Segundo Trabalho Laboratorial

Redes de Computadores

Mestrado Integrado em Engenharia Informática

Redes de computadores

Grupo:
Carolina Centeio Jorge - up201403090
João Fidalgo - up201303098
Mónica Fernandes - up201404789
Tiago Almeida - up201305665

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

23 de Dezembro de 2016

1 Summary

Este projeto foi desenvolvido no âmbito da disciplina de Redes de Computadores (RCOM) do 3º ano do Mestrado Integrado em Engenharia Informática e Computadores da Faculdade de Engenharia da Universidade do Porto. Todos os conceitos usados neste trabalho foram lecionados tanto nas aulas teóricas como nas aulas teórico-práticas, dando especial atenção aos slides de *Network*. O guia de laboratório foi também extremamente importante, pois continha todos os passos e comandos necessários para a realização deste projeto.

2 Introdução

Este projeto consiste em desenvolver uma aplicação através do protocolo FTP (descrito no RFC959) e realizar um conjunto de experiências com o objetivo de consolidar os conceitos lecionados nas aulas teóricas e perceber como configurar uma rede de computadores. Este projeto foi realizado em ambiente *Linux* e desenvolvido em C.

A aplicação reduz-se a fazer download de um ficheiro, passado como argumento pelo utilizador, de um servidor FTP assim como também é possível especificar o *username* e a *password* se for necessário fazer *login*, com a sintaxe descrita no RFC1738.

O propósito das experiências laboratoriais é dar a conhecer os comandos necessários para conseguir configurar uma redes de computadores. Esta segunda parte tem como objetivo configurar uma redes de computadores, todos ligados entre si através de duas VLANs diferentes configuradas num *switch* assim como também configurar um *router* comercial com *NAT* e utilizar *DNS* para a conversão de *hostnames* em *IP Addresses*.

O relatório tem uma secção dedicada a cada uma das duas partes deste projeto. O desenvolvimento da aplicação será explicado com grande detalhe na secção 3 e na secção 4 teremos subsecções para cada experiência realizada.

3 Aplicação de Download

Foi elaborado uma aplicação em C para fazer download de um ficheiro de um servidor FTP, usando ligações TCP. Esta aplicação implementa o protocolo FTP, como descrito em RFC959.

Esta aplicação é de um só argumento, sendo este um URL que segue a sintaxe descrita em RFC1738:

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

A aplicação começa por fazer parse desse URL, de forma a extrair a informação necessária para estabelecer a ligação TCP (user, password e host) e para, posteriormente, fazer download do ficheiro url-path. O user e password podem ser omitidos, sendo assumidos como "anonymous" e "mail@domain", respetivamente. Seguidamente, usando a função *getip*, vamos buscar o ip correspondente ao endereço host escolhido pelo utilizador. Assim, já podemos abrir um socket TCP (chamemos-lhe A), usando a porta 21 e conectar ao servidor, através da função *ftpConnect*.

Se a ligação for bem sucedida, é enviado o user seguido da password em *ftpLogin*. Se a resposta aos comandos de login for, também, positiva, tenta-se estabelecer o Passive Mode com a função *ftpPasv*. Esta função, se não falhar, retorna a porta que vai servir para abrir o novo socket TCP (chamemos-lhe B). Neste novo socket, B, vai ser recebido o ficheiro que queremos fazer download, assim que, no primeiro socket aberto, A, enviarmos o comando "retr", através da função *ftpDownload*. A *ftpDownload*, lê também o que vai sendo recebido no socket B e guarda num ficheiro, de forma a reproduzir inteiramente o ficheiro que queremos transferir. No final, a ligação é terminada com o comando "quit" e são também fechados os sockets A e B e liberto o espaço em memória.

4 Experiências Laboratoriais

4.1 Configuração de um IP de Rede

Esta experiência permitiu-nos saber como distinguir os diferentes pacotes de dados assim como a sua funcionalidade e também como estabelecer ligados entre dois computadores.

Um dos pacotes de dados usados nesta experiência são os pacotes ARP (*Address Resolution Protocol*). Este pacotes servem para obter o *MAC Address* (endereço que identifica um computador na rede) de um determinado *IP Address* (endereço que distingue um computador na rede). O computador onde é executado o comando *ping* envia um comando para todos os computadores com que tem ligação e pergunta que computador tem o um determinado *IP Address*. Este primeiro pacote envia o *IP Address* do computador que se pretende descobrir o *MAC Address* assim como o *IP Address* do computador para o qual se deve responder. Um segundo pacote ARP é enviado como resposta com o *IP Address* e o *MAC Address* do computador em questão. Este mecanismo pode ser visto no anexo 7.1 nas linhas 22 e 23.

Outro tipo de pacotes são os ICMP (*Internet Control Message Protocol*). Estes pacotes são gerados pelo comando ping e servem para enviar mensagens de erro ou de controlo para outros *hosts* ou *routers*. Quando um computador executa um comando ping, este envia um *echo request* para o outro computador, com o seu *IP Address* e o seu *MAC Address* assim como o *IP Address* e o *MAC Address* do computador de destino. Por sua vez, o computador de destino envia uma resposta (*echo reply*), com as mesmas informações. Este mecanismo pode ser confirmado no anexo 7.1 nas linhas 24 e 25.

