as long as they did not belong to any pre-configured vlan).

In order to configure the IP network, the following configuration was made :
tux41 >> ifconfig eth0 172.16.40.1/24
tux44 >> ifconfig eth0 172.16.40.254/24

The ping command was used to test connectivity between the two machines. This command sends ICMP echo (ping) request packets to the desired host, which replies with ICMP echo (ping) reply packets. In the case that the emitter of the ping does not have an ARP table entry with the IP of the receiver, then an ARP request is sent to the subnetwork broadcast channel (with an empty MAC address and the IP address of the receiver). The receiver then identifies itself as the correct receiver, replying with an ARP reply message, specifying its own MAC address. Thus, the emitter is now able to send the ICMP echo (ping) request packets to the desired host.

The IP and ARP packets can be identified via the Ethernet II layer data type. The ICMP packets can be identified via the IPV4 layer protocol field (as observed in Wireshark).

Experiment 2



Experiment Objectives: Implementing two virtual lans in a switch

In order to configure the network, the following configuration was made, in addition to the configuration in the previous experiments:
tux41 >> route add default gw 172.16.40.254 (Added a default gateway from tux41 to tux44)

tux42 >> ifconfig eth0 172.16.41.1/24

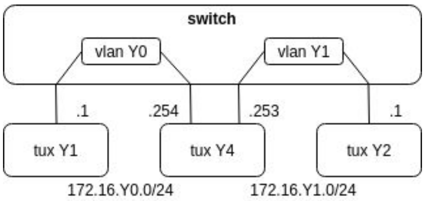
Switch configuration:
- Connected tux41 eth0 port to Fa0/14
- Connected tux44 eth0 port to Fa0/16
- Connected tux42 eth0 port to Fa0/13
- Added Fa0/14 and Fa0/16 to **vlan 40** and added Fa0/13 to **vlan 41** (using the switch configuration commands specified in **attachment B.1**).

In this experiment two vlans were created. Vlan 40 connecting tux41 and tux44 and vlan 41 only containing tux42. The ping command was used to test the connectivity between the machines. It was verified that tux41 and tux44 could communicate with each other but that tux42 could not reach any other machine. This verifies the theoretical hypothesis, in which two machines in separate virtual lans cannot communicate.

Afterwards, a default gateway from tux41 to tux44 was added . When pinging from tux41 to to tux42, tux41 sent the ICMP echo (ping) request packets to tux44 (the default gateway). These ICMP packets contained the MAC Address of the tux44 machine and the tux42 IP address (since tux42 was the final target receiver and tux44 was the next hop in the ping route).

In this configuration there are 2 broadcast domains, one for each subnetwork (vlan 40 and vlan 41 - 172.168.40.255 and 172.168.41.255 respectively).

Experiment 3



Experiment Objectives: Configuring a Router in Linux

In order to configure the network, the following configuration was made, in addition to the configurations in the previous experiments:
tux44 >> ifconfig eth1 172.16.41.253/24
tux44 >> echo 1 > /proc/sys/net/ipv4/ip_forward (Enabling IP forwarding)
tux42 >> route add default gw 172.16.41.253 (Added a default gateway from tux42 to tux44)

Switch configuration:
- Connected tux44 eth1 port to Fa0/15
- Added Fa0/15 to **vlan 41** (using the switch configuration commands specified in **attachment B.1**).

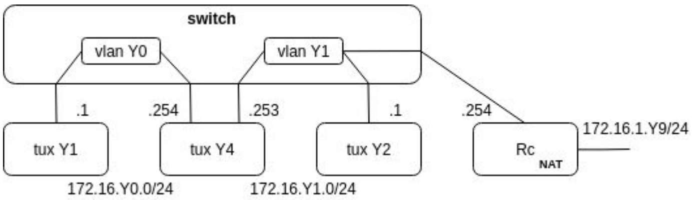
In this experiment, tux44 was transformed into a router by being present in both vlan 40 and vlan 41 and by having ip forwarding enabled. Due to this, tux44 is able to route packets from vlan 40 to vlan 41 and vice-versa, thus connecting both subnetworks. In order to verify connectivity, all the network interfaces were pinged.

The communication between all network interfaces is possible, since both tux41 and tux42 possess a default gateway to tux44 eth0 and eth1 interfaces (respectively). The default gateways are used as intermediate packet destinations when the target is not present in the emitter's subnetwork.

For example, when tux41 tries to ping tux42, it does not find tux42's ip address in its subnetwork (as tux42's ip address belongs to the 172.16.41.0/24 range). Thus, the ping request packet is sent to tux44's eth0 interface (containing tux42 destination IP tux44 eth0 interface MAC address), which now has a direct route to tux42, since tux44's eth1 interface is in the same subnetwork as the latter. When tux42 receives the ping request, the reply it tries to send follows the same process, because tux42 notices that the desired packet destination host is not in its subnetwork and forwards it to tux44 which handles the routing as described before (but in the inverse direction).

Each entry of the routing table contemplates the following information: Network Destination, Netmask, Destination Gateway, Destination Interface and Metric (the metric is used to choose the best route, if several are available).

Experiment 4



Experiment Objectives: Configuring a Commercial Router and Configuring NAT

In order to configure the network, the following configuration was made, in addition to the configurations in the previous experiments:
tux42 >> route del default gw 172.16.41.253 (Removed the default gateway from tux42 to tux44)
tux42 >> route add default gw 172.16.41.254 (Added a default gateway from tux42 to Rc)
tux44 >> route add default gw 172.16.41.254 (Added a default gateway from tux44 to Rc)

Switch configuration:
- Connected Router GE0/0 port to Gi0/1
- Connected Router GE0/1 port to the central patch (port labeled 4.1)
- Added Gi0/1 to **vlan 41** (using the switch configuration commands specified in **attachment B.1**).

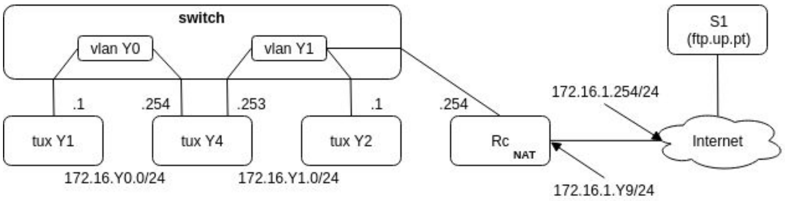
Router configuration:
- Configured NAT inside in the Router gigabit ethernet 0 interface and NAT outside in the Router gigabit ethernet 1 interface, and configured routing and the valid access list (using the switch configuration commands specified in **attachment B.3**).

The default gateways of tux42 and tux44 were changed in order for both of the devices to be able to connect to the internet via Rc. After configuring them as mentioned, their routing is done via Rc by default if the specified IP is not in the device's subnetwork (as explained in previous experiments).

When pinging tux41 from tux42, with "accept-redirects" disabled on tux42, the ICMP echo packets followed the path tux42→Rc→tux44→tux41. The ICMP response packets followed the same path in the opposite way: tux41→tux44→Rc→tux42. Subsequently, “accept-redirects” was enabled on tux42. When tux42 attempted pinging tux41, tux42 sent the ICMP packet to Rc (its default gateway). Rc, after consulting its routing table, found that the next-hop to reach tux41 was tux44’s eth1 interface. Rc forwarded the packet to tux44 and also sent an ICMP redirect message to tux42. This informed tux42 that the best route to reach tux41 is by way of tux44 (interface eth1). In the following ICMP packets destined for tux41, tux42 forwarded all the traffic directly to tux44 (without hopping through Rc).

NAT (**N**etwork **A**ddress **T**ranslation) is a protocol that translates a public IP address to the destination IP address inside a private network. This avoids making the IP of the destination public. In order to test internet connectivity, 8.8.8.8 (the static IP of Google’s Public DNS service) was pinged from tux41. This IP was used because it is an address that does not require DNS and is always online, thus perfect for testing connectivity.

Experiment 5



Experiment Objectives:
Configuring DNS in tux linux machines

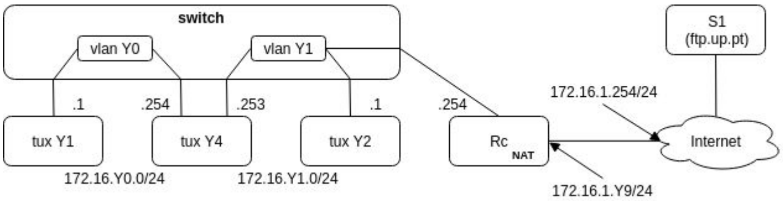
In order to configure DNS (**D**omain **N**ame **S**ervice) on the tux machines, the following configuration was made, in addition to the configurations in the previous experiments:

```
tux41 >> echo -e "search netlab.fe.up.pt\nnameserver 172.16.1.1" > /etc/resolv.conf
tux42 >> echo -e "search netlab.fe.up.pt\nnameserver 172.16.1.1" > /etc/resolv.conf
tux44 >> echo -e "search netlab.fe.up.pt\nnameserver 172.16.1.1" > /etc/resolv.conf
```

After DNS was configured on all the tux machines, www.google.com was pinged from tux41 to test the DNS configuration. The wireshark analysis of the ICMP echo ping showed that a DNS “Standard Query” was sent to the target IP 172.16.1.1 (the IP address configured for DNS resolution) containing the *www.google.com* query field. Afterwards (following DNS resolution), a DNS “Standard Query Response” was issued by 172.16.1.1 to tux41 containing the translated IP (which was 216.58.210.164). After these packets were exchanged, the ICMP packets were swapped directly between tux41 and 216.58.210.164 (*www.google.com*’s IP address).

Subsequently, a second DNS packet exchange was observed. This exchange corresponded to a reverse dns lookup (rDNS). Reverse DNS consists in mapping an IP address to a domain name, using the special domain *in-addr.arpa*. In this domain, the IP addresses are represented with their four octets reversed, and appended with the suffix .in-addr.arpa. In the example observed in wireshark, tux41 sent a DNS “Standard Query” to tux 172.16.1.1 containing 164.210.58.216.in-addr.arpa (these octets correspond to the four 216.58.210.164 Google IP octets reversed).

Experiment 6



Experiment Objectives: Using the developed download application to download a file from sigarra's FTP servers using the configured network

In this experiment, the same network configuration as in experiment 5 was used.

The developed download application was used in tux41 to download a file from UPorto's FTP server. Throughout the application lifetime, two TCP connections were open. The communication started with the client sending a SYN TCP packet to the server (attempting to start the control connection), to which the server responded with SYN-ACK, accepting the control connection. Afterwards, a set of FTP packets were sent between the client and the server. For each FTP packet sent, a TCP ACK packet was received. After the client sent the request to enter passive mode and the server responded with the IP/Port that should be used by the client to download the file, the client sent another SYN TCP packet (attempting to start the data connection). Once again, the server responded with SYN-ACK, accepting the data connection. Subsequently, a set of FTP-DATA packets were sent from the server to the client (which responded with TCP ACK packets to each of the FTP-DATA packets). After the file transfer was complete, the server sent a FIN-ACK TCP packet in order to terminate TCP connection and the client responded with a TCP ACK, acknowledging the data connection termination. The client then sent an FTP QUIT request, which resulted in the server terminating the control connection (in the same way the data connection was terminated).

In order to visualize the congestion avoidance mechanism, the following graph was plotted with the transfer data captures using wireshark:

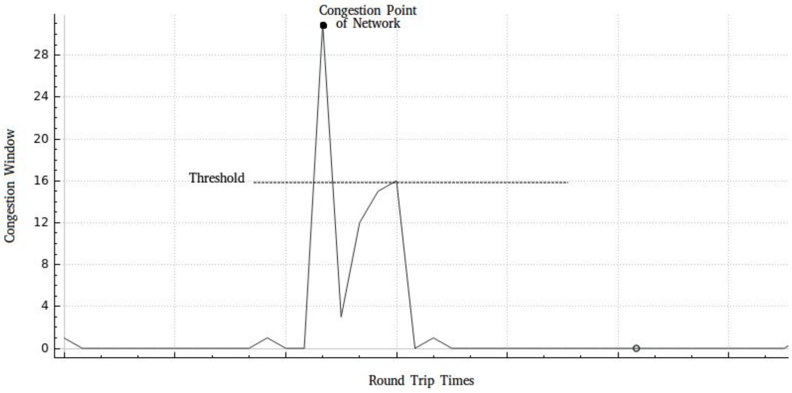


Chart.1. TCP Congestion Avoidance Mechanism when download file from UPorto's FTP server in tux41

In **Chart 1**, it is visible that the congestion window quickly increases after the transfer process starts. After the congestion point of the network is reached, the congestion window is reset and its threshold is set to half the size of the congestion window when the network's congestion point was reached. However, before the congestion avoidance phase was reached, the file download finished (causing the congestion window to decrease to zero).

For the second part of the experiment, a download was started on tux42 (using the developed FTP download application) while the download on tux41 did not yet finish. In order to analyse the impact of the second download in the ftp connection, the following graph was plotted with the transfer data captures using wireshark on tux41:

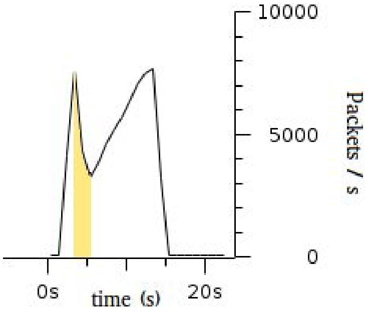


Chart.2. TCP Connection throughput variation with multiple download

The area highlighted in yellow represents the period of time during which tux42 was downloading the file. The TCP throughput in tux41's download decreased significantly during this phase, due to the network's bandwidth limitations. After this simultaneous download finished, it is possible to observe the rise of the TCP throughput once again, as tux41 now had a larger bandwidth available to it.

