Aplicação de Download e Configuração de Rede

Relatório Final

Mestrado Integrado em Engenharia Informática e Computação

Redes de Computadores

Turma: 3MIEIC01

Gabriel Martins Souto - ei12087@fe.up.pt
Márcio Filipe Vilela Fontes - ei12183@fe.up.pt

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

23 de Dezembro de 2014

Sumário

Com o presente projeto pretendeu-se desenvolver uma aplicação de transferência de ficheiros utilizando FTP e realizar experiências para configurar uma rede de uma forma apropriada.

Relativamente à aplicação de transferência de ficheiros, esta teve como objetivo realizar o *download* de forma segura e sob forma de autenticação anónima ou especificando um nome de utilizador e palavra-passe. Já relativamente às experiências realizadas, estas tiveram como objetivo a aprendizagem da comunicação entre computadores em redes virtuais (VLANs), configuração correta do *switch* e *router*, e a diferença entre a utilização e a não utilização do NAT[1].

O presente relatório tem como objetivo descrever, sucintamente, o processo de implementação da aplicação, as respetivas funções, experiência realizadas na configuração da rede e os seus objetivos.

Palavras-Chave: Redes de Computadores; Transferência; Ficheiro; Endereço; Router; Switch

[1] NAT - Network Address Translation

Conteúdo

Lista de Figuras

1 Introdução

Este relatório está associado e pretende demonstrar o desenvolvimento do projeto *Aplicação de Download e Configuração de Rede* para a unidade curricular de Redes de Computadores onde será caracterizado, definido e analisado todos os aspetos referentes ao mesmo, sendo nele descrito todo o processo de implementação da aplicação (parte 1) e configuração da rede (parte 2).

1.1 Objetivos

Com o desenvolvimento deste projeto pretende-se adquirir conhecimentos mais coesos de download de ficheiros através de FTP, configuração de redes de uma forma apropriada, uma aplicação (ainda que diminuta) de redes de computadores e facilidade de abordagem do problema em questão em futuros projetos. Não obstante, pretende-se no final deste projeto que se cumpra aquilo que foi estipulado, mantendo acima de tudo a qualidade exigida, quer pelos próprios alunos que desenvolveram a aplicação, quer pelos docentes da unidade curricular.

1.2 Estrutura do Relatório

O presente relatório está dividido nas seguintes secções:

- **Aplicação de Download** - descrição da arquitetura da aplicação e amostra de uma transferência de download;
- **Configuração e Análise da Rede** - para cada experiência é descrita a arquitetura da rede, os seus objetivos, comandos utilizados e análise dos *logs* capturados;
- **Considerações Finais** - descrição das conclusões, dificuldades e reflexão sobre os objetivos de aprendizagem alcançados;
- **Anexos** - código da aplicação, comandos de configuração da rede e *logs* capturados.

2 Aplicação de Download

2.1 Arquitetura

Relativamente à parte da aplicação, a função `main`, numa primeira fase, trata de validar o *input* do utilizador e, posteriormente, de fazer o *parser*. Depois de obtidos todos os dados é chamada a função `ftp`, que recebe como parâmetros a estrutura `FTP_Data` que contém os dados. Depois prossegue-se (utilizando a função `ftp_init`) para a fase de *send* e *receive* dos códigos (`USER`, `PASS`, `PASV`, etc.) até que sejam obtidos todos os dados necessários (`filesize` e `retr_port`) para se efetuar a transferência do ficheiro. A função `ftp` trata depois então de chamar a `ftp_transfer` para realizar a transferência - através de "partes"(*chunks*) - ficando a aplicação em *loop*, até que seja totalmente transferido. Quando a transferência terminar, é chamada a função `ftp_quit` que termina a ligação e, consequentemente, a aplicação.

2.2 Estrutura do Código

2.2.1 Estruturas de Dados

Neste projeto decidimos dividir o código por funcionalidades/objetivos de funções. Para tal foram criados os ficheiros tais como `ftp.c`, `message.c`, `config.c`, etc. como é possível verificar na secção 5.

Abaixo segue-se uma breve descrição sobre a estrutura utilizada:

- **FTP_Data** - estrutura os dados que são obtidos através do *parse*, tais como *username*, *password*, *hostname*, *url path*, *filename* e *port*.

2.2.2 Funções

Relativamente às funções implementadas, é de realçar uma especial importância as seguintes:

- **int ftp(FTP_Data data)** - esta é a função principal da aplicação. Trata de determinar o IP através do *hostname* e associar o *socket*, como também iniciar toda a ligação FTP (`ftp_init`), fazer a transferência do ficheiro (`ftp_transfer`) e fechar a ligação (`ftp_quit`).
- **int ftp_init(int sockfd, FTP_Data data, int *retr_port, int *filesize)** - inicia a ligação FTP, fazendo a autenticação (anónima ou não), determinando o caminho URL, passando para o modo passivo, determinando a porta e o tamanho do respetivo ficheiro.

- **int ftp_transfer(int sockfd, char *ip, int retr_port, char *filename, int filesize)** - realiza a transferência do ficheiro.
- **int asocket(char *ip, int port)** - tenta obter o socket tendo em conta o IP e a porta.

2.2.3 Casos de Uso

A aplicação aceita apenas um formato de parâmetros. Se esta for executada, então deverá ser invocada da seguinte forma:

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

Se a função `test_args` validar o *input* da invocação da aplicação, então proceder-se-á à parte do *parser*.

No caso de o utilizador inserir incorretamente os parâmetros, então o programa terminará, através da função `exit`, mostrando uma mensagem ao utilizador sobre o *input* do programa.

2.3 Transferência de Ficheiro

Para efeitos de teste da aplicação/transferência de ficheiro, ao realizar a seguinte invocação:

```
./download ftp://mirrors.fe.up.pt/debian/doc/bug-reporting.txt
```

O resultado que será mostrado no ecrã será o seguinte:

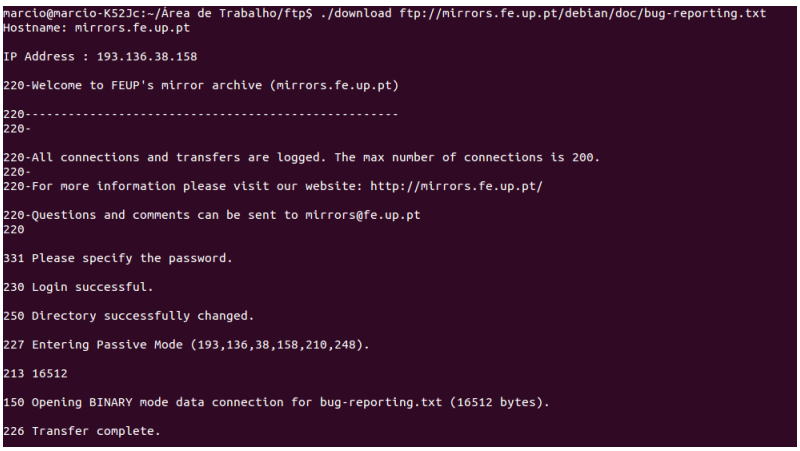


Figura 1: Transferência de ficheiro com sucesso

3 Configuração e Análise da Rede

3.1 Experiência 1

Nesta primeira experiência, foi pedido para configurar endereços de IP entre dois computadores através de um *switch*.

Inicialmente, desligou-se o cabo que ligava o *switch* ao *router*, resultando na perda de ligação à *Internet* a partir dos computadores. Feito isso, configuraram-se os computadores com o IP pedido.

O protocolo ARP (*Address Resolution Protocol*) é usado quando um computador, conectado a uma rede, pretende enviar uma trama *ethernet*, sendo conhecedor de um endereço IP. Deste modo, cada trama ARP contém ambos o IP de origem e de destino, identificando assim o caminho que se pretende fazer. No entanto, é também identificado o endereço MAC do computador onde esteve na última vez, assim como o endereço MAC para onde esta se dirige. Tal deve-se devido ao facto de, em cada momento, a trama ARP poder situar-se num qualquer sítio aleatório da rede.

Quando se envia um comando `ping`, no caso do IP ainda não estar na sua tabela ARP, envia antes uma trama ARP, de forma a obter esta informação, seguindo-se depois do `ping` normal. No `ping` normal, será enviada uma trama ICMP, de `echo request`, para o computador de destino. Este computador irá, por sua vez, enviar uma trama ICMP, de `echo reply`. Os endereços MAC e IP destas duas tramas estarão de acordo com a *ARP table*.

Para determinar o tipo de trama *ethernet* é preciso analisar os 2 bytes correspondentes ao *header*. No caso de conter o valor 0x0800, corresponde ao protocolo ARP. Caso contenha o valor 0x0806 corresponderá ao tipo IPv4. No caso de se tratar de um protocolo IP, existe no *header* de IP a informação sobre o protocolo correspondente que, no caso do ICMP, conterá o valor 1. Em ambos os casos, haverá informação sobre o tamanho da trama, de acordo com o respetivo protocolo.

Por último, uma interface *loopback* é uma *interface* virtual que não está associada a qualquer componente físico num computador, sendo habitualmente usada para testes. Assim, qualquer mensagem transmitida para esta *interface* é imediatamente recebida pelo emissor. Esta interface deve estar sempre ativa e que, para qualquer vizinhança, esta apenas será perdida se a componente física for desativada.

3.2 Experiência 2

Na segunda experiência, foi pedido para se configurar duas VLANs no *switch*.

Para isto, foi necessário criar no *switch* as VLANs, assim como adicionar portas a cada VLAN. Para além disto, também foi necessário permitir, em cada computador, a resposta a `pings` de tipo *broadcast*.

Nesta experiência, existem dois domínios *broadcast*, sendo tal comprovado pelos *logs* obtidos. Os *logs* mostram o que acontece na rede após um `ping broadcast` enviado pelo tux61. Conclui-se que o tux62 não recebe os pacotes ICMP, ao contrário do tux64. No entanto, este último apenas não responde aos pacotes pois a opção para resposta a `pings broadcast` está desativada.

3.3 Experiência 3

Esta experiência teve por base a configuração de um computador como *router* (neste caso, do tux64), por forma a permitir a comunicação entre as duas VLANs criadas na experiência anterior.

Deste modo, é necessário que os tux61 e tux62 conheçam um caminho pelo qual pode enviar o pacote quando o computador destinado não se encontra na rede. Configurando o tux64 como *router*, permitindo *ip forwarding*, colocando-o como *interface* em cada uma das VLANs criadas e definindo-o como *gateway* dos outros computadores, é obtido o pretendido.

Esta configuração é descrita pelas tabelas de reencaminhamento em cada tux, a qual contém as rotas com os destinos para os quais se devem enviar pacotes, sabendo o IP de destino. As tabelas de encaminhamento caracterizam as rotas em várias componentes: IP de destino, *gateway* (IP para o qual se devem enviar os pacotes, caso não exista nenhuma rota para o IP de destino), máscara de rede e a *interface* na qual se envia o pacote.

