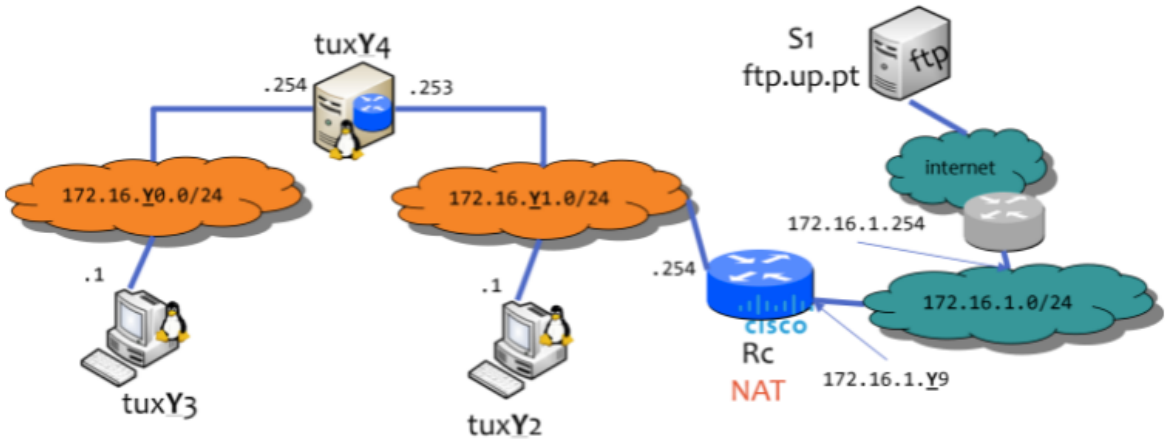
U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

2º Trabalho Laboratorial

Desenvolvimento de Aplicação de Download e Configuração e Estudo de uma Rede de Computadores



Trabalho realizado por:

Henrique Nunes - up21906852
Frederico Rodrigues - up201807626

Unidade Curricular: RC - Redes de Computadores
Data de Entrega: 22 de janeiro de 2022

Índice

Índice **2**

Sumário **2**

Introdução **3**

Parte 1 - Aplicação Download **3**

Arquitetura 3

Resultados 4

Parte 2 - Configuração e Análise de Rede **5**

Experiência 1 - Configuração de Endereços IP 5

Experiência 2 - LANs Virtuais 7

Experiência 3 - Configuração de Router 8

Experiência 4 - Configuração de Router (Lab) 9

Conclusão **11**

Anexos **13**

Comandos de configuração 13

tux13 13

tux14 13

tux12 13

Switch 13

Cisco 14

Logs Capturados 16

Código 21

download.h 21

download.c 22

Sumário

Este relatório descreve o trabalho desenvolvido no 2º trabalho prático desenvolvido na Unidade Curricular de Redes de Computadores. As experiências foram sobretudo realizadas nas aulas práticas onde nos foi fornecida uma bancada com o equipamento necessário à realização da mesma. Já o desenvolvimento da aplicação foi desnvolvido totalmente fora das mesmas.

Introdução

No âmbito da Unidade Curricular de Redes de Computadores, foi proposto o desenvolvimento de um trabalho prático dividido em duas partes. A primeira parte incide no desenvolvimento de uma aplicação de download por FTP, na linguagem C e tomando partido da utilização de sockets, e a segunda numa sequência de experiências com várias etapas da configuração de uma rede, com o objetivo final de conectar a rede local à Internet.

Parte 1 - Aplicação Download

A primeira parte deste trabalho consiste na criação de uma aplicação que permita transferir um ficheiro de acordo com o protocolo *FTP (File Transfer Protocol)*, descrito no RFC959, e com ligações *TCP (Transmisson Control Protocol)* baseadas em *sockets*. Assim, a aplicação espera receber um argumento, que permita conectar a um servidor *FTP,* do tipo *ftp://[<user>:<pass>@]<host>/<url-path>*, isto é, a indicação do protocolo, do *host* e do *url-path* é mandatória, já a indicação do *user* e da *pass* são facultativas.

Arquitetura

A aplicação de download começa a sua execução com o *parsing* do argumento de entrada, isto é, tenta fazer a separação do *url* que é enviado através da consola em *user*, *pass*, *host*, *path* e *protocol*. Para isso foi implementada a função ***parse_args*** que recebe como argumento o *url* a processar e preenche uma *struct args* com os argumentos anteriormente mencionados, e ainda a porta (*port*) ao qual o protocolo (*protocol*) corresponde, e o nome do ficheiro *filename* obtido a partir do caminho (*path*) indicado no *url.* Caso não seja fornecido um *user* e uma *pass* estes são configurados com '*anonymous'*. Este processo do input é realizado com uma máquina de estados que lê a *string* e vai alterando o seu estado. Caso termine a leitura sem atingir o estado final, que exige que todos os parâmetros mandatórios estejam preenchidos, termina com a indicação do erro.
De seguida, é utilizada a função ***hostname_to_IP*** que recebe o nome do *host* e faz a conversão para o endereço *IP* correspondente, configurando o respetivo campo na estrutura *args*.
Com o endereço *IP* e a porta (deverá ser a porta 21) já configurados, é chamada a função ***connect_socket***, que cria e conecta a um *socket*.
Para enviar comandos é utlizada a função ***ftp_send_cmd*** que por sua vez faz uma chamada à função ***send*** do módulo *sys/socket.h* enviar o comando que lhe foi fornecido e para receber respostas é utilizada a função ***ftp_recv_resp*** que por sua vez lê a mensagem recebida, linha a linha, com a chamda à função ***recv*** do módulo *sys/socket.h*. Tal como mencionado na documentação do protocolo *FTP*, a leitura da mensagem é terminada quando recebe uma secção da mensagem que contém o caracter *' '* imediatamente a seguir ao código de estado da resposta, que deverá corresponder com a primeira secção da mensagem enviada.
Depois de estabelecida a ligação com o *socket*, é necessário realizar o *login*. Para isso é enviado o comando *USER <args.user>* seguido do comando *PASS <args.pass>* caso o primeiro tenha sido bem sucedido, ou seja recebido o código esperado na resposta.

Feito o *login* com sucesso, é enviado o comando *PASV* e verificado se foi recebida a resposta esperada. A partir desta resposta é calculada a porta à qual se deve conectar a *data socket* (socket de dados). De seguida é criada esta *socket* e estabelecida a conexão na porta calculada com uma nova chamada à função ***connect_socket***.
Posteriormente, é enviado o comando *RETR <args.path>* e aguardada uma resposta da gama 100 seguida de uma mensagem de transferência concluída.
Antes de ser feita a leitura da mensagem que deverá indicar que a transferência terá sido concluída com sucesso é chamada a função ***download_file*** que lê da *data socket* o ficheiro transferido e copia-o para um ficheiro de nome *<args.filename>* no diretório corrente.
Por fim, desconecta tanto o socket de controlo como o socket de dados e termina a sua execução.

Resultados

A aplicação desenvolvida funcionou conforme o esparado, sendo capaz de fazer a transferência de variados ficheiros, em diferentes *hosts* e com diferentes permissões de acesso, isto é, com e sem autenticação.

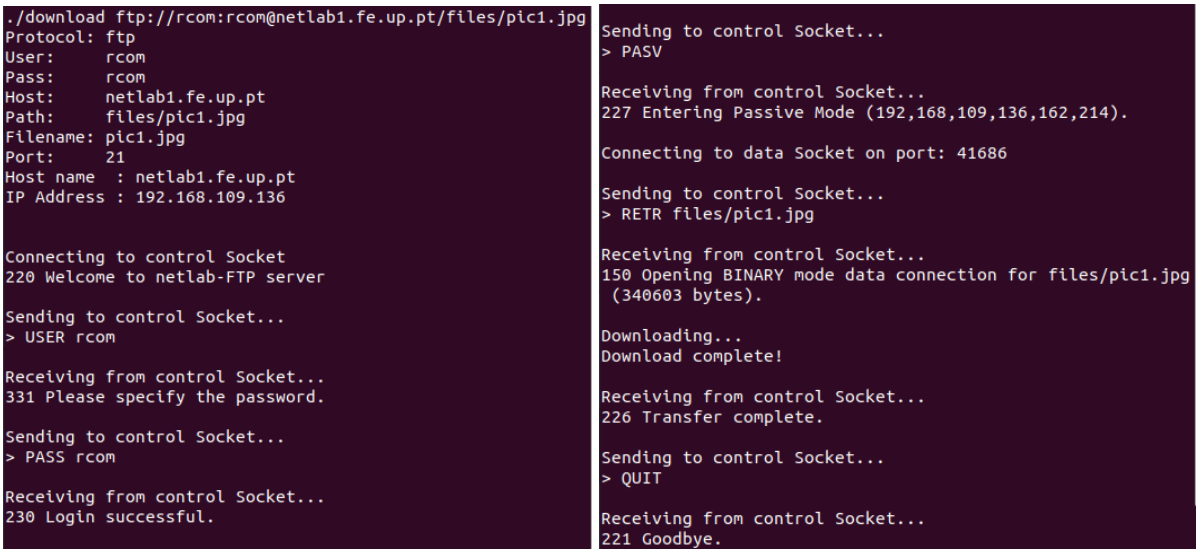


Figura 1 - Download do ficheiro pic1.jpg, com autenticação, a partir do netlab1.fe.up.pt

Parte 2 - Configuração e Análise de Rede

Os endereços IPs das configurações esperadas variam consoante o laboratório e a bancada utilizada para a realização das experiências, por isso, dado que foi utilizada a bancada 1 da sala I320, Y e W foram substituídos por 1 e 2, respetivamente, tanto na execução das experiências como na sua descrição.