Para descodificar estes pacotes é necessário analisar o *IP Diagram Format* descrito nas aulas teóricas. O campo *type of service* distingue se o pacote é ARP (0x0806) ou ICMP (0x0800). Isto pode ser visto no anexo 7.1 na segunda e terceira separação horizontal. O campo *length* representa o tamanho total do *datagram*. Este campo vem imediatamente a seguir ao campo *type of service* e como podemos na zona mencionada acima este campo tem 0x0054 como valor, o que representa 84 bytes.

O mecanismo de *loopback* é um mecanismo que reenvia uma mensagem de volta para o computador que a envia. Isto permite detetar erros na transmissão de dados assim como problemas nos cabos de transmissão.

4.2 Implementação de duas *Virtual LANS* num switch

Nesta experiência foi solicitada a criação de duas VLANS no switch. Para configurar uma VLAN temos de executar o comando "vlan x", em que x corresponde ao número da vlan, na consola do switch. Depois de configurar as VLANS, foi necessário adicionar as respetivas portas no switch. Para adicionar a porta, basta selecioná-la e usar o comando "interface fastethernet 0/y", em que y corresponde ao número da porta que queremos adicionar. Antes de adicionar a porta à VLAN com o comando "switchport access vlan x", devemos alterar o seu modo com o comando "switchport mode access".

Uma VLAN permite criar uma rede que é inacessível do interior para o exterior e vice versa. Assim, depois da observação dos logs da experiência 2, podemos concluir que existem 2 broadcast domains, um em cada VLAN. Podemos verificar, a partir do anexo 7.2, que quando o tux1 fez um pedido broadcast apenas o tux dessa mesma VLAN (Tux 4) recebeu o request. Embora os tuxs estejam todos no mesmo switch, os pacotes só eram recebidos por tuxs dentro da mesma VLAN.

4.3 Configurar um *router* em Linux

Esta experiência consiste em "ligar"as duas VLANs criada na experiência anterior, fazendo com o Tux4 de comporte como um *router*.

Nesta experiência começamos por configurar a eth0 e eth1 do Tux4 para que cada uma pertencesse a uma VLAN diferente. Para isso basta configurar os IPs de cada interface de forma a corresponder com a VLAN repetiva, utilizando o comando *ifconfig ethX [IP Address]*, sendo X 0 ou 1. De seguida adicionámos rotas ao Tux1 e ao Tux2 de forma a que seja possível enviar pacotes entre um e outro. Para isso utilizamos o comando *route add -net [IP da subrede] gw [IP Address do gateway]*, em que neste caso o gateway é o Tux4. Sendo, um exemplo deste comando seria *route add -net 172.16.30.0/24 gw 172.16.30.254*. Nesta

fase da experiência, o Tux1 e o Tux2 devem ter duas rotas, uma criada automaticamente quando é atribuído um IP à interface *ethernet* e outra que os liga mutuamente através do Tux4. O Tux4 deve ter apenas as duas rotas que são criadas automaticamente para cada um das suas interfaces *ethernet*.

A *forwarding table* guarda informações sobre a rede de destino, o *gateway* que faz a ligação a essa rede, a máscara dessa rede e a interface associada.

Para esta experiência, o eth0 do Tux4 foi ligada à mesma VLAN que o Tux1 e a eth1 do Tux4 foi ligada à mesma VLAN que o Tux2. Como podemos ver no Anexo 7.3, o Tux1 tanto consegue comunicar com o Tux4, que faz parte da sua VLAN, como também conseguir comunicar com o Tux2 através do Tux4. Conseguirmos observar troca de pacotes ARP e como já referido anteriormente, este servem para corresponder *IP Addresses* a *MAC Addresses*. Os pacotes ICMP observados são gerados pelo comando *ping* e estes "carregam"sempre o *IP Address* e o *MAC Address* de origem assim como o *IP Address* e o *MAC Address* de destino.

4.4 Configurar um router comercial e implementar NAT

O objetivo desta experiência era configurar um *router* comercial com NAT (*Network Address Resolution*) e perceber a sua funcionalidade.

A primeira parte desta experiência consiste em configurar o *router* sem NAT. Para isso, configuramos cada uma das interfaces *gigabitethternet* com os seguintes comandos:

- *ip address [ip address] [mascara de rede]*
- *no shutdown*

e configurmos as rotas com o comando *ip route [rede de destino] [mascara de rede] [gateway IP]*. Com isto, foi possível configurar cada uma das interfaces de forma a que cada computador na rede privada tivesse acesso ao *router* comercial. Neste momento da experiência percebemos que os pacotes do comando *ping* entre o computador 2 e o computador 1 percorriam o seguinte caminho tux2 ->tux4 ->tux1. Isto acontecia, porque o tux2 tinha uma route para a outra VLAN apartir do tux4. Removendo essa rota, o caminho passa a ser tux2->*router* ->tux4 ->tux1. Sem *redirects*, este caminho é percorrido todas as vezes que é enviado um pacote, no entanto, se ativarmos os *redirects* este caminho é apenas percorrido uma vez e os seguintes já percorrem a rota tux2 ->tux4 ->tux1.

A segunda parte consiste em adicionar NAT ao router para que esta rede privada possa ter acesso a uma rede na *internet*. A NAT é uma funcionalidade que permite que o endereço de um computador numa rede privada seja traduzido num *IP Address* válido na *internet*. A NAT recebe o *IP Address* interno e a porta local do computador e gera um endereço de 16 bits usando uma tabela hash, sendo este endereço então escrito no campo da porta de origem. Com este mecanismo, o *router* é capaz de enviar pacotes de dados para o exterior da rede privada a partir de um *IP Address* global e do *IP Address* gerado com o mecanismo descrito anteriormente e quando recebe uma resposta, consegue descobrir qual será o computador que terá de receber a resposta.

