2º Trabalho Laboratorial – Rede de Computadores

Redes de Computadores

3º ano, 1º semestre, L.EIC 22/23

Turma 15:

Alexandre Costa up202005319
António Santos up201907156
Isabel Silva up201904925

Índice

Sumário **3**

Introdução **3**

Parte 1 – Aplicação de Download **3**
- Arquitetura 3
- Relatório de um Download Bem Sucedido 4

Parte 2 – Configuração e Análise de uma Rede **4**
- Experiência 1 – Configurar uma Rede IP 4
- Experiência 2 – Implementação de Duas Bridges num Switch 5
- Experiência 3 – Configuração de um Router em Linux 5
- Experiência 4 – Configuração de um Router Comercial e Implementação da NAT 6
- Experiência 5 – DNS 7
- Experiência 6 – Conexões TCP 7

Conclusões **7**

Referências **8**

Anexos **8**
- Código da Aplicação 8
- Experiências 18
  - Experiência 1 18
  - Experiência 2 18
  - Experiência 3 19
  - Experiência 4 19
  - Experiência 5 19
  - Experiência 6 20

Sumário

Este trabalho foi realizado no âmbito da unidade curricular de Redes de Computadores do 3º ano da Licenciatura em Engenharia Informática e Computação e é dividido em duas partes.

Na primeira parte, é desenvolvida uma aplicação de download de ficheiros por FTP. A segunda parte consiste num conjunto de 6 experiências percorrendo as várias etapas da configuração de uma rede.

Introdução

Este projeto desenvolvido maioritariamente nos laboratórios, durante as aulas práticas, tem como objetivo a melhor compreensão de uma Rede de Computadores.

Para o desenvolvimento da aplicação de download foi utilizada a linguagem C. Esta aplicação permite a correta transferência do ficheiro pretendido e é capaz de lidar com erros e inputs inválidos por parte do utilizador.

Na segunda parte do projeto, todas as experiências foram concluídas com sucesso, excluindo os passos 4 e 5 da experiência 6, sendo, por isso, os objetivos devidamente cumpridos quase na totalidade. No presente relatório será possível observar as respostas às questões presentes no guião ao longo da *Parte 2 – Configuração e Análise de uma Rede*.

Parte 1 – Aplicação de Download

Arquitetura

```c
struct connection{
    char* user;
    char* password;
    char* host;
    char* urlPath;
    char* filename;
    int annonymous;
    int ctrlfd;
    int datafd;
};
```

```c
void initConnection (struct connection *connection);
int parseInput(struct connection *connection, char* input);
int getIP(char* ip, char* host);
int createConnection(char* addr, int port);
int checkResponse(int sockfd, char * expectedResponse, char* response);
int login(int sockfd, char * username, char * password, int annonymous);
int sendCommand(int sockfd, char * command, char* args, int hasArg);
int enterPassiveMode(int sockfd, struct connection *connection);
int transfer(int sockfd, char * name);
```

Em primeiro lugar, é feito o parse (processamento) do URL que é passado quando se corre a aplicação. na função *parseInput* também são guardados os valores user (ou anonymous, password, host, path do ficheiro e o nome do ficheiro (valores que estão na *struct connection*.

Seguidamente, utiliza-se a função getIP (que foi fornecida) para a qual se envia o nome do host e é retornado o endereço IP. Após a obtenção deste endereço IP, é chamada a função createConnection que permite a conexão com o socket. Para haver certezas de que não ocorreram problemas, é chamada a função checkResponse, que compara a resposta recebida pelo socket, com a resposta esperada, verificando possíveis erros.

Depois de aberto o socket, é feito o login na aplicação. Neste caso, o utilizador pode escolher entrar de forma anónima (*anonymous*) ou autenticar-se com um *user* e uma *password*. O que foi descrito neste parágrafo é feito na função login.

Posteriormente, na função sendCommand, é verificado o comando existente e, caso exista, este é enviado. Em seguida, é feita a entrada em modo passivo na função *enterPassiveMode*.

Se tudo isto correr sem erros ocorrerem, então é feita a transferência do ficheiro . Após a transferência o ficheiro é fechado. Com a chamada desta última função *transfer*, termina a nossa aplicação.

Relatório de um Download Bem Sucedido

Após correr o programa e executar uma transferência, obtemos estes resultados:

```
$ ./download ftp://ftp.up.pt/pub/kodi/timestamp.txt
Parsing input...

/timestamp.txt
User: anonymous
Password: (null)
Host name: ftp.up.pt
File path: /pub/kodi/timestamp.txt
Host name  : mirrors.up.pt
IP Address : 193.137.29.15

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

Login...
Sending command...
USER anonymous

Receiving response...
331 Please specify the password.

Sending command...
PASS

Receiving response...
230 Login successful.

Sending command...
PASV

Receiving response...
227 Entering Passive Mode (193,137,29,15,226,242).

Sending command...
RETR /pub/kodi/timestamp.txt

Receiving response...
150 Opening BINARY mode data connection for /pub/kodi/timestamp.txt (11 bytes).

Downloading timestamp.txt...
Receiving response...
226 Transfer complete.
```

Como é possível observar, a transferência foi efetuada com sucesso. Testou-se também o programa com diferentes ficheiros, com ou sem credenciais, obtendo-se sempre um resultado positivo.

Parte 2 – Configuração e Análise de uma Rede

As imagens referentes às experiências descritas de seguida estão presentes nos Anexos do presente relatório.

Experiência 1 – Configurar uma Rede IP

O objetivo desta experiência é a configuração de duas máquinas. Para isto, foi necessário atribuir um IP a cada uma delas. Foi, também, possível observar a troca de mensagens entre elas.

Comandos importantes:

- ifconfig <interface> <IP>/<máscara>
- route add -net <destino>/<máscara> gw <gatewayFinal>
- ping <destino>

O protocolo ARP (Address Resolution Protocol) mapeia um endereço IP de uma máquina a um endereço MAC de uma máquina na rede local. Quando um computador tenta enviar ao outro um pacote na mesma rede local, enviará um pacote ARP, por broadcast, para todas as máquinas na rede local perguntando qual delas tem aquele endereço MAC correspondente ao IP do destinatário. Este enviará mais um pacote ARP que indique à máquina qual o seu endereço MAC. Após isto, a transferência pode proceder.

Quando uma máquina tenta enviar um pacote a outra, como a entrada da tabela ARP que corresponde à máquina do recetor for apagada, a máquina que envia não sabe o endereço MAC deste. Assim, envia um pacote ARP por broadcast, contendo o seu endereço IP e o endereço MAC. De seguida, o recetor envia um pacote ARP com o seu endereço MAC e o seu endereço IP. Desta forma, é possível concluir que cada pacote ARP contém o endereço MAC e o endereço IP da máquina que envia, bem como para a máquina que recebe.

O comando ping é utilizado para testar a conectividade entre os computadores e pode gerar pacotes ARP onde obtém o endereço MAC e pacotes ICMP que fornece relatório de erros ao host.

Para obtermos o tipo da trama Ethernet temos de analisar os 2 bytes do header da trama Ethernet. Se o valor da trama for 0x0806 trata-se de uma trama do tipo ARP. No caso do valor ser 0x0800 representa um pacote do tipo IP. Sendo as tramas ICMP um subprotocolo do protocolo IP, para distinguirmos estas duas mesmas teremos de avaliar o IP header. No caso do IP header ser 1 trata-se de uma trama ICMP senão trata-se de uma trama IP.

A interface loopback é uma interface de rede virtual que permite que o computador comunique com ele mesmo. É um mecanismo utilizado para testar a correta configuração da rede, permitindo a existência de um IP sempre ativo no router.

