Redes de Computadores

2º Trabalho Laboratorial – Redes de computadores

FEUP

Licenciatura em Engenharia Informática e Computação

Joana Rita Batista Marques - up202103346
João Tomás Matos Fernandes Garcia Padrão – 202108766

Porto, 7 de Dezembro de 2023

Sumário

No âmbito da unidade curricular de Redes de Computadores, foi desenvolvido o segundo trabalho laboratorial com o objetivo de criar uma aplicação de "download" usando o protocolo FTP e a configuração de uma rede de computadores.

O presente relatório visa fazer uma exposição e análise da implementação desenvolvida ao longo do trabalho, assim como apresentar as principais conclusões obtidas ao longo do mesmo.

Introdução

Como mencionado anteriormente, o objetivo principal deste segundo trabalho laboratorial é o desenvolvimento de uma aplicação de "download" usando o protocolo FTP e a configuração de uma rede de computadores visando transferir um ficheiro da internet utilizando a mesma. Em seguida, ao longo das várias secções deste relatório iremos descrever a implementação do trabalho e lógica subjacente ao mesmo. De uma forma breve, a estrutura dos tópicos presentes neste relatório é a seguinte:

- Parte 1 – Desenvolvimento de um aplicativo de download

  Arquitetura da aplicação
  Resultados

- Parte 2 – Configuração e Estudo de uma Rede

  Experiência 1 – Configurar uma rede IP
  Experiência 2 - Implementar duas *bridges* num *switch*
  Experiência 3 - Configurar um *router* em Linux
  Experiência 4 – Configurar um *router* comercial e implementar NAT
  Experiência 5 – DNS
  Experiência 6 – Ligações TCP

- Conclusões - Síntese da informação apresentada nas secções anteriores e reflexão sobre os objetivos de aprendizagem alcançados.

Parte 1 – Desenvolvimento de um aplicativo de download

Arquitetura da aplicação

A primeira parte deste segundo trabalho laboratorial tinha como objetivo o desenvolvimento de uma aplicação que realiza o *download* de um ficheiro através do protocolo FTP, na linguagem de programação C. Como referido no guião laboratorial, foi consultada a RFC *(Request for Comments)* "RFC959-FTP", informação referente ao protocolo FTP.

Em primeiro lugar, a aplicação processa o URL fornecido como argumento. Para esse processamento e visando obter os dados para preencher as informações necessárias é chamada a função ***parseURL.*** Esta função além de realizar o tratamento de erros no URL (por exemplo: um URL que não seja FTP), vai obter o *host,* nome do servidor, o *path,* caminho do ficheiro pretendido, o *user* e a *password*, para autenticar-se no servidor podendo estes serem *default* caso sejam omitidos, o *filename,* nome do ficheiro pretendido para transferir, e finalmente o *ip,* que vai ser obtido pela função ***getIP*** (código fornecido). A porta usada é a 21.

Uma das funções mais importantes é a ***readResponse*** que é a responsável por obter as respostas enviadas pelo servidor através de uma máquina de estados. Retorna então o código da resposta que servirá como validação para outras funções.

De seguida, é criado então um *socketA* chamando a função ***createSocket*** e vamos ver se o servidor está pronto para começar a autenticação usando a função ***readResponse.*** Após a verificação é garantida a autenticação no servidor com a função ***authentication.*** Após isso, chamando a função ***changePassiveMode*** vamos enviar o comando "pasv\n" para que o servidor entre em modo passivo e deste modo seja o cliente a iniciar a conexão de dados contornando possíveis problemas de firewall. Além disto, obteve-se também o *ip* e a *port* para esta conexão criando então um segundo *socket, socketB,* com estes dados.

Na última fase, é chamada a função ***requestPath*** que vai enviar para o servidor o comando "retr <path>\n" com o objetivo de especificar o ficheiro a ser transferir e espera pela resposta do servidor. Finalmente, é então transferido usando a função ***gefFile.***

Por último, após a transferência concluída, são fechadas as duas conexões *socketA*, transferência de comandos, e *socketB,* transferência de dados, função ***endConnection.***

Resultados

A aplicação foi testada em diversas circunstâncias: transferências de ficheiros com vários tamanhos, modo anónimo, modo não anónimo, URL's errados em diversos campos, ficheiros não existentes e credenciais erradas. A aplicação termina em caso de erro e com respostas impressas na consola para maior controlo por parte do utilizador. Assim, verificamos que a aplicação se mostra robusta e eficiente.

Podemos ver exemplos do funcionamento da aplicação nas imagens 1 e 2 em anexo. O código da aplicação também se encontra em anexo.

Parte 2 – Configuração e análise de redes

Experiência 1 – Configurar uma rede IP

- What are the commands required to configure this experience?

1. Manualmente fazer a ligação do E0 do tux33 e do tux34 (experiência feita na bancada 3) às entradas do "switch" 16 e 18 respetivamente.
2. Reniciar o serviço de rede nos três tux's existentes usando no terminal de cada um o comando: "systemctl restart networking".
3. Configurar os IP's do tux33 e tux34 usando respetivamente em cada computador os seguintes comandos:
    - "ifconfig eth0 up" (Ativar a interface eth0, em ambos os tux)
    - "ifconfig eth0 172.16.30.1/24" (No tux33)
    - "ifconfig eth0 172.16.30.254/24" (No tux34)
4. Usando o comando "ipconfig" em cada computador conseguimos saber os endereços IP e MAC.
5. De seguida, para verificar a conectividade entre estes dois computadores vamos, por exemplo, ao tux33 e executamos o comando "ping 172.16.30.254" para dar "ping" ao tux34 e verificar se existe conectividade.
6. Usamos o comando "route -n" para ver a tabela de rotas, onde verificamos que só temos uma. Verificamos que o gateway é 0 e faz sentido porque está ligada diretamente à rede.

7. Usamos o comando “arp -a” no tux33 para ver a tabela Address Resolution Protocol (ARP) onde tem um único valor com o endereço IP e o MAC do tux34.
8. Apagamos, no tux33, essa entrada da tabela arp executando o comando “arp -d 172.16.30.254/24”, que é o endereço IP do tux34.
9. Finalmente, voltamos a usar o comando “ping 172.16.30.254” para testar a conectividade e enquanto usamos o Wireshark para capturar esses “packets”.

- What are the ARP packets and what are they used for?

O ARP (Address Resolution Protocol) é um protocolo de comunicação usado para associar um endereço IP a um endereço MAC.

- What are the MAC and IP addresses of ARP packets and why?

No tux33 quando executamos o comando “arp-a” para ver a tabela ARP vemos um único valor com o endereço IP: 172.16.30.254 associado ao endereço MAC: 00:21:5a:5a:7d:74. Esta informação está disponível na imagem 3 em anexo.

- What packets does the ping command generate?

Os pacotes que são gerados pelo “ping 172.16.30.254” são em primeiro lugar pacotes ARP para obter o endereço MAC associado ao IP e depois pacotes ICMP (Internet Control Message Protocol).

- What are the MAC and IP addresses of the ping packets?

Os pacotes que são gerados pelo tux33 quando faz “ping” ao tux34 e os seus endereços MAC e IP estão disponíveis em anexo.

- How to determine if a receiving Ethernet frame is ARP, IP, ICMP?

Para verificar se uma trama é ARP,IP ou ICMP vemos o Ethernet Header e conseguimos assim determinar de que tipo é. Se tiver o valor 0x0800, significa que a trama é do tipo IP. Por outro lado, se tiver o valor 0x0806, então a trama é do tipo ARP. Finalmente, caso a trama seja do tipo IP, podemos analisar o seu IP header. Se este header tiver o valor 1, então o tipo de protocolo é ICMP. Esta informação está disponível nas imagens 4 e 5 em anexo.

- How to determine the length of a receiving frame?

Para determinar o comprimento de uma trama basta verificar o software Wireshark. Esta informação está disponível na imagem 6 em anexo.

- What is the loopback interface and why is it important?

A interface loopback é uma interface virtual de rede no computador usada para estabelecer conexões de rede consigo mesmo, o que permite poder realizar testes de diagnóstico.

Experiência 2 – Implementar duas bridges no Switch

Esta experiência tinha como objectivo a implementação de duas bridges no Switch para assim criar duas LANs virtuais. Na primeira VLAN foram conectados os tux31 e tux34, e na segunda VLAN o tux32. Assim, foi possível a comunicação entre os tux31 e tux34 dado que estão na mesma subrede, mas não com o tux22, que se encontrava numa subrede diferente.

- How to configure bridgeY0?

A configuração da bridge30 tinha como objetivo estabelecer a ligação entre o tux33 e o tux34 formando assim uma "subrede". Visando este objetivo, primeiramente criamos a bridge30. De seguida, removemos das portas do *switch* que faziam ligação ao tux33 e ao tux34 a bridge *default*. Por último, adicionamos a essas mesmas portas a bridge que foi criada, a bridge30. Segue-se , respetivamente, os comandos que usamos para esta configuração:

1. /interface bridge add name=bridge30 #Criar a bridge30
2. /interface bridge port remove [find interface =ether16] #Remover a bridge *default* da porta 16
3. /interface bridge port remove [find interface =ether18] #Remover a bridge *default* da porta 18
4. /interface bridge port add bridge=bridge30 interface=ether16 #Adicionar a bridge30 à porta 16
5. /interface bridge port add bridge=bridge30 interface=ether18 #Adicionar a bridge30 à porta 18

- How many broadcast domains are there? How can you conclude it from the logs?

Foi possível concluir através da análise dos logs obtidos após a execução do comando 'ping-b' desde o tux33 e do tux32 a existência de 2 domínios de broadcast. Esta informação está disponível nas imagens 7 e 8 em anexo.

