Faculdade de Engenharia da Universidade do Porto

Rede de Computadores

RC - Relatório Projeto 2

Turma: 3LEIC04

Trabalho realizado por:
Anete Pereira (up202008856)
Isabel Amaral (up202006677)
Milena Gouveia (up202008862)

Introdução

Este trabalho laboratorial foi desenvolvido no âmbito da unidade curricular de Redes de Computadores. O objetivo foi, numa primeira parte, o desenvolvimento de uma aplicação de download através de um cliente FTP e, numa segunda parte, a configuração de uma rede segundo uma série de experiências.

As experiências foram realizadas na bancada 4 do laboratório I321. No entanto, os *logs* apresentados neste relatório foram recolhidos na bancada 5 do laboratório I320.

Part 1 - Aplicação de Download

Arquitetura

A aplicação de download desenvolvida está dividida em componentes.

A componente *url_parser* inclui funções para verificar se um url é válido e para efetuar o seu *parsing*, ou seja divisão nas diferentes componentes que o constituem: *user*, *password*, *host* e *url_path*.

A componente *server_cmds* constitui uma interface para a comunicação com *sockets* incluindo funções genéricas para enviar comandos para o servidor, receber respostas, entrar em modo passivo e efetuar o download de um ficheiro.

A componente *download*, auxiliada pelas funções definidas nas duas componentes acima mencionadas, é onde o cliente FTP se encontra em funcionamento e são geridas as várias fases do protocolo: processamento dos argumentos, criação das *sockets* e sua conexão com o servidor, autenticação no servidor, passagem a modo passivo, envio do comando para iniciar o download do ficheiro e, por fim, para fechar as *sockets* criadas.

Download de Ficheiros

O programa da aplicação deve ser compilado utilizando o Makefile disponibilizado. Para correr o programa deverá ser fornecido o url do ficheiro a ser transferido no seguinte formato: **ftp://[<user>:<password>@]<host>/<url-path>**, em que:

- **user** e **password** são as credenciais de autenticação no servidor e são opcionais, caso estas componentes não sejam fornecidas será realizada uma autenticação em modo anónimo
- **host** é o servidor
- **url-path** é o *path* dentro do servidor para o ficheiro a ser transferido

Para testar a aplicação de download, efetuamos o download de ficheiros de diferentes tamanhos, com diferentes extensões e de diferentes servidores com e sem autenticação.

Alguns exemplos de execução:

- ./download ftp://netlab1.fe.up.pt/pub.txt
- ./download ftp://rcom:rcom@netlab1.fe.up.pt/pipe.txt
- ./download ftp://rcom:rcom@netlab1.fe.up.pt/files/pic1.jpg
- ./download ftp://rcom:rcom@netlab1.fe.up.pt/files/crab.mp4
- ./download ftp://ftp.up.pt/pub/debian/README

Part 2 - Configuração da Rede e Análise

Experiência 1 - Configuração de uma Rede IP

O objetivo desta experiência foi a configuração do IP do *tux3* e do *tux4* e das suas respectivas ligações ao *switch*, de modo a permitir a comunicação entre os mesmos.

Para concluir este objetivo, foi necessário conectar ambos os computadores ao *switch*.

Comandos utilizados:

- no *tux3:*
  - **ifconfig eth0 172.16.40.1/24**
- no *tux4:*
  - **ifconfig eth0 172.16.40.254/24**

Para testar a conexão entre os dois computadores foi utilizado o comando **ping**. Este gera pacotes do tipo ICMP, os quais nos permitem enviar *requests* e receber *replies* sendo, assim, possível confirmar que a conexão foi estabelecida com sucesso.

Para enviar um pacote de uma máquina para outra, é necessário saber o endereço MAC do destino. ARP *(Address Resolution Protocol*) é um protocolo usado para associar o endereço IP de uma máquina ao seu endereço MAC dentro de uma rede local, permitindo assim que esta tenha uma identificação exclusiva na rede. Estes mapeamentos são armazenados numa estrutura designada por ARP *table*.

No *tux3,* foram eliminadas as entradas existentes nesta tabela através dos comandos **arp -a** e **arp -d <endereço IP>.**Assim, o endereço MAC do *tux4* passou a ser desconhecido. Quando o *tux3* tenta dar ping em *tux4*, é possível verificar, através da análise dos *logs* do wireshark (ver figura 02), que o *tux3* envia inicialmente, por *broadcast*, um pacote ARP (os bytes 12-13 do cabeçalho da *frame ethernet* contém o valor 0x0806) do tipo *request* para obter o endereço MAC correspondente ao endereço IP do *tux4*.

Este pacote contém:

- endereço IP de origem: 172.16.40.1 (*tux3)*
- endereço MAC de origem: 00:21:5a:61:2c:54 (*tux3)*
- endereço IP de destino: 172.16.40.254 (*tux4*)
- endereço MAC de destino: 00:00:00:00:00:00 (*tux4*) - por ser desconhecido

O *tux4* responde igualmente com um pacote ARP do tipo *reply* que contém:

- endereço IP de origem: 172.16.40.254 (*tux4)*
- endereço MAC de origem: 00:c0:df:25:21:9e (*tux4)*
- endereço IP de destino: 172.16.40.1 (*tux3)*
- endereço MAC de destino: 00:21:5a:61:2c:54 (*tux3)*

Sabendo o endereço MAC pretendido, o comando ping passa a gerar pacotes do tipo IP (os bytes 12-13 do cabeçalho da frame ethernet contém o valor 0x0800) com o protocolo ICMP (byte 23 do cabeçalho com valor 0x01) que contêm o endereço IP e MAC de origem e de destino. O tamanho de cada trama recebida pode ser visualizada através do wireshark (ver figura 03).

A interface *loopback* é uma interface de rede virtual que os computadores usam para direcionar as respostas para si mesmos. É útil para testar vários programas de rede sem interferir com outros computadores na mesma rede.

Experiência 2 - Implementação de duas bridges num switch

O objetivo desta experiência foi, por um lado, perceber como é feita a criação de *bridges* dentro de um *switch* e a ligação dos computadores a cada uma delas e, por outro, perceber como funciona a comunicação entre máquinas dentro e fora da mesma rede local virtual.

Foram criadas duas *bridges* no *switch*, *bridge40* e *bridge41*. O *tux3* e o *tux4* foram ligados à primeira e o *tux2* à segunda. Quando uma máquina é ligada ao *switch*, é automaticamente adicionada à *bridge default*. Para efetuar a ligação de cada um dos computadores com a *bridge* adequada foi necessário, primeiro, removê-los da *bridge default* e, posteriormente, adicioná-los à *bridge* pretendida. Os IPs do *tux3* e do *tux4* já tinham sido configurados na experiência anterior, no entanto, foi necessário configurar o IP do *tux2*.

Comandos utilizados:

- no *tux2:*
  - **ifconfig eth0 172.16.41.1/24**
- no *switch* através do gtkTerm (eth0 do *tux3* foi ligado com o porto 1, eth0 do *tux4* foi ligado com o porto 12 e eth0 do *tux2* foi ligado com o porto 20)*:*
  - **/interface bridge add name=bridge40**
  - **/interface bridge add name=bridge41**
  - **/interface bridge port remove [find interface =ether1]**
  - **/interface bridge port remove [find interface =ether12]**
  - **/interface bridge port remove [find interface =ether20]**
  - **/interface bridge port add bridge=bridge40 interface=ether1**
  - **/interface bridge port add bridge=bridge40 interface=ether12**
  - **/interface bridge port add bridge=bridge41 interface=ether20**

A configuração foi testada, primeiro, através do ping de *tux4* em *tux3,* tendo-se verificado conectividade entre ambos (ver figura 05). No entanto, o ping de *tux2* em *tux3* não gerou

resposta (ver figura 06). Isto acontece porque, enquanto que o *tux3* e *tux4* se encontram na mesma sub-rede, o *tux2* encontra-se numa diferente sendo, deste modo, inacessível.

