

Desenvolvimento de uma aplicação de Download e Configuração de uma Rede

Trabalho 2 - Relatório



Mestrado Integrado em Engenharia Informática e
Computação

Redes de Computadores

Professor: Manuel Ricardo

Turma 7:

Bernardo José Coelho Leite
Luís Miguel Santos Monteiro Saraiva
João Carlos Oliveira Lago

22 de Dezembro de 2017

Sumário

No âmbito da Unidade Curricular de Redes de Computadores e do projeto 2 “Parte 1 - Desenvolvimento de uma Aplicação de Download” e “Parte 2 - Configuração e estudo de uma Rede”, apresentamos este relatório para descrever e explicar todos os aspetos do nosso trabalho.

Os objetivos para este projeto foram cumpridos, ou seja, os comandos de configuração do router e do switch que utilizamos para desenvolver a aplicação permitem a transferência do mesmo , através de **FTP**, sem quaisquer erros. A rede foi também configurada com sucesso.

Introdução

São inerentes a este projeto dois objetivos principais. Por um lado, utilizar o **File Transfer Protocol** (FTP) para criar uma aplicação capaz de realizar o download de um determinado ficheiro. Depois, configurar e analisar uma rede cujos passos são os descritos no guião envolvendo a configuração do **IP**, implementação de **LANs** virtuais no Switch, a configuração de um **Router** em Linux, a configuração de um **Router com a implantação de NAT**, a configuração do **DNS** e ainda conexões **TCP**.

Através deste relatório pretendemos reportar todo o percurso que levamos para conseguir implementar a aplicação, detalhar toda a sua estrutura, caracterizar o seu funcionamento e ainda fundamentar as decisões que tomamos. Por fim, explicar os comandos que levaram à configuração de uma Rede bem como fazer a sua análise.

Aplicação de Download

Arquitetura

Para a implementação da Aplicação de *Download* adotou-se a seguinte estrutura:

download.c

Em *download.c* trata-se do processamento da string que é passada como argumento e que segue a sintaxe *URL* descrita em *RFC1738*: *ftp://[<user>:<password>@]<host>/<url-path>*. Através destes componentes, ou seja, a *string* de utilizador, palavra-passe, *host* e *path* obtém-se a informação necessária para que se proceda à transferência de um ficheiro.

ftp.h e ftp.c

Em **ftp.h** é onde estão declaradas as funções que permitem estabelecer com o host definido, ligações de conexão ao *socket*, conexão ftp, login e ainda operações de leitura e escrita. Também aqui se apresenta uma estrutura *FTP* cujas variáveis permitem a configuração de uma instância dessa mesma estrutura. Em **ftp.c** faz-se a implementação das funções referidas.

```
14  typedef struct FTP {
15      int control_socket_fd;
16      int data_socket_fd;
17  } ftp;
18  char* read_Sock(FILE* fp, char* code);
19  int connect_Sock(char* ip, int port);
20  void connect_ftp(ftp* ftp, char* ip, int port);
21  void login_Sock(ftp* ftp, char* user, char* pass);
22
23  void write_Sock(ftp* ftp, char* str);
```

imagem 1 - struct FTP.

download e ftp

Na função principal, main, começa-se por ler *url_init* que deverá conter a primeira string com "ftp://".

```
if(strcmp(url_init, "ftp://\0") != 0) {  
    fprintf(stderr, "usage: start the url with ftp://\n");  
    exit(1);  
}
```

imagem 2 - url_init.

Depois, lêem-se as strings de user e pass que devem conter a informação relativas ao utilizador e *password*, respetivamente.

```
if(strlen(user) == strlen(url_aux[0])-1) {  
    fprintf(stderr, "usage: Declare an user\n");  
    exit(1);  
}  
printf("USER: %s\n", user);  
  
memcpy(pass, url_aux[0] + 2 + strlen(user), strlen(url_aux[0]));  
strtok(pass, "@");  
  
if(strlen(pass) == strlen(url_aux[0]) - 2 - strlen(user)) {  
    fprintf(stderr, "usage: Declare a password\n");  
    exit(1);  
}  
printf("PASS: %s\n", pass);
```

imagem 3 - user e password.

O mesmo processo é aplicado para as strings de *host* e *path*.

```
if(strlen(host) == strlen(url_aux_2[0])) {  
    fprintf(stderr, "usage: Declare an host\n");  
    exit(1);  
}  
printf("HOST: %s\n", host);
```

imagem 4 - host.

```
if(strlen(path) == 0) {  
    fprintf(stderr, "usage: Declare an path\n");  
    exit(1);  
}  
printf("PATH: %s\n", path);
```

imagem 5 - path.

O próximo passo passa por estabelecer as ligações necessárias com o host definido anteriormente. Para tal, começa-se por chamar a função **gethostbyname** que leva como argumento o **host** e retorna uma estrutura do tipo **hostent**. De seguida, chama-se a função **inet_ntoa** com o objetivo de converter o endereço da rede da **struct in_addr** numa string composta apenas por pontos e números.

Nesta fase verifica-se que as variáveis que guardam as informações sobre o utilizador e password estão atribuídas e se não for o caso, atribui-se "**anonymous**" e " " a essas variáveis, respetivamente. Através da chamada da função **fdopen()** associa-se o **ftp.control_socket** com o ficheiro descritor fp em modo de leitura. Considera-se agora que o programa está pronto a enviar mensagens e, por isso, é altura de se fazer a autenticação. Utiliza-se **write_Sock()** para escrever a string do utilizador e **read_Sock()** para ler a mensagem de resposta, neste caso 331. Utiliza-se novamente **write_Sock()** para escrever a password e **read_Sock()** para ler a mensagem de resposta, neste caso 230. Para que o **servidor FTP** seja capaz de transferir informação em modo passivo é enviado o comando para esse efeito e depois desse envio, interpreta-se a resposta do **host** para que as informações de **IP** e **Port** sejam guardadas.

```
FILE* fp = fdopen(ftp.control_socket_fd, "r");
write_Sock(&ftp, write_user);

read_Sock(fp, "331 ");

write_Sock(&ftp, write_pass);

read_Sock(fp, "230 ");
```

imagem 6 - write_Sock().

```
char* write_pasv = malloc(6);
sprintf(write_pasv, "PASV\r\n");
write_Sock(&ftp, write_pasv);
char* read_pasv = malloc(52);
read_pasv = read_Sock(fp, "227 ");
printf("READ_PASV: %s", read_pasv);
```

imagem 7 - sprintf() .

Quando já estão guardados todos os dados de ligação, depois de entrar em modo passivo, é o momento para se fazer a ligação através de **connect_Sock()**. Depois da ligação ter sido efetuada com sucesso uma mensagem é enviada com o caminho do ficheiro no servidor. Se o ficheiro não existir, o programa alerta para esse facto.

```
if ((ftp.data_socket_fd = connect_Sock(read_pasv, portResult)) < 0) {
    printf(
        "ERROR: Incorrect file descriptor associated to ftp data socket fd.\n");
    return 1;
}
```

imagem 8 - connect_Sock() .

Agora que os dados necessários para que se realize a transferência foram corretamente enviados, o *download* inicia-se. Através da abertura de um ficheiro em modo de escrita procede-se à escrita dos dados recebidos a cada número pré-definido de bytes.

```
while ((bytes = read(ftp.data_socket_fd, buf, sizeof(buf)))) {
    if (bytes < 0) {
        printf("ERROR: Nothing was received from data socket fd.\n");
        return 1;
    }

    if ((bytes = fwrite(buf, bytes, 1, file)) < 0) {
        printf("ERROR: Cannot write data in file.\n");
        return 1;
    }
}
```

imagem 9 - escrita dos dados.

