U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Relatório do 2º trabalho laboratorial

Redes de Computadores

3º ano do Mestrado integrado em engenharia Informática e Computação

Dezembro de 2019

Carlos Jorge Albuquerque up201706735@fe.up.pt

Joaquim Manuel Rodrigues up201704844@fe.up.pt

Maria Helena Ferreira up201704508@fe.up.pt

Índice

Resumo

Este relatório foi realizado no âmbito da disciplina de Redes de Computadores. Pretendia-se desenvolver uma aplicação download e configurar e analisar uma rede de computadores, tendo por isso o relatório duas seções principais correspondentes aos dois principais objetivos:

- A primeira secção corresponde à aplicação de download onde são feitas experiências com aplicação Telnet, se especifica a arquitetura da aplicação e são apresentados os resultados da sua execução, assim como a sua análise;
- A segunda secção descreve a configuração e análise da rede de computadores que foi sendo criada ao longo das seis experiências propostas para este trabalho laboratorial, tendo cada uma delas aspetos importantes da matéria teórica que são também analisados.

O trabalho proposto foi realizado com a linguagem de programação C e em ambiente Linux e concluído com sucesso, dado que todos os objetivos pretendidos foram concretizados.

Introdução

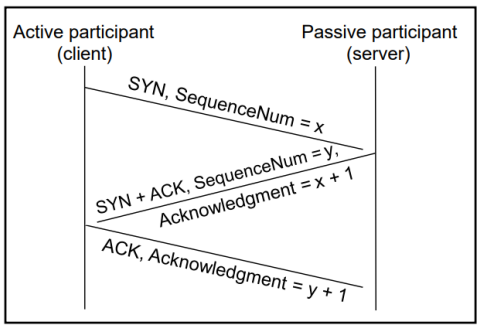
Este projeto foi realizado ao longo de diversas aulas práticas tendo começado com a monitorização dos protocolos de aplicação no trabalho. O protocolo usado na aplicação download foi o FTP (File Transfer Protocol) em conjunto com o servidor de FTP da FEUP. Na segunda parte, este trabalho tinha em vista a utilização de comandos de configuração do Router Cisco e do Cisco Switch. Na última experiência as duas partes unem-se, utilizando a aplicação de download com uma configuração de rede própria permitindo a execução de uma aplicação, a partir de duas VLANs dentro de um switch. Numa das VLAN foi implementado o NAT, estando este ativo, e na outra não, tendo esta última que conseguir ter ligação à Internet para a aplicação de download funcionar corretamente.

Em relação aos objetivos pretendidos na primeira parte, era desejado entender os conceitos de cliente e servidor e as suas peculiaridades em TCP/IP, saber caracterizar os protocolos em aplicações no geral, como definir um URL e descrever o comportamento de um servidor FTP. Depois de compreender o protocolo, deveríamos conseguir implementar um cliente FTP na linguagem C e uma ligação TCP a partir de sockets. Por fim, teríamos de concluir a importância do DNS na conversão de um URL para um IP, permitindo a sua localização num host com domínio determinado.

Parte 1 – A aplicação download

Comecemos por explorar o conceito Cliente-Servidor. Estas duas entidades estão envolvidas na maioria das comunicações Web. O cliente faz pedidos ao servidor enquanto que, por sua vez, o servidor vai respondendo aos pedidos à medida que estes chegam. Pode-se dizer que o cliente é o participante ativo e o servidor o participante passivo desta comunicação.

Aplicando o protocolo TCP a este conceito, obtemos uma comunicação mais bem estruturada. As mensagens entre as duas partes são transmitidas em segmentos (que simulam um fluxo de bytes), cada um desses segmentos possui um número de sequência que permite reconstruir a mensagem inicial pela ordem correta. Quando o número de sequência recebido é o esperado (ocorre um Acknowledgment), a comunicação procede sem grandes problemas como exemplificado na figura seguinte:



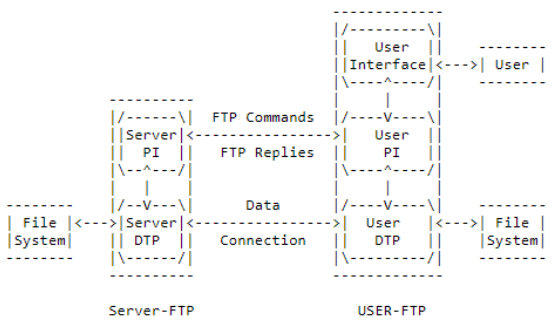
As principais vantagens do protocolo TCP (como o controlo de congestionamento e o modelo ARQ adotado) são descritas mais à frente na Experiência 6. Nesta altura o mais importante é discutir protocolos usados ao nível das aplicações. Estes protocolos ajudam a suportar as dificuldades impostas as aplicações que, por sua vez, vão dificultar ou até impossibilitar a utilização por parte do utilizador final.

O DNS (Domain Name System), definido nos RFCs 1034, 1035 e 2181, é um ótimo exemplo deste tipo de protocolos. O modelo TCP/IP permite-nos identificar de forma inequívoca uma máquina

através do seu endereço IP, mas decorar um conjunto de 4 números não é propriamente cómodo para o utilizador e, "para além disso, navegar nas páginas de uma empresa com IP 128.111.24.41 significa que se a empresa mudar de servidor Web para uma máquina diferente com um endereço de IP diferente, todos os utilizadores têm de ser notificados do novo endereço IP". (Traduzido de Tanenbaum e Wetherall 2011, 611)

O DNS resolve este problema com a atribuição de nomes de domínio a endereços IP. "A essência do DNS é a criação de um esquema de nomes hierárquico e baseado em domínios e de um sistema distribuído de bases de dados para implementar este esquema de nomes" (Traduzido de Tanenbaum e Wetherall 2011, 612). Desta forma a gestão de nomes de domínio cria uma espécie de abstração ao utilizador sobre como os servidores/máquinas na Internet são especificados e também permite a fácil realocação de endereços IP (basta literalmente mudar as entradas na base de dados de acordo com o nome de domínio que tem de ser alterado).

Contudo, o principal protocolo utilizado pela nossa aplicação é o FTP (File Transfer Protocol), definido no RFC 959, do qual foi retirada a seguinte imagem que ilustram o seu funcionamento:



Segundo o RFC, "os objetivos do FTP são: promover a partilhar de ficheiros […], encorajar direta ou indiretamente […] a partilha de ficheiros, proteger o utilizador das variações do sistema de armazenamento de ficheiros entre os hosts e transferir dados fiável e eficientemente". (Traduzido de Tanenbaum e Wetherall 2011, 611)

Tendo em conta estes quatro aspetos e a figura acima, o protocolo FTP é usado em comunicações Cliente-Servidor processando-se da seguinte forma: o *user-protocol interpreter* (User PI) inicia a ligação de controlo, que segue o protocolo Telnet, de onde a mando do utilizador são enviados os comandos FTP para o processo servidor (Server PI). Estes comandos permitem configurar a ligação de dados (Data Connection) que é estabelecida entre os dois processos – uma ligação *full-duplex* especificamente usada para a transferência de dados. É possível configurar a porta, modo de transferência, tipo de representação e estrutura desta ligação. O protocolo exige que as ligações de controlo estejam abertas enquanto decorrer a transferência de dados, sendo responsabilidade do utilizador fechar as ligações de controlo quando deixar de usar o serviço FTP.

Sabendo todos estes pormenores sobre os protocolos e tecnologias utilizadas, a nossa aplicação torna-se bastante simples de compreender. Em primeiro lugar, o utilizador corre a aplicação passando-lhe como argumento o seguinte url:

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

A aplicação começa por dividir o url e isolar as várias partes necessárias (indicadas entre <>) invocando as funções *parseFilename* e *parseUrl* por nós criadas (simples máquinas de estados para fazer o processamento de uma string caracter a caracter). A função *gethostbyname* da biblioteca de C de Linux aplica o protocolo DNS e envia pedidos aos servidores de DNS registados para saber qual o

endereço IP do campo <host> do url. Assim, passamos de um nome de domínio para um endereço IP (chamando a função *getip*) onde é possível abrir um socket TCP (através da função *open_socket*). É precisamente isso que a nossa aplicação faz a seguir: pegando no endereço IP do servidor e numa porta predefinida é aberta uma ligação full-duplex através de um socket que implementa o protocolo TCP. A partir deste momento o socket tem um descritor de ficheiro associado e é possível ler e escrever para ele como se de um ficheiro se tratasse.

De seguida, é invocada a função *config_server* que envia os comandos FTP necessários para a ligação de controlo, por forma a preparar a ligação de dados. Os comandos enviados são, por ordem: USER <user>, PASS <pass> e PASV. Os primeiros dois comandos permitem a autenticação no servidor enquanto que o terceiro configura o servidor a executar em modo passivo (i.e., a responsabilidade de abrir a ligação de dados deixa de ser do servidor e passa a ser do cliente). Invocando a função *get_ip_port* o buffer do socket é lido até ser encontrada a resposta ao comando PASV, que contém o endereço de IP e porta a utilizar no socket para a ligação de dados. A aplicação abre então o socket com esses dados e envia para a ligação de controlo o comando RETR <url-path> que ordena o servidor a enviar o ficheiro no caminho especificado para a ligação de dados (ver função *retrieve_file*). Por fim, o servidor vai enviando os dados para a socket e há medida que eles são lidos são guardados num ficheiro na máquina local cujo caminho é exibido ao utilizador na linha de comandos.

A nossa aplicação, embora simples e focada apenas num único caso de uso, serviu o seu propósito de ser uma aplicação simples de download de ficheiros pelo protocolo FTP e ajudou-nos a aprofundar uma boa parte dos protocolos envolvidos na Apllication Layer do modelo TCP/IP

Parte 2 – Configuração e análise de uma rede

Experiência 1 – Configurar um IP de uma rede

A experiência 1 tinha como objetivo a comunicação de duas máquinas diferentes, fazendo nos compreender sobre a montagem de cabos necessário, a configuração de IP's e sobre protocolo ARP (Address Resolution Protocol). Inicialmente, configuramos os IP's das portas de dois computadores (tux31 e tux34 por nos encontramos na bancada 3), sendo os seus ips 172.16.30.1 e 172.16.30.254 respetivamente. Seguidamente adicionamos as rotas necessárias à tabela de encaminhamento e enviamos o sinal "ping" de um computador para o outro, para verificar o epílogo da nossa ligação.