Experiência 2 – Implementação de Duas *Bridges* num *Switch*

Nesta experiência o objetivo principal é a criação de duas bridges no switch. Os computadores tuxY3 e o tuxY4 foram adicionados à bridge Y0 e o tuxY2 à bridgeY1. Começámos por ligar o tuxY2 a uma porta do switch e procedemos à configuração do mesmo utilizando os seguintes comandos:

ifconfig eth0 up
ifconfig eth0 172.16.Y1.1/24
ifconfig eth0

Para configurar a bridgeY0 começámos por ligar um dos tux à porta do console do switch e utilizando o GtkTerm procedeu-se à criação da bridge utilizando o comando

/interface bridge add name=bridgeY0. Depois da criação das bridges removeram-se as portas onde tuxY3, tuxY4 e tuxY2 estão ligadas á bridge default e adicionaram-se as portas correspondentes a bridgeY0 e bridgeY1 utilizando os seguintes comandos:

/interface bridge porte remove [find interface=ether1]
/interface bridge porte remove [find interface=ether9]
/interface bridge porte remove [find interface=ether17]
/interface bridge port add bridge=bridgeY0 interface=ether1
/interface bridge port add bridge=bridgeY0 interface=ether9
/interface bridge port add bridge=bridgeY1 interface=ether17

Depois foi-nos pedido para dar **ping broadcast**. Através dos logs, é possível concluir que existem dois domínios de broadcast, sendo que o **ping broadcast** só chega ao tuxY4, ou seja, as duas *VLANS* têm domínios diferentes, um com o tuxY3 e tuxY4 e outro com o tuxY2.

Experiência 3 – Configuração de um *Router* em Linux

Esta experiência tinha como objetivo a configuração do tuxY4 como um router entre as duas bridges criadas na experiência anterior.

Algumas das routes são criadas automaticamente, conectando as máquinas às bridges a que pertencem. Neste caso, o gateway para estas routes é o 0.0.0.0. No entanto, é também possível adicionar routes com o comando *route add -net <destino>/<máscara> gw <gatewayFinal>*, quando pretendemos “ligar” duas máquinas.

A tabela forwarding pode ser obtida através do comando route -n e cada entrada possui informação sobre o destino da rota (**Destination**), o endereço ip pela qual será transmitida a mensagem entre a origem e o destino (**Gateway**), a **Netmask** que é utilizado para determinar o ID da rede a partir do endereço IP do destino, a **Interface** para a qual o pacote será enviado e outros elementos menos relevantes para as nossas experiências.

Utilizando o comando ping é possível analisar a sequência existente entre mensagens ARP e endereços MAC. Na tentativa de um tux dar ping a outro e o tux emissor desconhecer o endereço MAC do tux recetor, este envia uma mensagem ARP para saber qual é o endereço MAC correspondente ao ip. É também possível observar *ICMP Packets* do tipo request e reply, cujos *IP* e *MAC Adresses* associados pertencem aos tuxesY3 e Y2, sendo que são estas as duas máquinas envolvidas no **ping**.

Experiência 4 – Configuração de um Router Comercial e Implementação da NAT

Esta experiência teve como principal objetivo a configuração de uma route comercial, efetuando a ligação à rede do laboratório. Este router foi configurado de modo a ser possível a utilização da técnica NAT para ser possível garantir a conexão entre as máquinas da rede e a internet. Para além de compreender o funcionamento de tudo isto, outro dos objetivos desta experiência foi a compreensão e análise do modo de funcionamento dos pacotes ICMP.

Alguns dos principais comandos são os de configuração do router, e os da NAT também presentes no guião.

Para configurar uma route estática num router comercial é necessário, em primeiro lugar, iniciar sessão através do gtkterm. Para configurar as routes é preciso correr o comando ip route no gtkterm. Este comando funciona da seguinte forma: /ip route add dst-address=172.16.Y0.0/24 gateway=172.16.Y1.253, segundo o guião deste projeto.

No passo 4 desta experiência, nós desativamos os redirects no tuxY2 e apagamos a route para a rede 172.16.Y0.0/24 vai o tuxY4, ou seja, o tuxY2 vai ter de enviar os packets para o router e depois o router envia para a rede 172.16.Y0.0/24 (isto acontece porque o router é a route default do tuxY2). Por outro lado, quando reativamos os redirects, depois da primeira vez que o que foi previamente mencionado ocorre, é enviado um pacote ICMP de redirect que serve para adicionar na forwarding table do tuxY2 uma entrada que diz que pode usar a gateway 172.16.Y1.253 para chegar ao destino pretendido.

Para se conseguir configurar o NAT num router comercial, foi configurada a interface da NAT, seguindo os passos do guião do presente projeto, correndo todos os comandos na ordem correta.

O NAT (Network Address Translation) é um protocolo que visa a associação e transformação de um IP noutro IP, para mascarar o remetente ou destinatário dos pacotes que foram enviados. Isto pode ser útil para garantir a privacidade/segurança das máquinas numa rede privada que esteja a comunicar com máquinas que são externas à rede. O NAT opera num router como o que foi configurado nesta mesma experiência. Este estabelece o ponto de ligação entre a rede local e a rede pública ou Internet. No caso de uma máquina da rede local querer enviar um pacote para um endereço de uma rede pública (tal como foi feito nesta experiência), o pacote é enviado para o router e este modifica o endereço de origem do pacote para o endereço exterior, mascarando o remetente do pacote. Este pacote é enviado para o destinatário, que responde com um novo pacote que tem como destino o remetente inicial. Quando o router recebe este pacote reenvia-o para a máquina da rede privada, mudando o destinatário do pacote para o seu endereço, tornando possível a comunicação entre a rede privada e a rede pública. Se o router não estivesse configurado com este protocolo, a máquina da rede pública, ao receber um pacote de uma rede privada, não saberia como responder para esse endereço IP.

Experiência 5 – DNS

Esta experiência teve como objetivo a conexão das máquinas de uma rede IP a um servidor DNS, que traduz hosts para endereços IP, verificando, também, a forma como as máquinas se conectam com a internet altera.

Primeiramente, tivemos de configurar o DNS e para isso foi necessário alterar o conteúdo do ficheiro **/etc/resolv.conf** para **nameserver 172.16.1.1**.

O host envia um pacote por DNS com o hostname, esperando que o seu IP seja retornado. A isto, o servidor responde com um novo pacote que contém o endereço IP do hostname que foi enviado.

Experiência 6 – Conexões TCP

Nesta última experiência, o objetivo foi observar o funcionamento e o comportamento do protocolo TCP, utilizando a aplicação que desenvolvemos.

Para correr o programa, compilamos o programa com *make* e depois escrevemos o comando **./download ftp://[<user>:<password>@]<host>/<url-path>**.

Nesta experiência, utilizando a nossa aplicação, são abertas duas conexões TCP. A primeira ocorre quando se entra em contacto com o servidor, através de onde se envia e recebe comandos para a preparação da transferência do ficheiro. O controlo de informação é transportado nesta conexão. A segunda conexão ocorre para ser feita a transferência do ficheiro.

Durante uma conexão TCP, existem 3 fases distintas. Em primeiro lugar é estabelecida a conexão. Depois disso ocorre a transferência de dados. No final, é terminada a conexão.

O mecanismo ARQ TCP consiste no controlo dos erros ocorridos na transmissão dos dados. Para isto são utilizados acknowledgment numbers, indicando a correta receção da trama, tamanho da janela, gama de pacotes recebidos, número de sequência e o número do pacote a ser enviado.

O TCP tem também um mecanismo de controlo de congestionamento, que funciona como uma janela que se ajusta baseada na quantidade de *ACKs* recebidos.

Apesar de não termos realizado os passos 4 e 5 desta experiência, na teoria, caso apareça uma segunda conexão TCP, a taxa de transmissão que já estava a ocorrer decresce, fazendo com que a taxa de transmissão seja dividida igualmente pelas duas conexões.

Conclusões

Este projeto teve como principal objetivo o desenvolvimento de uma aplicação, onde fosse possível fazer o download de um ficheiro através de conexões, utilizando os protocolos TCP e FTP, e a configuração de uma rede IP. Isto foi fulcral para compreendermos o funcionamento de várias máquinas e de vários dispositivos, tais como o router e o switch, bem como o reconhecimento de várias técnicas como NAT e DNS. Para além disto passamos a conhecer muito melhor os protocolos ICMP e ARP, e estruturas de dados como forwarding tables, tabelas ARP, entre outras, utilizadas para a comunicação entre duas máquinas.

Tendo em conta o guião do trabalho e os nossos resultados, concluímos que todos os objetivos propostos foram cumpridos, e o nosso conhecimento acerca da construção de uma rede de computadores foi devidamente aprofundado e aprimorado.

Referências

O trabalho foi realizado com base no guião e nos slides das aulas teóricas da Unidade Curricular de Redes de Computadores.