Assim que a totalidade dos bytes que compõem o ficheiro estiverem corretamente guardados, fecha-se a ligação enviando uma mensagem de terminação ao servidor.

```
fclose(file);
close(ftp.data_socket_fd);

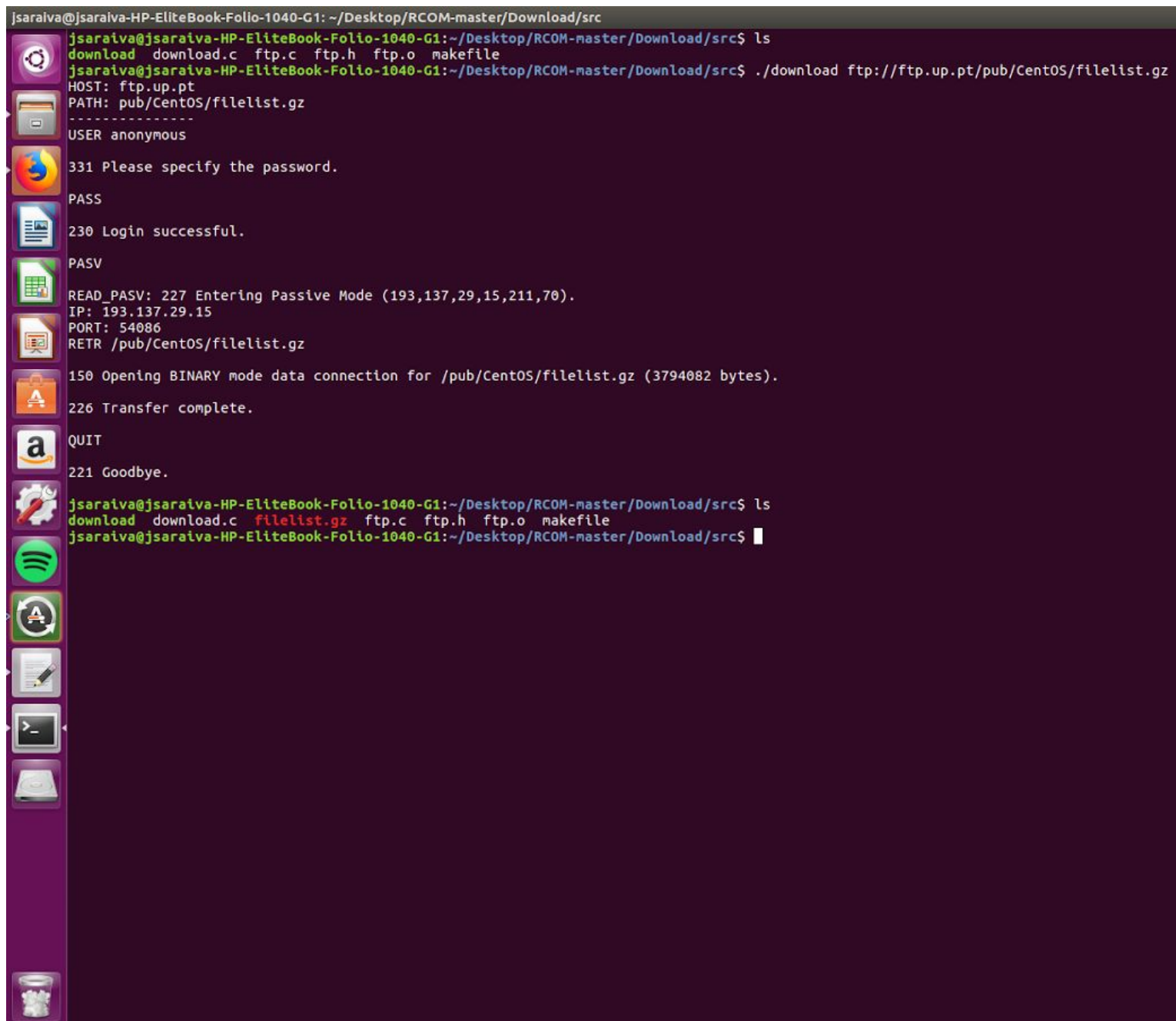
char disc[1024];

read_Sock(fp, "226 ");
sprintf(disc, "QUIT\r\n");
write_Sock(&ftp, disc);
read_Sock(fp, "221 ");
if (ftp.control_socket_fd)
close(ftp.control_socket_fd);
```

imagem 10 - fim da ligação.

Resultados

Para fins de teste, a imagem abaixo exemplifica a transferência de um ficheiro com sucesso. Note-se que neste caso o utilizador é “Annoymous”, o host é “ftp.up.pt” e que o ficheiro a transferir é o “filelist.gz”. O download foi feito com sucesso sendo que todos os bytes que compõe o ficheiro são recebidos na íntegra.

A terminal window with a dark purple background and a sidebar of application icons on the left. The terminal text shows the execution of an FTP client. The user 'jsaraiva' is in the directory '~/Desktop/RCOM-master/Download/src'. They run 'ls' showing files like 'download.c', 'ftp.c', 'ftp.h', 'ftp.o', and 'makefile'. Then they run './download ftp://ftp.up.pt/pub/CentOS/filelist.gz'. The output shows the connection process: 'HOST: ftp.up.pt', 'PATH: pub/CentOS/filelist.gz', 'USER anonymous', a password prompt '331 Please specify the password.', 'PASS', '230 Login successful.', 'PASV', 'READ_PASV: 227 Entering Passive Mode (193,137,29,15,211,70).', 'IP: 193.137.29.15', 'PORT: 54086', 'RETR /pub/CentOS/filelist.gz', '150 Opening BINARY mode data connection for /pub/CentOS/filelist.gz (3794082 bytes).', '226 Transfer complete.', 'QUIT', and '221 Goodbye.'. Finally, they run 'ls' again, and 'filelist.gz' is now listed among the files.

```
jsaraiva@jsaraiva-HP-EliteBook-Folio-1040-G1: ~/Desktop/RCOM-master/Download/src
jsaraiva@jsaraiva-HP-EliteBook-Folio-1040-G1:~/Desktop/RCOM-master/Download/src$ ls
download download.c ftp.c ftp.h ftp.o makefile
jsaraiva@jsaraiva-HP-EliteBook-Folio-1040-G1:~/Desktop/RCOM-master/Download/src$ ./download ftp://ftp.up.pt/pub/CentOS/filelist.gz
HOST: ftp.up.pt
PATH: pub/CentOS/filelist.gz
-----
USER anonymous

331 Please specify the password.
PASS
230 Login successful.
PASV
READ_PASV: 227 Entering Passive Mode (193,137,29,15,211,70).
IP: 193.137.29.15
PORT: 54086
RETR /pub/CentOS/filelist.gz
150 Opening BINARY mode data connection for /pub/CentOS/filelist.gz (3794082 bytes).
226 Transfer complete.
QUIT
221 Goodbye.
jsaraiva@jsaraiva-HP-EliteBook-Folio-1040-G1:~/Desktop/RCOM-master/Download/src$ ls
download download.c filelist.gz ftp.c ftp.h ftp.o makefile
jsaraiva@jsaraiva-HP-EliteBook-Folio-1040-G1:~/Desktop/RCOM-master/Download/src$
```

imagem 11 - Análise dos Resultados, download do ficheiro filelist.gz.

Experiência 1 - Estudo e Configuração de uma Rede

O principal objetivo desta primeira experiência foi habilitar a comunicação entre dois computadores através da configuração dos seus endereços **IP**. O comando **ifconfig** foi utilizado para esse efeito, ou seja, atribuir um determinado **IP** ao **IP** da interface. Desta forma, depois de configurar os **IP**'s das portas **eth0** e de adicionar as rotas pretendidas procedeu-se ao envio do sinal **ping**.

O uso deste comando justifica-se pelo facto de nos querermos certificar que existe uma ligação entre os dois computadores. Para além disso, foi também possível observar os pedidos **ARP** bem como os seus pings e respostas com o endereço **MAC**, que é um endereço único e identificador de uma só porta. Nesse momento, compreendemos que o **Address Resolution Protocol (ARP)** caracteriza-se por ser um protocolo utilizado não só em endereços da camada da Internet mas também para a resolução de endereços da camada de rede (**IP**). Vimos também que o comando ping gera pacotes do protocolo **ICMP** após obtenção do endereço **MAC** através dos pacotes **ARP** e que para os distinguir é preciso observar o cabeçalho da trama **Ethernet**. Relativamente à determinação do tamanho de uma trama notou-se que os pacotes **IP** contêm essa informação.

Por último, analisamos a **interface loopback** e concluímos que se trata de uma interface de rede virtual que permite a comunicação entre **client-host**, é reservada para todo o bloco de endereços **IP** com identificação **127.0.0.0/8**. As suas principais aplicações são a de diagnosticar e resolver problemas na rede e fazer testes de **software** e de conectividade.

Experiência 2 - Implementação de duas LANs virtuais no switch

Através desta experiência pudemos compreender como criar e configurar duas **LANs** virtuais no **switch**. A finalidade inerente a esta experiência é a de criar duas sub-redes diferentes e assim, impossibilitar a comunicação entre os computadores dessas mesmas redes.

Para tal, uma **LAN** virtual foi criada entre os computadores 1 e 4 e outra, no computador 2. Utilizando a consola de configuração para configurar o **switch** utilizou-se:

O comando **vlan x** em que **x** é a identificação da **VLAN**;

O comando **interface fastethernet 0/y**, em que **y** é o identificador da porta switch, para adicionar as portas do switch às VLANs e permitir assim a criação das duas sub-redes;

O comando **switchport mode access**, para forçar a que a porta do switch se comporte sempre apenas como uma porta de acesso;

O comando **switchport access vlan k**, em que **k** é a identificação da **VLAN**;

A título de teste, efetuou-se um "**ping**" para uma das máquinas para comprovar a impossibilidade de comunicação entre os computadores pelo facto de se encontrarem em redes (sub-redes) diferentes.

Finalmente, concluiu-se que, tendo em conta o seguimento lógico desta rede de computadores onde apenas duas **VLANs** foram criadas, existem por isso, dois domínios de **broadcast**.