Para configure a NAT no *router* basta adicionar *ip nar inside* à interface que está ligada à rede privada e *ip nat outside* à interface ligado à rede externa.

4.5 DNS

Com esta experiência percebemos como configurar o servidor *DNS* assim como o seu papel e a sua importância nas redes de computadores.

De uma forma breve, o servidor *DNS* serve para traduzir um *hostname* (www.google.com) num *IP Address* (194.210.238.155) atrás do *DNS Resolver*.

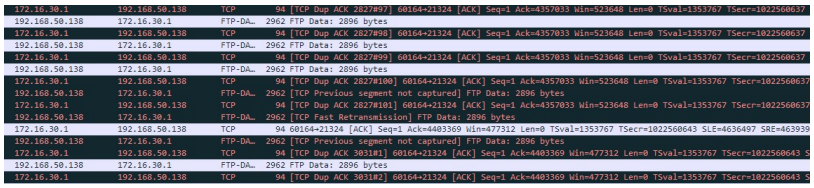
A configuração do servidor *DNS* trata-se apenas da edição do ficheiro /etc/resolv.conf, adicionando uma linha a especificar o *search* (netlab.fe.up.pt) e o *nameserver* (172.16.1.1).

O primeiro pacote é enviado ao servidor *DNS* com o nome do *host* que por sua vez é passado como argumento ao *resolver* e este devolve o *IP Address* ligado a esse *hostname*. O segundo pacote envia um *IP Address* e espera pelo respetivo *hostname*. Este último mecanismo é chamado *reverse DNS lookup*.

4.6 Ligações TCP

Esta experiência pretende testar e analisar resultados relativos à aplicação FTP que elaborámos. A aplicação, como analisado anteriormente, estabelece duas ligações TCP. A primeira, através do socket a que chamámos A, onde é transportada a informação de controlo FTP através do comando "pasv".

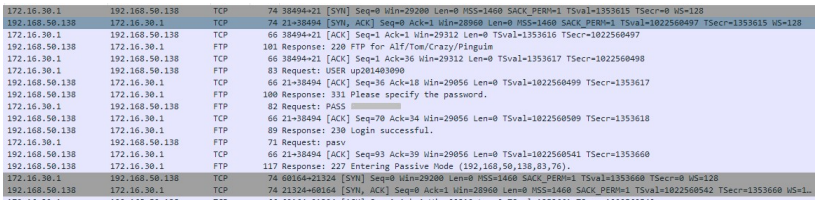
A ligação TCP divide-se em três fases e em todas é possível analisar um conjunto de parâmetros como a source port e a destination port (as mesmas durante toda a aplicação), o ARQ (por flag, neste caso ACK ou SYN), o sequence number, o Acknowledgment number e o window size. O mecanismo de ARQ em TCP funciona como Selective Repeat ARQ: o emissor envia o número de frames equivalente ao window size sem sem esperar por um ACK. Este window size é determinado pelo recetor, enquanto que a congestion window é determinada pelo emissor e não é visível nos logs. O recetor pode rejeitar frames individualmente, que posteriormente serão reenviados.



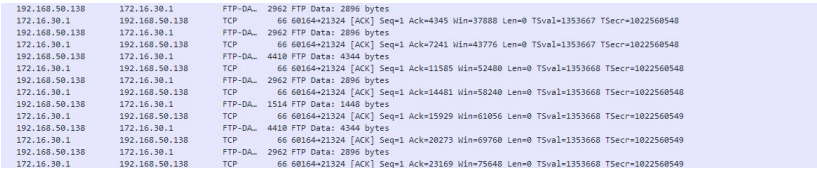
Aqui, o parâmetro ACK terá sido menor que o último byte enviado pelo emissor. O recetor detetou a falha na sequência de números e gerou um ACK duplicado para pacote subsequente recebido nessa ligação, até que o pacote em falta tenha sido recebido com sucesso.

As três fases são, então:

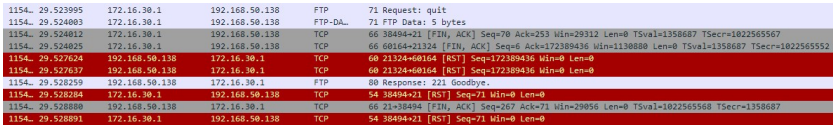
- Estabelecimento de ligação: desde a abertura do socket, ao processamento dos dados de user e password



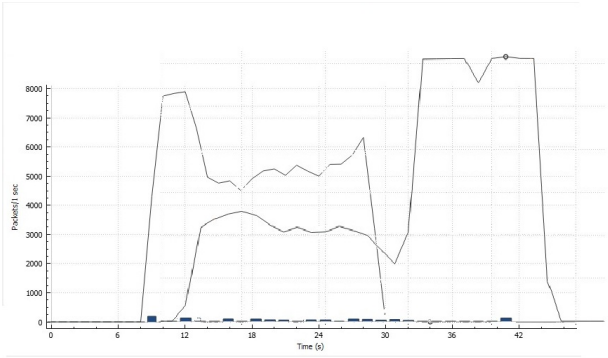
- Transferência de dados: desde o envio do comando "retr" até ao final da transferência do ficheiro em causa.