Conclusions

The implemented download application achieved the desired results, being able to download files from any FTP server using the FTP standard, which allowed a very good understanding of this protocol.

The network experiments allowed the understanding and consolidation of various computer network configuration aspects, such as configuring ethernet interfaces in linux machines, configuring virtual lans with multiple machines, configuring linux routers to exchange packets between distinct virtual lans and configuring NAT and DNS in a commercial router.

The analysis of the various experiments logs captured from the different tux machines using Wireshark also allowed a good understanding of how the different communication layers take part in a computer network

The network experiments also provided a deep understanding about the role of the various network communication protocols in a computer network, such as the ICMP (**I**nternet **C**ontrol **M**essage **P**rotocol), ARP (**A**ddress **R**esolution **P**rotocol), IP (**I**nternet **P**rotocol) and TCP (**T**ransmission **C**ontrol **P**rotocol) protocols, as well as the DNS (**D**omain **N**ame **S**ystem) computer naming system.

References

- Active FTP vs. Passive FTP, http://slacksite.com/other/ftp.html#passive
- RFC 959 - File Transfer Procol (FTP), https://www.w3.org/Protocols/rfc959/
- RFC 1738 - Uniform Resource Locator (URL), https://www.ietf.org/rfc/rfc1738.txt

Attachment A - Download Application Source Code

```
/*************************************
    download.c
*************************************/

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include "connection.h"
#include "commands.h"
#include "parser.h"

int main(int argc, char* argv[]) {
    if (argc != 2) {
        fprintf(stderr, "usage: %s ftp://[<user>:<password>@]<host>/<url-path>.\n", argv[0]);
        exit(INVALID_ARGS);
    }

    char* user = NULL;
    char* password = NULL;
    char* host = NULL;
    char* path = NULL;
    char* file = NULL;

    validate_url(argv[1], &user, &password, &host, &path, &file);

    char* ip = NULL;
    if (hostname_to_ip(host, &ip) != 0) {
        exit(HOSTNAME_TRANSLATION_ERROR);
    }

    if (transfer_file(user, password, ip, path, file) != 0) {
        exit(FILE_TRANSFER_ERROR);
    }

    free(user);
    free(password);
    free(host);
    free(ip);
    free(path);
    free(file);

    return 0;
}

/*************************************
    parser.h
*************************************/

#ifndef _PARSER_H_
#define _PARSER_H_

#define NUM_PASV_FIELDS 6
#define IP_STRING_SIZE 16

#define INVALID_ARGS        -1
#define INVALID_URL         -2
#define INVALID_USERNAME    -3
#define INVALID_PASSWORD    -4
```

```
#define HOST_UNSPECIFIED    -5
#define INVALID_HOST        -6
#define INVALID_PATH        -7
#define URL_START       "ftp://"

int parsePASV(const char* pasv, char** ip, unsigned* port);

void validate_url(const char* url, char** user, char** password, char** host, char** path,
char** file);

#endif //_PARSER_H_

/*************************************
    parser.c
*************************************/

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include "commands.h"
#include "parser.h"

static size_t last_index_of(const char * str, const char to_find);
static bool url_has_password(const char* url);
static bool url_has_user(const char* url);
static bool host_is_specified(const char* url);

int parsePASV(const char* pasv, char** ip, unsigned* port) {
    unsigned short h1, h2, h3, h4, p1, p2;

    if (sscanf(pasv, PASV_SUCCESS, &h1, &h2, &h3, &h4, &p1, &p2) != NUM_PASV_FIELDS) {
        return -1;
    } else {
        *port = (p1<<8) | p2;
        *ip = malloc(IP_STRING_SIZE * sizeof(**ip));
        snprintf(*ip, IP_STRING_SIZE, "%d.%d.%d.%d", h1, h2, h3, h4);
        return 0;
    }
}

void validate_url(const char* url, char** user, char** password, char** host, char** path,
char** file) {
    if (strncmp(url, URL_START, strlen(URL_START)) != 0) {
        fprintf(stderr, "Invalid URL. URL should start with '" URL_START "'.\n");
        exit(INVALID_URL);
    }

    size_t index = strlen(URL_START);

    if (url_has_user(url)) {
        bool has_password = url_has_password(url);

        size_t user_len = 0, password_len = 0;

        for (; url[index] != (has_password ? ':' : '@') ; ++index) {
            user_len++;
        }
        if (user_len == 0) {
            fprintf(stderr, "Invalid URL. User can not be empty.\n");
            exit(INVALID_USERNAME);
        }
        *user = strndup(url + index - user_len, user_len);

        if (has_password) {
            index++;    //ignore ':' char
            for (; url[index] != '@'; ++index) {
                password_len++;
            }

            *password = strndup(url + index - password_len, password_len);
        } else {
            *password = strndup("", 0);
        }

        index++;    //ignore '@' char
    } else {
        *user = strndup("", 0);
        *password = strndup("", 0);
    }

    if (!host_is_specified(url + index)) {
        fprintf(stderr, "Invalid URL. URL must include host and path.\n");
        exit(HOST_UNSPECIFIED);
    }

    size_t host_len = 0;
    for (; url[index] != '/'; ++index) {
        host_len++;
    }
    if (host_len == 0) {
        fprintf(stderr, "Invalid URL. Host can not be empty.\n");
        exit(INVALID_HOST);
    }

    *host = strndup(url + index - host_len, host_len);
    index++;

    size_t path_len = strlen(url + index);
    if (path_len == 0) {
        fprintf(stderr, "Invalid URL. Path can not be empty.\n");
        exit(INVALID_PATH);
    }

    // Actually the index of the '/'
    size_t file_name_index = last_index_of(url + index, '/');
    if (file_name_index == 0) {
        // There is no path specified, file is in root
        *path = strndup(".", 1);
        *file = strndup(url + index, path_len);
    } else {
        // There is a path AND a file
        *path = strndup(url + index, file_name_index);
        *file = strndup(url + index + file_name_index + 1, path_len - file_name_index - 1);
    }

}

static size_t last_index_of(const char * str, const char to_find) {
    const size_t str_size = strlen(str);
    size_t i = str_size - 1;
    for (; i > 0; --i) {
        if (str[i] == to_find) {
            return i;
        }
    }

    return 0;
}

static bool url_has_user(const char* url) {
    size_t len = strlen(url);

    size_t i;
    for (i = strlen(URL_START); i < len; ++i) {
        if (url[i] == '@') {
            return true;
        }
    }

    return false;
}

static bool url_has_password(const char* url) {
    size_t len = strlen(url);
    bool colon_found = false;
    bool at_sign_found = false;

    size_t i;
    for (i = strlen(URL_START); i < len; ++i) {
        // Check for : separator, separating the user and the password
        if (url[i] == ':') {
            colon_found = true;
        }
        // Check for @ separator, separating a user:password block from the url
        else if (url[i] == '@' && colon_found) {
            at_sign_found = true;
        }
    }

    return colon_found && at_sign_found;
}

static bool host_is_specified(const char* url) {
    size_t len = strlen(url);

    size_t i;
    for (i = strlen(URL_START); i < len; ++i) {
        if (url[i] == '/') {
            return true;
        }
    }

    return false;
}



/**************************************
    connection.h
**************************************/

#ifndef _CONNECTION_H_
#define _CONNECTION_H_
```

```
#define HOSTNAME_TRANSLATION_ERROR          1
#define SOCKET_ERROR                        2
#define CONNECTION_ERROR                    3
#define FILE_TRANSFER_ERROR                 4
#define LOGIN_ERROR                         5
#define CHANGE_DIR_ERROR                    6
#define CWD_ERROR                           7
#define SET_BINARY_MODE_ERROR               8
#define SET_PASISVE_MODE_ERROR              9
#define PARSE_PASV_FAILED                   10
#define REQUEST_FILE_FAILED                 11
#define DOWNLOAD_FILE_FAILED                12
#define RETRIEVE_FINAL_RESPONSE_FAILED      13
#define FAILED_FILE_TRANSFER                14
#define RETR_ERROR                          15
#define RETR_FINAL_ERROR                    16
#define READ_INITIAL_RESPONSE_ERROR         17

#define FTP_CONTROL_PORT 21

int hostname_to_ip(const char* hostname, char** ip);

int transfer_file(const char* user, const char* password, const char* host, const char* path,
const char* file);

int connect_to_ip(const char * ip, unsigned );

#endif  //_CONNECTION_H_

/*************************************
    connection.c
*************************************/

#include "connection.h"
#include "commands.h"
#include "parser.h"
#include "file.h"
#include <stdio.h>
#include <stdlib.h>
#include <netdb.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <unistd.h>


static int login(int command_socketfd, const char* user, const char* password);
static void close_connection(int command_socketfd);
static int change_directory(int command_socketfd, const char* path);
static int set_binary_mode(int command_socketfd);
static int set_passive_mode(int command_socketfd, int* data_socketfd);
static int request_file(int command_socketfd, const char * file);
static int read_retrieve_final_response(int command_socketfd);
static int read_initial_response(int command_socketfd);


int hostname_to_ip(const char* hostname, char** ip) {
    struct hostent * h;

    if ((h = gethostbyname(hostname)) == NULL) {
        fprintf(stderr, "Could not translate hostname %s to an IP address\n", hostname);
        return HOSTNAME_TRANSLATION_ERROR;
    }

    char * temp_ip = inet_ntoa(*((struct in_addr *) h->h_addr_list[0]));
    *ip = strndup(temp_ip, strlen(temp_ip));

    return 0;
}

int transfer_file(const char* user, const char* password, const char* ip, const char* path,
const char* file) {
    int command_socketfd = connect_to_ip(ip, FTP_CONTROL_PORT);

    if (command_socketfd < 0) {
        fprintf(stderr, "Error creating command socket\n");
        return SOCKET_ERROR;
    }

    // Read initial response
    if (read_initial_response(command_socketfd) != 0) {
        fprintf(stderr, "Failed to read initial response!\n");
        return READ_INITIAL_RESPONSE_ERROR;
    }

    // Login to the server
    if (login(command_socketfd, user, password) != 0) {
        fprintf(stderr, "Login failed!\n");
        return LOGIN_ERROR;
    }

    // Change directory to the desired one
    if (change_directory(command_socketfd, path) != 0) {
        fprintf(stderr, "Change directory failed!\n");
        return CHANGE_DIR_ERROR;
    }

    // Change to binary mode
    if (set_binary_mode(command_socketfd) != 0) {
        fprintf(stderr, "Set binary mode failed!\n");
        return SET_BINARY_MODE_ERROR;
    }

    // Enter passive mode
    int data_socketfd;
    if (set_passive_mode(command_socketfd, &data_socketfd) != 0) {
        fprintf(stderr, "Set passive mode failed!\n");
        return SET_PASISVE_MODE_ERROR;
    }

    // Requesting file and reading initial response
    if (request_file(command_socketfd, file) != 0) {
        fprintf(stderr, "Failure in requesting file!\n");
        return REQUEST_FILE_FAILED;
    }

    // Downloading file
    if (copy_file(data_socketfd, file) != 0) {
        fprintf(stderr, "Failure in downloading file!\n");
        return DOWNLOAD_FILE_FAILED;
    }

    // Reading retrieve final response
    if (read_retrieve_final_response(command_socketfd) != 0) {
        fprintf(stderr, "Error reading retrieve final response!\n");
        return RETRIEVE_FINAL_RESPONSE_FAILED;
    }

    // Close connection
    close_connection(command_socketfd);

    // Close socket
    close(command_socketfd);
    close(data_socketfd);

    return 0;
}

static int login(int command_socketfd, const char* user, const char* password) {
    if (strncmp(user, "", 1) == 0) {
        // Unauthenticated server, no login necessary
        return 0;
    }

    char* username_cmd = malloc((strlen(user) + strlen(USER) + 2) * sizeof(*username_cmd));
            char*   password_cmd   =   malloc((strlen(password)   +   strlen(PASS)   +   2)   *
sizeof(*password_cmd));

    if (username_cmd == NULL || password_cmd == NULL) {
        free(username_cmd);
        free(password_cmd);
        return MALLOC_ERROR;
    }

    // Building Username Command
    username_cmd[0] = '\0';
    strcat(username_cmd, USER);
    strcat(username_cmd, " ");
    strcat(username_cmd, user);

    // Building Password Command
    password_cmd[0] = '\0';
    strcat(password_cmd, PASS);
    strcat(password_cmd, " ");
    strcat(password_cmd, password);

    unsigned short response_code;
    char* response = NULL;
    size_t response_size;

    // Send Username
    if (send_command(command_socketfd, username_cmd) != 0) {
        fprintf(stderr, "Failed to send command: %s\n", username_cmd);
        free(username_cmd);
        free(password_cmd);
        return SENDING_COMMAND_ERROR;
    }

    if (DEBUG_MODE) {
        printf("->> %s\n", username_cmd);
    }
    free(username_cmd);

     if (read_command_reply(command_socketfd, &response_code, &response, &response_size) != 0)
{
        fprintf(stderr, "Failed to read user command response\n");
        free(response);
        return READING_RESPONSE_ERROR;
    }

    if (DEBUG_MODE) {
        printf("%s\n", response);
    }

    if (response_code != USER_SUCCESS_CODE) {
        fprintf(stderr, "Login failed (user)\nResponse: %hd - %s\n", response_code, response);
        free(response);
        return LOGIN_ERROR;
    }

    // Because we are reusing variables
    free(response);
    response = NULL;

    // Send Password
    if (send_command(command_socketfd, password_cmd) != 0) {
        fprintf(stderr, "Failed to send command: %s\n", password_cmd);
        free(password_cmd);
        return SENDING_COMMAND_ERROR;
    }

    if (DEBUG_MODE) {
        printf("->> %s ****\n", PASS);
    }
    free(password_cmd);

     if (read_command_reply(command_socketfd, &response_code, &response, &response_size) != 0)
{
        fprintf(stderr, "Failed to read pass command response\n");
        free(response);
        return READING_RESPONSE_ERROR;
    }

    if (DEBUG_MODE) {
        printf("%s\n", response);
    }

    if (response_code != PASS_SUCCESS_CODE) {