Em relação aos *logs* observado, é possível observar o que se conclui na experiência 1 sobre os endereços IP e MAC existentes nos pacotes ARP e ICMP, obtendo-se a partir de um `ping` enviado do tux61 para o tux62, observado a partir do tux64. Com esses dados, vê-se que, para enviar esse `ping` para o tux62, o pacote ICMP respetivo que passa na *interface* `eth0` contém o endereço MAC de destino do tux64(do lado da *interface* `eth0`) e não o do tux62, e na situação de resposta o mesmo se repete. Em relação aos pacotes ICMP que passam pela *interface* `eth1`, a situação é semelhante.

3.4 Experiência 4

Esta experiência teve como objetivo a configuração de um *router* para que, posteriormente, possa ser implementado o NAT.

A configuração do *router* começa por atribuir as interfaces para a rede configurada, tendo conta as ligações efetuadas no *switch*, adicionando posteriormente as rotas que possibilitem a correta comunicação entre todos os tuxs.

Implementou-se, então, no tux61, o tux64 como *default gateway*. De seguida, definiu-se o *router* como *default gateway* dos tux64 e tux62. Efetuou-se um `ping` a partir de cada um dos tux para o *router* de modo a confirmar que esta operação tinha sido sucedida.

Fez-se um `ping` de teste ao *router* central, mas sem sucesso. Isto deveu-se ao facto de ainda não se ter implementado NAT. O que se passou foi que o pacote ICMP tinha como origem o IP do computador e como destino o *router*. O pacote chegava com sucesso ao *router*, mas, como este não tinha acesso à sub-rede onde estava o IP do computador, não soubesse para onde enviar o pacote de resposta.

Configurando o NAT, todos os endereços da rede são mapeados para endereços públicos e, desta forma, a resposta é concretizável, tendo-se sucesso com o `ping`.

3.5 Experiência 5

Esta experiência teve como objetivo a possibilidade de a rede criada possuir um serviço resolução de nomes, ou DNS (*Domain Name System*).

Para um *host* poder ter serviços de DNS basta que tenha, no mínimo, uma tabela na qual, para cada *hostname*, seja identificado o IP que lhe está associado. Assim, na experiência, bastou apenas identificar no computador a localização de um servidor DNS existente, contendo já estas tabelas. Para o computador poder ter acesso a estas informações, bastou editar o ficheiro resolv.conf, existente na pasta /etc, colocando a informação seguinte: `search netlab.fe.up.pt nameserver 172.16.1.1`.

Quando se executa um ping para *www.google.com*, é enviado um pacote DNS para o servidor DNS configurado, requerendo o IP do site. Após este pedido, é recebido outro pacote DNS enviado pelo servidor com a informação pretendida e, após isso, o primeiro `ping` é enviado. Posteriormente, antes de cada pacote enviado, é visualizado o mecanismo reverso, que perante um IP conhecido, o servidor DNS informa sobre qual a máquina correspondente.

3.6 Experiência 6

Esta experiência teve como objetivo a criação de ligações TCP usando para o efeito a aplicação desenvolvida para download de um ficheiro de um servidor FTP.

Pela análise dos *logs* capturados na realização da experiência, repara-se que ao iniciar a execução da aplicação, é estabelecida uma ligação TCP com o servidor. Esta ligação diz respeito ao primeiro *socket* aberto pela aplicação, que é usado para o envio de comandos e recebe as respostas do servidor.

A seguir ao envio do comando `PASV`, é calculada a nova porta e feita uma nova ligação. Essa segunda ligação pode ser também observada no *log*, onde se repara que embora esteja a ser feita uma comunicação para o mesmo endereço IP, a porta de destino é diferente (inicialmente estava a comunicar para a porta 21 para o controlo da ligação FTP, passando de seguida a comunicar para a porta 42009).

As ligações TCP estão divididas em três fases: estabelecimento de comunicação, transferência de dados e encerramento da comunicação.

Na fase de estabelecimento, acontece o seguinte:

1. O cliente envia uma trama SYN ao servidor, que o informa que se pretende realizar uma transferência de dados;
2. O servidor responde com uma trama SYN ACK que informa o cliente que o servidor está pronto para iniciar a comunicação;
3. O cliente responde com uma trama ACK que informa o servidor que também ele está pronto.

No modo de transferência de dados, transferem-se os dados que foram pedidos ao servidor. Nesta fase existem diversos mecanismos que asseguram que a transferência ocorre com sucesso.

Feita a transferência, o cliente envia uma trama `FIN`, ao qual o servidor responde com uma trama `ACK` e, quando estiver também preparado para terminar a comunicação, irá ocorrer novamente a mesma troca de tramas, mas com sentidos inversos.

Na segunda parte da experiência, quando se abre uma segunda conexão a partir do tux62, verifica-se que há uma estabilização na velocidade de download em ambos os computadores. Consegue-se ver, no gráfico gerado pelo *Wireshark* da captura do segundo computador, que há um pico inicial que depois vai diminuindo. No gráfico obtido pelos *logs* do tux61 vê-se antes um gráfico com vários picos.

4 Considerações Finais

4.1 Dificuldades

A nossa maior dificuldade residiu no desenvolvimento da aplicação de *download* e da gestão da transferência dos ficheiros. Após entendermos melhor como funcionava o FTP, tornou-se bastante fácil em desenvolver a aplicação. Contudo, deve-se realçar a importância das dificuldades que tivemos, visto que nos levou a empenhar cada vez mais para entregar uma aplicação com uma boa qualidade.

4.2 Conclusões

A parte realmente crucial a reter do desenvolvimento deste projeto é mesmo a complexidade e perfecionismo que é necessário para configurar uma rede (da forma mais apropriada). Relativamente à aplicação de *download*, um dos aspetos a reter é que um simples aumento do tamanho do ficheiro pode levar a crescimento quase exponencial da complexidade temporal. Este é um dos aspetos que leva a transmissão de dados a ser uma das áreas mais complexas e difíceis de otimizar.

5 Anexos

Nesta secção segue-se o código-fonte desenvolvido para a aplicação, os comandos de configuração e *logs* capturados.

Código Fonte

config.h

Descrição: header file das configurações.

```
#ifndef _CONFIG_H_
#define _CONFIG_H_

#include <stdio.h>
#include <netdb.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include "defines.h"

/**
@brief Establishes the connection based on hostname
@param hostname - hostname
@return Returns the IP address if succeeded
*/

char* getIP(char *hostname);

/**
@brief Tries to assign the socket file descriptor
(asocket -> assign socket)
@param ip - IP address that comes from getIP function
@param port - port that comes from parsing
@return sockfd if success
*/

int asocket(char *ip, int port);

#endif
```

config.c

Descrição: definição do IP e do *socket*.

```
#include "config.h"

char* getIP(char *hostname){

  struct hostent *h;

  if((h = gethostbyname(hostname)) == NULL){
    herror("gethostbyname");
    exit(ABORT);
  }

  printf("Hostname: %s\n", h->h_name);

  char *ip = inet_ntoa(*((struct in_addr *)h->h_addr));

  printf("\nIP Address: %s\n\n",
         inet_ntoa(*((struct in_addr *)h->h_addr)));

  return ip;
}

int asocket(char *ip, int port){

  int sockfd;
  struct sockaddr_in server_addr;
  struct in_addr address;

  bzero((char*) &server_addr, sizeof(server_addr));
  server_addr.sin_family = AF_INET;
  server_addr.sin_port = htons(port);

  if ((sockfd = socket(AF_INET, SOCK_STREAM, ZERO))
       < ZERO){
    printf("[ERROR]: Socket could not be created\n");
    exit(ABORT);
  }
```

```
  if(inet_aton(ip, &address) == ZERO) {
    printf("[ERROR]: Failure getting an IP\n");
    exit(ABORT);
  }

  server_addr.sin_addr.s_addr = address.s_addr;

  if(connect(sockfd, (struct sockaddr*) &server_addr,
             sizeof (server_addr)) != ZERO){
    printf("[ERROR]: Failure connecting to hport\n");
    exit(ABORT);
  }

  return sockfd;
}
```

defines.h

Descrição: definição das constantes.

```
#define OK 0
#define ABORT -1
#define ERROR -1
#define NOT_FOUND -1

#define ZERO 0
#define ONE 1
#define TWO 2
#define FOUR 4
#define FIVE 5
#define SIX 6

#define TWO_POW_EIGHT 256

#define MAX_SIZE 255
#define MAX_SIZE_WITH_NULL 256

#define FILE_DATA_SIZE 255
#define FTP_SLEEP 200000
#define FILE_SLEEP 150000

#define FTP_PORT 21
#define FTP_START "ftp://"

#define STR_USER "USER "
#define STR_PASSWORD "PASS "
#define STR_CWD "CWD "
#define STR_PASV "PASV"
#define STR_SIZE "SIZE "
#define STR_RETR "RETR "
#define STR_QUIT "QUIT"
#define BLANK ""

#define CODE_USER "220"
#define CODE_PASSWORD "331"
#define CODE_USER_LOGGED "230"
#define CODE_CWD "250"
```

```
#define CODE_PASV "227"
#define CODE_SIZE "213"
#define CODE_QUIT "226"

#define USER_ANONYMOUS "anonymous"
#define PASSWORD_ANONYMOUS "anonymous"

#define COMMA ','
#define RIGHT_PARENTHESIS ')'
#define SLASH '/'
#define AT '@'
#define COLON ':'
#define NEWLINE '\n'
#define NULL_CHAR '\0'
#define NEWLINE_STRING "\n"

#define FTP_CODE_NO_FILE "550"
#define FTP_CODE_WRONG_CREDENTIALS "430"

#define WRITE_BINARY "wb"
```