- Término de ligação: o envio do quit seguido da receção do "Goodbye" termina a ligação



É notável que o número de pacotes transferidos com sucesso, por segundo, diminui quando outra ligação TCP é iniciada. Nesta segunda, notamos também um aumento significativo quando a primeira transferência acaba. Tentámos juntar num só gráfico o que acabámos de explicar:



5 Conclusão

Todas as experiências foram realizadas com sucesso e com elas fomos capazes de aplicar os conceitos dados nas aulas teóricas.

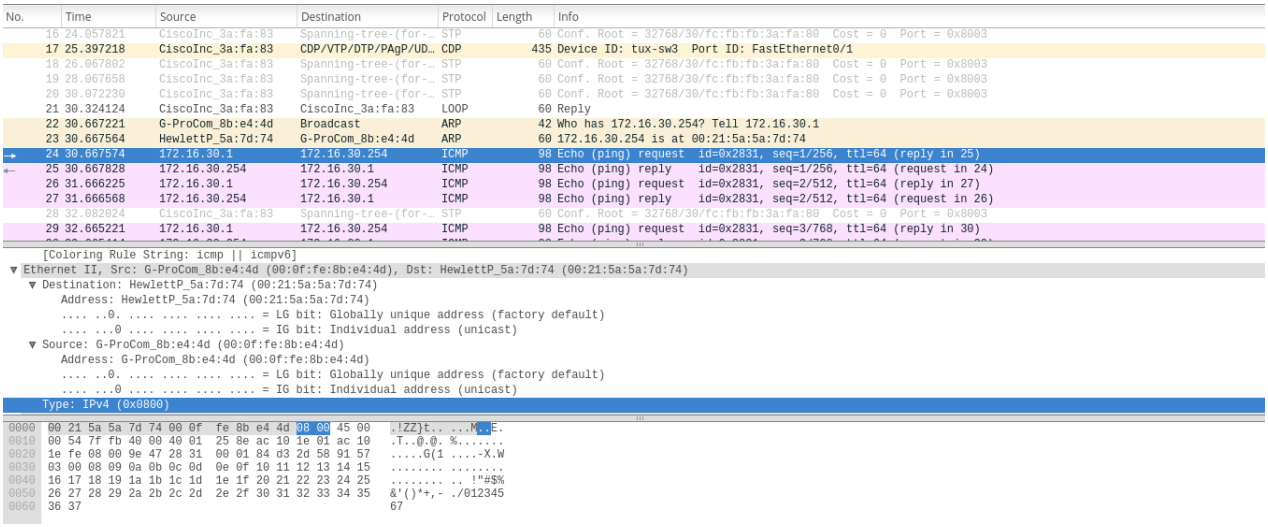
Conclui-se que a configuração de um rede de pequena dimensão é um problema sem grande complexidade, no entanto para a configuração de uma rede de mais dimensão é necessário aprofundar certos conceitos e aprender novos que não foram lecionados nesta unidade curricular.

6 Contribuições

Sumário, experiência 4 e 5 foram feitos pelo Tiago Almeida.
Introdução, experiência 1 e conclusão foram feitos pela Mónica Fernandes.
Aplicação FTP e experiência 6 foram feitos pela Carolina Centeio.
Experiência 2 e 3 foram feitas pelo João Fidalgo.

7 Anexos

7.1 Experiência 1



No.	Time	Source	Destination	Protocol	Length	Info
16	24.057821	CiscoInc_3a:fa:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
17	25.397218	CiscoInc_3a:fa:83	CDP/VTP/DTP/PAgP/UD…	CDP	435	Device ID: tux-sw3 Port ID: FastEthernet0/1
18	26.067802	CiscoInc_3a:fa:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
19	28.067658	CiscoInc_3a:fa:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
20	30.072230	CiscoInc_3a:fa:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
21	30.324124	CiscoInc_3a:fa:83	CiscoInc_3a:fa:83	LOOP	60	Reply
22	30.667221	G-ProCom_8b:e4:4d	Broadcast	ARP	42	Who has 172.16.30.254? Tell 172.16.30.1
23	30.667564	HewlettP_5a:7d:74	G-ProCom_8b:e4:4d	ARP	60	172.16.30.254 is at 00:21:5a:5a:7d:74
24	30.667574	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x2831, seq=1/256, ttl=64 (reply in 25)
25	30.667828	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x2831, seq=1/256, ttl=64 (request in 24)
26	31.666225	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x2831, seq=2/512, ttl=64 (reply in 27)
27	31.666568	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x2831, seq=2/512, ttl=64 (request in 26)
28	32.082024	CiscoInc_3a:fa:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
29	32.665221	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x2831, seq=3/768, ttl=64 (reply in 30)

```
[Coloring Rule String: icmp || icmpv6]
▼ Ethernet II, Src: G-ProCom_8b:e4:4d (00:0f:fe:8b:e4:4d), Dst: HewlettP_5a:7d:74 (00:21:5a:5a:7d:74)
  ▼ Destination: HewlettP_5a:7d:74 (00:21:5a:5a:7d:74)
      Address: HewlettP_5a:7d:74 (00:21:5a:5a:7d:74)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  ▼ Source: G-ProCom_8b:e4:4d (00:0f:fe:8b:e4:4d)
      Address: G-ProCom_8b:e4:4d (00:0f:fe:8b:e4:4d)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
```

```
0000  00 21 5a 5a 7d 74 00 0f  fe 8b e4 4d 08 00 45 00   .!ZZ}t.. ...M..E.
0010  00 54 7f fb 40 00 40 01  25 8e ac 10 1e 01 ac 10   .T..@.@. %.......
0020  1e fe 08 00 9e 47 28 31  00 01 84 d3 2d 58 91 57   .....G(1 ....-X.W
0030  03 00 08 09 0a 0b 0c 0d  0e 0f 10 11 12 13 14 15   ........ ........
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ........ .. !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                             67
```