Inicialmente configuramos os cabos necessários para a comunicação ser efetuada, dado que a experiência não é feita por wireless. A seguir, a configuração dos computadores foi feita com o comando **<ifconfig eth0 [ip]>**, que atribui ao IP da interface o IP passado como segundo argumento. Dada a configuração, "pingamos" de um computador para o outro com os IP's acima mostrados. O resultado da execução do comando ping encontra-se na figura 1 e 2 do anexo na seção da experiência 1.

O protocolo ARP é o responsável pela conversão de endereços da camada de internet em endereços de camada Network. Desta maneira, o comando ping, depois de obtido o endereço MAC (sendo 00:0f:fe:8b:e4:4d e 00:21:5a:5a:7d:74 para o tux31 e tux34 respetivamente) através de pacotes ARP, gera pacotes do protocolo ICMP. Nos frames do tipo Ethernet, os bits na posição vinte e um e vinte e dois identificam o protocolo para o qual deve ser enviado o payload. O log a seguir demonstra a sequência de pacotes assim como explicada:

29	16.496626634	HewlettP_5a:7d:74	G-ProCom_8b:e4:4d	ARP	60 Who has 172.16.30.1? Tell 172.16.30.254
30	16.496650127	G-ProCom_8b:e4:4d	HewlettP_5a:7d:74	ARP	42 172.16.30.1 is at 00:0f:fe:8b:e4:4d
31	16.496901030	172.16.30.254	193.136.28.10	DNS	97 Standard query 0x3292 A 1.debian.pool.ntp.org.netlab.fe.up.pt
32	16.496914934	172.16.30.254	193.136.28.10	DNS	97 Standard query 0x05d6 AAAA 1.debian.pool.ntp.org.netlab.fe.up.pt
33	17.084361941	172.16.30.1	193.136.28.10	DNS	97 Standard query 0xcd70 A 2.debian.pool.ntp.org.netlab.fe.up.pt
34	17.084381324	172.16.30.1	193.136.28.10	DNS	97 Standard query 0x6c1d AAAA 2.debian.pool.ntp.org.netlab.fe.up.pt
35	18.043275294	Cisco_3a:fa:83	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/1/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
36	19.473236403	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x0fb0, seq=1/256, ttl=64 (reply in 37)
37	19.473408418	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x0fb0, seq=1/256, ttl=64 (request in 36)

A rede montada corresponde a um circuito virtual dado que é necessária a configuração do circuito e depois todos os pacotes com a mesma origem e destino seguem o mesmo caminho contrariamente as redes datagrama, que apresentam propriedades opostas. A interface de rede virtual desta experiência é a loopback, utilizada para um computador comunicar na própria rede/computador, normalmente para aceder a servidores na própria máquina ou verificar a correta montagem da rede.

Experiência 2 – Configurar duas LAN's virtuais no switch

Dadas as configurações acima, mantendo-se, portanto, os ip's e o MAC do tux31 e tux34, criamos duas LANs virtuais no switch. A primeira LAN virtual era constituída pelo tux31 e pelo tux34, sendo a vlan30 e a segunda era constituída pelo tux32, com ip 172.16.31.1 e Mac 00:21:5A:61:30:63, constituindo a vlan31. Com a configuração em causa, o tux32 não tem maneira de comunicar com o tux31 e tux34, dado que não se encontram na mesma sub-rede.

Para configurar o switch Cisco começamos por entrar na consola de configuração através do comando <**configure terminal**>. Dentro da consola executamos o comando <**vlan [n]**>, sendo n 30 e 31 dependo do número identificador da vlan que estávamos a configurar no momento. Dada a configuração das duas VLANs foi necessário adicionar as portas do switch às respetivas VLANs, de maneira a criar duas sub-redes distintas. Para esse efeito, foram executados os comandos <**interface fastethernet 0/[w]>, <switchport mode access> e <switchport access vlan [x]>,** sequencialmente, sendo w o número da porta de switch Cisco que estamos a configurar e z o identificador da vlan criada (30 ou 31 nesta experiência). No final utilizamos o comando **<end>** para sair da consola de configuração.

Após a configurações da duas vlans e portas devidas, foi executado um ping tanto do computador 1 como do computador 4 para o computador 2 que não foi sucedido. Esta falha era esperada dado que o tux31 e tux34 se encontram numa sub-rede diferente do tux32, não havendo assim maneira de comunicarem de os primeiros comunicarem com o último, nem vice-versa. No log captura ao fazer broadcast a partir do tux31 conseguimos comprovar que apenas existem pacotes de ARP a alcançar o tux34 e nunca o tux32:

27	17.394806555	HewlettP_5a:7d:74	G-ProCom_8b:e4:4d	ARP	60 Who has 172.16.30.1? Tell 172.16.30.254
28	17.394823217	G-ProCom_8b:e4:4d	HewlettP_5a:7d:74	ARP	42 172.16.30.1 is at 00:0f:fe:8b:e4:4d
29	17.700747717	172.16.30.1	193.136.28.10	DNS	81 Standard query 0x965a A 1.debian.pool.ntp.org
30	17.700766719	172.16.30.1	193.136.28.10	DNS	81 Standard query 0x46c1 AAAA 1.debian.pool.ntp.org
31	18.043482617	Cisco_3a:fa:83	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0
32	20.048201353	Cisco_3a:fa:83	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0
33	22.053043813	Cisco_3a:fa:83	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0
34	22.162774313	172.16.30.254	172.16.1.1	DNS	81 Standard query 0x8575 A 0.debian.pool.ntp.org
35	22.162791436	172.16.30.254	172.16.1.1	DNS	81 Standard query 0x7280 AAAA 0.debian.pool.ntp.org
36	22.481128233	Cisco_3a:fa:83	Cisco_3a:fa:83	LOOP	60 Reply

Experiência 3 – Configurar um router em Linux

Tendo em consideração o objetivo não conseguido da experiência 2, de comunicar do tux31 para o tux32 e vice-versa, nesta experiência configurou-se o tux34 como um router, fazendo com que esta comunicação fosse possível entre as duas sub-redes criadas anteriormente.

O primeiro passo foi ligar a interface ethernet1 do tux34 e configurá-lo com um IP dentro da mesma gama dos IP's na sub-rede do tux32 e adicionar a eth1 à sub-rede do tux32. Dada a configuração, criou-se uma rota no tux31 executando o comando <**route add –net 172.16.31.0/24 gw 172.16.30.254>,** dado encontrarmo-nos na bancada 3**.** Dada a execução do comando, origina-se uma rota onde o primeiro endereço fornecido identifica a gama de endereços e o segundo o IP para o qual se deve encaminhar o pacote, sendo o IP do tux34. De seguida criou-se uma rota com o mesmo comando, mas com os endereços 172.16.30.0/24 e 172.16.31.253 respetivamente como primeiro e segundo endereço. Desta maneira o tux34 passou a ter o IP 172.16.31.253 e mac 00:c0:df:25:26:0a na eth1 e manteve o seu ip na eth0.

Dadas as configurações, atingiu-se o objetivo pretendido, sendo possível fazer com sucesso ping do tux32 para o tux31 e vice-versa. Dado não se apresentarem na mesma sub-rede, o pedido para o IP do tux32 é reencaminhado para o tux34. Dado que este tem conhecimento do IP do tux32 e das rotas para o mesmo torna a transferência possível através da eth1. O processo de resposta é muito semelhante sendo apenas efetuado no sentido inverso, continuando o tux34 a funcionar como router. As tabelas de encaminhamento do router 34 descrevem o caminho percorrido desde o tux31 até ao tux32 (ou vice-versa dependendo do ponto de origem e fim). No log do ping feito a partir do tux31 para o tux32 conseguimos observar isto com muita facilidade. Após o pedido ARP efetuado tux31 para achar o endereço MAC do tux34, a comunicação entre os tux31 e 32 passou a ser conseguida, como é visível nos 8º e 9º pacotes do log seguinte:

2	0.207027415	G-ProCom_8b:e4:4d	HewlettP_5a:7d:74	ARP	42 Who has 172.16.30.254? Tell 172.16.30.1
3	0.207080162	172.16.30.1	172.16.1.1	DNS	97 Standard query 0x6e0b A 2.debian.pool.ntp.org.netlab.fe.up.pt
4	0.207087841	172.16.30.1	172.16.1.1	DNS	97 Standard query 0x6cd4 AAAA 2.debian.pool.ntp.org.netlab.fe.up.pt
5	0.207184468	HewlettP_5a:7d:74	G-ProCom_8b:e4:4d	ARP	60 172.16.30.254 is at 00:21:5a:5a:7d:74
6	0.207440212	172.16.30.254	172.16.30.1	ICMP	125 Destination unreachable (Network unreachable)
7	0.207456638	172.16.30.254	172.16.30.1	ICMP	125 Destination unreachable (Network unreachable)
8	1.042946818	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x1dd9, seq=1/256, ttl=64 (reply in 9)
9	1.043443404	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x1dd9, seq=1/256, ttl=63 (request in 8)

Experiência 4 - Configurar um router comercial e implementar o NAT

O objetivo desta experiência divide-se em duas partes. A primeira consiste na configuração de um router sem implementação do NAT, ligando-o apenas à rede do laboratório. Na segunda parte, dá-se então a implementação do NAT.

De forma a configurar-se o router foi, então, necessário configurar as suas interfaces (comando **<interface gigabitEthernet 0/[interface]>**, definindo o seu endereço IP e máscara (IP 172.16.21.254 para a interface 0/0 e 172.16.1.29 para a interface 0/1, sendo a máscara 255.255.255.0 para as duas)). A configuração do router permite que, mesmo não havendo rota direta para um dos TUX para o envio de pacotes, este envio aconteça através do mesmo. É de notar que sem NAT não foi possível efetuar ping ao router da sala (com ip 172.16.1.254) pois o router não permite a passagem de ips privados para o exterior da internet. Essa falha de comunicação pode ser comprovada no log seguinte:

1	0.000000000	172.16.30.1	172.16.1.254	ICMP	100 Echo (ping) request id=0x61db, seq=1/256, ttl=64 (no response found!)
2	0.999927044	172.16.30.1	172.16.1.254	ICMP	100 Echo (ping) request id=0x61db, seq=2/512, ttl=64 (no response found!)
3	1.999884790	172.16.30.1	172.16.1.254	ICMP	100 Echo (ping) request id=0x61db, seq=3/768, ttl=64 (no response found!)