ftp.h

Descrição: header file de ftp.

```
#ifndef _FTP_H_
#define _FTP_H_

#include <stdio.h>
#include "defines.h"
#include "ftp_data.h"
#include "config.h"

/**
@brief Logins the user (anonymous or not) to the server
@param sockfd - socket file descriptor
@param str - string received
@param user - username
@param password - user's password
@return Returns 0 if success
*/

int login(int sockfd, char *str, char *user,
          char *password);

/**
@brief If there is a path, sends the CWD (250) code to
server and then waits for response
@param sockfd - socket file descriptor
@param str - buffer string
@param url - url path
@return Returns OK (0) if success
*/

int path(int sockfd, char *str, char *url_path);

/**
@brief Sends the Passive Mode code to server and then
awaits for response
@param sockfd - socket file descriptor
@param str - buffer string
@return Returns OK (0) if success
```

```
*/

int passive_mode(int sockfd, char *str);

/**
@brief Saves the port
@param str - string in the format (%d,%d,%d,%d,%d,%d)
@param port - port
@return Returns OK (0) if success
*/

int port(char *str, int *port);

/**
@brief Sends the Size code to server and then awaits
for response
@param sockfd - socket file descriptor
@param str - buffer string
@param filename - file name
@param filesize - file size
@return Returns OK (0) if success
*/

int file_size(int sockfd, char *str, char *filename,
              int *filesize);

/**
@brief Sends the Retrieve code to server
@param sockfd - socket file descriptor
@param str - buffer string
@param filename - file name
@return Returns OK (0) if success
*/

int retrieve(int sockfd, char *str, char *filename);

/**
@brief Receives the QUIT command to end connection and
sends the response
@param sockfd - socket file descriptor
```

```
@param str - string received
@return Returns 0 if success
*/

int quit(int sockfd, char *str);

/**
@brief Retrieves the file specified
@param sockfd - socket file descriptor
@param filename - file name
@param filesize - file size
@return Returns OK (0) if success
*/

int retrieve_file(int sockfd, char *filename,
                  int filesize);

/**
@brief Tests if there is an error code in the message
(http://en.wikipedia.org/wiki/
List_of_FTP_server_return_codes)
@note Only some errors were taken into consideration,
since there are a lot of them
@param message - message string
@return Returns OK (0) if success
*/

int ftp_valid(char *message);

/**
@brief Initializes the FTP (login, path, passive mode,
retrieve and filesize)
@param sockfd - socket file descriptor
@param data - data retrieved from parser
@param retr_port - retrieve port
@param filesize - file size
@return Returns OK (0) if success
*/

int ftp_init(int sockfd, FTP_Data data, int *retr_port,
```

```
            int *filesize);

/**
@brief Begins the file transfer
@param sockfd - socket file descriptor
@param ip - internet protocol
@param retr_port - retrieve port
@param filename - file name
@param filesize - file size
@return Returns OK (0) if success
*/

int ftp_transfer(int sockfd, char *ip, int retr_port,
                 char *filename, int filesize);

/**
@brief Quits the FTP application (closes the socket)
@param sockfd - socket file descriptor
@return Returns OK (0) if success
*/

int ftp_quit(int sockfd);

/**
@brief Begins the FTP application
@param data - data retrieved from parser
*/

int ftp(FTP_Data data);

#endif
```

ftp.c

Descrição: aplicação FTP, inicialização e transferência dos ficheiros.

```
#include "ftp.h"

int login(int sockfd, char *str, char *user,
          char *password){

  receive_from(sockfd, str, CODE_USER);
  send_to(sockfd, str, STR_USER, user);

  receive_from(sockfd, str, CODE_PASSWORD);
  send_to(sockfd, str, STR_PASSWORD, password);

  receive_from(sockfd, str, CODE_USER_LOGGED);

  return OK;

}

int path(int sockfd, char *str, char *url_path){

  if (strcmp(url_path, BLANK) != ZERO){
    send_to(sockfd, str, STR_CWD, url_path);
    receive_from(sockfd, str, CODE_CWD);
  }

  return OK;

}

int passive_mode(int sockfd, char *str){

  send_to(sockfd, str, STR_PASV, BLANK);
  receive_from(sockfd, str, CODE_PASV);

  return OK;

}
```

```
int port(char *str, int *port){

  int pos1, pos2, pos3;
  char *buffer = malloc(MAX_SIZE_WITH_NULL);

  pos1 = find_nth(str, COMMA, ZERO, FOUR);
  pos2 = find_nth(str, COMMA, pos1 + ONE, ONE);
  pos3 = find_nth(str, RIGHT_PARENTHESIS,
                  pos2 + ONE, ONE);

  *port = TWO_POW_EIGHT*atoi(strncpy(buffer,
          str + pos1 + ONE, pos2)) +
          atoi(strncpy(buffer, str+ pos2 +ONE, pos3));

  return OK;
}

int file_size(int sockfd, char *str, char *filename,
              int *filesize){

  send_to(sockfd, str, STR_SIZE, filename);
  receive_from(sockfd, str, CODE_SIZE);

  *filesize = atoi(&str[FOUR]);

  return OK;
}

int retrieve(int sockfd, char *str, char *filename){

  send_to(sockfd, str, STR_RETR, filename);

  return OK;

}

int quit(int sockfd, char *str){

  receive_from(sockfd, str, CODE_QUIT);
  send_to(sockfd, str, STR_QUIT, BLANK);
```

```
  return OK;

}

int retrieve_file(int sockfd, char *filename,
                  int filesize){

  FILE *file = fopen(filename, WRITE_BINARY);

  int data_size = FILE_DATA_SIZE;
  char *str = malloc(FILE_DATA_SIZE + ONE);

  while(filesize > ZERO){
    if(filesize < FILE_DATA_SIZE) data_size = filesize;

    bzero(str, FILE_DATA_SIZE);

    test_receive(sockfd, str);

    fwrite(str, ONE, data_size, file);

    filesize = filesize - data_size;

    usleep(FILE_SLEEP);
  }

  fclose(file);

  return OK;

}

int ftp_valid(char *message){

  if(strstr(message, FTP_CODE_NO_FILE) != NULL)
    exit(ABORT);
  else if(strstr(message, FTP_CODE_WRONG_CREDENTIALS)
          != NULL)
    exit(ABORT);
```

```
  else
    return OK;

}

int ftp_init(int sockfd, FTP_Data data, int *retr_port,
             int *filesize){

  char *str = malloc(MAX_SIZE_WITH_NULL);

  login(sockfd, str, data.user, data.password);

  path(sockfd, str, data.url_path);

  passive_mode(sockfd, str);

  port(str, retr_port);

  file_size(sockfd, str, data.filename, filesize);

  return OK;

}

int ftp_transfer(int sockfd, char *ip, int retr_port,
                 char *filename, int filesize){

  int pid = fork();
  char *str = malloc(MAX_SIZE_WITH_NULL);

  if(!pid){

    retrieve(sockfd, str, filename);
    sockfd = asocket(ip, retr_port);
    retrieve_file(sockfd, filename, filesize);

  } else
    quit(sockfd, str);

  return OK;
```

```
}

int ftp_quit(int sockfd){

  usleep(FTP_SLEEP);
  close(sockfd);

  return OK;

}

int ftp(FTP_Data data){

  char *ip;
  int sockfd;
  int retr_port;
  int filesize;

  ip = getIP(data.host);

  sockfd = asocket(ip, data.port);

  ftp_init(sockfd, data, &retr_port, &filesize);

  ftp_transfer(sockfd, ip, retr_port, data.filename,
               filesize);

  ftp_quit(sockfd);

  return OK;

}
```

ftpdata.h

Descrição: header file da estrutura `FTP_Data`.

```
#ifndef _FTP_DATA_H_
#define _FTP_DATA_H_

#include "defines.h"

/**
Struct to save all info related to FTP
Includes saving:
User and Password (OPTIONAL)
Host and UrlPath and Filename (NOT OPTIONAL)
Port (OPTIONAL)
*/

typedef struct {
        char *user, *password;
        char *host, *url_path, *filename;
        int port;
} FTP_Data;

/**
@brief Allocates memory for variables
@param data - data
*/

void init(FTP_Data *data);

/**
@brief Sets the default settings
(user, password and port)
@param data - data
*/

void set_default(FTP_Data *data);

/**
@brief Sets all data
@param data - data
```

```
@param user - username
@param password - user's password
@param host - hostname
@param url_path - url path
@param filename - file name
@param port - port
*/

void set_all(FTP_Data *data, char *user,
             char *password, char *host,
             char *url_path, char *filename);

/**
@brief Parses the data from arguments received
@param arg - argv[1]
@param data - data
@return Returns OK (0) if success
*/

int parse_data(char *arg, FTP_Data *data);

#endif
```

ftpdata.c

Descrição: funções relacionadas com a estrutura.

```
#include "ftp_data.h"

void init(FTP_Data *data){

  data->user = malloc(MAX_SIZE_WITH_NULL);
  data->password = malloc(MAX_SIZE_WITH_NULL);
  data->host = malloc(MAX_SIZE_WITH_NULL);
  data->url_path = malloc(MAX_SIZE_WITH_NULL);
  data->filename = malloc(MAX_SIZE_WITH_NULL);

}

void set_default(FTP_Data *data){

  strcpy(data->user, USER_ANONYMOUS);
  strcpy(data->password, PASSWORD_ANONYMOUS);
  data->port = FTP_PORT;

}

void set_all(FTP_Data *data, char *user,
             char *password, char *host,
             char *url_path, char *filename){

  strcpy(data->user, user);
  strcpy(data->password, password);
  strcpy(data->host, host);
  strcpy(data->url_path, url_path);
  strcpy(data->filename, filename);

}

int parse_data(char *arg, FTP_Data *data){

  int tmp_final;
  int colon_pos, at_pos = ERROR, slash_pos,
      final_slash_pos;
```

```
init(data);
set_default(data);

colon_pos = find_nth(arg, COLON, SIX, ONE);
slash_pos = find_nth(arg, SLASH, SIX, ONE);

if(colon_pos != NOT_FOUND){
  at_pos = find_nth(arg, AT, colon_pos
                              + ONE, ONE);

  if(at_pos != NOT_FOUND){
    data->user = str_cpy(arg, SIX, colon_pos);
    data->password = str_cpy(arg, colon_pos + ONE,
                              at_pos);
    colon_pos = find_nth(arg, COLON, at_pos + ONE,
                         ONE);
  }

  if(colon_pos != NOT_FOUND)
    data->port = atoi(str_cpy(arg, colon_pos + ONE,
                              slash_pos));
}


if(at_pos == NOT_FOUND) at_pos = FIVE;

if(colon_pos == NOT_FOUND) colon_pos = slash_pos;

final_slash_pos = slash_pos + ONE;
tmp_final = find_nth(arg, SLASH, final_slash_pos,
                     ONE);

while(tmp_final != NOT_FOUND){
  final_slash_pos = tmp_final;
  tmp_final = find_nth(arg, SLASH, tmp_final + ONE,
                       ONE);
}

data->url_path = str_cpy(arg, slash_pos + ONE,
```

```
                                final_slash_pos);
  data->host = str_cpy(arg, at_pos + ONE, colon_pos);

  if(final_slash_pos == slash_pos + ONE)
    final_slash_pos--;

  data->filename = str_cpy(arg, final_slash_pos + ONE,
                                strlen(arg));

  return OK;
}
```

