Redes de Computadores

Turma 1 Grupo 5

Diogo Samuel Gonçalves Fernandes up201806250@fe.up.pt
Paulo Jorge Salgado Marinho Ribeiro up201806505@fe.up.pt

22 de dezembro de 2020

Projeto RCOM - 2019/20 - MIEIC

Professor: Rui Campos rcampos@fe.up.pt

1 Introdução

Este trabalho consiste no desenvolvimento de uma aplicação de download via ftp e na criação de uma rede. O trabalho está portanto, dividido em duas partes distintas:

- Parte 1 - Aplicação de download FTP
- Parte 2 - Configuração e estudo de uma rede

2 Aplicação de download FTP

A primeira parte deste segundo projeto consiste no desenvolvimento de uma aplicação de download, que permite transferir um ficheiro de qualquer tipo, de um dado servidor FTP. Após compilar o código recorrendo ao comando "make", o utilizador deve escrever na consola o seguinte comando, para correr o programa:

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

O campo <user> deverá conter o username com que o utilizador deseja entrar no servidor, e <password> a respetiva password. No caso de desejar entrar de forma anónima, o utilizador pode introduzir o username "anonymous"e qualquer password ou pode omitir os campos do <user>:<password>, ficando o input na forma:

./download ftp://<host>/<url-path>

O campo [host] indicará o endereço do servidor FTP ao qual se deseja conectar, e [url-path] o caminho para o ficheiro que se pretende transferir.

2.1 Arquitetura

A estrutura principal do programa encontra-se bem explícita no ficheiro clientTCP.c. O programa começa por processar o argumento introduzido pelo utilizador, armazenando o seu username, password, o host, e o path para o ficheiro, recorrendo à função parseArguments() que se encontra definida no ficheiro utils.c. Se o input recebido for inválido, o programa termina e é apresentada uma mensagem indicando a correta utilização do programa.

De seguida, é processado o campo host, obtendo-se o endereço IP correspondente, com recurso à função getIP(). Este endereço IP é utilizado logo na conexão ao servidor, após criação de um socket que será utilizado para troca de comandos entre o cliente e o servidor.

Após conectar este socket ao servidor desejado, é lida a resposta do servidor a qual se espera que contenha o código 220, que indica que a conexão foi estabelecida e que o servidor espera pelo login de um novo utilizador.

Assim, o próximo passo será efetuar o login (função login()), que consiste numa troca de mensagens entre o cliente e o servidor, estabelecida da seguinte forma:

- Envio do comando "user [user] newline", em que [user] é o username recebido como input
- Receção da resposta ao comando user. Se o primeiro dígito do código recebido for 2, então não é requerida password, e o login é efetuado com sucesso. Se esse dígito for 3, então é necessária uma password, e os próximos passos são efetuados.

- Envio do comando "pass [password] newline", em que [password] é a password recebida como input
- Receção da resposta ao comando pass. Se o código recebido for 230, então a password foi aceite e o login foi efetuado com sucesso. Caso contrário, o programa termina acusando erro no login.

Após sucesso no login, é necessário pedir ao servidor para transferir dados em modo passivo. Isto é efetuado na função activatePassiveMode(), que começa por enviar o comando pasv newline para o servidor. Segue-se uma máquina de estados, que vai receber a resposta do servidor a este comando, e que vai armazenar os valores retornados, utilizando-os para calcular a porta para a qual serão enviados os dados.

Após isto, é efetuada a criação de um novo socket e a sua conexão ao servidor, pela porta resultante do passo anterior, de onde serão lidos os dados do ficheiro.

Já com tudo configurado, é efetuada a transferência do ficheiro, na função download_file(), que começa por mandar o comando "retr [path] newline"para pedir o ficheiro desejado. Segue-se a leitura da resposta do servidor face a este comando, a qual se espera ser o código 150, que indica que o ficheiro está pronto para download e o pedido foi aceite. Assim, pode-se começar a ler a informação do ficheiro, do socket aberto para leitura dos dados, e enviar a informação para um ficheiro criado imediatamente antes, cujo nome é obtido aplicando a função basename() ao path recebido como input. Se não tiver ocorrido nenhum erro durante o processo, é apresentada uma mensagem de sucesso, que indica que o ficheiro foi transferido.

Por último, o programa fecha os dois sockets abertos.

2.2 Resultados

O nosso programa foi testado para diversos casos, nomeadamente a utilização de diferentes servidores FTP, diferentes logins (introdução de username e password e entrada em modo anónimo), e a utilização de diferentes tipos e tamanhos, nos ficheiros transferidos. Para maior compreensão do processo, são imprimidos na consola todos os passos efetuados, assim como as respetivas respostas do servidor. Concluímos todos os requisitos desta primeira parte com sucesso, pelo que a aplicação encontra-se totalmente funcional e de acordo com o especificado.

3 Configuração e estudo de uma rede

Experiência 1 - Configurar uma rede IP

Nesta primeira experiência foi ligado o GNU3 ao GNU64, ligando ambos os GNU ao switch e recorrendo à configuração dos seus endereços IP para que pudessem comunicar entre si.

```
# No GNU63
ifconfig eth0 172.16.60.1/24

# No GNU64
ifconfig eth0 172.16.60.254/24
```

Esta conexão foi testada recorrendo ao comando ping, e uma vez que foi recebida resposta foi possível confirmar que a mesma estava bem configurada. Através da **figura 1** é possível verificar que se obteve uma resposta do GNU64 (pacotes número 26 e 28) ao ping efetuado a partir do GNU63 (pacotes números 25 e 27).

É possível verificar também que o primeiro pacote trocado é um pacote do tipo ARP. O ARP (Address Resolution Protocol) é um protocolo utilizado para obter o endereço MAC associado a um dado endereço de IP. Para o envio de uma trama para um dado computador presente na rede, o emissor necessita do endereço MAC correspondente ao endereço IP de destino. Para isto, é enviado um pacote ARP em modo Broadcast, que contém esse IP e que espera o retorno do endereço MAC desejado. Isto pode ser observado na **figura 1**, em que é enviado um pacote ARP em broadcast. Além de pacotes ARP, são trocados pacotes do tipo ICMP.

É também possível verificar que os IPs de origem e destino dos pacotes ping são os IPs e MACs do GNU63 e do GNU64, respetivamente no caso das requests. No caso das replies, o IP e MAC da origem vai pertencer ao GNU64 e o de destino ao GNU63.

A distinção entre tramas Ethernet do tipo ARP, IP ou ICMP pode ser feita analisando o cabeçalho dessa trama, que terá valores distintos conforme o tipo de trama. Por sua vez, o comprimento das tramas pode ser obtido no wireshark, como consta na **figura 24**, uma vez que essa informação está também presente no cabeçalho da trama.