        fprintf(stderr, "Login failed (pass)\nResponse: %hd - %s\n", response_code, response);
        free(response);
        return LOGIN_ERROR;
    }

    free(response);
    return 0;
}

static void close_connection(int command_socketfd) {
    if (send_command(command_socketfd, QUIT) != 0) {
        fprintf(stderr, "Failed to send command: %s\n", QUIT);
        return;
    }

    if (DEBUG_MODE) {
        printf("->> %s\n", QUIT);
    }
    unsigned short response_code;
    char* response = NULL;
    size_t response_size;

    if (read_command_reply(command_socketfd, &response_code, &response, &response_size) != 0)
{
        fprintf(stderr, "Failed to read quit command response\n");
        free(response);
        return;
    }

    if (DEBUG_MODE) {
        printf("%s\n", response);
    }

    if (response_code != QUIT_SUCCESS_CODE) {
        fprintf(stderr, "Quit failed\nResponse: %hd - %s\n", response_code, response);
        free(response);
        return;
    }

    free(response);
}

static int change_directory(int command_socketfd, const char* path) {
    char* change_dir_command = malloc((strlen(path) + strlen(CWD) + 2) *
sizeof(*change_dir_command));

    if (change_dir_command == NULL) {
        return MALLOC_ERROR;
    }

    // Building Username Command
    change_dir_command[0] = '\0';
    strcat(change_dir_command, CWD);
    strcat(change_dir_command, " ");
    strcat(change_dir_command, path);

    if (send_command(command_socketfd, change_dir_command) != 0) {
        fprintf(stderr, "Failed to send command: %s\n", change_dir_command);
        free(change_dir_command);
        return SENDING_COMMAND_ERROR;
    }

    if (DEBUG_MODE) {
        printf("->> %s\n", change_dir_command);
    }
    free(change_dir_command);

    unsigned short response_code;
    char* response = NULL;
    size_t response_size;

    if (read_command_reply(command_socketfd, &response_code, &response, &response_size) != 0)
{
        fprintf(stderr, "Failed to read cwd command response\n");
        free(response);
        return READING_RESPONSE_ERROR;
    }

    if (DEBUG_MODE) {
        printf("%s\n", response);
    }

    if (response_code != CWD_SUCCESS_CODE) {
        fprintf(stderr, "CWD failed\nResponse: %hd - %s\n", response_code, response);
        free(response);
        return CWD_ERROR;
    }

    free(response);
    return 0;
}

static int set_binary_mode(int command_socketfd) {
    if (send_command(command_socketfd, TYPE_BINARY) != 0) {
        fprintf(stderr, "Failed to send command: %s\n", TYPE_BINARY);
        return SENDING_COMMAND_ERROR;
    }

    if (DEBUG_MODE) {
        printf("->> %s\n", TYPE_BINARY);
    }

    unsigned short response_code;
    char* response = NULL;
    size_t response_size;

    if (read_command_reply(command_socketfd, &response_code, &response, &response_size) != 0)
{
        fprintf(stderr, "Failed to read set binary mode command response\n");
        free(response);
        return READING_RESPONSE_ERROR;
    }

    if (DEBUG_MODE) {
        printf("%s\n", response);
    }

    if (response_code != TYPE_SUCCESS_CODE) {
        fprintf(stderr, "Set binary mode failed\nResponse: %hd - %s\n", response_code,
response);
        free(response);
        return INVALID_RESPONSE;
    }

    free(response);
    return 0;
}

static int set_passive_mode(int command_socketfd, int* data_socketfd) {
    if (send_command(command_socketfd, PASV) != 0) {
        fprintf(stderr, "Failed to send command: %s\n", PASV);
        return SENDING_COMMAND_ERROR;
    }

    if (DEBUG_MODE) {
        printf("->> %s\n", PASV);
    }

    unsigned short response_code;
    char* response = NULL;
    size_t response_size;
    if (read_command_reply(command_socketfd, &response_code, &response, &response_size) != 0)
{
        fprintf(stderr, "Failed to read set passive mode command response\n");
        free(response);
        return READING_RESPONSE_ERROR;
    }

    if (DEBUG_MODE) {
        printf("%s\n", response);
    }

    if (response_code != PASV_SUCCESS_CODE) {
        fprintf(stderr, "Set passive mode failed\nResponse: %hd - %s\n", response_code,
response);
        free(response);
        return INVALID_RESPONSE;
    }

    char* ip_pasv = NULL;
    unsigned port_pasv;

    if (parsePASV(response, &ip_pasv, &port_pasv) != 0) {
        fprintf(stderr, "Failed to parse passive mode response\n");
        free(response);
        return PARSE_PASV_FAILED;
    }

    free(response);

    // Connecting to the data socket so that the process can resume
    if ((*data_socketfd = connect_to_ip(ip_pasv, port_pasv)) < 0) {
        fprintf(stderr, "Could not open connection to the data port\n");
        free(ip_pasv);
    }

    free(ip_pasv);
    return 0;
}

int request_file(int command_socketfd, const char * file) {
    char* request_file_command = malloc((strlen(file) + strlen(RETR) + 2) *
sizeof(*request_file_command));

    if (request_file_command == NULL) {
        return MALLOC_ERROR;
    }

    // Building Username Command
    request_file_command[0] = '\0';
    strcat(request_file_command, RETR);
    strcat(request_file_command, " ");
    strcat(request_file_command, file);

    if (send_command(command_socketfd, request_file_command) != 0) {
        fprintf(stderr, "Failed to send command: %s\n", request_file_command);
        free(request_file_command);
        return SENDING_COMMAND_ERROR;
    }

    if (DEBUG_MODE) {
        printf("->> %s\n", request_file_command);
    }
    free(request_file_command);

    unsigned short response_code;
    char* response = NULL;
    size_t response_size;

    if (read_command_reply(command_socketfd, &response_code, &response, &response_size) != 0)
{
        fprintf(stderr, "Failed to read retr initial command response\n");
        free(response);
        return READING_RESPONSE_ERROR;
    }

    if (DEBUG_MODE) {
        printf("%s", response);
    }

    if (response_code != RETR_INITIAL_SUCCESS_CODE && response_code !=
RETR_INITIAL_SUCCESS_CODE_2) {
        fprintf(stderr, "RETR failed\nResponse: %hd - %s\n", response_code, response);
        free(response);
        return RETR_ERROR;
    }

    free(response);

    return 0;
}

int read_retrieve_final_response(int command_socketfd) {
    unsigned short response_code;
    char* response = NULL;
    size_t response_size;

    if (read_command_reply(command_socketfd, &response_code, &response, &response_size) != 0)
{
        fprintf(stderr, "Failed to read retr final command response\n");
        free(response);
        return READING_RESPONSE_ERROR;
    }

    if (DEBUG_MODE) {
        printf("%s\n", response);
    }

    if (response_code != RETR_FINAL_SUCCESS_CODE) {
        fprintf(stderr, "RETR final failed\nResponse: %hd - %s\n", response_code, response);
        free(response);
        return RETR_FINAL_ERROR;
    }

    free(response);

    return 0;
}

int read_initial_response(int command_socketfd) {
    unsigned short response_code;
    char* response = NULL;
    size_t response_size;
```

```
     if (read_command_reply(command_socketfd, &response_code, &response, &response_size) != 0)
{
        fprintf(stderr, "Error reading initial response!\n");
        free(response);
        return ERROR_READING_INITIAL_RESPONSE;
    }

    if (DEBUG_MODE) {
        printf("%s\n", response);
    }

    if (response_code != INITIAL_CONNECTION_CODE) {
          fprintf(stderr, "Invalid initial response code\nResponse: %hd - %s", response_code,
response);
        free(response);
        return INVALID_RESPONSE;
    }

    free(response);

    return 0;
}

/**************************************
    commands.h
**************************************/

#ifndef _COMMANDS_H_
#define _COMMANDS_H_

#include <unistd.h>

#define DEBUG_MODE  1

// Commands
#define USER        "USER"
#define PASS        "PASS"
#define CWD         "CWD"
#define TYPE_BINARY "TYPE I"
#define PASV        "PASV"
#define RETR        "RETR"
#define QUIT        "QUIT"

// Replies
#define INITIAL_CONNECTION_CODE         220
#define USER_SUCCESS_CODE               331
#define USER_SUCCESS                    "331 Please specify the password."
#define PASS_SUCCESS_CODE               230
#define PASS_SUCCESS                    "230 Login successful."
#define CWD_SUCCESS_CODE                250
#define CWD_SUCCESS                     "250 Directory successfully changed."
#define CWD_FAILURE_CODE                550
#define CWD_FAILURE                     "550 Failed to change directory."
#define TYPE_SUCCESS_CODE               200
#define TYPE_SUCCESS                    "200 Switching to Binary mode."
#define PASV_SUCCESS_CODE               227
#define PASV_SUCCESS                    "227 Entering Passive Mode (%hd,%hd,%hd,%hd,%hd,%hd)."
#define RETR_INITIAL_SUCCESS_CODE       150
#define RETR_INITIAL_SUCCESS              "150 Opening BINARY mode data connection for <file>
(<size> bytes)."
#define RETR_INITIAL_SUCCESS_CODE_2     125
#define RETR_INITIAL_SUCCESS_2          "125 Data connection already open; Transfer starting."
```

```
#define RETR_FINAL_SUCCESS_CODE         226
#define RETR_FINAL_SUCCESS              "226 Transfer complete."
#define RETR_FAILURE_CODE               550
#define RETR_FAILURE                    "550 Failed to change directory."
#define QUIT_SUCCESS_CODE               221
#define QUIT_SUCCESS                    "221 Goodbye."

#define CODE_SIZE                       3
#define RESPONSE_MAX_SIZE               1025
#define COMMAND_TERMINATOR              "\r\n"
#define COMMAND_TERMINATOR_SIZE         2

#define READ_CMD_ERROR                  1
#define MALLOC_ERROR                    2
#define ERROR_READING_EXTRA_RESPONSE    3
#define SENDING_COMMAND_ERROR           4
#define READING_RESPONSE_ERROR          5
#define ERROR_READING_INITIAL_RESPONSE  6
#define INVALID_RESPONSE                7
#define SOCKET_CREATE_ERROR             -1
#define SOCKET_CONNECT_ERROR            -2


int read_command_reply(int socketfd, unsigned short* response_code, char** response_str,
size_t * response_str_size);

int read_initial_command_reply(int socketfd, unsigned short* response_code, char**
response_str, size_t * response_str_size);

int send_command(int socketfd, const char* command);

#endif  //_COMMANDS_H_

/*************************************
    commands.c
*************************************/

#include "commands.h"
#include "connection.h"
#include "file.h"
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <netdb.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int connect_to_ip(const char * ip, unsigned port) {
    struct sockaddr_in server_addr;

       memset(&server_addr, 0, sizeof(server_addr));

    server_addr.sin_family = AF_INET;
    // 32 bit Internet address network byte ordered
       server_addr.sin_addr.s_addr = inet_addr(ip);
       server_addr.sin_port = htons(port);

    int socket_fd = socket(AF_INET, SOCK_STREAM, 0);

    if (socket_fd < 0) {
        fprintf(stderr, "Error creating socket\n");
        return SOCKET_CREATE_ERROR;
    }

    // Open connection to the server
    if (connect(socket_fd, (struct sockaddr *) &server_addr, sizeof(server_addr)) < 0) {
        fprintf(stderr, "Error connecting to the given ip\n");
               return SOCKET_CONNECT_ERROR;
    }

    return socket_fd;
}

int read_command_reply(int socketfd, unsigned short* response_code, char** response_str,
size_t * response_str_size) {
    *response_str = calloc(RESPONSE_MAX_SIZE, sizeof(**response_str));

    if (*response_str == NULL) {
        fprintf(stderr, "Could not allocate buffer\n");
        return MALLOC_ERROR;
    }

    *response_str[0] = '\0';

    int socketfd_dup = dup(socketfd);

    if (socketfd_dup == -1) {
        return READ_CMD_ERROR;
    }

    FILE* socket_fileptr = fdopen(socketfd_dup, "r");

    if (socket_fileptr == NULL) {
        close(socketfd_dup);
        return READ_CMD_ERROR;
    }

    char* buf = NULL;
    size_t num_bytes = 0;
    *response_str_size = 0;
    while ((num_bytes = getline(&buf, &num_bytes, socket_fileptr)) >= 0) {
        strncat(*response_str, buf, num_bytes);
        response_str_size += num_bytes;

        // Last line in multi line responses have a space character after the code
        if (buf[CODE_SIZE] == ' ') {
            break;
        }
    }

    free(buf);
    fclose(socket_fileptr);

    if (num_bytes < 0) {
        fprintf(stderr, "Error reading command reply!\n");
        return READ_CMD_ERROR;
    }

    if (*response_str_size == 0) {
        fprintf(stderr, "No extra response was received\n");
    }
    char* code_str = strndup(*response_str, 3);
    *response_code = atoi(code_str);
    free(code_str);
    (*response_str)[RESPONSE_MAX_SIZE-1] = '\0';

    return 0;
}

int send_command(int socketfd, const char* command) {
    const size_t command_len = strlen(command);

    if (write(socketfd, command, command_len) != command_len) {
        fprintf(stderr, "Error sending command!\n");
        return SENDING_COMMAND_ERROR;
    }

            if    (write(socketfd,    COMMAND_TERMINATOR,    COMMAND_TERMINATOR_SIZE)    !=
COMMAND_TERMINATOR_SIZE) {
        fprintf(stderr, "Error sending command terminator!\n");
        return SENDING_COMMAND_ERROR;
    }

    return 0;
}

/**************************************
    file.h
**************************************/

#ifndef _FILE_H_
#define _FILE_H_

#define BUF_SIZE                256
#define FILE_TRANSFER_FAILED    1
#define FILE_PERMISSIONS        0644

int copy_file(int fd, const char* file_name);

#endif  // _FILE_H_

/**************************************
    file.c
**************************************/

#include "file.h"
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int copy_file(int origin_fd, const char* file_name) {
    int destin_fd = open(file_name, O_CREAT | O_WRONLY, FILE_PERMISSIONS);

    if (destin_fd == -1) {
        return FILE_TRANSFER_FAILED;
    }

    ssize_t num_read_bytes;
    ssize_t num_written_bytes;
    char buf[BUF_SIZE];
    while((num_read_bytes = read(origin_fd, buf, BUF_SIZE)) > 0) {
        num_written_bytes = write(destin_fd, buf, num_read_bytes);

        if (num_written_bytes < num_read_bytes) {
            close(destin_fd);
            return FILE_TRANSFER_FAILED;
        }
    }

    close(destin_fd);

    if (num_read_bytes == -1) {
        return FILE_TRANSFER_FAILED;
    }

    return 0;
}
```