main.c

Descrição: função `main`.

```
#include <stdio.h>
#include "defines.h"
#include "ftp_data.h"

/**
@brief Checks if the only argument that exists is the
FTP request
@param argc - number of parameters
@param argv - parameters value
@return Returns OK (0) if success
*/

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

  if (argc != TWO) {
    printf("Program: ./download ftp://[<user>:
            <password>@]<host>[:port]/
            <url-path>\n");
    exit(ABORT);
  } else if(argv[ONE] == NULL){
      printf("[ERROR]: Argument is missing\n");
      exit(ABORT);
  } else if(strncmp(argv[ONE], FTP_START, SIX)!=ZERO){
      printf("[ERROR]: It should start by ftp://\n");
      exit(ABORT);
  }

  return OK;
}

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

  test_args(argc, argv);

  FTP_Data data;

  parse_data(argv[ONE], &data);
```

```
    ftp(data);

    return OK;
}
```

message.h

Descrição: header file de message.

```
#ifndef _MESSAGE_H_
#define _MESSAGE_H_

#include "defines.h"
#include "test.h"

/**
@brief Listens to any message that comes from server
@param sockfd - socket file descriptor
@param str - string to get read message
@return Returns 0 if success
*/

int listen_to(int sockfd, char *str);

/**
@brief Sends info to server (USER | PASS | etc...)
@param sockfd - socket file descriptor
@param str - string to be sent
@param code_str - USER | PASS | PASV | etc...
@param value - value
@return Returns 0 if success
*/

int send_to(int sockfd, char *str, char *code_str,
            char *value);

/**
@brief Awaits the response from server with an
expected code
@param sockfd - socket file descriptor
@param str - string received (from listen_to)
@param code_str - expected code
@return Returns 0 when succeeds
*/

int receive_from(int sockfd,char *str,char *code_str);
```

```
#endif
```

message.c

Descrição: trata de enviar os comandos e receber a resposta do servidor.

```
#include "message.h"

int listen_to(int sockfd, char *str){

  bzero(str, MAX_SIZE);
  test_receive(sockfd, str);

  printf("%s\n", str);
  ftp_valid(str);

  return OK;
}

int send_to(int sockfd, char *str, char *code_str,
            char *value){

  bzero(str, MAX_SIZE);

  strcpy(str, code_str);
  strcat(str, value);
  strcat(str, NEWLINE_STRING);

  if(send(sockfd, str, strlen(str), ZERO) < ZERO)
    printf("[ERROR]: Socket problem (sending...):");

  return OK;
}

int receive_from(int sockfd,char *str,char *code_str){

  for(; !test_response(str, code_str);)
    listen_to(sockfd, str);
```

```
    return OK;
}
```

test.h

Descrição: header file de test.

```
#ifndef _TEST_H_
#define _TEST_H_

#include <errno.h>
#include "defines.h"

/**
@brief Tests if it could receive anything
@param sockfd - socket file descriptor
@param str - buffer string
@return Returns OK (0) if success
*/

int test_receive(int sockfd, char *str);

/**
@brief Checks if the message has the expected code
@param str - string received (message)
@param code_str - expected code
@return Returns ONE (1 -> true) if success
(code exists) and 0 (false) otherwise
*/

int test_response(char *str, char *code_str);

#endif
```

test.c

Descrição: verifica se a mensagem contem o código expectável e se consegue receber algo.

```
#include "test.h"

int test_receive(int sockfd, char *str){

  if(recv(sockfd, str, MAX_SIZE, ZERO) < ZERO){
```

```
    printf("[ERROR]: %s", strerror(errno));
    return !OK;
  }

  return OK;

}

int test_response(char *str, char *code_str){
  int i, str_pos = ZERO;
  int length = strlen(code_str), position = ZERO;

  for(; str_pos != ERROR;){
    if(str_pos != ERROR){
      if(str_pos != ZERO) str_pos++;

      for(i = ZERO; i < length; i++)
        if(str[str_pos + i] != code_str[i])
          break;

        if(length == i) return ONE;
    }
    str_pos = find_nth(str, NEWLINE, str_pos+ONE, ONE);
  }

  return ZERO;
}
```

utils.h

Descrição: header file de utils.

```
#ifndef _UTILS_H_
#define _UTILS_H_

#include "defines.h"

/**
@brief Finds the nth element of a string
@param str - string
@param c - character
@param start - starting position
@param nth - nth position of character
@return Returns the position or NOT_FOUND otherwise
*/

int find_nth(char *str, char c, int start, int nth);

/**
@brief Copies a string orig to dest, from start to end
@param orig - origin string
@param start - starting position
@param end - ending position
@return Returns the destination string
*/

char* str_cpy(char *orig, int start, int end);

#endif
```

utils.c

Descrição: funções utilitárias.

```
#include "utils.h"

int find_nth(char *str, char c, int start, int nth){

  if(start >= strlen(str))
    return NOT_FOUND;

  int i = start;

  while(str[i] != NULL_CHAR){

    if(str[i] == c){
      if(nth == ONE) return i;
      else nth--;
    }
    i++;
  }

  return NOT_FOUND;
}

char* str_cpy(char *orig, int start, int end){

  int i = start, j;
  char *dest = malloc(end - start + ONE);

  for(j = ZERO; i < end; i++, j++)
    dest[j] = orig[i];

  dest[j] = NULL_CHAR;

  return dest;

}
```

Comandos de Configuração

Experiência 1

```
tux61 >> ifconfig eth0 172.16.60.1/24
tux64 >> ifconfig eth0 172.16.60.254/24
```

Experiência 2

```
tux61 >> ifconfig eth0 172.16.60.1/24
tux64 >> ifconfig eth0 172.16.60.254/24
tux62 >> ifconfig eth1 172.16.61.1/24
tux64 >> echo 0 > /proc/sys/net/ipv4/
                  icmp_echo_ignore_broadcasts

>> configure terminal
>> vlan 60
>> vlan 61
>> interface fastethernet 0/1
>> switchport mode access
>> switchport access vlan 60
>> interface fastethernet 0/4
>> switchport mode access
>> switchport access vlan 60
>> interface fastethernet 0/2
>> switchport mode access
>> switchport access vlan 61
>> end
```

Experiência 3

```
tux61 >> ifconfig eth0 172.16.60.1/24
tux61 >> route add default gateway 172.16.60.254/24
tux62 >> ifconfig eth1 172.16.61.1/24
tux62 >> route add default gateway 172.16.61.253/24
tux64 >> ifconfig eth0 172.16.60.254/24
tux64 >> ifconfig eth1 172.16.61.253/24
tux64 >> echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
>> configure terminal
>> vlan 60
>> vlan 61
>> interface fastethernet 0/1
>> switchport mode access
>> switchport access vlan 60
>> interface fastethernet 0/4
>> switchport mode access
>>switchport access vlan 60
>> interface fastethernet 0/3
>> switchport mode access
>> switchport access vlan 61
>> interface fastethernet 0/2
>> switchport mode access
>> switchport access vlan 61
>> end
```

Experiência 4

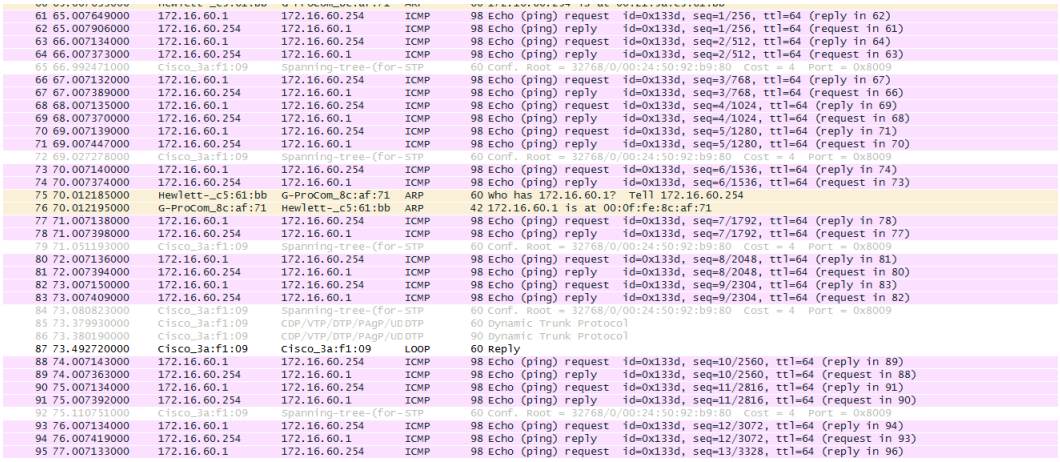
```
tux61 >> ifconfig eth0 172.16.60.1/24
tux61 >> route add default gateway 172.16.60.254/24
tux64 >> ifconfig eth0 172.16.60.254/24
tux64 >> ifconfig eth1 172.16.61.253/24
tux64 >> route add default gateway 172.16.61.254/24
tux62 >> ifconfig eth1 172.16.61.1/24
tux62 >> route add default gateway 172.16.21.253/24
tux64 >> echo 0 > /proc/sys/net/ipv4/
                  icmp_echo_ignore_broadcasts
tux64 >> echo 1 > /proc/sys/net/ipv4/ip_forward

>> configure terminal
>> vlan 60
>> vlan 61
>> interface fastethernet 0/1
>> switchport mode access
>> switchport access vlan 60
>> interface fastethernet 0/4
>> switchport mode access
```

```
>>switchport access vlan 60
>> interface fastethernet 0/3
>> switchport mode access
>> switchport access vlan 61
>> interface fastethernet 0/2
>> switchport mode access
>> switchport access vlan 61
>> interface fastethernet 0/5
>> switchport mode access
>> switchport access vlan 61
>> end

>> configure terminal
>> interface gigabitethernet 0/0
>> ip address 172.16.61.254  255.255.255.0
>> no shutdown
>> exit
>> interface gigabitethernet 0/1
>> ip address 172.16.1.69  255.255.255.0
>> no shutdown
>> exit
>> ip route 0.0.0.0 0.0.0.0 172.16.1.254
>> ip route 172.16.60.0 255.255.255.0 172.16.61.253
>> end

>> configure terminal
>> interface gigabitethernet 0/0
>> ip address 172.16.61.254 255.255.255.0
>> no shutdown
>> ip nat inside
>> exit
>> interface gigabitethernet 0/1
>> ip address 172.16.2.69 255.255.255.0
>> no shutdown
>> ip nat outside
>> exit
>> ip nat pool ovrld 172.16.1.69 172.16.1.69 prefix 24
>> ip nat inside source list 1 pool ovrld overload
>> access-list 1 permit 172.66.20.0 0.0.0.255
>> access-list 1 permit 172.66.21.0 0.0.0.255
```

```
>> ip route 0.0.0.0 0.0.0.0 172.16.1.254
>> ip route 172.16.20.0 255.255.255.0 172.16.61.253
>> end
```