Por último, a interface loopback é responsável por realizar o diagnóstico de problemas e testes de conectividade. É o método mais usado para determinar se um dispositivo está online. Esta interface é responsável pelos pacotes que podem ser observados nas **figuras 25 e 26**.

Experiência 2 - Implementar duas LAN virtuais num switch

Nesta experiência, foram criadas duas LANs virtuais, ficando o GNU63 e o GNU64 conectados à VLAN60 e o GNU62 conectado à VLAN61. O GNU62 uma vez que se vai encontrar numa rede diferente do GNU63 e GNU64 não consegue comunicar com os mesmos.

Para configurar esta rede foi necessário ligar o GNU62 ao switch e posteriormente foi necessário criar as VLANs e atribuir as respetivas portas. As VLANs foram criadas através do switch sendo utilizado o seguinte comando:

```
conf t
vlan <y>
end
```

Em que <y > representa o número da VLAN.

Seguiu-se a configuração das VLANs, onde foi necessário adicionar as portas do switch às respetivas VLANs, recorrendo ao seguinte comandos:

```
conf t
interface fastethernet 0/<n>
switchport mode access
switchport por access vlan <y>
end
```

Nesta caso está a ser adicionada a porta <n> do switch à vlan<y>. No nosso caso, o valor de y foi 60 e 61, como mencionado previamente.

Após adicionar as duas VLAN e adicionar respetivas portas a cada uma, passam a existir dois domínios de transmissão, sendo que um deles contém o GNU63 e GNU64 e o outro contém o GNU62.

Deste modo, quando é feito ping em modo de broadcast a partir do GNU63, o GNU64 como se encontra na mesma VLAN e portanto no mesmo domínio de transmissão irá conseguir receber esses pacotes como é possível constatar com as **figuras 4 e 5**, enquanto que o GNU62 não recebe qualquer pacote uma vez que se encontra noutra VLAN. Da mesma forma, quando é feito ping em modo de broadcast a partir do GNU62, quer o GNU63 e GNU64 não irão receber qualquer pacote, como se pode verificar nas **figuras 7 e 8**, uma vez que estes dois se encontram num domínio de transmissão diferente.

Experiência 3 - Configurar Router em Linux

Nesta experiência, o GNU64 foi configurado de modo a funcionar como um router, estabelecendo assim a ligação entre as duas VLANs criadas anteriormente.

Para isto, configurou-se a porta ETH1 do GNU64 com um IP no mesmo domínio que o GNU62. Depois, foi necessário configurar as rotas com o comando <route add>. No GNU63 adicionou-se a rota que redireciona os pacotes que têm como destino a VLAN61 para o GNU64, e o mesmo foi realizado para o GNU62, em que os pacotes que têm como destino a VLAN60 são redirecionados para o GNU64. Isto pode ser verificado na **figura 11** quando a partir do GNU63 foi efetuado o comando para dar ping no GNU62 este obteve resposta, indicando que os computadores conseguem comunicar entre si.

É também possível verificar a forma como o GNU64 opera através da análise das **figuras 12 e 13**. Na **figura 12** é possível observar o tráfego de pacotes através da ETH0, que corresponde à VLAN60, enquando que na **figura 13** é possível observar relativamente aos pacotes que são trocados na ETH1, correspondente à VLAN61. Os pacotes recebidos a partir da ETH0 correspondente à VLAN60 provêm do endereço 172.16.60.1 e têm o destino de 172.16.61.1 são reencaminhados para a ETH1 correspondente à VLAN61 Podemos também realçar que o primeiro pacote trocado é do tipo ARP, uma vez que antes da captura destes logs foram eliminadas as tabelas ARP dos três GNUs e estes pacotes são necessários para ser possivel a comunicação.

Após isto, torna-se possível realizar ping do GNU63 para o GNU62 e vice-versa, uma vez que os pedidos são encaminhados para o GNU64 e este por sua vez consegue comunicar com os outros dois PCs.

A tabela de forwarding define a forma como uma trama será encaminhada de um switch ou router na rede. Contém o destino da rota, a "gateway", que corresponde ao próximo ponto por onde a rota passará, a máscara (netmask), usada para determinar o ID da rede a partir do endereço IP de destino, informações e custo da rota, o número de referências para a rota, um contador de pesquisas da rota, e a placa de rede responsável pela gateway.

Sempre que um GNU dá ping a outro e o GNU que recebeu o pedido não conhece o endereço MAC do emissor, é enviado um pacote ARP a pedir esta informação, como mostrado na experiência 1, que conterá os endereços dos GNU de origem e de destino. Os pacotes ICMP observados resultam do comando ping, e tratam-se de pacotes de request e reply, uma vez que, após a configuração desta experiência, todos os PCs conseguem comunicar entre si. No caso de não conseguirem comunicar entre si, seriam enviados pacotes ICMP do tipo Host Unreachable. Os endereços IP e MAC dos pacotes ICMP são os dos PCs de origem e destino.

Experiência 4 - Configurar um Router comercial e implementar NAT

O principal objetivo desta experiência consiste na configuração do router com NAT.

Para a configuração, foi necessário ligar a entrada FE0 do router ao switch. A porta a que o router se encontra ligado ao switch tem de pertencer à VLAN61 e tal pode ser efetuado como foi mencionado no experiência 2, adicionando a porta à respetiva VLAN. Após o router estar ligado à VLAN61, é necessário ligar a entrada FE1 do router à primeira entrada da régua 1 (6.1) que irá permitir a ligação à internet.

Foi necessário adicionar rotas estáticas para permitir que a conexão com a internet, quer a conexão com a VLAN60. Os comandos que permitem a criação destas rotas estáticas são os seguintes:

```
ip route 0.0.0.0 0.0.0.0 172.16.2.254
ip route 172.16.60.0 255.255.255.0 172.16.61.253
```

Posteriormente foi pedido para ser retirado o redirecionamento ICMP no GNU62 e a remover a rota para o GNU64. Ao ser feito o ping para o GNU63, os pacotes enviados são direcionados para a rota default do router, sendo posteriormente enviadas para o GNU64 e a partir deste chega ao GNU63. Neste caso é possível observar como mostra a **figura 14** que o router vai redirecionar os pacotes que chegam até ele.

Após adicionar novamente a rota é possível observar que os pacotes vão diretamente para o GNU64 e após esse seguem para o GNU63. Tal permite concluir que os pacotes enviados durante esta seguem uma dada rota, no caso de esta já estar definida. Caso contrário, estes são direcionados para a rota default do router.

Foi efetuada uma última experiência que consistia em voltar a ativar o redirecionamento ICMP e remover a rota para o GNU64. Ao ser efetuado o comando de ping do GNU62 para o GNU63, o router envia apenas um pacote para o ICMP Redirect. Como o redirecionamento está ativo, o GNU62 guarda a rota na tabela de reencaminhamento o que permite a este computador estabelecer a ligação com o GNU64. Nesta situação foi observado apenas um pacote do ICMP Redirect ao contrário do que acontecia anteriormente como mostra a **figura 14** em que a cada ping era recebido um ICMP Redirect.