Experiência 3 – Configuração de um Router em Linux

Para a experiência 3, configuramos o tux34 para servir como router entre as duas VLANs criadas na experiência anterior, permitindo desta forma a comunicação entre máquinas que estejam em VLAN's distintas, como é o caso do tux32 e tux33. Os "logs" que serviram de base para as respostas seguintes estão disponíveis nas imagens 9,10,11,12 e 13 em anexo.

- What are the commands required to configure this experience?

Foram executados os seguintes comandos para configuração de IPs:

No tux33: ***ifconfig eth0 172.16.20.1/24*;**
No tux34: ***ifconfig eth0 172.16.20.254/24*** e ***ifconfig eth1 172.16.21.253/24*;**
No tux32: ***ifconfig eth0 172.16.21.1/24*;**

Para eliminar as portas ligadas por defeito ao tux34, e adicionar á posteriori uma nova porta, foram executados os seguintes comandos:

/interface bridge port remove [find interface=ether4]
/interface bridge port add bridge=bridge21 interface=ether4

Para ativar IP forwarding e ativar ICMP, executamos no tux34 os seguintes comandos:

echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

- What routes are there in the tuxes? What are their meaning?

Algumas rotas são criadas de forma automática, conectando as máquinas às suas respectivas VLANs, nomeadamente, o tux33 possui uma rota para a VLAN30 e o tux32 por sua vez tem uma rota para a VLAN31. Já o tux34 possui uma rota para ambas VLANs. Acresce ainda que todos os tuxs tem uma rota para o domínio a que pertencem, onde o gateway é 0.0.0.0.

De forma a estabelecer a conexão entre o tux33 e o tux32, foi necessário adicionar uma rota ao tux33. Para esse efeito, executamos no tux33 o comando ***route add -net 172.16.Y1.0/24 gw 172.16.Y0.254***, sendo que o primeiro endereço especifica a gama de endereços no qual será adicionada a rota, e o segundo endereço serve para especificar o IP que irá reencaminhar o pacote, que na experiência 3 corresponde ao IP do tux34 na interface eth0. O mesmo procedimento foi feito para o tux32, onde executamos o comando ***route add -net 172.16.Y0.0/24 gw 172.16.Y1.253*** sendo que 172.16.Y1.253 representa o IP do tux34 na interface eth1.

Estas rotas servem para que cada computador saiba o "caminho" para onde deverá enviar os pacotes para que estes cheguem ao seu destino correctamente.

- What information does an entry of the forwarding table contain?

Uma entrada na tabela de encaminhamento contém informações como o destino da rota, IP de gateway, máscara de sub-rede, flags (informações da rota), métrica, referências, uso (contador do número de vezes que esta rota foi consultada) e a interface de rede associada (eth0 ou eth1.). Essas informações ajudam a determinar a maneira como os pacotes de dados são encaminhados.

- What ARP messages, and associated MAC addresses, are observed and why?

As mensagens ARP são enviadas quando não é conhecido á priori o endereço MAC de um certo IP. Quando é enviado um ping a partir do tux 3 para a interface do tux 4, verifica-se que a interface eth0 do tux3 envia uma mensagem ARP para saber o endereço MAC da interface eth0 do tux4 e que o tux4 envia um pedido para saber o endereço MAC do tux3. O mesmo verifica-se quando é enviado um ping para a interface eth1 do tux4 e tux2.

- What ICMP packets are observed and why?

Os pacotes ICMP observados são tipo *request* e *reply,* uma vez que tendo todas as rotas adicionadas, todos os tuxs conseguem estabelecer comunicação uns com os outros. Caso não tivessem sido adicionadas as rotas, os pacotes ICMP enviados seriam do tipo *Host Unreachable*.

- What are the IP and MAC addresses associated to ICMP packets and why?

Observando os logs da ligação entre o tux32 e o tux33, é possível verificar que os pacotes ICMP request continham como endereço IP de destino o endereço MAC do tux34. Como expectável, o mesmo acontece nos pacotes ICMP reply, uma vez que o tux34 efectua o redirecionamento da comunicação entre as duas VLANs criadas.

Experiência 4 – Configurar um Router Comercial e Implementar NAT

O objetivo da experiência 4 é a correcta configuração de um router comercial com NAT implementado de modo a permitir que as redes criadas pudessem se conectar á rede do laboratório (172.16.1.0/24) e á VLAN1. Par esse efeito, o router foi adicionado à bridge31. Os "logs" capturados nesta experiência estão disponíveis nas imagens 14,15,16,17 e 18 em anexo.

- How to configure a static route in a commercial router?

Para configurar uma static route num router comercial, efectuamos a ligação do cabo de série S0 do tux34 à entrada de configuração do router e em seguida procedemos com a sua configuração no GTKTerm. Para configurar as rotas, executamos o comando ***ip route*** no GTKTerm.

- What are the paths followed by the packets in the experiments carried out and why?

Caso exista uma rota específica para onde se pretende enviar os pacotes, estes seguem essa mesma rota. Caso contrário, os pacotes são direcionados ao router através da rota default.

No decorrer desta experiência foram desativados os redirecionamentos do tux32 para o tux34 e em seguida foi definido o router Rc como route default para o tux32 e tux34. Quando enviamos um ping para o tux33 a partir do tux32 verificamos que os pacotes foram encaminhados para a sua rota default, ou seja, o router Rc. Ao remover a rota até 172.16.20.0/24 via tux34 no tux32, este deixa de ter uma rota para o tux33 pelo que, encaminha os pacotes através do router Rc (rota default). Uma vez que o router RC tem uma rota directa ao tux34, os pacotes do tux32 são encaminhados primeiramente para o tux34 e depois para o tux33. No que toca aos pacotes enviados através do tux33, estes irão pela gateway da interface eth0 do tux4.

- How to configure NAT in a commercial router?

  Para configurar o Router com NAT, deve-se executar no terminal do router o seguinte comando: `/ip firewall nat enable 0`

- What does NAT do?

O Network Address Translation (NAT) traduz endereços IP privados para um único IP público. Isto permite que redes privadas, como aquelas que foram criadas no decorrer deste trabalho, se possam conectar com a internet ou rede pública. Desta forma, um pacote enviado para uma rede publica tem o endereço publico como origem e a resposta é enviada para esse endereço que será depois traduzido para o endereço local que efectuou o envio do pacote.

Experiência 5 – DNS

A experiência 5 tem como objetivo conectar as máquinas da rede configurada neste trabalho a um servidor DNS, que traduz os hostnames para endereços IP, e assim permitir que seja possível aceder a um website através do seu nome de domínio.

- How to configure the DNS service in a host?

Para configurar o DNS, editamos o ficheiro /etc/resolv.conf adicionando a linha ‘nameserver 172.16.1.1’ em todos os tuxs. Para confirmar a correcta configuração de DNS exacutamos ping www.google.com. Em seguida foi possível aceder á internet a partir do tux.

- What packets are exchanged by DNS and what information is transported

A partir do Host é enviado um pacote contendo o hostaname pretendido para o Server, e o servidor responde com um pacote que contém o respectivo endereço IP do hostname.

Experiência 6 – TCP connections

Na experiência 6 utilizamos a aplicação desenvolvida para poder observar o comportamento do protocolo TCP.

- How many TCP connections are opened by your FTP application?

Através da aplicação desenvolvida foram abertas duas conexões TCP, uma quando é estabelecido contacto com o servidor e pelo qual se envia e recebe comandos, e outra efectuar a transferência do ficheiro.

- In what connection is transported the FTP control information?

O controlo de informação é transportado pela primeira conexão TCP, ou seja, na que efectua o envio e receção de comandos do servidor.

- What are the phases of a TCP connection?

As fases de uma conexão TCP são as seguintes: inicialização (estabelecimento da conexão), handshake (cumprimentos: SYN-ACK e ACK), transferência de dados e terminação da conexão.

- How does the ARQ TCP mechanism work? What are the relevant TCP fields? What relevant information can be observed in the logs?

O mecanismo ARQ (Automatic Repeat reQuest) é um método para o controlo de erros na retransmissão de dados. Para isso, este mecanismo usa os campos ACK (acknowledgment) e SEQ (sequence number) nos cabeçalhos TCP. Quando um dispositivo recebe dados, ele envia um acknowledgment de volta para confirmar a recepção dos mesmos. Se o remetente não recebe um acknowledgment dentro de um determinado tempo (timeouts), ele retransmite os dados até ser recebido um acknowledgment. Esse processo continua até que todos os dados sejam entregues correctamente.

- How does the TCP congestion control mechanism work? What are the relevant fields. How did the throughput of the data connection evolve along the time? Is it according to the TCP congestion control mechanism?

O mecanismo de controle de congestionamento TCP regula a transmissão com base nos ACKs recebidos. A "Congestion Window" ajusta dinamicamente o volume de dados enviado, ou seja, se o nível de congestionamento da rede aumenta então a “Congestion Window diminui e vice-versa. O *throughput* evolui conforme a rede: durante um download único, há aumento rápido, enquanto com múltiplos downloads, o *throughput* inicialmente diminui devido ao congestionamento, estabilizando em um nível mais baixo. Essas mudanças seguem o esperado do controle de congestionamento TCP, que se adapta às condições da rede para otimizar a eficiência da transmissão. Podemos verificar este feito nas imagens 19 e 20 em anexo.

- Is the throughput of a TCP data connections disturbed by the appearance of a second TCP connection? How?

Sim, o *throughput* de uma conexão de dados TCP é perturbado pelo surgimento de uma segunda conexão TCP. Assim, quando há múltiplas conexões TCP concorrendo pela largura de banda, a capacidade de transmissão é dividida entre essas conexões. Isto resulta então numa redução na taxa de transmissão de pacotes de cada conexão individual, já que a largura de banda disponível é distribuída pelas conexões existentes. Podemos verificar este feito na imagem 19 em anexo.