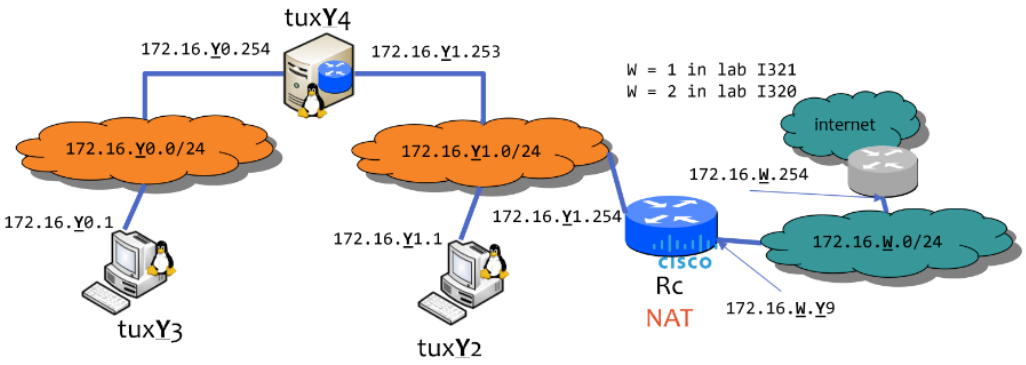


Figura 2 - Objetivo de configuração de rede

Experiência 1 - Configuração de Endereços IP

Nesta experiência configuramos endereços IP e conectamo-nos ao Switch, resultando na seguinte *network*:

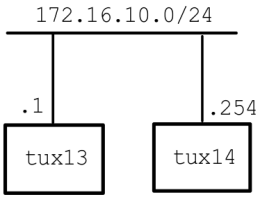


Figura 3 - Configuração da rede na experiência 1

Inicialmente, configuramos o *tux13* e *tux14* de modo a ter um endereço IP e a network *172.16.10.0/24* definida de acordo com a figura 3, tomando também nota do endereço MAC e IP de cada um.

Para tal introduzimos os seguintes comandos:
No tux13,

```
$ ifconfig eth0 up
$ ifconfig eth0 172.16.10.1/24
```

No tux14,

```
$ ifconfig eth0 up
$ ifconfig eth0 172.16.10.254/24
```

Para verificar a conetividade entre os computadores utilizamos o comando *ping*. Este comando gera e envia pacotes ICMP (Internet Control Message Protocol).

Quando fazemos *ping* do *tux13* para o *tux14*, o *tux13* envia um pacote ARP com o seu IP (172.16.10.1), o seu MAC (00:21:5a:5a:7d:16), o IP do destino (172.16.10.254) (*tux14*) e o MAC do destino que fica 00:00:00:00:00:00 uma vez que é desconhecido. O pacote de

resposta contém o IP do *tux14*, MAC do *tux14* (00:21:5a:5a:7b:3f), IP do *tux13* (destino) e MAC.

ICMP *Request Packet*:

IP da fonte	172.16.10.1
MAC da fonte	00:21:5a:5a:7d:16
IP de recetor	172.16.10.254
MAC de recetor	00:21:5a:5a:7b:3f

Tabela 1: Request Packet ICMP

ICMP *Reply Packet*:

IP da fonte	172.16.10.254
MAC da fonte	00:21:5a:5a:7b:3f
IP de recetor	172.16.10.1
MAC de recetor	00:21:5a:5a:7d:16

Tabela 2: Reply Packet ICMP

O ARP (Address Resolution Protocol) é um protocolo de comunicação utilizado para a resolução de um endereço do *link layer*, como um endereço MAC, associado a um endereço da *internet layer*, endereço IP. Assim, um pacote ARP contém a questão: Qual é o MAC que corresponde a um dado IP? O host configurado para usar esse endereço IP responde com um pacote ARP que contém o seu MAC.

Para distinguir os pacotes Ethernet entre ARP, IP e ICMP, observamos os resultados obtidos da captura no wireshark. Após uma análise atenta, concluímos que se o packet tiver 0x0806 como valor do parâmetro *type* é um pacote ARP. Se tiver o valor 0x0800 é um pacote IP. Sendo um pacote IP, se tiver o valor 1 no campo do protocolo então é um pacote ICMP.

De modo a determinar o comprimento do pacote recetor, analisamos o ICMP *Reply Packet* e observamos a *frame length*. A *frame length* é de 98 bytes (anexo 1).

A *loopback interface* é uma interface virtual que está sempre alcançável desde que pelo menos uma das interfaces IP no Switch esteja operacional. É usada para tarefas de debugging e para identificar o dispositivo, uma vez que o *loopback address* não se altera.

Experiência 2 - LANs Virtuais

Nesta experiência criamos duas LANs virtuais, *vlan10* e *vlan11*.

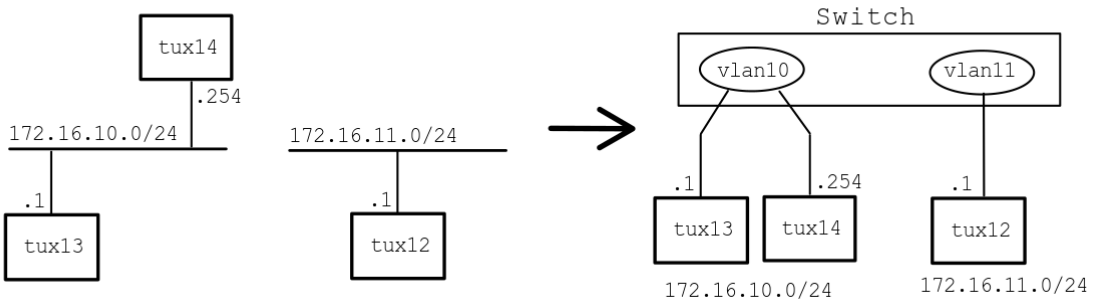


Figura 4 - Configuração da rede na experiência 2

Começamos por configurar a rede do *tux2* e registamos o seu IP e endereço MAC. 172.16.11.1 e 00:21:5a:5a:7e:51 respetivamente.

Para tal introduzimos os seguintes comandos:

```
$ ifconfig eth0 up
$ ifconfig eth0 172.16.11.1/24
```

De seguida conectamos a porta série do *tux* ao Switch de modo a aceder à consola e criar e configurar ambas as VLANs. Os comandos usados na criação e configuração destas podem ser consultados na secção switch dos comandos e configurações nos Anexos.

Os seguintes comando foram responsáveis por criar as VLANs:

Para a *vlan10*:

```
configure terminal
vlan 10
end
```

Para a *vlan11*:

```
configure terminal
vlan 11
end
```

Os *tux13* e 14 foram associados à *vlan10* e o *tux12* associado à *vlan11*, de acordo com a figura acima (figura 4).

Associamos *tux13* à *vlan10* adicionando o port correspondente:

```
configure terminal
interface fastethernet 0/3  # tux13 eth0
switchport mode access
switchport access vlan 10
end
```

Associamos *tux14* à *vlan10* adicionando o port correspondente:

```
configure terminal
interface fastethernet 0/4  # tux14 eth0
switchport mode access
switchport access vlan 10
end
```

Associamos *tux12* à *vlan11* adicionando o port correspondente:

```
configure terminal
interface fastethernet 0/5  # tux12 eth0
switchport mode access
switchport access vlan 11
end
```

Depois de configuradas as LANs virtuais, realizamos vários *pings*.
Do *tux13* para o *tux14* e para o *tux12*, um *ping broadcast* a partir do *tux13* e um ping *broadcast* a partir do *tux12*.
Após análise dos logs das capturas no wireshark, é de notar que os *pings* realizados a partir do *tux13* não alcançaram o *tux12* e os pings realizados no *tux12* não alcançaram o 3 nem 4, o que faz sentido uma vez que não existe conexão entre as duas LANs.
Assim, somos levados a concluir que existem dois domínios de *broadcast*, um na vlan10 e outro na vlan11**.**

Experiência 3 - Configuração de Router

O objetivo desta experiência era sobretudo aprender mais sobre como configurar um router e de que forma era feito o roteamento dos pacotes, isto é, que saltos davam até chegar ao destino final e porquê.
Para configurar uma route estática basta introduzir o comando ***ip route <dest> <mask> <source>*** (por exemplo no nosso caso será utilizado ***ip route 172.16.10.0 255.255.255.0 172.16.11.253*** para estabelecer um caminho entre o router e a vlan10 na experiência 4).
De seguida foi necessário interpretar o que fazia o NAT e concluiu-se que servia para estabelecer a comunicação entre uma rede local e uma rede externa, de forma a que a rede externa não necessita de ter qualquer conhecimento sobre o interior da rede local, pois é o NAT que faz esta correspondência.
Para configurar o NAT no Router Cisco, é necessário primeiramente configurar as suas interfaces fe0/fe1 e indicar qual delas é interna à rede (***ip nat inside***) ou externa à rede (***ip nat outside***). De seguida é necessário utilizar os comandos ***ip nat pool ovrld 172.16.2.19 172.16.2.19 prefix-length 24*** e ***ip nat inside source list 1 pool ovrld overload***. O primeiro define os endereços que podem ser usados para fazer a tradução da rede externa para a rede local sendo que no caso apenas um endereço IP tem essa capacidade. O segundo garante que qualquer pacote que seja recebido pela interface interna que seja permitido pela *access list* terá tradução para um endereço IP válido fora da rede. A *access list* foi definida pelo comandos ***access-list 1 permit 172.16.10.0 0.0.0.7*** e ***access-list 1 permit 172.16.11.0 0.0.0.7***, garantindo assim tradução para os primeiros 7 endereços das redes 172.16.10.0/24 e 172.16.11.0/24.

Experiência 4 - Configuração de Router (Lab)

O objetivo desta experiência era primeiro configurar o *tux14* como router de forma a estabelecer uma conexão entre a *vlan10* e *vlan11* e segundo configurar o Router Cisco para criar uma ligação entre a rede local arquiteturada e a internet representada pela figura 1.