Experiência 3 - Configurar um Router em Linux

Esta experiência veio colmatar a última tendo em conta que, através da configuração do computador 4 é criado um router entre as duas **VLANs** criadas até então. Com a utilização do comando **eth1 up** ligou-se a interface 1 do computador 4 para se proceder à configuração de um IP na mesma gama que o computador 2. O objetivo foi conseguir-se fazer ping no computador 1 a partir do computador 2. Para tal, com o comando **route add -net ip gw ip2** adicionaram-se as rotas pretendidas, no computador 1 adicionou-se uma rota que indica que os pacotes deverão ser redirecionados para o **IP** do computador 4 repetindo-se o processo no computador 2. Ao analisar este último comando compreendemos que o primeiro endereço **identifica a gama de endereços para a qual se quer adicionar uma determinada rota** e que o segundo define o IP para o qual se deve redirecionar o pacote.

Nesta fase torna-se possível a comunicação entre o computador 1 e 2 tendo em conta que os seus pedidos são redirecionados para o computador 4, computador este que está ligado à sub-rede partilhada pelos outros dois.

Ao interpretar os **resultados dos logs** concluímos que quando se efetua o ping do computador 2 ao 1, o pacote **ICMP** contém o endereço **MAC** do computador 4 que é o computador de destino e que também na resposta do computador 1 o endereço **MAC** é o 4, mais uma vez, o endereço de destino.

Experiência 4 - Configurar um Router Comercial e implementar NAT

Neste exercício, configuramos o router e adicionamos a funcionalidade **NAT(Network address translation)**. Com esta funcionalidade o **router traduz os endereços da rede interna em endereços a usar no exterior** através de um procedimento que gera um número de 16 bits, utilizando esse valor numa *hash table*, e escrevendo-o no campo da porta de origem. Assim possibilita-se a comunicação na Internet a partir de endereços externos.

Quando um computador recebe uma resposta do exterior, um endereço, este é convertido num endereço da rede interior através da **NAT Translation Table** e assim é localizado o computador para onde a resposta deve ser enviada na rede interna. Para configurar uma rota estática no router comercial utilizou-se o comando '**ip route ...**' na consola do router. O **tux1** tem uma ligação com o router pois é estabelecida uma ligação entre a **vlan0**, onde se encontra o **tux1**, com a **vlan1** ligado ao router, utilizando o **tux4** como **gateway**. Também é definido o **tux4** como default router do **tux1**, e o **router(Rc)** como default router do **tux4**. Desta forma, podem ser enviados pacotes do **tux1** para o **tux4** e de seguida para o router. Para configurar **NAT** no router comercial seguimos os comandos apresentados no guião.

Experiência 5 - DNS

O principal objetivo nesta experiência foi conseguir aceder a uma Rede externa, e assim conseguir aceder à Internet. Para tal, foi necessário configurar o **DNS**.

O **DNS** foi configurado nos três computadores através do servidor **DNS services.netlab.fe.up.pt (172.16.1.1)**. Para tal, editou-se o ficheiro **etc/resolv.conf** de forma a que contivesse os comandos “**search netlab.fe.up.pt nameserver 172.16.1.1***”.

Depois da configuração, executamos ping a um servidor externo enviando um packet de **DNS** que pede pelo **IP** e o servidor **DNS services.netlab.fe.up.pt** responde com um packet contendo o **IP** do servidor externo e mais outras informações como o *data length*, *time to live* e informação igual ao packet enviado como *query*.

Experiência 6 – Ligação TCP

Nesta experiência foi testado o programa ‘*download*’ desenvolvido para o projeto e foi transferido com sucesso um ficheiro de um servidor **FTP**, assegurando assim a correta configuração da rede.

O **protocolo TCP** utiliza um sistema idêntico ao **Go-Back-N ARQ**, um método que previne erros na transmissão de dados utilizando **acknowledgments(ACK)**, o controlo de erros tem com base a sequência dos bytes. Uma trama é retransmitida se não for recebido o **ACK** dessa trama.

O **TCP** usa um controlo de congestão onde é estimado o tráfico ideal ao incrementar e decrementar a janela de congestionamento com base no congestionamento da rede. Devido à taxa de transferência ser distribuída igualmente por todas as ligações, a taxa de transferência para cada ligação diminui quantas mais ligações existirem.

Conclusões

Com o finalizar do trabalho consideramos que os objetivos esperados foram cumpridos. De acordo com as especificações dadas, implementou-se com sucesso uma aplicação de download e ainda, a configuração de uma rede de forma correta. Ao observar os resultados finais concluímos que, da nossa parte, houve um notório desenvolvimento e progresso sobre a matéria em causa. Sentimos que somos agora capazes de configurar uma rede e perceber as suas principais características de configuração e funcionamento. Com a implementação da aplicação que faz o *download* ficamos a compreender melhor como funcionam os protocolos de ligação e a sua importância no âmbito das Redes de Computadores. Em suma, contamos com a experiência adquirida deste trabalho para que futuramente consigamos resolver novos desafios.

Referências

Guião do trabalho, Manuel Ricardo

Beej's Guide to Network Programming - Using Internet Sockets, Beej's

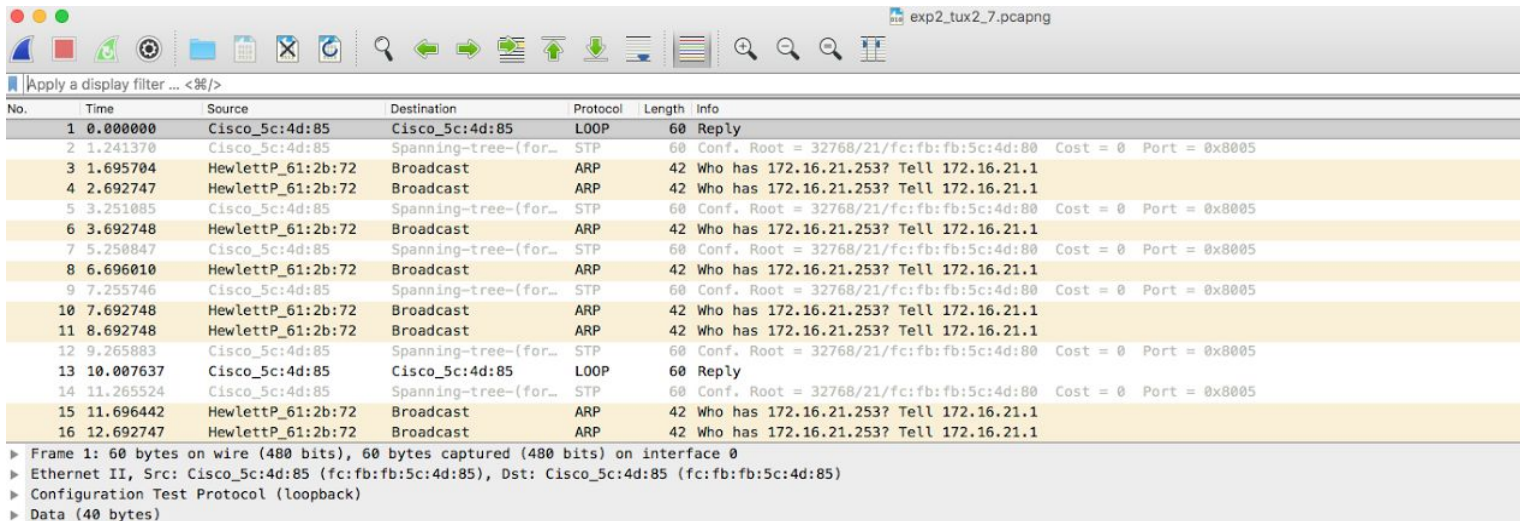
Linux Network Guide

Switch Catalyst 3560 Switch Software Configuration Guide

Anexos

Ficheiros logs:

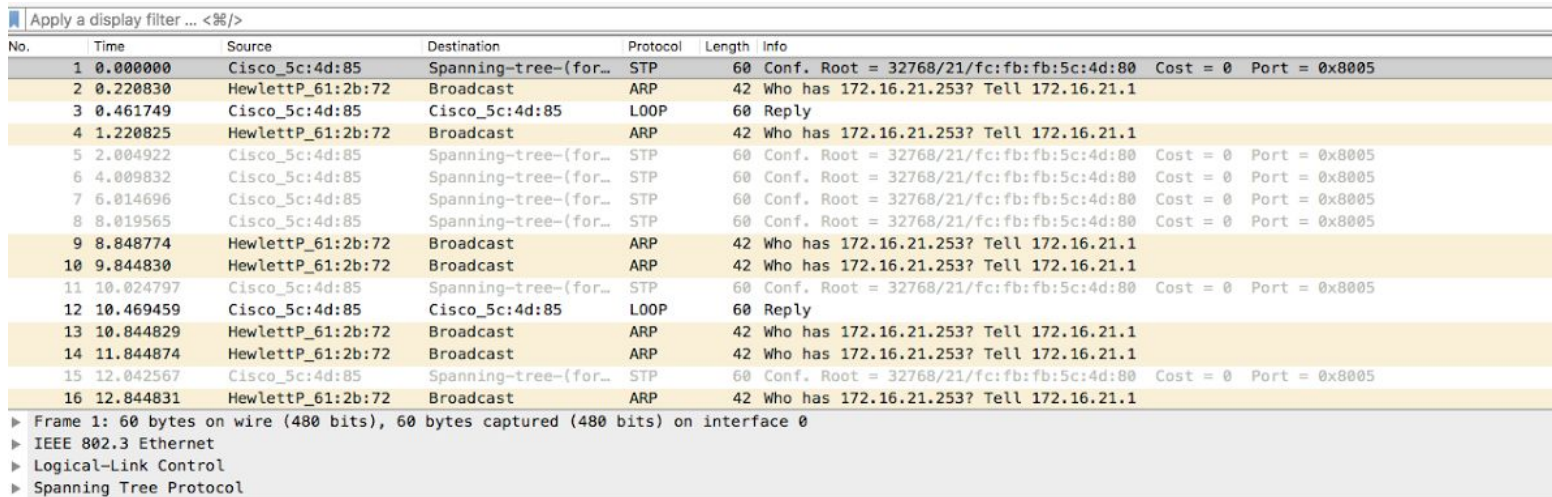
exp2_tux2_7.pcapng



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_5c:4d:85	Cisco_5c:4d:85	LOOP	60	Reply
2	1.241370	Cisco_5c:4d:85	Spanning-tree-(for...	STP	60	Conf. Root = 32768/21/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8005
3	1.695704	HewlettP_61:2b:72	Broadcast	ARP	42	Who has 172.16.21.253? Tell 172.16.21.1
4	2.692747	HewlettP_61:2b:72	Broadcast	ARP	42	Who has 172.16.21.253? Tell 172.16.21.1
5	3.251085	Cisco_5c:4d:85	Spanning-tree-(for...	STP	60	Conf. Root = 32768/21/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8005
6	3.692748	HewlettP_61:2b:72	Broadcast	ARP	42	Who has 172.16.21.253? Tell 172.16.21.1
7	5.250847	Cisco_5c:4d:85	Spanning-tree-(for...	STP	60	Conf. Root = 32768/21/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8005
8	6.696010	HewlettP_61:2b:72	Broadcast	ARP	42	Who has 172.16.21.253? Tell 172.16.21.1
9	7.255746	Cisco_5c:4d:85	Spanning-tree-(for...	STP	60	Conf. Root = 32768/21/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8005
10	7.692748	HewlettP_61:2b:72	Broadcast	ARP	42	Who has 172.16.21.253? Tell 172.16.21.1
11	8.692748	HewlettP_61:2b:72	Broadcast	ARP	42	Who has 172.16.21.253? Tell 172.16.21.1
12	9.265883	Cisco_5c:4d:85	Spanning-tree-(for...	STP	60	Conf. Root = 32768/21/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8005
13	10.007637	Cisco_5c:4d:85	Cisco_5c:4d:85	LOOP	60	Reply
14	11.265524	Cisco_5c:4d:85	Spanning-tree-(for...	STP	60	Conf. Root = 32768/21/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8005
15	11.696442	HewlettP_61:2b:72	Broadcast	ARP	42	Who has 172.16.21.253? Tell 172.16.21.1
16	12.692747	HewlettP_61:2b:72	Broadcast	ARP	42	Who has 172.16.21.253? Tell 172.16.21.1

► Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
► Ethernet II, Src: Cisco_5c:4d:85 (fc:fb:5c:4d:85), Dst: Cisco_5c:4d:85 (fc:fb:fb:5c:4d:85)
► Configuration Test Protocol (loopback)
► Data (40 bytes)

exp2_tux2_10.pcapng



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_5c:4d:85	Spanning-tree-(for...	STP	60	Conf. Root = 32768/21/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8005
2	0.220830	HewlettP_61:2b:72	Broadcast	ARP	42	Who has 172.16.21.253? Tell 172.16.21.1
3	0.461749	Cisco_5c:4d:85	Cisco_5c:4d:85	LOOP	60	Reply
4	1.220825	HewlettP_61:2b:72	Broadcast	ARP	42	Who has 172.16.21.253? Tell 172.16.21.1
5	2.004922	Cisco_5c:4d:85	Spanning-tree-(for...	STP	60	Conf. Root = 32768/21/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8005
6	4.009832	Cisco_5c:4d:85	Spanning-tree-(for...	STP	60	Conf. Root = 32768/21/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8005
7	6.014696	Cisco_5c:4d:85	Spanning-tree-(for...	STP	60	Conf. Root = 32768/21/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8005
8	8.019565	Cisco_5c:4d:85	Spanning-tree-(for...	STP	60	Conf. Root = 32768/21/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8005
9	8.848774	HewlettP_61:2b:72	Broadcast	ARP	42	Who has 172.16.21.253? Tell 172.16.21.1
10	9.844830	HewlettP_61:2b:72	Broadcast	ARP	42	Who has 172.16.21.253? Tell 172.16.21.1
11	10.024797	Cisco_5c:4d:85	Spanning-tree-(for...	STP	60	Conf. Root = 32768/21/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8005
12	10.469459	Cisco_5c:4d:85	Cisco_5c:4d:85	LOOP	60	Reply
13	10.844829	HewlettP_61:2b:72	Broadcast	ARP	42	Who has 172.16.21.253? Tell 172.16.21.1
14	11.844874	HewlettP_61:2b:72	Broadcast	ARP	42	Who has 172.16.21.253? Tell 172.16.21.1
15	12.042567	Cisco_5c:4d:85	Spanning-tree-(for...	STP	60	Conf. Root = 32768/21/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8005
16	12.844831	HewlettP_61:2b:72	Broadcast	ARP	42	Who has 172.16.21.253? Tell 172.16.21.1

► Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
► IEEE 802.3 Ethernet
► Logical-Link Control
► Spanning Tree Protocol

exp2_tux21_5.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
2	1.999416	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
3	2.414469	172.16.20.1	172.16.20.254	ICMP	98	Echo (ping) request id=0x0ba4, seq=1/256, ttl=64 (reply in 4)
4	2.414683	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0ba4, seq=1/256, ttl=64 (request in 3)
5	3.413476	172.16.20.1	172.16.20.254	ICMP	98	Echo (ping) request id=0x0ba4, seq=2/512, ttl=64 (reply in 6)
6	3.413707	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0ba4, seq=2/512, ttl=64 (request in 5)
7	4.004343	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
8	4.412476	172.16.20.1	172.16.20.254	ICMP	98	Echo (ping) request id=0x0ba4, seq=3/768, ttl=64 (reply in 9)
9	4.412628	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0ba4, seq=3/768, ttl=64 (request in 8)
10	5.411480	172.16.20.1	172.16.20.254	ICMP	98	Echo (ping) request id=0x0ba4, seq=4/1024, ttl=64 (reply in 11)
11	5.411689	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0ba4, seq=4/1024, ttl=64 (request in 10)
12	6.013629	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
13	6.411114	172.16.20.1	172.16.20.254	ICMP	98	Echo (ping) request id=0x0ba4, seq=5/1280, ttl=64 (reply in 14)
14	6.411272	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0ba4, seq=5/1280, ttl=64 (request in 13)
15	7.411136	172.16.20.1	172.16.20.254	ICMP	98	Echo (ping) request id=0x0ba4, seq=6/1536, ttl=64 (reply in 16)
16	7.411369	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0ba4, seq=6/1536, ttl=64 (request in 15)

▶ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
 ▶ IEEE 802.3 Ethernet
 ▶ Logical-Link Control
 ▶ Spanning Tree Protocol

exp2_tux21_7.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
2	1.276362	172.16.20.1	172.16.20.255	ICMP	98	Echo (ping) request id=0x0bf8, seq=1/256, ttl=64 (no response found!)
3	1.276735	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0bf8, seq=1/256, ttl=64
4	2.004806	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
5	2.275369	172.16.20.1	172.16.20.255	ICMP	98	Echo (ping) request id=0x0bf8, seq=2/512, ttl=64 (no response found!)
6	2.275520	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0bf8, seq=2/512, ttl=64
7	3.274374	172.16.20.1	172.16.20.255	ICMP	98	Echo (ping) request id=0x0bf8, seq=3/768, ttl=64 (no response found!)
8	3.274583	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0bf8, seq=3/768, ttl=64
9	4.009738	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
10	4.273650	172.16.20.1	172.16.20.255	ICMP	98	Echo (ping) request id=0x0bf8, seq=4/1024, ttl=64 (no response found!)
11	4.273809	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0bf8, seq=4/1024, ttl=64
12	5.273654	172.16.20.1	172.16.20.255	ICMP	98	Echo (ping) request id=0x0bf8, seq=5/1280, ttl=64 (no response found!)
13	5.273885	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0bf8, seq=5/1280, ttl=64
14	5.293034	Cisco_5c:4d:83	Cisco_5c:4d:83	LOOP	60	Reply
15	6.014647	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
16	6.273647	172.16.20.1	172.16.20.255	ICMP	98	Echo (ping) request id=0x0bf8, seq=6/1536, ttl=64 (no response found!)