Conclusões

Após a realização deste projeto, entendemos que o mesmo teve um impacto positivo na compreensão e aprofundamento das nossas competências acerca dos protocolos envolvidos na transferência de dados através de redes de computadores.

ANEXOS

```
root@LAPTOP-23AL74VG:/mnt/c/Users/joaot/OneDrive/Ambiente de Trabalho/rcom_proj2# ./download ftp://netlab1.fe.up.pt/pub.txt
WARNING: Anonymous MODE
Host name: netlab1.fe.up.pt
Path: pub.txt
User: anonymous
Password: anonymous@
File name: pub.txt
IP Address: 192.168.109.136
Buffer: 220Welcome to netlab-FTP server
Code: 220
Buffer: 331Please specify the password.
Code: 331
Buffer: 230Login successful.
Code: 230
Buffer: 227Entering Passive Mode (192,168,109,136,183,206).
Code: 227
Buffer: 150Opening BINARY mode data connection for pub.txt (672 bytes).
Code: 150
Buffer: 226Transfer complete.
Code: 226
Buffer: 221Goodbye.
Code: 221
```

Imagem 1 – Download de um ficheiro em modo anónimo

```
root@LAPTOP-23AL74VG:/mnt/c/Users/joaot/OneDrive/Ambiente de Trabalho/rcom_proj2# ./download ftp://rcom:rcom@netlab1.fe.up.pt/pipe.txt
Host name: netlab1.fe.up.pt
Path: pipe.txt
User: rcom
Password: rcom
File name: pipe.txt
IP Address: 192.168.109.136
Buffer: 220Welcome to netlab-FTP server
Code: 220
Buffer: 331Please specify the password.
Code: 331
Buffer: 230Login successful.
Code: 230
Buffer: 227Entering Passive Mode (192,168,109,136,186,79).
Code: 227
Buffer: 150Opening BINARY mode data connection for pipe.txt (1863 bytes).
Code: 150
Buffer: 226Transfer complete.
Code: 226
Buffer: 221Goodbye.
Code: 221
```

Imagem 2 – Download de um ficheiro em modo não anónimo

```
        TX packets 76  bytes 7080 (6.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0

root@tux33:~# 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=0.29
64 bytes from 172.16.30.254: icmp_seq=2 ttl=64 time=0.13
64 bytes from 172.16.30.254: icmp_seq=3 ttl=64 time=0.12
64 bytes from 172.16.30.254: icmp_seq=4 ttl=64 time=0.12
64 bytes from 172.16.30.254: icmp_seq=5 ttl=64 time=0.12
^C
--- 172.16.30.254 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time
rtt min/avg/max/mdev = 0.122/0.159/0.291/0.067 ms
root@tux33:~# arp -a
? (172.16.30.254) at 00:21:5a:5a:7d:74 [ether] on eth0
root@tux33:~#
```

Imagem 3 – Tabela ARP

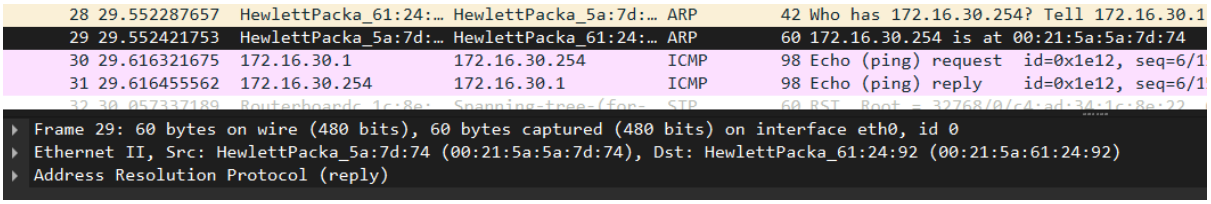
What are the MAC and IP addresses of the ping packets?

Pacote ARP de request

```
    28 29.552287657  HewlettPacka_61:24:… HewlettPacka_5a:7d:… ARP      42 Who has 172.16.30.254? Tell 172.16.30.1
    29 29.552421753  HewlettPacka_5a:7d:… HewlettPacka_61:24:… ARP      60 172.16.30.254 is at 00:21:5a:5a:7d:74
    30 29.616321675  172.16.30.1          172.16.30.254        ICMP     98 Echo (ping) request  id=0x1e12, seq=6/15
    31 29.616455562  172.16.30.254        172.16.30.1          ICMP     98 Echo (ping) reply    id=0x1e12, seq=6/15
    32 30.057337189  Routerboardc_1c:8e:… Spanning-tree-(for-… STP      60 RST. Root = 32768/0/c4:ad:34:1c:8e:22
▸ Frame 28: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface eth0, id 0
▸ Ethernet II, Src: HewlettPacka_61:24:92 (00:21:5a:61:24:92), Dst: HewlettPacka_5a:7d:74 (00:21:5a:5a:7d:74)
▸ Address Resolution Protocol (request)
```

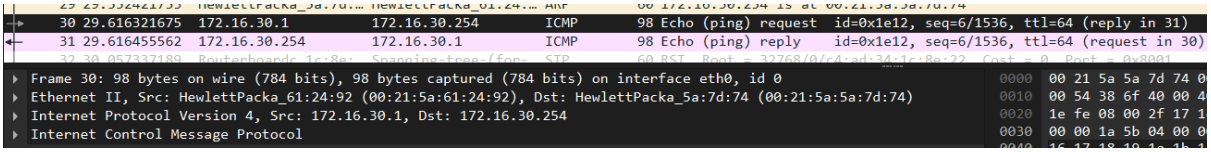
Endereço MAC da origem do pacote: 00:21:5a:61:24:92
Endereço IP da origem do pacote: 172.16.30.1
Endereço MAC do destino do pacote: 00:21:5a:5a:7d:74
Endereço IP do destino do pacote: 172.16.30.254

Pacote ARP de reply:



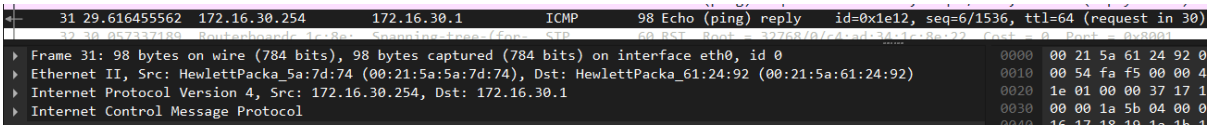
Endereço MAC da origem do pacote: 00:21:5a:5a:7d:74
Endereço IP da origem do pacote: 172.16.30.254
Endereço MAC do destino do pacote: 00:21:5a:61:24:92
Endereço IP do destino do pacote: 172.16.30.1

Pacote ICMP de request:



Endereço MAC da origem do pacote: 00:21:5a:61:24:92
Endereço IP da origem do pacote: 172.16.30.1
Endereço MAC do destino do pacote: 00:21:5a:5a:7d:74
Endereço IP do destino do pacote: 172.16.30.254

Pacote ICMP de reply:



Endereço MAC da origem do pacote: 00:21:5a:5a:7d:74
Endereço IP da origem do pacote: 172.16.30.254
Endereço MAC do destino do pacote: 00:21:5a:61:24:92
Endereço IP do destino do pacote: 172.16.30.1