De seguida, foi necessária a configuração do NAT, sendo usado o NAT Inside (**<nat inside>**) na interface 0/0 e o NAT Outside (**<nat outside>**) na interface 0/1. Utilizando o comando **<no shutdown>**, nenhuma das configurações das interfaces é perdida para o caso de o router ser desligado.

Após o término da configuração das interfaces é necessário garantir a gama de endereços através dos comandos **<ip nat pool ovrld 172.16.1.29 172.16.1.29 prefix 24>**, a lista de permissões de pacotes e acessos (**<access-list 1 permit 172.16.20.0.0.0.7>** e **<access-list 1 permit 172.16.20.0.0.0.7>**) e foram definidas as duas rotas no router com os comandos **<ip route 0.0.0.0 0.0.0.0 172.16.1.254>** e **<ip route 172.16.20.0 255.255.255.0 172.16.21.253>**. Pode então concluir-se que a NAT (Network Address Translation) permite a conservação de endereços IP, permitindo assim que redes IP privadas, isto é, com endereços IP não registados, se conectem à Internet ou rede pública. Para esta rede ter acesso ao exterior, apenas um único endereço IP é exigido. A partir deste momento já foi possível fazer ping do tux31 ao router da sala como se pode ver no log:

```
1 0.000000000  172.16.30.1   172.16.1.254   ICMP  100 Echo (ping) request  id=0x62cb, seq=1/256, ttl=64 (reply in 2)
2 0.000979829  172.16.1.254  172.16.30.1    ICMP  100 Echo (ping) reply    id=0x62cb, seq=1/256, ttl=62 (request in 1)
3 1.001200684  172.16.30.1   172.16.1.254   ICMP  100 Echo (ping) request  id=0x62cb, seq=2/512, ttl=64 (reply in 4)
4 1.001994854  172.16.1.254  172.16.30.1    ICMP  100 Echo (ping) reply    id=0x62cb, seq=2/512, ttl=62 (request in 3)
```

Experiência 5 – DNS

Nesta experiência, o objetivo era configurar o DNS (Domain Name System) de forma a permitir a ligação dos computadores à Internet utilizando nomes de domínios, uma vez que este é responsável por traduzir em endereço IP esses mesmos nomes de domínio.

Para se fazer a configuração do DNS, procedeu-se à alteração do ficheiro **resolv.conf** que passou a ter as entradas **search netlab.fe.up.pt** e **nameserver 172.16.1.1**. Após esta alteração, o Host passa a enviar para o DNS um pacote contendo o pedido dos atributos do domínio. Como resposta, o servidor envia um pacote que, para além de outras informações, contém o endereço IP do destino.

Sem DNS configura, o resultado de um ping do tux31 ao servidor ftp.up.pt falhava pois não conseguia descobrir o endereço IP do servidor. É curioso notar que o tux31 tentou descobrir o IP do servidor ftp através do localhost (endereço IP 127.0.0.1), ou seja, não conhecendo nenhum servidor de DNS o dispositivo procurou nele próprio a conversão de nome para IP que precisava. O log associado é o seguinte:

```
1 0.000000000  127.0.0.1  127.0.0.1  DNS   71 Standard query 0x581f A ftp.up.pt
2 0.000021125  127.0.0.1  127.0.0.1  ICMP  99 Destination unreachable (Port unreachable)
3 0.000077825  127.0.0.1  127.0.0.1  DNS   71 Standard query 0x7360 AAAA ftp.up.pt
4 0.000089359  127.0.0.1  127.0.0.1  ICMP  99 Destination unreachable (Port unreachable)
```

Após acrescentar o servidor netlab.fe.up.pt à lista de servidores DNS, o tux31 conseguiu fazer ping ao ftp.up.pt, descobrindo o IP 193.137.29.15 apenas com dois pacotes de DNS:

```
1 0.000000000  172.16.30.1   172.16.1.1     DNS   71 Standard query 0x8ce6 A ftp.up.pt
2 0.000039219  172.16.30.1   172.16.1.1     DNS   71 Standard query 0x379b AAAA ftp.up.pt
3 0.001865830  172.16.1.1    172.16.30.1    DNS  536 Standard query response 0x8ce6 A ftp.up.pt CNAME mirrors.up.pt A 1
4 0.001896508  172.16.1.1    172.16.30.1    DNS  548 Standard query response 0x379b AAAA ftp.up.pt CNAME mirrors.up.pt
5 0.002446175  172.16.30.1   193.137.29.15  ICMP 100 Echo (ping) request  id=0x64f2, seq=1/256, ttl=64 (reply in 6)
6 0.005000057  193.137.29.15  172.16.30.1    ICMP 100 Echo (ping) reply    id=0x64f2, seq=1/256, ttl=57 (request in 5)
```

A partir deste momento conseguimos também fazer ping a outros sites na Web, mais concretamente o Youtube. Contudo, o número de pedidos DNS foi ligeiramente mais elevado:

```
 7 1.305547688   172.16.31.1          172.16.1.1           DNS        71 Standard query 0x8f50 A i.ytimg.com
 8 1.305558862   172.16.31.1          172.16.1.1           DNS        71 Standard query 0x9e5f AAAA i.ytimg.com
 9 1.305617667   172.16.31.1          172.16.1.1           DNS        73 Standard query 0xcb28 A yt3.ggpht.com
10 1.305625070   172.16.31.1          172.16.1.1           DNS        73 Standard query 0x8737 AAAA yt3.ggpht.com
11 1.305666136   172.16.31.1          172.16.1.1           DNS        80 Standard query 0x2014 A fonts.googleapis.com
12 1.305671374   172.16.31.1          172.16.1.1           DNS        80 Standard query 0x221a AAAA fonts.googleapis.com
13 1.307240542   172.16.1.1           172.16.31.1          DNS       342 Standard query response 0x8f50 A i.ytimg.com A 216.58.
14 1.307313874   172.16.1.1           172.16.31.1          DNS       354 Standard query response 0x9e5f AAAA i.ytimg.com AAAA 2
15 1.307397962   172.16.1.1           172.16.31.1          DNS       389 Standard query response 0xcb28 A yt3.ggpht.com CNAME p
16 1.307474576   172.16.1.1           172.16.31.1          DNS       401 Standard query response 0x8737 AAAA yt3.ggpht.com CNAM
17 1.307672014   172.16.31.1          172.16.1.1           DNS        77 Standard query 0xb640 A fonts.gstatic.com
18 1.307679976   172.16.31.1          172.16.1.1           DNS        77 Standard query 0x9048 AAAA fonts.gstatic.com
19 1.308297083   172.16.1.1           172.16.31.1          DNS       351 Standard query response 0x2014 A fonts.googleapis.com
20 1.308433481   172.16.1.1           172.16.31.1          DNS       363 Standard query response 0x221a AAAA fonts.googleapis.c
21 1.308713610   172.16.1.1           172.16.31.1          DNS       377 Standard query response 0xb640 A fonts.gstatic.com CNA
22 1.309501546   172.16.1.1           172.16.31.1          DNS       389 Standard query response 0x9048 AAAA fonts.gstatic.com
```

Experiência 6 - Conexões TCP

Nesta experiência foi utilizada a aplicação desenvolvida durante a realização do projeto para se observar o comportamento do protocolo TCP. Esta aplicação abre duas ligações TCP, uma para receber e enviar comandos FTP para o servidor (a informação de controlo é, então, transportada nesta ligação) e outra para receber os dados enviados pelo servidor.

Esta conexão desenvolve-se através de um estabelecimento da mesma, seguindo-se a troca de dados e o encerramento da mesma. É utilizado o mecanismo ARQ (Automatic Repeat Request) TCP, que consiste no controlo de erros aquando da transmissão de dados. Para este efeito, o recetor não deixa de processar os frames recebidos quando deteta um erro. O recetor continua a receber frames, enviando no **acknowledgement number** o número do frame que falhou, até o conseguir receber. O emissor, por sua vez, verifica os ACKs reenviando frames não processados/perdidos.

Para além do mecanismo ARQ TCP, também é utilizado um mecanismo de controlo de congestionamento, que controla a taxa de envio de dados em função do congestionamento da rede. Este mecanismo faz também com que o aparecimento de uma segunda conexão TCP (ou mais) exista a queda na taxa de transmissão, dando-se uma divisão igualitária da taxa de transferência (uma demonstração prática da aplicação de uma justiça “Max-Min” que impede a *starvation* dos fluxos).

Para testar estes mecanismos fizemos o download de um ficheiro grande (mais que 1GB) a partir do tux31, de onde se concluiu que a capacidade média de transferência foi de 12 MB/s :

Measurement	Captured	Displayed	Marked
Packets	1179202	1179202 (100.0%)	—
Time span, s	95.799	95.799	—
Average pps	12309.1	12309.1	—
Average packet size, B	980	980	—
Bytes	1155830682	1155830682 (100.0%)	0
Average bytes/s	12 M	12 M	—
Average bits/s	96 M	96 M	—

De seguida fizemos o download do mesmo ficheiro a partir do tux31, mas logo a seguir começámos o download do mesmo ficheiro no tux32 e obtivemos o capacidade média de 7.731KB/s :

Measurement	Captured	Displayed	Marked
Packets	1190650	1190650 (100.0%)	—
Time span, s	149.279	149.279	—
Average pps	7976.0	7976.0	—
Average packet size, B	969	969	—
Bytes	1154112537	1154112537 (100.0%)	0
Average bytes/s	7,731 k	7,731 k	—
Average bits/s	61 M	61 M	—

Note-se que o mecanismo entrou em ação e dividiu a capacidade do sistema pelos dois computadores de modo a não haver starvation para nenhum dos dois, sendo que sobrou a cada um cerca de metade da capacidade original do sistema.