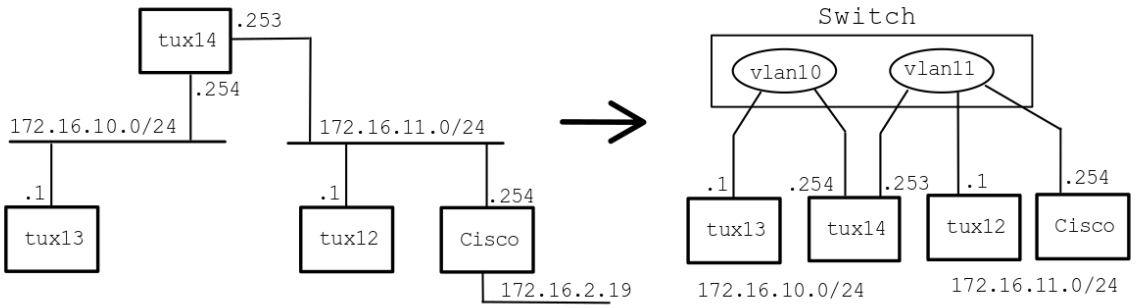


Figura 5 - Configuração da rede na experiência 4

Para isso, começou-se por reproduzir os passos de configuração de endereços IP da experiência 1 e configurar as LANs virtuais de acordo com a experiência 2. De seguida foi configurado o interface *eth1* do *tux14* e adicionado à *vlan11*. Assim, por esta altura a rede encontrava-se configurada da seguinte forma:

IP	MAC	TUX	INTERFACE	VLAN	SWITCH PORT
172.16.10.1	00:21:5a:5a:7d:16	tux13	eth0	VLAN10	3
172.16.10.254	00:21:5a:5a:7b:3f	tux14	eth0	VLAN10	4
172.16.11.253	00:40:f4:6f:b6:a5	tux14	eth1	VLAN11	6
172.16.11.1	00:21:5a:5a:7e:51	tux12	eth0	VLAN11	5

Tabela 3: Interfaces configuradas nos tux

De seguida, foi necessário introduzir o comando **echo 1 > /proc/sys/net/ipv4/ip_forward** no *tux14* para possibilitar a transmissão de pacotes entre os seus interfaces (*eth0* e *eth1*).
O passo seguinte foi configurar as *routes* nos tux13 e tux12 para que conseguissem comunicar um com o outro. Algumas *routes* são adicionadas automaticamente quando se configura a VLAN: o *tux13* tem uma *route* para a *vlan10* (172.16.10.0/24), o tux12 tem uma *route* para a vlan11 (172.16.11.0/24) e o *tux14* tem uma route para ambas as VLANs. Em todas o *gateway* é o *default* (0.0.0.0) indicando assim que não necessita de nenhuma reencaminhamento para atingir estas redes. Além destas, foi utilizado o comando ***route add -net 172.16.11.0/24 gw 172.16.10.254*** no *tux13* para definir que um pacote destinado à rede *172.16.11.0/24* (*vlan11*) poderá ser encaminhado através do endereço *172.16.10.254*, que corresponde à interface *eth0* do *tux14* pertencente, tal como o *tux13*, à *vlan10*, pois este é o ponto de ligação entre as duas VLANs. De forma semelhante mas em sentido inverso foi executado o comando ***route add -net 172.16.10.0/24 gw 172.16.11.253*** para que o *tux12* alcançasse o *tux13*.

Finalmente, foi necessário configurar o router cisco. O interface *fe0* foi conectado ao Switch e o *fe1* conectado ao Lab Router. A isto seguiu-se a configuração da nova conexão no

Switch, adicionando o interface *fe0* do router à *vlan11*, e a configuração do Router Cisco. Para configurar o *router* foi utilizado o script fornecido e alterado de forma a corresponder à arquitetura da rede pretendida na experiência.

Foi necessário também estabelecer *routes* que permitissem conectar os *tux* ao Router Cisco. Para isso foram adicionadas aos *tux12* e *tux14* as rotas *default* com *gateway* atribuido ao endereço IP do Cisco na vlan11 com o comando (***route add default gw 172.16.11.254***) que permitiu redirecionar para o Router Cisco qualquer endereço IP que não tivesse resolução na rede local.

De forma esquemática, as *forwarding tables*[1] configuradas são as seguintes:

tux13

DESTINATION	GATEWAY	GENMASK	FLAGS	METRIC	IFACE
0.0.0.0	172.16.10.254	0.0.0.0	UG	0	eth0
172.16.10.0	0.0.0.0	255.255.255.0	U	0	eth0
172.16.11.0	172.16.10.254	255.255.255.0	UG	0	eth0

Tabela 4: Forwarding Table do tux13

tux14

DESTINATION	GATEWAY	GENMASK	FLAGS	METRIC	IFACE
0.0.0.0	172.16.11.254	0.0.0.0	UG	0	eth1
172.16.10.0	0.0.0.0	255.255.255.0	U	0	eth0
172.16.11.0	0.0.0.0	255.255.255.0	U	0	eth1

Tabela 5: Forwarding Table do tux14

tux12

DESTINATION	GATEWAY	GENMASK	FLAGS	METRIC	IFACE
0.0.0.0	172.16.11.254	0.0.0.0	UG	0	eth0
172.16.10.0	172.16.11.253	255.255.255.0	UG	0	eth0
172.16.11.0	0.0.0.0	255.255.255.0	U	0	eth0

Tabela 6: Forwarding Table do tux12

Por fim, foi feito *ping* das máquinas, tanto do *router* como dos *tux*, para as outras máquinas e para o IP *104.17.113.188* na internet, e do *router* para todos os *tux* com sucesso,

[1] Cada entrada da tabela possui informação sobre o *destination* (IP da máquina/rede de destino), sobre o *gateway* (IP para o qual vai ser rencaminhada a mensagem que por sua vez irá enviar para o destino), sobre a *genmask* (máscara da rede onde 255.255.255.255 representa que o destino é um host e n uma rede), sobre *flags* (indicação de algumas características da rota), sobre *metric* (custo da rota) e sobre o *interface* (placa de rede utilizada para enviar a mesnagem).

concluindo assim que a rede foi bem configurada e que os objetivos da experiência foram alcançados

.

24	17.387755766	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2afa, seq=1/256, ttl=63 (reply in 27)
25	17.387900200	HewlettP_5a:7e:51	Broadcast	ARP	60	Who has 172.16.11.253? Tell 172.16.11.1
26	17.387924575	CameoCom_6f:b6:a5	HewlettP_5a:7e:51	ARP	42	172.16.11.253 is at 00:40:f4:6f:b6:a5
27	17.388019422	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2afa, seq=1/256, ttl=64 (request in 24)
28	17.387584092	HewlettP_5a:7d:16	Broadcast	ARP	60	Who has 172.16.10.254? Tell 172.16.10.1
29	17.387608537	HewlettP_5a:7b:3f	HewlettP_5a:7d:16	ARP	42	172.16.10.254 is at 00:21:5a:5a:7b:3f
30	17.387742775	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2afa, seq=1/256, ttl=64 (reply in 31)
31	17.388032971	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2afa, seq=1/256, ttl=63 (request in 30)
32	18.044263131	Cisco_78:94:86	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/11/00:1e:bd:78:94:80 Cost = 0 Port = 0x8006
33	18.407785427	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2afa, seq=2/512, ttl=63 (reply in 34)
34	18.407902832	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2afa, seq=2/512, ttl=64 (request in 33)
35	18.407759515	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2afa, seq=2/512, ttl=64 (reply in 36)
36	18.407927138	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2afa, seq=2/512, ttl=63 (request in 35)

Figura 6 - Pacotes ICMP enviados pelo tux13 para o tux12 capturados pelas interfaces eth0 e eth1 to tux14

Nestes pings foram observadas mensagens ARP uma vez que os endereços MAC eram desconhecidos entre todos eles. No exemplo da figura 6, é possível observar que o *tux12* enviou uma mensagem ARP em *broadcast* para obter o endereço MAC da *eth1* do *tux14*, à qual esta obteve a resposta por parte do mesmo, com o próprio endereço MAC. O mesmo sucedeu para entre o *tux13* e a *eth0* do *tux14*. De notar que, por exemplo no envio de pacotes do tux13 para o tux12, o endereço MAC pretendido não é o de destino, mas sim o da interface intermediária, uma vez que o *tux13* só necessita de conhecer a interface para a qual vai redirecionar o pacote (*eth0* do *tux14*), e esta sim, conhecer o interface de destino (*eth0* do *tux12*), no caso.

18	6.388328407	HewlettP_5a:7d:16	HewlettP_5a:7b:3f	ARP	42	Who has 172.16.10.254? Tell 172.16.10.1
19	6.388452095	HewlettP_5a:7b:3f	HewlettP_5a:7d:16	ARP	60	172.16.10.254 is at 00:21:5a:5a:7b:3f
20	6.485926488	HewlettP_5a:7b:3f	HewlettP_5a:7d:16	ARP	60	Who has 172.16.10.1? Tell 172.16.10.254
21	6.485948208	HewlettP_5a:7d:16	HewlettP_5a:7b:3f	ARP	42	172.16.10.1 is at 00:21:5a:5a:7d:16
3	1.299647992	172.16.10.1	104.17.113.188	ICMP	98	Echo (ping) request id=0x2c56, seq=1/256, ttl=64 (reply in 4)
4	1.308233004	104.17.113.188	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c56, seq=1/256, ttl=53 (request in 3)
6	2.301390351	172.16.10.1	104.17.113.188	ICMP	98	Echo (ping) request id=0x2c56, seq=2/512, ttl=64 (reply in 7)
7	2.308206162	104.17.113.188	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c56, seq=2/512, ttl=53 (request in 6)
8	3.303317241	172.16.10.1	104.17.113.188	ICMP	98	Echo (ping) request id=0x2c56, seq=3/768, ttl=64 (reply in 9)
9	3.311607456	104.17.113.188	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c56, seq=3/768, ttl=53 (request in 8)
11	4.304364081	172.16.10.1	104.17.113.188	ICMP	98	Echo (ping) request id=0x2c56, seq=4/1024, ttl=64 (reply in 12)
12	4.310071941	104.17.113.188	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c56, seq=4/1024, ttl=53 (request in 11)
13	5.306164442	172.16.10.1	104.17.113.188	ICMP	98	Echo (ping) request id=0x2c56, seq=5/1280, ttl=64 (reply in 14)
14	5.312112484	104.17.113.188	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c56, seq=5/1280, ttl=53 (request in 13)
16	6.307208860	172.16.10.1	104.17.113.188	ICMP	98	Echo (ping) request id=0x2c56, seq=6/1536, ttl=64 (reply in 17)
17	6.312931178	104.17.113.188	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c56, seq=6/1536, ttl=53 (request in 16)
22	7.308363498	172.16.10.1	104.17.113.188	ICMP	98	Echo (ping) request id=0x2c56, seq=7/1792, ttl=64 (reply in 23)

Figura 7 - Pacotes ICMP enviados pelo tux13 para a internet capturados pela eth0 tux13

Na figura 7, é possível observar a comunicação entre o *tux13* e o IP *104.17.113.188* que se encontra na internet. O primeiro apenas necessita de conhecer o endereço MAC da *eth0* do *tux4* para este realizar o mesmo processo, a partir da sua interface eth1, enviando os pacotes para o gateway obtido pela forwarding table até ser possível chegar ao endereço IP de destino. De seguida é realizado o percurso inverso e apenas o eth0 do tux 14 necessita de conhecer o endereço MAC do *tux13*. É por isso que apenas estas mensagens ARP são transmitidas/recebidas pelo *eth0* do *tux13*.