Attachment B.1 - Switch Configuration Commands

```
// Tux Y1 eth0
configure terminal
interface fastEthernet 0/14
switchport mode access
switchport access vlan 40
end

// Tux Y4 eth0
configure terminal
interface fastEthernet 0/16
switchport mode access
switchport access vlan 40
end

// Tux Y2 eth0
configure terminal
interface fastEthernet 0/13
switchport mode access
switchport access vlan 41
end

// Tux Y4 eth1
configure terminal
interface fastEthernet 0/15
switchport mode access
switchport access vlan 41
end

// Cisco Router (Rc)
configure terminal
interface gigabitEthernet 0/1
switchport mode access
switchport access vlan 41
end
```

Attachment B.2 - Tux Configuration Commands

tux Y1

```
// Configuring eth0
ifconfig eth0 172.16.Y0.1/24

// Adding a default gateway to tux Y4
route add default gw 172.16.Y0.254

// Configuring DNS
echo -e "search netlab.fe.up.pt\nnameserver 172.16.1.1" >
/etc/resolv.conf
```

tux Y2

```
// Configuring eth0
ifconfig eth0 172.16.Y1.1/24

// Adding a default gateway to the CISCO router
route add default gw 172.16.Y1.254

// Configuring DNS
echo -e "search netlab.fe.up.pt\nnameserver 172.16.1.1" >
/etc/resolv.conf
```

tux Y4

```
// Configuring eth0 and eth1
ifconfig eth0 172.16.Y0.254/24
ifconfig eth1 172.16.Y1.253/24

// Adding a default gateway to the CISCO router
route add default gw 172.16.Y1.254

// Enabling IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward

// Configuring DNS
echo -e "search netlab.fe.up.pt\nnameserver 172.16.1.1" >
/etc/resolv.conf
```

Attachment B.3 - Router Configuration Commands

```
// Configuring NAT inside
conf t
interface gigabitethernet 0/0
ip address 172.16.41.254 255.255.255.0
no shutdown
ip nat inside
exit

// Configuring NAT outside
interface gigabitethernet 0/1
ip address 172.16.1.49 255.255.255.0
no shutdown
ip nat outside
exit

// Configuring nat properties
ip nat pool ovrld 172.16.1.49 172.16.1.49 prefix 24
ip nat inside source list 1 pool ovrld overload

// Declaring the valid access list
access-list 1 permit 172.16.40.0 0.0.0.7
access-list 1 permit 172.16.41.0 0.0.0.7

// Configuring router IP routing
ip route 0.0.0.0 0.0.0.0 172.16.1.254
ip route 172.16.40.0 255.255.255.0 172.16.41.253
end
```

Attachment C - Computer Network experiments captured logs

Network IP and MAC addresses:

Machine	IP Address	MAC Address
tux41 eth0	172.16.40.1	00:0F:FE:8C:AF:AF
tux42 eth0	172.16.41.1	00:1f:29:d7:45:c4
tux44 eth0	172.16.40.254	00:21:5A:5A:7B:EA
tux44 eth1	172.16.41.253	00:C0:DF:25:1A:F4
Cisco Router	172.16.41.254	68:ef:bd:e3:df:10

Attachment C.1. - Experiment 1: Pinging tux44 from tux41

Capturing Machine: tux41

```
No.     Time          Source             Destination           Protocol Length Info
      1 0.000000     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP     60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
      2 2.009677     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP     60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
      3 2.029086     G-ProCom_8c:af:af   Broadcast            ARP      42     Who has 172.16.40.254? Tell 172.16.40.1
      4 2.029291     HewlettP_5a:7b:ea   G-ProCom_8c:af:af    ARP      60     172.16.40.254 is at 00:21:5a:5a:7b:ea
      5 2.029301     172.16.40.1         172.16.40.254        ICMP     98     Echo (ping) request  id=0x1226, seq=1/256, ttl=64 (reply in 6)
      6 2.029552     172.16.40.254       172.16.40.1          ICMP     98     Echo (ping) reply    id=0x1226, seq=1/256, ttl=64 (request in 5)
      7 3.029060     172.16.40.1         172.16.40.254        ICMP     98     Echo (ping) request  id=0x1226, seq=2/512, ttl=64 (reply in 8)
      8 3.029324     172.16.40.254       172.16.40.1          ICMP     98     Echo (ping) reply    id=0x1226, seq=2/512, ttl=64 (request in 7)
      9 3.822298     Cisco_d4:1c:10      Cisco_d4:1c:10       LOOP     60     Reply
     10 4.009552     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP     60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
     11 4.029059     172.16.40.1         172.16.40.254        ICMP     98     Echo (ping) request  id=0x1226, seq=3/768, ttl=64 (reply in 12)
     12 4.029313     172.16.40.254       172.16.40.1          ICMP     98     Echo (ping) reply    id=0x1226, seq=3/768, ttl=64 (request in 11)
     13 5.029047     172.16.40.1         172.16.40.254        ICMP     98     Echo (ping) request  id=0x1226, seq=4/1024, ttl=64 (reply in 14)
     14 5.029280     172.16.40.254       172.16.40.1          ICMP     98     Echo (ping) reply    id=0x1226, seq=4/1024, ttl=64 (request in 13)
     15 6.014486     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP     60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
     16 6.029053     172.16.40.1         172.16.40.254        ICMP     98     Echo (ping) request  id=0x1226, seq=5/1280, ttl=64 (reply in 17)
     17 6.029305     172.16.40.254       172.16.40.1          ICMP     98     Echo (ping) reply    id=0x1226, seq=5/1280, ttl=64 (request in 16)
     18 7.031019     HewlettP_5a:7b:ea   G-ProCom_8c:af:af    ARP      60     Who has 172.16.40.1? Tell 172.16.40.254
     19 7.031042     G-ProCom_8c:af:af   HewlettP_5a:7b:ea    ARP      42     172.16.40.1 is at 00:0f:fe:8c:af:af
```

Attachment C.2.1. - Experiment 2, step 4: Pinging tux44 and tux42 from tux41

Capturing Machine: tux41

```
No.     Time          Source             Destination           Protocol Length Info
      1 0.000000     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP     60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
      2 2.009691     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP     60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
      3 3.591457     172.16.40.1         172.16.40.254        ICMP     98     Echo (ping) request  id=0x13ac, seq=1/256, ttl=64 (reply in 4)
      4 3.591604     172.16.40.254       172.16.40.1          ICMP     98     Echo (ping) reply    id=0x13ac, seq=1/256, ttl=64 (request in 3)
      5 4.009462     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP     60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
      6 4.591040     172.16.40.1         172.16.40.254        ICMP     98     Echo (ping) request  id=0x13ac, seq=2/512, ttl=64 (reply in 7)
      7 4.591236     172.16.40.254       172.16.40.1          ICMP     98     Echo (ping) reply    id=0x13ac, seq=2/512, ttl=64 (request in 6)
      8 5.591022     172.16.40.1         172.16.40.254        ICMP     98     Echo (ping) request  id=0x13ac, seq=3/768, ttl=64 (reply in 9)
      9 5.591254     172.16.40.254       172.16.40.1          ICMP     98     Echo (ping) reply    id=0x13ac, seq=3/768, ttl=64 (request in 8)
```

```
  10 6.014229      Cisco_d4:1c:10        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  11 6.591039      172.16.40.1         172.16.40.254        ICMP     98     Echo (ping) request  id=0x13ac, seq=4/1024, ttl=64 (reply in 12)
  12 6.591233      172.16.40.254         172.16.40.1          ICMP     98     Echo (ping) reply    id=0x13ac, seq=4/1024, ttl=64 (request in 11)
  13 7.591041      172.16.40.1         172.16.40.254        ICMP     98     Echo (ping) request  id=0x13ac, seq=5/1280, ttl=64 (reply in 14)
  14 7.591275      172.16.40.254         172.16.40.1          ICMP     98     Echo (ping) reply    id=0x13ac, seq=5/1280, ttl=64 (request in 13)
  15 8.019043      Cisco_d4:1c:10        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  16 8.539709      Cisco_d4:1c:10        Cisco_d4:1c:10        LOOP     60     Reply
  17 8.592960      HewlettP_5a:7b:ea     G-ProCom_8c:af:af     ARP      60     Who has 172.16.40.1? Tell 172.16.40.254
  18 8.592989      G-ProCom_8c:af:af     HewlettP_5a:7b:ea     ARP      42     172.16.40.1 is at 00:0f:fe:8c:af:af
  19 9.911250      172.16.40.1         172.16.41.1        ICMP     98     Echo (ping) request  id=0x13b0, seq=1/256, ttl=64 (no response found!)
  20 10.023905     Cisco_d4:1c:10        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  21 10.911022     172.16.40.1         172.16.41.1        ICMP     98     Echo (ping) request  id=0x13b0, seq=2/512, ttl=64 (no response found!)
  22 11.911025     172.16.40.1         172.16.41.1        ICMP     98     Echo (ping) request  id=0x13b0, seq=3/768, ttl=64 (no response found!)
  23 12.028668     Cisco_d4:1c:10        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  24 12.908676     172.16.40.254         172.16.40.1          ICMP     126    Destination unreachable (Host unreachable)
  25 12.908707     172.16.40.254         172.16.40.1          ICMP     126    Destination unreachable (Host unreachable)
  26 12.908717     172.16.40.254         172.16.40.1          ICMP     126    Destination unreachable (Host unreachable)
  27 12.910027     172.16.40.1         172.16.41.1        ICMP     98     Echo (ping) request  id=0x13b0, seq=4/1024, ttl=64 (no response found!)
  28 13.917091     172.16.40.1         172.16.41.1        ICMP     98     Echo (ping) request  id=0x13b0, seq=5/1280, ttl=64 (no response found!)
  29 14.033518     Cisco_d4:1c:10        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  30 14.925129     172.16.40.1         172.16.41.1        ICMP     98     Echo (ping) request  id=0x13b0, seq=6/1536, ttl=64 (no response found!)
  31 14.926983     G-ProCom_8c:af:af     HewlettP_5a:7b:ea     ARP      42     Who has 172.16.40.254? Tell 172.16.40.1
  32 14.927232     HewlettP_5a:7b:ea     G-ProCom_8c:af:af     ARP      60     172.16.40.254 is at 00:21:5a:5a:7b:ea
  33 15.908750     172.16.40.254         172.16.40.1          ICMP     126    Destination unreachable (Host unreachable)
  34 15.908771     172.16.40.254         172.16.40.1          ICMP     126    Destination unreachable (Host unreachable)
  35 15.908776     172.16.40.254         172.16.40.1          ICMP     126    Destination unreachable (Host unreachable)
  36 16.038357     Cisco_d4:1c:10        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  37 18.043236     Cisco_d4:1c:10        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  38 18.543891     Cisco_d4:1c:10        Cisco_d4:1c:10        LOOP     60     Reply
```

Attachment C.2.2. - Experiment 2, step 7: Pinging vlan 40 broadcast channel from tux41