Conclusões

Em suma, depois da realização de todos os objetivos pretendidos, foi possível consolidar todos os conceitos importantes da disciplina de redes e computadores e compreender um protocolo que utilizamos diariamente, até agora, sem ter conhecimento.

É também de salientar que foi possível observar no laboratório conceitos e configurações vistas na teoria nas aulas teóricas. Com a combinação da teoria com a prática o nosso conhecimento ficou inteiramente completado. Por estas razões podemos concluir com grande certeza que o nosso conhecimento e gosto pela área de Redes de Computadores cresceram bastante.

Bibliografia

1. Andrew Tanenbaum, David Wetherall, Computer Networks, 5/E, Prentice Hall, 2011

2. J. Postel, J. Reynolds, RFC 959 – FTP, October 1985

Anexos

Experiência 1

A figura 1 foi tirada ao terminal do tux31, mostrando a tabela de encaminhamento e o ping sucedido do tux31 para o tux34. Na mesma experiência tiramos também foto as configurações e a um pingar sucedido do tux34 para o tux31, sendo, como esperado, efetuado com sucesso. Os correspondentes logs encontram-se na figura 3.

```
tux31:~# arp -e
Address                  HWtype  HWaddress           Flags Mask            Iface
172.16.30.254            ether   00:21:5a:5a:7d:74   C                     eth0
tux31:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.30.254   0.0.0.0         UG    0      0        0 eth0
172.16.30.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
tux31:~# arp -a
? (172.16.30.254) at 00:21:5a:5a:7d:74 [ether] on eth0
tux31:~# arp -d 172.16.30.254
tux31:~# arp -a
? (172.16.30.254) at <incomplete> on eth0
tux31:~# ping 172.16.30.254
PING 172.16.30.254 (172.16.30.254) 56(84) bytes of data.
64 bytes from 172.16.30.254: icmp_seq=1 ttl=64 time=0.190 ms
64 bytes from 172.16.30.254: icmp_seq=2 ttl=64 time=0.295 ms
64 bytes from 172.16.30.254: icmp_seq=3 ttl=64 time=0.278 ms
64 bytes from 172.16.30.254: icmp_seq=4 ttl=64 time=0.225 ms
```

Figura 1 – configuração e pingar do tux31 para o tux34

```
root@tux34:~# ping 172.16.30.254
PING 172.16.30.254 (172.16.30.254) 56(84) bytes of data.
64 bytes from 172.16.30.254: icmp_seq=1 ttl=64 time=0.033 ms
64 bytes from 172.16.30.254: icmp_seq=2 ttl=64 time=0.017 ms
64 bytes from 172.16.30.254: icmp_seq=3 ttl=64 time=0.041 ms
64 bytes from 172.16.30.254: icmp_seq=4 ttl=64 time=0.015 ms
64 bytes from 172.16.30.254: icmp_seq=5 ttl=64 time=0.015 ms
64 bytes from 172.16.30.254: icmp_seq=6 ttl=64 time=0.017 ms
^C
--- 172.16.30.254 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 122ms
rtt min/avg/max/mdev = 0.015/0.023/0.041/0.010 ms
root@tux34:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.30.1     0.0.0.0         UG    0      0        0 eth0
172.16.30.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
```

Figura 2 – configuração e pingar do tux34 para o tux31

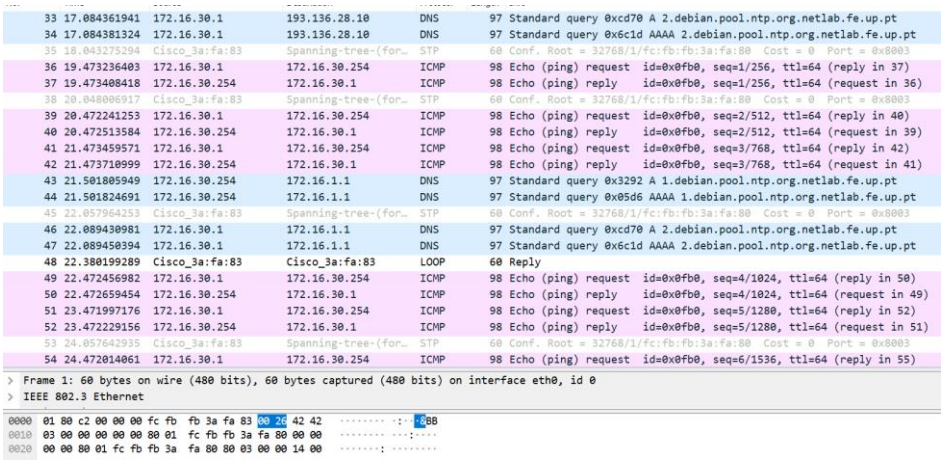
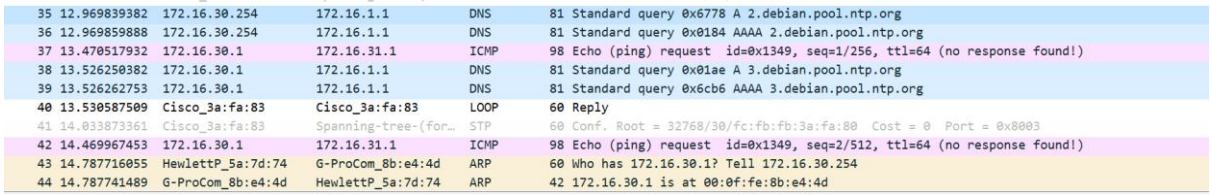


Figura 3 – logs no wireshark

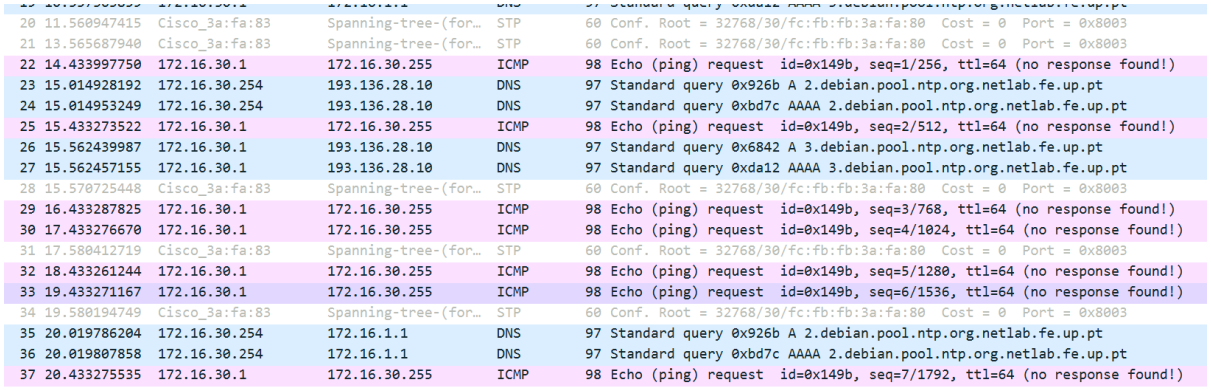
Experiência 2

Como visto anteriormente e seria de esperar o log da experiência 2 mostrar que a comunicação entre o tux31 e o tux34 não foi possível por se encontrarem em sub-redes diferentes.



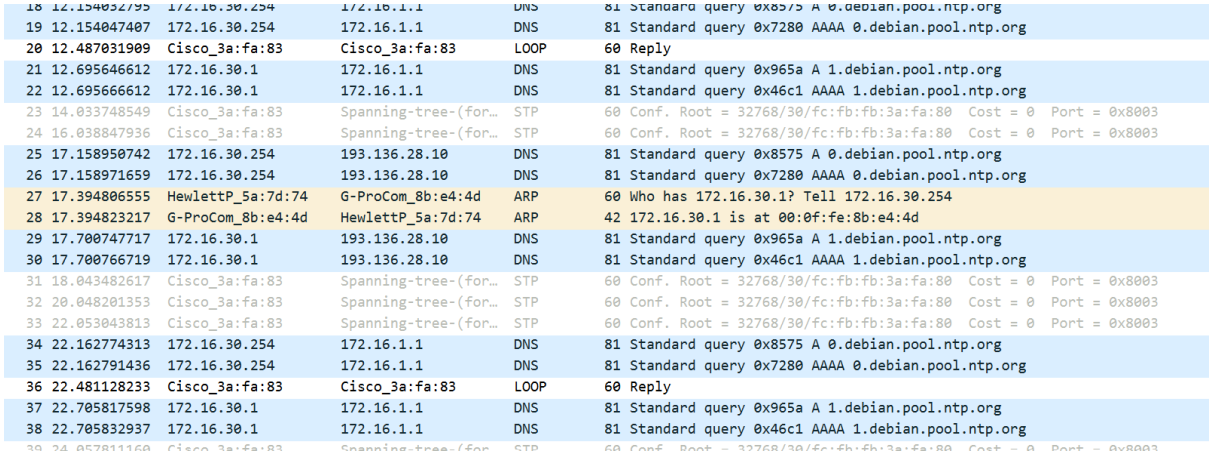
35	12.969839382	172.16.30.254	172.16.1.1	DNS	81 Standard query 0x6778 A 2.debian.pool.ntp.org
36	12.969859888	172.16.30.254	172.16.1.1	DNS	81 Standard query 0x0184 AAAA 2.debian.pool.ntp.org
37	13.470517932	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x1349, seq=1/256, ttl=64 (no response found!)
38	13.526250382	172.16.30.1	172.16.1.1	DNS	81 Standard query 0x01ae A 3.debian.pool.ntp.org
39	13.526262753	172.16.30.1	172.16.1.1	DNS	81 Standard query 0x6cb6 AAAA 3.debian.pool.ntp.org
40	13.530587509	Cisco_3a:fa:83	Cisco_3a:fa:83	LOOP	60 Reply
41	14.033873361	Cisco_3a:fa:83	Spanning-tree-(for…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
42	14.469967453	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x1349, seq=2/512, ttl=64 (no response found!)
43	14.787716055	HewlettP_5a:7d:74	G-ProCom_8b:e4:4d	ARP	60 Who has 172.16.30.1? Tell 172.16.30.254
44	14.787741489	G-ProCom_8b:e4:4d	HewlettP_5a:7d:74	ARP	42 172.16.30.1 is at 00:0f:fe:8b:e4:4d