Após estas experiências, quando foi efetuado ping no router a partir do GNU63 verificamos que não foi obtida qualquer resposta pelo que tivemos de implementar o NAT (Network Address Translation). Para a configuração do NAT foi necessária a configuração da interface interna e externa do router, assim como criar uma lista de acessos, sendo este o motivo do GNU64 não conseguir ter acesso à internet. Os comandos a serem executados no router podem ser vistos nos anexos deste relatório.

O NAT é uma técnica que consiste em reescrever através da utilização de uma tabela hash, os endereços IP que utilizam o router de forma a que um computador de uma rede privada tenha acesso a uma rede pública e tem como objetivo poupar espaço de endereçamento público. Resumidamente, permite que as redes privadas que usam endereços IPs não registados se conetem à Internet ou a uma

rede pública, a partir de um único endereço público. Desta forma, apenas um endereço IP é exigido e este irá representar todos os computadores da rede local.

Após a configuração do NAT foi possível a comunicação com o router como é possível verificar na **figura 16**.

Experiência 5 - DNS

Nesta experiência, foi configurado o DNS (Domain Name Service). Trata-se de um sistema hierárquico e distribuído de gestão de nomes para computadores, serviços ou qualquer equipamento conectado à Internet ou a uma rede privada. Isto é, é um sistema que traduz os hostnames nos respetivos endereços IP, recorrendo a servidores de DNS que contém uma base de dados com esta correspondência entre hostnames e IPs.

Para a sua configuração foi necessário alterar o ficheiro resolv.conf, adicionando o nome do servidor de DNS e o seu endereço IP, de acordo com o slide 14 do guião do projeto.

Quando se faz ping a um servidor externo, é enviado um pacote de DNS com o pedido do IP do servidor. A resposta chega na forma de um outro pacote DNS, que após fazer a conversão, devolverá o endereço IP correspondente. Esta troca de pacotes pode ser observada na **figura 17**.

Experiência 6 - TCP connections

Esta experiência serviu para observação do comportamento do protocolo TCP, recorrendo à aplicação que desenvolvemos na parte 1 deste projeto.

Através da análise das **figura 18** é possível verificar que a aplicação estabelece duas conexões TCP - uma para troca de comandos entre o cliente e o servidor FTP, que trata do transporte do controlo de informação FTP, e outra para a receção dos dados enviados pelo servidor. Cada conexão FTP subdivide-se em três fases: o estabelecimento da conexão, a troca de dados entre o cliente e servidor, e o encerramento da conexão. O mecanismo ARQ (Automatic Repeat Request) do TCP (Transmission Control Protocol) é utilizado com o método da janela deslizante, e consiste no controlo dos erros que ocorram durante a transmissão dos dados. Recorre a ACK (Acknowledgement numbers), que estão incluídos nas tramas enviadas pelo recetor, indicando se esta foi recebida corretamente, sem quaisquer problemas, e recorre também ao "window size", que indica o domínio de pacotes possíveis de ser enviados pelo emissor, e ao "sequence number", que indica o número do pacote a ser enviado. O TCP usa um mecanismo de controlo de congestão end-to-end. Isto significa que o emissor limita ou aumenta a taxa de transferência de dados para conexão em função do congestionamento percebido por ele. A conexão TCP é composta por diversas variáveis, sendo uma delas a janela de congestionamento, que limitará a taxa de envio de pacotes de um dado emissor TCP.

Foi realizada uma segunda experiência em que a meio de uma transferência de um ficheiro no GNU63, era iniciada uma transferência a partir do GNU62. Os gráficos das taxas de transferência obtidos podem ser visualizados nas **figuras 21 e 22**, em que a primeira corresponde à transferência no GNU62 e a segunda à transferência no GNU63. Os gráficos obtidos durante a experiência permitem concluir que no início do primeiro download no GNU63, a taxa de transferência aumentou até atingir um máximo. Quando o segundo download no gnu62 foi iniciado, verificou-se que à medida que a taxa de transferência aumentava no GNU62, ocorria uma igual diminuição no GNU63. Quando a transferência no GNU63 terminou, foi possível observar que a taxa de transferência do GNU62 aumentou, atingiu um máximo e manteve-se constante até ao final da transferência. Assim, com o aparecimento de uma segunda conexão TCP, dá-se uma diminuição da taxa de transferência e conclui-se que o fluxo dos dados está de acordo com o mecanismo de controlo de congestão, tendo em conta que se observa uma

menor taxa de transferência no momento em que a rede estava mais congestionada (mais downloads simultâneos).

4 Conclusões

Este segundo projeto da unidade curricular Redes de Computadores teve como objetivo, numa primeira parte, a implementação de uma aplicação de download de ficheiros recorrendo ao protocolo FTP, e, numa segunda parte, à configuração de uma rede de computadores. Apesar de dificultado pelo reduzido tempo que tivemos para aproveitar o laboratório, devido à situação pandémica que atualmente enfrentamos, este projeto foi terminado com sucesso, sendo que conseguimos desenvolver todos os aspetos pedidos. Ao mesmo tempo, interiorizamos conceitos importantes nesta área, e compreendemos os diversos protocolos que foram abordados.

Em suma, conseguimos alcançar todos os nossos objetivos, e conhecemos conceitos dos quais nunca antes tínhamos ouvido falar, mas que utilizávamos com frequência no nosso dia-a-dia.

5 Anexos

5.1 Anexo I - Imagens de experiências

Experiência 1

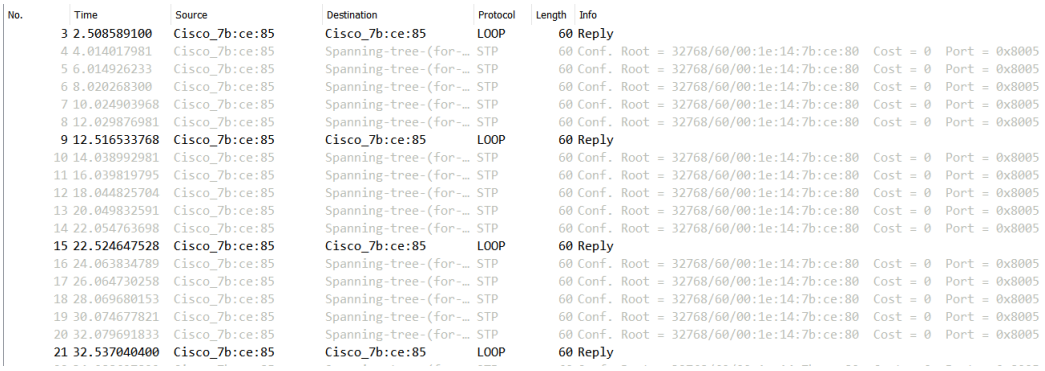
No.	Time	Source	Destination	Protocol	Length	Info
22	30.073637016	Cisco_7b:ce:82	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/1/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8002
23	30.514773341	HewlettP_61:2d:df	Broadcast	ARP	42	Who has 172.16.60.254? Tell 172.16.60.1
24	30.514908833	HewlettP_5a:79:97	HewlettP_61:2d:df	ARP	60	172.16.60.254 is at 00:21:5a:5a:79:97
25	30.514916656	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) request id=0x0954, seq=1/256, ttl=64 (reply in 26)
26	30.515053684	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0954, seq=1/256, ttl=64 (request in 25)
27	31.541194042	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) request id=0x0954, seq=2/512, ttl=64 (reply in 28)
28	31.541324785	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0954, seq=2/512, ttl=64 (request in 27)