▶ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
 ▶ IEEE 802.3 Ethernet
 ▶ Logical-Link Control
 ▶ Spanning Tree Protocol

exp2_tux21_10.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
2	2.004874	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
3	4.009590	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
4	4.991263	Cisco_5c:4d:83	Cisco_5c:4d:83	LOOP	60	Reply
5	6.014508	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
6	8.019373	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
7	10.024178	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
8	12.029295	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
9	14.033849	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
10	14.990267	Cisco_5c:4d:83	Cisco_5c:4d:83	LOOP	60	Reply
11	16.038737	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
12	18.043543	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
13	20.048582	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
14	22.053474	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
15	24.058081	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
16	24.997917	Cisco_5c:4d:83	Cisco_5c:4d:83	LOOP	60	Reply

▶ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
 ▶ IEEE 802.3 Ethernet
 ▶ Logical-Link Control
 ▶ Spanning Tree Protocol

exp2_tux24_7.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_5c:4d:87	CDP/VTP/DTP/PAGP/U...	CDP	453	Device ID: tux-sw2 Port ID: FastEthernet0/5
2	1.163015	Cisco_5c:4d:87	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8007
3	2.461359	Cisco_5c:4d:87	Cisco_5c:4d:87	LOOP	60	Reply
4	3.166607	Cisco_5c:4d:87	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8007
5	5.172327	Cisco_5c:4d:87	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8007
6	7.175791	Cisco_5c:4d:87	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8007
7	8.452175	172.16.20.1	172.16.20.255	ICMP	98	Echo (ping) request id=0x0bf8, seq=1/256, ttl=64 (no response found!)
8	8.452205	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0bf8, seq=1/256, ttl=64
9	9.181543	Cisco_5c:4d:87	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8007
10	9.451196	172.16.20.1	172.16.20.255	ICMP	98	Echo (ping) request id=0x0bf8, seq=2/512, ttl=64 (no response found!)
11	9.451221	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0bf8, seq=2/512, ttl=64
12	10.450216	172.16.20.1	172.16.20.255	ICMP	98	Echo (ping) request id=0x0bf8, seq=3/768, ttl=64 (no response found!)
13	10.450245	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0bf8, seq=3/768, ttl=64
14	11.185458	Cisco_5c:4d:87	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8007
15	11.449509	172.16.20.1	172.16.20.255	ICMP	98	Echo (ping) request id=0x0bf8, seq=4/1024, ttl=64 (no response found!)
16	11.449537	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0bf8, seq=4/1024, ttl=64

▶ Frame 1: 453 bytes on wire (3624 bits), 453 bytes captured (3624 bits) on interface 0
 ▶ IEEE 802.3 Ethernet
 ▶ Logical-Link Control
 ▶ Cisco Discovery Protocol

exp2_tux24_10.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_5c:4d:87	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8007
2	2.083960	Cisco_5c:4d:87	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8007
3	4.089429	Cisco_5c:4d:87	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8007
4	4.965070	Cisco_5c:4d:87	Cisco_5c:4d:87	LOOP	60	Reply
5	6.015256	Cisco_5c:4d:87	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8007
6	8.018989	Cisco_5c:4d:87	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8007
7	10.024791	Cisco_5c:4d:87	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8007
8	12.028470	Cisco_5c:4d:87	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8007
9	14.033993	Cisco_5c:4d:87	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8007
10	14.972662	Cisco_5c:4d:87	Cisco_5c:4d:87	LOOP	60	Reply
11	16.037627	Cisco_5c:4d:87	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8007
12	18.043749	Cisco_5c:4d:87	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8007
13	20.047832	Cisco_5c:4d:87	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8007
14	22.053462	Cisco_5c:4d:87	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8007
15	22.429933	Cisco_5c:4d:87	CDP/VTP/DTP/PAGP/U...	CDP	453	Device ID: tux-sw2 Port ID: FastEthernet0/5
16	24.058942	Cisco_5c:4d:87	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8007

▶ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
 ▶ IEEE 802.3 Ethernet
 ▶ Logical-Link Control
 ▶ Spanning Tree Protocol

exp3_tux21_6.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.20.1	172.16.20.254	ICMP	98	Echo (ping) request id=0x0d3d, seq=1/256, ttl=64 (reply in 2)
2	0.000160	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0d3d, seq=1/256, ttl=64 (request in 1)
3	0.436624	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
4	0.999004	172.16.20.1	172.16.20.254	ICMP	98	Echo (ping) request id=0x0d3d, seq=2/512, ttl=64 (reply in 5)
5	0.999252	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0d3d, seq=2/512, ttl=64 (request in 4)
6	1.199763	Cisco_5c:4d:83	Cisco_5c:4d:83	LOOP	60	Reply
7	1.998004	172.16.20.1	172.16.20.254	ICMP	98	Echo (ping) request id=0x0d3d, seq=3/768, ttl=64 (reply in 8)
8	1.998160	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0d3d, seq=3/768, ttl=64 (request in 7)
9	2.441372	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
10	2.997007	172.16.20.1	172.16.20.254	ICMP	98	Echo (ping) request id=0x0d3d, seq=4/1024, ttl=64 (reply in 11)
11	2.997247	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0d3d, seq=4/1024, ttl=64 (request in 10)
12	3.996945	172.16.20.1	172.16.20.254	ICMP	98	Echo (ping) request id=0x0d3d, seq=5/1280, ttl=64 (reply in 13)
13	3.997104	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0d3d, seq=5/1280, ttl=64 (request in 12)
14	4.445832	Cisco_5c:4d:83	Spanning-tree-(for...	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
15	4.996951	172.16.20.1	172.16.20.254	ICMP	98	Echo (ping) request id=0x0d3d, seq=6/1536, ttl=64 (reply in 16)
16	4.997144	172.16.20.254	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0d3d, seq=6/1536, ttl=64 (request in 15)

▶ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
 ▶ Ethernet II, Src: G-Proc08_8c:af:9d (00:0f:fe:8c:af:9d), Dst: HewlettP_a6:a4:f1 (00:22:64:a6:a4:f1)
 ▶ Internet Protocol Version 4, Src: 172.16.20.1, Dst: 172.16.20.254
 ▶ Internet Control Message Protocol

exp3_tux4_8_1.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_3a:fc:07	Spanning-tree-(for...	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8007
2	1.149621	Cisco_3a:fc:07	Cisco_3a:fc:07	LOOP	60	Reply
3	2.005262	Cisco_3a:fc:07	Spanning-tree-(for...	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8007
4	4.010211	Cisco_3a:fc:07	Spanning-tree-(for...	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8007
5	6.015211	Cisco_3a:fc:07	Spanning-tree-(for...	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8007
6	8.020289	Cisco_3a:fc:07	Spanning-tree-(for...	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8007
7	9.487368	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2ac8, seq=1/256, ttl=64 (reply in 8)
8	9.487715	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2ac8, seq=1/256, ttl=63 (request in 7)
9	10.025654	Cisco_3a:fc:07	Spanning-tree-(for...	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8007
10	10.486379	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2ac8, seq=2/512, ttl=64 (reply in 11)
11	10.486593	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2ac8, seq=2/512, ttl=63 (request in 10)
12	11.161676	Cisco_3a:fc:07	Cisco_3a:fc:07	LOOP	60	Reply
13	11.485712	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2ac8, seq=3/768, ttl=64 (reply in 14)
14	11.485904	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2ac8, seq=3/768, ttl=63 (request in 13)
15	12.028590	Cisco_3a:fc:07	Spanning-tree-(for...	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8007
16	12.485723	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2ac8, seq=4/1024, ttl=64 (reply in 17)