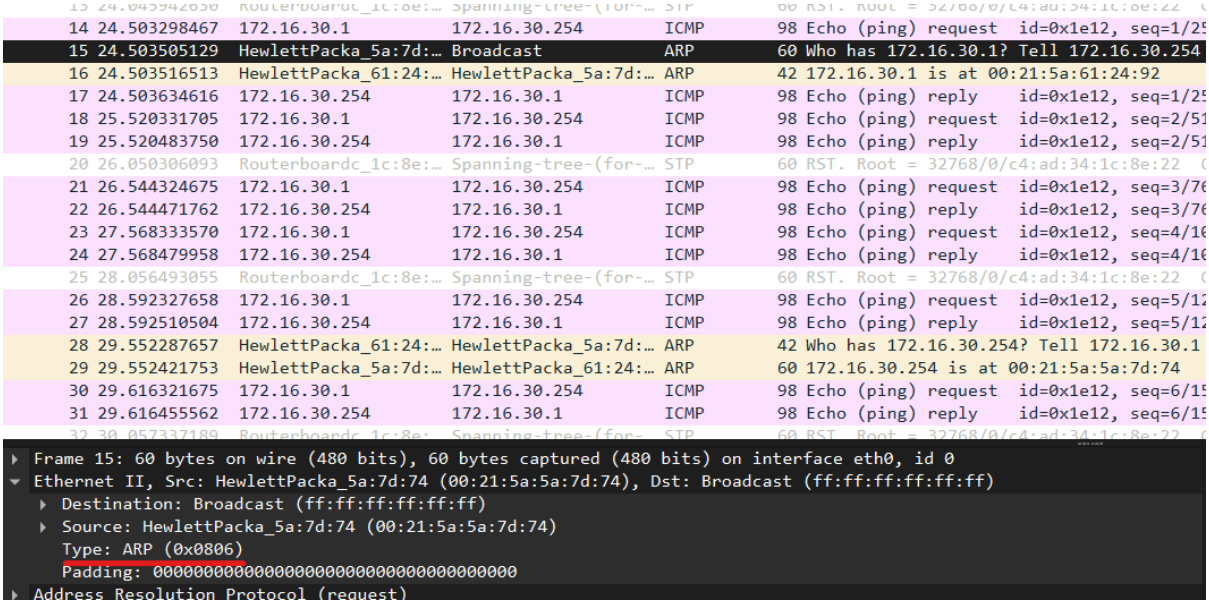


Imagem 4 – ARP

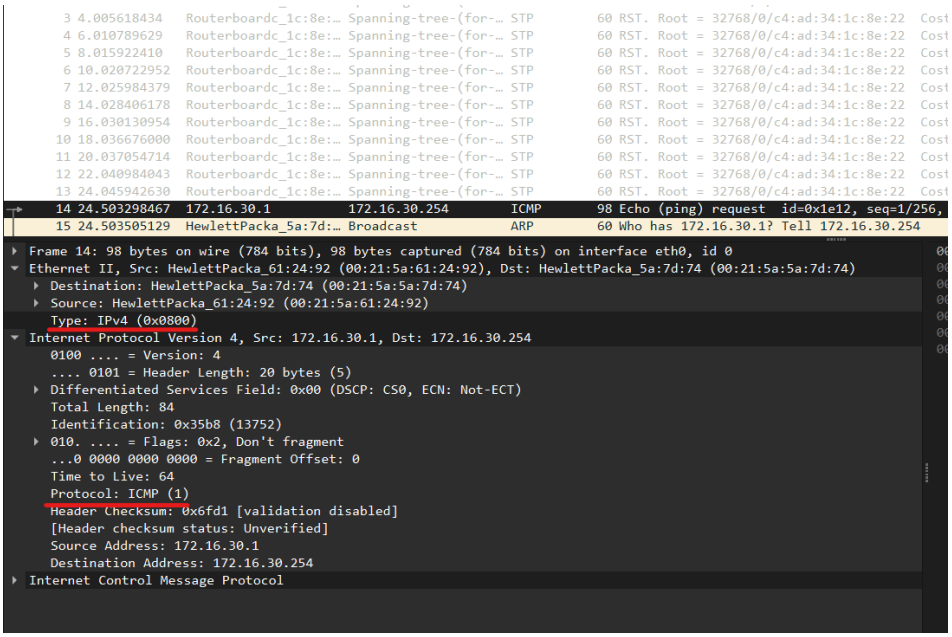


Imagem 5 – IP e ICMP

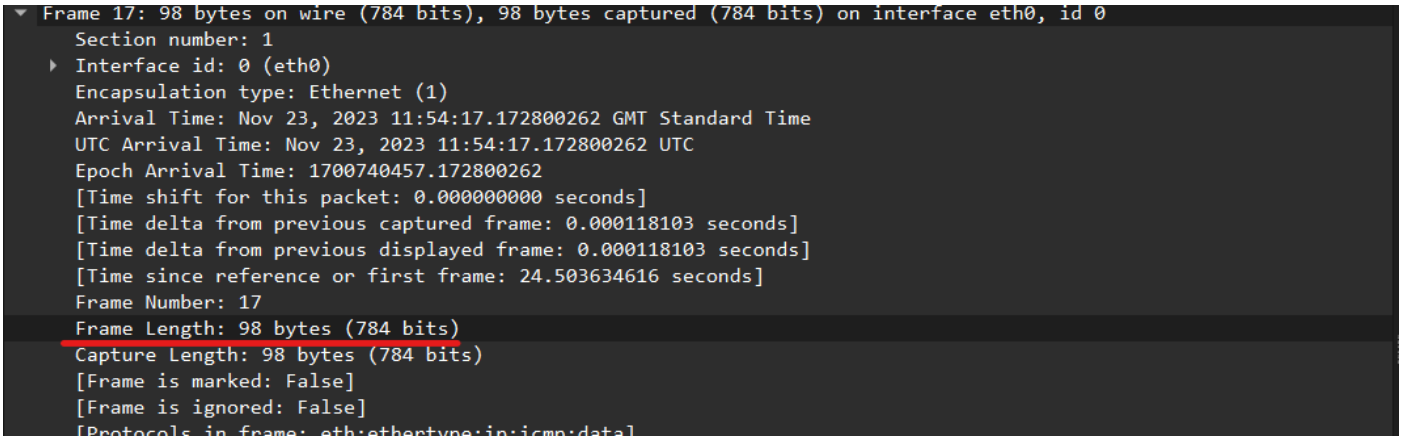


Imagem 6 – Comprimento de uma trama

25	41.998019076	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:26 Cost = 0 Port = 0x8001
26	43.025996052	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x2748, seq=1/256, ttl=64 (no response found!)
27	44.001565675	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:26 Cost = 0 Port = 0x8001
28	44.041306346	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x2748, seq=2/512, ttl=64 (no response found!)
29	45.065300553	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x2748, seq=3/768, ttl=64 (no response found!)
30	45.997270283	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:26 Cost = 0 Port = 0x8001
31	46.089304328	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x2748, seq=4/1024, ttl=64 (no response found!)
32	47.113300910	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x2748, seq=5/1280, ttl=64 (no response found!)
33	47.997319222	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:26 Cost = 0 Port = 0x8001
34	48.137300844	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x2748, seq=6/1536, ttl=64 (no response found!)
35	49.161304549	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x2748, seq=7/1792, ttl=64 (no response found!)
36	50.001935840	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:26 Cost = 0 Port = 0x8001
37	50.185298546	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x2748, seq=8/2048, ttl=64 (no response found!)
38	51.209300086	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x2748, seq=9/2304, ttl=64 (no response found!)
39	52.000412121	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:26 Cost = 0 Port = 0x8001
40	52.233298973	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x2748, seq=10/2560, ttl=64 (no response found!)
41	53.257306379	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x2748, seq=11/2816, ttl=64 (no response found!)
42	53.997458077	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:26 Cost = 0 Port = 0x8001
43	54.281304986	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x2748, seq=12/3072, ttl=64 (no response found!)
44	55.305292838	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x2748, seq=13/3328, ttl=64 (no response found!)
45	56.001518280	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:26 Cost = 0 Port = 0x8001
46	56.329300384	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x2748, seq=14/3584, ttl=64 (no response found!)
47	57.353299270	172.16.31.1	172.16.31.255	ICMP	98 Echo (ping) request id=0x2748, seq=15/3840, ttl=64 (no response found!)

Imagem 7 – Ping broadcast a partir de tux32

20	8.686020116	HewlettPacka_5a:7d:…	HewlettPacka_61:24:…	ARP	60 Who has 172.16.30.1? Tell 172.16.30.254
21	8.686042255	HewlettPacka_61:24:…	HewlettPacka_5a:7d:…	ARP	42 172.16.30.1 is at 00:21:5a:61:24:92
22	9.062159757	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
23	9.645889819	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x2833, seq=7/1792, ttl=64 (no response found!)
24	9.646095293	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x2833, seq=7/1792, ttl=64
25	10.669895083	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x2833, seq=8/2048, ttl=64 (no response found!)
26	10.670067312	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x2833, seq=8/2048, ttl=64
27	11.067278146	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
28	11.693894759	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x2833, seq=9/2304, ttl=64 (no response found!)
29	11.694063915	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x2833, seq=9/2304, ttl=64
30	12.717892549	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x2833, seq=10/2560, ttl=64 (no response found!)
31	12.718062055	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x2833, seq=10/2560, ttl=64
32	13.066788172	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
33	13.741894321	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x2833, seq=11/2816, ttl=64 (no response found!)
34	13.742063268	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x2833, seq=11/2816, ttl=64
35	14.765893159	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x2833, seq=12/3072, ttl=64 (no response found!)
36	14.766084036	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x2833, seq=12/3072, ttl=64
37	15.062029049	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
38	15.789897864	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x2833, seq=13/3328, ttl=64 (no response found!)
39	15.790063738	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x2833, seq=13/3328, ttl=64
40	16.813891603	172.16.30.1	172.16.30.255	ICMP	98 Echo (ping) request id=0x2833, seq=14/3584, ttl=64 (no response found!)
41	16.814059503	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x2833, seq=14/3584, ttl=64

Imagem 8 – Ping broadcast a partir de tux33

85	146.906027021	HewlettPacka_61:24:…	Broadcast	ARP	60 Who has 172.16.30.254? Tell 172.16.30.1
86	146.906053979	HewlettPacka_5a:7d:…	HewlettPacka_61:24:…	ARP	42 172.16.30.254 is at 00:21:5a:5a:7d:74
87	146.906164678	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x29ce, seq=1/256, ttl=64 (reply in 88)
88	146.906420926	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x29ce, seq=1/256, ttl=63 (request in 87)
89	147.913031962	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x29ce, seq=2/512, ttl=64 (reply in 90)
90	147.913170317	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x29ce, seq=2/512, ttl=63 (request in 89)
91	148.164707611	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8002
92	148.937058481	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x29ce, seq=3/768, ttl=64 (reply in 93)
93	148.937227427	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x29ce, seq=3/768, ttl=63 (request in 92)
94	149.961109584	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x29ce, seq=4/1024, ttl=64 (reply in 95)
95	149.961248987	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x29ce, seq=4/1024, ttl=63 (request in 94)
96	150.166949586	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8002
97	150.985112147	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x29ce, seq=5/1280, ttl=64 (reply in 98)
98	150.985249455	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x29ce, seq=5/1280, ttl=63 (request in 97)
99	152.009122743	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x29ce, seq=6/1536, ttl=64 (reply in 100)
100	152.009242101	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x29ce, seq=6/1536, ttl=63 (request in 99)
101	152.023567526	HewlettPacka_5a:7d:…	HewlettPacka_61:24:…	ARP	42 Who has 172.16.30.1? Tell 172.16.30.254
102	152.023653571	HewlettPacka_61:24:…	HewlettPacka_5a:7d:…	ARP	60 172.16.30.1 is at 00:21:5a:61:24:92
103	152.169198685	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8002
104	153.033163020	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x29ce, seq=7/1792, ttl=64 (reply in 105)
105	153.033303681	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x29ce, seq=7/1792, ttl=63 (request in 104)
106	154.057238359	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x29ce, seq=8/2048, ttl=64 (reply in 107)
107	154.057408632	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x29ce, seq=8/2048, ttl=63 (request in 106)