Figura 1: Ping gnu64 from gnu63

Experiência 2

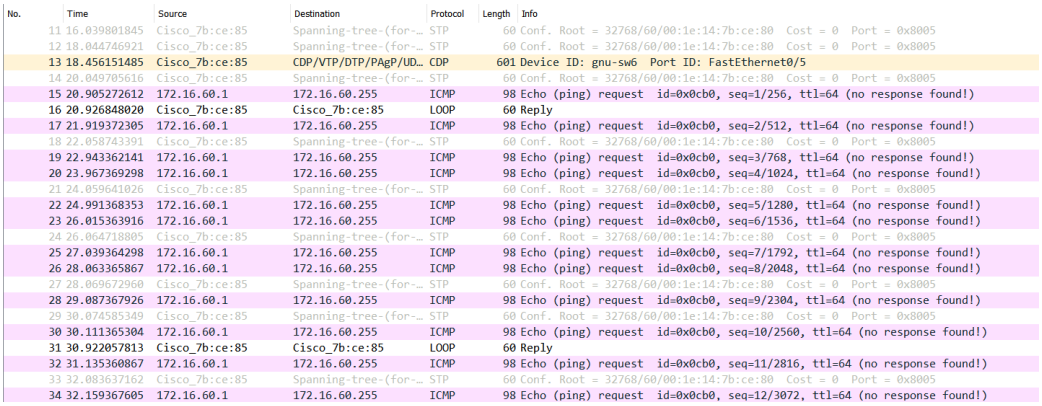
No.	Time	Source	Destination	Protocol	Length	Info
33	32.641106557	HewlettP_5a:79:97	HewlettP_61:2d:df	ARP	60	Who has 172.16.60.1? Tell 172.16.60.254
34	32.641113891	HewlettP_61:2d:df	HewlettP_5a:79:97	ARP	42	172.16.60.1 is at 00:21:5a:61:2d:df
35	32.731095096	HewlettP_61:2d:df	HewlettP_5a:79:97	ARP	42	Who has 172.16.60.254? Tell 172.16.60.1
36	32.731201258	HewlettP_5a:79:97	HewlettP_61:2d:df	ARP	60	172.16.60.254 is at 00:21:5a:5a:79:97
37	33.627134333	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) request id=0x0a60, seq=7/1792, ttl=64 (reply in 38)
38	33.627266128	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0a60, seq=7/1792, ttl=64 (request in 37)
39	34.084686635	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
40	34.651126055	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) request id=0x0a60, seq=8/2048, ttl=64 (reply in 41)
41	34.651260364	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0a60, seq=8/2048, ttl=64 (request in 40)
42	35.675130418	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) request id=0x0a60, seq=9/2304, ttl=64 (reply in 43)
43	35.675291058	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0a60, seq=9/2304, ttl=64 (request in 42)

Figura 2: Ping gnu64 from gnu63



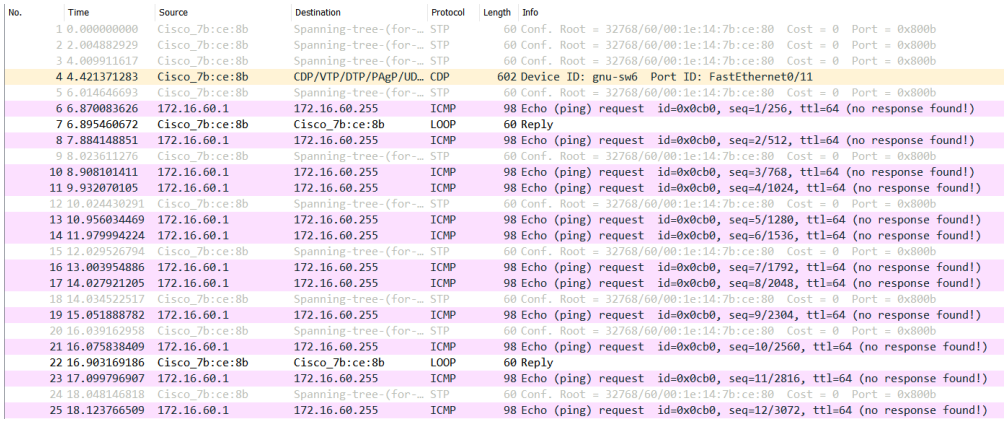
No.	Time	Source	Destination	Protocol	Length	Info
3	2.508589100	Cisco_7b:ce:85	Cisco_7b:ce:85	LOOP	60	Reply
4	4.014017981	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
5	6.014926233	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
6	8.020268300	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
7	10.024903968	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
8	12.029876981	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
9	12.516533768	Cisco_7b:ce:85	Cisco_7b:ce:85	LOOP	60	Reply
10	14.038992981	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
11	16.039819795	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
12	18.044825704	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
13	20.049832591	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
14	22.054763698	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
15	22.524647528	Cisco_7b:ce:85	Cisco_7b:ce:85	LOOP	60	Reply
16	24.063834789	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
17	26.064730258	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
18	28.069680153	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
19	30.074677821	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
20	32.079691833	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
21	32.537040400	Cisco_7b:ce:85	Cisco_7b:ce:85	LOOP	60	Reply

Figura 3: Ping broadcast a partir do gnu63 (ping –b 172.16.60.255) capturado no gnu62