Posteriormente, experimentamos testar a conexão através de um ping broadcast na sub-rede de *tux3* através do comando **ping –b 172.16.40.255** no *tux3*. Este não gerou resposta (ver figura 07), porém, foi possível observar a chegada de pacotes ICMP no *tux4* por este se encontrar na mesma sub-rede que o *tux3*. Ao repetir-se o mesmo processo na sub-rede do *tux2* através do comando **ping –b 172.16.41.255** no *tux2*, novamente não foi possível obter qualquer resposta (ver figura 08). No entanto, desta vez verificou-se que nem o *tux3* nem o *tux4* receberam quaisquer pacotes. Isto deve-se ao facto do *tux2* estar isolado numa sub-rede que não possui nenhuma ligação com a sub-rede do *tux3/tux4*. Deste modo, é possível concluir que existem dois domínios de *broadcast* correspondentes à *bridge40* e à *bridge41*.

Experiência 3 - Configuração de um router

O objetivo desta experiência foi a utilização do *tux4* como um router com a finalidade de permitir a transmissão de pacotes entre o *tux2* e o *tux3*, os quais se encontram em sub-redes diferentes, não sendo possível, conforme foi verificado na experiência anterior, que essa transmissão seja feita diretamente.

Para concluir este objetivo, foi necessário ligar outra das interfaces do *tux4* ao *switch*, desta vez a interface eth1 à *bridge41*. Assim, o *tux4* passou a estar ligado às duas bridges do *switch*:

- *bridge40* através do host 172.16.40.254
- *bridge41* através do host 172.16.41.253

Comandos utilizados:

- no *tux4*:
  - **ifconfig eth1 172.16.41.253/24**
- no *switch*, através do gtkTerm (eth1 do *tux4* foi ligado com o porto 22):
  - **/interface bridge port remove [find interface=ether22]**
  - **/interface bridge port add bridge=bridge41 interface=ether22**

De seguida, foi necessário configurar a utilização do *tux4* como *router* através da ativação do *IP forwarding*, da desativação do *ICMP echo-ignore-broadcast* e da definição das *routes* necessárias no *tux2* e no *tux3* para que cada um conseguisse alcançar o outro por meio do *tux4*.

As *routes* definidas para cada máquina encontram-se listadas na respetiva ***forwarding table***. A informação sobre cada *route* encontra-se listada segundo o seguinte formato:

- **destination** - IP da máquina/sub-rede de destino
- **gateway** - IP da interface da máquina para a qual será realizado o próximo salto
- **mask** - combinação de bits utilizada para efetuar a divisão entre o endereço de rede e de *host* e utilizada para determinar se um IP pertence a uma determinada sub-rede ou não
- **interface** - placa de rede da máquina de origem utilizada para o envio da mensagem

Em todas as máquinas existem algumas *routes* geradas automaticamente ao serem ligadas ao *switch*. Estas *routes* têm destination correspondente ao IP da sub-rede de cada uma das interfaces da máquina e **gateway 0.0.0.0**.

- no tux2:
    - destination: 172.16.41.0, gateway 0.0.0.0 - devido à ligação à bridge41
- no tux3:
    - destination: 172.16.40.0, gateway 0.0.0.0 - devido à ligação à bridge40
- no tux4:
    - destination: 172.16.40.0, gateway 0.0.0.0 - devido à ligação à bridge40
    - destination: 172.16.41.0, gateway 0.0.0.0 - devido à ligação à bridge41

Isto significa que qualquer host dentro da sub-rede é alcançado diretamente, i.e. sem saltos intermédios. Além disso, podem também ser definidas *routes* para destinos específicos e *routes default* com **destination 0.0.0.0/0**, a ser usadas sempre que não houver nenhum destino mais específico definido na *forwarding table*.

Comandos utilizados:

- no *tux4*:
    - **echo 1 > /proc/sys/net/ipv4/ip_forward**
    - **echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts**
- no *tux2*:
    - **route add default gw 172.16.41.253**
- no *tux3*:
    - **route add default gw 172.16.40.254**

Através da análise dos logs do wireshark obtidos através do ping de *tux2* em *tux3* (ver figura 10), foi possível verificar verificar a transmissão e receção das seguintes mensagens ARP no *tux3*:

- **Who has 172.16.40.254? Tell 172.16.40.1** - apesar de estar a tentar enviar um pacote para a máquina com IP 172.16.41.1, como não há nenhum destino na *forwarding table* que dê match com este IP, a mensagem será enviado primeiro para a máquina 172.16.40.254 para, de seguida, ser enviado para o destino pretendido; por esta razão, é enviada uma mensagem ARP do *tux3* para a sub-rede de modo a encontrar a interface para onde o pacote deverá ser enviada
- **172.16.40.254 is at 00:22:64:19:09:5c** - o MAC da interface para onde a mensagem deverá ser enviada, correspondente ao IP 172.16.40.254, é enviado de volta para o *tux3*

Após a transmissão e receção das mensagens ARP, começam a ser observadas mensagens ICMP tanto do tipo *request*, enviadas do *tux3* para o *tux2*, como do tipo *reply*, enviadas do *tux2* para o *tux3*. Os IPs associados às mensagens ICMP são os das máquinas de origem e de destino, neste caso:

- 172.16.40.1 para o *tux3*
- 172.16.41.1 para o *tux2*

No entanto, os MACs associados são os das máquinas que, em cada salto, efetuam o papel de emissor e de recetor. No caso, do primeiro ICMP *request* enviado,

- 00:21:5a:61:2c:54 para a interface eth0 do *tux3*

- 00:22:64:19:09:5c para a interface eth0 do *tux4*, dado que é o primeiro salto intermédio dado para atingir o destino

Experiência 4 - Configuração de um router comercial

O objetivo desta experiência foi a configuração de um *router* comercial com a finalidade de efetuar a ligação da rede configurada entre os computadores do laboratório à rede do mesmo (172.16.1.0/24) usufruindo da funcionalidade NAT (*Network Address Translation*).

Para concluir este objetivo, foi necessário, por um lado, ligar a interface eth1 do *router* do laboratório à interface que fornece acesso à rede do laboratório, cuja funcionalidade NAT se encontra ativada por *default*, e, por outro lado, ligar a interface eth2 do *router* ao *switch* de modo a ser feita a ligação com os restantes computadores da bancada. Esta última ligação foi feita com a *bridge41*. Assim, o *router* passou a apresentar as seguintes ligações:

- *bridge41* através do host 172.16.41.254
- *internet* através do host 172.16.1.49

Comandos utilizados:

- no *router*, através do gtkTerm:
  - **/ip address add address=172.16.1.49/24 interface=ether1**
  - **/ip address add address=172.16.41.254/24 interface=ether2**
- no *switch*, através do gtkTerm (*router* foi ligado com o porto 23):
  - **/interface bridge port remove [find interface=ether23]**
  - **/interface bridge port add bridge=bridge41 interface=ether23**

De seguida, foi necessário configurar as *routes* estáticas de cada uma das máquinas de modo a que todos os elementos da configuração passassem a conseguir alcançar qualquer outro elemento e que o *router* comercial começasse a ser usado como *default*.