Anexos

Experiências

Experiência 1

6	8.358730783	HewlettP_61:24:92	Broadcast	ARP	42 Who has 172.16.30.254? Tell 172.16.30.1
7	8.358851467	HewlettP_5a:7d:74	HewlettP_61:24:92	ARP	60 172.16.30.254 is at 00:21:5a:5a:7d:74
8	8.358870255	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x33d2, seq=1/256, ttl=64 (reply in 9)
9	8.358959930	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x33d2, seq=1/256, ttl=64 (request in 8)
10	9.366777268	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x33d2, seq=2/512, ttl=64 (reply in 11)
11	9.366879445	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x33d2, seq=2/512, ttl=64 (request in 10)
12	10.009946053	Routerbo_1c:8e:14	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:13 Cost = 0 Port = 0x8002
13	10.390756499	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x33d2, seq=3/768, ttl=64 (reply in 14)
14	10.390858955	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x33d2, seq=3/768, ttl=64 (request in 13)
15	11.414759127	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x33d2, seq=4/1024, ttl=64 (reply in 16)
16	11.414862840	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x33d2, seq=4/1024, ttl=64 (request in 15)
17	12.012082627	Routerbo_1c:8e:14	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:13 Cost = 0 Port = 0x8002
18	12.438767831	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x33d2, seq=5/1280, ttl=64 (reply in 19)
19	12.438899551	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x33d2, seq=5/1280, ttl=64 (request in 18)
20	13.423802313	HewlettP_5a:7d:74	HewlettP_61:24:92	ARP	60 Who has 172.16.30.1? Tell 172.16.30.254
21	13.423823125	HewlettP_61:24:92	HewlettP_5a:7d:74	ARP	42 172.16.30.1 is at 00:21:5a:61:24:92
22	13.462746504	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x33d2, seq=6/1536, ttl=64 (reply in 23)
23	13.462841627	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x33d2, seq=6/1536, ttl=64 (request in 22)

Experiência 2

49	88.900837602	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x06a4, seq=1/256, ttl=64 (no response found!)
50	89.920188666	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x06a4, seq=2/512, ttl=64 (no response found!)
51	90.094544916	Routerbo_1c:8e:16	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:16 Cost = 0 Port = 0x8001
52	90.944183966	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x06a4, seq=3/768, ttl=64 (no response found!)
53	91.968203361	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x06a4, seq=4/1024, ttl=64 (no response found!)
54	92.095750028	Routerbo_1c:8e:16	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:16 Cost = 0 Port = 0x8001
55	92.992184903	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x06a4, seq=5/1280, ttl=64 (no response found!)
56	94.016182717	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x06a4, seq=6/1536, ttl=64 (no response found!)
57	94.097942206	Routerbo_1c:8e:16	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:16 Cost = 0 Port = 0x8001
58	95.040179903	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x06a4, seq=7/1792, ttl=64 (no response found!)
59	96.064182815	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x06a4, seq=8/2048, ttl=64 (no response found!)

13	16.920046677	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x0c23, seq=1/256, ttl=64 (no response found!)
14	17.940852046	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x0c23, seq=2/512, ttl=64 (no response found!)
15	18.019494798	Routerbo_1c:8e:14	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:14 Cost = 0 Port = 0x8001
16	18.960854869	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x0c23, seq=3/768, ttl=64 (no response found!)
17	19.984852195	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x0c23, seq=4/1024, ttl=64 (no response found!)
18	20.021740196	Routerbo_1c:8e:14	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:14 Cost = 0 Port = 0x8001
19	21.008851825	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x0c23, seq=5/1280, ttl=64 (no response found!)
20	22.023992717	Routerbo_1c:8e:14	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:14 Cost = 0 Port = 0x8001
21	22.032850757	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x0c23, seq=6/1536, ttl=64 (no response found!)
22	23.056855136	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x0c23, seq=7/1792, ttl=64 (no response found!)
23	24.026248800	Routerbo_1c:8e:14	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:14 Cost = 0 Port = 0x8001
24	24.080852741	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x0c23, seq=8/2048, ttl=64 (no response found!)
25	25.104848180	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x0c23, seq=9/2304, ttl=64 (no response found!)

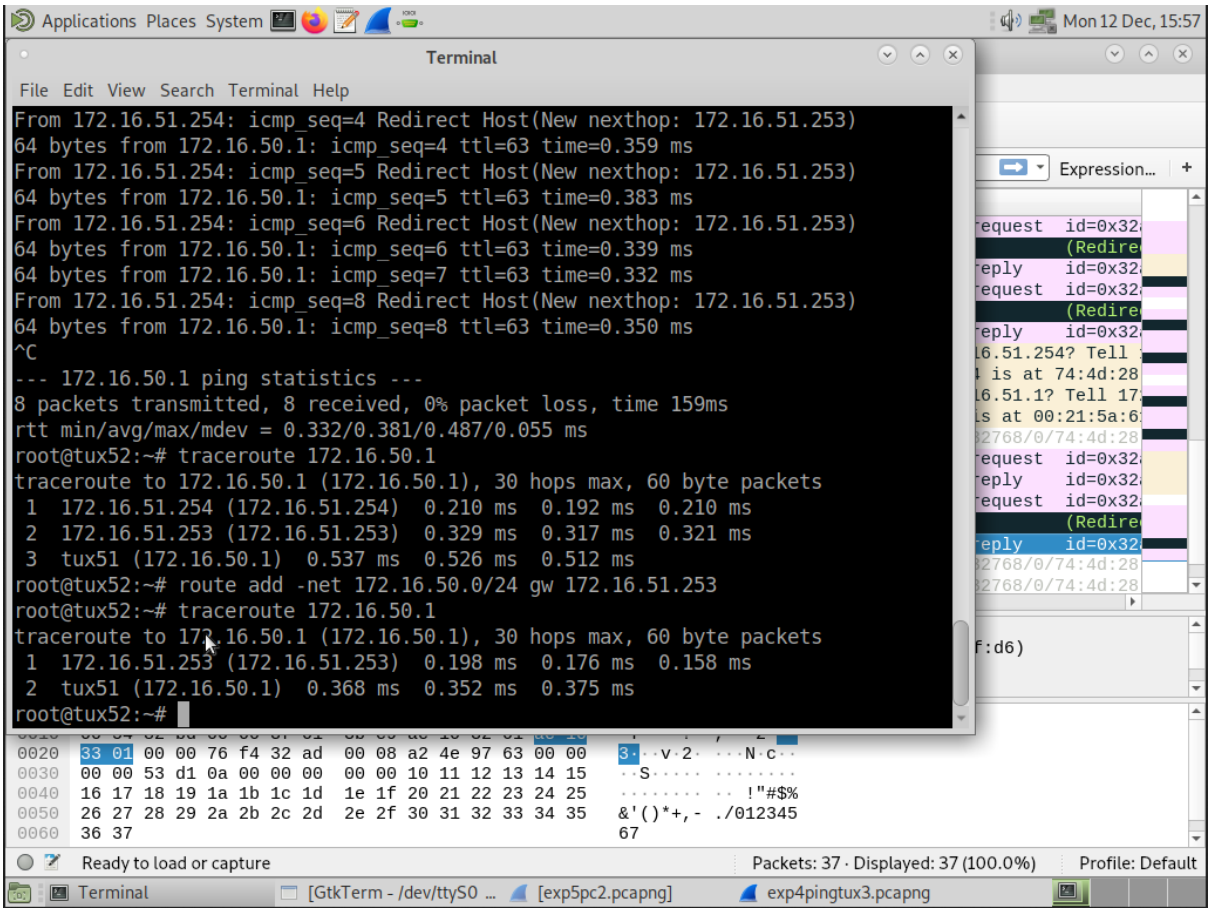
Experiência 3

61	104.330803257	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x4cec, seq=3/768, ttl=64 (reply in 62)
62	104.330952367	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x4cec, seq=3/768, ttl=63 (request in 61)
63	105.354783875	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x4cec, seq=4/1024, ttl=64 (reply in 64)
64	105.354931100	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x4cec, seq=4/1024, ttl=63 (request in 63)
65	106.117199576	Routerbo_1c:8e:13	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:13 Cost = 0 Port = 0x8001
66	106.378754506	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x4cec, seq=5/1280, ttl=64 (reply in 67)
67	106.378902708	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x4cec, seq=5/1280, ttl=63 (request in 66)
68	107.306676601	HewlettP_61:24:92	HewlettP_5a:7d:74	ARP	60 Who has 172.16.30.254? Tell 172.16.30.1
69	107.306696924	HewlettP_5a:7d:74	HewlettP_61:24:92	ARP	42 172.16.30.254 is at 00:21:5a:5a:7d:74
70	107.402711797	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x4cec, seq=6/1536, ttl=64 (reply in 71)
71	107.402872012	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x4cec, seq=6/1536, ttl=63 (request in 70)
72	107.482459738	HewlettP_5a:7d:74	HewlettP_61:24:92	ARP	42 Who has 172.16.30.1? Tell 172.16.30.254
73	107.482594601	HewlettP_61:24:92	HewlettP_5a:7d:74	ARP	60 172.16.30.1 is at 00:21:5a:61:24:92
74	108.119422543	Routerbo_1c:8e:13	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:13 Cost = 0 Port = 0x8001
75	108.426678377	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x4cec, seq=7/1792, ttl=64 (reply in 76)
76	108.426833285	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x4cec, seq=7/1792, ttl=63 (request in 75)
77	109.450658646	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x4cec, seq=8/2048, ttl=64 (reply in 78)
78	109.450807687	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x4cec, seq=8/2048, ttl=63 (request in 77)