No.	Time	Source	Destination	Protocol	Length	Info
11	16.039801845	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
12	18.044746921	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
13	18.456151485	Cisco_7b:ce:85	CDP/VTP/DTP/PAgP/UD…	CDP	601	Device ID: gnu-sw6 Port ID: FastEthernet0/5
14	20.049705616	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
15	20.905272612	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=1/256, ttl=64 (no response found!)
16	20.926848020	Cisco_7b:ce:85	Cisco_7b:ce:85	LOOP	60	Reply
17	21.919372305	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=2/512, ttl=64 (no response found!)
18	22.058743391	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
19	22.943362141	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=3/768, ttl=64 (no response found!)
20	23.967369298	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=4/1024, ttl=64 (no response found!)
21	24.059641026	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
22	24.991368353	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=5/1280, ttl=64 (no response found!)
23	26.015363916	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=6/1536, ttl=64 (no response found!)
24	26.064718805	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
25	27.039364298	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=7/1792, ttl=64 (no response found!)
26	28.063365867	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=8/2048, ttl=64 (no response found!)
27	28.069672960	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
28	29.087367926	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=9/2304, ttl=64 (no response found!)
29	30.074585349	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
30	30.111365304	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=10/2560, ttl=64 (no response found!)
31	30.922057813	Cisco_7b:ce:85	Cisco_7b:ce:85	LOOP	60	Reply
32	31.135360867	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=11/2816, ttl=64 (no response found!)
33	32.083637162	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
34	32.159367605	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=12/3072, ttl=64 (no response found!)

Figura 4: Ping broadcast a partir do gnu63 (ping –b 172.16.60.255) capturado no gnu63



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
2	2.004882929	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
3	4.009911617	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
4	4.421371283	Cisco_7b:ce:8b	CDP/VTP/DTP/PAgP/UD…	CDP	602	Device ID: gnu-sw6 Port ID: FastEthernet0/11
5	6.014646693	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
6	6.870083626	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=1/256, ttl=64 (no response found!)
7	6.895460672	Cisco_7b:ce:8b	Cisco_7b:ce:8b	LOOP	60	Reply
8	7.884148851	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=2/512, ttl=64 (no response found!)
9	8.023611276	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
10	8.908101411	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=3/768, ttl=64 (no response found!)
11	9.932070105	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=4/1024, ttl=64 (no response found!)
12	10.024430291	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
13	10.956034469	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=5/1280, ttl=64 (no response found!)
14	11.979994224	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=6/1536, ttl=64 (no response found!)
15	12.029526794	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
16	13.003954886	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=7/1792, ttl=64 (no response found!)
17	14.027921205	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=8/2048, ttl=64 (no response found!)
18	14.034522517	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
19	15.051888782	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=9/2304, ttl=64 (no response found!)
20	16.039162958	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
21	16.075838409	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=10/2560, ttl=64 (no response found!)
22	16.903169186	Cisco_7b:ce:8b	Cisco_7b:ce:8b	LOOP	60	Reply
23	17.099796907	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=11/2816, ttl=64 (no response found!)
24	18.048146818	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
25	18.123766509	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x0cb0, seq=12/3072, ttl=64 (no response found!)

Figura 5: Ping broadcast a partir do gnu63 (ping –b 172.16.60.255) capturado no gnu64

No.	Time	Source	Destination	Protocol	Length	Info
7	2.232501575	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x0b41, seq=3/768, ttl=64 (no response found!)
8	3.256504690	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x0b41, seq=4/1024, ttl=64 (no response found!)
9	4.009674767	Cisco_7b:ce:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8003
10	4.280503125	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x0b41, seq=5/1280, ttl=64 (no response found!)
11	5.304501909	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x0b41, seq=6/1536, ttl=64 (no response found!)
12	6.019091862	Cisco_7b:ce:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8003
13	6.328513964	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x0b41, seq=7/1792, ttl=64 (no response found!)
14	7.352502831	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x0b41, seq=8/2048, ttl=64 (no response found!)
15	8.019451572	Cisco_7b:ce:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8003
16	8.376512650	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x0b41, seq=9/2304, ttl=64 (no response found!)
17	9.400506476	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x0b41, seq=10/2560, ttl=64 (no response found!)
18	10.024366130	Cisco_7b:ce:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8003
19	10.424506796	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x0b41, seq=11/2816, ttl=64 (no response found!)

Figura 6: Ping broadcast a partir do gnu62 (ping –b 172.16.61.255) capturado no gnu62

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Cisco_7b:ce:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8003
2	2.005032799	Cisco_7b:ce:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8003
3	4.014273266	Cisco_7b:ce:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8003
4	6.014706449	Cisco_7b:ce:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8003
5	8.019581826	Cisco_7b:ce:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8003
6	8.799776484	Cisco_7b:ce:83	Cisco_7b:ce:83	LOOP	60	Reply
7	10.024475013	Cisco_7b:ce:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8003
8	12.029387475	Cisco_7b:ce:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8003
9	14.038769999	Cisco_7b:ce:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8003
10	16.039234820	Cisco_7b:ce:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8003
11	18.044067734	Cisco_7b:ce:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8003
12	18.799211320	Cisco_7b:ce:83	Cisco_7b:ce:83	LOOP	60	Reply

Figura 7: Ping broadcast a partir do gnu62 (ping –b 172.16.61.255) capturado no gnu63

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
2	2.005271176	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
3	4.009918322	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
4	6.014855308	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
5	6.514579119	Cisco_7b:ce:8b	Cisco_7b:ce:8b	LOOP	60	Reply
6	8.023870247	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
7	10.024533936	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
8	12.029628344	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
9	14.034401553	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
10	16.039265765	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
11	16.517419568	Cisco_7b:ce:8b	Cisco_7b:ce:8b	LOOP	60	Reply
12	18.048348659	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
13	20.049444943	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
14	22.054045854	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
15	24.058971526	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
16	26.063840208	Cisco_7b:ce:8b	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
17	26.525092743	Cisco_7b:ce:8b	Cisco_7b:ce:8b	LOOP	60	Reply

Figura 8: Ping broadcast a partir do gnu62 (ping –b 172.16.61.255) capturado no gnu64

Experiência 3

No.	Time	Source	Destination	Protocol	Length	Info
30	16.713633449	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) request id=0x19d1, seq=8/2048, ttl=64 (reply in 31)
31	16.713800166	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x19d1, seq=8/2048, ttl=64 (request in 30)
32	17.004778320	Cisco_7b:ce:85	Cisco_7b:ce:85	LOOP	60	Reply
33	17.737634181	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) request id=0x19d1, seq=9/2304, ttl=64 (reply in 34)
34	17.737801805	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x19d1, seq=9/2304, ttl=64 (request in 33)
35	18.048876009	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
36	18.761624366	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) request id=0x19d1, seq=10/2560, ttl=64 (reply in 37)
37	18.761793038	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x19d1, seq=10/2560, ttl=64 (request in 36)