Comandos utilizados:

- no *router*, através do gtkTerm:
  - **/ip route add dst-address=172.16.40.0/24 gateway=172.16.41.253**
- no *tux2*:
  - **route del default gw 172.16.41.253** (definido na experiência anterior)
  - **route add default gw 172.16.41.254**
  - **route add -net 172.16.40.0/24 gw 172.16.41.253**
- *no tux4*
  - **route add default gw 172.16.41.254**

(no *tux3* continuava a estar definida uma *route default* para a interface eth0 do *tux4* proveniente da experiência anterior)

Como foi visto na experiência anterior, em cada uma das máquinas se, na tentativa de enviar um pacote para um determinado IP de destino, esse destino constituir uma entrada da *forwarding table*, a respetiva *gateway* é utilizada. Caso contrário, é utilizada a *route default*. Após este passo, todas as máquinas da configuração passam a conseguir alcançar qualquer outro elemento. Neste sentido, testamos a configuração através de pings em *tux3*:

da interface eth0 do *tux4* (ver figura 12), da interface eth1 do *tux4* (ver figura 13), do *tux2* (ver figura 14) e do *router* (ver figura 15).

Posteriormente, estudamos os caminhos seguidos pelos pacotes resultantes do ping de *tux3* em *tux2*. Como *tux2* e *tux3* não se encontram na mesma sub-rede, a ligação é feita através do *tux4*, uma vez que a interface eth1 do *tux4* foi definida no *tux2* como *gateway* para comunicar com a sub-rede 172.16.40.0/24:

- os pacotes do *tux2* são encaminhados para a interface eth1 do *tux4*
- os pacotes são encaminhados da interface eth0 do *tux4* para o *tux3*

Foi feita a experiência de apagar a *route* que liga o *tux2* à sub-rede do *tux3* através do *tux4*. O ping de *tux3* em *tux2* continua a ser possível, uma vez que o *router* é utilizado como *gateway default* do *tux2* e este, por sua vez, também utiliza a interface eth1 do *tux4* como *gateway* para comunicar com a sub-rede 172.16.40.0/24. O caminho seguido pelos pacotes será, portanto, o seguinte:

- os pacotes do *tux2* são encaminhados para a interface eth2 do *router*
- os pacotes são encaminhados da interface eth2 do *router* para a interface eth1 do *tux4*
- os pacotes são encaminhados da interface eth0 do *tux4* para o *tux3*

O último passo da experiência consistia em aceder ao *router* do laboratório (172.16.1.254), o qual, por sua vez, faz a ligação com a rede da FEUP e fornece *internet* às máquinas do laboratório. Para isto, foi necessário adicionar ao *router* comercial uma *route default* para o *router* do laboratório.

Comando utilizado:

- no *router*, através do gtkTerm:
    - **/ip route add dst-address=0.0.0.0/0 gateway=172.16.1.254**

Para que este último passo funcione, é necessário que a NAT esteja ativada, dado que é isto que permite a ligação das máquinas utilizadas nas experiências com a rede do laboratório. A funcionalidade NAT consiste na tradução dos endereços privados utilizados localmente nas máquinas do laboratório para endereços públicos compreensíveis à da rede do laboratório, i.e. fora da rede configurada.

A interface utilizada para ligar o *router* à rede do laboratório nesta experiência já possui NAT ativada por *default*. No entanto, é possível desativar e reativar esta funcionalidade através dos comandos **/ip firewall nat disable 0** e **/ip firewall nat ena 0**.

Experiência 5 - Configuração de DNS

O objetivo desta experiência foi configurar um serviço DNS (*Domain Name System*) nos computadores da rede.

A principal função deste serviço é a tradução de nomes de domínios para os seus endereços IP respetivos. O DNS é implementado como uma hierarquia de servidores de

nomes, onde cada servidor de nomes mapeia um nome para um endereço IP ou aponta para outros servidores de nomes que podem ser pesquisados pelo nome de domínio.

Depois de configurar o DNS, quando executamos um ping (ver figura 16), o *host* envia para o servidor um pacote com o *hostname* esperando que seja retornado o seu endereço IP (ver figura 17). O servidor responde com um pacote que contém o endereço IP correspondente ao *hostname* em causa (ver figura 18).

Para concluir este objetivo, foi necessário aceder ao ficheiro /etc/resolv.conf, apagar o que havia no ficheiro e adicionar “nameserver” seguido do IP que vai ser traduzido, neste caso para “172.16.1.1” que corresponde ao IP do servidor DNS services.netlab.fe.up.pt.

Depois de terminada a configuração, já foi possível a utilização de *hostnames* no acesso à *internet* pelos computadores, quer através do *browser*, quer através da execução do comando ping.

Comando utilizado:

- **ping www.google.com**

Experiência 6 - Ligação TCP

O objetivo desta experiência foi testar a aplicação de download na rede configurada e observar o comportamento do protocolo TCP (*Transmission Control Protocol*).

A aplicação de download desenvolvida abre duas conexões TCP. A primeira conexão é utilizada para enviar/receber comandos para/do servidor e para transportar informações de controlo FTP, tais como credenciais de utilizadores. A segunda conexão foi criada através da porta especificada pelo servidor após a entrada em modo passivo e foi utilizada para a transferência de dados entre o servidor e o cliente.

Cada conexão estabelecida subdivide-se em três fases. A primeira corresponde ao estabelecimento da ligação, o cliente envia um número de sequência inicial e o servidor responde com SYN e ACK, simultaneamente dando *acknowledgement* do cliente (ACK) e enviando o seu próprio número de sequência inicial (SYN), por fim, o cliente envia também um ACK para o servidor (ver figura 19). Na segunda fase, dá-se a troca de dados entre o cliente e o servidor sendo também usadas tramas ACK para dar *acknowledge* da receção de packets. Na terceira e última fase, é feita a desconexão nas duas direções (ver figura 20).

O mecanismo ARQ (*Automatic Repeat Request*) do TCP é utilizado através do método da janela deslizante e consiste no controlo dos erros que possam ocorrer durante a transmissão de dados. Esse controlo de erros recorre a *acknowledgement numbers*, que estão incluídos nas tramas enviadas pelo recetor indicando se esta foi recebida corretamente, ao parâmetro *window size*, que indica o domínio de pacotes possíveis de ser recebidos pelo recetor, e ao parâmetro *sequence number*, que indica o número do pacote a ser enviado (ver figura 21).

O TCP usa um mecanismo de controlo de congestão *end-to-end*, o que significa que o emissor limita ou aumenta a taxa de transferência de dados para a conexão em função do congestionamento percebido por ele de modo a evitar um grande declínio no desempenho da rede. A conexão TCP é composta por diversas variáveis, por exemplo a *CongestionWindow* que limita a taxa de envio de pacotes de um dado emissor TCP de acordo com o congestionamento da conexão.

Com o objetivo de testar se a taxa de transferência era afetada ao realizar dois downloads em simultâneo, primeiro iniciamos a transferência do crab.mp4 no *tux3*, um ficheiro maior e que, portanto, demora mais tempo a ser transferido, e, seguidamente, mudamos para o *tux2* e iniciamos a transferência do ficheiro pub.txt, dado que é menor e demora menos tempo a ser transferido.