Imagem 9 – Ping do tux33 para o tux32 na interface eth0

78	136.151258275	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:26 Cost = 0 Port = 0x8002
79	136.895002643	KYE_25:24:5b	Broadcast	ARP	42 Who has 172.16.31.1? Tell 172.16.31.253
80	136.895132338	HewlettPacka_61:30:…	KYE_25:24:5b	ARP	60 172.16.31.1 is at 00:21:5a:61:30:63
81	136.895136529	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x29ce, seq=1/256, ttl=63 (reply in 82)
82	136.895241361	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x29ce, seq=1/256, ttl=64 (request in 81)
83	137.901874467	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x29ce, seq=2/512, ttl=63 (reply in 84)
84	137.901988727	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x29ce, seq=2/512, ttl=64 (request in 83)
85	138.153488377	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:26 Cost = 0 Port = 0x8002
86	138.925904478	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x29ce, seq=3/768, ttl=63 (reply in 87)
87	138.926045767	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x29ce, seq=3/768, ttl=64 (request in 86)
88	139.949953276	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x29ce, seq=4/1024, ttl=63 (reply in 89)
89	139.950068095	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x29ce, seq=4/1024, ttl=64 (request in 88)
90	140.155731190	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:26 Cost = 0 Port = 0x8002
91	140.973955910	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x29ce, seq=5/1280, ttl=63 (reply in 92)
92	140.974067935	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x29ce, seq=5/1280, ttl=64 (request in 91)
93	141.983057207	HewlettPacka_61:30:…	KYE_25:24:5b	ARP	60 Who has 172.16.31.253? Tell 172.16.31.0
94	141.983074178	KYE_25:24:5b	HewlettPacka_61:30:…	ARP	42 172.16.31.253 is at 00:c0:df:25:24:5b
95	141.997956587	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x29ce, seq=6/1536, ttl=63 (reply in 96)
96	141.998061978	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x29ce, seq=6/1536, ttl=64 (request in 95)
97	142.157979520	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:26 Cost = 0 Port = 0x8002
98	143.022009576	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x29ce, seq=7/1792, ttl=63 (reply in 99)
99	143.022121672	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x29ce, seq=7/1792, ttl=64 (request in 98)
100	144.046085473	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x29ce, seq=8/2048, ttl=63 (reply in 101)
101	144.046227321	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x29ce, seq=8/2048, ttl=64 (request in 100)

Imagem 10 – Ping do tux33 para o tux32 na interface eth1

9	12.200480254	172.16.31.253	172.16.30.1	ICMP	98 Echo (ping) reply id=0x24ae, seq=1/256, ttl=64 (request in 8)
10	13.218973480	172.16.30.1	172.16.31.253	ICMP	98 Echo (ping) request id=0x24ae, seq=2/512, ttl=64 (reply in 11)
11	13.219103103	172.16.31.253	172.16.30.1	ICMP	98 Echo (ping) reply id=0x24ae, seq=2/512, ttl=64 (request in 10)
12	14.005707539	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
13	14.242951564	172.16.30.1	172.16.31.253	ICMP	98 Echo (ping) request id=0x24ae, seq=3/768, ttl=64 (reply in 14)
14	14.243074062	172.16.31.253	172.16.30.1	ICMP	98 Echo (ping) reply id=0x24ae, seq=3/768, ttl=64 (request in 13)
15	15.266951716	172.16.30.1	172.16.31.253	ICMP	98 Echo (ping) request id=0x24ae, seq=4/1024, ttl=64 (reply in 16)
16	15.267081548	172.16.31.253	172.16.30.1	ICMP	98 Echo (ping) reply id=0x24ae, seq=4/1024, ttl=64 (request in 15)
17	16.007894991	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
18	16.290951101	172.16.30.1	172.16.31.253	ICMP	98 Echo (ping) request id=0x24ae, seq=5/1280, ttl=64 (reply in 19)
19	16.291106075	172.16.31.253	172.16.30.1	ICMP	98 Echo (ping) reply id=0x24ae, seq=5/1280, ttl=64 (request in 18)
20	17.314949787	172.16.30.1	172.16.31.253	ICMP	98 Echo (ping) request id=0x24ae, seq=6/1536, ttl=64 (reply in 21)
21	17.315075917	172.16.31.253	172.16.30.1	ICMP	98 Echo (ping) reply id=0x24ae, seq=6/1536, ttl=64 (request in 20)
22	17.378922555	HewlettPacka_61:24:…	HewlettPacka_5a:7d:…	ARP	42 Who has 172.16.30.254? Tell 172.16.30.1
23	17.379043238	HewlettPacka_5a:7d:…	HewlettPacka_61:24:…	ARP	60 172.16.30.254 is at 00:21:5a:5a:7d:74
24	17.381063699	HewlettPacka_5a:7d:…	HewlettPacka_61:24:…	ARP	60 Who has 172.16.30.1? Tell 172.16.30.254
25	17.381077317	HewlettPacka_61:24:…	HewlettPacka_5a:7d:…	ARP	42 172.16.30.1 is at 00:21:5a:61:24:92
26	18.010091453	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
27	18.338950009	172.16.30.1	172.16.31.253	ICMP	98 Echo (ping) request id=0x24ae, seq=7/1792, ttl=64 (reply in 28)
28	18.339075791	172.16.31.253	172.16.30.1	ICMP	98 Echo (ping) reply id=0x24ae, seq=7/1792, ttl=64 (request in 27)
29	19.362953375	172.16.30.1	172.16.31.253	ICMP	98 Echo (ping) request id=0x24ae, seq=8/2048, ttl=64 (reply in 30)
30	19.363082858	172.16.31.253	172.16.30.1	ICMP	98 Echo (ping) reply id=0x24ae, seq=8/2048, ttl=64 (request in 29)
31	20.012277718	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
32	20.386962048	172.16.30.1	172.16.31.253	ICMP	98 Echo (ping) request id=0x24ae, seq=9/2304, ttl=64 (reply in 33)
33	20.387093766	172.16.31.253	172.16.30.1	ICMP	98 Echo (ping) reply id=0x24ae, seq=9/2304, ttl=64 (request in 32)
34	21.410966182	172.16.30.1	172.16.31.253	ICMP	98 Echo (ping) request id=0x24ae, seq=10/2560, ttl=64 (reply in 35)

Imagem 11 – Ping do tux33 para o 172.16.31.253

16	28.037898830	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x23b9, seq=1/256, ttl=64 (reply in 17)
17	28.038057226	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x23b9, seq=1/256, ttl=64 (request in 16)
18	29.043686339	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x23b9, seq=2/512, ttl=64 (reply in 19)
19	29.043816101	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x23b9, seq=2/512, ttl=64 (request in 18)
20	30.032946017	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
21	30.067686981	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x23b9, seq=3/768, ttl=64 (reply in 22)
22	30.067808851	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x23b9, seq=3/768, ttl=64 (request in 21)
23	31.091686994	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x23b9, seq=4/1024, ttl=64 (reply in 24)
24	31.091846228	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x23b9, seq=4/1024, ttl=64 (request in 23)
25	32.035133050	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
26	32.115697204	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x23b9, seq=5/1280, ttl=64 (reply in 27)
27	32.115820052	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x23b9, seq=5/1280, ttl=64 (request in 26)
28	33.079658516	HewlettPacka_61:24:…	HewlettPacka_5a:7d:…	ARP	42 Who has 172.16.30.254? Tell 172.16.30.1
29	33.079779060	HewlettPacka_5a:7d:…	HewlettPacka_61:24:…	ARP	60 172.16.30.254 is at 00:21:5a:5a:7d:74
30	33.083307847	HewlettPacka_5a:7d:…	HewlettPacka_61:24:…	ARP	60 Who has 172.16.30.1? Tell 172.16.30.254
31	33.083323701	HewlettPacka_61:24:…	HewlettPacka_5a:7d:…	ARP	42 172.16.30.1 is at 00:21:5a:61:24:92
32	33.139688906	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x23b9, seq=6/1536, ttl=64 (reply in 33)
33	33.139803513	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x23b9, seq=6/1536, ttl=64 (request in 32)
34	34.037318547	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
35	34.163682494	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x23b9, seq=7/1792, ttl=64 (reply in 36)
36	34.163806249	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x23b9, seq=7/1792, ttl=64 (request in 35)
37	35.187669866	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x23b9, seq=8/2048, ttl=64 (reply in 38)
38	35.187793831	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x23b9, seq=8/2048, ttl=64 (request in 37)
39	36.039502437	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
40	36.211686500	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x23b9, seq=9/2304, ttl=64 (reply in 41)

Imagem 12 – Ping do tux33 para o 172.16.30.254

10	10.462375853	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x252f, seq=1/256, ttl=64 (reply in 11)
11	10.462647320	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x252f, seq=1/256, ttl=63 (request in 10)
12	11.475522576	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x252f, seq=2/512, ttl=64 (reply in 13)
13	11.475764361	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x252f, seq=2/512, ttl=63 (request in 12)
14	12.013130162	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
15	12.499523287	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x252f, seq=3/768, ttl=64 (reply in 16)
16	12.499763466	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x252f, seq=3/768, ttl=63 (request in 15)
17	13.523522672	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x252f, seq=4/1024, ttl=64 (reply in 18)
18	13.523760685	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x252f, seq=4/1024, ttl=63 (request in 17)
19	14.015318033	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
20	14.547523523	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x252f, seq=5/1280, ttl=64 (reply in 21)
21	14.547789053	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x252f, seq=5/1280, ttl=63 (request in 20)
22	15.474857502	HewlettPacka_5a:7d:…	HewlettPacka_61:24:…	ARP	60 Who has 172.16.30.1? Tell 172.16.30.254
23	15.474879292	HewlettPacka_61:24:…	HewlettPacka_5a:7d:…	ARP	42 172.16.30.1 is at 00:21:5a:61:24:92
24	15.475484802	HewlettPacka_61:24:…	HewlettPacka_5a:7d:…	ARP	42 Who has 172.16.30.254? Tell 172.16.30.1
25	15.475572101	HewlettPacka_5a:7d:…	HewlettPacka_61:24:…	ARP	60 172.16.30.254 is at 00:21:5a:5a:7d:74
26	15.571517879	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x252f, seq=6/1536, ttl=64 (reply in 27)
27	15.571751562	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x252f, seq=6/1536, ttl=63 (request in 26)
28	16.017511003	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
29	16.595522571	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x252f, seq=7/1792, ttl=64 (reply in 30)
30	16.595764845	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x252f, seq=7/1792, ttl=63 (request in 29)
31	17.619521467	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x252f, seq=8/2048, ttl=64 (reply in 32)
32	17.619759061	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x252f, seq=8/2048, ttl=63 (request in 31)
33	18.019696220	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
34	18.643520223	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x252f, seq=9/2304, ttl=64 (reply in 35)
35	18.643754884	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x252f, seq=9/2304, ttl=63 (request in 34)