Figura 9: Ping 172.16.60.254 a partir do gnu63

No.	Time	Source	Destination	Protocol	Length	Info
86	57.033097567	Cisco_7b:ce:85	Cisco_7b:ce:85	LOOP	60	Reply
87	57.097631372	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x19f1, seq=2/512, ttl=64 (reply in 88)
88	57.097924086	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x19f1, seq=2/512, ttl=63 (request in 87)
89	58.121604026	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x19f1, seq=3/768, ttl=64 (reply in 90)
90	58.121883610	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x19f1, seq=3/768, ttl=63 (request in 89)

Figura 10: Ping 172.16.61.1 a partir do gnu63

No.	Time	Source	Destination	Protocol	Length	Info
48	32.694060074	172.16.60.1	172.16.61.253	ICMP	98	Echo (ping) request id=0x19e4, seq=1/256, ttl=64 (reply in 49)
49	32.694230213	172.16.61.253	172.16.60.1	ICMP	98	Echo (ping) reply id=0x19e4, seq=1/256, ttl=64 (request in 48)
50	33.705628632	172.16.60.1	172.16.61.253	ICMP	98	Echo (ping) request id=0x19e4, seq=2/512, ttl=64 (reply in 51)
51	33.705798352	172.16.61.253	172.16.60.1	ICMP	98	Echo (ping) reply id=0x19e4, seq=2/512, ttl=64 (request in 50)
52	34.084623708	Cisco_7b:ce:85	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8005
53	34.729627478	172.16.60.1	172.16.61.253	ICMP	98	Echo (ping) request id=0x19e4, seq=3/768, ttl=64 (reply in 54)
54	34.729795871	172.16.61.253	172.16.60.1	ICMP	98	Echo (ping) reply id=0x19e4, seq=3/768, ttl=64 (request in 53)
55	35.753612005	172.16.60.1	172.16.61.253	ICMP	98	Echo (ping) request id=0x19e4, seq=4/1024, ttl=64 (reply in 56)
56	35.753780678	172.16.61.253	172.16.60.1	ICMP	98	Echo (ping) reply id=0x19e4, seq=4/1024, ttl=64 (request in 55)

Figura 11: Ping 172.16.61.253 a partir do gnu63

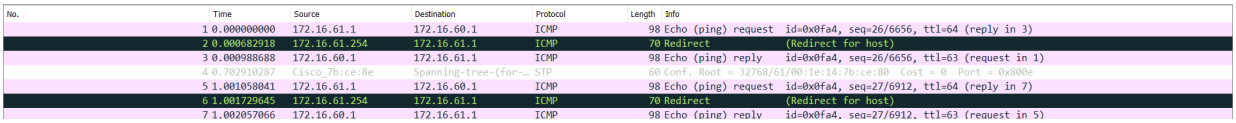
No.	Time	Source	Destination	Protocol	Length	Info
17	25.804178233	Cisco_7b:ce:8b	CDP/VTP/DTP/PAgP/UDLD	CDP	602	Device ID: gnu-sw6 Port ID: FastEthernet0/11
18	26.063863115	Cisco_7b:ce:8b	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
19	28.068775937	Cisco_7b:ce:8b	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
20	30.073511920	Cisco_7b:ce:8b	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
21	30.363332263	Cisco_7b:ce:8b	Cisco_7b:ce:8b	LOOP	60	Reply
22	32.082488307	Cisco_7b:ce:8b	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
23	34.083308649	Cisco_7b:ce:8b	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
24	34.706914172	HewlettP_61:2d:df	Broadcast	ARP	60	Who has 172.16.60.254? Tell 172.16.60.1
25	34.706944832	HewlettP_5a:79:97	HewlettP_61:2d:df	ARP	42	172.16.60.254 is at 00:21:5a:5a:79:97
26	34.707071454	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1b04, seq=1/256, ttl=64 (reply in 27)
27	34.707357663	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1b04, seq=1/256, ttl=63 (request in 26)
28	35.739012231	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1b04, seq=2/512, ttl=64 (reply in 29)
29	35.739188580	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1b04, seq=2/512, ttl=63 (request in 28)
30	36.088318829	Cisco_7b:ce:8b	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
31	36.762969750	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1b04, seq=3/768, ttl=64 (reply in 32)
32	36.763142607	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1b04, seq=3/768, ttl=63 (request in 31)
33	37.786933206	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1b04, seq=4/1024, ttl=64 (reply in 34)
34	37.787110882	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1b04, seq=4/1024, ttl=63 (request in 33)
35	38.093257002	Cisco_7b:ce:8b	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/60/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800b
36	38.810899875	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1b04, seq=5/1280, ttl=64 (reply in 37)
37	38.811085024	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1b04, seq=5/1280, ttl=63 (request in 36)
38	39.733669782	HewlettP_5a:79:97	HewlettP_61:2d:df	ARP	42	Who has 172.16.60.1? Tell 172.16.60.254
39	39.733806321	HewlettP_61:2d:df	HewlettP_5a:79:97	ARP	60	172.16.60.1 is at 00:21:5a:61:2d:df
40	39.834850620	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1b04, seq=6/1536, ttl=64 (reply in 41)
41	39.835021521	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1b04, seq=6/1536, ttl=63 (request in 40)

Figura 12: Ping do gnu63 para gnu62 e capturar no eth0 do gnu64

No.	Time	Source	Destination	Protocol	Length	Info
7	8.829429385	Cisco_7b:ce:93	CDP/VTP/DTP/PAgP/UDLD	CDP	602	Device ID: gnu-sw6 Port ID: FastEthernet0/19
8	10.024507814	Cisco_7b:ce:93	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8013
9	12.029422452	Cisco_7b:ce:93	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8013
10	14.034243921	Cisco_7b:ce:93	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8013
11	16.043712408	Cisco_7b:ce:93	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8013
12	18.044183823	Cisco_7b:ce:93	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8013
13	18.362935496	Cisco_7b:ce:93	Cisco_7b:ce:93	LOOP	60	Reply
14	20.049270619	Cisco_7b:ce:93	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8013
15	22.056012579	Cisco_7b:ce:93	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8013
16	24.058879824	Cisco_7b:ce:93	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8013
17	26.068122235	Cisco_7b:ce:93	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8013
18	28.068570394	Cisco_7b:ce:93	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8013
19	28.375547484	Cisco_7b:ce:93	Cisco_7b:ce:93	LOOP	60	Reply
20	30.073585952	Cisco_7b:ce:93	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8013
21	32.078593757	Cisco_7b:ce:93	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8013
22	32.677024773	Netronix_71:73:da	Broadcast	ARP	42	Who has 172.16.61.1? Tell 172.16.61.253
23	32.677145039	HewlettP_19:02:ba	Netronix_71:73:da	ARP	60	172.16.61.1 is at 00:22:64:19:02:ba
24	32.677163896	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1b04, seq=1/256, ttl=63 (reply in 25)
25	32.677282976	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1b04, seq=1/256, ttl=64 (request in 24)
26	33.708971277	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1b04, seq=2/512, ttl=63 (reply in 27)
27	33.709104464	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1b04, seq=2/512, ttl=64 (request in 26)
28	34.083291677	Cisco_7b:ce:93	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8013
29	34.732929285	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1b04, seq=3/768, ttl=63 (reply in 30)
30	34.733061006	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1b04, seq=3/768, ttl=64 (request in 29)
31	35.756893021	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1b04, seq=4/1024, ttl=63 (reply in 32)
32	35.757029630	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1b04, seq=4/1024, ttl=64 (request in 31)
33	36.092735580	Cisco_7b:ce:93	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8013
34	36.780859200	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1b04, seq=5/1280, ttl=63 (reply in 35)
35	36.781002654	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1b04, seq=5/1280, ttl=64 (request in 34)
36	37.804807571	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1b04, seq=6/1536, ttl=63 (reply in 37)
37	37.804941526	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1b04, seq=6/1536, ttl=64 (request in 36)
38	37.836390558	HewlettP_19:02:ba	Netronix_71:73:da	ARP	60	Who has 172.16.61.253? Tell 172.16.61.1
39	37.836400754	Netronix_71:73:da	HewlettP_19:02:ba	ARP	42	172.16.61.253 is at 00:08:54:71:73:da
40	38.093100698	Cisco_7b:ce:93	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8013
41	38.370709672	Cisco_7b:ce:93	Cisco_7b:ce:93	LOOP	60	Reply
42	38.828785414	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1b04, seq=7/1792, ttl=63 (reply in 43)
43	38.828917903	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1b04, seq=7/1792, ttl=64 (request in 42)
44	39.852746635	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x1b04, seq=8/2048, ttl=63 (reply in 45)