Conclusão

Chegando ao final das experiências pudemos concluir que ficamos a compreender significativamente mais acerca de redes de computadores, nomeadamente no que toca ao processo de encaminhamento de pacotes pelas *forwarding tables*, as resoluções de endereços por análise do protocolo ARP e resoluções DNS, bem como à própria configuração destas redes.

Assim, fomos capazes de configurar a rede pretendida e posteriormente, usando a Aplicação de Download elaborada, conectar à Internet e descarregar ficheiros de servidores FTP. Revelou-se bastante gratificante concluir estas tarefas com sucesso e assim atingir os objetivos esperados neste trabalho prático.

Anexos

Comandos de configuração

De seguida são apresentados os scripts utilizados para a configuração dos *tux*, do switch e do cisco.

tux13

```
#!/bin/bash
ifconfig eth0 up
ifconfig eth0 172.16.10.1/24
ifconfig eth0
route add -net 172.16.11.0/24 gw 172.16.10.254
route add default gw 172.16.10.254
```

tux14

```
#!/bin/bash
ifconfig eth0 up
ifconfig eth0 172.16.10.254/24
ifconfig eth0
ifconfig eth1 up
ifconfig eth1 172.16.11.253/24
ifconfig eth1
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
route add default gw 172.16.11.254
```

tux12

```
#!/bin/bash
ifconfig eth0 up
ifconfig eth0 172.16.11.1/24
ifconfig eth0
route add -net 172.16.10.0/24 gw 172.16.11.253
route add default gw 172.16.11.254
```

Switch

```
!
configure terminal
vlan 10
end
show vlan id 10
!
```

```
configure terminal
interface fastethernet 0/3
switchport mode access
switchport access vlan 10
end
!
configure terminal
interface fastethernet 0/4
switchport mode access
switchport access vlan 10
end
!
!
configure terminal
vlan 11
end
show vlan id 11
!
configure terminal
interface fastethernet 0/5
switchport mode access
switchport access vlan 11
end
!
configure terminal
interface fastethernet 0/6
switchport mode access
switchport access vlan 11
end
!
configure terminal
interface fastethernet 0/7
switchport mode access
switchport access vlan 11
end
```

Cisco

```
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
service password-encryption
!
hostname gnu-rtr1
!
boot-start-marker
boot-end-marker
```

```
!
! card type command needed for slot/vwic-slot 0/0
logging message-counter syslog
logging buffered 51200 warnings
enable secret 5 $1$u53Q$vBawpP8.1YpCT6ypap1zX.
!
no aaa new-model
dot11 syslog
ip source-route
!
!
ip cef
no ip domain lookup
no ipv6 cef
!
multilink bundle-name authenticated
!
!
username root privilege 15 secret 5 $1$8AFR$bNAYevxPFjXFExpnZI2fj.
username cisco password 7 02050D480809
archive
 log config
  hidekeys
!
!
!
interface FastEthernet0/0
 description $ETH-LAN$$ETH-SW-LAUNCH$$INTF-INFO-FE 0$
 ip address 172.16.11.254 255.255.255.0
 ip nat inside
 ip virtual-reassembly
 duplex auto
 speed auto
!
interface FastEthernet0/1
 ip address 172.16.2.19 255.255.255.0
 ip nat outside
 ip virtual-reassembly
 duplex auto
 speed auto
!
ip forward-protocol nd
ip route 0.0.0.0 0.0.0.0 172.16.2.254
ip route 172.16.10.0 255.255.255.0 172.16.11.253
ip http server
ip http access-class 23
ip http authentication local
ip http secure-server
```

```
ip http timeout-policy idle 60 life 86400 requests 10000
!
!
ip nat pool ovrld 172.16.2.19 172.16.2.19 prefix-length 24
ip nat inside source list 1 pool ovrld overload
!
access-list 1 permit 172.16.10.0 0.0.0.7
access-list 1 permit 172.16.11.0 0.0.0.7
!
!
!
control-plane
!
line con 0
 login local
line aux 0
line vty 0 4
 access-class 23 in
 privilege level 15
 login local
 transport input telnet ssh
line vty 5 15
 access-class 23 in
 privilege level 15
 login local
 transport input telnet ssh
!
scheduler allocate 20000 1000
end
```

Logs Capturados

11	12.157673333	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x1fca, seq=1/256, ttl=64 (reply in 14)
12	12.157841162	HewlettP_5a:7b:3f	Broadcast	ARP	60	Who has 172.16.10.1? Tell 172.16.10.254
13	12.157851777	HewlettP_5a:7d:16	HewlettP_5a:7b:3f	ARP	42	172.16.10.1 is at 00:21:5a:5a:7d:16
14	12.157965409	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x1fca, seq=1/256, ttl=64 (request in 11)
15	13.164978941	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x1fca, seq=2/512, ttl=64 (reply in 16)
16	13.165146979	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x1fca, seq=2/512, ttl=64 (request in 15)
17	14.029585606	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/1/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
18	14.188955174	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x1fca, seq=3/768, ttl=64 (reply in 19)
19	14.189088850	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x1fca, seq=3/768, ttl=64 (request in 18)
20	15.216975567	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x1fca, seq=4/1024, ttl=64 (reply in 21)
21	15.217111757	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x1fca, seq=4/1024, ttl=64 (request in 20)
22	16.035525607	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/1/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
23	16.236979368	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x1fca, seq=5/1280, ttl=64 (reply in 24)
24	16.237114441	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x1fca, seq=5/1280, ttl=64 (request in 23)
25	16.878852262	Cisco_78:94:83	CDP/VTP/DTP/PAgP/…	DTP	60	Dynamic Trunk Protocol
26	16.878949970	Cisco_78:94:83	CDP/VTP/DTP/PAgP/…	DTP	90	Dynamic Trunk Protocol
27	17.196933773	HewlettP_5a:7d:16	HewlettP_5a:7b:3f	ARP	42	Who has 172.16.10.254? Tell 172.16.10.1
28	17.197067938	HewlettP_5a:7b:3f	HewlettP_5a:7d:16	ARP	60	172.16.10.254 is at 00:21:5a:5a:7b:3f
29	17.260967474	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x1fca, seq=6/1536, ttl=64 (reply in 30)
30	17.261108204	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x1fca, seq=6/1536, ttl=64 (request in 29)

Anexo 1 - Exp: 1 - Captura de pacotes do tux13 para o tux14 pelo interface eth0 do tux13

16	23.177779536	172.16.11.1	172.16.11.255	ICMP	98	Echo (ping) request id=0x075c, seq=1/256, ttl=64 (no response found!)
17	23.177930886	CameoCom_6f:b6:a5	Broadcast	ARP	60	Who has 172.16.11.1? Tell 172.16.11.253
18	23.177957985	HewlettP_5a:7e:51	CameoCom_6f:b6:a5	ARP	42	172.16.11.1 is at 00:21:5a:5a:7e:51
19	23.178045428	172.16.11.253	172.16.11.1	ICMP	98	Echo (ping) reply id=0x075c, seq=1/256, ttl=64
20	23.178658929	172.16.11.254	172.16.11.1	ICMP	98	Echo (ping) reply id=0x075c, seq=1/256, ttl=255
21	24.059111550	Cisco_78:94:85	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/11/00:1e:bd:78:94:80 Cost = 0 Port = 0x8005
22	24.204859309	172.16.11.1	172.16.11.255	ICMP	98	Echo (ping) request id=0x075c, seq=2/512, ttl=64 (no response found!)
23	24.205000462	172.16.11.253	172.16.11.1	ICMP	98	Echo (ping) reply id=0x075c, seq=2/512, ttl=64
24	24.205649512	172.16.11.254	172.16.11.1	ICMP	98	Echo (ping) reply id=0x075c, seq=2/512, ttl=255
25	25.228869744	172.16.11.1	172.16.11.255	ICMP	98	Echo (ping) request id=0x075c, seq=3/768, ttl=64 (no response found!)
26	25.228984076	172.16.11.253	172.16.11.1	ICMP	98	Echo (ping) reply id=0x075c, seq=3/768, ttl=64
27	25.229661553	172.16.11.254	172.16.11.1	ICMP	98	Echo (ping) reply id=0x075c, seq=3/768, ttl=255
28	26.068169543	Cisco_78:94:85	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/11/00:1e:bd:78:94:80 Cost = 0 Port = 0x8005
29	26.252866817	172.16.11.1	172.16.11.255	ICMP	98	Echo (ping) request id=0x075c, seq=4/1024, ttl=64 (no response found!)
30	26.252978915	172.16.11.253	172.16.11.1	ICMP	98	Echo (ping) reply id=0x075c, seq=4/1024, ttl=64
31	26.253652341	172.16.11.254	172.16.11.1	ICMP	98	Echo (ping) reply id=0x075c, seq=4/1024, ttl=255
32	27.276871552	172.16.11.1	172.16.11.255	ICMP	98	Echo (ping) request id=0x075c, seq=5/1280, ttl=64 (no response found!)
33	27.277019130	172.16.11.253	172.16.11.1	ICMP	98	Echo (ping) reply id=0x075c, seq=5/1280, ttl=64
34	27.277665667	172.16.11.254	172.16.11.1	ICMP	98	Echo (ping) reply id=0x075c, seq=5/1280, ttl=255
35	28.068948903	Cisco_78:94:85	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/11/00:1e:bd:78:94:80 Cost = 0 Port = 0x8005
36	28.300866629	172.16.11.1	172.16.11.255	ICMP	98	Echo (ping) request id=0x075c, seq=6/1536, ttl=64 (no response found!)
37	28.300994651	172.16.11.253	172.16.11.1	ICMP	98	Echo (ping) reply id=0x075c, seq=6/1536, ttl=64
38	28.301652781	172.16.11.254	172.16.11.1	ICMP	98	Echo (ping) reply id=0x075c, seq=6/1536, ttl=255

Anexo 2 - Exp: 4 - Captura de pacotes do tux12 para broadcast pelo interface eth0 do tux12*