Imagem 13 – Ping do tux33 para o tux32

30	17.513855149	HewlettPacka_61:24:…	HewlettPacka_5a:7d:…	ARP	42 Who has 172.16.30.254? Tell 172.16.30.1
31	17.513963753	HewlettPacka_5a:7d:…	HewlettPacka_61:24:…	ARP	60 172.16.30.254 is at 00:21:5a:5a:7d:74
32	17.952659344	172.16.30.1	172.16.31.254	ICMP	98 Echo (ping) request id=0x0ef3, seq=2/512, ttl=64 (reply in 33)
33	17.952924741	172.16.31.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x0ef3, seq=2/512, ttl=63 (request in 32)
34	18.009404479	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
35	18.953892252	172.16.30.1	172.16.31.254	ICMP	98 Echo (ping) request id=0x0ef3, seq=3/768, ttl=64 (reply in 36)
36	18.954155065	172.16.31.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x0ef3, seq=3/768, ttl=63 (request in 35)
37	19.977896073	172.16.30.1	172.16.31.254	ICMP	98 Echo (ping) request id=0x0ef3, seq=4/1024, ttl=64 (reply in 38)
38	19.978155114	172.16.31.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x0ef3, seq=4/1024, ttl=63 (request in 37)
39	20.011573260	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
40	21.001892699	172.16.30.1	172.16.31.254	ICMP	98 Echo (ping) request id=0x0ef3, seq=5/1280, ttl=64 (reply in 41)
41	21.002176115	172.16.31.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x0ef3, seq=5/1280, ttl=63 (request in 40)
42	22.013739807	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
43	22.020821433	HewlettPacka_5a:7d:…	HewlettPacka_61:24:…	ARP	60 Who has 172.16.30.1? Tell 172.16.30.254
44	22.020837776	HewlettPacka_61:24:…	HewlettPacka_5a:7d:…	ARP	42 172.16.30.1 is at 00:21:5a:61:24:92
45	22.025884996	172.16.30.1	172.16.31.254	ICMP	98 Echo (ping) request id=0x0ef3, seq=6/1536, ttl=64 (reply in 46)
46	22.026116800	172.16.31.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x0ef3, seq=6/1536, ttl=63 (request in 45)
47	22.457921919	172.16.30.1	8.8.8.8	DNS	81 Standard query 0x961f A 2.debian.pool.ntp.org
48	22.457932116	172.16.30.1	8.8.8.8	DNS	81 Standard query 0x392a AAAA 2.debian.pool.ntp.org

Imagem 14 – Ping do tux33 para o router

95	45.676851296	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x0ebf, seq=11/2816, ttl=64 (reply in 96)
96	45.676993423	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x0ebf, seq=11/2816, ttl=64 (request in 95)
97	46.622221444	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
98	46.692893280	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x0ebf, seq=12/3072, ttl=64 (reply in 99)
99	46.693017528	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x0ebf, seq=12/3072, ttl=64 (request in 98)
100	47.716890256	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x0ebf, seq=13/3328, ttl=64 (reply in 101)
101	47.717029869	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x0ebf, seq=13/3328, ttl=64 (request in 100)
102	48.624378632	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
103	48.740892890	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x0ebf, seq=14/3584, ttl=64 (reply in 104)
104	48.741018045	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x0ebf, seq=14/3584, ttl=64 (request in 103)
105	49.764891193	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x0ebf, seq=15/3840, ttl=64 (reply in 106)
106	49.765020050	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x0ebf, seq=15/3840, ttl=64 (request in 105)
107	50.053406024	172.16.30.1	8.8.8.8	DNS	81 Standard query 0x1ba7 A 1.debian.pool.ntp.org
108	50.053415942	172.16.30.1	8.8.8.8	DNS	81 Standard query 0x08b2 AAAA 1.debian.pool.ntp.org

Imagem 15 – Ping do tux33 para o tux34

47	26.685646408	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x0e59, seq=3/768, ttl=64 (reply in 48)
48	26.685898047	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x0e59, seq=3/768, ttl=63 (request in 47)
49	27.702284977	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x0e59, seq=4/1024, ttl=64 (reply in 50)
50	27.702524183	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x0e59, seq=4/1024, ttl=63 (request in 49)
51	28.020264798	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
52	28.726295782	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x0e59, seq=5/1280, ttl=64 (reply in 53)
53	28.726537712	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x0e59, seq=5/1280, ttl=63 (request in 52)
54	29.750282631	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x0e59, seq=6/1536, ttl=64 (reply in 55)
55	29.750500885	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x0e59, seq=6/1536, ttl=63 (request in 54)
56	30.021770856	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8e:22 Cost = 0 Port = 0x8001
57	30.774259912	172.16.30.1	172.16.31.1	ICMP	98 Echo (ping) request id=0x0e59, seq=7/1792, ttl=64 (reply in 58)
58	30.774521048	172.16.31.1	172.16.30.1	ICMP	98 Echo (ping) reply id=0x0e59, seq=7/1792, ttl=63 (request in 57)
59	31.409620246	172.16.30.1	8.8.8.8	DNS	81 Standard query 0xd34f A 2.debian.pool.ntp.org
60	31.409630303	172.16.30.1	8.8.8.8	DNS	81 Standard query 0x485a AAAA 2.debian.pool.ntp.org

Imagem 16 – Ping do tux33 para o tux32

21 22.643738786	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) request id=0x6ea3, seq=4/1024, ttl=64 (reply in 23)
22 22.643882796	172.16.31.254	172.16.31.1	ICMP	126	Redirect (Redirect for host)
23 22.644090849	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x6ea3, seq=4/1024, ttl=63 (request in 21)
24 23.667740108	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) request id=0x6ea3, seq=5/1280, ttl=64 (reply in 25)
25 23.668046077	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x6ea3, seq=5/1280, ttl=63 (request in 24)
26 24.025273716	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/74:4d:28:ea:74:f5 Cost = 10 Port = 0x8001
27 24.595705752	HewlettPacka_61:30:…	Routerboardc_ea:74:…	ARP	42	Who has 172.16.31.254? Tell 172.16.31.1
28 24.595828391	Routerboardc_ea:74:…	HewlettPacka_61:30:…	ARP	60	172.16.31.254 is at 74:4d:28:ea:74:f5
29 24.596308609	KYE_25:24:5b	HewlettPacka_61:30:…	ARP	60	Who has 172.16.31.1? Tell 172.16.31.253
30 24.596317199	HewlettPacka_61:30:…	KYE_25:24:5b	ARP	42	172.16.31.1 is at 00:21:5a:61:30:63
31 24.691725996	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) request id=0x6ea3, seq=6/1536, ttl=64 (reply in 33)
32 24.691863790	172.16.31.254	172.16.31.1	ICMP	126	Redirect (Redirect for host)
33 24.692045094	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x6ea3, seq=6/1536, ttl=63 (request in 31)
34 25.715737795	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) request id=0x6ea3, seq=7/1792, ttl=64 (reply in 35)
35 25.716030773	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x6ea3, seq=7/1792, ttl=63 (request in 34)
36 26.027222661	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/74:4d:28:ea:74:f5 Cost = 10 Port = 0x8001
37 26.739737301	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) request id=0x6ea3, seq=8/2048, ttl=64 (reply in 39)
38 26.739888365	172.16.31.254	172.16.31.1	ICMP	126	Redirect (Redirect for host)
39 26.740073161	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x6ea3, seq=8/2048, ttl=63 (request in 37)
40 27.763735760	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) request id=0x6ea3, seq=9/2304, ttl=64 (reply in 42)
41 27.763889408	172.16.31.254	172.16.31.1	ICMP	126	Redirect (Redirect for host)
42 27.764097600	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x6ea3, seq=9/2304, ttl=63 (request in 40)
43 28.029369463	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/74:4d:28:ea:74:f5 Cost = 10 Port = 0x8001
44 28.787737502	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) request id=0x6ea3, seq=10/2560, ttl=64 (reply in 45)
45 28.788048917	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x6ea3, seq=10/2560, ttl=63 (request in 44)

Imagem 17 – Ping do tux32 para o tux33 (Sem o tux34, apenas com router)