62	100.349254883	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x4cec, seq=4/1024, ttl=63 (reply in 63)
63	100.349370610	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x4cec, seq=4/1024, ttl=64 (request in 62)
64	101.373224397	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x4cec, seq=5/1280, ttl=63 (reply in 65)
65	101.373341730	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x4cec, seq=5/1280, ttl=64 (request in 64)
66	102.112738418	Routerbo_1c:8e:17	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:16 Cost = 0 Port = 0x8002
67	102.397171072	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x4cec, seq=6/1536, ttl=63 (reply in 68)
68	102.397313199	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x4cec, seq=6/1536, ttl=64 (request in 67)
69	102.476908118	KYE_25:26:0a	HewlettP_61:30:63	ARP	42 Who has 172.16.31.1? Tell 172.16.31.253
70	102.477013439	HewlettP_61:30:63	KYE_25:26:0a	ARP	60 172.16.31.1 is at 00:21:5a:61:30:63
71	102.488712494	HewlettP_61:30:63	KYE_25:26:0a	ARP	60 Who has 172.16.31.253? Tell 172.16.31.1
72	102.488717732	KYE_25:26:0a	HewlettP_61:30:63	ARP	42 172.16.31.253 is at 00:c0:df:25:26:0a
73	103.421147221	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x4cec, seq=7/1792, ttl=63 (reply in 74)
74	103.421273144	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x4cec, seq=7/1792, ttl=64 (request in 73)
75	104.114958801	Routerbo_1c:8e:17	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:16 Cost = 0 Port = 0x8002
76	104.445128607	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x4cec, seq=8/2048, ttl=63 (reply in 77)
77	104.445247197	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x4cec, seq=8/2048, ttl=64 (request in 76)

22	34.420684903	HewlettP_61:24:92	Broadcast	ARP	42 Who has 172.16.30.254? Tell 172.16.30.1
23	34.420863912	HewlettP_5a:7d:74	HewlettP_61:24:92	ARP	60 172.16.30.254 is at 00:21:5a:5a:7d:74
24	34.420882980	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x4c5a, seq=1/256, ttl=64 (reply in 25)
25	34.421005346	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x4c5a, seq=1/256, ttl=64 (request in 24)
26	35.429833484	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x4c5a, seq=2/512, ttl=64 (reply in 27)
27	35.429994054	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x4c5a, seq=2/512, ttl=64 (request in 26)
28	36.041388881	Routerbo_1c:8e:14	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:13 Cost = 0 Port = 0x8002
29	36.453831296	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x4c5a, seq=3/768, ttl=64 (reply in 30)
30	36.453957154	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x4c5a, seq=3/768, ttl=64 (request in 29)
31	37.477833787	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x4c5a, seq=4/1024, ttl=64 (reply in 32)
32	37.477964465	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x4c5a, seq=4/1024, ttl=64 (request in 31)
33	38.043687883	Routerbo_1c:8e:14	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:13 Cost = 0 Port = 0x8002
34	38.501838235	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x4c5a, seq=5/1280, ttl=64 (reply in 35)
35	38.501968703	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x4c5a, seq=5/1280, ttl=64 (request in 34)
36	39.525833533	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x4c5a, seq=6/1536, ttl=64 (reply in 37)
37	39.525961835	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x4c5a, seq=6/1536, ttl=64 (request in 36)
38	39.634390625	HewlettP_5a:7d:74	HewlettP_61:24:92	ARP	60 Who has 172.16.30.1? Tell 172.16.30.254
39	39.634412137	HewlettP_61:24:92	HewlettP_5a:7d:74	ARP	42 172.16.30.1 is at 00:21:5a:61:24:92
40	40.045980390	Routerbo_1c:8e:14	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:13 Cost = 0 Port = 0x8002
41	42.048285398	Routerbo_1c:8e:14	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:13 Cost = 0 Port = 0x8002
42	44.050576298	Routerbo_1c:8e:14	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:13 Cost = 0 Port = 0x8002
43	46.052866151	Routerbo_1c:8e:14	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:13 Cost = 0 Port = 0x8002
44	48.055158797	Routerbo_1c:8e:14	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:13 Cost = 0 Port = 0x8002
45	48.852749318	172.16.30.1	172.16.31.253	ICMP	98 Echo (ping) request id=0x4c64, seq=1/256, ttl=64 (reply in 46)
46	48.852915197	172.16.31.253	172.16.30.1	ICMP	98 Echo (ping) reply id=0x4c64, seq=1/256, ttl=64 (request in 45)

Experiência 4



5	6.054160909	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x32ad, seq=1/256, ttl=64 (reply in 6)
6	6.054625928	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x32ad, seq=1/256, ttl=63 (request in 5)
7	7.066647032	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x32ad, seq=2/512, ttl=64 (reply in 11)
8	7.066823038	Routerbo_eb:18:e6	Broadcast	ARP	60	Who has 172.16.51.1? Tell 172.16.51.254
9	7.066833654	HewlettP_61:2f:d6	Routerbo_eb:18:e6	ARP	42	172.16.51.1 is at 00:21:5a:61:2f:d6
10	7.066919352	172.16.51.254	172.16.51.1	ICMP	126	Redirect (Redirect for host)
11	7.067046956	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x32ad, seq=2/512, ttl=63 (request in 7)
12	8.007825950	Routerbo_1c:8b:bc	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/74:4d:28:eb:18:e6 Cost = 10 Port = 0x8002
13	8.086650791	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x32ad, seq=3/768, ttl=64 (reply in 15)
14	8.086817437	172.16.51.254	172.16.51.1	ICMP	126	Redirect (Redirect for host)
15	8.086998332	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x32ad, seq=3/768, ttl=63 (request in 13)
16	9.110646342	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x32ad, seq=4/1024, ttl=64 (reply in 18)
17	9.110799509	172.16.51.254	172.16.51.1	ICMP	126	Redirect (Redirect for host)
18	9.110979775	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x32ad, seq=4/1024, ttl=63 (request in 16)
19	10.010795980	Routerbo_1c:8b:bc	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/74:4d:28:eb:18:e6 Cost = 10 Port = 0x8002
20	10.134643311	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x32ad, seq=5/1280, ttl=64 (reply in 22)
21	10.134787818	172.16.51.254	172.16.51.1	ICMP	126	Redirect (Redirect for host)
22	10.135002517	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x32ad, seq=5/1280, ttl=63 (request in 20)
23	11.158648055	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x32ad, seq=6/1536, ttl=64 (reply in 25)
24	11.158787393	172.16.51.254	172.16.51.1	ICMP	126	Redirect (Redirect for host)
25	11.158962002	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x32ad, seq=6/1536, ttl=63 (request in 23)
26	11.254610054	HewlettP_61:2f:d6	Routerbo_eb:18:e6	ARP	42	Who has 172.16.51.254? Tell 172.16.51.1
27	11.254725505	Routerbo_eb:18:e6	HewlettP_61:2f:d6	ARP	60	172.16.51.254 is at 74:4d:28:eb:18:e6
28	11.266370525	KYE_08:d5:b0	HewlettP_61:2f:d6	ARP	60	Who has 172.16.51.1? Tell 172.16.51.253
29	11.266383027	HewlettP_61:2f:d6	KYE_08:d5:b0	ARP	42	172.16.51.1 is at 00:21:5a:61:2f:d6
30	12.012948168	Routerbo_1c:8b:bc	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/74:4d:28:eb:18:e6 Cost = 10 Port = 0x8002
31	12.182640668	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x32ad, seq=7/1792, ttl=64 (reply in 32)
32	12.182947421	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x32ad, seq=7/1792, ttl=63 (request in 31)
33	13.206633653	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x32ad, seq=8/2048, ttl=64 (reply in 35)
34	13.206780604	172.16.51.254	172.16.51.1	ICMP	126	Redirect (Redirect for host)
35	13.206965899	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x32ad, seq=8/2048, ttl=63 (request in 33)