2	2.000815023	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
3	3.980744810	Cisco_78:94:83	Cisco_78:94:83	LOOP	60	Reply
4	4.005795433	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
5	6.010619542	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
6	8.019616161	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
7	10.020419322	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
8	12.025368104	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
9	13.980130018	Cisco_78:94:83	Cisco_78:94:83	LOOP	60	Reply
10	14.030467886	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
11	16.035161055	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
12	18.044108240	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
13	20.045029515	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
14	22.049864259	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
15	23.987862401	Cisco_78:94:83	Cisco_78:94:83	LOOP	60	Reply
16	24.054971596	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
17	26.059679166	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
18	28.068695366	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
19	29.918815434	Cisco_78:94:83	CDP/VTP/DTP/PAgP/…	CDP	601	Device ID: gnu-sw1 Port ID: FastEthernet0/3
20	30.069477948	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
21	32.074480324	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
22	33.995621315	Cisco_78:94:83	Cisco_78:94:83	LOOP	60	Reply
23	34.079298601	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
24	36.084620434	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
25	38.093259785	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003

Anexo 3 - Exp: 4 - Captura de pacotes do tux12 para broadcast pelo interface eth0 do tux13 - nenhum pacote ICMP recebido*

2	2.000716789	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
3	4.005663236	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
4	5.960390731	Cisco_78:94:84	Cisco_78:94:84	LOOP	60	Reply
5	6.010876342	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
6	8.015547540	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
7	10.024495261	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
8	12.025334344	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
9	14.030169812	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
10	15.968138473	Cisco_78:94:84	Cisco_78:94:84	LOOP	60	Reply
11	16.035436277	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
12	18.039990629	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
13	20.049026142	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
14	21.899341767	Cisco_78:94:84	CDP/VTP/DTP/PAgP/…	CDP	601	Device ID: gnu-sw1 Port ID: FastEthernet0/4
15	22.049795592	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
16	24.054950868	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
17	25.975903675	Cisco_78:94:84	Cisco_78:94:84	LOOP	60	Reply
18	26.059791085	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
19	28.064951250	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
20	30.073542845	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
21	32.074421459	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
22	34.079232132	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
23	35.987796076	Cisco_78:94:84	Cisco_78:94:84	LOOP	60	Reply
24	36.084212872	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
25	38.089136062	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
26	40.098156140	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
27	42.098984327	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004

Anexo 4 - Exp: 4 - Captura de pacotes do tux12 para broadcast pelo interface eth0 do tux14 - nenhum pacote ICMP recebido*

12	16.181806766	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x327f, seq=1/256, ttl=64 (no response fou…
13	16.181968588	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x327f, seq=1/256, ttl=64
14	17.193913428	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x327f, seq=2/512, ttl=64 (no response fou…
15	17.194083980	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x327f, seq=2/512, ttl=64
16	18.040004526	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
17	18.217880690	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x327f, seq=3/768, ttl=64 (no response fou…
18	18.218034340	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x327f, seq=3/768, ttl=64
19	19.241876657	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x327f, seq=4/1024, ttl=64 (no response fo…
20	19.242037431	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x327f, seq=4/1024, ttl=64
21	20.049006215	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
22	20.265873115	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x327f, seq=5/1280, ttl=64 (no response fo…
23	20.266029488	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x327f, seq=5/1280, ttl=64
24	21.191842564	HewlettP_5a:7b:3f	HewlettP_5a:7d:16	ARP	60	Who has 172.16.10.1? Tell 172.16.10.254
25	21.191864913	HewlettP_5a:7d:16	HewlettP_5a:7b:3f	ARP	42	172.16.10.1 is at 00:21:5a:5a:7d:16
26	21.289899255	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x327f, seq=6/1536, ttl=64 (no response fo…
27	21.290045153	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x327f, seq=6/1536, ttl=64
28	22.049815425	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
29	22.234584473	Cisco_78:94:83	Cisco_78:94:83	LOOP	60	Reply
30	22.313874273	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x327f, seq=7/1792, ttl=64 (no response fo…
31	22.314027713	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x327f, seq=7/1792, ttl=64
32	23.337888472	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x327f, seq=8/2048, ttl=64 (no response fo…
33	23.338059093	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x327f, seq=8/2048, ttl=64

Anexo 5 - Exp: 4 - Captura de pacotes do tux13 para broadcast pelo interface eth0 do tux13*

15	22.200593950	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x327f, seq=1/256, ttl=64 (no response fou…
16	22.200625798	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x327f, seq=1/256, ttl=64
17	23.212706513	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x327f, seq=2/512, ttl=64 (no response fou…
18	23.212742482	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x327f, seq=2/512, ttl=64
19	24.058893683	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
20	24.236666241	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x327f, seq=3/768, ttl=64 (no response fou…
21	24.236696902	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x327f, seq=3/768, ttl=64
22	25.260069202	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x327f, seq=4/1024, ttl=64 (no response fo…
23	25.260700351	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x327f, seq=4/1024, ttl=64
24	26.067990308	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
25	26.284663012	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x327f, seq=5/1280, ttl=64 (no response fo…
26	26.284693534	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x327f, seq=5/1280, ttl=64
27	27.210500671	HewlettP_5a:7b:3f	HewlettP_5a:7d:16	ARP	42	Who has 172.16.10.1? Tell 172.16.10.254
28	27.210646432	HewlettP_5a:7d:16	HewlettP_5a:7b:3f	ARP	60	172.16.10.1 is at 00:21:5a:5a:7d:16
29	27.308687275	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x327f, seq=6/1536, ttl=64 (no response fo…
30	27.308712139	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x327f, seq=6/1536, ttl=64
31	28.068788813	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
32	28.253438896	Cisco_78:94:84	Cisco_78:94:84	LOOP	60	Reply
33	28.332663695	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x327f, seq=7/1792, ttl=64 (no response fo…
34	28.332694076	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x327f, seq=7/1792, ttl=64
35	29.356688656	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x327f, seq=8/2048, ttl=64 (no response fo…
36	29.356725183	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x327f, seq=8/2048, ttl=64
37	30.073599626	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
38	30.380667241	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x327f, seq=9/2304, ttl=64 (no response fo…
39	30.380702930	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x327f, seq=9/2304, ttl=64
40	31.404673693	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x327f, seq=10/2560, ttl=64 (no response f…
41	31.404709453	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x327f, seq=10/2560, ttl=64

Anexo 6 - Exp: 4 - Captura de pacotes do tux13 para broadcast pelo interface eth0 do tux14*

12	14.614629067	HewlettP_5a:7d:16	Broadcast	ARP	42	Who has 172.16.10.254? Tell 172.16.10.1
13	14.614773429	HewlettP_5a:7b:3f	HewlettP_5a:7d:16	ARP	60	172.16.10.254 is at 00:21:5a:5a:7b:3f
14	14.614781461	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x21d8, seq=1/256, ttl=64 (reply in 15)
15	14.614921353	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x21d8, seq=1/256, ttl=64 (request in 14)
16	15.636727617	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x21d8, seq=2/512, ttl=64 (reply in 17)
17	15.636868278	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x21d8, seq=2/512, ttl=64 (request in 16)
18	16.039337055	Cisco_78:94:81	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8001
19	16.660737305	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x21d8, seq=3/768, ttl=64 (reply in 20)
20	16.660875102	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x21d8, seq=3/768, ttl=64 (request in 19)
21	17.684730090	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x21d8, seq=4/1024, ttl=64 (reply in 22)
22	17.684866421	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x21d8, seq=4/1024, ttl=64 (request in 21)
23	18.044183342	Cisco_78:94:81	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8001
24	18.708736984	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x21d8, seq=5/1280, ttl=64 (reply in 25)
25	18.708874502	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x21d8, seq=5/1280, ttl=64 (request in 24)
26	19.617971322	HewlettP_5a:7b:3f	HewlettP_5a:7d:16	ARP	60	Who has 172.16.10.1? Tell 172.16.10.254
27	19.617990598	HewlettP_5a:7d:16	HewlettP_5a:7b:3f	ARP	42	172.16.10.1 is at 00:21:5a:5a:7d:16
28	19.732724532	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x21d8, seq=6/1536, ttl=64 (reply in 29)
29	19.732866659	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x21d8, seq=6/1536, ttl=64 (request in 28)
30	20.049103172	Cisco_78:94:81	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8001
31	20.756732613	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x21d8, seq=7/1792, ttl=64 (reply in 32)
32	20.756873413	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x21d8, seq=7/1792, ttl=64 (request in 31)
33	21.780735386	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x21d8, seq=8/2048, ttl=64 (reply in 34)
34	21.780873113	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x21d8, seq=8/2048, ttl=64 (request in 33)
35	22.054016366	Cisco_78:94:81	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8001
36	22.783897589	Cisco_78:94:81	Cisco_78:94:81	LOOP	60	Reply
37	22.804735784	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x21d8, seq=9/2304, ttl=64 (reply in 38)
38	22.804866317	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x21d8, seq=9/2304, ttl=64 (request in 37)

Anexo 7 - Exp: 2 - Captura de pacotes do tux13 para o tux14 e tux12 pelo interface eth0 do tux13 - o tux12 não foi alcançado