Logs

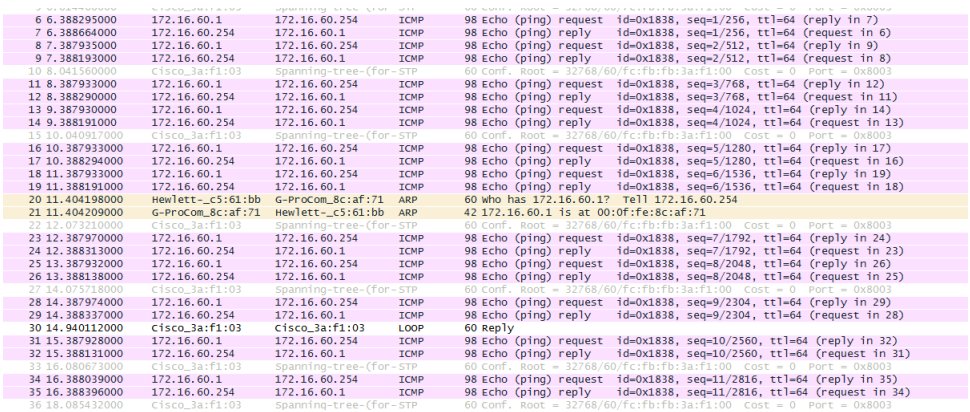
Exp1: tux61 -> tux64



61	65.007649000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x133d, seq=1/256, ttl=64 (reply in 62)
62	65.007906000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x133d, seq=1/256, ttl=64 (request in 61)
63	66.007134000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x133d, seq=2/512, ttl=64 (reply in 64)
64	66.007373000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x133d, seq=2/512, ttl=64 (request in 63)
65	66.992471000	Cisco_3a:f1:09	Spanning-tree-(for-STP		60 Conf. Root = 32768/0/00:24:50:92:b9:80 Cost = 4 Port = 0x8009
66	67.007132000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x133d, seq=3/768, ttl=64 (reply in 67)
67	67.007389000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x133d, seq=3/768, ttl=64 (request in 66)
68	68.007135000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x133d, seq=4/1024, ttl=64 (reply in 69)
69	68.007370000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x133d, seq=4/1024, ttl=64 (request in 68)
70	69.007139000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x133d, seq=5/1280, ttl=64 (reply in 71)
71	69.007447000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x133d, seq=5/1280, ttl=64 (request in 70)
72	69.027278000	Cisco_3a:f1:09	Spanning-tree-(for-STP		60 Conf. Root = 32768/0/00:24:50:92:b9:80 Cost = 4 Port = 0x8009
73	70.007140000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x133d, seq=6/1536, ttl=64 (reply in 74)
74	70.007374000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x133d, seq=6/1536, ttl=64 (request in 73)
75	70.012185000	Hewlett-_c5:61:bb	G-ProCom_8c:af:71	ARP	60 Who has 172.16.60.1? Tell 172.16.60.254
76	70.012195000	G-ProCom_8c:af:71	Hewlett-_c5:61:bb	ARP	42 172.16.60.1 is at 00:0f:fe:8c:af:71
77	71.007138000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x133d, seq=7/1792, ttl=64 (reply in 78)
78	71.007398000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x133d, seq=7/1792, ttl=64 (request in 77)
79	71.051193000	Cisco_3a:f1:09	Spanning-tree-(for-STP		60 Conf. Root = 32768/0/00:24:50:92:b9:80 Cost = 4 Port = 0x8009
80	72.007136000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x133d, seq=8/2048, ttl=64 (reply in 81)
81	72.007394000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x133d, seq=8/2048, ttl=64 (request in 80)
82	73.007150000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x133d, seq=9/2304, ttl=64 (reply in 83)
83	73.007409000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x133d, seq=9/2304, ttl=64 (request in 82)
84	73.080823000	Cisco_3a:f1:09	Spanning-tree-(for-STP		60 Conf. Root = 32768/0/00:24:50:92:b9:80 Cost = 4 Port = 0x8009
85	73.379930000	Cisco_3a:f1:09	CDP/VTP/DTP/PAgP/UDDTP		60 Dynamic Trunk Protocol
86	73.380190000	Cisco_3a:f1:09	CDP/VTP/DTP/PAgP/UDDTP		90 Dynamic Trunk Protocol
87	73.492720000	Cisco_3a:f1:09	Cisco_3a:f1:09	LOOP	60 Reply
88	74.007143000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x133d, seq=10/2560, ttl=64 (reply in 89)
89	74.007363000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x133d, seq=10/2560, ttl=64 (request in 88)
90	75.007134000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x133d, seq=11/2816, ttl=64 (reply in 91)
91	75.007392000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x133d, seq=11/2816, ttl=64 (request in 90)
92	75.110751000	Cisco_3a:f1:09	Spanning-tree-(for-STP		60 Conf. Root = 32768/0/00:24:50:92:b9:80 Cost = 4 Port = 0x8009
93	76.007134000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x133d, seq=12/3072, ttl=64 (reply in 94)
94	76.007419000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x133d, seq=12/3072, ttl=64 (request in 93)
95	77.007133000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x133d, seq=13/3328, ttl=64 (reply in 96)

Figura 2: `Ping` de tux61 para tux64 na experiência 1

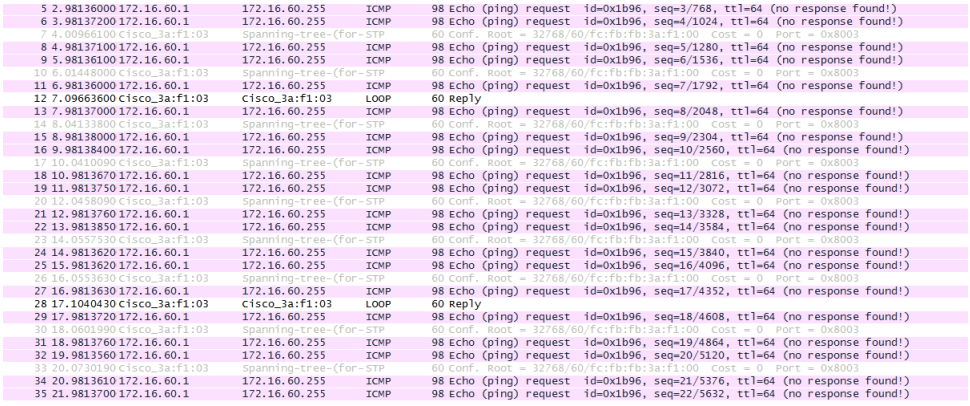
Exp2: tux61 -> tux64



6	6.388295000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x1838, seq=1/256, ttl=64 (reply in 7)
7	6.388664000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x1838, seq=1/256, ttl=64 (request in 6)
8	7.387935000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x1838, seq=2/512, ttl=64 (reply in 9)
9	7.388193000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x1838, seq=2/512, ttl=64 (request in 8)
10	8.041560000	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
11	8.387933000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x1838, seq=3/768, ttl=64 (reply in 12)
12	8.388290000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x1838, seq=3/768, ttl=64 (request in 11)
13	9.387930000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x1838, seq=4/1024, ttl=64 (reply in 14)
14	9.388191000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x1838, seq=4/1024, ttl=64 (request in 13)
15	10.040917000	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
16	10.387933000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x1838, seq=5/1280, ttl=64 (reply in 17)
17	10.388294000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x1838, seq=5/1280, ttl=64 (request in 16)
18	11.387933000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x1838, seq=6/1536, ttl=64 (reply in 19)
19	11.388191000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x1838, seq=6/1536, ttl=64 (request in 18)
20	11.404198000	Hewlett-_c5:61:bb	G-ProCom_8c:af:71	ARP	60 Who has 172.16.60.1? Tell 172.16.60.254
21	11.404209000	G-ProCom_8c:af:71	Hewlett-_c5:61:bb	ARP	42 172.16.60.1 is at 00:0f:fe:8c:af:71
22	12.073210000	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
23	12.387970000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x1838, seq=7/1792, ttl=64 (reply in 24)
24	12.388313000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x1838, seq=7/1792, ttl=64 (request in 23)
25	13.387932000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x1838, seq=8/2048, ttl=64 (reply in 26)
26	13.388138000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x1838, seq=8/2048, ttl=64 (request in 25)
27	14.057318000	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
28	14.387974000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x1838, seq=9/2304, ttl=64 (reply in 29)
29	14.388337000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x1838, seq=9/2304, ttl=64 (request in 28)
30	14.940112000	Cisco_3a:f1:03	Cisco_3a:f1:03	LOOP	60 Reply
31	15.387928000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x1838, seq=10/2560, ttl=64 (reply in 32)
32	15.388131000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x1838, seq=10/2560, ttl=64 (request in 31)
33	16.080673000	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
34	16.388039000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x1838, seq=11/2816, ttl=64 (reply in 35)
35	16.388396000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x1838, seq=11/2816, ttl=64 (request in 34)
36	18.085432000	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003

Figura 3: `Ping` de tux61 para tux64 na experiência 2

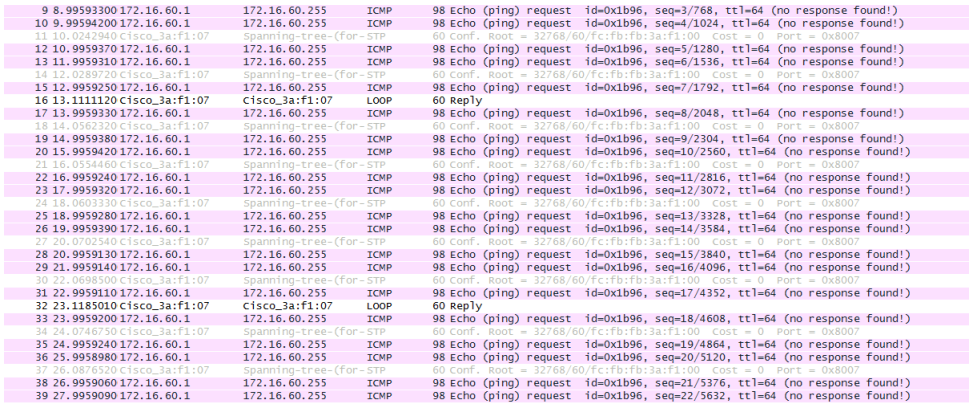
Exp2: tux61 -> 172.16.60.255 (vista de tux61)



5	2.98136000	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=3/768, ttl=64 (no response found!)
6	3.98137200	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=4/1024, ttl=64 (no response found!)
7	4.00966100	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
8	4.98137100	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=5/1280, ttl=64 (no response found!)
9	5.98136100	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=6/1536, ttl=64 (no response found!)
10	6.01448000	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
11	6.98136000	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=7/1792, ttl=64 (no response found!)
12	7.09663600	Cisco_3a:f1:03	Cisco_3a:f1:03	LOOP	60 Reply
13	7.98137000	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=8/2048, ttl=64 (no response found!)
14	8.04133600	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
15	8.98138000	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=9/2304, ttl=64 (no response found!)
16	9.98138400	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=10/2560, ttl=64 (no response found!)
17	10.0410090	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
18	10.9813670	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=11/2816, ttl=64 (no response found!)
19	11.9813750	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=12/3072, ttl=64 (no response found!)
20	12.0458090	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
21	12.9813760	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=13/3328, ttl=64 (no response found!)
22	13.9813850	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=14/3584, ttl=64 (no response found!)
23	14.0557530	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
24	14.9813620	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=15/3840, ttl=64 (no response found!)
25	15.9813620	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=16/4096, ttl=64 (no response found!)
26	16.0533630	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
27	16.9813630	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=17/4352, ttl=64 (no response found!)
28	17.1040430	Cisco_3a:f1:03	Cisco_3a:f1:03	LOOP	60 Reply
29	17.9813720	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=18/4608, ttl=64 (no response found!)
30	18.0601990	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
31	18.9813760	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=19/4864, ttl=64 (no response found!)
32	19.9813560	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=20/5120, ttl=64 (no response found!)
33	20.0730190	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
34	20.9813610	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=21/5376, ttl=64 (no response found!)
35	21.9813700	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=22/5632, ttl=64 (no response found!)