11	6.367083279	172.16.50.1	172.16.51.254	ICMP	98 Echo (ping) request id=0x32d9, seq=4/1024, ttl=64 (reply in 12)
12	6.367391709	172.16.51.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x32d9, seq=4/1024, ttl=63 (request in 11)
13	7.387076519	172.16.50.1	172.16.51.254	ICMP	98 Echo (ping) request id=0x32d9, seq=5/1280, ttl=64 (reply in 14)
14	7.387347653	172.16.51.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x32d9, seq=5/1280, ttl=63 (request in 13)
15	7.998699964	Routerbo_1c:8b:bf	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bf Cost = 0 Port = 0x8001
16	8.391098279	HewlettP_c3:78:70	HewlettP_61:2d:72	ARP	60 Who has 172.16.50.1? Tell 172.16.50.254
17	8.391107499	HewlettP_61:2d:72	HewlettP_c3:78:70	ARP	42 172.16.50.1 is at 00:21:5a:61:2d:72
18	8.411055031	172.16.50.1	172.16.51.254	ICMP	98 Echo (ping) request id=0x32d9, seq=6/1536, ttl=64 (reply in 19)
19	8.411295154	172.16.51.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x32d9, seq=6/1536, ttl=63 (request in 18)
20	8.539034735	HewlettP_61:2d:72	HewlettP_c3:78:70	ARP	42 Who has 172.16.50.254? Tell 172.16.50.1
21	8.539154936	HewlettP_c3:78:70	HewlettP_61:2d:72	ARP	60 172.16.50.254 is at 00:21:5a:c3:78:70
22	9.435071242	172.16.50.1	172.16.51.254	ICMP	98 Echo (ping) request id=0x32d9, seq=7/1792, ttl=64 (reply in 23)
23	9.435333017	172.16.51.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x32d9, seq=7/1792, ttl=63 (request in 22)
24	10.000886279	Routerbo_1c:8b:bf	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bf Cost = 0 Port = 0x8001
25	10.459066415	172.16.50.1	172.16.51.254	ICMP	98 Echo (ping) request id=0x32d9, seq=8/2048, ttl=64 (reply in 26)
26	10.459321694	172.16.51.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x32d9, seq=8/2048, ttl=63 (request in 25)

4	4.639941959	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) request id=0x3420, seq=1/256, ttl=64 (reply in 5)
5	4.640242632	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) reply id=0x3420, seq=1/256, ttl=63 (request in 4)
6	5.669061016	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) request id=0x3420, seq=2/512, ttl=64 (reply in 7)
7	5.669325161	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) reply id=0x3420, seq=2/512, ttl=63 (request in 6)
8	6.006549512	Routerbo_1c:8b:bc	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/74:4d:28:eb:18:e6 Cost = 10 Port = 0x8002
9	6.693060446	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) request id=0x3420, seq=3/768, ttl=64 (reply in 10)
10	6.693360351	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) reply id=0x3420, seq=3/768, ttl=63 (request in 9)
11	7.717062930	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) request id=0x3420, seq=4/1024, ttl=64 (reply in 12)
12	7.717327005	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) reply id=0x3420, seq=4/1024, ttl=63 (request in 11)
13	8.008750080	Routerbo_1c:8b:bc	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/74:4d:28:eb:18:e6 Cost = 10 Port = 0x8002
14	8.741068605	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) request id=0x3420, seq=5/1280, ttl=64 (reply in 15)
15	8.741337779	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) reply id=0x3420, seq=5/1280, ttl=63 (request in 14)
16	9.765061410	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) request id=0x3420, seq=6/1536, ttl=64 (reply in 17)
17	9.765302926	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) reply id=0x3420, seq=6/1536, ttl=63 (request in 16)
18	10.010928988	Routerbo_1c:8b:bc	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/74:4d:28:eb:18:e6 Cost = 10 Port = 0x8002
19	10.789061108	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) request id=0x3420, seq=7/1792, ttl=64 (reply in 20)
20	10.789361851	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) reply id=0x3420, seq=7/1792, ttl=63 (request in 19)
21	11.813060856	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) request id=0x3420, seq=8/2048, ttl=64 (reply in 22)
22	11.813332125	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) reply id=0x3420, seq=8/2048, ttl=63 (request in 21)
23	12.012367905	Routerbo_1c:8b:bc	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/74:4d:28:eb:18:e6 Cost = 10 Port = 0x8002
24	12.837060235	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) request id=0x3420, seq=9/2304, ttl=64 (reply in 25)
25	12.837321237	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) reply id=0x3420, seq=9/2304, ttl=63 (request in 24)

2	1.191524028	172.16.50.1	172.16.51.254	ICMP	98 Echo (ping) request id=0x34a9, seq=1/256, ttl=64 (reply in 3)
3	1.191811981	172.16.51.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x34a9, seq=1/256, ttl=63 (request in 2)
4	2.002054478	Routerbo_1c:8b:bf	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bf Cost = 0 Port = 0x8001
5	2.207336109	172.16.50.1	172.16.51.254	ICMP	98 Echo (ping) request id=0x34a9, seq=2/512, ttl=64 (reply in 6)
6	2.207589282	172.16.51.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x34a9, seq=2/512, ttl=63 (request in 5)
7	3.231335476	172.16.50.1	172.16.51.254	ICMP	98 Echo (ping) request id=0x34a9, seq=3/768, ttl=64 (reply in 8)
8	3.231627061	172.16.51.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x34a9, seq=3/768, ttl=63 (request in 7)
9	4.004164718	Routerbo_1c:8b:bf	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bf Cost = 0 Port = 0x8001
10	4.255338257	172.16.50.1	172.16.51.254	ICMP	98 Echo (ping) request id=0x34a9, seq=4/1024, ttl=64 (reply in 11)
11	4.255583468	172.16.51.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x34a9, seq=4/1024, ttl=63 (request in 10)
12	5.279334325	172.16.50.1	172.16.51.254	ICMP	98 Echo (ping) request id=0x34a9, seq=5/1280, ttl=64 (reply in 13)
13	5.279594691	172.16.51.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x34a9, seq=5/1280, ttl=63 (request in 12)
14	6.006219742	Routerbo_1c:8b:bf	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bf Cost = 0 Port = 0x8001
15	6.303336322	172.16.50.1	172.16.51.254	ICMP	98 Echo (ping) request id=0x34a9, seq=6/1536, ttl=64 (reply in 16)
16	6.303586701	172.16.51.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x34a9, seq=6/1536, ttl=63 (request in 15)
17	7.327334678	172.16.50.1	172.16.51.254	ICMP	98 Echo (ping) request id=0x34a9, seq=7/1792, ttl=64 (reply in 18)
18	7.327595813	172.16.51.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x34a9, seq=7/1792, ttl=63 (request in 17)
19	8.008307617	Routerbo_1c:8b:bf	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bf Cost = 0 Port = 0x8001
20	8.351335122	172.16.50.1	172.16.51.254	ICMP	98 Echo (ping) request id=0x34a9, seq=8/2048, ttl=64 (reply in 21)
21	8.351628453	172.16.51.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x34a9, seq=8/2048, ttl=63 (request in 20)
22	9.375342401	172.16.50.1	172.16.51.254	ICMP	98 Echo (ping) request id=0x34a9, seq=9/2304, ttl=64 (reply in 23)
23	9.375603676	172.16.51.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x34a9, seq=9/2304, ttl=63 (request in 22)
24	10.010383296	Routerbo_1c:8b:bf	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bf Cost = 0 Port = 0x8001
25	10.399336822	172.16.50.1	172.16.51.254	ICMP	98 Echo (ping) request id=0x34a9, seq=10/2560, ttl=64 (reply in 26)
26	10.399606617	172.16.51.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x34a9, seq=10/2560, ttl=63 (request in 25)