24	17.387755766	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2afa, seq=1/256, ttl=63 (reply in 27)
25	17.387900200	HewlettP_5a:7e:51	Broadcast	ARP	60	Who has 172.16.11.253? Tell 172.16.11.1
26	17.387924575	CameoCom_6f:b6:a5	HewlettP_5a:7e:51	ARP	42	172.16.11.253 is at 00:40:f4:6f:b6:a5
27	17.388019422	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2afa, seq=1/256, ttl=64 (request in 24)
28	17.387584092	HewlettP_5a:7d:16	Broadcast	ARP	60	Who has 172.16.10.254? Tell 172.16.10.1
29	17.387608537	HewlettP_5a:7b:3f	HewlettP_5a:7d:16	ARP	42	172.16.10.254 is at 00:21:5a:5a:7b:3f
30	17.387742775	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2afa, seq=1/256, ttl=64 (reply in 31)
31	17.388032971	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2afa, seq=1/256, ttl=63 (request in 30)
32	18.044263131	Cisco_78:94:86	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/11/00:1e:bd:78:94:80 Cost = 0 Port = 0x8006
33	18.407785427	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2afa, seq=2/512, ttl=63 (reply in 34)
34	18.407902832	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2afa, seq=2/512, ttl=64 (request in 33)
35	18.407759515	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2afa, seq=2/512, ttl=64 (reply in 36)
36	18.407927138	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2afa, seq=2/512, ttl=63 (request in 35)
37	18.816182851	Cisco_78:94:84	CDP/VTP/DTP/PAgP/…	CDP	601	Device ID: gnu-sw1 Port ID: FastEthernet0/4
38	19.021443242	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
39	19.431787182	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2afa, seq=3/768, ttl=63 (reply in 40)
40	19.431904379	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2afa, seq=3/768, ttl=64 (request in 39)
41	19.431760642	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2afa, seq=3/768, ttl=64 (reply in 42)
42	19.431934900	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2afa, seq=3/768, ttl=63 (request in 41)
43	20.049227679	Cisco_78:94:86	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/11/00:1e:bd:78:94:80 Cost = 0 Port = 0x8006
44	20.455774900	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2afa, seq=4/1024, ttl=63 (reply in 45)
45	20.455920871	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2afa, seq=4/1024, ttl=64 (request in 44)
46	20.455748988	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2afa, seq=4/1024, ttl=64 (reply in 47)
47	20.455946364	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2afa, seq=4/1024, ttl=63 (request in 46)
48	21.022555895	Cisco_78:94:84	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8004
49	21.466683941	Cisco_78:94:86	Cisco_78:94:86	LOOP	60	Reply
50	21.479757169	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2afa, seq=5/1280, ttl=63 (reply in 51)
51	21.479870315	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2afa, seq=5/1280, ttl=64 (request in 50)

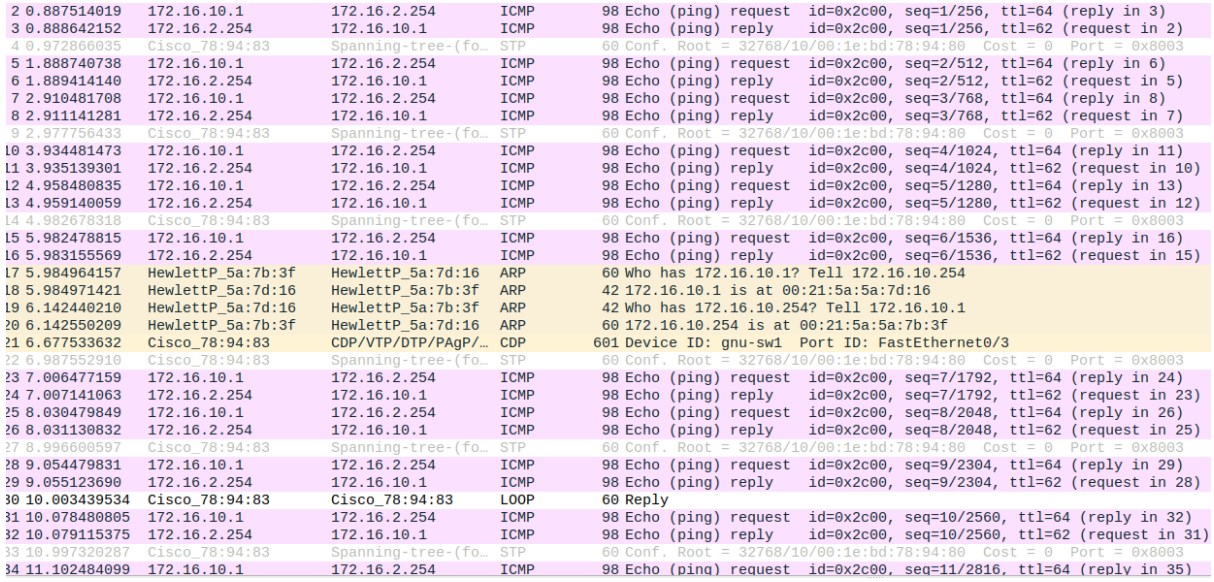
Anexo 8 - Exp: 4 - Captura de pacotes do tux13 para o tux12 pelo interface eth0 e eth1 do tux14

13	14.841622737	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x2a5f, seq=1/256, ttl=64 (reply in 14)
14	14.841795525	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2a5f, seq=1/256, ttl=64 (request in 13)
15	15.225100343	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
16	15.846834573	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x2a5f, seq=2/512, ttl=64 (reply in 17)
17	15.846972719	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2a5f, seq=2/512, ttl=64 (request in 16)
18	15.898314164	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xd8287826
19	16.870834692	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x2a5f, seq=3/768, ttl=64 (reply in 20)
20	16.871003010	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2a5f, seq=3/768, ttl=64 (request in 19)
21	17.234354313	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
22	17.894834253	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x2a5f, seq=4/1024, ttl=64 (reply in 23)
23	17.894975611	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2a5f, seq=4/1024, ttl=64 (request in 22)
24	18.918834302	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x2a5f, seq=5/1280, ttl=64 (reply in 25)
25	18.918969026	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2a5f, seq=5/1280, ttl=64 (request in 24)
26	19.235193666	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
27	19.846798116	HewlettP_5a:7d:16	HewlettP_5a:7b:3f	ARP	42	Who has 172.16.10.254? Tell 172.16.10.1
28	19.846922923	HewlettP_5a:7b:3f	HewlettP_5a:7d:16	ARP	60	172.16.10.254 is at 00:21:5a:5a:7b:3f
29	19.942831907	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x2a5f, seq=6/1536, ttl=64 (reply in 30)
30	19.942959297	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2a5f, seq=6/1536, ttl=64 (request in 29)
31	20.015330791	Cisco_78:94:83	Cisco_78:94:83	LOOP	60	Reply
32	20.051898080	HewlettP_5a:7b:3f	HewlettP_5a:7d:16	ARP	60	Who has 172.16.10.1? Tell 172.16.10.254
33	20.051908766	HewlettP_5a:7d:16	HewlettP_5a:7b:3f	ARP	42	172.16.10.1 is at 00:21:5a:5a:7d:16
34	20.966831607	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x2a5f, seq=7/1792, ttl=64 (reply in 35)
35	20.966966889	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2a5f, seq=7/1792, ttl=64 (request in 34)
36	21.239825260	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
37	23.244909845	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
38	25.249785396	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
39	26.457580395	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x2a69, seq=1/256, ttl=64 (reply in 40)
40	26.457750179	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2a69, seq=1/256, ttl=64 (request in 39)
41	27.258588470	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
42	27.462828828	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x2a69, seq=2/512, ttl=64 (reply in 43)
43	27.462996517	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2a69, seq=2/512, ttl=64 (request in 42)

Anexo 9 - Exp: 4 - Captura de pacotes do tux13 para o eth0 e eth1 do tux14 e tux2 pelo interface eth0 do tux13

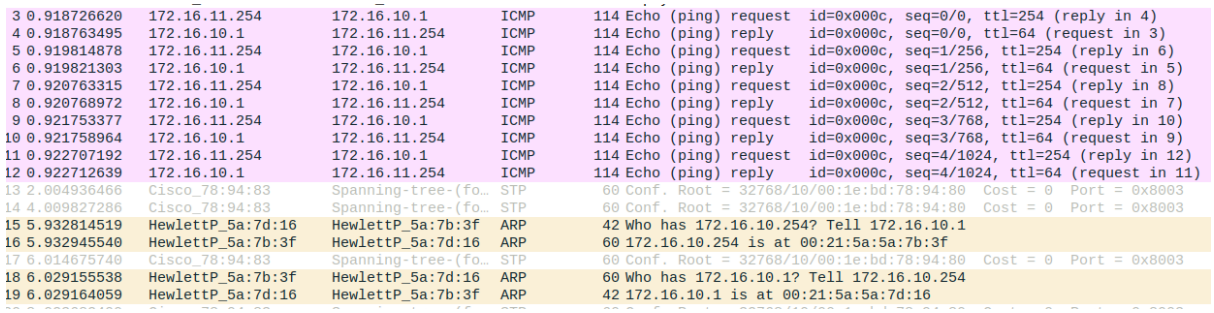
3	1.299647992	172.16.10.1	104.17.113.188	ICMP	98	Echo (ping) request id=0x2c56, seq=1/256, ttl=64 (reply in 4)
4	1.308233004	104.17.113.188	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c56, seq=1/256, ttl=53 (request in 3)
5	2.000861349	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
6	2.301390351	172.16.10.1	104.17.113.188	ICMP	98	Echo (ping) request id=0x2c56, seq=2/512, ttl=64 (reply in 7)
7	2.308206162	104.17.113.188	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c56, seq=2/512, ttl=53 (request in 6)
8	3.303317241	172.16.10.1	104.17.113.188	ICMP	98	Echo (ping) request id=0x2c56, seq=3/768, ttl=64 (reply in 9)
9	3.311607456	104.17.113.188	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c56, seq=3/768, ttl=53 (request in 8)
10	4.005666641	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
11	4.304364081	172.16.10.1	104.17.113.188	ICMP	98	Echo (ping) request id=0x2c56, seq=4/1024, ttl=64 (reply in 12)
12	4.310071941	104.17.113.188	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c56, seq=4/1024, ttl=53 (request in 11)
13	5.306164442	172.16.10.1	104.17.113.188	ICMP	98	Echo (ping) request id=0x2c56, seq=5/1280, ttl=64 (reply in 14)
14	5.312112484	104.17.113.188	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c56, seq=5/1280, ttl=53 (request in 13)
15	6.010557044	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
16	6.307208860	172.16.10.1	104.17.113.188	ICMP	98	Echo (ping) request id=0x2c56, seq=6/1536, ttl=64 (reply in 17)
17	6.312931178	104.17.113.188	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c56, seq=6/1536, ttl=53 (request in 16)
18	6.388328407	HewlettP_5a:7d:16	HewlettP_5a:7b:3f	ARP	42	Who has 172.16.10.254? Tell 172.16.10.1
19	6.388452095	HewlettP_5a:7b:3f	HewlettP_5a:7d:16	ARP	60	172.16.10.254 is at 00:21:5a:5a:7b:3f
20	6.485926488	HewlettP_5a:7b:3f	HewlettP_5a:7d:16	ARP	60	Who has 172.16.10.1? Tell 172.16.10.254
21	6.485948208	HewlettP_5a:7d:16	HewlettP_5a:7b:3f	ARP	42	172.16.10.1 is at 00:21:5a:5a:7d:16
22	7.308363498	172.16.10.1	104.17.113.188	ICMP	98	Echo (ping) request id=0x2c56, seq=7/1792, ttl=64 (reply in 23)
23	7.314266843	104.17.113.188	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c56, seq=7/1792, ttl=53 (request in 22)
24	8.015479689	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
25	8.310358539	172.16.10.1	104.17.113.188	ICMP	98	Echo (ping) request id=0x2c56, seq=8/2048, ttl=64 (reply in 26)
26	8.316237509	104.17.113.188	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c56, seq=8/2048, ttl=53 (request in 25)
27	9.312343814	172.16.10.1	104.17.113.188	ICMP	98	Echo (ping) request id=0x2c56, seq=9/2304, ttl=64 (reply in 28)
28	9.318461010	104.17.113.188	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c56, seq=9/2304, ttl=53 (request in 27)
29	10.024435184	Cisco_78:94:83	Spanning-tree-(fo…	STP	60	Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
30	10.313554009	172.16.10.1	104.17.113.188	ICMP	98	Echo (ping) request id=0x2c56, seq=10/2560, ttl=64 (reply in 31)
31	10.320080960	104.17.113.188	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c56, seq=10/2560, ttl=53 (request in 30)
32	10.783956622	Cisco_78:94:83	Cisco_78:94:83	LOOP	60	Reply
33	11.315173411	172.16.10.1	104.17.113.188	ICMP	98	Echo (ping) request id=0x2c56, seq=11/2816, ttl=64 (reply in 34)
34	11.320914027	104.17.113.188	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c56, seq=11/2816, ttl=53 (request in 33)