Capturing Machine: tux41

```
No.     Time           Source                Destination           Protocol Length Info
   1 0.000000      Cisco_d4:1c:10        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
   2 2.004769      Cisco_d4:1c:10        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
   3 4.009618      Cisco_d4:1c:10        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
   4 5.989754      Cisco_d4:1c:10        Cisco_d4:1c:10        LOOP     60     Reply
   5 6.019809      Cisco_d4:1c:10        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
   6 8.019261      Cisco_d4:1c:10        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
   7 10.027829     Cisco_d4:1c:10        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
   8 11.270529     172.16.40.1         172.16.40.255        ICMP     98     Echo (ping) request  id=0x1473, seq=1/256, ttl=64 (no response found!)
   9 11.270791     172.16.40.254         172.16.40.1          ICMP     98     Echo (ping) reply    id=0x1473, seq=1/256, ttl=64
  10 12.028938     Cisco_d4:1c:10        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  11 12.269670     172.16.40.1         172.16.40.255        ICMP     98     Echo (ping) request  id=0x1473, seq=2/512, ttl=64 (no response found!)
  12 12.269818     172.16.40.254         172.16.40.1          ICMP     98     Echo (ping) reply    id=0x1473, seq=2/512, ttl=64
  13 13.269677     172.16.40.1         172.16.40.255        ICMP     98     Echo (ping) request  id=0x1473, seq=3/768, ttl=64 (no response found!)
  14 13.269936     172.16.40.254         172.16.40.1          ICMP     98     Echo (ping) reply    id=0x1473, seq=3/768, ttl=64
  15 14.033736     Cisco_d4:1c:10        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  16 14.269683     172.16.40.1         172.16.40.255        ICMP     98     Echo (ping) request  id=0x1473, seq=4/1024, ttl=64 (no response found!)
  17 14.269830     172.16.40.254         172.16.40.1          ICMP     98     Echo (ping) reply    id=0x1473, seq=4/1024, ttl=64
  18 15.269700     172.16.40.1         172.16.40.255        ICMP     98     Echo (ping) request  id=0x1473, seq=5/1280, ttl=64 (no response found!)
  19 15.269959     172.16.40.254         172.16.40.1          ICMP     98     Echo (ping) reply    id=0x1473, seq=5/1280, ttl=64
  20 16.002159     Cisco_d4:1c:10        Cisco_d4:1c:10        LOOP     60     Reply
  21 16.043876     Cisco_d4:1c:10        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  22 16.285147     HewlettP_5a:7b:ea     G-ProCom_8c:af:af     ARP      60     Who has 172.16.40.1? Tell 172.16.40.254
  23 16.285170     G-ProCom_8c:af:af     HewlettP_5a:7b:ea     ARP      42     172.16.40.1 is at 00:0f:fe:8c:af:af
```

Capturing Machine: tux42

```
No.     Time           Source                Destination           Protocol Length Info
   1 0.000000      Cisco_d4:1c:11        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8011
   2 0.570617      Cisco_d4:1c:11        CDP/VTP/DTP/PAgP/UDLD CDP      436    Device ID: tux-sw4  Port ID: FastEthernet0/15
   3 1.980101      Cisco_d4:1c:11        Cisco_d4:1c:11        LOOP     60     Reply
   4 2.010066      Cisco_d4:1c:11        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8011
   5 4.009761      Cisco_d4:1c:11        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8011
   6 6.018418      Cisco_d4:1c:11        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8011
   7 8.019565      Cisco_d4:1c:11        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8011
   8 10.024418     Cisco_d4:1c:11        Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8011
```

```
  9 11.992813     Cisco_d4:1c:11      Cisco_d4:1c:11      LOOP    60    Reply
 10 12.034510     Cisco_d4:1c:11      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8011
 11 14.034196     Cisco_d4:1c:11      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8011
 12 16.041680     Cisco_d4:1c:11      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8011
 13 18.049060     Cisco_d4:1c:11      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8011
```

Capturing Machine: tux44

```
No.     Time           Source              Destination          Protocol Length Info
  1 0.000000      Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
  2 2.008543      Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
  3 3.251097      172.16.40.1         172.16.40.255       ICMP     98    Echo (ping) request  id=0x1473, seq=1/256, ttl=64 (no response found!)
  4 3.251130      172.16.40.254       172.16.40.1         ICMP     98    Echo (ping) reply    id=0x1473, seq=1/256, ttl=64
  5 4.009922      Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
  6 4.250269      172.16.40.1         172.16.40.255       ICMP     98    Echo (ping) request  id=0x1473, seq=2/512, ttl=64 (no response found!)
  7 4.250296      172.16.40.254       172.16.40.1         ICMP     98    Echo (ping) reply    id=0x1473, seq=2/512, ttl=64
  8 5.250324      172.16.40.1         172.16.40.255       ICMP     98    Echo (ping) request  id=0x1473, seq=3/768, ttl=64 (no response found!)
  9 5.250351      172.16.40.254       172.16.40.1         ICMP     98    Echo (ping) reply    id=0x1473, seq=3/768, ttl=64
 10 6.014606      Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
 11 6.250355      172.16.40.1         172.16.40.255       ICMP     98    Echo (ping) request  id=0x1473, seq=4/1024, ttl=64 (no response found!)
 12 6.250381      172.16.40.254       172.16.40.1         ICMP     98    Echo (ping) reply    id=0x1473, seq=4/1024, ttl=64
 13 7.250417      172.16.40.1         172.16.40.255       ICMP     98    Echo (ping) request  id=0x1473, seq=5/1280, ttl=64 (no response found!)
 14 7.250443      172.16.40.254       172.16.40.1         ICMP     98    Echo (ping) reply    id=0x1473, seq=5/1280, ttl=64
 15 7.982976      Cisco_d4:1c:12      Cisco_d4:1c:12      LOOP    60    Reply
 16 8.024761      Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
 17 8.265655      HewlettP_5a:7b:ea   G-ProCom_8c:af:af   ARP     42    Who has 172.16.40.1? Tell 172.16.40.254
 18 8.265906      G-ProCom_8c:af:af   HewlettP_5a:7b:ea   ARP     60    172.16.40.1 is at 00:0f:fe:8c:af:af
 19 10.024405     Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
 20 12.032029     Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
 21 14.039593     Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
 22 16.039184     Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
 23 17.977244     Cisco_d4:1c:12      Cisco_d4:1c:12      LOOP    60    Reply
 24 18.044344     Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
```

Attachment C.2.3. - Experiment 2, step 10: Pinging vlan 41 broadcast channel from tux42

Capturing Machine: tux41

```
No.     Time           Source              Destination          Protocol Length Info
  1 0.000000      Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  2 2.004628      Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  3 4.009741      Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  4 6.014483      Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  5 7.549738      Cisco_d4:1c:10      Cisco_d4:1c:10      LOOP    60    Reply
  6 8.019277      Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  7 10.024131     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  8 12.028844     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  9 14.033592     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
 10 16.038420     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
 11 17.557106     Cisco_d4:1c:10      Cisco_d4:1c:10      LOOP    60    Reply
 12 18.043431     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
 13 18.714764     Cisco_d4:1c:10      CDP/VTP/DTP/PAgP/UDLD CDP      436    Device ID: tux-sw4  Port ID: FastEthernet0/14
 14 20.048084     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
```

Capturing Machine: tux42

```
No.     Time           Source              Destination          Protocol Length Info
  1 0.000000      Cisco_d4:1c:11      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8011
  2 1.170120      172.16.41.1         172.16.41.255       ICMP     98    Echo (ping) request  id=0x106c, seq=1/256, ttl=64 (no response found!)
  3 2.004896      Cisco_d4:1c:11      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8011
  4 2.169661      172.16.41.1         172.16.41.255       ICMP     98    Echo (ping) request  id=0x106c, seq=2/512, ttl=64 (no response found!)
  5 3.169657      172.16.41.1         172.16.41.255       ICMP     98    Echo (ping) request  id=0x106c, seq=3/768, ttl=64 (no response found!)
  6 4.009892      Cisco_d4:1c:11      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8011
  7 4.169659      172.16.41.1         172.16.41.255       ICMP     98    Echo (ping) request  id=0x106c, seq=4/1024, ttl=64 (no response found!)
  8 5.169659      172.16.41.1         172.16.41.255       ICMP     98    Echo (ping) request  id=0x106c, seq=5/1280, ttl=64 (no response found!)
  9 6.014679      Cisco_d4:1c:11      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8011
 10 6.169659      172.16.41.1         172.16.41.255       ICMP     98    Echo (ping) request  id=0x106c, seq=6/1536, ttl=64 (no response found!)
 11 8.019565      Cisco_d4:1c:11      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8011
```

Capturing Machine: tux44

```
No.     Time           Source              Destination          Protocol Length Info
  1 0.000000      Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
  2 2.004536      Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
```

```
   3 3.539926      Cisco_d4:1c:12       Cisco_d4:1c:12       LOOP    60     Reply
   4 4.009342      Cisco_d4:1c:12       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
   5 6.014249      Cisco_d4:1c:12       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
   6 8.019257      Cisco_d4:1c:12       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
   7 10.024320      Cisco_d4:1c:12       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
   8 12.028955      Cisco_d4:1c:12       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
   9 13.547664      Cisco_d4:1c:12       Cisco_d4:1c:12       LOOP    60     Reply
  10 14.033943      Cisco_d4:1c:12       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
  11 16.038844      Cisco_d4:1c:12       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
  12 18.044203      Cisco_d4:1c:12       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
  13 20.048549      Cisco_d4:1c:12       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
```

Attachment C.3.1. - Experiment 3, step 5: Pinging all interfaces from tux 41

Capturing Machine: tux41

```
No.     Time           Source                Destination           Protocol Length Info
   1 0.000000      Cisco_d4:1c:10       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
   2 0.011437      172.16.40.1          172.16.40.254         ICMP     98      Echo (ping) request  id=0x16d6, seq=1/256, ttl=64 (reply in 3)
   3 0.011697      172.16.40.254        172.16.40.1           ICMP     98      Echo (ping) reply    id=0x16d6, seq=1/256, ttl=64 (request in 2)
   4 1.010575      172.16.40.1          172.16.40.254         ICMP     98      Echo (ping) request  id=0x16d6, seq=2/512, ttl=64 (reply in 5)
   5 1.010780      172.16.40.254        172.16.40.1           ICMP     98      Echo (ping) reply    id=0x16d6, seq=2/512, ttl=64 (request in 4)
   6 1.999814      Cisco_d4:1c:10       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
   7 2.010573      172.16.40.1          172.16.40.254         ICMP     98      Echo (ping) request  id=0x16d6, seq=3/768, ttl=64 (reply in 8)
   8 2.010823      172.16.40.254        172.16.40.1           ICMP     98      Echo (ping) reply    id=0x16d6, seq=3/768, ttl=64 (request in 7)
   9 3.010573      172.16.40.1          172.16.40.254         ICMP     98      Echo (ping) request  id=0x16d6, seq=4/1024, ttl=64 (reply in 10)
  10 3.010779      172.16.40.254        172.16.40.1           ICMP     98      Echo (ping) reply    id=0x16d6, seq=4/1024, ttl=64 (request in 9)
  11 4.004666      Cisco_d4:1c:10       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  12 4.010575      172.16.40.1          172.16.40.254         ICMP     98      Echo (ping) request  id=0x16d6, seq=5/1280, ttl=64 (reply in 13)
  13 4.010825      172.16.40.254        172.16.40.1           ICMP     98      Echo (ping) reply    id=0x16d6, seq=5/1280, ttl=64 (request in 12)
  14 5.019061      HewlettP_5a:7b:ea    G-ProCom_8c:af:af     ARP      60      Who has 172.16.40.1? Tell 172.16.40.254
  15 5.019090      G-ProCom_8c:af:af    HewlettP_5a:7b:ea     ARP      42      172.16.40.1 is at 00:0f:fe:8c:af:af
  16 6.014484      Cisco_d4:1c:10       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  17 7.699016      172.16.40.1          172.16.41.253         ICMP     98      Echo (ping) request  id=0x16dd, seq=1/256, ttl=64 (reply in 18)
  18 7.699381      172.16.41.253        172.16.40.1           ICMP     98      Echo (ping) reply    id=0x16dd, seq=1/256, ttl=64 (request in 17)
  19 8.014181      Cisco_d4:1c:10       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  20 8.120953      Cisco_d4:1c:10       Cisco_d4:1c:10       LOOP    60     Reply
  21 8.698572      172.16.40.1          172.16.41.253         ICMP     98      Echo (ping) request  id=0x16dd, seq=2/512, ttl=64 (reply in 22)
  22 8.698782      172.16.41.253        172.16.40.1           ICMP     98      Echo (ping) reply    id=0x16dd, seq=2/512, ttl=64 (request in 21)
  23 9.698574      172.16.40.1          172.16.41.253         ICMP     98      Echo (ping) request  id=0x16dd, seq=3/768, ttl=64 (reply in 24)
  24 9.698923      172.16.41.253        172.16.40.1           ICMP     98      Echo (ping) reply    id=0x16dd, seq=3/768, ttl=64 (request in 23)
  25 10.019083      Cisco_d4:1c:10       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  26 10.698579      172.16.40.1          172.16.41.253         ICMP     98      Echo (ping) request  id=0x16dd, seq=4/1024, ttl=64 (reply in 27)
  27 10.698719      172.16.41.253        172.16.40.1           ICMP     98      Echo (ping) reply    id=0x16dd, seq=4/1024, ttl=64 (request in 26)
  28 11.698578      172.16.40.1          172.16.41.253         ICMP     98      Echo (ping) request  id=0x16dd, seq=5/1280, ttl=64 (reply in 29)
  29 11.698921      172.16.41.253        172.16.40.1           ICMP     98      Echo (ping) reply    id=0x16dd, seq=5/1280, ttl=64 (request in 28)
  30 12.028812      Cisco_d4:1c:10       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  31 14.028702      Cisco_d4:1c:10       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  32 14.634898      172.16.40.1          172.16.41.1           ICMP     98      Echo (ping) request  id=0x16e1, seq=1/256, ttl=64 (reply in 33)
  33 14.635397      172.16.41.1          172.16.40.1           ICMP     98      Echo (ping) reply    id=0x16e1, seq=1/256, ttl=63 (request in 32)
  34 15.634584      172.16.40.1          172.16.41.1           ICMP     98      Echo (ping) request  id=0x16e1, seq=2/512, ttl=64 (reply in 35)
  35 15.635057      172.16.41.1          172.16.40.1           ICMP     98      Echo (ping) reply    id=0x16e1, seq=2/512, ttl=63 (request in 34)
  36 16.033401      Cisco_d4:1c:10       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  37 16.634583      172.16.40.1          172.16.41.1           ICMP     98      Echo (ping) request  id=0x16e1, seq=3/768, ttl=64 (reply in 38)
  38 16.635038      172.16.41.1          172.16.40.1           ICMP     98      Echo (ping) reply    id=0x16e1, seq=3/768, ttl=63 (request in 37)
  39 17.634575      172.16.40.1          172.16.41.1           ICMP     98      Echo (ping) request  id=0x16e1, seq=4/1024, ttl=64 (reply in 40)
  40 17.635057      172.16.41.1          172.16.40.1           ICMP     98      Echo (ping) reply    id=0x16e1, seq=4/1024, ttl=63 (request in 39)
  41 18.043306      Cisco_d4:1c:10       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  42 18.127999      Cisco_d4:1c:10       Cisco_d4:1c:10       LOOP    60     Reply
  43 18.634585      172.16.40.1          172.16.41.1           ICMP     98      Echo (ping) request  id=0x16e1, seq=5/1280, ttl=64 (reply in 44)
  44 18.634837      172.16.41.1          172.16.40.1           ICMP     98      Echo (ping) reply    id=0x16e1, seq=5/1280, ttl=63 (request in 43)
  45 20.043188      Cisco_d4:1c:10       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  46 22.047983      Cisco_d4:1c:10       Spanning-tree-(for-bridges)_00 STP      60      Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
```