7 6.315898031	172.16.31.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x70c2, seq=2/512, ttl=64 (reply in 8)
8 6.316045742	172.16.1.254	172.16.31.1	ICMP	98	Echo (ping) reply id=0x70c2, seq=2/512, ttl=64 (request in 7)
9 7.339898585	172.16.31.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x70c2, seq=3/768, ttl=64 (reply in 10)
10 7.340050626	172.16.1.254	172.16.31.1	ICMP	98	Echo (ping) reply id=0x70c2, seq=3/768, ttl=64 (request in 9)
11 8.008584832	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/74:4d:28:ea:74:f5 Cost = 10 Port = 0x8001
12 8.363913876	172.16.31.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x70c2, seq=4/1024, ttl=64 (reply in 13)
13 8.364069060	172.16.1.254	172.16.31.1	ICMP	98	Echo (ping) reply id=0x70c2, seq=4/1024, ttl=64 (request in 12)
14 9.387916804	172.16.31.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x70c2, seq=5/1280, ttl=64 (reply in 15)
15 9.388090845	172.16.1.254	172.16.31.1	ICMP	98	Echo (ping) reply id=0x70c2, seq=5/1280, ttl=64 (request in 14)
16 10.010760477	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/74:4d:28:ea:74:f5 Cost = 10 Port = 0x8001
17 10.294056249	Routerboardc_ea:74:…	HewlettPacka_61:30:…	ARP	60	Who has 172.16.31.1? Tell 172.16.31.254
18 10.294065119	HewlettPacka_61:30:…	Routerboardc_ea:74:…	ARP	42	172.16.31.1 is at 00:21:5a:61:30:63
19 10.411892775	172.16.31.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x70c2, seq=6/1536, ttl=64 (reply in 20)
20 10.412031267	172.16.1.254	172.16.31.1	ICMP	98	Echo (ping) reply id=0x70c2, seq=6/1536, ttl=64 (request in 19)
21 10.475876100	HewlettPacka_61:30:…	Routerboardc_ea:74:…	ARP	42	Who has 172.16.31.254? Tell 172.16.31.1
22 10.475962422	Routerboardc_ea:74:…	HewlettPacka_61:30:…	ARP	60	172.16.31.254 is at 74:4d:28:ea:74:f5
23 11.435894935	172.16.31.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x70c2, seq=7/1792, ttl=64 (reply in 24)
24 11.436043694	172.16.1.254	172.16.31.1	ICMP	98	Echo (ping) reply id=0x70c2, seq=7/1792, ttl=64 (request in 23)
25 12.002856002	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/74:4d:28:ea:74:f5 Cost = 10 Port = 0x8001
26 12.459896956	172.16.31.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x70c2, seq=8/2048, ttl=64 (reply in 27)
27 12.460036636	172.16.1.254	172.16.31.1	ICMP	98	Echo (ping) reply id=0x70c2, seq=8/2048, ttl=64 (request in 26)
28 13.483914621	172.16.31.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x70c2, seq=9/2304, ttl=64 (reply in 29)
29 13.484063170	172.16.1.254	172.16.31.1	ICMP	98	Echo (ping) reply id=0x70c2, seq=9/2304, ttl=64 (request in 28)
30 14.005183968	Routerboardc_1c:8e:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/74:4d:28:ea:74:f5 Cost = 10 Port = 0x8001
31 14.507923905	172.16.31.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x70c2, seq=10/2560, ttl=64 (reply in 32)
32 14.508071756	172.16.1.254	172.16.31.1	ICMP	98	Echo (ping) reply id=0x70c2, seq=10/2560, ttl=64 (request in 31)
33 15.531921316	172.16.31.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x70c2, seq=11/2816, ttl=64 (reply in 34)

Imagem 18 – Ping do tux32 para o router (Com accept redirect)

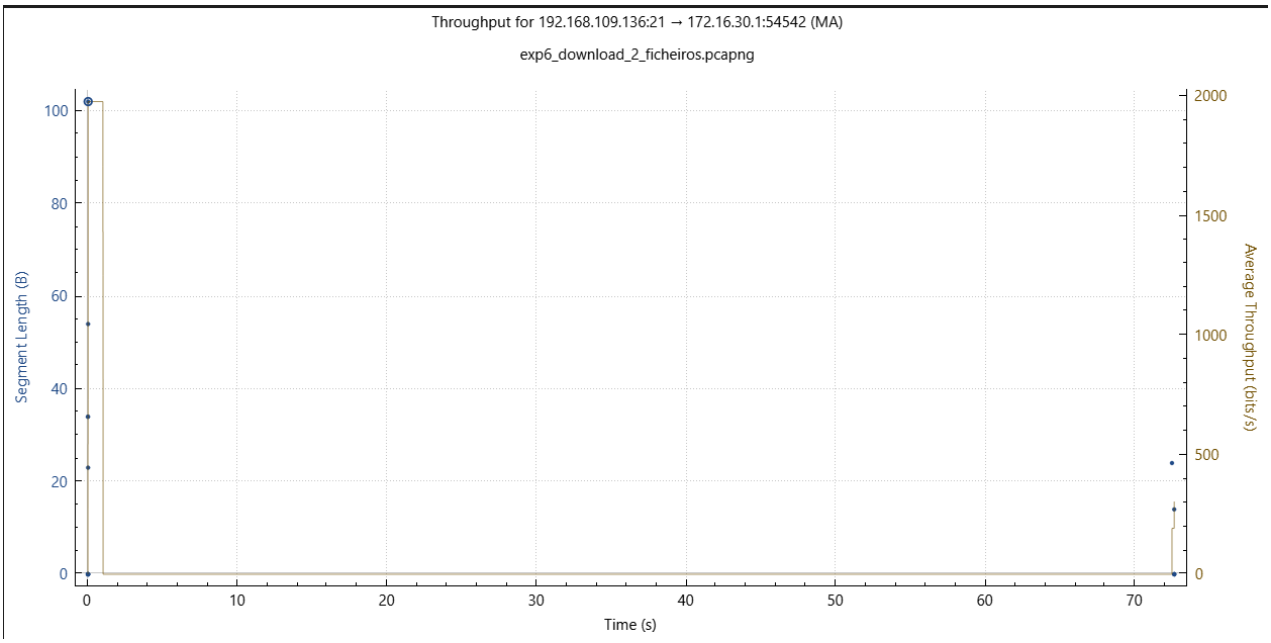


Imagem 19 – *Throughput* da captura do download de dois ficheiros em simultâneo

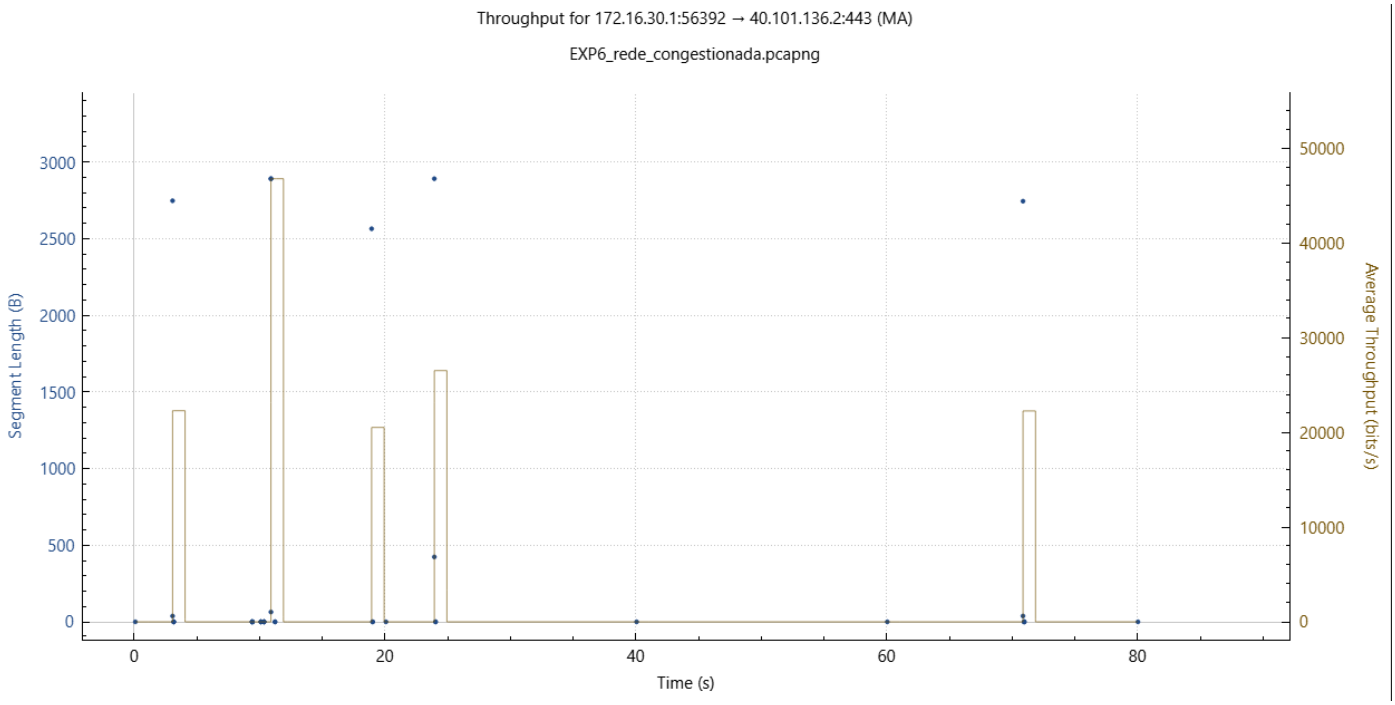


Imagem 20 – *Throughput* da captura do download de um ficheiro numa rede congestionada