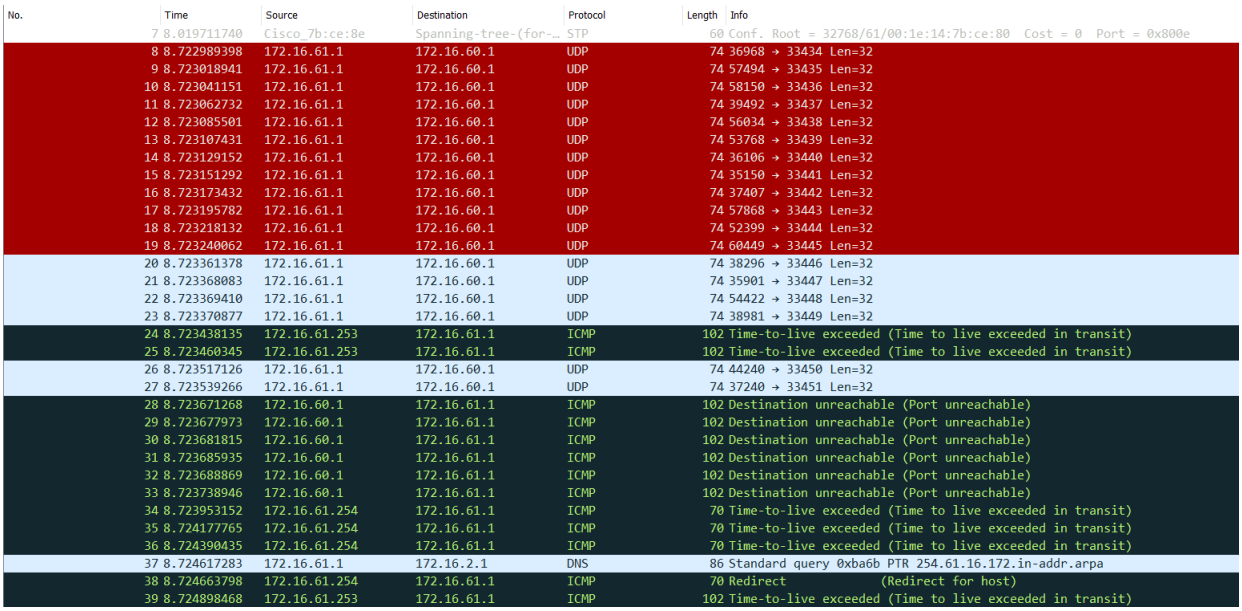
Figura 13: Ping do gnu63 para gnu62 e capturar no eth1 do gnu64

Experiência 4



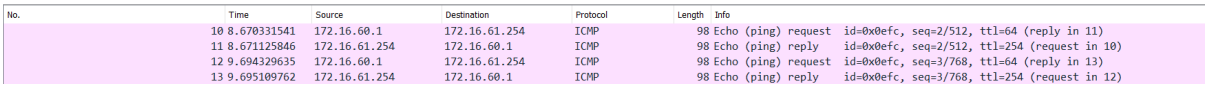
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) request id=0x0fa4, seq=26/6656, ttl=64 (reply in 3)
2	0.000682918	172.16.61.254	172.16.61.1	ICMP	70	Redirect (Redirect for host)
3	0.000988688	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) reply id=0x0fa4, seq=26/6656, ttl=63 (request in 1)
4	0.702910287	Cisco_7b:ce:8e	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800e
5	1.001058041	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) request id=0x0fa4, seq=27/6912, ttl=64 (reply in 7)
6	1.001729645	172.16.61.254	172.16.61.1	ICMP	70	Redirect (Redirect for host)
7	1.002057066	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) reply id=0x0fa4, seq=27/6912, ttl=63 (request in 5)