Figura 4: `Ping` *broadcast* de tux61 para 172.16.60.255 na experiência 2

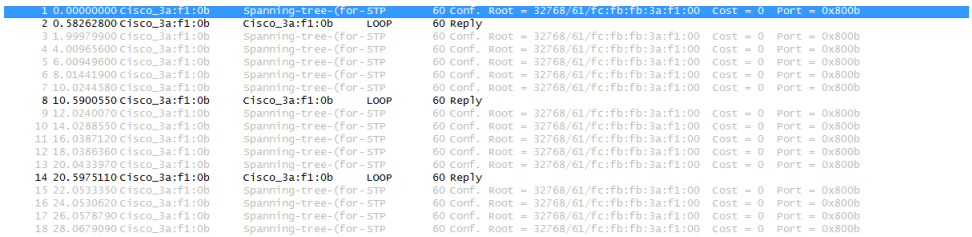
Exp2: tux61 -> 172.16.60.255 (vista de tux64)



9	8.99593300	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=3/768, ttl=64 (no response found!)
10	9.99594200	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=4/1024, ttl=64 (no response found!)
11	10.0242940	Cisco_3a:f1:07	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
12	10.9959370	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=5/1280, ttl=64 (no response found!)
13	11.9959310	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=6/1536, ttl=64 (no response found!)
14	12.0259720	Cisco_3a:f1:07	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
15	12.9959250	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=7/1792, ttl=64 (no response found!)
16	13.1111120	Cisco_3a:f1:07	Cisco_3a:f1:07	LOOP	60 Reply
17	13.9959330	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=8/2048, ttl=64 (no response found!)
18	14.0562320	Cisco_3a:f1:07	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
19	14.9959380	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=9/2304, ttl=64 (no response found!)
20	15.9959420	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=10/2560, ttl=64 (no response found!)
21	16.0554460	Cisco_3a:f1:07	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
22	16.9959240	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=11/2816, ttl=64 (no response found!)
23	17.9959320	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=12/3072, ttl=64 (no response found!)
24	18.0603330	Cisco_3a:f1:07	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
25	18.9959280	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=13/3328, ttl=64 (no response found!)
26	19.9959390	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=14/3584, ttl=64 (no response found!)
27	20.0762340	Cisco_3a:f1:07	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
28	20.9959130	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=15/3840, ttl=64 (no response found!)
29	21.9959140	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=16/4096, ttl=64 (no response found!)
30	22.0698500	Cisco_3a:f1:07	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
31	22.9959110	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=17/4352, ttl=64 (no response found!)
32	23.1185010	Cisco_3a:f1:07	Cisco_3a:f1:07	LOOP	60 Reply
33	23.9959200	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=18/4608, ttl=64 (no response found!)
34	24.0746750	Cisco_3a:f1:07	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
35	24.9959240	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=19/4864, ttl=64 (no response found!)
36	25.9958980	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=20/5120, ttl=64 (no response found!)
37	26.0876920	Cisco_3a:f1:07	Spanning-tree-(for-	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
38	26.9959060	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=21/5376, ttl=64 (no response found!)
39	27.9959090	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request id=0x1b96, seq=22/5632, ttl=64 (no response found!)

Figura 5: `Ping` *broadcast* de tux61 para 172.16.60.255 na experiência 2, na vista de tux64

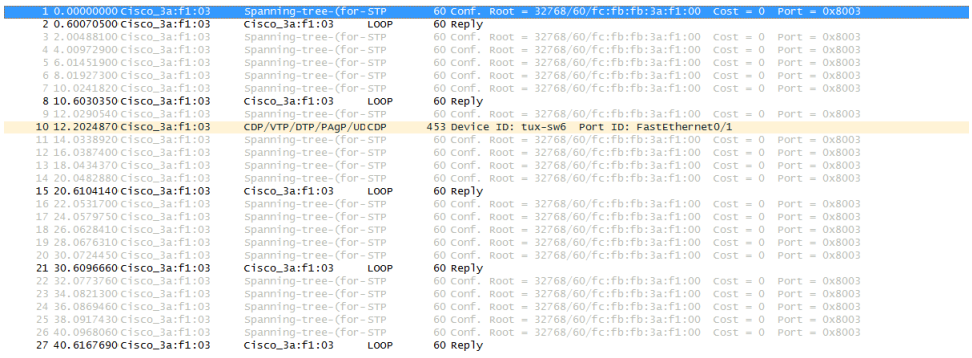
Exp2: tux61 -> 172.16.60.255 (vista de tux62)



1	0.00000000	Cisco_3a:f1:0b	Spanning-tree-(for-	STP	60 Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
2	0.58262800	Cisco_3a:f1:0b	Cisco_3a:f1:0b	LOOP	60 Reply
3	1.99979900	Cisco_3a:f1:0b	Spanning-tree-(for-	STP	60 Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
4	4.00965600	Cisco_3a:f1:0b	Spanning-tree-(for-	STP	60 Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
5	6.00949600	Cisco_3a:f1:0b	Spanning-tree-(for-	STP	60 Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
6	8.01441900	Cisco_3a:f1:0b	Spanning-tree-(for-	STP	60 Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
7	10.0244580	Cisco_3a:f1:0b	Spanning-tree-(for-	STP	60 Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
8	10.5900550	Cisco_3a:f1:0b	Cisco_3a:f1:0b	LOOP	60 Reply
9	12.0240070	Cisco_3a:f1:0b	Spanning-tree-(for-	STP	60 Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
10	14.0288550	Cisco_3a:f1:0b	Spanning-tree-(for-	STP	60 Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
11	16.0387120	Cisco_3a:f1:0b	Spanning-tree-(for-	STP	60 Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
12	18.0386360	Cisco_3a:f1:0b	Spanning-tree-(for-	STP	60 Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
13	20.0433970	Cisco_3a:f1:0b	Spanning-tree-(for-	STP	60 Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
14	20.5975110	Cisco_3a:f1:0b	Cisco_3a:f1:0b	LOOP	60 Reply
15	22.0533350	Cisco_3a:f1:0b	Spanning-tree-(for-	STP	60 Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
16	24.0530620	Cisco_3a:f1:0b	Spanning-tree-(for-	STP	60 Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
17	26.0578790	Cisco_3a:f1:0b	Spanning-tree-(for-	STP	60 Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
18	28.0679090	Cisco_3a:f1:0b	Spanning-tree-(for-	STP	60 Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b

Figura 6: `Ping` *broadcast* de tux61 para 172.16.60.255 na experiência 2, na vista de tux62

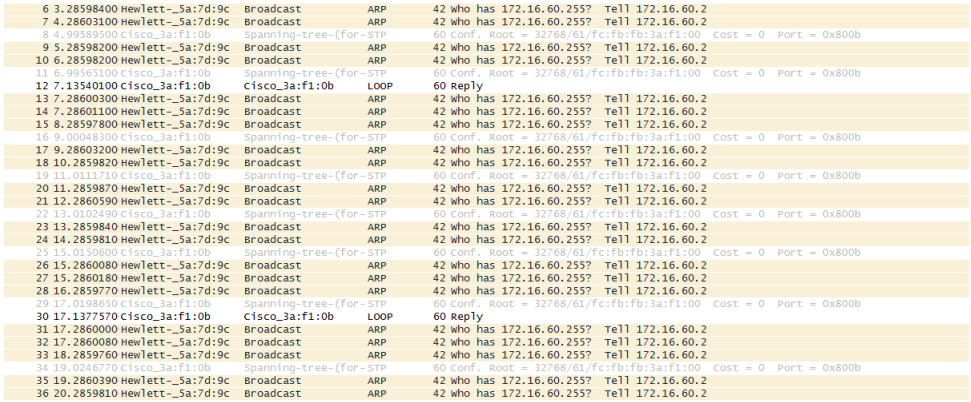
Exp2: tux62 -> 172.16.61.255 (vista de tux61)



1	0.00000000	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
2	0.60070500	Cisco_3a:f1:03	Cisco_3a:f1:03 LOOP	60	Reply
3	2.00488100	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
4	4.00972900	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
5	6.01451900	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
6	8.01927300	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
7	10.0241820	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
8	10.6030350	Cisco_3a:f1:03	Cisco_3a:f1:03 LOOP	60	Reply
9	12.0290540	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
10	12.2024870	Cisco_3a:f1:03	CDP/VTP/DTP/PAgP/UDCDP	453	Device ID: tux-sw6 Port ID: FastEthernet0/1
11	14.0338920	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
12	16.0387400	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
13	18.0434370	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
14	20.0482880	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
15	20.6104140	Cisco_3a:f1:03	Cisco_3a:f1:03 LOOP	60	Reply
16	22.0531700	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
17	24.0579750	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
18	26.0628410	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
19	28.0676310	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
20	30.0724450	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
21	30.6096660	Cisco_3a:f1:03	Cisco_3a:f1:03 LOOP	60	Reply
22	32.0773760	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
23	34.0821300	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
24	36.0869460	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
25	38.0917430	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
26	40.0968060	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
27	40.6167690	Cisco_3a:f1:03	Cisco_3a:f1:03 LOOP	60	Reply

Figura 7: `Ping` *broadcast* de tux62 para 172.16.61.255 na experiência 2, na vista de tux61