Anexo 11 - Exp: 4 -Captura de pacotes do tux13 para a Internet pelo interface eth0 do tux13



2	0.887514019	172.16.10.1	172.16.2.254	ICMP	98 Echo (ping) request id=0x2c00, seq=1/256, ttl=64 (reply in 3)
3	0.888642152	172.16.2.254	172.16.10.1	ICMP	98 Echo (ping) reply id=0x2c00, seq=1/256, ttl=62 (request in 2)
4	0.972866035	Cisco_78:94:83	Spanning-tree-(fo…	STP	60 Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
5	1.888740738	172.16.10.1	172.16.2.254	ICMP	98 Echo (ping) request id=0x2c00, seq=2/512, ttl=64 (reply in 6)
6	1.889414140	172.16.2.254	172.16.10.1	ICMP	98 Echo (ping) reply id=0x2c00, seq=2/512, ttl=62 (request in 5)
7	2.910481708	172.16.10.1	172.16.2.254	ICMP	98 Echo (ping) request id=0x2c00, seq=3/768, ttl=64 (reply in 8)
8	2.911141281	172.16.2.254	172.16.10.1	ICMP	98 Echo (ping) reply id=0x2c00, seq=3/768, ttl=62 (request in 7)
9	2.977756433	Cisco_78:94:83	Spanning-tree-(fo…	STP	60 Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
10	3.934481473	172.16.10.1	172.16.2.254	ICMP	98 Echo (ping) request id=0x2c00, seq=4/1024, ttl=64 (reply in 11)
11	3.935139301	172.16.2.254	172.16.10.1	ICMP	98 Echo (ping) reply id=0x2c00, seq=4/1024, ttl=62 (request in 10)
12	4.958480835	172.16.10.1	172.16.2.254	ICMP	98 Echo (ping) request id=0x2c00, seq=5/1280, ttl=64 (reply in 13)
13	4.959140059	172.16.2.254	172.16.10.1	ICMP	98 Echo (ping) reply id=0x2c00, seq=5/1280, ttl=62 (request in 12)
14	4.982678318	Cisco_78:94:83	Spanning-tree-(fo…	STP	60 Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
15	5.982478815	172.16.10.1	172.16.2.254	ICMP	98 Echo (ping) request id=0x2c00, seq=6/1536, ttl=64 (reply in 16)
16	5.983155569	172.16.2.254	172.16.10.1	ICMP	98 Echo (ping) reply id=0x2c00, seq=6/1536, ttl=62 (request in 15)
17	5.984964157	HewlettP_5a:7b:3f	HewlettP_5a:7d:16	ARP	60 Who has 172.16.10.1? Tell 172.16.10.254
18	5.984971421	HewlettP_5a:7d:16	HewlettP_5a:7b:3f	ARP	42 172.16.10.1 is at 00:21:5a:5a:7d:16
19	6.142440210	HewlettP_5a:7d:16	HewlettP_5a:7b:3f	ARP	42 Who has 172.16.10.254? Tell 172.16.10.1
20	6.142550209	HewlettP_5a:7b:3f	HewlettP_5a:7d:16	ARP	60 172.16.10.254 is at 00:21:5a:5a:7b:3f
21	6.677533632	Cisco_78:94:83	CDP/VTP/DTP/PAgP/…	CDP	601 Device ID: gnu-sw1 Port ID: FastEthernet0/3
22	6.987552910	Cisco_78:94:83	Spanning-tree-(fo…	STP	60 Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
23	7.006477159	172.16.10.1	172.16.2.254	ICMP	98 Echo (ping) request id=0x2c00, seq=7/1792, ttl=64 (reply in 24)
24	7.007141063	172.16.2.254	172.16.10.1	ICMP	98 Echo (ping) reply id=0x2c00, seq=7/1792, ttl=62 (request in 23)
25	8.030479849	172.16.10.1	172.16.2.254	ICMP	98 Echo (ping) request id=0x2c00, seq=8/2048, ttl=64 (reply in 26)
26	8.031130832	172.16.2.254	172.16.10.1	ICMP	98 Echo (ping) reply id=0x2c00, seq=8/2048, ttl=62 (request in 25)
27	8.996600597	Cisco_78:94:83	Spanning-tree-(fo…	STP	60 Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
28	9.054479831	172.16.10.1	172.16.2.254	ICMP	98 Echo (ping) request id=0x2c00, seq=9/2304, ttl=64 (reply in 29)
29	9.055123690	172.16.2.254	172.16.10.1	ICMP	98 Echo (ping) reply id=0x2c00, seq=9/2304, ttl=62 (request in 28)
30	10.003439534	Cisco_78:94:83	Cisco_78:94:83	LOOP	60 Reply
31	10.078480805	172.16.10.1	172.16.2.254	ICMP	98 Echo (ping) request id=0x2c00, seq=10/2560, ttl=64 (reply in 32)
32	10.079115375	172.16.2.254	172.16.10.1	ICMP	98 Echo (ping) reply id=0x2c00, seq=10/2560, ttl=62 (request in 31)
33	10.997320287	Cisco_78:94:83	Spanning-tree-(fo…	STP	60 Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
34	11.102484099	172.16.10.1	172.16.2.254	ICMP	98 Echo (ping) request id=0x2c00, seq=11/2816, ttl=64 (reply in 35)

Anexo 12 - Exp: 4 -Captura de pacotes do tux13 para a rede externa ao Cisco pelo interface eth0 do tux13



3	0.918726620	172.16.11.254	172.16.10.1	ICMP	114 Echo (ping) request id=0x000c, seq=0/0, ttl=254 (reply in 4)
4	0.918763495	172.16.10.1	172.16.11.254	ICMP	114 Echo (ping) reply id=0x000c, seq=0/0, ttl=64 (request in 3)
5	0.919814878	172.16.11.254	172.16.10.1	ICMP	114 Echo (ping) request id=0x000c, seq=1/256, ttl=254 (reply in 6)
6	0.919821303	172.16.10.1	172.16.11.254	ICMP	114 Echo (ping) reply id=0x000c, seq=1/256, ttl=64 (request in 5)
7	0.920763315	172.16.11.254	172.16.10.1	ICMP	114 Echo (ping) request id=0x000c, seq=2/512, ttl=254 (reply in 8)
8	0.920768972	172.16.10.1	172.16.11.254	ICMP	114 Echo (ping) reply id=0x000c, seq=2/512, ttl=64 (request in 7)
9	0.921753377	172.16.11.254	172.16.10.1	ICMP	114 Echo (ping) request id=0x000c, seq=3/768, ttl=254 (reply in 10)
10	0.921758964	172.16.10.1	172.16.11.254	ICMP	114 Echo (ping) reply id=0x000c, seq=3/768, ttl=64 (request in 9)
11	0.922707192	172.16.11.254	172.16.10.1	ICMP	114 Echo (ping) request id=0x000c, seq=4/1024, ttl=254 (reply in 12)
12	0.922712639	172.16.10.1	172.16.11.254	ICMP	114 Echo (ping) reply id=0x000c, seq=4/1024, ttl=64 (request in 11)
13	2.004936466	Cisco_78:94:83	Spanning-tree-(fo…	STP	60 Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
14	4.009827286	Cisco_78:94:83	Spanning-tree-(fo…	STP	60 Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
15	5.932814519	HewlettP_5a:7d:16	HewlettP_5a:7b:3f	ARP	42 Who has 172.16.10.254? Tell 172.16.10.1
16	5.932945540	HewlettP_5a:7b:3f	HewlettP_5a:7d:16	ARP	60 172.16.10.254 is at 00:21:5a:5a:7b:3f
17	6.014675740	Cisco_78:94:83	Spanning-tree-(fo…	STP	60 Conf. Root = 32768/10/00:1e:bd:78:94:80 Cost = 0 Port = 0x8003
18	6.029155538	HewlettP_5a:7b:3f	HewlettP_5a:7d:16	ARP	60 Who has 172.16.10.1? Tell 172.16.10.254
19	6.029164059	HewlettP_5a:7d:16	HewlettP_5a:7b:3f	ARP	42 172.16.10.1 is at 00:21:5a:5a:7d:16

Anexo 13 - Exp: 4 -Captura de pacotes do Cisco para o tux13 pelo interface eth0 do tux13

**Estas capturas deveriam ter sido realizadas no decorrer da exepriência 2, no entando foram apenas realizadas com a configuração da rede de acordo com a experiência 4, por isso não apresentam os resultados que se esperava que apresentassem, nomeadamente o broadcast do tux12 não deveria alcançar nenhum tux uma vez que não deveria estar conectado a nenhuma outra rede, nomeadamente ao tux4 e o broadcast do tux13 só deveria alcançar o tux14.*