7.2 Experiência 2

7.2.1 Tux 1

No.	Time	Source	Destination	Protocol	Length	Info
25	37.472634	172.16.30.1	172.16.30.255	ICMP	98	Echo (ping) request id=0x2fed, seq=1/256, ttl=64 (no response found!)
26	37.472664	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x2fed, seq=1/256, ttl=64
27	38.088754	CiscoInc_3a:fa:86	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8006
28	38.471642	172.16.30.1	172.16.30.255	ICMP	98	Echo (ping) request id=0x2fed, seq=2/512, ttl=64 (no response found!)
29	38.471672	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x2fed, seq=2/512, ttl=64
30	39.470679	172.16.30.1	172.16.30.255	ICMP	98	Echo (ping) request id=0x2fed, seq=3/768, ttl=64 (no response found!)
31	39.470709	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x2fed, seq=3/768, ttl=64

7.2.2 Tux 2

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	CiscoInc_3a:fa:84	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
2	1.999833	CiscoInc_3a:fa:84	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
3	4.004829	CiscoInc_3a:fa:84	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
4	6.014999	CiscoInc_3a:fa:84	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
5	6.102317	CiscoInc_3a:fa:84	CiscoInc_3a:fa:84	LOOP	60	Reply
6	8.014602	CiscoInc_3a:fa:84	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
7	10.019545	CiscoInc_3a:fa:84	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
8	12.029549	CiscoInc_3a:fa:84	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
9	14.029429	CiscoInc_3a:fa:84	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
10	16.034316	CiscoInc_3a:fa:84	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004

7.2.3 Tux 4

No.	Time	Source	Destination	Protocol	Length	Info
163	98.240670	CiscoInc_3a:fa:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
164	98.505512	172.16.30.1	172.16.30.255	ICMP	98	Echo (ping) request id=0x2fed, seq=50/12800, ttl=64 (no response found!)
165	98.505884	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x2fed, seq=50/12800, ttl=64
166	99.505515	172.16.30.1	172.16.30.255	ICMP	98	Echo (ping) request id=0x2fed, seq=51/13056, ttl=64 (no response found!)
167	99.505778	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x2fed, seq=51/13056, ttl=64

7.3 Experiência 3

7.3.1 Tux 1 ->2

No.	Time	Source	Destination	Protocol	Length	Info
169	153.156635	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x19ed, seq=1/256, ttl=64 (reply in 170)
170	153.157265	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x19ed, seq=1/256, ttl=63 (request in 169)
171	154.155640	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x19ed, seq=2/512, ttl=64 (reply in 172)
172	154.155872	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x19ed, seq=2/512, ttl=63 (request in 171)
173	154.396298	CiscoInc_3a:fa:83	Spanning-tree-(for-	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003

7.3.2 Tux 1 ->4

No.	Time	Source	Destination	Protocol	Length	Info
22	32.848713	CiscoInc_3a:fa:83	CiscoInc_3a:fa:83	LOOP	60	Reply
23	33.637699	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x19a0, seq=1/256, ttl=64 (reply in 26)
24	33.637847	HewlettP_5a:7d:74	Broadcast	ARP	60	Who has 172.16.30.1? Tell 172.16.30.254
25	33.637874	G-ProCom_8b:e4:4d	HewlettP_5a:7d:74	ARP	42	172.16.30.1 is at 00:0f:fe:8b:e4:4d
26	33.638126	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x19a0, seq=1/256, ttl=64 (request in 23)
27	34.081618	CiscoInc_3a:fa:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
28	34.636702	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x19a0, seq=2/512, ttl=64 (reply in 29)
29	34.636913	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x19a0, seq=2/512, ttl=64 (request in 28)
30	35.635704	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x19a0, seq=3/768, ttl=64 (reply in 31)
31	35.636055	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x19a0, seq=3/768, ttl=64 (request in 30)

7.3.3 Tux 4 - Eth0

No.	Time	Source	Destination	Protocol	Length	Info
94	150.369502	CiscoInc_3a:fa:86	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8006
95	151.825961	G-ProCom_8b:e4:4d	Broadcast	ARP	60	Who has 172.16.30.254? Tell 172.16.30.1
96	151.825984	HewlettP_5a:7d:74	G-ProCom_8b:e4:4d	ARP	42	172.16.30.254 is at 00:21:5a:5a:7d:74
97	151.826223	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x1b73, seq=1/256, ttl=64 (reply in 98)
98	151.826516	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1b73, seq=1/256, ttl=63 (request in 97)
99	152.379506	CiscoInc_3a:fa:86	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8006
100	152.824985	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x1b73, seq=2/512, ttl=64 (reply in 101)
101	152.825143	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1b73, seq=2/512, ttl=63 (request in 100)
102	153.824935	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x1b73, seq=3/768, ttl=64 (reply in 103)
103	153.825086	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1b73, seq=3/768, ttl=63 (request in 102)
104	154.103039	CiscoInc_3a:fa:86	CiscoInc_3a:fa:86	LOOP	60	Reply