Através da análise da variação do *throughput* ao longo do tempo, é possível observar que, quando o segundo download começa, a taxa de transferência total é conservada, mas a taxa de transferência de cada download varia consideravelmente. A taxa de transmissão de pacotes da conexão TCP que já estava iniciada diminui, uma vez que foi atribuída uma parte da taxa de transferência à outra conexão (ver figura 22). Após a conclusão do primeiro download, o segundo download passa a ocupar toda a largura de banda (ver figura 23).

Conclusões

As experiências realizadas permitiram entender o funcionamento de uma rede local simples com acesso à *internet* e aprofundar os nossos conhecimentos nos vários tópicos necessários à sua execução, nomeadamente endereços IP e MAC, o papel dos vários dispositivos utilizados, como o *switch* e o *router*, diversas funcionalidades, como NAT e DNS, e protocolos, tais como ICMP e ARP.

Anexos

Experiência 1

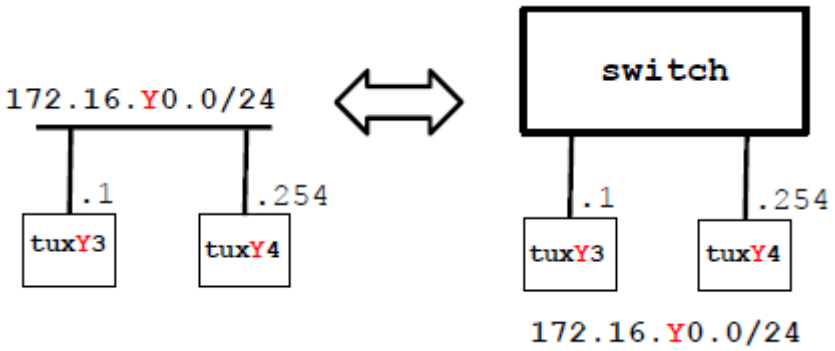


Figura 01: Experiência 1: Arquitetura da configuração

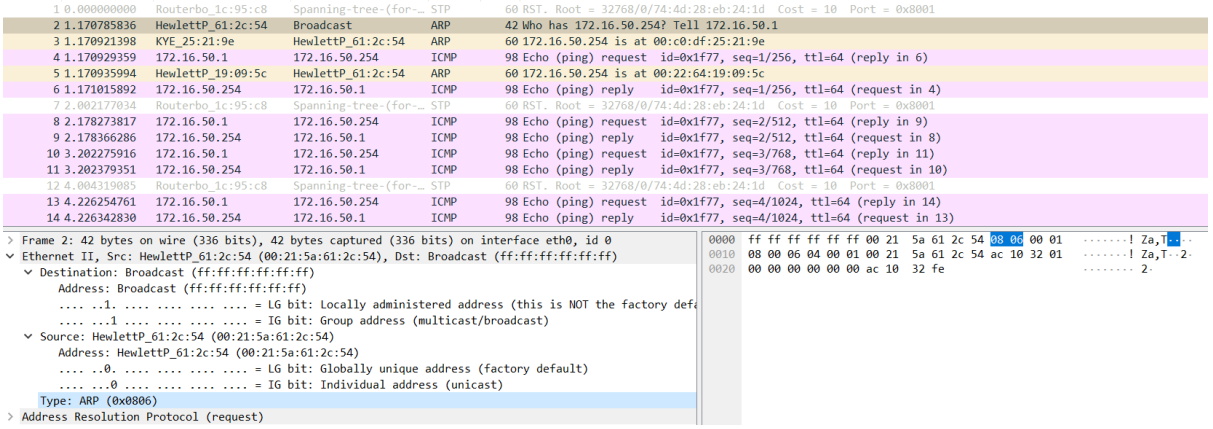


Figura 02: Experiência 1: Ping de *tux4* em *tux3*, análise de uma trama ARP *(exp1_step8.pcapng)*

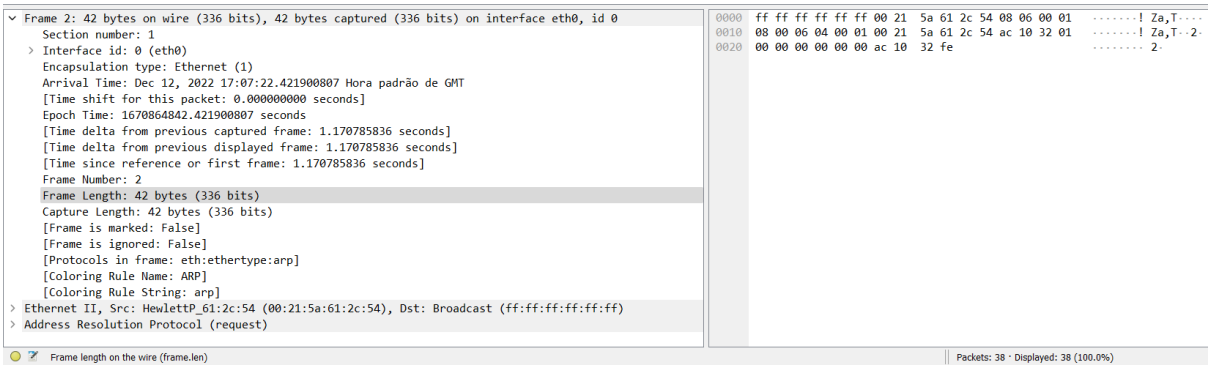


Figura 03: Experiência 1: Determinação do tamanho de uma trama recebida *(exp1_step8.pcapng)*

Experiência 2

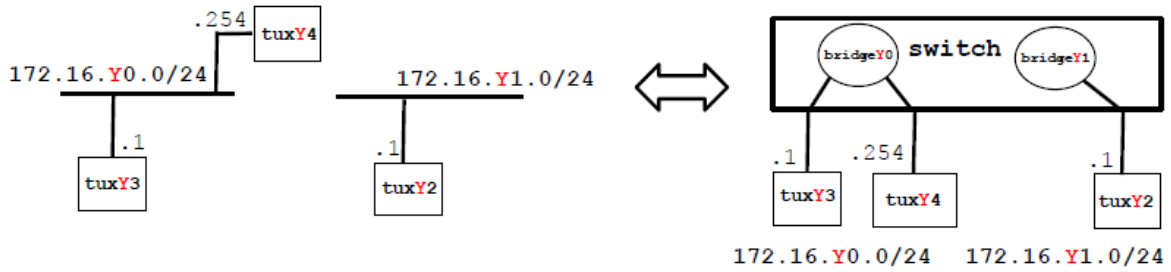


Figura 04: Experiência 2: Arquitetura da configuração

0.000000000	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
0.073837357	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x20d3, seq=1/256, ttl=64 (reply in 3)
0.073999039	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x20d3, seq=1/256, ttl=64 (request in 2)
1.098217573	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x20d3, seq=2/512, ttl=64 (reply in 5)
1.098348246	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x20d3, seq=2/512, ttl=64 (request in 4)
2.002105493	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
2.122193271	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x20d3, seq=3/768, ttl=64 (reply in 8)
2.122348807	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x20d3, seq=3/768, ttl=64 (request in 7)
3.146190271	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x20d3, seq=4/1024, ttl=64 (reply in 10)
3.146317242	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x20d3, seq=4/1024, ttl=64 (request in 9)
4.004203447	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001