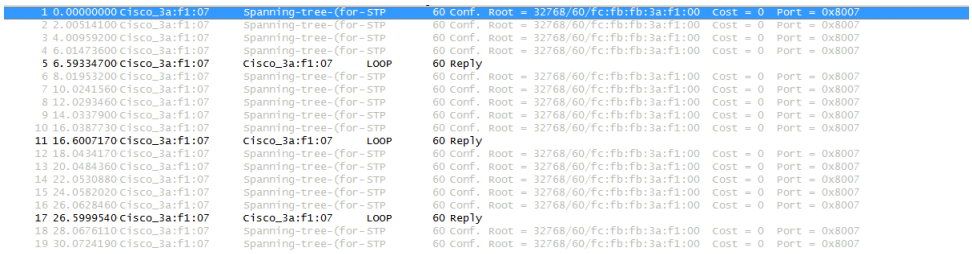
Exp2: tux62 -> 172.16.61.255 (vista de tux62)



6	3.28598400	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
7	4.28603100	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
8	4.99589500	Cisco_3a:f1:0b	Spanning-tree-(for-STP	60	Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
9	5.28598200	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
10	6.28598200	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
11	6.99565100	Cisco_3a:f1:0b	Spanning-tree-(for-STP	60	Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
12	7.13540100	Cisco_3a:f1:0b	Cisco_3a:f1:0b LOOP	60	Reply
13	7.28600300	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
14	7.28601100	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
15	8.28597800	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
16	9.00048300	Cisco_3a:f1:0b	Spanning-tree-(for-STP	60	Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
17	9.28603200	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
18	10.2859820	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
19	11.0011810	Cisco_3a:f1:0b	Spanning-tree-(for-STP	60	Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
20	11.2859870	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
21	12.2860590	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
22	13.0102490	Cisco_3a:f1:0b	Spanning-tree-(for-STP	60	Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
23	13.2859840	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
24	14.2859810	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
25	15.0150600	Cisco_3a:f1:0b	Spanning-tree-(for-STP	60	Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
26	15.2860080	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
27	15.2860180	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
28	16.2859770	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
29	17.0198650	Cisco_3a:f1:0b	Spanning-tree-(for-STP	60	Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
30	17.1377570	Cisco_3a:f1:0b	Cisco_3a:f1:0b LOOP	60	Reply
31	17.2860000	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
32	17.2860080	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
33	18.2859760	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
34	19.0246770	Cisco_3a:f1:0b	Spanning-tree-(for-STP	60	Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x800b
35	19.2860390	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2
36	20.2859810	Hewlett-_5a:7d:9c	Broadcast ARP	42	who has 172.16.60.255? Tell 172.16.60.2

Figura 8: `Ping` *broadcast* de tux62 para 172.16.61.255 na experiência 2, na vista de tux62

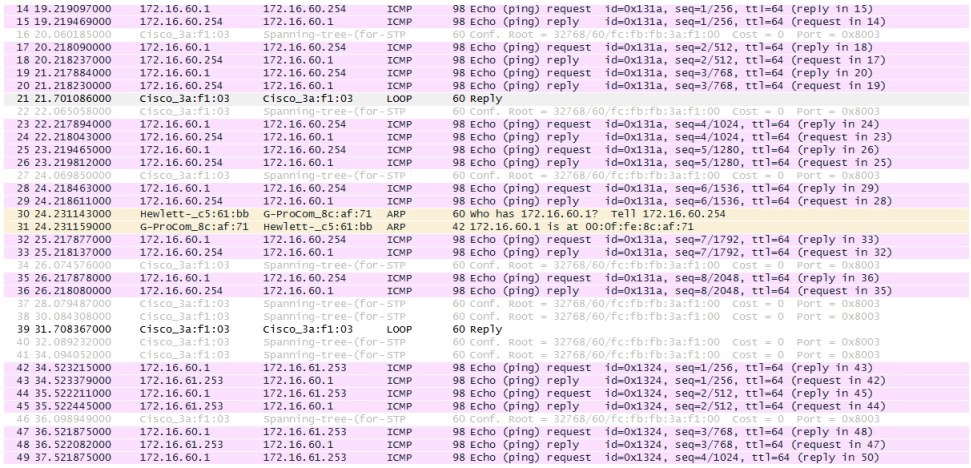
Exp2: tux62 -> 172.16.61.255 (vista de tux64)



1	0.00000000	Cisco_3a:f1:07	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
2	2.00514100	Cisco_3a:f1:07	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
3	4.00959200	Cisco_3a:f1:07	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
4	6.01473600	Cisco_3a:f1:07	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
5	6.59334700	Cisco_3a:f1:07	Cisco_3a:f1:07 LOOP	60	Reply
6	8.01953200	Cisco_3a:f1:07	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
7	10.0241560	Cisco_3a:f1:07	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
8	12.0293460	Cisco_3a:f1:07	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
9	14.0337900	Cisco_3a:f1:07	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
10	16.0387730	Cisco_3a:f1:07	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
11	16.6007170	Cisco_3a:f1:07	Cisco_3a:f1:07 LOOP	60	Reply
12	18.0434170	Cisco_3a:f1:07	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
13	20.0484360	Cisco_3a:f1:07	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
14	22.0530880	Cisco_3a:f1:07	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
15	24.0582020	Cisco_3a:f1:07	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
16	26.0628460	Cisco_3a:f1:07	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
17	26.5999540	Cisco_3a:f1:07	Cisco_3a:f1:07 LOOP	60	Reply
18	28.0676110	Cisco_3a:f1:07	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007
19	30.0724190	Cisco_3a:f1:07	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8007

Figura 9: `Ping` *broadcast* de tux62 para 172.16.61.255 na experiência 2, na vista de tux64

Exp3: tux61 -> tux64(eth0 e eth1) e tux62



14	19.219097000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x131a, seq=1/256, ttl=64 (reply in 15)
15	19.219469000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x131a, seq=1/256, ttl=64 (request in 14)
16	20.060183000	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
17	20.218090000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x131a, seq=2/512, ttl=64 (reply in 18)
18	20.218237000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x131a, seq=2/512, ttl=64 (request in 17)
19	21.217884000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x131a, seq=3/768, ttl=64 (reply in 20)
20	21.218230000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x131a, seq=3/768, ttl=64 (request in 19)
21	21.701086000	Cisco_3a:f1:03	Cisco_3a:f1:03	LOOP	60 Reply
22	22.065058000	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
23	22.217894000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x131a, seq=4/1024, ttl=64 (reply in 24)
24	22.218043000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x131a, seq=4/1024, ttl=64 (request in 23)
25	23.219465000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x131a, seq=5/1280, ttl=64 (reply in 26)
26	23.219812000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x131a, seq=5/1280, ttl=64 (request in 25)
27	24.068430000	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
28	24.218463000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x131a, seq=6/1536, ttl=64 (reply in 29)
29	24.218611000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x131a, seq=6/1536, ttl=64 (request in 28)
30	24.231143000	Hewlett-_c5:61:bb	G-ProCom_8c:af:71	ARP	60 Who has 172.16.60.1? Tell 172.16.60.254
31	24.231159000	G-ProCom_8c:af:71	Hewlett-_c5:61:bb	ARP	42 172.16.60.1 is at 00:0f:fe:8c:af:71
32	25.217877000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x131a, seq=7/1792, ttl=64 (reply in 33)
33	25.218137000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x131a, seq=7/1792, ttl=64 (request in 32)
34	26.074576000	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
35	26.217878000	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request id=0x131a, seq=8/2048, ttl=64 (reply in 36)
36	26.218080000	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x131a, seq=8/2048, ttl=64 (request in 35)
37	28.079487000	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
38	30.084308000	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
39	31.708367000	Cisco_3a:f1:03	Cisco_3a:f1:03	LOOP	60 Reply
40	32.089232000	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
41	34.094052000	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
42	34.523215000	172.16.60.1	172.16.61.253	ICMP	98 Echo (ping) request id=0x1324, seq=1/256, ttl=64 (reply in 43)
43	34.523379000	172.16.61.253	172.16.60.1	ICMP	98 Echo (ping) reply id=0x1324, seq=1/256, ttl=64 (request in 42)
44	35.522211000	172.16.60.1	172.16.61.253	ICMP	98 Echo (ping) request id=0x1324, seq=2/512, ttl=64 (reply in 45)
45	35.522445000	172.16.61.253	172.16.60.1	ICMP	98 Echo (ping) reply id=0x1324, seq=2/512, ttl=64 (request in 44)
46	36.098949000	Cisco_3a:f1:03	Spanning-tree-(for-STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
47	36.521875000	172.16.60.1	172.16.61.253	ICMP	98 Echo (ping) request id=0x1324, seq=3/768, ttl=64 (reply in 48)
48	36.522082000	172.16.61.253	172.16.60.1	ICMP	98 Echo (ping) reply id=0x1324, seq=3/768, ttl=64 (request in 47)
49	37.521875000	172.16.60.1	172.16.61.253	ICMP	98 Echo (ping) request id=0x1324, seq=4/1024, ttl=64 (reply in 50)

Figura 10: `Ping` de tux61 para tux64 e tux62 na experiência 3

Exp3: tux61 -> tux64(eth0 e eth1) e tux62 (continuação)

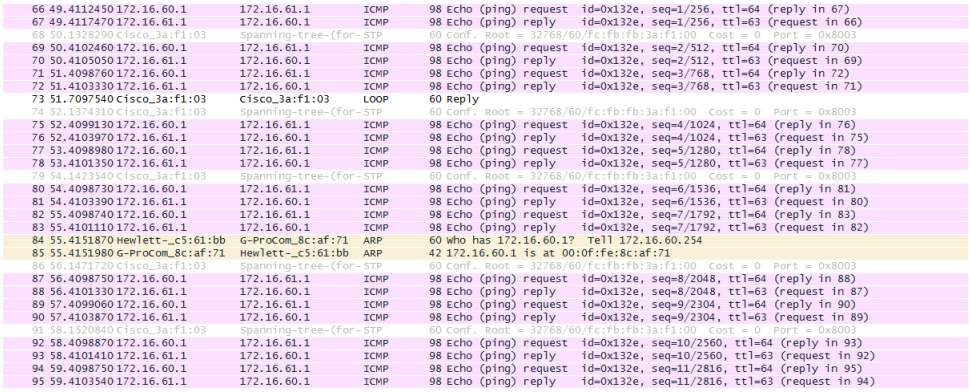


Figura 11: `Ping` de tux61 para tux64 e tux62 na experiência 3 (continuação)

Exp3: tux61 -> tux62 (vista de tux64 eth0)