Código da aplicação: “app.c”

```c
#include "../include/app.h"

//protocol FTP (File Transfer Protocol) client

int parseURL(char *input_url, struct urlArguments *url){
    //input_url format - ftp://[<user>:<password>@]<host>/<url-path>
    //input_url format anonymous - ftp://<host>/<url-path>

    const char *ftp_prefix = "ftp://";
    if (strncmp(input_url, ftp_prefix, strlen(ftp_prefix)) != 0) {
        printf("Invalid URL format\n");
        return -1;  // Indicate failure
    }

    char *url_part = input_url + strlen(ftp_prefix);

    // Find the user and password
    char *user_end = strchr(url_part, ':');
    if (user_end == NULL) {
```

```
        printf("WARNING: Anonymous MODE\n");
        strncpy(url->user, "anonymous", sizeof(url->user) - 1);
        strncpy(url->password, "anonymous@", sizeof(url->password) - 1);

        url_part = url_part;
    } else {
        // Parse the user and password
        strncpy(url->user, url_part, user_end - url_part);
        url->user[user_end - url_part] = '\0';

        char *password_end = strchr(user_end, '@');
        if (password_end == NULL) {
            printf("ERROR: No password\n");
            return -1;
        }

        strncpy(url->password, user_end + 1, password_end - user_end - 1);
        url->password[password_end - user_end - 1] = '\0'; //

        url_part = password_end + 1;
    }

    // Find the host and path
    char *host_end = strchr(url_part, '/');
    if (host_end == NULL) {
        printf("ERROR: No path\n");
        return -1;
    }

    strncpy(url->host, url_part, host_end - url_part);
    url->host[host_end - url_part] = '\0';

    strncpy(url->path, host_end + 1, MAX_LEN - 1);
```

```
    // Find the filename
    char *filename_start = strrchr(input_url, '/');
    if (filename_start == NULL) {
        printf("Invalid URL format\n");
        return -1;
    }

    strncpy(url->filename, filename_start + 1, MAX_LEN - 1);

    if(getIP(url->host, url->ip) != 0){
        printf("ERROR: Could not get IP\n");
        return -1;
    }
    return 0;

}

int createSocket(char *ip, int port){
    int sockfd;
    struct sockaddr_in server_addr;

    /*server address handling*/
    bzero((char *) &server_addr, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = inet_addr(ip);
    server_addr.sin_port = htons(port);

    /*open a TCP socket*/
    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket()");
        exit(-1);
    }
    /*connect to the server*/
    if (connect(sockfd,
```

```
                (struct sockaddr *) &server_addr,
                sizeof(server_addr)) < 0) {
        perror("connect()");
        exit(-1);
    }
    return sockfd;
}

int readResponse(int socket, char *buf){
    char byte = 0;
    int i = 0;
    State state = INITIALIZING;
    memset(buf, 0, MAX_LEN);

    while (state != RESPONSE_COMPLETE) {
        if (read(socket, &byte, 1) < 0) {
            printf("ERROR: Could not read from socket\n");
            return -1;
        }

        switch (state) {
            case INITIALIZING:
                if (byte == ' ') {
                    state = RECEIVING_SINGLE;
                } else if (byte == '-') {
                    state = RECEIVING_MULTIPLE;
                } else if (byte == '\n') {
                    state = RESPONSE_COMPLETE;
                }
                else buf[i++] = byte;
                break;
            case RECEIVING_SINGLE:
                if (byte == '\n') {
                    state = RESPONSE_COMPLETE;
```

```
                }
                else buf[i++] = byte;
                break;
            case RECEIVING_MULTIPLE:
                if (byte == '\n') {
                    memset(buf, 0, MAX_LEN);
                    state = INITIALIZING;
                    i = 0;
                }
                else buf[i++] = byte;
                break;
            case RESPONSE_COMPLETE:
                break;
            default:
                printf("ERROR: Invalid state\n");
        }
    }

    printf("Buffer: %s\n", buf);
    int responseCode = atoi(buf);
    printf("Code: %d\n", responseCode);

    return responseCode;
}


int getIP(char *host, char *ip) {
    struct hostent *h;

    if ((h = gethostbyname(host)) == NULL) {
        herror("gethostbyname");
        return -1;
    }

    strcpy(ip, inet_ntoa(*((struct in_addr *) h->h_addr)));
```

```
    return 0;
}

int authentication(int socket, char *user, char *password) {
    char userCommand[6 + strlen(user) + 1];  // "USER " + user + "\n"
    char passCommand[6 + strlen(password) + 1];  // "PASS " + password + "\n"
    char result[MAX_LEN];

    snprintf(userCommand, sizeof(userCommand), "USER %s\n", user);

    write(socket, userCommand, strlen(userCommand));

    // Check server response
    if (readResponse(socket, result) != READY_TO_PASS) {
        printf("ERROR: Server not ready to receive password\n");
        return -1;
    }

    snprintf(passCommand, sizeof(passCommand), "PASS %s\n", password);

    write(socket, passCommand, strlen(passCommand));

    return readResponse(socket, result);
}

int changePassiveMode(int socket, char *ip_address, int *port_socketB) {
    char pasvCommand[] = "pasv\n";
    char result[MAX_LEN];
    int ip1, ip2, ip3, ip4, port1, port2;

    //Send passive mode command
    write(socket, pasvCommand, strlen(pasvCommand));

    if (readResponse(socket, result) != PASSIVE_MODE) {
```

```
        printf("ERROR: Server not ready to enter passive mode\n");
        return -1;
    }
    sscanf(result, "227 Entering Passive Mode (%d,%d,%d,%d,%d,%d)", &ip1, &ip2, &ip3,
&ip4, &port1, &port2);

    *port_socketB = port1 * 256 + port2; // Calculate port number

    sprintf(ip_address, "%d.%d.%d.%d", ip1, ip2, ip3, ip4);

    return 0;
}

int requestPath(int socket, char *path) {
    char retrCommand[5+strlen(path)+1];
    char result[MAX_LEN];

    sprintf(retrCommand, "retr %s\n", path);
    write(socket, retrCommand, strlen(retrCommand));

    if (readResponse(socket, result) != FILE_OK) {
        printf("ERROR: Server not ready to receive file\n");
        return -1;
    }

    return 0;
}

int getFile(int socketB, int socketA, char *filename) {

    FILE *file = fopen(filename, "w");

    if (file == NULL) {
        printf("ERROR: Could not open file\n");
        return -1;
```

```
    }

    char buf[MAX_LEN];
    int bytes;

    // Read from socket B and write to file
    while ((bytes = read(socketB, buf, MAX_LEN)) > 0) {
        if (fwrite(buf, bytes, 1, file) < 0) {
        fclose(file);
        return -1;
    }
    }

    fclose(file);

    if (readResponse(socketA, buf) != TRANSFER_COMPLETE) {
        printf("ERROR: File transfer not complete\n");
        return -1;
    }

    return 0;
}

int endConnection(const int socketA, const int socketB) {

    char result[MAX_LEN];
    char quitCommand[] = "quit\n";

    write(socketA, quitCommand, strlen(quitCommand));

    if(readResponse(socketA, result) != QUIT) {
        printf("ERROR: Could not quit\n");
        return -1;
    }
```

```
    return close(socketA) || close(socketB);
}

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

    if (argc != 2)
    {
        fprintf(stderr,"Wrong number of arguments. Usage: download
ftp://[<user>:<password>@]<host>/<url-path>\n");
        exit(1);
    }

    struct urlArguments url;
    memset(&url,0,sizeof(url)); // Initialize struct to 0

    if (parseURL(argv[1],&url) != 0)
    {
        fprintf(stderr,"Error parsing URL\n");
        exit(1);
    }

    // Print all the URL arguments
    printf("Host name: %s\n", url.host);
    printf("Path: %s\n", url.path);
    printf("User: %s\n", url.user);
    printf("Password: %s\n", url.password);
    printf("File name: %s\n", url.filename);
    printf("IP Address: %s\n", url.ip);

    int socketA = createSocket(url.ip, SERVER_PORT);
    if (socketA < 0) {
        printf("ERROR: Could not create socket A\n");
        exit(-1);
```

```
}

char buf[MAX_LEN];

if (readResponse(socketA, buf) != READY_TO_AUTH) {
    printf("ERROR: Server not ready to authenticate\n");
    exit(-1);
}

// Authentication
if (authentication(socketA, url.user, url.password) != LOGIN_SUCCESS) {
    printf("ERROR: Could not authenticate\n");
    exit(-1);
}

// Passive mode
int port_socketB;
char ip_address[MAX_LEN];
if (changePassiveMode(socketA, ip_address, &port_socketB) != 0) {
    printf("ERROR: Could not enter passive mode\n");
    exit(-1);
}

// Create socket for data transfer
int socketB = createSocket(ip_address, port_socketB);
if (socketB < 0) {
    printf("ERROR: Could not create socket B\n");
    exit(-1);
}

if (requestPath(socketA, url.path) != 0) {
    printf("ERROR: Could not request path\n");
    exit(-1);
}
```

```
    if (getFile(socketB, socketA, url.filename) != 0) {
        printf("ERROR: Could not get file\n");
        exit(-1);
    }

    if (endConnection(socketA, socketB) != 0) {
        printf("ERROR: Could not close sockets\n");
        exit(-1);
    }

    return 0;

}
```

Código da aplicação: “app.h”

```
#pragma once

#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <netdb.h>
#include <regex.h>

#include <string.h>
#include <strings.h>

#define MAX_LEN 256
#define SERVER_PORT 21
```

```c
#define READY_TO_AUTH 220
#define READY_TO_PASS 331

#define LOGIN_SUCCESS 230

#define PASSIVE_MODE 227

#define FILE_OK 150

#define TRANSFER_COMPLETE 226

#define QUIT 221


struct urlArguments {

    char host[MAX_LEN]; // 'ftp.up.pt'

    char path[MAX_LEN];

    char user[MAX_LEN];

    char password[MAX_LEN];

    char filename[MAX_LEN];

    char ip[MAX_LEN];

};


typedef enum{

    INITIALIZING,

    RECEIVING_SINGLE,  //state that receives a single response

    RECEIVING_MULTIPLE,  // state that receives a multiple response

    RESPONSE_COMPLETE

} State;


//Functions


int parseURL(char *url, struct urlArguments *arguments);

int createSocket(char *ip, int port);

int readResponse(int socket, char *buf);

int getIP(char *host, char *ip);

int authentication(int socket, char *user, char *password);

int changePassiveMode(int socket, char *ip, int *port_socketB);

int requestPath(int socket, char *path);

int getFile(int socketA, int socketB, char *filename);

int endConnection(const int socketA, const int socketB);
```