Código

download.h

```
#pragma once

#define MAX_CMD_SIZE  128
#define MAX_RESP_SIZE 1024

#define RESP_WELCOME    220
#define RESP_SPEC_PASS  331
#define RESP_SUC_LOGIN  230
#define RESP_PASV_MODE  227
#define RESP_BIN_MODE   150
#define RESP_TRSF_COMP  226
#define RESP_GDBY       221

struct args_t {
 char protocol[100];
 char user[100];
 char pass[100];
 char host[100];
 char path[100];
 char filename[30];
 char ip[20];
 int port;
};


int get_port(char* protocol);
int parse_args(struct args_t* args, int argc, char** argv);
int connect_socket(char* addr, int port);
int disconnect_socket(int sockfd);
int hostname_to_IP(char* hostname, char* ip);
int ftp_send_cmd(int socket, char* cmd);
int ftp_recv_resp(int socket, char* buffer, int len);
int download_file(int socket, char* filename);
```

download.c

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

int get_port(char* protocol) {
 if      (strcmp(protocol, "ftp") == 0)  return 21;
 else if (strcmp(protocol, "ssh") == 0)  return 22;
 else if (strcmp(protocol, "smtp") == 0) return 25;
 else if (strcmp(protocol, "http") == 0) return 80;
 else if (strcmp(protocol, "pop3") == 0) return 110;
 else                                    return -1;
}

int parse_args(struct args_t* args, int argc, char** argv) {
 if (argc != 2) {
   printf("Wrong number of arguments\n");
   printf("usage: download
ftp://[<user>:<password>@]<host>/<url-path>\n");
   return -1;
 }
  char* ptrI = argv[1];
 char* ptrJ = argv[1];
 char* endPtr = argv[1]+strlen(argv[1]);
 int size;
 int state = 0;

 char* at = strchr(argv[1], '@');
 bool hasLogin = at != NULL;
  while (ptrJ < endPtr && state != 5) {
   switch (state) {
     case (0):
       if (*ptrJ == ':' && *(ptrJ+1) == '/' && *(ptrJ+2) == '/') {
         size = ptrJ-ptrI;
```

```
        strncpy(args->protocol, ptrI, size);
        ptrJ += 3;
        ptrI = ptrJ;
        state = hasLogin ? 1 : 3;
      }
      break;
    case (1):
      if (*ptrJ == ':') {
        size = ptrJ-ptrI;
        strncpy(args->user, ptrI, size);
        ptrI = ptrJ+1;
        state = 2;
      }
      break;
    case (2):
      if (*ptrJ == '@') {
        size = ptrJ-ptrI;
        strncpy(args->pass, ptrI, size);
        ptrI = ptrJ+1;
        state = 3;
      }
      break;
    case (3):
      if (*ptrJ == '/') {
        size = ptrJ-ptrI;
        strncpy(args->host, ptrI, size);
        ptrI = ptrJ+1;
        state = 4;
      }
      else if (ptrJ == endPtr-1) {
        size = ptrJ-ptrI+1;
        strncpy(args->host, ptrI, size);
      }
      break;
    case (4):
      size = endPtr-ptrI;
      strncpy(args->path, ptrI, size);
      ptrI = ptrJ+1;
      state = 5;
      break;
    default:
      break;
  }
```

```
    ptrJ++;
  }
   args->port = get_port(args->protocol);

  if (hasLogin && args->user == NULL && args->pass == NULL || args->host
== NULL || args->port < 0 || strlen(args->path) == 0) {
    printf("Bad input\n");
    return -1;
  }
  if (args->port != 21) {
    printf("Use of protocol '%s' is not allowed", args->protocol);
    return -1;
  }

  if (args->path != NULL) {
    char* ptr = endPtr-1;
    while (*(ptr--) != '/');
    strncpy(args->filename, ptr+2, endPtr-ptr-2);
  }

  if (strlen(args->user) == 0 && strlen(args->pass) == 0) {
    strcpy(args->user, "anonymous");
    strcpy(args->pass, "anonymous");
  }

  printf("Protocol: %s\n", args->protocol);
  printf("User:     %s\n", args->user);
  printf("Pass:     %s\n", args->pass);
  printf("Host:     %s\n", args->host);
  printf("Path:     %s\n", args->path);
  printf("Filename: %s\n", args->filename);
  printf("Port:     %d\n", args->port);
   return 0;
}

int connect_socket(char* addr, int port) {
  int sockfd;
  struct sockaddr_in server_addr;
  size_t bytes;

  /*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);    /*32 bit Internet
address network byte ordered*/
 server_addr.sin_port = htons(port);               /*server TCP port
must be network byte ordered */

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

int disconnect_socket(int sockfd) {
 if (close(sockfd)<0) {
   perror("close()");
   return -1;
 }
 return 0;
}

int hostname_to_IP(char* hostname, char* ip) {
 struct hostent *h;
 h = gethostbyname(hostname);
 if (h == NULL) {
   herror("gethostbyname()");
   exit(-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", ip);

 return 0;
```

```
}


int ftp_send_cmd(int socket, char* cmd) {
 int ret = send(socket, cmd, strlen(cmd), 0);
 if (ret < 0) {
   printf("Fail to send cmd '%s'", cmd);
   return -1;
 }
 return ret;
}

int ftp_recv_resp(int socket, char* buffer, int len) {
 char code[4];
 memset(buffer, 0, len);
 memset(code, 0, 4);
 int off = 0;
 while (len != off) {
   int ret = recv(socket, &buffer[off], len-off, 0);
   if (ret < 0) {
     printf("Fail to recv from socket\n");
     return -1;
   }
   else if (ret > 3) {
     if (off == 0 ) {
       strncpy(code, buffer, 3);
     }
     if (strncmp(buffer, &buffer[off], 3) == 0 && buffer[off+3] == ' ')
{
       break;
     }
   }
   off += ret;
 }
 printf("%s", buffer);
 code[4] = 0;
 return atoi(code);
}


int download_file(int socket, char* filename) {
 int fd = open(filename, O_WRONLY | O_CREAT, S_IRUSR | S_IWUSR |
S_IRGRP | S_IWGRP );
```

```
 if (fd < 1) {
   printf("Fail openning file\n");
   return -1;
 }
 char buf[1];
 while (true) {
   int ret = recv(socket, buf, 1, 0);
   if (ret < 0) {
     printf("Fail to recv from socket\n");
     return -1;
   } else if (ret == 0) {
    break;
   }
   write(fd, buf, ret);
 }
 if (close(fd) < 0) {
   printf("Fail closing file\n");
   return -1;
 }
 return 0;
}

int main(int argc, char** argv) {
 struct args_t args = {"", "", "", "", "", -1};
 if (parse_args(&args, argc, argv) < 0) {
   printf("usage: download
ftp://[<user>:<password>@]<host>/<url-path>\n");
   return -1;
 }

 char cmd[MAX_CMD_SIZE];
 char res[MAX_RESP_SIZE];

 hostname_to_IP(args.host, args.ip);
  printf("\nConnecting to control Socket\n");
 int term_A = connect_socket(args.ip, args.port);
 if (ftp_recv_resp(term_A, res, MAX_RESP_SIZE) != RESP_WELCOME) {
   fprintf(stderr, "Error on connection\n");
   disconnect_socket(term_A);
   return -1;
 }

 sprintf(cmd, "USER %s\r\n", args.user);
```

```
printf("\nSending to control Socket...\n> %s\n", cmd);
ftp_send_cmd(term_A, cmd);
printf("Receiving from control Socket...\n");
if (ftp_recv_resp(term_A, res, MAX_RESP_SIZE) != RESP_SPEC_PASS) {
  fprintf(stderr, "Error setting User\n");
  disconnect_socket(term_A);
  return -1;
}

sprintf(cmd, "PASS %s\r\n", args.pass);
printf("\nSending to control Socket...\n> %s\n", cmd);
ftp_send_cmd(term_A, cmd);
printf("Receiving from control Socket...\n");
if (ftp_recv_resp(term_A, res, MAX_RESP_SIZE) != RESP_SUC_LOGIN) {
  fprintf(stderr, "Error setting Pass\n");
  disconnect_socket(term_A);
  return -1;
}

sprintf(cmd, "PASV\r\n");
printf("\nSending to control Socket...\n> %s\n", cmd);
ftp_send_cmd(term_A, cmd);
printf("Receiving from control Socket...\n");
if (ftp_recv_resp(term_A, res, MAX_RESP_SIZE) != RESP_PASV_MODE) {
  fprintf(stderr, "Error entering pasv mode\n");
  disconnect_socket(term_A);
  return -1;
}

int a, b, c, d, pa, pb;
char* start = strchr(res, '(');
char ip_host[32];
int port;
sscanf(start, "(%d,%d,%d,%d,%d,%d)", &a, &b, &c, &d, &pa, &pb);
sprintf(ip_host, "%d.%d.%d.%d", a, b, c, d);
port = 256*pa + pb;

printf("\nConnecting to data Socket on port: %d\n", port);
int term_B = connect_socket(ip_host, port);
 sprintf(cmd, "RETR %s\r\n", args.path);
printf("\nSending to control Socket...\n> %s\n", cmd);
ftp_send_cmd(term_A, cmd);
printf("Receiving from control Socket...\n");
```

```
if (ftp_recv_resp(term_A, res, MAX_RESP_SIZE) != RESP_BIN_MODE) {
  fprintf(stderr, "Error opening BINARY mode data connection\n");
  disconnect_socket(term_A);
  disconnect_socket(term_B);
  return -1;
}

printf("\nDownloading...\n");
download_file(term_B, args.filename);
printf("Download complete!\n\n");

printf("Receiving from control Socket...\n");
if (ftp_recv_resp(term_A, res, MAX_RESP_SIZE) != RESP_TRSF_COMP) {
  fprintf(stderr, "Error completing transfer\n");
  disconnect_socket(term_A);
  disconnect_socket(term_B);
  return -1;
}

sprintf(cmd, "QUIT\r\n");
printf("\nSending to control Socket...\n> %s\n", cmd);
ftp_send_cmd(term_A, cmd);
printf("Receiving from control Socket...\n");
if (ftp_recv_resp(term_A, res, MAX_RESP_SIZE) != RESP_GDBY) {
  fprintf(stderr, "Error quiting\n");
  disconnect_socket(term_A);
  disconnect_socket(term_B);
  return -1;
}


disconnect_socket(term_A);
disconnect_socket(term_B);
}
```