Figura 4 - logs



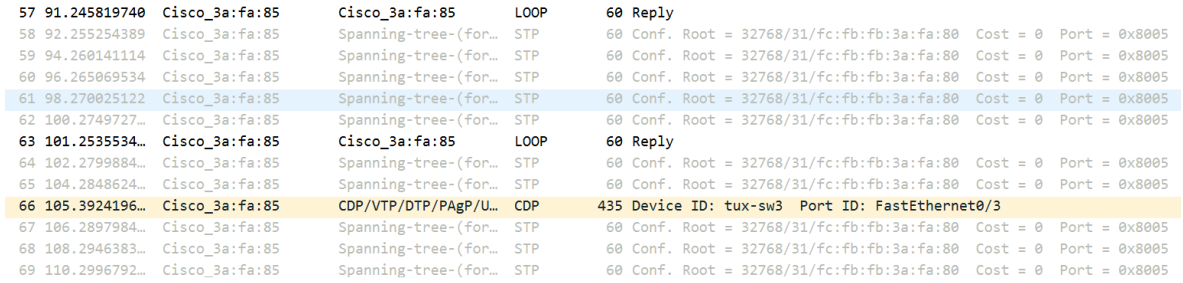
20	11.560947415	Cisco_3a:fa:83	Spanning-tree-(for…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
21	13.565687940	Cisco_3a:fa:83	Spanning-tree-(for…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
22	14.433997750	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x149b, seq=1/256, ttl=64 (no response found!)
23	15.014928192	172.16.30.254	193.136.28.10	DNS	97 Standard query 0x926b A 2.debian.pool.ntp.org.netlab.fe.up.pt
24	15.014953249	172.16.30.254	193.136.28.10	DNS	97 Standard query 0xbd7c AAAA 2.debian.pool.ntp.org.netlab.fe.up.pt
25	15.433273522	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x149b, seq=2/512, ttl=64 (no response found!)
26	15.562439987	172.16.30.1	193.136.28.10	DNS	97 Standard query 0x6842 A 3.debian.pool.ntp.org.netlab.fe.up.pt
27	15.562457155	172.16.30.1	193.136.28.10	DNS	97 Standard query 0xda12 AAAA 3.debian.pool.ntp.org.netlab.fe.up.pt
28	15.570725448	Cisco_3a:fa:83	Spanning-tree-(for…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
29	16.433287825	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x149b, seq=3/768, ttl=64 (no response found!)
30	17.433276670	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x149b, seq=4/1024, ttl=64 (no response found!)
31	17.580412719	Cisco_3a:fa:83	Spanning-tree-(for…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
32	18.433261244	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x149b, seq=5/1280, ttl=64 (no response found!)
33	19.433271167	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x149b, seq=6/1536, ttl=64 (no response found!)
34	19.580194749	Cisco_3a:fa:83	Spanning-tree-(for…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
35	20.019786204	172.16.30.254	172.16.1.1	DNS	97 Standard query 0x926b A 2.debian.pool.ntp.org.netlab.fe.up.pt
36	20.019807858	172.16.30.254	172.16.1.1	DNS	97 Standard query 0xbd7c AAAA 2.debian.pool.ntp.org.netlab.fe.up.pt
37	20.433275535	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x149b, seq=7/1792, ttl=64 (no response found!)

Figura 5 – log do tux31 broadcast 30



19	12.154047407	172.16.30.254	172.16.1.1	DNS	81 Standard query 0x7280 AAAA 0.debian.pool.ntp.org
20	12.487031909	Cisco_3a:fa:83	Cisco_3a:fa:83	LOOP	60 Reply
21	12.695646612	172.16.30.1	172.16.1.1	DNS	81 Standard query 0x965a A 1.debian.pool.ntp.org
22	12.695666612	172.16.30.1	172.16.1.1	DNS	81 Standard query 0x46c1 AAAA 1.debian.pool.ntp.org
23	14.033748549	Cisco_3a:fa:83	Spanning-tree-(for…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
24	16.038847936	Cisco_3a:fa:83	Spanning-tree-(for…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
25	17.158950742	172.16.30.254	193.136.28.10	DNS	81 Standard query 0x8575 A 0.debian.pool.ntp.org
26	17.158971659	172.16.30.254	193.136.28.10	DNS	81 Standard query 0x7280 AAAA 0.debian.pool.ntp.org
27	17.394806555	HewlettP_5a:7d:74	G-ProCom_8b:e4:4d	ARP	60 Who has 172.16.30.1? Tell 172.16.30.254
28	17.394823217	G-ProCom_8b:e4:4d	HewlettP_5a:7d:74	ARP	42 172.16.30.1 is at 00:0f:fe:8b:e4:4d
29	17.700747717	172.16.30.1	193.136.28.10	DNS	81 Standard query 0x965a A 1.debian.pool.ntp.org
30	17.700766719	172.16.30.1	193.136.28.10	DNS	81 Standard query 0x46c1 AAAA 1.debian.pool.ntp.org
31	18.043482617	Cisco_3a:fa:83	Spanning-tree-(for…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
32	20.048201353	Cisco_3a:fa:83	Spanning-tree-(for…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
33	22.053043813	Cisco_3a:fa:83	Spanning-tree-(for…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
34	22.162774313	172.16.30.254	172.16.1.1	DNS	81 Standard query 0x8575 A 0.debian.pool.ntp.org
35	22.162791436	172.16.30.254	172.16.1.1	DNS	81 Standard query 0x7280 AAAA 0.debian.pool.ntp.org
36	22.481128233	Cisco_3a:fa:83	Cisco_3a:fa:83	LOOP	60 Reply
37	22.705817598	172.16.30.1	172.16.1.1	DNS	81 Standard query 0x965a A 1.debian.pool.ntp.org
38	22.705832937	172.16.30.1	172.16.1.1	DNS	81 Standard query 0x46c1 AAAA 1.debian.pool.ntp.org

Figura 6 – log do tux31 broadcast 31



57	91.245819740	Cisco_3a:fa:85	Cisco_3a:fa:85	LOOP	60 Reply
58	92.255254389	Cisco_3a:fa:85	Spanning-tree-(for…	STP	60 Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8005
59	94.260141114	Cisco_3a:fa:85	Spanning-tree-(for…	STP	60 Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8005
60	96.265069534	Cisco_3a:fa:85	Spanning-tree-(for…	STP	60 Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8005
61	98.270025122	Cisco_3a:fa:85	Spanning-tree-(for…	STP	60 Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8005
62	100.2749727…	Cisco_3a:fa:85	Spanning-tree-(for…	STP	60 Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8005
63	101.2535534…	Cisco_3a:fa:85	Cisco_3a:fa:85	LOOP	60 Reply
64	102.2799884…	Cisco_3a:fa:85	Spanning-tree-(for…	STP	60 Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8005
65	104.2848624…	Cisco_3a:fa:85	Spanning-tree-(for…	STP	60 Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8005
66	105.3924196…	Cisco_3a:fa:85	CDP/VTP/DTP/PAgP/U…	CDP	435 Device ID: tux-sw3 Port ID: FastEthernet0/3
67	106.2897984…	Cisco_3a:fa:85	Spanning-tree-(for…	STP	60 Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8005
68	108.2946383…	Cisco_3a:fa:85	Spanning-tree-(for…	STP	60 Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8005
69	110.2996792…	Cisco_3a:fa:85	Spanning-tree-(for…	STP	60 Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8005

Figura 7 – log do tux32 broadcast 30

2	1.282142443	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x0727, seq=1/256, ttl=64 (no response found!)
3	2.004943995	Cisco_3a:fa:85	Spanning-tree-(for…	STP	60 Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8005
4	2.311488214	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x0727, seq=2/512, ttl=64 (no response found!)
5	3.335489660	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x0727, seq=3/768, ttl=64 (no response found!)
6	4.009852301	Cisco_3a:fa:85	Spanning-tree-(for…	STP	60 Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8005
7	4.359483354	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x0727, seq=4/1024, ttl=64 (no response found!)
8	5.383481796	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x0727, seq=5/1280, ttl=64 (no response found!)
9	6.014783095	Cisco_3a:fa:85	Spanning-tree-(for…	STP	60 Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8005
10	6.407483731	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x0727, seq=6/1536, ttl=64 (no response found!)
11	6.569004900	Cisco_3a:fa:85	Cisco_3a:fa:85	LOOP	60 Reply

Figura 8 – log do tux32 broadcast 31

118	79.058102153	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x149b, seq=1/256, ttl=64 (no response found!)
119	79.638757137	172.16.30.254	193.136.28.10	DNS	97 Standard query 0x926b A 2.debian.pool.ntp.org.netlab.fe.up.pt
120	79.638767613	172.16.30.254	193.136.28.10	DNS	97 Standard query 0xbd7c AAAA 2.debian.pool.ntp.org.netlab.fe.up.pt
121	80.057408140	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x149b, seq=2/512, ttl=64 (no response found!)
122	80.186581899	172.16.30.1	193.136.28.10	DNS	97 Standard query 0x6842 A 3.debian.pool.ntp.org.netlab.fe.up.pt
123	80.186588813	172.16.30.1	193.136.28.10	DNS	97 Standard query 0xda12 AAAA 3.debian.pool.ntp.org.netlab.fe.up.pt
124	80.196456199	Cisco_3a:fa:84	Spanning-tree-(for…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
125	81.057463035	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x149b, seq=3/768, ttl=64 (no response found!)
126	82.057475467	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x149b, seq=4/1024, ttl=64 (no response found!)
127	82.205180493	Cisco_3a:fa:84	Spanning-tree-(for…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
128	83.057492439	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x149b, seq=5/1280, ttl=64 (no response found!)
129	84.057535531	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x149b, seq=6/1536, ttl=64 (no response found!)
130	84.205562804	Cisco_3a:fa:84	Spanning-tree-(for…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
131	84.643842002	172.16.30.254	172.16.1.1	DNS	97 Standard query 0x926b A 2.debian.pool.ntp.org.netlab.fe.up.pt
132	84.643852129	172.16.30.254	172.16.1.1	DNS	97 Standard query 0xbd7c AAAA 2.debian.pool.ntp.org.netlab.fe.up.pt
133	85.057572616	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x149b, seq=7/1792, ttl=64 (no response found!)