Attachment C.3.2. - Experiment 3, step 8: Pinging tux 42 from tux 41

Capturing Machine: tux44, eth0 interface

```
No.     Time           Source               Destination          Protocol Length Info
      1 0.000000    Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
      2 1.999168    Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
      3 4.004056    Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
      4 6.014166    Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
      5 8.013918    Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
      6 8.056214    Cisco_d4:1c:12      Cisco_d4:1c:12       LOOP     60    Reply
      7 10.019022     Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
      8 12.023849     Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
      9 14.028828     Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
     10 16.033640     Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
     11 16.310316     Cisco_d4:1c:12      CDP/VTP/DTP/PAgP/UDLD CDP      436   Device ID: tux-sw4  Port ID: FastEthernet0/16
     12 18.038461     Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
     13 18.063969     Cisco_d4:1c:12      Cisco_d4:1c:12       LOOP     60    Reply
     14 20.043324     Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
     15 21.039620     G-ProCom_8c:af:af   Broadcast            ARP      60    Who has 172.16.40.254? Tell 172.16.40.1
     16 21.039642     HewlettP_5a:7b:ea   G-ProCom_8c:af:af    ARP      42    172.16.40.254 is at 00:21:5a:5a:7b:ea
     17 21.039978     172.16.40.1         172.16.41.1          ICMP     98    Echo (ping) request  id=0x199a, seq=1/256, ttl=64 (reply in 18)
     18 21.040251     172.16.41.1         172.16.40.1          ICMP     98    Echo (ping) reply    id=0x199a, seq=1/256, ttl=63 (request in 17)
     19 22.040733     172.16.40.1         172.16.41.1          ICMP     98    Echo (ping) request  id=0x199a, seq=2/512, ttl=64 (reply in 20)
     20 22.040882     172.16.41.1         172.16.40.1          ICMP     98    Echo (ping) reply    id=0x199a, seq=2/512, ttl=63 (request in 19)
     21 22.048167     Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
     22 23.039770     172.16.40.1         172.16.41.1          ICMP     98    Echo (ping) request  id=0x199a, seq=3/768, ttl=64 (reply in 23)
     23 23.039919     172.16.41.1         172.16.40.1          ICMP     98    Echo (ping) reply    id=0x199a, seq=3/768, ttl=63 (request in 22)
     24 24.038803     172.16.40.1         172.16.41.1          ICMP     98    Echo (ping) request  id=0x199a, seq=4/1024, ttl=64 (reply in 25)
     25 24.038942     172.16.41.1         172.16.40.1          ICMP     98    Echo (ping) reply    id=0x199a, seq=4/1024, ttl=63 (request in 24)
     26 24.053067     Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
     27 25.038781     172.16.40.1         172.16.41.1          ICMP     98    Echo (ping) request  id=0x199a, seq=5/1280, ttl=64 (reply in 28)
     28 25.038960     172.16.41.1         172.16.40.1          ICMP     98    Echo (ping) reply    id=0x199a, seq=5/1280, ttl=63 (request in 27)
     29 26.053818     HewlettP_5a:7b:ea   G-ProCom_8c:af:af    ARP      42    Who has 172.16.40.1? Tell 172.16.40.254
     30 26.054116     G-ProCom_8c:af:af   HewlettP_5a:7b:ea    ARP      60    172.16.40.1 is at 00:0f:fe:8c:af:af
     31 26.058264     Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
     32 28.062936     Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
     33 28.071680     Cisco_d4:1c:12      Cisco_d4:1c:12       LOOP     60    Reply
     34 30.067774     Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
     35 32.072718     Cisco_d4:1c:12      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8012
```

Capturing Machine: tux44, eth1 interface

```
No.     Time           Source               Destination          Protocol Length Info
      1 0.000000    Cisco_d4:1c:0f      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x800f
      2 2.004887    Cisco_d4:1c:0f      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x800f
      3 4.014845    Cisco_d4:1c:0f      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x800f
      4 6.014748    Cisco_d4:1c:0f      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x800f
      5 6.056845    Cisco_d4:1c:0f      Cisco_d4:1c:0f       LOOP     60    Reply
      6 8.019670    Cisco_d4:1c:0f      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x800f
      7 10.024601     Cisco_d4:1c:0f      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x800f
      8 12.029419     Cisco_d4:1c:0f      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x800f
      9 14.034320     Cisco_d4:1c:0f      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x800f
     10 16.039255     Cisco_d4:1c:0f      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x800f
     11 16.064598     Cisco_d4:1c:0f      Cisco_d4:1c:0f       LOOP     60    Reply
     12 18.044160     Cisco_d4:1c:0f      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x800f
     13 19.040994     Kye_25:1a:f4        Broadcast            ARP      42    Who has 172.16.41.1? Tell 172.16.41.253
     14 19.041113     HewlettP_d7:45:c4   Kye_25:1a:f4         ARP      60    172.16.41.1 is at 00:1f:29:d7:45:c4
     15 19.041129     172.16.40.1         172.16.41.1          ICMP     98    Echo (ping) request  id=0x199a, seq=1/256, ttl=63 (reply in 16)
     16 19.041242     172.16.41.1         172.16.40.1          ICMP     98    Echo (ping) reply    id=0x199a, seq=1/256, ttl=64 (request in 15)
     17 20.041753     172.16.40.1         172.16.41.1          ICMP     98    Echo (ping) request  id=0x199a, seq=2/512, ttl=63 (reply in 18)
     18 20.041867     172.16.41.1         172.16.40.1          ICMP     98    Echo (ping) reply    id=0x199a, seq=2/512, ttl=64 (request in 17)
     19 20.049000     Cisco_d4:1c:0f      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x800f
     20 21.040789     172.16.40.1         172.16.41.1          ICMP     98    Echo (ping) request  id=0x199a, seq=3/768, ttl=63 (reply in 21)
     21 21.040904     172.16.41.1         172.16.40.1          ICMP     98    Echo (ping) reply    id=0x199a, seq=3/768, ttl=64 (request in 20)
     22 22.039816     172.16.40.1         172.16.41.1          ICMP     98    Echo (ping) request  id=0x199a, seq=4/1024, ttl=63 (reply in 23)
     23 22.039927     172.16.41.1         172.16.40.1          ICMP     98    Echo (ping) reply    id=0x199a, seq=4/1024, ttl=64 (request in 22)
     24 22.053903     Cisco_d4:1c:0f      Spanning-tree-(for-bridges)_00 STP      60    Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x800f
     25 23.039800     172.16.40.1         172.16.41.1          ICMP     98    Echo (ping) request  id=0x199a, seq=5/1280, ttl=63 (reply in 26)
     26 23.039946     172.16.41.1         172.16.40.1          ICMP     98    Echo (ping) reply    id=0x199a, seq=5/1280, ttl=64 (request in 25)
```

```
 27 24.049063    HewlettP_d7:45:c4   Kye_25:1a:f4      ARP    60    Who has 172.16.41.253? Tell 172.16.41.1
 28 24.049081    Kye_25:1a:f4        HewlettP_d7:45:c4   ARP    42    172.16.41.253 is at 00:c0:df:25:1a:f4
 29 24.059067    Cisco_d4:1c:0f      Spanning-tree-(for-bridges)_00 STP    60    Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x800f
 30 26.063709    Cisco_d4:1c:0f      Spanning-tree-(for-bridges)_00 STP    60    Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x800f
 31 26.072312    Cisco_d4:1c:0f      Cisco_d4:1c:0f    LOOP    60    Reply
 32 28.068607    Cisco_d4:1c:0f      Spanning-tree-(for-bridges)_00 STP    60    Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x800f
```

Attachment C.4.1. - Experiment 4, step 3: Pinging all interfaces from tux41

Capturing Machine: tux41

```
No.     Time           Source              Destination           Protocol Length Info
  1 0.000000     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP    60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  2 1.475111     172.16.40.1         172.16.40.254       ICMP    98    Echo (ping) request  id=0x1bc2, seq=1/256, ttl=64 (reply in 3)
  3 1.475479     172.16.40.254       172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bc2, seq=1/256, ttl=64 (request in 2)
  4 2.010084     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP    60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  5 2.474112     172.16.40.1         172.16.40.254       ICMP    98    Echo (ping) request  id=0x1bc2, seq=2/512, ttl=64 (reply in 6)
  6 2.474268     172.16.40.254       172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bc2, seq=2/512, ttl=64 (request in 5)
  7 3.473979     172.16.40.1         172.16.40.254       ICMP    98    Echo (ping) request  id=0x1bc2, seq=3/768, ttl=64 (reply in 8)
  8 3.474214     172.16.40.254       172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bc2, seq=3/768, ttl=64 (request in 7)
  9 4.009889     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP    60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
 10 4.473976     172.16.40.1         172.16.40.254       ICMP    98    Echo (ping) request  id=0x1bc2, seq=4/1024, ttl=64 (reply in 11)
 11 4.474132     172.16.40.254       172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bc2, seq=4/1024, ttl=64 (request in 10)
 12 5.473986     172.16.40.1         172.16.40.254       ICMP    98    Echo (ping) request  id=0x1bc2, seq=5/1280, ttl=64 (reply in 13)
 13 5.474225     172.16.40.254       172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bc2, seq=5/1280, ttl=64 (request in 12)
 14 6.014677     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP    60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
 15 8.024482     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP    60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
 16 8.499134     172.16.40.1         172.16.41.253       ICMP    98    Echo (ping) request  id=0x1bc6, seq=1/256, ttl=64 (reply in 17)
 17 8.499294     172.16.41.253       172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bc6, seq=1/256, ttl=64 (request in 16)
 18 8.548598     Cisco_d4:1c:10      Cisco_d4:1c:10      LOOP    60    Reply
 19 9.498142     172.16.40.1         172.16.41.253       ICMP    98    Echo (ping) request  id=0x1bc6, seq=2/512, ttl=64 (reply in 20)
 20 9.498374     172.16.41.253       172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bc6, seq=2/512, ttl=64 (request in 19)
 21 10.024515    Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP    60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
 22 10.497974    172.16.40.1         172.16.41.253       ICMP    98    Echo (ping) request  id=0x1bc6, seq=3/768, ttl=64 (reply in 23)
 23 10.498322    172.16.41.253       172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bc6, seq=3/768, ttl=64 (request in 22)
 24 11.497978    172.16.40.1         172.16.41.253       ICMP    98    Echo (ping) request  id=0x1bc6, seq=4/1024, ttl=64 (reply in 25)
 25 11.498241    172.16.41.253       172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bc6, seq=4/1024, ttl=64 (request in 24)
 26 12.029100    Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP    60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
 27 12.497978    172.16.40.1         172.16.41.253       ICMP    98    Echo (ping) request  id=0x1bc6, seq=5/1280, ttl=64 (reply in 28)
 28 12.498129    172.16.41.253       172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bc6, seq=5/1280, ttl=64 (request in 27)
 29 14.039001    Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP    60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
 30 15.819092    172.16.40.1         172.16.41.1         ICMP    98    Echo (ping) request  id=0x1bca, seq=1/256, ttl=64 (reply in 31)
 31 15.819586    172.16.41.1         172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bca, seq=1/256, ttl=63 (request in 30)
 32 16.038716    Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP    60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
 33 16.818101    172.16.40.1         172.16.41.1         ICMP    98    Echo (ping) request  id=0x1bca, seq=2/512, ttl=64 (reply in 34)
 34 16.818353    172.16.41.1         172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bca, seq=2/512, ttl=63 (request in 33)
 35 17.817989    172.16.40.1         172.16.41.1         ICMP    98    Echo (ping) request  id=0x1bca, seq=3/768, ttl=64 (reply in 36)
 36 17.818227    172.16.41.1         172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bca, seq=3/768, ttl=63 (request in 35)
 37 18.043590    Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP    60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
 38 18.555743    Cisco_d4:1c:10      Cisco_d4:1c:10      LOOP    60    Reply
 39 18.817977    172.16.40.1         172.16.41.1         ICMP    98    Echo (ping) request  id=0x1bca, seq=4/1024, ttl=64 (reply in 40)
 40 18.818422    172.16.41.1         172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bca, seq=4/1024, ttl=63 (request in 39)
 41 19.818002    172.16.40.1         172.16.41.1         ICMP    98    Echo (ping) request  id=0x1bca, seq=5/1280, ttl=64 (reply in 42)
 42 19.818242    172.16.41.1         172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bca, seq=5/1280, ttl=63 (request in 41)
 43 20.053507    Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP    60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
 44 22.053123    Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP    60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
 45 22.531276    172.16.40.1         172.16.41.254       ICMP    98    Echo (ping) request  id=0x1bd1, seq=1/256, ttl=64 (reply in 46)
 46 22.532015    172.16.41.254       172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bd1, seq=1/256, ttl=254 (request in 45)
 47 23.530281    172.16.40.1         172.16.41.254       ICMP    98    Echo (ping) request  id=0x1bd1, seq=2/512, ttl=64 (reply in 48)
 48 23.530933    172.16.41.254       172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bd1, seq=2/512, ttl=254 (request in 47)
 49 24.063223    Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP    60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
 50 24.529995    172.16.40.1         172.16.41.254       ICMP    98    Echo (ping) request  id=0x1bd1, seq=3/768, ttl=64 (reply in 51)
 51 24.530624    172.16.41.254       172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bd1, seq=3/768, ttl=254 (request in 50)
 52 25.529983    172.16.40.1         172.16.41.254       ICMP    98    Echo (ping) request  id=0x1bd1, seq=4/1024, ttl=64 (reply in 53)
 53 25.530749    172.16.41.254       172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bd1, seq=4/1024, ttl=254 (request in 52)
 54 26.062903    Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP    60    Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
 55 26.529986    172.16.40.1         172.16.41.254       ICMP    98    Echo (ping) request  id=0x1bd1, seq=5/1280, ttl=64 (reply in 56)
 56 26.530635    172.16.41.254       172.16.40.1         ICMP    98    Echo (ping) reply    id=0x1bd1, seq=5/1280, ttl=254 (request in 55)
```

```
  57 28.067787     Cisco_d4:1c:10       Spanning-tree-(for-bridges)_00 STP     60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
  58 28.540279     HewlettP_5a:7b:ea     G-ProCom_8c:af:af     ARP     60     Who has 172.16.40.1? Tell 172.16.40.254
  59 28.540302     G-ProCom_8c:af:af     HewlettP_5a:7b:ea     ARP     42     172.16.40.1 is at 00:0f:fe:8c:af:af
  60 28.568305     Cisco_d4:1c:10       Cisco_d4:1c:10       LOOP     60     Reply
  61 30.072593     Cisco_d4:1c:10       Spanning-tree-(for-bridges)_00 STP     60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
```