Figura 05: Experiência 2: Ping de *tux4* em *tux3*
(exp2_step5_tux3_tux4.pcapng)

2.002187490	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
4.004355010	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
6.006541878	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
7.998696641	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001

Figura 06: Experiência 2: Ping de *tux2* em *tux3*
(exp2_step5_tux3_tux2.pcapng)

34.606941200	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x218c, seq=1/256, ttl=64 (no response found!)
34.607134241	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x218c, seq=1/256, ttl=64
35.616641386	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x218c, seq=2/512, ttl=64 (no response found!)
35.616831423	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x218c, seq=2/512, ttl=64
36.018217152	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
36.640641739	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x218c, seq=3/768, ttl=64 (no response found!)
36.640819415	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x218c, seq=3/768, ttl=64
37.664641185	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x218c, seq=4/1024, ttl=64 (no response found!)
37.664824518	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x218c, seq=4/1024, ttl=64
38.020297362	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001

Figura 07: Experiência 2: Ping broadcast no tux3
(exp2_step7_tux3.*pcapng*)

3.994212487	Routerbo_1c:95:db	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:db Cost = 0 Port = 0x8001
5.996307848	Routerbo_1c:95:db	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:db Cost = 0 Port = 0x8001
7.998213652	Routerbo_1c:95:db	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:db Cost = 0 Port = 0x8001
10.000539837	Routerbo_1c:95:db	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:db Cost = 0 Port = 0x8001
12.002645525	Routerbo_1c:95:db	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:db Cost = 0 Port = 0x8001
13.994760113	Routerbo_1c:95:db	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:db Cost = 0 Port = 0x8001

Figura 08: Experiência 2: Ping broadcast no tux2
(exp2_step7_tux2.*pcapng*)

Experiência 3

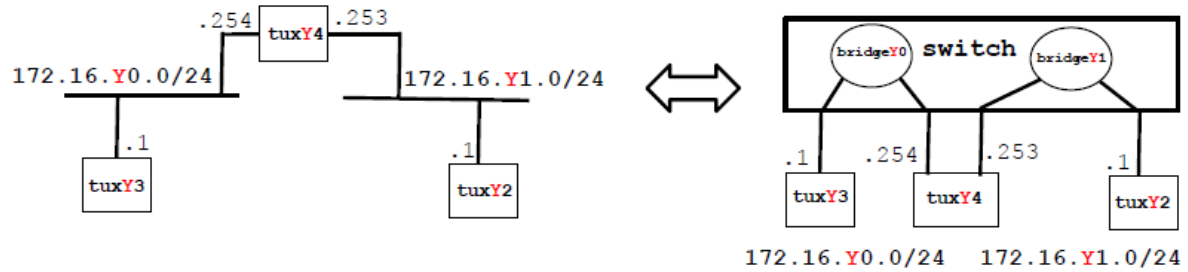


Figura 09: Experiência 3: Arquitetura da configuração

Time	Source	Destination	Protocol	Length	Info
6.006502120	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
6.393800491	HewlettP_61:2c:54	HewlettP_19:09:5c	ARP	42	Who has 172.16.50.254? Tell 172.16.50.1
6.393923621	HewlettP_19:09:5c	HewlettP_61:2c:54	ARP	60	172.16.50.254 is at 00:22:64:19:09:5c
6.457828344	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x231c, seq=7/1792, ttl=64 (reply in 20)
6.458072229	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x231c, seq=7/1792, ttl=63 (request in 19)
7.481832871	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x231c, seq=8/2048, ttl=64 (reply in 22)
7.482085975	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x231c, seq=8/2048, ttl=63 (request in 21)
8.008670496	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
8.505833906	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x231c, seq=9/2304, ttl=64 (reply in 25)
8.506148680	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x231c, seq=9/2304, ttl=63 (request in 24)
9.529831031	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x231c, seq=10/2560, ttl=64 (reply in 27)
9.530089652	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x231c, seq=10/2560, ttl=63 (request in 26)
10.010830073	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
10.553827038	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x231c, seq=11/2816, ttl=64 (reply in 30)
10.554089361	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x231c, seq=11/2816, ttl=63 (request in 29)
11.577833452	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x231c, seq=12/3072, ttl=64 (reply in 32)
11.578091026	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x231c, seq=12/3072, ttl=63 (request in 31)
12.013006344	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001

Figura 10: Experiência 3: Ping de *tux2* em *tux3* (exp3_step6_tux3_tux2.pcapng)

Experiência 4

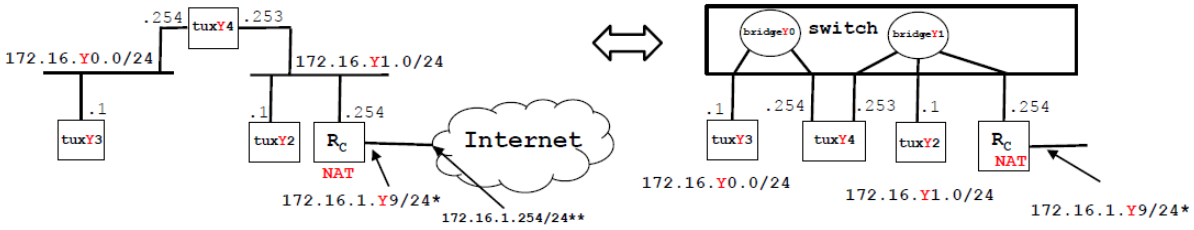


Figura 11: Experiência 4: Arquitetura da configuração

0.000000000	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
2.011142363	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
2.224259890	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x1c18, seq=1/256, ttl=64 (reply in 4)
2.224429952	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x1c18, seq=1/256, ttl=64 (request in 3)
3.233626583	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x1c18, seq=2/512, ttl=64 (reply in 6)
3.233761725	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x1c18, seq=2/512, ttl=64 (request in 5)
4.013247864	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
4.257633162	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x1c18, seq=3/768, ttl=64 (reply in 9)
4.257764323	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x1c18, seq=3/768, ttl=64 (request in 8)
5.281625922	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x1c18, seq=4/1024, ttl=64 (reply in 11)
5.281787882	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x1c18, seq=4/1024, ttl=64 (request in 10)
6.015343068	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001

Figura 12: Experiência 4: Ping de eth0 do *tux4* em *tux3*
(exp4_step3_tux3_tux4_eth0.pcapng)

0.000000000	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
2.002148934	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
2.259412051	172.16.50.1	172.16.51.253	ICMP	98	Echo (ping) request id=0x1d80, seq=1/256, ttl=64 (reply in 4)
2.259583231	172.16.51.253	172.16.50.1	ICMP	98	Echo (ping) reply id=0x1d80, seq=1/256, ttl=64 (request in 3)
3.278034564	172.16.50.1	172.16.51.253	ICMP	98	Echo (ping) request id=0x1d80, seq=2/512, ttl=64 (reply in 6)
3.278168868	172.16.51.253	172.16.50.1	ICMP	98	Echo (ping) reply id=0x1d80, seq=2/512, ttl=64 (request in 5)
4.004264581	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
4.302035248	172.16.50.1	172.16.51.253	ICMP	98	Echo (ping) request id=0x1d80, seq=3/768, ttl=64 (reply in 9)
4.302172205	172.16.51.253	172.16.50.1	ICMP	98	Echo (ping) reply id=0x1d80, seq=3/768, ttl=64 (request in 8)
5.326033424	172.16.50.1	172.16.51.253	ICMP	98	Echo (ping) request id=0x1d80, seq=4/1024, ttl=64 (reply in 11)
5.326167658	172.16.51.253	172.16.50.1	ICMP	98	Echo (ping) reply id=0x1d80, seq=4/1024, ttl=64 (request in 10)
6.006402815	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001