Figura 9 – log do tux34 broadcast 30

16	10.010717774	172.16.30.254	193.136.28.10	DNS	81 Standard query 0x8575 A 0.debian.pool.ntp.org
17	10.010727412	172.16.30.254	193.136.28.10	DNS	81 Standard query 0x7280 AAAA 0.debian.pool.ntp.org
18	10.246599935	HewlettP_5a:7d:74	G-ProCom_8b:e4:4d	ARP	42 Who has 172.16.30.1? Tell 172.16.30.254
19	10.246827617	G-ProCom_8b:e4:4d	HewlettP_5a:7d:74	ARP	60 172.16.30.1 is at 00:0f:fe:8b:e4:4d
20	10.552790827	172.16.30.1	193.136.28.10	DNS	81 Standard query 0x965a A 1.debian.pool.ntp.org
21	10.552804096	172.16.30.1	193.136.28.10	DNS	81 Standard query 0x46c1 AAAA 1.debian.pool.ntp.org
22	10.895662989	Cisco_3a:fa:84	Spanning-tree-(for…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
23	12.900338374	Cisco_3a:fa:84	Spanning-tree-(for…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
24	14.905338731	Cisco_3a:fa:84	Spanning-tree-(for…	STP	60 Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004

Figura 10 – log do tux34 broadcast 31

Experiência 3

```
tux31:~# ping 172.16.31.1
PING 172.16.31.1 (172.16.31.1) 56(84) bytes of data.
64 bytes from 172.16.31.1: icmp_seq=1 ttl=63 time=0.508 ms
64 bytes from 172.16.31.1: icmp_seq=2 ttl=63 time=0.509 ms
64 bytes from 172.16.31.1: icmp_seq=3 ttl=63 time=0.496 ms
64 bytes from 172.16.31.1: icmp_seq=4 ttl=63 time=0.254 ms
64 bytes from 172.16.31.1: icmp_seq=5 ttl=63 time=0.291 ms
64 bytes from 172.16.31.1: icmp_seq=6 ttl=63 time=0.254 ms
64 bytes from 172.16.31.1: icmp_seq=7 ttl=63 time=0.291 ms
64 bytes from 172.16.31.1: icmp_seq=8 ttl=63 time=0.494 ms
64 bytes from 172.16.31.1: icmp_seq=9 ttl=63 time=0.479 ms
64 bytes from 172.16.31.1: icmp_seq=10 ttl=63 time=0.470 ms
^C
--- 172.16.31.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 8998ms
rtt min/avg/max/mdev = 0.254/0.404/0.509/0.11[illegible] ms
tux31:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.30.254   0.0.0.0         UG    0      0        0 eth0
172.16.30.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
```

Figura 11 - ping do tux31

```
root@tux32:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.31.1  netmask 255.255.255.0  broadcast 172.16.31.255
        inet6 fe80::221:5aff:fe61:3063  prefixlen 64  scopeid 0x20<link>
        ether 00:21:5a:61:30:63  txqueuelen 1000  (Ethernet)
        RX packets 237  bytes 21974 (21.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1517  bytes 137178 (133.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 17

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 819  bytes 82940 (80.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 819  bytes 82940 (80.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@tux32:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.30.0     172.16.31.253   255.255.255.0   UG    0      0        0 eth0
172.16.31.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
```

Figura 12- routes na experiência 3

7	0.207456638	172.16.30.254	172.16.30.1	ICMP	125	Destination unreachable (Network unreachable)
8	1.042946818	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x1dd9, seq=1/256, ttl=64 (reply in 9)
9	1.043443404	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1dd9, seq=1/256, ttl=63 (request in 8)
10	2.004867097	Cisco_3a:fa:83	Spanning-tree-(for…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
11	2.043070575	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x1dd9, seq=2/512, ttl=64 (reply in 12)
12	2.043530264	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1dd9, seq=2/512, ttl=63 (request in 11)
13	3.043064587	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x1dd9, seq=3/768, ttl=64 (reply in 14)
14	3.043309901	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1dd9, seq=3/768, ttl=63 (request in 13)
15	4.009631351	Cisco_3a:fa:83	Spanning-tree-(for…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
16	4.043118891	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x1dd9, seq=4/1024, ttl=64 (reply in 17)
17	4.043622710	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1dd9, seq=4/1024, ttl=63 (request in 16)
18	5.021614243	Cisco_3a:fa:83	Cisco_3a:fa:83	LOOP	60	Reply
19	5.043066202	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x1dd9, seq=5/1280, ttl=64 (reply in 20)
20	5.043524462	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1dd9, seq=5/1280, ttl=63 (request in 19)

Figura 13 – tux31 pin 1

2	1.121357411	172.16.30.1	172.16.31.253	ICMP	98	Echo (ping) request id=0x1db5, seq=1/256, ttl=64 (reply in 3)
3	1.121527396	172.16.31.253	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1db5, seq=1/256, ttl=64 (request in 2)
4	2.009872438	Cisco_3a:fa:83	Spanning-tree-(for…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
5	2.120358397	172.16.30.1	172.16.31.253	ICMP	98	Echo (ping) request id=0x1db5, seq=2/512, ttl=64 (reply in 6)
6	2.120563359	172.16.31.253	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1db5, seq=2/512, ttl=64 (request in 5)
7	2.289409086	172.16.30.1	172.16.1.1	DNS	97	Standard query 0x88ce A 1.debian.pool.ntp.org.netlab.fe.up.pt
8	2.289429878	172.16.30.1	172.16.1.1	DNS	97	Standard query 0x278b AAAA 1.debian.pool.ntp.org.netlab.fe.up.pt
9	2.289576359	172.16.30.254	172.16.30.1	ICMP	125	Destination unreachable (Network unreachable)
10	2.289599346	172.16.30.254	172.16.30.1	ICMP	125	Destination unreachable (Network unreachable)
11	3.121532279	172.16.30.1	172.16.31.253	ICMP	98	Echo (ping) request id=0x1db5, seq=3/768, ttl=64 (reply in 12)
12	3.121874244	172.16.31.253	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1db5, seq=3/768, ttl=64 (request in 11)
13	3.145992939	Cisco_3a:fa:83	CDP/VTP/DTP/PAgP/U…	CDP	435	Device ID: tux-sw3 Port ID: FastEthernet0/1
14	4.009790634	Cisco_3a:fa:83	Spanning-tree-(for…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
15	4.120526457	172.16.30.1	172.16.31.253	ICMP	98	Echo (ping) request id=0x1db5, seq=4/1024, ttl=64 (reply in 16)
16	4.120737786	172.16.31.253	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1db5, seq=4/1024, ttl=64 (request in 15)
17	5.120306280	172.16.30.1	172.16.31.253	ICMP	98	Echo (ping) request id=0x1db5, seq=5/1280, ttl=64 (reply in 18)
18	5.120524826	172.16.31.253	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1db5, seq=5/1280, ttl=64 (request in 17)

Figura 14 – tux31 pin 253

4	3.472634997	172.16.30.1	193.136.28.10	DNS	97	Standard query 0xe9d7 A 2.debian.pool.ntp.org.netlab.fe.up.pt
5	3.472654491	172.16.30.1	193.136.28.10	DNS	97	Standard query 0x66a1 AAAA 2.debian.pool.ntp.org.netlab.fe.up.pt
6	3.472842285	172.16.30.254	172.16.30.1	ICMP	125	Destination unreachable (Network unreachable)
7	3.472858155	172.16.30.254	172.16.30.1	ICMP	125	Destination unreachable (Network unreachable)
8	3.963527575	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x1d63, seq=1/256, ttl=64 (reply in 9)
9	3.963874407	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1d63, seq=1/256, ttl=64 (request in 8)
10	4.004829047	Cisco_3a:fa:83	Spanning-tree-(for…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
11	4.963511121	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x1d63, seq=2/512, ttl=64 (reply in 12)
12	4.963745838	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1d63, seq=2/512, ttl=64 (request in 11)

Figura 15 – tux31 pin 254

18	13.474576271	G-ProCom_8b:e4:4d	HewlettP_5a:7d:74	ARP	60	Who has 172.16.30.254? Tell 172.16.30.1
19	13.474595128	HewlettP_5a:7d:74	G-ProCom_8b:e4:4d	ARP	42	172.16.30.254 is at 00:21:5a:5a:7d:74
20	13.478692158	172.16.30.1	172.16.1.1	DNS	81	Standard query 0xc59c A 3.debian.pool.ntp.org
21	13.478713389	172.16.30.254	172.16.30.1	ICMP	109	Destination unreachable (Network unreachable)
22	13.478716742	172.16.30.1	172.16.1.1	DNS	81	Standard query 0x71b1 AAAA 3.debian.pool.ntp.org
23	13.478723028	172.16.30.254	172.16.30.1	ICMP	109	Destination unreachable (Network unreachable)
24	14.042575246	Cisco_3a:fa:84	Spanning-tree-(for…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
25	15.198536141	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x1f58, seq=1/256, ttl=64 (reply in 26)
26	15.198807404	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1f58, seq=1/256, ttl=63 (request in 25)
27	16.052557102	Cisco_3a:fa:84	Spanning-tree-(for…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8004
28	16.198711792	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x1f58, seq=2/512, ttl=64 (reply in 29)
29	16.198876617	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1f58, seq=2/512, ttl=63 (request in 28)
30	17.198757607	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x1f58, seq=3/768, ttl=64 (reply in 31)
31	17.198901411	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1f58, seq=3/768, ttl=63 (request in 30)