▶ Frame 1: 60 bytes on wire (480 bits). 60 bytes captured (480 bits) on interface 0

exp3_tux4_8_2.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_3a:fc:09	Spanning-tree-(for...	STP	60	Conf. TC + Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8009
2	2.005053	Cisco_3a:fc:09	Spanning-tree-(for...	STP	60	Conf. TC + Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8009
3	4.014338	Cisco_3a:fc:09	Spanning-tree-(for...	STP	60	Conf. TC + Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8009
4	4.277968	Kye_04:20:99	Broadcast	ARP	42	Who has 172.16.11.1? Tell 172.16.11.253
5	4.278085	HewlettP_61:2e:c3	Kye_04:20:99	ARP	60	172.16.11.1 is at 00:21:5a:61:2e:c3
6	4.278100	172.16.11.253	172.16.11.1	ICMP	98	Echo (ping) request id=0x2ac8, seq=1/256, ttl=63 (reply in 7)
7	4.278258	172.16.11.1	172.16.11.253	ICMP	98	Echo (ping) reply id=0x2ac8, seq=1/256, ttl=64 (request in 6)
8	5.276965	172.16.11.253	172.16.11.1	ICMP	98	Echo (ping) request id=0x2ac8, seq=2/512, ttl=63 (reply in 9)
9	5.277118	172.16.11.1	172.16.11.253	ICMP	98	Echo (ping) reply id=0x2ac8, seq=2/512, ttl=64 (request in 8)
10	5.952365	Cisco_3a:fc:09	Cisco_3a:fc:09	LOOP	60	Reply
11	6.015514	Cisco_3a:fc:09	Spanning-tree-(for...	STP	60	Conf. TC + Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8009
12	6.276300	172.16.11.253	172.16.11.1	ICMP	98	Echo (ping) request id=0x2ac8, seq=3/768, ttl=64 (request in 12)
13	6.276429	172.16.11.1	172.16.11.253	ICMP	98	Echo (ping) reply id=0x2ac8, seq=3/768, ttl=64 (request in 12)
14	7.276309	172.16.11.253	172.16.11.1	ICMP	98	Echo (ping) request id=0x2ac8, seq=4/1024, ttl=63 (reply in 15)
15	7.276428	172.16.11.1	172.16.11.253	ICMP	98	Echo (ping) reply id=0x2ac8, seq=4/1024, ttl=64 (request in 14)
16	8.020587	Cisco_3a:fc:09	Spanning-tree-(for...	STP	60	Conf. TC + Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8009

▶ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0

▶ IEEE 802.3 Ethernet

▶ Logical-Link Control

▶ Spanning Tree Protocol

exp4_tux1_3.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_3a:fc:03	Spanning-tree-(for...	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
2	2.002800	Cisco_3a:fc:03	Spanning-tree-(for...	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
3	4.007766	Cisco_3a:fc:03	Spanning-tree-(for...	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
4	6.012522	Cisco_3a:fc:03	Spanning-tree-(for...	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
5	6.217829	Cisco_3a:fc:03	Cisco_3a:fc:03	LOOP	60	Reply
6	8.017423	Cisco_3a:fc:03	Spanning-tree-(for...	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
7	10.022261	Cisco_3a:fc:03	Spanning-tree-(for...	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
8	10.832506	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x3478, seq=1/256, ttl=64 (reply in 9)
9	10.832894	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x3478, seq=1/256, ttl=64 (request in 8)
10	11.831502	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x3478, seq=2/512, ttl=64 (reply in 11)
11	11.831759	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x3478, seq=2/512, ttl=64 (request in 10)
12	12.027119	Cisco_3a:fc:03	Spanning-tree-(for...	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
13	12.830504	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x3478, seq=3/768, ttl=64 (reply in 14)
14	12.830855	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x3478, seq=3/768, ttl=64 (request in 13)
15	13.829505	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x3478, seq=4/1024, ttl=64 (reply in 16)
16	13.829762	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x3478, seq=4/1024, ttl=64 (request in 15)

▶ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0

▶ IEEE 802.3 Ethernet

▶ Logical-Link Control

▶ Spanning Tree Protocol

exp4_tux1_5

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.10.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x34c4, seq=3/768, ttl=64 (no response found!)
2	0.327807	Cisco_3a:fc:03	Spanning-tree-(for...	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
3	1.007988	172.16.10.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x34c4, seq=4/1024, ttl=64 (no response found!)
4	2.015995	172.16.10.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x34c4, seq=5/1280, ttl=64 (no response found!)
5	2.332659	Cisco_3a:fc:03	Spanning-tree-(for...	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
6	2.576347	Cisco_3a:fc:03	Cisco_3a:fc:03	LOOP	60	Reply
7	2.989879	G-ProCom_8b:e4:ef	HewlettP_a6:a4:f8	ARP	42	Who has 172.16.10.254? Tell 172.16.10.1
8	2.990218	HewlettP_a6:a4:f8	G-ProCom_8b:e4:ef	ARP	60	172.16.10.254 is at 00:22:64:a6:a4:f8
9	3.024000	172.16.10.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x34c4, seq=6/1536, ttl=64 (no response found!)
10	4.031984	172.16.10.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x34c4, seq=7/1792, ttl=64 (no response found!)
11	4.342704	Cisco_3a:fc:03	Spanning-tree-(for...	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
12	5.039986	172.16.10.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x34c4, seq=8/2048, ttl=64 (no response found!)
13	6.047985	172.16.10.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x34c4, seq=9/2304, ttl=64 (no response found!)
14	6.342396	Cisco_3a:fc:03	Spanning-tree-(for...	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
15	7.055991	172.16.10.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x34c4, seq=10/2560, ttl=64 (no response found!)
16	8.063990	172.16.10.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x34c4, seq=11/2816, ttl=64 (no response found!)

▶ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
 ▶ Ethernet II, Src: G-ProCom_8b:e4:ef (00:0f:fe:8b:e4:ef), Dst: HewlettP_a6:a4:f8 (00:22:64:a6:a4:f8)
 ▶ Internet Protocol Version 4, Src: 172.16.10.1, Dst: 172.16.1.254
 ▶ Internet Control Message Protocol

exp4_tux1_7.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_7c:8f:81	Spanning-tree-(for...	STP	60	Conf. Root = 32768/40/00:1e:14:7c:8f:80 Cost = 0 Port = 0x8001
2	1.036893	172.16.40.1	172.16.2.254	ICMP	98	Echo (ping) request id=0x37e4, seq=1/256, ttl=64 (reply in 3)
3	1.038017	172.16.2.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x37e4, seq=1/256, ttl=62 (request in 2)
4	2.005038	Cisco_7c:8f:81	Spanning-tree-(for...	STP	60	Conf. Root = 32768/40/00:1e:14:7c:8f:80 Cost = 0 Port = 0x8001
5	2.038096	172.16.40.1	172.16.2.254	ICMP	98	Echo (ping) request id=0x37e4, seq=2/512, ttl=64 (reply in 6)
6	2.039007	172.16.2.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x37e4, seq=2/512, ttl=62 (request in 5)
7	3.037617	172.16.40.1	172.16.2.254	ICMP	98	Echo (ping) request id=0x37e4, seq=3/768, ttl=64 (reply in 8)
8	3.038475	172.16.2.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x37e4, seq=3/768, ttl=62 (request in 7)
9	3.887966	Cisco_7c:8f:81	Cisco_7c:8f:81	LOOP	60	Reply
10	4.014891	Cisco_7c:8f:81	Spanning-tree-(for...	STP	60	Conf. Root = 32768/40/00:1e:14:7c:8f:80 Cost = 0 Port = 0x8001
11	4.037614	172.16.40.1	172.16.2.254	ICMP	98	Echo (ping) request id=0x37e4, seq=4/1024, ttl=64 (reply in 12)
12	4.038463	172.16.2.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x37e4, seq=4/1024, ttl=62 (request in 11)
13	5.037610	172.16.40.1	172.16.2.254	ICMP	98	Echo (ping) request id=0x37e4, seq=5/1280, ttl=64 (reply in 14)
14	5.038488	172.16.2.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x37e4, seq=5/1280, ttl=62 (request in 13)
15	6.014815	Cisco_7c:8f:81	Spanning-tree-(for...	STP	60	Conf. Root = 32768/40/00:1e:14:7c:8f:80 Cost = 0 Port = 0x8001
16	6.037620	172.16.40.1	172.16.2.254	ICMP	98	Echo (ping) request id=0x37e4, seq=6/1536, ttl=64 (reply in 17)

▶ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
 ▶ IEEE 802.3 Ethernet
 ▶ Logical-Link Control
 ▶ Spanning Tree Protocol

exp4_tux2_4_4.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_3a:fc:05	Spanning-tree-(for...	STP	60	Conf. Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8005
2	2.000003	Cisco_3a:fc:05	Spanning-tree-(for...	STP	60	Conf. Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8005
3	3.567587	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) request id=0x03c6, seq=1/256, ttl=64 (no response found!)
4	4.004796	Cisco_3a:fc:05	Spanning-tree-(for...	STP	60	Conf. Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8005
5	4.575615	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) request id=0x03c6, seq=2/512, ttl=64 (no response found!)
6	5.583605	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) request id=0x03c6, seq=3/768, ttl=64 (no response found!)
7	6.014657	Cisco_3a:fc:05	Spanning-tree-(for...	STP	60	Conf. Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8005
8	6.089366	Cisco_3a:fc:05	Cisco_3a:fc:05	LOOP	60	Reply
9	6.554455	172.16.11.254	224.0.0.9	RIPv2	66	Response
10	6.591610	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) request id=0x03c6, seq=4/1024, ttl=64 (no response found!)
11	7.599613	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) request id=0x03c6, seq=5/1280, ttl=64 (no response found!)
12	8.014550	Cisco_3a:fc:05	Spanning-tree-(for...	STP	60	Conf. Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8005
13	8.581510	HewlettP_61:2e:c3	Kye_04:20:99	ARP	42	Who has 172.16.11.253? Tell 172.16.11.1
14	8.581612	Kye_04:20:99	HewlettP_61:2e:c3	ARP	60	172.16.11.253 is at 00:c0:df:04:20:99
15	8.607603	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) request id=0x03c6, seq=6/1536, ttl=64 (no response found!)
16	9.615603	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) request id=0x03c6, seq=7/1792, ttl=64 (no response found!)