11	6.195092061	172.16.50.1	172.16.51.254	ICMP	98 Echo (ping) request id=0x34ea, seq=4/1024, ttl=64 (reply in 12)
12	6.195338531	172.16.51.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x34ea, seq=4/1024, ttl=63 (request in 11)
13	7.219093044	172.16.50.1	172.16.51.254	ICMP	98 Echo (ping) request id=0x34ea, seq=5/1280, ttl=64 (reply in 14)
14	7.219382396	172.16.51.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x34ea, seq=5/1280, ttl=63 (request in 13)
15	8.008238691	Routerbo_1c:8b:bf	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bf Cost = 0 Port = 0x8001
16	8.243059157	HewlettP_61:2d:72	HewlettP_c3:78:70	ARP	42 Who has 172.16.50.254? Tell 172.16.50.1
17	8.243106300	172.16.50.1	172.16.51.254	ICMP	98 Echo (ping) request id=0x34ea, seq=6/1536, ttl=64 (reply in 19)
18	8.243191716	HewlettP_c3:78:70	HewlettP_61:2d:72	ARP	60 172.16.50.254 is at 00:21:5a:c3:78:70
19	8.243339779	172.16.51.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x34ea, seq=6/1536, ttl=63 (request in 17)
20	8.371057029	HewlettP_c3:78:70	HewlettP_61:2d:72	ARP	60 Who has 172.16.50.1? Tell 172.16.50.254
21	8.371076026	HewlettP_61:2d:72	HewlettP_c3:78:70	ARP	42 172.16.50.1 is at 00:21:5a:61:2d:72
22	9.267090414	172.16.50.1	172.16.51.254	ICMP	98 Echo (ping) request id=0x34ea, seq=7/1792, ttl=64 (reply in 23)
23	9.267337652	172.16.51.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x34ea, seq=7/1792, ttl=63 (request in 22)

3	3.168451325	172.16.50.1	172.16.50.254	ICMP	98 Echo (ping) request id=0x3247, seq=1/256, ttl=64 (reply in 4)
4	3.168613434	172.16.50.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x3247, seq=1/256, ttl=64 (request in 3)
5	4.004327535	Routerbo_1c:8b:bf	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bf Cost = 0 Port = 0x8001
6	4.176960600	172.16.50.1	172.16.50.254	ICMP	98 Echo (ping) request id=0x3247, seq=2/512, ttl=64 (reply in 7)
7	4.177116352	172.16.50.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x3247, seq=2/512, ttl=64 (request in 6)
8	5.200969091	172.16.50.1	172.16.50.254	ICMP	98 Echo (ping) request id=0x3247, seq=3/768, ttl=64 (reply in 9)
9	5.201099350	172.16.50.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x3247, seq=3/768, ttl=64 (request in 8)
10	6.006475029	Routerbo_1c:8b:bf	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bf Cost = 0 Port = 0x8001
11	6.224965126	172.16.50.1	172.16.50.254	ICMP	98 Echo (ping) request id=0x3247, seq=4/1024, ttl=64 (reply in 12)
12	6.225089658	172.16.50.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x3247, seq=4/1024, ttl=64 (request in 11)
13	7.248967632	172.16.50.1	172.16.50.254	ICMP	98 Echo (ping) request id=0x3247, seq=5/1280, ttl=64 (reply in 14)
14	7.249096215	172.16.50.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x3247, seq=5/1280, ttl=64 (request in 13)

15	10.237527458	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request id=0x32bf, seq=5/1280, ttl=64 (reply in 16)
16	10.237767442	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply id=0x32bf, seq=5/1280, ttl=63 (request in 15)
17	11.240317919	HewlettP_c3:78:70	HewlettP_61:2d:72	ARP	60 Who has 172.16.50.1? Tell 172.16.50.254
18	11.240337056	HewlettP_61:2d:72	HewlettP_c3:78:70	ARP	42 172.16.50.1 is at 00:21:5a:61:2d:72
19	11.261518389	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request id=0x32bf, seq=6/1536, ttl=64 (reply in 20)
20	11.261743286	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply id=0x32bf, seq=6/1536, ttl=63 (request in 19)
21	11.389489351	HewlettP_61:2d:72	HewlettP_c3:78:70	ARP	42 Who has 172.16.50.254? Tell 172.16.50.1
22	11.389609692	HewlettP_c3:78:70	HewlettP_61:2d:72	ARP	60 172.16.50.254 is at 00:21:5a:c3:78:70
23	12.013028079	Routerbo_1c:8b:bf	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bf Cost = 0 Port = 0x8001
24	12.285518102	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request id=0x32bf, seq=7/1792, ttl=64 (reply in 25)
25	12.285757666	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply id=0x32bf, seq=7/1792, ttl=63 (request in 24)

15	10.957508967	172.16.50.1	172.16.51.253	ICMP	98 Echo (ping) request id=0x32a8, seq=5/1280, ttl=64 (reply in 16)
16	10.957644184	172.16.51.253	172.16.50.1	ICMP	98 Echo (ping) reply id=0x32a8, seq=5/1280, ttl=64 (request in 15)
17	11.981469810	HewlettP_61:2d:72	HewlettP_c3:78:70	ARP	42 Who has 172.16.50.254? Tell 172.16.50.1
18	11.981523730	172.16.50.1	172.16.51.253	ICMP	98 Echo (ping) request id=0x32a8, seq=6/1536, ttl=64 (reply in 20)
19	11.981619416	HewlettP_c3:78:70	HewlettP_61:2d:72	ARP	60 172.16.50.254 is at 00:21:5a:c3:78:70
20	11.981634991	172.16.51.253	172.16.50.1	ICMP	98 Echo (ping) reply id=0x32a8, seq=6/1536, ttl=64 (request in 18)
21	12.012779354	Routerbo_1c:8b:bf	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bf Cost = 0 Port = 0x8001
22	12.087035618	HewlettP_c3:78:70	HewlettP_61:2d:72	ARP	60 Who has 172.16.50.1? Tell 172.16.50.254
23	12.087057619	HewlettP_61:2d:72	HewlettP_c3:78:70	ARP	42 172.16.50.1 is at 00:21:5a:61:2d:72
24	13.005506067	172.16.50.1	172.16.51.253	ICMP	98 Echo (ping) request id=0x32a8, seq=7/1792, ttl=64 (reply in 25)
25	13.005638701	172.16.51.253	172.16.50.1	ICMP	98 Echo (ping) reply id=0x32a8, seq=7/1792, ttl=64 (request in 24)
26	14.014894649	Routerbo_1c:8b:bf	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bf Cost = 0 Port = 0x8001
27	14.029507663	172.16.50.1	172.16.51.253	ICMP	98 Echo (ping) request id=0x32a8, seq=8/2048, ttl=64 (reply in 28)
28	14.029659434	172.16.51.253	172.16.50.1	ICMP	98 Echo (ping) reply id=0x32a8, seq=8/2048, ttl=64 (request in 27)

Experiência 5

14	6.042265570	8.8.8.8	172.16.51.253	ICMP	98 Echo (ping) reply id=0x18cd, seq=5/1280, ttl=109 (request in 13)
15	6.997344705	172.16.51.253	8.8.8.8	ICMP	98 Echo (ping) request id=0x18cd, seq=6/1536, ttl=64 (reply in 18)
16	7.038089411	Routerbo_eb:18:e6	KYE_08:d5:b0	ARP	60 Who has 172.16.51.253? Tell 172.16.51.254
17	7.038100795	KYE_08:d5:b0	Routerbo_eb:18:e6	ARP	42 172.16.51.253 is at 00:c0:df:08:d5:b0
18	7.043098498	8.8.8.8	172.16.51.253	ICMP	98 Echo (ping) reply id=0x18cd, seq=6/1536, ttl=109 (request in 15)
19	7.221341485	KYE_08:d5:b0	Routerbo_eb:18:e6	ARP	42 Who has 172.16.51.254? Tell 172.16.51.253
20	7.221445060	Routerbo_eb:18:e6	KYE_08:d5:b0	ARP	60 172.16.51.254 is at 74:4d:28:eb:18:e6
21	7.998142571	172.16.51.253	8.8.8.8	ICMP	98 Echo (ping) request id=0x18cd, seq=7/1792, ttl=64 (reply in 23)
22	7.998418514	Routerbo_1c:8b:bb	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/74:4d:28:eb:18:e6 Cost = 10 Port = 0x8001
23	8.043942809	8.8.8.8	172.16.51.253	ICMP	98 Echo (ping) reply id=0x18cd, seq=7/1792, ttl=109 (request in 21)
24	8.998992400	172.16.51.253	8.8.8.8	ICMP	98 Echo (ping) request id=0x18cd, seq=8/2048, ttl=64 (reply in 25)
25	9.044834962	8.8.8.8	172.16.51.253	ICMP	98 Echo (ping) reply id=0x18cd, seq=8/2048, ttl=109 (request in 24)