7.3.4 Tux 4 - Eth1

No.	Time	Source	Destination	Protocol	Length	Info
91	144.955493	CiscoInc_3a:fa:87	CDP/VTP/DTP/PAgP/UD…	CDP	435	Device ID: tux-sw3 Port ID: FastEthernet0/5
92	146.381511	CiscoInc_3a:fa:87	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8007
93	148.157281	Kye_25:26:0a	Broadcast	ARP	42	Who has 172.16.31.1? Tell 172.16.31.253
94	148.157398	HewlettP_61:30:63	Kye_25:26:0a	ARP	60	172.16.31.1 is at 00:21:5a:61:30:63
95	148.157409	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x1b73, seq=1/256, ttl=63 (reply in 96)
96	148.157542	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1b73, seq=1/256, ttl=64 (request in 95)
97	148.396453	CiscoInc_3a:fa:87	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8007
98	149.156038	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x1b73, seq=2/512, ttl=63 (reply in 99)
99	149.156159	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1b73, seq=2/512, ttl=64 (request in 98)
100	150.155987	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x1b73, seq=3/768, ttl=63 (reply in 101)
101	150.156101	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1b73, seq=3/768, ttl=64 (request in 100)
102	150.391167	CiscoInc_3a:fa:87	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8007

7.4 Experiência 4

7.4.1 Passo 4 - Sem Redirect

No.	Time	Source	Destination	Protocol	Length	Info
5	6.014810	CiscoInc_3a:fa:84	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
6	7.483499	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) request id=0x1894, seq=1/256, ttl=64 (reply in 8)
7	7.483881	172.16.31.254	172.16.31.1	ICMP	70	Redirect (Redirect for host)
8	7.484159	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x1894, seq=1/256, ttl=63 (request in 6)
9	8.019703	CiscoInc_3a:fa:84	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
10	8.482504	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) request id=0x1894, seq=2/512, ttl=64 (reply in 12)
11	8.482815	172.16.31.254	172.16.31.1	ICMP	70	Redirect (Redirect for host)
12	8.483011	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x1894, seq=2/512, ttl=63 (request in 10)
13	9.481501	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) request id=0x1894, seq=3/768, ttl=64 (reply in 15)
14	9.481813	172.16.31.254	172.16.31.1	ICMP	70	Redirect (Redirect for host)
15	9.482088	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x1894, seq=3/768, ttl=63 (request in 13)
16	10.024638	CiscoInc_3a:fa:84	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
17	10.481409	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) request id=0x1894, seq=4/1024, ttl=64 (reply in 19)
18	10.481714	172.16.31.254	172.16.31.1	ICMP	70	Redirect (Redirect for host)
19	10.482103	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x1894, seq=4/1024, ttl=63 (request in 17)
20	11.481390	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) request id=0x1894, seq=5/1280, ttl=64 (reply in 22)
21	11.481692	172.16.31.254	172.16.31.1	ICMP	70	Redirect (Redirect for host)
22	11.482006	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x1894, seq=5/1280, ttl=63 (request in 20)
23	12.029565	CiscoInc_3a:fa:84	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
24	12.481392	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) request id=0x1894, seq=6/1536, ttl=64 (reply in 26)
25	12.481690	172.16.31.254	172.16.31.1	ICMP	70	Redirect (Redirect for host)

7.4.2 Passo 4 - Com Redirect

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	CiscoInc_3a:fa:84	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
2	0.461789	CiscoInc_3a:fa:84	CiscoInc_3a:fa:84	LOOP	60	Reply
3	1.258621	CiscoInc_3a:fa:84	CDP/VTP/DTP/PAgP/UD…	CDP	435	Device ID: tux-sw3 Port ID: FastEthernet0/2
4	2.004965	CiscoInc_3a:fa:84	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
5	4.009905	CiscoInc_3a:fa:84	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
6	5.626150	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) request id=0x1a46, seq=1/256, ttl=64 (reply in 8)
7	5.626548	172.16.31.254	172.16.31.1	ICMP	70	Redirect (Redirect for host)
8	5.626973	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x1a46, seq=1/256, ttl=63 (request in 6)
9	6.014825	CiscoInc_3a:fa:84	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
10	6.625159	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) request id=0x1a46, seq=2/512, ttl=64 (reply in 11)
11	6.625613	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x1a46, seq=2/512, ttl=63 (request in 10)
12	7.624159	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) request id=0x1a46, seq=3/768, ttl=64 (reply in 13)
13	7.624569	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x1a46, seq=3/768, ttl=63 (request in 12)
14	8.019652	CiscoInc_3a:fa:84	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
15	8.624117	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) request id=0x1a46, seq=4/1024, ttl=64 (reply in 16)
16	8.624557	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x1a46, seq=4/1024, ttl=63 (request in 15)
17	9.624106	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) request id=0x1a46, seq=5/1280, ttl=64 (reply in 18)
18	9.624520	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x1a46, seq=5/1280, ttl=63 (request in 17)