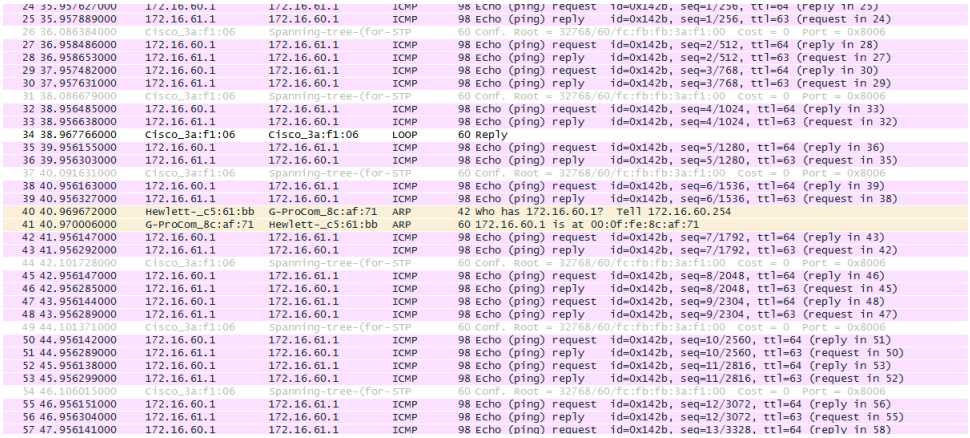


Figura 12: `Ping` de tux61 para tux62 na vista de tux64 eth0

Exp3: tux61 -> tux62 (vista de tux64 eth1)

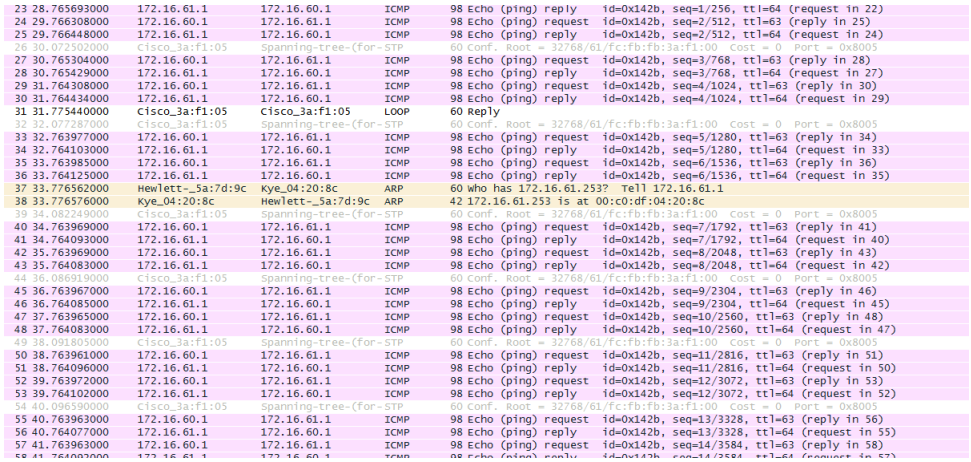


Figura 13: `Ping` de tux61 para tux62 na vista de tux64 eth1

Exp4: tux61 -> tux64 (eth0 e eth1)

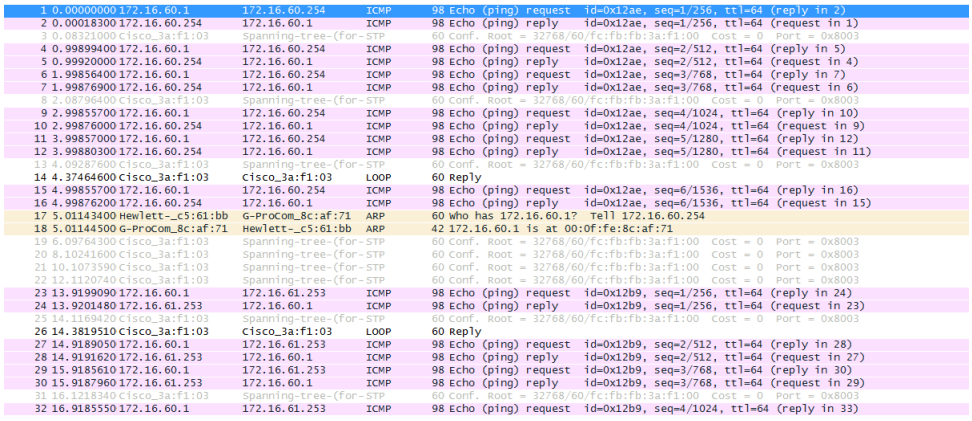


Figura 14: `Ping` de tux61 para tux64 (eth0 e eth1)

Exp4: tux61 -> tux62

38	18.9187620	172.16.61.253	172.16.60.1	ICMP	98	Echo (ping) reply id=0x12b9, seq=6/1536, ttl=64 (request in 37)
39	20.1314380	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
40	21.8681360	Cisco_3a:f1:03	CDP/VTP/DTP/PAgP/UD	CDP	453	Device ID: tux-sw6 Port ID: FastEthernet0/1
41	22.1362200	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
42	23.4478930	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x12c0, seq=1/256, ttl=64 (reply in 43)
43	23.4484010	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x12c0, seq=1/256, ttl=63 (request in 42)
44	24.1414090	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
45	24.3844270	Cisco_3a:f1:03	Cisco_3a:f1:03	LOOP	60	Reply
46	24.4468930	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x12c0, seq=2/512, ttl=64 (reply in 47)
47	24.4471560	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x12c0, seq=2/512, ttl=63 (request in 46)
48	25.4465600	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x12c0, seq=3/768, ttl=64 (reply in 49)
49	25.4468080	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x12c0, seq=3/768, ttl=63 (request in 48)
50	26.1537250	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
51	26.4465820	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x12c0, seq=4/1024, ttl=64 (reply in 52)
52	26.4470760	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x12c0, seq=4/1024, ttl=63 (request in 51)
53	27.4465590	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x12c0, seq=5/1280, ttl=64 (reply in 54)
54	27.4468080	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x12c0, seq=5/1280, ttl=63 (request in 53)
55	28.1780550	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
56	28.4465540	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x12c0, seq=6/1536, ttl=64 (reply in 57)
57	28.4470100	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x12c0, seq=6/1536, ttl=63 (request in 56)
58	28.4514730	Hewlett-_c5:61:bb	G-ProCom_8c:af:71	ARP	60	Who has 172.16.60.1? Tell 172.16.60.254
59	28.4514850	G-ProCom_8c:af:71	Hewlett-_c5:61:bb	ARP	42	172.16.60.1 is at 00:0f:fe:8c:af:71
60	29.4465630	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x12c0, seq=7/1792, ttl=64 (reply in 61)
61	29.4470120	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x12c0, seq=7/1792, ttl=63 (request in 60)
62	30.1807810	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
63	30.4465530	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x12c0, seq=8/2048, ttl=64 (reply in 64)
64	30.4469980	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x12c0, seq=8/2048, ttl=63 (request in 63)
65	32.1855400	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
66	34.1904100	Cisco_3a:f1:03	Spanning-tree-(for-	STP	60	Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8003
67	34.2079540	172.16.60.1	172.16.61.254	ICMP	98	Echo (ping) request id=0x12c7, seq=1/256, ttl=64 (reply in 68)
68	34.2086840	172.16.61.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x12c7, seq=1/256, ttl=254 (request in 67)

Figura 15: `Ping` de tux61 para tux62

Exp4: tux61 -> tux62 (vista de tux62)

6	6.038987000	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1523, seq=1/256, ttl=63 (no response found!)
7	7.047766000	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1523, seq=2/512, ttl=63 (no response found!)
8	8.019452000	Cisco_3a:f1:05	Spanning-tree-(for-	STP	60	Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8005
9	8.055782000	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1523, seq=3/768, ttl=63 (no response found!)
10	9.063749000	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1523, seq=4/1024, ttl=63 (no response found!)
11	10.024174000	Cisco_3a:f1:05	Spanning-tree-(for-	STP	60	Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8005
12	10.071748000	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1523, seq=5/1280, ttl=63 (no response found!)
13	11.051957000	Kye_04:20:8c	Hewlett-_5a:7d:9c	ARP	60	Who has 172.16.61.1? Tell 172.16.61.253
14	11.051973000	Hewlett-_5a:7d:9c	Kye_04:20:8c	ARP	42	172.16.61.1 is at 00:21:5a:5a:7d:9c
15	11.079741000	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1523, seq=6/1536, ttl=63 (no response found!)
16	12.034119000	Cisco_3a:f1:05	Spanning-tree-(for-	STP	60	Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8005
17	12.087791000	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1523, seq=7/1792, ttl=63 (no response found!)
18	13.095775000	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1523, seq=8/2048, ttl=63 (no response found!)
19	14.033950000	Cisco_3a:f1:05	Spanning-tree-(for-	STP	60	Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8005
20	14.103781000	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1523, seq=9/2304, ttl=63 (no response found!)
21	15.062375000	Cisco_3a:f1:05	CDP/VTP/DTP/PAgP/UD	CDP	453	Device ID: tux-sw6 Port ID: FastEthernet0/3
22	15.111760000	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1523, seq=10/2560, ttl=63 (no response found!)
23	15.569084000	Cisco_3a:f1:05	Cisco_3a:f1:05	LOOP	60	Reply
24	16.038766000	Cisco_3a:f1:05	Spanning-tree-(for-	STP	60	Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8005
25	16.119769000	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1523, seq=11/2816, ttl=63 (no response found!)
26	17.127783000	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1523, seq=12/3072, ttl=63 (no response found!)
27	18.049120000	Cisco_3a:f1:05	Spanning-tree-(for-	STP	60	Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8005
28	18.135802000	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1523, seq=13/3328, ttl=63 (no response found!)
29	19.143781000	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1523, seq=14/3584, ttl=63 (no response found!)
30	20.048392000	Cisco_3a:f1:05	Spanning-tree-(for-	STP	60	Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8005
31	20.151776000	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1523, seq=15/3840, ttl=63 (no response found!)
32	21.159785000	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1523, seq=16/4096, ttl=63 (no response found!)
33	22.053324000	Cisco_3a:f1:05	Spanning-tree-(for-	STP	60	Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8005
34	22.167794000	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1523, seq=17/4352, ttl=63 (no response found!)
35	23.175813000	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1523, seq=18/4608, ttl=63 (no response found!)
36	24.063120000	Cisco_3a:f1:05	Spanning-tree-(for-	STP	60	Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8005

Figura 16: `Ping` de tux61 para tux62 (vista de tux62)

Exp6: Gráfico de transferência tux61

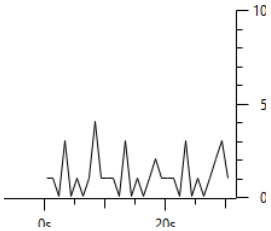


Figura 17: Gráfico de transferência tux61

Exp6: Gráfico de transferência tux62

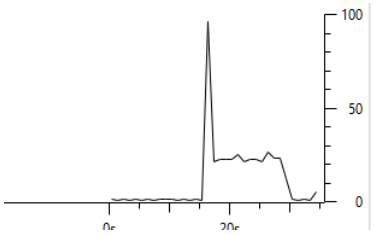


Figura 18: Gráfico de transferência tux62