▶ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
 ▶ IEEE 802.3 Ethernet
 ▶ Logical-Link Control
 ▶ Spanning Tree Protocol

exp4_tux2_4_7.pcapng

Apply a display filter ... <%%/>						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_3a:fc:05	Cisco_3a:fc:05	LOOP	60	Reply
2	0.477676	Cisco_3a:fc:05	Spanning-tree-(for...	STP	60	Conf. TC + Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8005
3	1.661548	172.16.11.254	224.0.0.9	RIPv2	66	Response
4	2.482665	Cisco_3a:fc:05	Spanning-tree-(for...	STP	60	Conf. TC + Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8005
5	4.487409	Cisco_3a:fc:05	Spanning-tree-(for...	STP	60	Conf. TC + Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8005
6	6.427863	HewlettP_61:2e:c3	Broadcast	ARP	42	Who has 172.16.11.254? Tell 172.16.11.1
7	6.428214	Cisco_e3:f9:b8	HewlettP_61:2e:c3	ARP	60	172.16.11.254 is at 68:ef:bd:e3:f9:b8
8	6.428222	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) request id=0x04b5, seq=1/256, ttl=64 (no response found!)
9	6.428561	172.16.11.254	172.16.11.1	ICMP	70	Redirect (Redirect for host)
10	6.428605	HewlettP_61:2e:c3	Broadcast	ARP	42	Who has 172.16.11.253? Tell 172.16.11.1
11	6.428612	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) request id=0x04b5, seq=1/256, ttl=63 (no response found!)
12	6.428694	Kye_04:20:99	HewlettP_61:2e:c3	ARP	60	172.16.11.253 is at 00:c0:df:04:20:99
13	6.492298	Cisco_3a:fc:05	Spanning-tree-(for...	STP	60	Conf. TC + Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8005
14	7.435861	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) request id=0x04b5, seq=2/512, ttl=64 (no response found!)
15	7.436162	172.16.11.254	172.16.11.1	ICMP	70	Redirect (Redirect for host)
16	8.443852	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) request id=0x04b5, seq=3/768, ttl=64 (no response found!)

▶ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
 ▶ Ethernet II, Src: Cisco_3a:fc:05 (fc:fb:fb:3a:fc:05), Dst: Cisco_3a:fc:05 (fc:fb:fb:3a:fc:05)
 ▶ Configuration Test Protocol (loopback)
 ▶ Data (40 bytes)

exp5_tux1.pcapng

Apply a display filter ... <%%/>						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_7c:8f:81	Spanning-tree-(for...	STP	60	Conf. Root = 32768/40/00:1e:14:7c:8f:80 Cost = 0 Port = 0x8001
2	0.231914	172.16.40.1	172.16.2.1	DNS	74	Standard query 0x84c4 A www.google.com
3	0.234428	172.16.2.1	172.16.40.1	DNS	226	Standard query response 0x84c4 A www.google.com A 216.58.205.36 NS ns4.google.com NS ns1.google.com
4	0.234534	172.16.40.1	216.58.205.36	ICMP	98	Echo (ping) request id=0x3ee4, seq=1/256, ttl=64 (reply in 5)
5	0.268317	216.58.205.36	172.16.40.1	ICMP	98	Echo (ping) reply id=0x3ee4, seq=1/256, ttl=48 (request in 4)
6	0.268429	172.16.40.1	172.16.2.1	DNS	86	Standard query 0xacef PTR 36.205.58.216.in-addr.arpa
7	0.270742	172.16.2.1	172.16.40.1	DNS	300	Standard query response 0xacef PTR 36.205.58.216.in-addr.arpa PTR mil04s24-in-f4.1e100.net PTR
8	1.235792	172.16.40.1	216.58.205.36	ICMP	98	Echo (ping) request id=0x3ee4, seq=2/512, ttl=64 (reply in 9)
9	1.270224	216.58.205.36	172.16.40.1	ICMP	98	Echo (ping) reply id=0x3ee4, seq=2/512, ttl=48 (request in 8)
10	2.004895	Cisco_7c:8f:81	Spanning-tree-(for...	STP	60	Conf. Root = 32768/40/00:1e:14:7c:8f:80 Cost = 0 Port = 0x8001
11	2.237259	172.16.40.1	216.58.205.36	ICMP	98	Echo (ping) request id=0x3ee4, seq=3/768, ttl=64 (reply in 12)
12	2.270399	216.58.205.36	172.16.40.1	ICMP	98	Echo (ping) reply id=0x3ee4, seq=3/768, ttl=48 (request in 11)
13	3.238114	172.16.40.1	216.58.205.36	ICMP	98	Echo (ping) request id=0x3ee4, seq=4/1024, ttl=64 (reply in 14)
14	3.271124	216.58.205.36	172.16.40.1	ICMP	98	Echo (ping) reply id=0x3ee4, seq=4/1024, ttl=48 (request in 13)
15	4.009726	Cisco_7c:8f:81	Spanning-tree-(for...	STP	60	Conf. Root = 32768/40/00:1e:14:7c:8f:80 Cost = 0 Port = 0x8001
16	4.239163	172.16.40.1	216.58.205.36	ICMP	98	Echo (ping) request id=0x3ee4, seq=5/1280, ttl=64 (reply in 17)

▶ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
 ▶ IEEE 802.3 Ethernet
 ▶ Logical-Link Control
 ▶ Spanning Tree Protocol

Comandos de configuração:

tux1.sh

```
#!/bin/bash

ifconfig eth0 down

ifconfig eth0 up 172.16.10.1/24

route add -net 172.16.11.0/24 gw
172.16.10.254

route add default gw
172.16.10.254
```

tux2.sh

```
#!/bin/bash

ifconfig eth0 down

ifconfig eth0 up 172.16.11.1/24

route add -net 172.16.10.0/24 gw
172.16.11.253

route add default gw
172.16.11.254
```

tux4.sh

```
#!/bin/bash

ifconfig eth0 down

ifconfig eth0 up 172.16.10.254/24

ifconfig eth1 down

ifconfig eth1 up 172.16.11.253/24

route add default gw 172.16.11.254

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

echo 0 >
/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

Código:

makefile

```
all: ftp.o download

ftp.o: ftp.c
    gcc -c ftp.c

download: ftp.h ftp.o
download.c
    gcc -o download ftp.o
download.c

splint:
    splint ftp.c download.c

run:
    download

clean:
    rm -f download *.o
```

ftp.h

```
#include <stdio.h>

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <netdb.h>
#include <strings.h>
#include <string.h>
#include <termios.h>

typedef struct FTP {
    int control_socket_fd;
    int data_socket_fd;
} ftp;

char* read_Sock(FILE* fp, char* code);
int connect_Sock(char* ip, int port);
void connect_ftp(ftp* ftp, char* ip, int port);
void login_Sock(ftp* ftp, char* user, char*
pass);

void write_Sock(ftp* ftp, char* str);
```


ftp.c

```
#include "ftp.h"

char* read_Sock(FILE* fp, char* code) {

    size_t bufsize = 52, c;

    char* msg = (char *)malloc(bufsize * sizeof(char));

    char* msgTotal = (char *)malloc(bufsize * sizeof(char));

    do {

        msg = fgets(msg, 5, fp);

    } while (!('1' <= msg[0] && msg[0] <= '5') || msg[3] != ' ');

    if(strcmp(msg, code) != 0) {

        fprintf(stderr, "usage: Wrong Code\n");

        exit(1);

    }

    getline(&msgTotal, &bufsize, fp);

    strcat(msg, msgTotal);

    return msg;

}

int connect_Sock(char* ip, int port) {

    int sockfd;
```

```

    struct sockaddr_in server_addr;

    /*server address handling*/

    bzero((char*)&server_addr,sizeof(server_addr));

    server_addr.sin_family = AF_INET;

    server_addr.sin_addr.s_addr = inet_addr(ip); /*32 bit
Internet address network byte ordered*/

    server_addr.sin_port = htons(port); /*server TCP
port must be network byte ordered */

    /*open an TCP socket*/

    if ((sockfd = socket(AF_INET,SOCK_STREAM,0)) < 0) {

        perror("socket()");

        exit(0);

    }

    /*connect to the server*/

    if(connect(sockfd,

                (struct sockaddr *)&server_addr,

                sizeof(server_addr)) < 0){

        perror("connect()");

        exit(0);

    }

    return sockfd;

}

void connect_ftp(ftp* ftp, char* ip, int port) {

    int sockfd;

    if((sockfd = connect_Sock(ip, port)) < 0) {

        fprintf(stderr,"usage: Error connecting socket\n");

        exit(1);