7.4.3 Passo 5 - NAT

No.	Time	Source	Destination	Protocol	Length	Info
7	9.305644	172.16.30.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x0d47, seq=1/256, ttl=64 (reply in 8)
8	9.306768	172.16.1.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0d47, seq=1/256, ttl=62 (request in 7)
9	10.029111	CiscoInc_3a:fa:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
10	10.306881	172.16.30.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x0d47, seq=2/512, ttl=64 (reply in 11)
11	10.307699	172.16.1.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0d47, seq=2/512, ttl=62 (request in 10)
12	11.306431	172.16.30.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x0d47, seq=3/768, ttl=64 (reply in 13)
13	11.307232	172.16.1.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0d47, seq=3/768, ttl=62 (request in 12)
14	12.028879	CiscoInc_3a:fa:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
15	12.306440	172.16.30.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x0d47, seq=4/1024, ttl=64 (reply in 16)
16	12.307241	172.16.1.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0d47, seq=4/1024, ttl=62 (request in 15)
17	13.306430	172.16.30.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x0d47, seq=5/1280, ttl=64 (reply in 18)
18	13.307242	172.16.1.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0d47, seq=5/1280, ttl=62 (request in 17)
19	14.033685	CiscoInc_3a:fa:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
20	14.306454	172.16.30.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x0d47, seq=6/1536, ttl=64 (reply in 21)
21	14.307276	172.16.1.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0d47, seq=6/1536, ttl=62 (request in 20)
22	14.316878	HewlettP_5a:7d:74	G-ProCom_8b:e4:4d	ARP	60	Who has 172.16.30.1? Tell 172.16.30.254
23	14.316890	G-ProCom_8b:e4:4d	HewlettP_5a:7d:74	ARP	42	172.16.30.1 is at 00:0f:fe:8b:e4:4d
24	15.306435	172.16.30.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x0d47, seq=7/1792, ttl=64 (reply in 25)
25	15.307256	172.16.1.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0d47, seq=7/1792, ttl=62 (request in 24)

7.5 Experiência 5

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.20.1	172.16.1.1	DNS	69	Standard query 0x9fbc A google.pt
2	0.001431	172.16.1.1	172.16.20.1	DNS	471	Standard query response 0x9fbc A google.pt A 194.210.238.155 A 194.210.238.159 A 194.210.238.163 A 194.210.238.165 A 194.210.238.166 A 19…
3	0.001644	172.16.20.1	194.210.238.155	ICMP	98	Echo (ping) request id=0x5900, seq=1/256, ttl=64 (reply in 4)
4	0.008032	194.210.238.155	172.16.20.1	ICMP	98	Echo (ping) reply id=0x5900, seq=1/256, ttl=55 (request in 3)
5	0.008186	172.16.20.1	172.16.1.1	DNS	88	Standard query 0xe4bd PTR 155.238.210.194.in-addr.arpa
6	0.009974	172.16.1.1	172.16.20.1	DNS	147	Standard query response 0xe4bd No such name PTR 155.238.210.194.in-addr.arpa SOA ns01.fccn.pt
7	0.111450	CiscoInc_5c:4d:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/20/fc:fb:fb:5c:4d:80 Cost = 0 Port = 0x8003
8	1.003101	172.16.20.1	194.210.238.155	ICMP	98	Echo (ping) request id=0x5900, seq=2/512, ttl=64 (reply in 9)
9	1.008966	194.210.238.155	172.16.20.1	ICMP	98	Echo (ping) reply id=0x5900, seq=2/512, ttl=55 (request in 8)
10	2.005097	172.16.20.1	194.210.238.155	ICMP	98	Echo (ping) request id=0x5900, seq=3/768, ttl=64 (reply in 11)
11	2.010644	194.210.238.155	172.16.20.1	ICMP	98	Echo (ping) reply id=0x5900, seq=3/768, ttl=55 (request in 10)

8 Código fonte da aplicação FTP

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <fcntl.h>

#define MESSAGE_SIZE 200

int getip(char *host, char *hostaddr)
{
struct hostent *h = malloc(sizeof(struct hostent));

/*
struct hostent {
char    *h_name; Official name of the host.
    char    **h_aliases; A NULL-terminated array of alternate names for the host.
int     h_addrtype; The type of address being returned; usually AF_INET.
    int     h_length; The length of the address in bytes.
char    **h_addr_list; A zero-terminated array of network addresses for the host.
Host addresses are in Network Byte Order.
};
#define h_addr h_addr_list[0] The first address in h_addr_list.
*/
    if ((h=gethostbyname(host)) == NULL) {
        herror("gethostbyname");
        return -1;
    }

    printf("Host name  : %s\n", h->h_name);
    printf("IP Address : %s\n",inet_ntoa(*((struct in_addr *)h->h_addr)));
    strcpy(hostaddr, inet_ntoa(*((struct in_addr *)h->h_addr)));
printf("Free h\n");
    return 0;
}

int ftpLogin(int socketfd, char* user, char* pass){
   printf("Login\n");
char reply[MESSAGE_SIZE] = "", *usercmd = malloc(sizeof(char) * MESSAGE_SIZE), *passcmd = malloc(siz

read(socketfd, reply, MESSAGE_SIZE);

sprintf(usercmd, "USER %s\n", user);
   printf("%s\n", usercmd);
write(socketfd, usercmd, strlen(usercmd));
```

```
read(socketfd, reply, MESSAGE_SIZE);
printf("%s\n", reply);

bzero(reply, strlen(reply));

sprintf(passcmd, "PASS %s\n", pass);
   printf("%s\n", passcmd);
write(socketfd, passcmd, strlen(passcmd));
read(socketfd, reply, MESSAGE_SIZE);

printf("%s\n", reply);

if (reply[0] == '2' && reply[1] == '3' && reply[2]== '0'){
free(usercmd);
free(passcmd);
return 0; //SUCCESS
}

free(usercmd);
free(passcmd);
return -1;