Figura 16 – tux34 eth0

7	9.809735931	Cisco_3a:fa:86	CDP/VTP/DTP/PAgP/U…	CDP	435	Device ID: tux-sw3 Port ID: FastEthernet0/4
8	10.019362169	Cisco_3a:fa:86	Spanning-tree-(for…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8006
9	12.029307289	Cisco_3a:fa:86	Spanning-tree-(for…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8006
10	13.356345664	3Com_21:83:0e	Broadcast	ARP	42	Who has 172.16.31.1? Tell 172.16.31.253
11	13.356471030	HewlettP_61:30:63	3Com_21:83:0e	ARP	60	172.16.31.1 is at 00:21:5a:61:30:63
12	13.356479131	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x1f58, seq=1/256, ttl=63 (reply in 13)
13	13.356582636	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1f58, seq=1/256, ttl=64 (request in 12)
14	14.029227391	Cisco_3a:fa:86	Spanning-tree-(for…	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8006
15	14.356514052	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x1f58, seq=2/512, ttl=63 (reply in 16)

Figura 17 – tux34 eth2

Experiência 4

```
root@tux34:~# ifconfig eth2 172.16.31.253/24
root@tux34:~# ping 172.16.31.1
PING 172.16.31.1 (172.16.31.1) 56(84) bytes of data.
64 bytes from 172.16.31.1: icmp_seq=1 ttl=64 time=0.264 ms
64 bytes from 172.16.31.1: icmp_seq=2 ttl=64 time=0.141 ms
64 bytes from 172.16.31.1: icmp_seq=3 ttl=64 time=0.159 ms
64 bytes from 172.16.31.1: icmp_seq=4 ttl=64 time=0.143 ms
64 bytes from 172.16.31.1: icmp_seq=5 ttl=64 time=0.126 ms
64 bytes from 172.16.31.1: icmp_seq=6 ttl=64 time=0.123 ms
64 bytes from 172.16.31.1: icmp_seq=7 ttl=64 time=0.135 ms
64 bytes from 172.16.31.1: icmp_seq=8 ttl=64 time=0.161 ms
64 bytes from 172.16.31.1: icmp_seq=9 ttl=64 time=0.139 ms
^C
--- 172.16.31.1 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 205ms
rtt min/avg/max/mdev = 0.123/0.154/0.264/0.042 ms
root@tux34:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.30.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
172.16.31.0     0.0.0.0         255.255.255.0   U     0      0        0 eth2
```

Figura 18 – execução de ifconfig e ping e tabela de encaminhamento no tux34

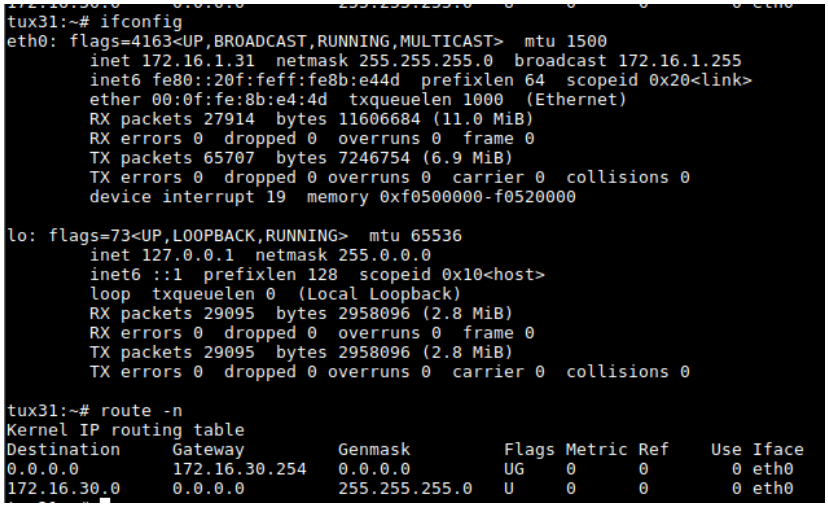


Figura 19 – configurações e tabela de encaminhamento no tux31

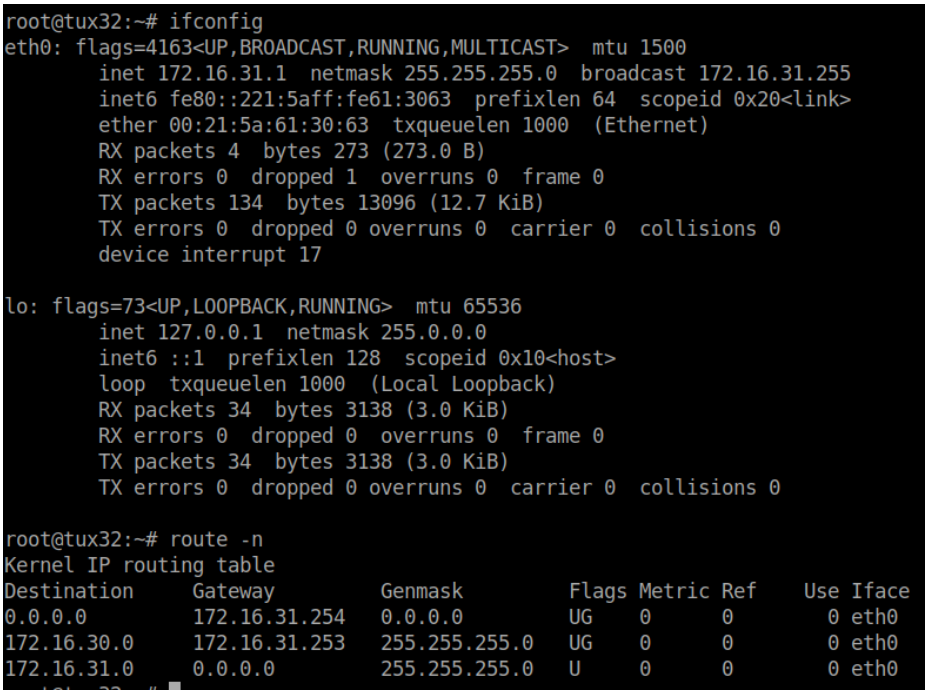


Figura 20 – configurações e tabela de encaminhamento no tux32

```
rtt min/avg/max/mdev = 0.521/0.676/0.824/0.072 ms
root@tux32:~# traceroute 172.16.30.1
traceroute to 172.16.30.1 (172.16.30.1), 30 hops max, 60 byte packets
^C
root@tux32:~# traceroute tux31
traceroute to tux31 (172.16.1.31), 30 hops max, 60 byte packets
 1  172.16.31.254 (172.16.31.254)  0.437 ms  0.446 ms  0.510 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  *^C
root@tux32:~# traceroute 172.16.30.1
traceroute to 172.16.30.1 (172.16.30.1), 30 hops max, 60 byte packets
 1  172.16.31.254 (172.16.31.254)  0.490 ms  0.527 ms  0.553 ms
 2  172.16.31.253 (172.16.31.253)  0.710 ms  0.328 ms  0.323 ms
 3  172.16.30.1 (172.16.30.1)  0.605 ms  0.590 ms  0.573 ms
root@tux32:~# route add -net 172.16.30.0/24 gw 172.16.31.253
root@tux32:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.31.254   0.0.0.0         UG    0      0        0 eth0
172.16.30.0     172.16.31.253   255.255.255.0   UG    0      0        0 eth0
172.16.31.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
root@tux32:~# traceroute 172.16.30.1
traceroute to 172.16.30.1 (172.16.30.1), 30 hops max, 60 byte packets
 1  172.16.31.253 (172.16.31.253)  0.180 ms  0.160 ms  0.143 ms
 2  172.16.30.1 (172.16.30.1)  0.435 ms  0.420 ms  0.402 ms
```

Figura 21 – execução de traceroute no tux32


```
From 172.16.31.254: icmp_seq=13 Redirect Host(New nexthop: 172.16.31.253)
64 bytes from 172.16.30.1: icmp_seq=13 ttl=63 time=0.703 ms
From 172.16.31.254: icmp_seq=14 Redirect Host(New nexthop: 172.16.31.253)
64 bytes from 172.16.30.1: icmp_seq=14 ttl=63 time=0.652 ms
From 172.16.31.254: icmp_seq=15 Redirect Host(New nexthop: 172.16.31.253)
64 bytes from 172.16.30.1: icmp_seq=15 ttl=63 time=0.713 ms
From 172.16.31.254: icmp_seq=16 Redirect Host(New nexthop: 172.16.31.253)
64 bytes from 172.16.30.1: icmp_seq=16 ttl=63 time=0.609 ms
^C
--- 172.16.30.1 ping statistics ---
16 packets transmitted, 16 received, 0% packet loss, time 367ms
rtt min/avg/max/mdev = 0.521/0.676/0.824/0.072 ms
root@tux32:~# traceroute 172.16.30.1
traceroute to 172.16.30.1 (172.16.30.1), 30 hops max, 60 byte packets
^C
root@tux32:~# traceroute tux31
traceroute to tux31 (172.16.1.31), 30 hops max, 60 byte packets
 1  172.16.31.254 (172.16.31.254)  0.437 ms  0.446 ms  0.510 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  *^C
root@tux32:~# traceroute 172.16.30.1
traceroute to 172.16.30.1 (172.16.30.1), 30 hops max, 60 byte packets
 1  172.16.31.254 (172.16.31.254)  0.490 ms  0.527 ms  0.553 ms
 2  172.16.31.253 (172.16.31.253)  0.710 ms  0.328 ms  0.323 ms
 3  172.16.30.1 (172.16.30.1)  0.605 ms  0.590 ms  0.573 ms
```


Figura 21 – execução de traceroute no tux32

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.30.254  netmask 255.255.255.0  broadcast 172.16.30.255
        inet6 fe80::221:5aff:fe5a:7d74  prefixlen 64  scopeid 0x20<link>
        ether 00:21:5a:5a:7d:74  txqueuelen 1000  (Ethernet)
        RX packets 24492  bytes 4501528 (4.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 26159  bytes 11478613 (10.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 17

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.31.253  netmask 255.255.255.0  broadcast 172.16.31.255
        inet6 fe80::2c0:dfff:fe25:260a  prefixlen 64  scopeid 0x20<link>
        ether 00:c0:df:25:26:0a  txqueuelen 1000  (Ethernet)
        RX packets 28115  bytes 11631897 (11.0 MiB)
        RX errors 0  dropped 203  overruns 0  frame 0
        TX packets 30215  bytes 4941073 (4.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 462  bytes 47014 (45.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 462  bytes 47014 (45.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figura 22 – execução do comando ifconfig

```
root@tux34:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.31.254   0.0.0.0         UG    0      0        0 eth1
172.16.30.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
172.16.31.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
```