Figura 13: Experiência 4: Ping de eth1 do *tux4* em *tux3*
(exp4_step3_tux3_tux4_eth1.pcapng)

0.000000000	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
0.641870045	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x1da9, seq=1/256, ttl=64 (reply in 3)
0.642195363	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x1da9, seq=1/256, ttl=63 (request in 2)
1.668853349	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x1da9, seq=2/512, ttl=64 (reply in 5)
1.669106941	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x1da9, seq=2/512, ttl=63 (request in 4)
2.002124783	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
2.692852010	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x1da9, seq=3/768, ttl=64 (reply in 8)
2.693169926	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x1da9, seq=3/768, ttl=63 (request in 7)
3.716843205	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x1da9, seq=4/1024, ttl=64 (reply in 10)
3.717087159	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x1da9, seq=4/1024, ttl=63 (request in 9)
4.004195605	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001

Figura 14: Experiência 4: Ping de *tux2* em *tux3*
(exp4_step3_tux3_tux2.pcapng)

5.890389605	HewlettP_19:09:5c	HewlettP_61:2c:54	ARP	60	Who has 172.16.50.1? Tell 172.16.50.254
5.890409510	HewlettP_61:2c:54	HewlettP_19:09:5c	ARP	42	172.16.50.1 is at 00:21:5a:61:2c:54
5.934877762	HewlettP_61:2c:54	HewlettP_19:09:5c	ARP	42	Who has 172.16.50.254? Tell 172.16.50.1
5.934912962	172.16.50.1	172.16.2.59	ICMP	98	Echo (ping) request id=0x1dd8, seq=6/1536, ttl=64 (reply in 19)
5.934999704	HewlettP_19:09:5c	HewlettP_61:2c:54	ARP	60	172.16.50.254 is at 00:22:64:19:09:5c
5.935143018	172.16.2.59	172.16.50.1	ICMP	98	Echo (ping) reply id=0x1dd8, seq=6/1536, ttl=63 (request in 17)
6.006323944	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
6.958913021	172.16.50.1	172.16.2.59	ICMP	98	Echo (ping) request id=0x1dd8, seq=7/1792, ttl=64 (reply in 22)
6.959220670	172.16.2.59	172.16.50.1	ICMP	98	Echo (ping) reply id=0x1dd8, seq=7/1792, ttl=63 (request in 21)
7.982912736	172.16.50.1	172.16.2.59	ICMP	98	Echo (ping) request id=0x1dd8, seq=8/2048, ttl=64 (reply in 24)
7.983166538	172.16.2.59	172.16.50.1	ICMP	98	Echo (ping) reply id=0x1dd8, seq=8/2048, ttl=63 (request in 23)

Figura 15: Experiência 4: Ping do *router* do laboratório em *tux3*
(exp4_step3_tux3_router_out.pcapng)

Experiência 5

2	1.464130…	172.16.50.1	172.16.2.1	DNS	74	Standard query 0x81c1 A www.google.com
3	1.464140…	172.16.50.1	172.16.2.1	DNS	74	Standard query 0x69ca AAAA www.google.com
4	1.464843…	172.16.2.1	172.16.50.1	DNS	90	Standard query response 0x81c1 A www.google.com A 216.58.215.164
5	1.464865…	172.16.2.1	172.16.50.1	DNS	102	Standard query response 0x69ca AAAA www.google.com AAAA 2a00:1450:4003:806::2004

Figura 16: Ping de *www.google.com* em *tux3*
(*exp5_google_tux3.pcapng*)

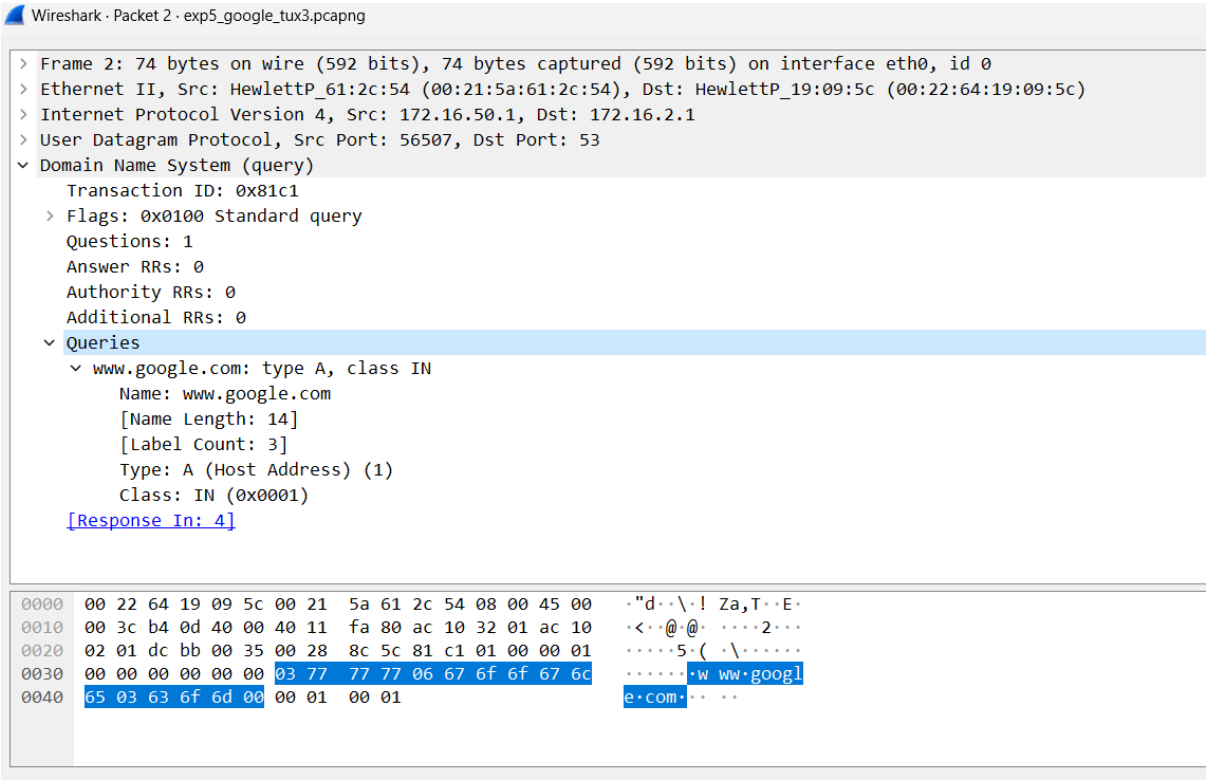


Figura 17: Pedido de tradução do nome de domínio www.google.com para o seu endereço IP (*exp5_google_tux3.pcapng*)