5	5.475084645	Routerbo_eb:18:e6	Broadcast	ARP	60 Who has 172.16.51.1? Tell 172.16.51.254
6	5.475098753	HewlettP_61:2f:d6	Routerbo_eb:18:e6	ARP	42 172.16.51.1 is at 00:21:5a:61:2f:d6
7	5.475185775	8.8.8.8	172.16.51.1	ICMP	98 Echo (ping) reply id=0x2f24, seq=1/256, ttl=109 (request in 4)
8	6.006339751	Routerbo_1c:8b:bc	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/74:4d:28:eb:18:e6 Cost = 10 Port = 0x8002
9	6.428243107	172.16.51.1	8.8.8.8	ICMP	98 Echo (ping) request id=0x2f24, seq=2/512, ttl=64 (reply in 10)
10	6.474280594	8.8.8.8	172.16.51.1	ICMP	98 Echo (ping) reply id=0x2f24, seq=2/512, ttl=109 (request in 9)
11	7.428335049	172.16.51.1	8.8.8.8	ICMP	98 Echo (ping) request id=0x2f24, seq=3/768, ttl=64 (reply in 12)
12	7.474149533	8.8.8.8	172.16.51.1	ICMP	98 Echo (ping) reply id=0x2f24, seq=3/768, ttl=109 (request in 11)
13	8.008428348	Routerbo_1c:8b:bc	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/74:4d:28:eb:18:e6 Cost = 10 Port = 0x8002
14	8.429206351	172.16.51.1	8.8.8.8	ICMP	98 Echo (ping) request id=0x2f24, seq=4/1024, ttl=64 (reply in 15)
15	8.475138587	8.8.8.8	172.16.51.1	ICMP	98 Echo (ping) reply id=0x2f24, seq=4/1024, ttl=109 (request in 14)
16	9.429086255	172.16.51.1	8.8.8.8	ICMP	98 Echo (ping) request id=0x2f24, seq=5/1280, ttl=64 (reply in 17)
17	9.475080789	8.8.8.8	172.16.51.1	ICMP	98 Echo (ping) reply id=0x2f24, seq=5/1280, ttl=109 (request in 16)
18	10.010569115	Routerbo_1c:8b:bc	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/74:4d:28:eb:18:e6 Cost = 10 Port = 0x8002
19	10.429129099	172.16.51.1	8.8.8.8	ICMP	98 Echo (ping) request id=0x2f24, seq=6/1536, ttl=64 (reply in 20)
20	10.475007068	8.8.8.8	172.16.51.1	ICMP	98 Echo (ping) reply id=0x2f24, seq=6/1536, ttl=109 (request in 19)
21	10.553062871	HewlettP_61:2f:d6	Routerbo_eb:18:e6	ARP	42 Who has 172.16.51.254? Tell 172.16.51.1
22	10.553179157	Routerbo_eb:18:e6	HewlettP_61:2f:d6	ARP	60 172.16.51.254 is at 74:4d:28:eb:18:e6
23	11.430055854	172.16.51.1	8.8.8.8	ICMP	98 Echo (ping) request id=0x2f24, seq=7/1792, ttl=64 (reply in 24)
24	11.477506370	8.8.8.8	172.16.51.1	ICMP	98 Echo (ping) reply id=0x2f24, seq=7/1792, ttl=109 (request in 23)

19	8.439664346	172.16.50.1	8.8.8.8	ICMP	98 Echo (ping) request id=0x2f03, seq=6/1536, ttl=64 (reply in 22)
20	8.463618927	HewlettP_61:2d:72	HewlettP_c3:78:70	ARP	42 Who has 172.16.50.254? Tell 172.16.50.1
21	8.463733197	HewlettP_c3:78:70	HewlettP_61:2d:72	ARP	60 172.16.50.254 is at 00:21:5a:c3:78:70
22	8.485652711	8.8.8.8	172.16.50.1	ICMP	98 Echo (ping) reply id=0x2f03, seq=6/1536, ttl=108 (request in 19)
23	8.513449587	HewlettP_c3:78:70	HewlettP_61:2d:72	ARP	60 Who has 172.16.50.1? Tell 172.16.50.254
24	8.513455105	HewlettP_61:2d:72	HewlettP_c3:78:70	ARP	42 172.16.50.1 is at 00:21:5a:61:2d:72
25	9.440706532	172.16.50.1	8.8.8.8	ICMP	98 Echo (ping) request id=0x2f03, seq=7/1792, ttl=64 (reply in 26)
26	9.486574616	8.8.8.8	172.16.50.1	ICMP	98 Echo (ping) reply id=0x2f03, seq=7/1792, ttl=108 (request in 25)

Experiência 6

Código da Aplicação

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

    struct connection c;
    initConnection(&c);
    if(parseInput(&c, argv[1]) != 0){
        printf("Couldn't parse input\n");
        return -1;
    }

    printf("User: %s\n", c.user);
    printf("Password: %s\n", c.password);
    printf("Host name: %s\n", c.host);
    printf("File path: %s\n", c.urlPath);

    char *ip = (char *)malloc(15);
    char *r = (char *)malloc(1000);
    if(getIP(ip, c.host) != 0){
        printf("Coudn't get IP\n");
        return -1;
    }

    if((c.ctrlfd = createConnection(ip, SERVER_PORT)) < 0){
        printf("Coudn't connect CTRLFD\n");
        return -1;
    }
```

Figura 1: main.c

```c
    if(checkResponse(c.ctrlfd, "220", r) != 0){
        printf("Incorrect response\n");
        return -1;
    }

    if(login(c.ctrlfd, c.user, c.password, c.annonymous) != 0 ){
        printf("Couldn't login\n");
        return -1;
    }

    if(enterPassiveMode(c.ctrlfd, &c) != 0){
        printf("Couldn't enter passive mode\n");
        return -1;
    }

    if(sendCommand(c.ctrlfd, "RETR", c.urlPath, 1)){
        printf("Couldn't send cmd retr\n");
        return -1;
    }

    if (checkResponse(c.ctrlfd, "150", r)){
        printf("Wrong response\n");
        return -1;
    }

    if(transfer(c.datafd, c.filename) != 0){
        printf("Couldn't transfer file\n");
        return -1;
    }
```

Figura 2: main.c (continuação)

```c
    if(checkResponse(c.ctrlfd, "226", r)){
        printf("Incorrect response\n");
        return -1;
    }

    if (close(c.ctrlfd) < 0){
        printf("Error closing controll socket!\n");
        return -1;
    }

    if (close(c.datafd) < 0){
        printf("Error closing data socket!\n");
        return -1;
    }

    free(ip);
    free(r);

    return 0;
}
```

Figura 3: main.c (continuação)

```c
#define SERVER_PORT 21
```

Figura 4: macro utilizada, ficheiro download.h

```c
struct connection{
    char* user;
    char* password;
    char* host;
    char* urlPath;
    char* filename;
    int annonymous;
    int ctrlfd;
    int datafd;
};
```

Figura 5: estrutura de dados utilizada para guardar informação da conexão, ficheiro download.h

```c
void initConnection (struct connection *connection);
int parseInput(struct connection *connection, char* input);
int getIP(char* ip, char* host);
int createConnection(char* addr, int port);
int checkResponse(int sockfd, char * expectedResponse, char* response);
int login(int sockfd, char * username, char * password, int annonymous);
int sendCommand(int sockfd, char * command, char* args, int hasArg);
int enterPassiveMode(int sockfd, struct connection *connection);
int transfer(int sockfd, char * name);
```

Figura 6: funções utilizadas, ficheiro download.h

```c
void initConnection(struct connection *connection){
    connection->user = NULL;
    connection->password = NULL;
    connection->host = NULL;
    connection->urlPath = NULL;
    connection->filename = NULL;
    connection->annonymous = -1;
    connection->ctrlfd = -1;
    connection->datafd = -1;
}
```

Figura 7: inicialização da estrutura da conexão, ficheiro download.c