Figura 23 – execução de traceroute no tux32

Experiência 5

1	0.000000000	172.16.30.1	172.16.1.1	DNS	71	Standard query 0x8ce6 A ftp.up.pt
2	0.000039219	172.16.30.1	172.16.1.1	DNS	71	Standard query 0x379b AAAA ftp.up.pt
3	0.001865830	172.16.1.1	172.16.30.1	DNS	536	Standard query response 0x8ce6 A ftp.up.pt CNAME mirrors.up.pt A 193.13
4	0.001896508	172.16.1.1	172.16.30.1	DNS	548	Standard query response 0x379b AAAA ftp.up.pt CNAME mirrors.up.pt AAAA
5	0.002446175	172.16.30.1	193.137.29.15	ICMP	100	Echo (ping) request id=0x64f2, seq=1/256, ttl=64 (reply in 6)
6	0.005000057	193.137.29.15	172.16.30.1	ICMP	100	Echo (ping) reply id=0x64f2, seq=1/256, ttl=57 (request in 5)
7	0.005192564	172.16.30.1	172.16.1.1	DNS	88	Standard query 0x95e2 PTR 15.29.137.193.in-addr.arpa
8	0.007886460	172.16.1.1	172.16.30.1	DNS	363	Standard query response 0x95e2 PTR 15.29.137.193.in-addr.arpa PTR mirro
9	1.004112678	172.16.30.1	193.137.29.15	ICMP	100	Echo (ping) request id=0x64f2, seq=2/512, ttl=64 (reply in 10)
10	1.005920291	193.137.29.15	172.16.30.1	ICMP	100	Echo (ping) reply id=0x64f2, seq=2/512, ttl=57 (request in 9)
11	2.005373616	172.16.30.1	193.137.29.15	ICMP	100	Echo (ping) request id=0x64f2, seq=3/768, ttl=64 (reply in 12)
12	2.007860825	193.137.29.15	172.16.30.1	ICMP	100	Echo (ping) reply id=0x64f2, seq=3/768, ttl=57 (request in 11)
13	3.006614765	172.16.30.1	193.137.29.15	ICMP	100	Echo (ping) request id=0x64f2, seq=4/1024, ttl=64 (reply in 14)
14	3.008504636	193.137.29.15	172.16.30.1	ICMP	100	Echo (ping) reply id=0x64f2, seq=4/1024, ttl=57 (request in 13)
15	4.007645399	172.16.30.1	193.137.29.15	ICMP	100	Echo (ping) request id=0x64f2, seq=5/1280, ttl=64 (reply in 16)
16	4.009911214	193.137.29.15	172.16.30.1	ICMP	100	Echo (ping) reply id=0x64f2, seq=5/1280, ttl=57 (request in 15)
17	5.008034511	HewlettP_5a:7d:74		ARP	62	Who has 172.16.30.1? Tell 172.16.30.254
18	5.008067189	G-ProCom_8b:e4:4d		ARP	44	172.16.30.1 is at 00:0f:fe:8b:e4:4d
19	5.009046270	172.16.30.1	193.137.29.15	ICMP	100	Echo (ping) request id=0x64f2, seq=6/1536, ttl=64 (reply in 20)
20	5.011112678	193.137.29.15	172.16.30.1	ICMP	100	Echo (ping) reply id=0x64f2, seq=6/1536, ttl=57 (request in 19)
21	6.009715321	172.16.30.1	193.137.29.15	ICMP	100	Echo (ping) request id=0x64f2, seq=7/1792, ttl=64 (reply in 22)
22	6.011581047	193.137.29.15	172.16.30.1	ICMP	100	Echo (ping) reply id=0x64f2, seq=7/1792, ttl=57 (request in 21)
23	7.011025357	172.16.30.1	193.137.29.15	ICMP	100	Echo (ping) request id=0x64f2, seq=8/2048, ttl=64 (reply in 24)
24	7.013433335	193.137.29.15	172.16.30.1	ICMP	100	Echo (ping) reply id=0x64f2, seq=8/2048, ttl=57 (request in 23)

Figura 24 – ping a partir do tux31 ao router da sala com DNS ativado

1	0.000000000	127.0.0.1	127.0.0.1	DNS	71 Standard query 0x581f A ftp.up.pt
2	0.000021125	127.0.0.1	127.0.0.1	ICMP	99 Destination unreachable (Port unreachable)
3	0.000077825	127.0.0.1	127.0.0.1	DNS	71 Standard query 0x7360 AAAA ftp.up.pt
4	0.000089359	127.0.0.1	127.0.0.1	ICMP	99 Destination unreachable (Port unreachable)
5	0.000138455	127.0.0.1	127.0.0.1	DNS	71 Standard query 0x581f A ftp.up.pt
6	0.000148812	127.0.0.1	127.0.0.1	ICMP	99 Destination unreachable (Port unreachable)
7	0.000175144	127.0.0.1	127.0.0.1	DNS	71 Standard query 0x7360 AAAA ftp.up.pt
8	0.000185275	127.0.0.1	127.0.0.1	ICMP	99 Destination unreachable (Port unreachable)
9	0.000250933	127.0.0.1	127.0.0.1	DNS	71 Standard query 0x4c5c A ftp.up.pt
10	0.000261290	127.0.0.1	127.0.0.1	ICMP	99 Destination unreachable (Port unreachable)
11	0.000287507	127.0.0.1	127.0.0.1	DNS	71 Standard query 0x026a AAAA ftp.up.pt
12	0.000297543	127.0.0.1	127.0.0.1	ICMP	99 Destination unreachable (Port unreachable)
13	0.000337099	127.0.0.1	127.0.0.1	DNS	71 Standard query 0x4c5c A ftp.up.pt
14	0.000347280	127.0.0.1	127.0.0.1	ICMP	99 Destination unreachable (Port unreachable)
15	0.000372700	127.0.0.1	127.0.0.1	DNS	71 Standard query 0x026a AAAA ftp.up.pt
16	0.000382555	127.0.0.1	127.0.0.1	ICMP	99 Destination unreachable (Port unreachable)

Figura 25 – ping a partir do tux31 ao router da sala sem DNS ativado

Experiência 6

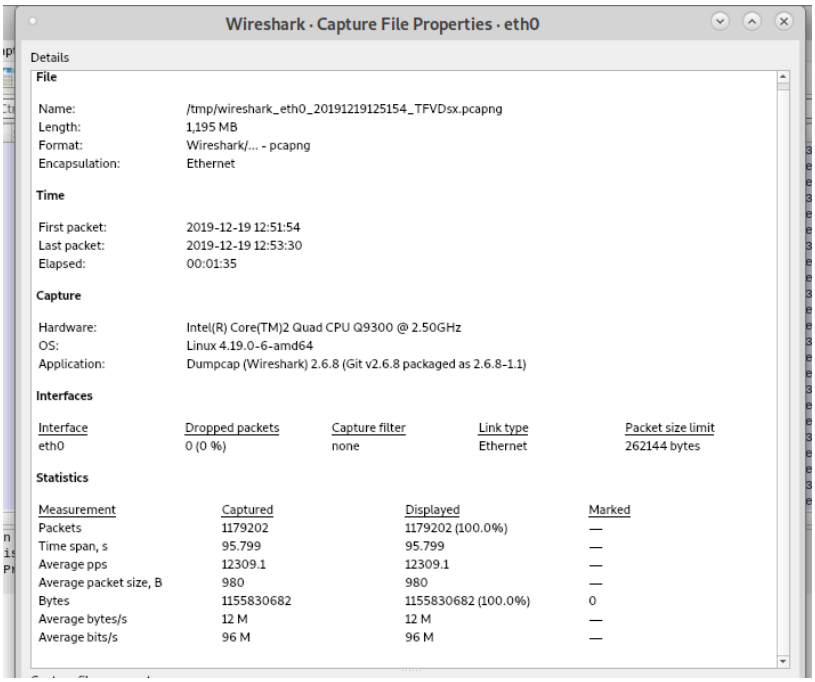


Figura 26 – estatísticas no tux31

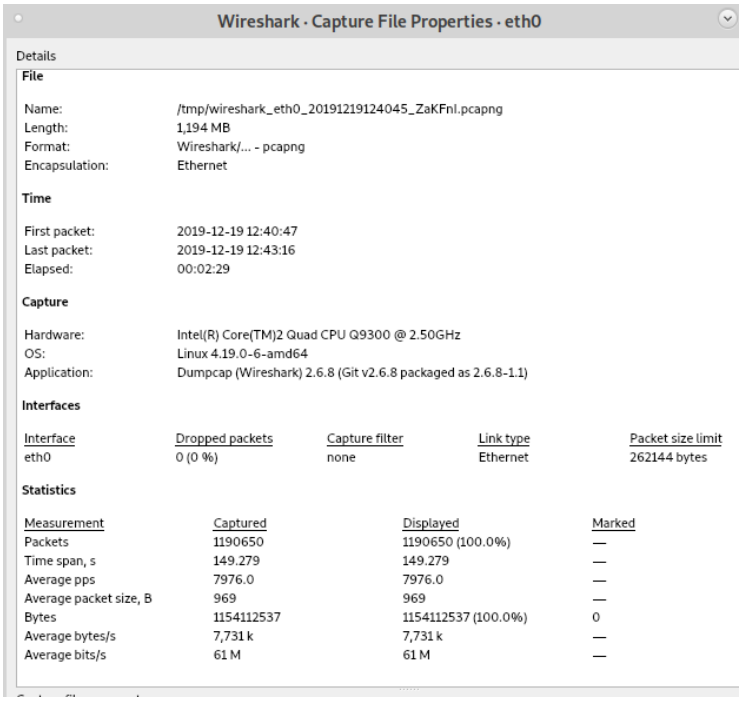


Figura 27 – estatísticas no tux31 quando foi começado um download no tux32 a meio do primeiro