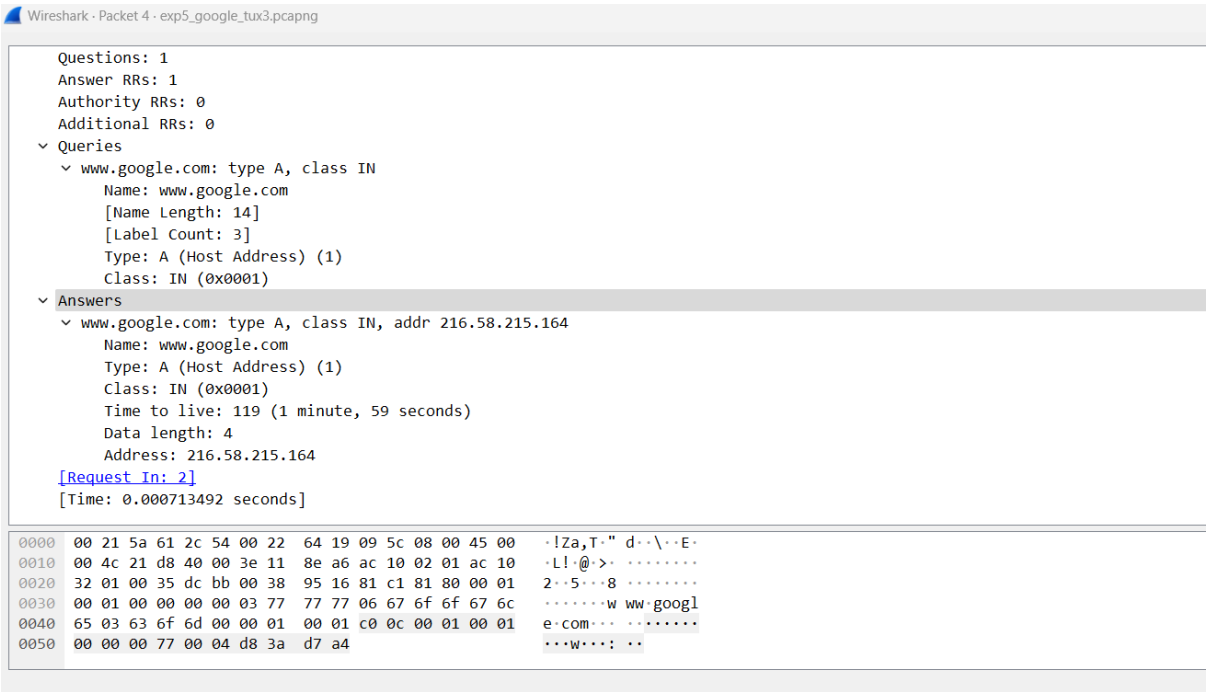
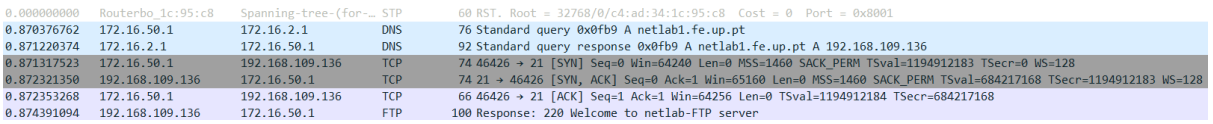


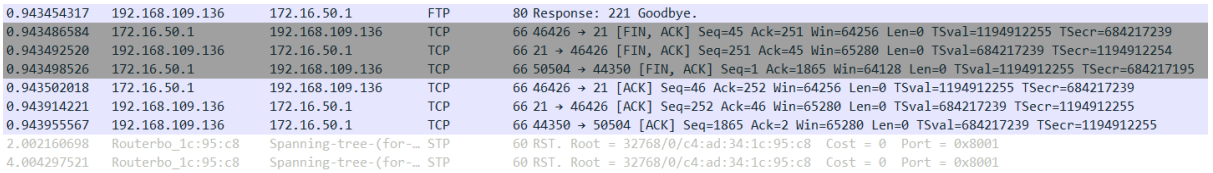
Figura 18: Resposta com o respetivo endereço IP (*exp5_google_tux3.pcapng*)

Experiência 6



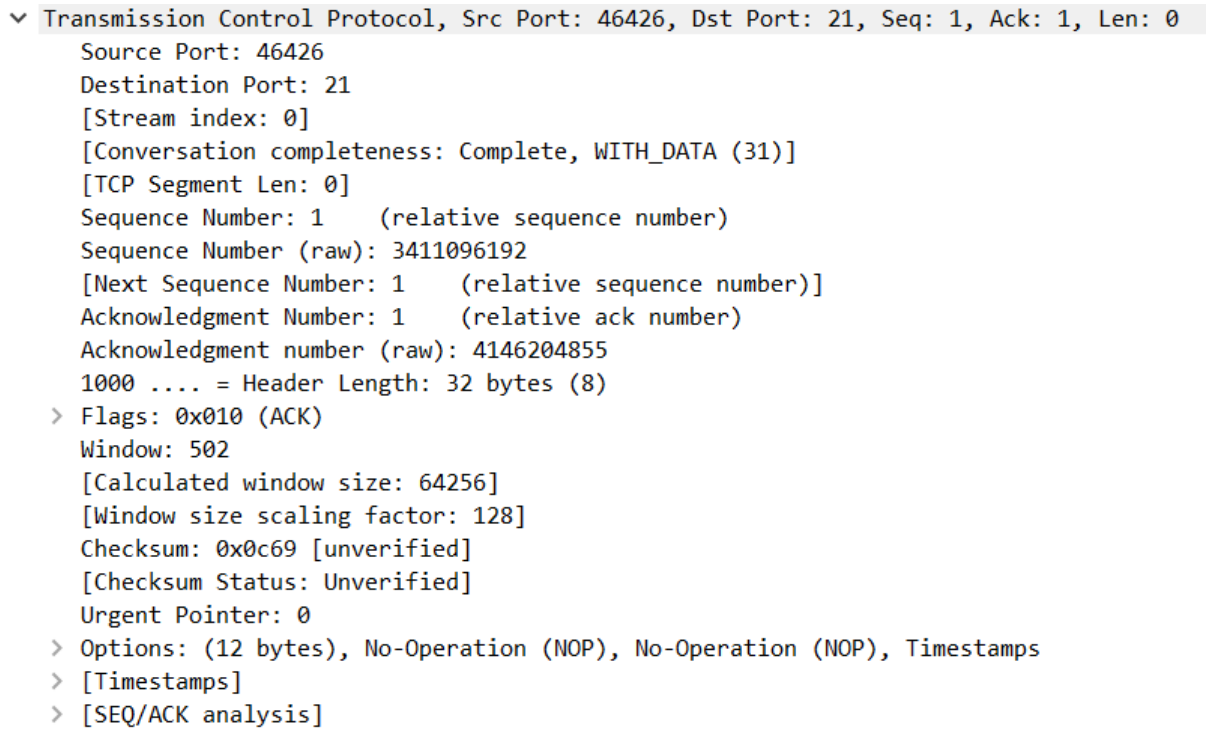
0.000000000	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
0.870376762	172.16.50.1	172.16.2.1	DNS	76	Standard query 0x0fb9 A netlab1.fe.up.pt
0.871220374	172.16.2.1	172.16.50.1	DNS	92	Standard query response 0x0fb9 A netlab1.fe.up.pt A 192.168.109.136
0.871317523	172.16.50.1	192.168.109.136	TCP	74	46426 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1194912183 TSecr=0 WS=128
0.872321350	192.168.109.136	172.16.50.1	TCP	74	21 → 46426 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=684217168 TSecr=1194912183 WS=128
0.872353268	172.16.50.1	192.168.109.136	TCP	66	46426 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1194912184 TSecr=684217168
0.874391094	192.168.109.136	172.16.50.1	FTP	100	Response: 220 Welcome to netlab-FTP server