```c
int parseInput(struct connection *connection, char *input){
    char *token;
    char *rest;

    // ftp://[<user>:<password>@]<host>/<url-path>
    printf("Parsing input...\n\n");

    // check ftp
    token = strtok_r(input, ":", &rest); // token should be ftp

    if(token == NULL){
        printf("Wrong input.\nUsage: ftp://[<user>:<password>@]<host>/<url-path>\n\n");
        return -1;
    }
    else if(strcmp(token, "ftp") != 0){
        printf("Protocol name should be ftp\n\n");
        return -1;
    }

    // user and password
    char *aux = (char *)malloc((sizeof(char) * strlen(rest)));
    strcpy(aux, rest);
    token = strtok(aux, "@");

    if(token == NULL || token[0] != '/' || token[1] != '/'){
        printf("Wrong input (protocol name).\nUsage: ftp://[<user>:<password>@]<host>/<url-path>\n\n");
        return -1;
    }
    if(strcmp(token, rest) == 0){
        // the user is annonymous
        connection->annonymous = 1;
        connection->user = "anonymous";
    }
    else{
        //user
```

Figura 8: função **parseInput**, download.c

```c
else{
    //user
    connection->annonymous = 0;
    token = strtok_r(rest, ":", &rest); //gets the user
    if(token == NULL){
        printf("Wrong input (user).\nUsage: ftp://[<user>:<password>@]<host>/<url-path>\n\n");
        return -1;
    }
    else if(strlen(token) == 0){
        printf("No username.\nUsage: ftp://[<user>:<password>@]<host>/<url-path>\n\n");
        return -1;
    }
    connection->user = (char *)malloc((sizeof(char) * strlen(token)) - 2);
    strcpy(connection->user, &token[2]);

    // password
    token = strtok_r(rest, "@", &rest); // gets the user
    if(token == NULL){
        printf("Wrong input (password).\nUsage: ftp://[<user>:<password>@]<host>/<url-path>\n\n");
        return -1;
    }
    else if(strlen(token) == 0){
        printf("No password.\nUsage: ftp://[<user>:<password>@]<host>/<url-path>\n\n");
        return -1;
    }

    connection->password = (char *)malloc((sizeof(char) * strlen(token)));
    strcpy(connection->password, token);
}

//host
token = strtok_r(rest, "/", &rest);
if(token == NULL){
    printf("Wrong input (host).\nUsage: ftp://[<user>:<password>@]<host>/<url-path>\n\n");
    return -1;
}
```

Figura 9: função **parseInput** (continuação), download.c

```
    else if(strlen(token) == 0){
        printf("No host.\nUsage: ftp://[<user>:<password>@]<host>/<url-path>\n\n");
        return -1;
    }

    connection->host = (char *)malloc((sizeof(char) * strlen(token)));
    strcpy(connection->host, token);

    // url path
    token = strtok_r(rest, "\0", &rest);
    if(token == NULL){
        printf("Wrong input (url).\nUsage: ftp://[<user>:<password>@]<host>/<url-path>\n\n");
        return -1;
    }
    else if(strlen(token) == 0){
        printf("No url.\nUsage: ftp://[<user>:<password>@]<host>/<url-path>\n\n");
        return -1;
    }

    connection->urlPath = (char *)malloc((sizeof(char) * strlen(token)));
    strcat(connection->urlPath, "/");
    strcat(connection->urlPath, token);

    char* name = strrchr(connection->urlPath, '/');
    printf("%s\n", name);

    connection->filename = (char *)malloc((sizeof(char) * strlen(name)));

    if(name != NULL){
        strcpy(connection->filename, name + 1);
    }

    free(aux);

    return 0;
}
```

Figura 10: função **parseInput** (continuação), download.c

```
int getIP(char *ip, char *host){
    struct hostent *h;
    if((h = gethostbyname(host)) == NULL)
    {
        herror("gethostbyname()");
        return -1;
    }
    strcpy(ip, inet_ntoa(*((struct in_addr *)h->h_addr)));
    printf("Host name  : %s\n", h->h_name);
    printf("IP Address : %s\n\n", inet_ntoa(*((struct in_addr *)h->h_addr)));
    return 0;
}
```

Figura 11: função **getIP**, download.c

```c
int createConnection(char *addr, 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(addr);
    server_addr.sin_port = htons(port);

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

    return sockfd;
}
```

Figura 12: função **createConnection**, download.c

```c
int checkResponse(int sockfd, char *expectedResponse, char *response){
    // Read response
    printf("Receiving response...\n");
    FILE *s = fdopen(sockfd, "r");
    char *string = (char *)malloc(1000);
    do{
        memset(string, 0, 1000);
        string = fgets(string, 1000, s);
        printf("%s", string);
    } while(!('1' <= string[0] && string[0] <= '5') || string[3] != ' ');

    // Check response
    if(string[0] != expectedResponse[0] || string[1] != expectedResponse[1] || string[2] != expectedResponse[2])
    {
        printf("Error in the Response\n");
        printf("Expected:%s\nReceived:%s\n", expectedResponse, string);
        return -1;
    }

    strcpy(response, string);

    free(string);
    printf("\n");
    return 0;
}
```

Figura 13: função **checkResponse**, download.c

```c
int login(int sockfd, char *username, char *password, int annonymous){
    printf("Login...\n");
    char *r = (char *)malloc(1000);

    int hasArg = annonymous ^ 1;
    if(sendCommand(sockfd, "USER", username, 1) != 0){
        printf("Error sending username\n");
        return -1;
    }

    if(checkResponse(sockfd, "331", r) != 0){
        return -1;
    }

    if(sendCommand(sockfd, "PASS", password, hasArg) != 0){
        printf("Error sending password\n");
        return -1;
    }

    if(checkResponse(sockfd, "230", r) != 0){
        return -1;
    }

    free(r);

    return 0;
}
```

Figura 14: função **login**, download.c

```c
int sendCommand(int sockfd, char *command, char *args, int hasArg){
    char *cmd = (char *)malloc(1000);

    strcpy(cmd, command);

    if(hasArg == 1){
        strcat(cmd, " ");
        strcat(cmd, args);
    }

    strcat(cmd, "\r\n");

    printf("Sending command...\n");
    printf("%s\n", cmd);
    if(write(sockfd, cmd, strlen(cmd)) != strlen(cmd)){
        return -1;
    }

    free(cmd);
    return 0;
}
```

Figura 15: função **sendCommand**, download.c

```c
int enterPassiveMode(int sockfd, struct connection *connection){
    char *r = (char *)malloc(1000);
    if (sendCommand(sockfd, "PASV", NULL, 0) != 0)
    {
        printf("Error entering passive mode\n");
        return -1;
    }

    if (checkResponse(sockfd, "227", r))
    {
        return -1;
    }

    int ip_arr[4], port_arr[2];
    sscanf(r, "227 Entering Passive Mode (%d,%d,%d,%d,%d,%d)",
           &ip_arr[0], &ip_arr[1], &ip_arr[2], &ip_arr[3], &port_arr[0], &port_arr[1]);

    char *ip = (char *)malloc(15);
    int port;

    sprintf(ip, "%d.%d.%d.%d", ip_arr[0], ip_arr[1], ip_arr[2], ip_arr[3]);
    port = 256 * port_arr[0] + port_arr[1];

    if ((connection->datafd = createConnection(ip, port)) < 0)
    {
        printf("Couldn't connect to data fd\n");
        return -1;
    }

    free(r);

    return 0;
}
```

Figura 16: função **enterPassiveMode**, download.c

```c
int transfer(int sockfd, char * name){
    FILE* filefd;

    filefd = fopen(name, "w");
    if(filefd == NULL){
        printf("Error opening file\n");
        return -1;
    }

    char* buf = (char *)malloc(2000);
    int bytes;

    printf("Downloading %s...\n", name);
    while((bytes = read(sockfd, buf, sizeof(buf)))){
        if(bytes < 0){
            printf("Error reading from data socket\n");
            return -1;
        }
        if((bytes = fwrite(buf, 1, bytes, filefd)) < 0){
            printf("Error writing data to file\n");
            return -1;
        }
    }

    if(fclose(filefd) < 0){
        printf("Error closing file\n");
        return -1;
    }

    return 0;
}
```

Figura 17: função **transfer**, download.c