Attachment C.4.2. - Experiment 4, step 4: Pinging tux41 from tux42

Capturing Machine: tux42

```
No.     Time          Source             Destination          Protocol Length Info
     1 0.000000     Cisco_d4:1c:11       Spanning-tree-(for-bridges)_00 STP     60     Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8011
     2 0.394515     Cisco_d4:1c:11       Cisco_d4:1c:11       LOOP     60     Reply
     3 0.848769     172.16.41.1          172.16.40.1          ICMP     98     Echo (ping) request  id=0x1995, seq=1/256, ttl=64 (reply in 5)
     4 0.849163     172.16.41.254        172.16.41.1          ICMP     70     Redirect             (Redirect for host)
     5 0.849458     172.16.40.1          172.16.41.1          ICMP     98     Echo (ping) reply    id=0x1995, seq=1/256, ttl=63 (request in 3)
     6 1.847778     172.16.41.1          172.16.40.1          ICMP     98     Echo (ping) request  id=0x1995, seq=2/512, ttl=64 (reply in 8)
     7 1.848106     172.16.41.254        172.16.41.1          ICMP     70     Redirect             (Redirect for host)
     8 1.848405     172.16.40.1          172.16.41.1          ICMP     98     Echo (ping) reply    id=0x1995, seq=2/512, ttl=63 (request in 6)
     9 2.004993     Cisco_d4:1c:11       Spanning-tree-(for-bridges)_00 STP     60     Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8011
    10 2.847670     172.16.41.1          172.16.40.1          ICMP     98     Echo (ping) request  id=0x1995, seq=3/768, ttl=64 (reply in 12)
    11 2.847981     172.16.41.254        172.16.41.1          ICMP     70     Redirect             (Redirect for host)
    12 2.848360     172.16.40.1          172.16.41.1          ICMP     98     Echo (ping) reply    id=0x1995, seq=3/768, ttl=63 (request in 10)
    13 3.847676     172.16.41.1          172.16.40.1          ICMP     98     Echo (ping) request  id=0x1995, seq=4/1024, ttl=64 (reply in 15)
    14 3.848012     172.16.41.254        172.16.41.1          ICMP     70     Redirect             (Redirect for host)
    15 3.848311     172.16.40.1          172.16.41.1          ICMP     98     Echo (ping) reply    id=0x1995, seq=4/1024, ttl=63 (request in 13)
    16 4.009769     Cisco_d4:1c:11       Spanning-tree-(for-bridges)_00 STP     60     Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8011
    17 4.847671     172.16.41.1          172.16.40.1          ICMP     98     Echo (ping) request  id=0x1995, seq=5/1280, ttl=64 (reply in 19)
    18 4.847996     172.16.41.254        172.16.41.1          ICMP     70     Redirect             (Redirect for host)
    19 4.848359     172.16.40.1          172.16.41.1          ICMP     98     Echo (ping) reply    id=0x1995, seq=5/1280, ttl=63 (request in 17)
    20 6.014749     Cisco_d4:1c:11       Spanning-tree-(for-bridges)_00 STP     60     Conf. Root = 32768/41/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8011
```

Attachment C.4.3. - Experiment 4, step 4: Pinging tux41 from tux42 traceroute result, without default gateway to tux44

```
traceroute to 172.16.40.1 (172.16.40.1), 30 hops max, 60 byte packets
 1  172.16.41.254 (172.16.41.254)  0.561 ms  0.647 ms  0.720 ms
 2  172.16.41.253 (172.16.41.253)  0.847 ms  0.359 ms  0.363 ms
 3  bancada4.netlab.fe.up.pt (172.16.40.1)  0.678 ms  0.672 ms  0.664 ms
```

Attachment C.4.4. - Experiment 4, step 4: Pinging tux41 from tux42 traceroute result, with default gateway to tux44

```
traceroute to 172.16.40.1 (172.16.40.1), 30 hops max, 60 byte packets
 1  172.16.41.254 (172.16.41.254)  0.527 ms  0.599 ms  0.664 ms
 2  172.16.41.253 (172.16.41.253)  0.805 ms  0.341 ms  0.340 ms
 3  bancada4.netlab.fe.up.pt (172.16.40.1)  0.540 ms  0.536 ms  0.529 ms
```

Attachment C.4.5. - Experiment 4, step 4: Pinging tux41 from tux42 traceroute result, with ICMP redirect acceptance at tux42

```
tux42:~# traceroute 172.16.40.1
traceroute to 172.16.40.1 (172.16.40.1), 30 hops max, 60 byte packets
 1  172.16.41.254 (172.16.41.254)  0.508 ms  0.556 ms  0.643 ms
 2  172.16.41.253 (172.16.41.253)  0.792 ms  0.341 ms  0.345 ms
 3  bancada4.netlab.fe.up.pt (172.16.40.1)  0.591 ms  0.586 ms  0.577 ms

tux42:~# traceroute 172.16.40.1
traceroute to 172.16.40.1 (172.16.40.1), 30 hops max, 60 byte packets
 1  172.16.41.253 (172.16.41.253)  0.159 ms  0.150 ms  0.140 ms
 2  bancada4.netlab.fe.up.pt (172.16.40.1)  0.451 ms  0.444 ms  0.436 ms
```

Attachment C.5. - Experiment 5: Pinging www.google.com from tux41

Capturing Machine: tux41

```
No.     Time          Source              Destination          Protocol Length Info
     1 0.000000      Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
     2 0.512122      Cisco_d4:1c:10      CDP/VTP/DTP/PAgP/UDLD CDP      436    Device ID: tux-sw4  Port ID: FastEthernet0/14
     3 1.263788      172.16.40.1         172.16.1.1           DNS      74     Standard query 0x8e4a A www.google.com
       4 1.265577       172.16.1.1          172.16.40.1           DNS       338     Standard query response 0x8e4a A www.google.com A 216.58.210.164 NS
ns1.google.com  NS  ns4.google.com  NS  ns2.google.com  NS  ns3.google.com  A  216.239.32.10  AAAA  2001:4860:4802:32::a  A  216.239.34.10  AAAA
2001:4860:4802:34::a A 216.239.36.10 AAAA 2001:4860:4802:36::a A 216.239.38.10 AAAA 2001:4860:4802:38::a
     5 1.265954      172.16.40.1         216.58.210.164       ICMP     98     Echo (ping) request  id=0x23e4, seq=1/256, ttl=64 (reply in 6)
     6 1.282625      216.58.210.164      172.16.40.1          ICMP     98     Echo (ping) reply    id=0x23e4, seq=1/256, ttl=50 (request in 5)
     7 1.282830      172.16.40.1         172.16.1.1           DNS      87     Standard query 0x9e4e PTR 164.210.58.216.in-addr.arpa
       8 1.284491          172.16.1.1              172.16.40.1             DNS        532     Standard query response 0x9e4e PTR 164.210.58.216.in-addr.arpa PTR
mad06s10-in-f164.1e100.net   PTR   mad06s10-in-f4.1e100.net   NS   d.in-addr-servers.arpa   NS   a.in-addr-servers.arpa   NS   c.in-addr-servers.arpa   NS
b.in-addr-servers.arpa  NS  e.in-addr-servers.arpa  NS  f.in-addr-servers.arpa  A  199.180.182.53  AAAA  2620:37:e000::53  A  199.253.183.183  AAAA
2001:500:87::87  A  196.216.169.10  AAAA  2001:43f8:110::10  A  200.10.60.53  AAAA  2001:13c7:7010::53  A  203.119.86.101  AAAA  2001:dd8:6::101  A
193.0.9.1 AAAA 2001:67c:e0::1
     9 2.009940      Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
    10 2.165038      Cisco_d4:1c:10      Cisco_d4:1c:10       LOOP     60     Reply
    11 2.267727      172.16.40.1         216.58.210.164       ICMP     98     Echo (ping) request  id=0x23e4, seq=2/512, ttl=64 (reply in 12)
    12 2.283935      216.58.210.164      172.16.40.1          ICMP     98     Echo (ping) reply    id=0x23e4, seq=2/512, ttl=50 (request in 11)
    13 3.268678      172.16.40.1         216.58.210.164       ICMP     98     Echo (ping) request  id=0x23e4, seq=3/768, ttl=64 (reply in 14)
    14 3.284890      216.58.210.164      172.16.40.1          ICMP     98     Echo (ping) reply    id=0x23e4, seq=3/768, ttl=50 (request in 13)
    15 4.010046      Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
    16 4.269999      172.16.40.1         216.58.210.164       ICMP     98     Echo (ping) request  id=0x23e4, seq=4/1024, ttl=64 (reply in 17)
    17 4.286157      216.58.210.164      172.16.40.1          ICMP     98     Echo (ping) reply    id=0x23e4, seq=4/1024, ttl=50 (request in 16)
    18 5.271249      172.16.40.1         216.58.210.164       ICMP     98     Echo (ping) request  id=0x23e4, seq=5/1280, ttl=64 (reply in 19)
    19 5.287431      216.58.210.164      172.16.40.1          ICMP     98     Echo (ping) reply    id=0x23e4, seq=5/1280, ttl=50 (request in 18)
    20 6.014632      Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
    21 6.272531      172.16.40.1         216.58.210.164       ICMP     98     Echo (ping) request  id=0x23e4, seq=6/1536, ttl=64 (reply in 24)
    22 6.275431      HewlettP_5a:7b:ea   G-ProCom_8c:af:af    ARP      60     Who has 172.16.40.1? Tell 172.16.40.254
    23 6.275456      G-ProCom_8c:af:af   HewlettP_5a:7b:ea    ARP      42     172.16.40.1 is at 00:0f:fe:8c:af:af
    24 6.288692      216.58.210.164      172.16.40.1          ICMP     98     Echo (ping) reply    id=0x23e4, seq=6/1536, ttl=50 (request in 21)
    25 7.273775      172.16.40.1         216.58.210.164       ICMP     98     Echo (ping) request  id=0x23e4, seq=7/1792, ttl=64 (reply in 26)
    26 7.289926      216.58.210.164      172.16.40.1          ICMP     98     Echo (ping) reply    id=0x23e4, seq=7/1792, ttl=50 (request in 25)
    27 8.024422      Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
    28 10.024140     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
    29 12.029267     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
    30 12.172207     Cisco_d4:1c:10      Cisco_d4:1c:10       LOOP     60     Reply
    31 12.276646     G-ProCom_8c:af:af   HewlettP_5a:7b:ea    ARP      42     Who has 172.16.40.254? Tell 172.16.40.1
    32 12.276904     HewlettP_5a:7b:ea   G-ProCom_8c:af:af    ARP      60     172.16.40.254 is at 00:21:5a:5a:7b:ea
```