Figura 19: Conexão TCP para a transferência de um ficheiro no *tux3* (*exp6_step2.pcapng*)



0.943454317	192.168.109.136	172.16.50.1	FTP	80	Response: 221 Goodbye.
0.943486584	172.16.50.1	192.168.109.136	TCP	66	46426 → 21 [FIN, ACK] Seq=45 Ack=251 Win=64256 Len=0 TSval=1194912255 TSecr=684217239
0.943492520	192.168.109.136	172.16.50.1	TCP	66	21 → 46426 [FIN, ACK] Seq=251 Ack=45 Win=65280 Len=0 TSval=684217239 TSecr=1194912254
0.943498526	172.16.50.1	192.168.109.136	TCP	66	50504 → 44350 [FIN, ACK] Seq=1 Ack=1865 Win=64128 Len=0 TSval=1194912255 TSecr=684217195
0.943502018	172.16.50.1	192.168.109.136	TCP	66	46426 → 21 [ACK] Seq=46 Ack=252 Win=64256 Len=0 TSval=1194912255 TSecr=684217239
0.943914221	192.168.109.136	172.16.50.1	TCP	66	21 → 46426 [ACK] Seq=252 Ack=46 Win=65280 Len=0 TSval=684217239 TSecr=1194912255
0.943955567	192.168.109.136	172.16.50.1	TCP	66	44350 → 50504 [ACK] Seq=1865 Ack=2 Win=65280 Len=0 TSval=684217239 TSecr=1194912255
2.002160698	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
4.004297521	Routerbo_1c:95:c8	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001

Figura 20: Desconexão TCP após a transferência de um ficheiro no *tux3* (*exp6_step2.pcapng*)



```
▾ Transmission Control Protocol, Src Port: 46426, Dst Port: 21, Seq: 1, Ack: 1, Len: 0
    Source Port: 46426
    Destination Port: 21
    [Stream index: 0]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 3411096192
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 4146204855
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x010 (ACK)
    Window: 502
    [Calculated window size: 64256]
    [Window size scaling factor: 128]
    Checksum: 0x0c69 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [Timestamps]
  ▸ [SEQ/ACK analysis]
```

Figura 21: Constituição de trama TCP durante a transferência de um ficheiro no *tux3* (*exp6_step2.pcapng*)

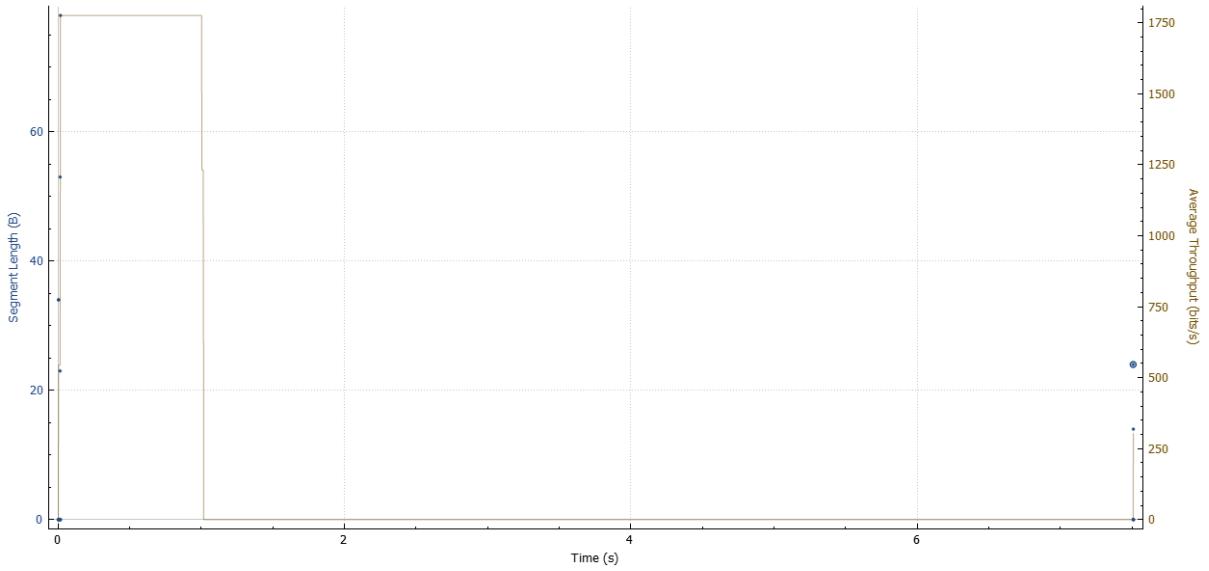


Figura 22: Gráfico da transferência no *tux3* de dois ficheiros em simultâneo iniciada no *tux3*

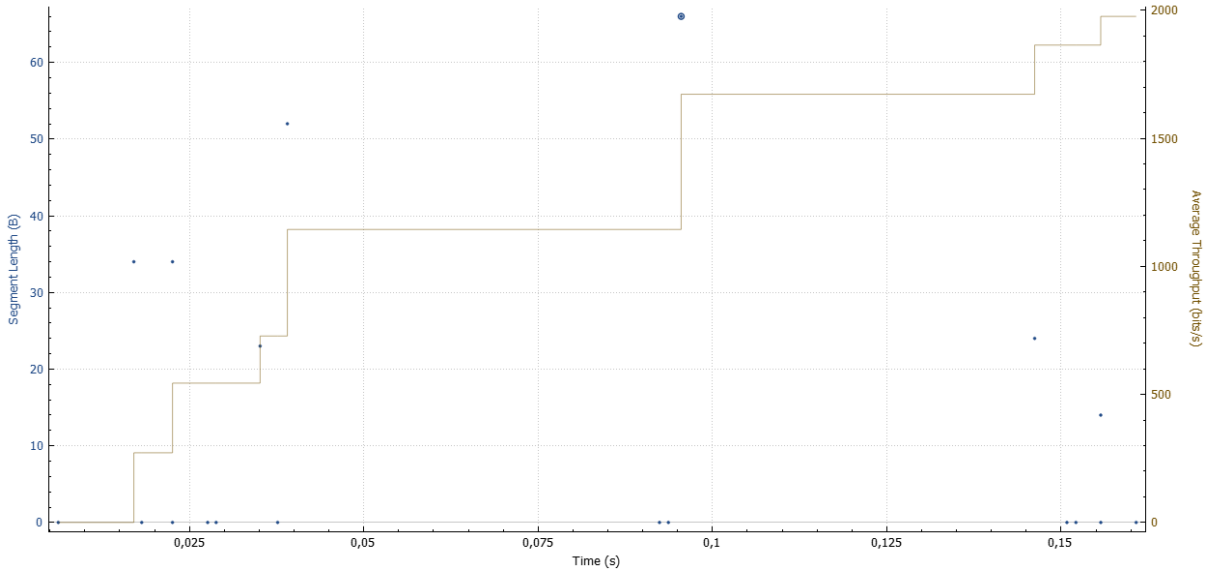


Figura 23: Gráfico da transferência no *tux2* de dois ficheiros em simultâneo iniciada no *tux3*