}

int ftpPasv(int shost, char* host, unsigned int *port){
char reply[MESSAGE_SIZE] = "", *command =  "pasv\n";
   int h1, h2, h3, h4, h5, h6;

   bzero(reply, strlen(reply));

write(shost, command, strlen(command));
read(shost, reply, MESSAGE_SIZE);

   if(sscanf(reply, "227 Entering Passive Mode (%d,%d,%d,%d,%d,%d)\n", &h1, &h2, &h3, &h4, &h5, &h6)
     printf("Answer does not match for Passive Mode:\n --> %s \n", reply);
     return -1;
   }

sprintf(host, "%d.%d.%d.%d", h1, h2, h3, h4);
*port = h5*256 + h6;

return 0;
}

int ftpDownload(int socketfd, int shost, char* filename, char* path) {
char reply[MESSAGE_SIZE] = "", *retrcmd = malloc(sizeof(char) * MESSAGE_SIZE);

bzero(reply, strlen(reply));
sprintf(retrcmd, "retr %s\n", path);
printf("Path: %s\nFilename. %s\n", path, filename);

write(socketfd, retrcmd, strlen(retrcmd));
```

```
read(socketfd, reply, MESSAGE_SIZE);
printf("Reply: %s\n", reply);


  if (reply[0] == '1' && reply[1] == '5' && reply[2]== '0') {

    int fd =  open(filename, O_WRONLY | O_CREAT, 0777), nbytes = 0, counter = 0;
    char* trans = malloc(sizeof(char) * MESSAGE_SIZE);

    while((nbytes = read(shost, trans, MESSAGE_SIZE)) != 0) {
if(!(counter % 1000))
printf(".");
fflush(stdout);
counter++;
write(fd, trans, nbytes);
}

printf("\n");

    close(fd);

    bzero(reply, strlen(reply));
    read(socketfd, reply, MESSAGE_SIZE);

    if (reply[0] == '2'){

printf("Download succeeded\n");
free(trans);
free(retrcmd);
return 0; //SUCCESS
}

printf("Download failed\n");

  }

free(retrcmd);
return -1;

}

int parseURL(char * url, char * host, char * path, char * user, char * pass, char* filename){

int ret = sscanf(url, "ftp://%[^:]:%[^@]@%[^/]/%s\n", user, pass, host, path);
if(ret != 4){
ret = sscanf(url, "ftp://%[^/]/%s\n", host, path);
if (ret != 2) {
return -1;
       }
       strcpy(user, "anonymous");
       strcpy(pass, "mail@domain");
    } //not success
```

```
char * last = strrchr(path,'/');
strcpy(filename, last+1);

printf("Filename: %s\n", filename);
return 0; //success
}

int ftpConnect(char* server_ip, const unsigned int port){
//connect socket
struct sockaddr_in server_addr;
int sockfd;

/*server address handling*/
bzero((char*)&server_addr, sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = inet_addr(server_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()");
return -1;
}

/*connect to the server*/
  if(connect(sockfd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0){
         perror("connect()");
return -1;
}

return sockfd;
}

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

char host[MESSAGE_SIZE] = "";
char path[MESSAGE_SIZE] = "";
char user[MESSAGE_SIZE] = "";
char pass[MESSAGE_SIZE] = "";
char filename[MESSAGE_SIZE] = "";



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

if(parseURL(argv[1], host, path, user, pass, filename) == -1){
printf("url does not match ftp://[<user>:<password>@]<host>/<url-path> \n");
return -1;
```

```
}
   printf("Finished parse\n");


char* hostaddr = malloc(sizeof(char) * MESSAGE_SIZE);

if(getip(host, hostaddr) != 0){
printf("Failed getting ip\n");
free(hostaddr);
return -1;
}

unsigned int port = 21, sport;

printf("Starting connection...\n");
printf("%s\n", hostaddr);

int sockfd = ftpConnect(hostaddr, port);

printf("ftpconnect done\n");

if(sockfd == -1){
free(hostaddr);
printf("Connection failed\n");
return -1;
}

//login
if(ftpLogin(sockfd, user, pass) != 0){
free(hostaddr);
close(sockfd);
    printf("Login failed\n");
    return -1;
  }

printf("Login succeeded\n");

//passv - calcular a porta
char *shost = malloc(sizeof(char)*MESSAGE_SIZE);
if(ftpPasv(sockfd, shost, &sport) != 0){
printf("Passive Mode Failed\n");
free(hostaddr);
free(shost);
close(sockfd);
return -1;
  }


printf("Passive Mode Established\n");
int sockhost = ftpConnect(shost, sport);
if(sockhost == -1){
printf("Connection failed\n");
```

```
free(hostaddr);
free(shost);
close(sockfd);
return -1;
}
//download
if(ftpDownload(sockfd, sockhost, filename, path) != 0){
free(hostaddr);
free(shost);
close(sockfd);
close(sockhost);

    printf("Download failed\n");
    return -1;
  }
  printf("Download finished\n");

char* quit = "quit\n";
write(sockfd, quit, strlen(quit));
write(sockhost, quit, strlen(quit));

free(hostaddr);
free(shost);
close(sockfd);
close(sockhost);

return 0;

}
```