Attachment C.6. - Experiment 6: Downloading file from UP ftp server in tux41

Capturing Machine: tux41

```
No.     Time          Source              Destination          Protocol Length Info
     1 0.000000      Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
     2 1.171599      Cisco_d4:1c:10      Cisco_d4:1c:10       LOOP     60     Reply
     3 1.469267      172.16.40.1         172.16.1.1           DNS      73     Standard query 0x1da1 A mirrors.up.pt
       4 1.470982        172.16.1.1            172.16.40.1          DNS      337    Standard query response 0x1da1 A mirrors.up.pt A 193.137.29.15 NS ns4.up.pt NS ns3.up.pt NS ns2.up.pt NS
ns1.up.pt  A  193.137.55.30  AAAA  2001:690:2200:a10::30  A  193.137.55.31  AAAA  2001:690:2200:a10::31  A  193.137.55.32  AAAA  2001:690:2200:a10::32  A  193.137.55.33  AAAA
2001:690:2200:a10::33
     5 1.471110      172.16.40.1         193.137.29.15        TCP      74     43939 → 21 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3204367 TSecr=0 WS=128
       6 1.474405        193.137.29.15         172.16.40.1          TCP      74     21 → 43939 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1380 SACK_PERM=1 TSval=256408331
TSecr=3204367 WS=128
     7 1.474428      172.16.40.1         193.137.29.15        TCP      66     43939 → 21 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3204368 TSecr=256408331
     8 1.481140      193.137.29.15       172.16.40.1          FTP      139    Response: 220-Welcome to the University of Porto's mirror archive (mirrors.up.pt)
     9 1.481161      193.137.29.15       172.16.40.1          FTP      135    Response: 220----------------------------------------------------------------
    10 1.481463      193.137.29.15       172.16.40.1          FTP      72     Response: 220-
    11 1.481470      193.137.29.15       172.16.40.1          FTP      151    Response: 220-All connections and transfers are logged. The max number of connections is 200.
    12 1.481473      193.137.29.15       172.16.40.1          FTP      72     Response: 220-
    13 1.481475      193.137.29.15       172.16.40.1          FTP      140    Response: 220-For more information please visit our website: http://mirrors.up.pt/
    14 1.481478      193.137.29.15       172.16.40.1          FTP      127    Response: 220-Questions and comments can be sent to mirrors@uporto.pt
    15 1.481731      193.137.29.15       172.16.40.1          FTP      72     Response: 220-
    16 1.481739      193.137.29.15       172.16.40.1          FTP      72     Response: 220-
    17 1.481742      193.137.29.15       172.16.40.1          FTP      72     Response: 220
    18 1.481990     172.16.40.1        193.137.29.15      TCP      66     43939 → 21 [ACK] Seq=1 Ack=74 Win=29312 Len=0 TSval=3204370 TSecr=256408333
    19 1.482008     172.16.40.1        193.137.29.15      TCP      66     43939 → 21 [ACK] Seq=1 Ack=143 Win=29312 Len=0 TSval=3204370 TSecr=256408333
    20 1.482014     172.16.40.1        193.137.29.15      TCP      66     43939 → 21 [ACK] Seq=1 Ack=149 Win=29312 Len=0 TSval=3204370 TSecr=256408333
    21 1.482018     172.16.40.1        193.137.29.15      TCP      66     43939 → 21 [ACK] Seq=1 Ack=234 Win=29312 Len=0 TSval=3204370 TSecr=256408333
    22 1.482023     172.16.40.1        193.137.29.15      TCP      66     43939 → 21 [ACK] Seq=1 Ack=240 Win=29312 Len=0 TSval=3204370 TSecr=256408333
    23 1.482028     172.16.40.1        193.137.29.15      TCP      66     43939 → 21 [ACK] Seq=1 Ack=314 Win=29312 Len=0 TSval=3204370 TSecr=256408333
    24 1.482033     172.16.40.1        193.137.29.15      TCP      66     43939 → 21 [ACK] Seq=1 Ack=375 Win=29312 Len=0 TSval=3204370 TSecr=256408333
    25 1.482037     172.16.40.1        193.137.29.15      TCP      66     43939 → 21 [ACK] Seq=1 Ack=381 Win=29312 Len=0 TSval=3204370 TSecr=256408333
    26 1.482042     172.16.40.1        193.137.29.15      TCP      66     43939 → 21 [ACK] Seq=1 Ack=387 Win=29312 Len=0 TSval=3204370 TSecr=256408333
    27 1.482046     172.16.40.1        193.137.29.15      TCP      66     43939 → 21 [ACK] Seq=1 Ack=393 Win=29312 Len=0 TSval=3204370 TSecr=256408333
    28 1.482388     172.16.40.1        193.137.29.15      FTP      80     Request: USER anonymous
    29 1.484971     193.137.29.15       172.16.40.1       TCP      66     21 → 43939 [ACK] Seq=393 Ack=15 Win=29056 Len=0 TSval=256408334 TSecr=3204370
    30 1.484992     172.16.40.1        193.137.29.15      FTP      68     Request: 
    31 1.487042     193.137.29.15       172.16.40.1       TCP      66     21 → 43939 [ACK] Seq=393 Ack=17 Win=29056 Len=0 TSval=256408335 TSecr=3204370
    32 1.487050     193.137.29.15       172.16.40.1       FTP      100    Response: 331 Please specify the password.
    33 1.487137     172.16.40.1        193.137.29.15      FTP      71     Request: PASS 
    34 1.530540     193.137.29.15       172.16.40.1       TCP      66     21 → 43939 [ACK] Seq=427 Ack=22 Win=29056 Len=0 TSval=256408345 TSecr=3204371
    35 1.530570     172.16.40.1        193.137.29.15      FTP      68     Request: 
    36 1.532558     193.137.29.15       172.16.40.1       TCP      66     21 → 43939 [ACK] Seq=427 Ack=24 Win=29056 Len=0 TSval=256408346 TSecr=3204382
    37 1.632771     193.137.29.15       172.16.40.1       FTP      89     Response: 230 Login successful.
    38 1.634224     172.16.40.1        193.137.29.15      FTP      76     Request: CWD debian
    39 1.636717     193.137.29.15       172.16.40.1       TCP      66     21 → 43939 [ACK] Seq=450 Ack=34 Win=29056 Len=0 TSval=256408372 TSecr=3204408
    40 1.636731     172.16.40.1        193.137.29.15      FTP      68     Request: 
    41 1.638643     193.137.29.15       172.16.40.1       TCP      66     21 → 43939 [ACK] Seq=450 Ack=36 Win=29056 Len=0 TSval=256408373 TSecr=3204408
    42 1.640319     193.137.29.15       172.16.40.1       FTP      103    Response: 250 Directory successfully changed.
    43 1.640541     172.16.40.1        193.137.29.15      FTP      72     Request: TYPE I
    44 1.682713     193.137.29.15       172.16.40.1       TCP      66     21 → 43939 [ACK] Seq=487 Ack=42 Win=29056 Len=0 TSval=256408384 TSecr=3204409
    45 1.682745     172.16.40.1        193.137.29.15      FTP      68     Request: 
    46 1.684784     193.137.29.15       172.16.40.1       TCP      66     21 → 43939 [ACK] Seq=487 Ack=44 Win=29056 Len=0 TSval=256408384 TSecr=3204420
    47 1.684793     193.137.29.15       172.16.40.1       FTP      97     Response: 200 Switching to Binary mode.
    48 1.684877     172.16.40.1        193.137.29.15      FTP      70     Request: PASV
    49 1.726909     193.137.29.15       172.16.40.1       TCP      66     21 → 43939 [ACK] Seq=518 Ack=48 Win=29056 Len=0 TSval=256408395 TSecr=3204420
    50 1.726937     172.16.40.1        193.137.29.15      FTP      68     Request: 
    51 1.729320     193.137.29.15       172.16.40.1       TCP      66     21 → 43939 [ACK] Seq=518 Ack=50 Win=29056 Len=0 TSval=256408395 TSecr=3204431
    52 1.729855     193.137.29.15       172.16.40.1       FTP      118    Response: 227 Entering Passive Mode (193,137,29,15,230,247).
     53 1.729999        172.16.40.1             193.137.29.15         TCP      74     60013 → 59127 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3204432 TSecr=0 
WS=128
     54 1.732114      193.137.29.15        172.16.40.1         TCP      74     59127 → 60013 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1380 SACK_PERM=1 TSval=256408396 
TSecr=3204432 WS=128
    55 1.732137     172.16.40.1        193.137.29.15      TCP      66     60013 → 59127 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3204432 TSecr=256408396
    56 1.732163     172.16.40.1        193.137.29.15      FTP      77     Request: RETR README
    57 1.770576     193.137.29.15       172.16.40.1       TCP      66     21 → 43939 [ACK] Seq=570 Ack=61 Win=29056 Len=0 TSval=256408406 TSecr=3204432
    58 1.770603     172.16.40.1        193.137.29.15      FTP      68     Request: 
    59 1.772731     193.137.29.15       172.16.40.1       TCP      66     21 → 43939 [ACK] Seq=570 Ack=63 Win=29056 Len=0 TSval=256408406 TSecr=3204442
    60 1.774892     193.137.29.15       172.16.40.1       FTP      132    Response: 150 Opening BINARY mode data connection for README (1184 bytes).
    61 1.785786     193.137.29.15       172.16.40.1       FTP-DATA 1250   FTP Data: 1184 bytes (PASV) (RETR README)
    62 1.785799     193.137.29.15       172.16.40.1       TCP      66     59127 → 60013 [FIN, ACK] Seq=1185 Ack=1 Win=29056 Len=0 TSval=256408409 TSecr=3204432
    63 1.785824     172.16.40.1        193.137.29.15      TCP      66     60013 → 59127 [ACK] Seq=1 Ack=1185 Win=32128 Len=0 TSval=3204445 TSecr=256408409
    64 1.809119     172.16.40.1        172.16.1.1        DNS      86     Standard query 0x3559 PTR 15.29.137.193.in-addr.arpa
     65 1.813329      172.16.1.1         172.16.40.1        DNS      361    Standard query response 0x3559 PTR 15.29.137.193.in-addr.arpa PTR mirrors.up.pt NS ns4.up.pt NS ns1.up.pt NS 
ns2.up.pt NS ns3.up.pt A 193.137.55.30 AAAA 2001:690:2200:a10::30 A 193.137.55.31 AAAA 2001:690:2200:a10::31 A 193.137.55.32 AAAA 2001:690:2200:a10::32 A 193.137.55.33 
AAAA 2001:690:2200:a10::33
    66 1.813943     172.16.40.1        193.137.29.15      TCP      66     43939 → 21 [ACK] Seq=63 Ack=636 Win=29312 Len=0 TSval=3204453 TSecr=256408407
    67 1.821944     172.16.40.1        193.137.29.15      TCP      66     60013 → 59127 [ACK] Seq=1 Ack=1186 Win=32128 Len=0 TSval=3204455 TSecr=256408409
    68 1.823993     193.137.29.15       172.16.40.1       FTP      90     Response: 226 Transfer complete.
    69 1.824030     172.16.40.1        193.137.29.15      TCP      66     43939 → 21 [ACK] Seq=63 Ack=660 Win=29312 Len=0 TSval=3204455 TSecr=256408419
    70 1.824090     172.16.40.1        193.137.29.15      FTP      70     Request: QUIT
    71 1.866620     193.137.29.15       172.16.40.1       TCP      66     21 → 43939 [ACK] Seq=660 Ack=67 Win=29056 Len=0 TSval=256408429 TSecr=3204455
    72 1.866648     172.16.40.1        193.137.29.15      FTP      68     Request: 
    73 1.868853     193.137.29.15       172.16.40.1       TCP      66     21 → 43939 [ACK] Seq=660 Ack=69 Win=29056 Len=0 TSval=256408430 TSecr=3204466
    74 1.868863     193.137.29.15       172.16.40.1       FTP      80     Response: 221 Goodbye.
    75 1.868871     193.137.29.15       172.16.40.1       TCP      66     21 → 43939 [FIN, ACK] Seq=674 Ack=69 Win=29056 Len=0 TSval=256408430 TSecr=3204466
    76 1.868956     172.16.40.1        193.137.29.15      TCP      66     43939 → 21 [FIN, ACK] Seq=69 Ack=675 Win=29312 Len=0 TSval=3204466 TSecr=256408430
    77 1.868977     172.16.40.1        193.137.29.15      TCP      66     60013 → 59127 [FIN, ACK] Seq=1 Ack=1186 Win=32128 Len=0 TSval=3204466 TSecr=256408409
    78 1.870828     193.137.29.15       172.16.40.1       TCP      66     59127 → 60013 [ACK] Seq=1186 Ack=2 Win=29056 Len=0 TSval=256408431 TSecr=3204466
    79 1.871227     193.137.29.15       172.16.40.1       TCP      66     21 → 43939 [ACK] Seq=675 Ack=70 Win=29056 Len=0 TSval=256408431 TSecr=3204466
    80 2.009875     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
    81 4.009879     Cisco_d4:1c:10      Spanning-tree-(for-bridges)_00 STP      60     Conf. Root = 32768/40/30:37:a6:d4:1c:00  Cost = 0  Port = 0x8010
```

Attachment D - Successful Download Report

Output of the developed “download” program after the execution of the following command, which should download the README file from inside de debian directory in mirrors.up.pt ftp server:

```
./download ftp://anonymous@mirrors.up.pt/debian/README

220-Welcome to the University of Porto's mirror archive (mirrors.up.pt)
220----------------------------------------------------------------
220-
220-All connections and transfers are logged. The max number of connections
is 200.
220-
220-For more information please visit our website: http://mirrors.up.pt/
220-Questions and comments can be sent to mirrors@uporto.pt
220-
220-
220

->> USER anonymous
331 Please specify the password. //User accepted, server requesting
password

->> PASS ****
230 Login successful. //Login credentials accepted, login successful

->> CWD debian
250 Directory successfully changed. //Successfully changed to ‘debian’
directory

->> TYPE I
200 Switching to Binary mode. //Successfully switched to Binary Mode

->> PASV
227 Entering Passive Mode (193,137,29,15,213,68). //Entering passive mode.
A data connection is open at 193.137.29.15:54596 to download the file

->> RETR README
150 Opening BINARY mode data connection for README (1184 bytes).
226 Transfer complete. //Transfer is complete

->> QUIT
221 Goodbye. //Server acknowledged quitting
```

The lines starting with “->>” are the FTP commands sent by the developed application and all the other lines are the server responses.