    }

    ftp->control_socket_fd = sockfd;

```

```

ftp->data_socket_fd = 0;

char rd[1024];

FILE* fp = fdopen(ftp->control_socket_fd, "r");


read_Sock(fp, "220 ");


}


void write_Sock(ftp* ftp, char* str) {
    int nBytes = 0;

    if ((nBytes = write(ftp->control_socket_fd, str,
strlen(str))) <= 0) {
        fprintf(stderr, "usage: Number of Bytes Written wrong
%d\n", nBytes);
        exit(1);
    }

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

```


download.c

```
#include <stdio.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <netdb.h>

#include <strings.h>

#include <string.h>

#include <termios.h>


#include "ftp.h"


char* processElementUntilChar(char* str, char chr) {

    // using temporary string to process substrings

    char* tempStr = (char*) malloc(strlen(str));


    // calculating length to copy element

    int index = strlen(str) - strlen(strcpy(tempStr, strchr(str,
chr)));


    tempStr[index] = '\0'; // termination char in the end of
string

    strncpy(tempStr, str, index);

    strcpy(str, str + strlen(tempStr) + 1);
```

```

        return tempStr;
    }

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

    if (argc != 2) {
        fprintf(stderr, "usage: wrong number of arguments\n");
        exit(1);
    }

    char* url_init = malloc(6 * sizeof(char));
    memcpy(url_init, argv[1], 6);

    if(strcmp(url_init, "ftp://\0") != 0) {
        fprintf(stderr, "usage: start the url with ftp://\n");
        exit(1);
    }

    char** url_aux = malloc(3 * sizeof(char *));
    int size = strlen(argv[1]) - 6;
    url_aux[0] = malloc(size);
    memcpy(url_aux[0], argv[1] + 6, size);
    url_aux[0][size] = '\0';

    char* user = malloc(strlen(url_aux[0]));
    char* pass = malloc(strlen(url_aux[0]));
    user[0] = '\0';
    pass[0] = '\0';

    if(url_aux[0][0] == '[') {

```

```

memcpy(user, url_aux[0] + 1, strlen(url_aux[0]));

strtok(user, ":");


if(strlen(user) == strlen(url_aux[0])-1) {
    fprintf(stderr, "usage: Declare an user\n");
    exit(1);
}

printf("USER: %s\n", user);


memcpy(pass, url_aux[0] + 2 + strlen(user),
strlen(url_aux[0]));
strtok(pass, "@");


if(strlen(pass) == strlen(url_aux[0]) - 2 -
strlen(user)) {
    fprintf(stderr, "usage: Declare a password\n");
    exit(1);
}

printf("PASS: %s\n", pass);
}


char** url_aux_2 = malloc(3 * sizeof(char *));
size = 0;


if(strlen(user) == 0) {
    size = strlen(url_aux[0]);
    url_aux_2[0] = malloc(size);
    memcpy(url_aux_2[0], url_aux[0], size);
}

```

```

    } else {

        size = strlen(url_aux[0]);

        url_aux_2[0] = malloc(size);

        memcpy(url_aux_2[0], url_aux[0] + strlen(user) +
strlen(pass) + 4, size);
    }


url_aux_2[0][size] = '\0';


char* host = malloc(strlen(url_aux_2[0]));
memcpy(host, url_aux_2[0], strlen(url_aux_2[0]));
strtok(host, "/");


if(strlen(host) == strlen(url_aux_2[0])) {

    fprintf(stderr, "usage: Declare an host\n");

    exit(1);

}

printf("HOST: %s\n", host);


size = strlen(url_aux_2[0]) - strlen(host) + 1;
char** url_aux_3 = (char **)malloc(3 * sizeof(char *));
url_aux_3[0] = (char *)malloc(size);
memcpy(url_aux_3[0], url_aux_2[0] + strlen(host) + 1, size);
url_aux_3[0][size] = '\0';


char* path = malloc(strlen(url_aux_3[0]));
memcpy(path, url_aux_3[0], strlen(url_aux_3[0]));

```

```

    if(strlen(path) == 0) {
        fprintf(stderr, "usage: Declare an path\n");
        exit(1);
    }

    printf("PATH: %s\n", path);


    struct hostent *h;

    if ((h=gethostbyname(host)) == NULL) {
        perror("gethostbyname");
        exit(1);
    }

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

    ftp ftp;
    connect_ftp(&ftp, ip, 21);

    /*send a string to the server*/

    if(strlen(user) == 0) {
        strcpy(user, "anonymous");
        user[strlen(user)] = '\0';
        strcpy(pass, "");
        pass[strlen(pass)] = '\0';
    }

```

```

char* write_user = malloc(strlen(user) + 1);
char* write_pass = malloc(strlen(pass) + 1);
printf("-----\n");
sprintf(write_user, "USER %s\r\n", user);
sprintf(write_pass, "PASS %s\r\n", pass);
write_user[strlen(write_user)] = '\0';
write_pass[strlen(write_pass)] = '\0';

int nBytes = 0;

FILE* fp = fdopen(ftp.control_socket_fd, "r");
write_Sock(&ftp, write_user);
char* read_user = malloc(52);
read_user = read_Sock(fp, "331 ");
printf("%s\n", read_user);

write_Sock(&ftp, write_pass);
char* read_pass = malloc(52);
read_pass = read_Sock(fp, "230 ");
printf("%s\n", read_pass);

char* write_pasv = malloc(6);
sprintf(write_pasv, "PASV\r\n");

```

```

write_Sock(&ftp,write_pasv);

char* read_pasv = malloc(52);

read_pasv = read_Sock(fp, "227 ");

printf("READ_PASV: %s",read_pasv);


// starting process information

int ipPart1, ipPart2, ipPart3, ipPart4;

int port1, port2;

if ((sscanf(read_pasv, "227 Entering Passive Mode
(%d,%d,%d,%d,%d,%d)", &ipPart1,
&ipPart2, &ipPart3, &ipPart4, &port1, &port2))
< 0) {

    printf("ERROR: Cannot process information to
calculating port.\n");
    return 1;

}


// cleaning buffer

memset(read_pasv, 0, sizeof(read_pasv));


// forming ip

if ((sprintf(read_pasv, "%d.%d.%d.%d", ipPart1, ipPart2,
ipPart3, ipPart4))
< 0) {

    printf("ERROR: Cannot form ip address.\n");
    return 1;

}


// calculating new port

int portResult = port1 * 256 + port2;

```

```

printf("IP: %s\n", read_pasv);

printf("PORT: %d\n", portResult);


if ((ftp.data_socket_fd = connect_Sock(read_pasv,
portResult)) < 0) {
    printf(
        "ERROR: Incorrect file descriptor
associated to ftp data socket fd.\n");
    return 1;
}


char* write_retr = malloc(6 + strlen(path));
sprintf(write_retr, "RETR /%s\r\n", path);
write_Sock(&ftp, write_retr);

char* read_retr = malloc(52);
read_retr = read_Sock(fp, "150 ");
printf("%s\n", read_retr);


char* element = (char*) malloc(strlen(url_aux_3[0]));
int startPath = 1;
while (strchr(url_aux_3[0], '/')) {
    element = processElementUntilChar(url_aux_3[0], '/');


    if (startPath) {
        startPath = 0;
        strcpy(path, element);
    } else {

```



```

        strcat(path, element);
    }

    strcat(path, "/");
}

char* filename = malloc(strlen(url_aux_3[0]));
strcpy(filename, url_aux_3[0]);

FILE* file;
int bytes;

if (!(file = fopen(filename, "w"))) {
    printf("ERROR: Cannot open file.\n");
    return 1;
}

char buf[1024];
while ((bytes = read(ftp.data_socket_fd, buf, sizeof(buf)))
{
    if (bytes < 0) {
        printf("ERROR: Nothing was received from data
socket fd.\n");
        return 1;
    }

    if ((bytes = fwrite(buf, bytes, 1, file)) < 0) {
        printf("ERROR: Cannot write data in file.\n");
        return 1;
    }
}

```

```

        }

    }

    fclose(file);
close(ftp.data_socket_fd);


char disc[1024];
char* read_aux = malloc(52);
read_aux = read_Sock(fp, "226 ");
printf("%s\n",read_aux);


sprintf(disc, "QUIT\r\n");
write_Sock(&ftp,disc);
char* read_quit = malloc(52);
read_quit = read_Sock(fp, "221 ");
printf("%s\n",read_quit);


if (ftp.control_socket_fd)
close(ftp.control_socket_fd);


free(write_pasv);
free(user);
free(pass);
free(write_user);
free(write_pass);
free(host);
free(path);

```

```
int i = 0;

for(i = 0; i < 3; i++) {

    free(url_aux[i]);

    free(url_aux_2[i]);

    free(url_aux_3[i]);

}

free(url_aux);

free(url_aux_2);

free(url_aux_3);

free(url_init);

}
```