Figura 14: Ping do GNU62 para GNU63 sem rota para GNU64 e com redirecionamento



No.	Time	Source	Destination	Protocol	Length	Info
7	8.019711740	Cisco_7b:ce:8e	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/61/00:1e:14:7b:ce:80 Cost = 0 Port = 0x800e
8	8.722989398	172.16.61.1	172.16.60.1	UDP	74	36968 → 33434 Len=32
9	8.723018941	172.16.61.1	172.16.60.1	UDP	74	57494 → 33435 Len=32
10	8.723041151	172.16.61.1	172.16.60.1	UDP	74	58150 → 33436 Len=32
11	8.723062732	172.16.61.1	172.16.60.1	UDP	74	39492 → 33437 Len=32
12	8.723085501	172.16.61.1	172.16.60.1	UDP	74	56034 → 33438 Len=32
13	8.723107431	172.16.61.1	172.16.60.1	UDP	74	53768 → 33439 Len=32
14	8.723129152	172.16.61.1	172.16.60.1	UDP	74	36106 → 33440 Len=32
15	8.723151292	172.16.61.1	172.16.60.1	UDP	74	35150 → 33441 Len=32
16	8.723173432	172.16.61.1	172.16.60.1	UDP	74	37407 → 33442 Len=32
17	8.723195782	172.16.61.1	172.16.60.1	UDP	74	57868 → 33443 Len=32
18	8.723218132	172.16.61.1	172.16.60.1	UDP	74	52399 → 33444 Len=32
19	8.723240062	172.16.61.1	172.16.60.1	UDP	74	60449 → 33445 Len=32
20	8.723361378	172.16.61.1	172.16.60.1	UDP	74	38296 → 33446 Len=32
21	8.723368083	172.16.61.1	172.16.60.1	UDP	74	35901 → 33447 Len=32
22	8.723369410	172.16.61.1	172.16.60.1	UDP	74	54422 → 33448 Len=32
23	8.723370877	172.16.61.1	172.16.60.1	UDP	74	38981 → 33449 Len=32
24	8.723438135	172.16.61.253	172.16.61.1	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
25	8.723460345	172.16.61.253	172.16.61.1	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
26	8.723517126	172.16.61.1	172.16.60.1	UDP	74	44240 → 33450 Len=32
27	8.723539266	172.16.61.1	172.16.60.1	UDP	74	37240 → 33451 Len=32
28	8.723671268	172.16.60.1	172.16.61.1	ICMP	102	Destination unreachable (Port unreachable)
29	8.723677973	172.16.60.1	172.16.61.1	ICMP	102	Destination unreachable (Port unreachable)
30	8.723681815	172.16.60.1	172.16.61.1	ICMP	102	Destination unreachable (Port unreachable)
31	8.723685935	172.16.60.1	172.16.61.1	ICMP	102	Destination unreachable (Port unreachable)
32	8.723688869	172.16.60.1	172.16.61.1	ICMP	102	Destination unreachable (Port unreachable)
33	8.723738946	172.16.60.1	172.16.61.1	ICMP	102	Destination unreachable (Port unreachable)
34	8.723953152	172.16.61.254	172.16.61.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
35	8.724177765	172.16.61.254	172.16.61.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
36	8.724390435	172.16.61.254	172.16.61.1	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
37	8.724617283	172.16.61.1	172.16.2.1	DNS	86	Standard query 0xba6b PTR 254.61.16.172.in-addr.arpa
38	8.724663798	172.16.61.254	172.16.61.1	ICMP	70	Redirect (Redirect for host)
39	8.724898468	172.16.61.253	172.16.61.1	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)

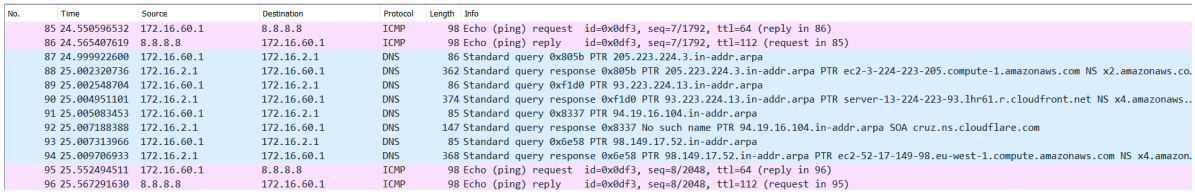
Figura 15: Traceroute



No.	Time	Source	Destination	Protocol	Length	Info
10	8.670331541	172.16.60.1	172.16.61.254	ICMP	98	Echo (ping) request id=0x0efc, seq=2/512, ttl=64 (reply in 11)
11	8.671125846	172.16.61.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0efc, seq=2/512, ttl=254 (request in 10)
12	9.694329635	172.16.60.1	172.16.61.254	ICMP	98	Echo (ping) request id=0x0efc, seq=3/768, ttl=64 (reply in 13)
13	9.695109762	172.16.61.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0efc, seq=3/768, ttl=254 (request in 12)

Figura 16: Ping do GNU63 para o router com NAT

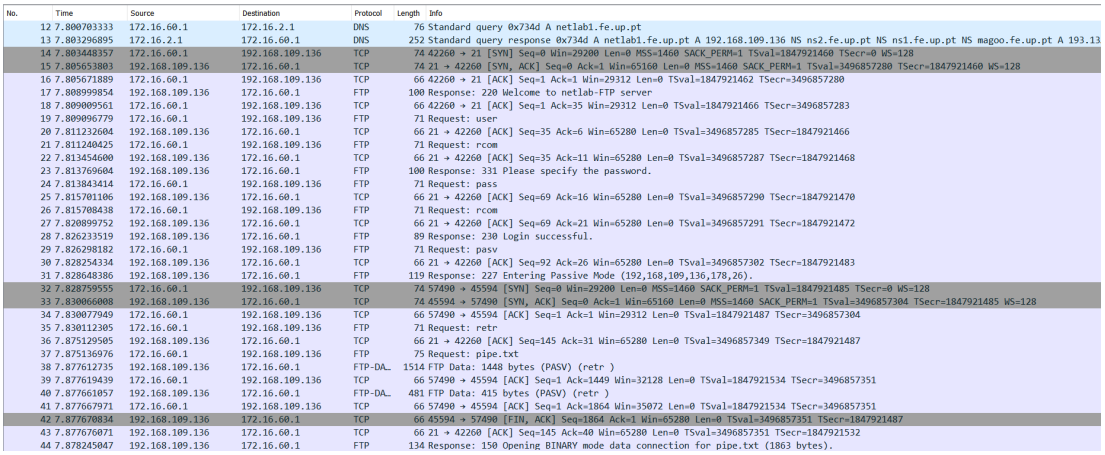
Experiência 5



No.	Time	Source	Destination	Protocol	Length	Info
85	24.550596532	172.16.60.1	8.8.8.8	ICMP	98	Echo (ping) request id=0x0df3, seq=7/1792, ttl=64 (reply in 86)
86	24.565407619	8.8.8.8	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0df3, seq=7/1792, ttl=112 (request in 85)
87	24.999922600	172.16.60.1	172.16.2.1	DNS	86	Standard query 0x805b PTR 205.223.224.3.in-addr.arpa
88	25.002320736	172.16.2.1	172.16.60.1	DNS	362	Standard query response 0x805b PTR 205.223.224.3.in-addr.arpa PTR ec2-3-224-223-205.compute-1.amazonaws.com NS x2.amazonaws.co…
89	25.002548704	172.16.60.1	172.16.2.1	DNS	86	Standard query 0xf1d0 PTR 93.223.224.13.in-addr.arpa
90	25.004951101	172.16.2.1	172.16.60.1	DNS	374	Standard query response 0xf1d0 PTR 93.223.224.13.in-addr.arpa PTR server-13-224-223-93.lhr61.r.cloudfront.net NS x4.amazonaws…
91	25.005083453	172.16.60.1	172.16.2.1	DNS	85	Standard query 0x8337 PTR 94.19.16.104.in-addr.arpa
92	25.007188388	172.16.2.1	172.16.60.1	DNS	147	Standard query response 0x8337 No such name PTR 94.19.16.104.in-addr.arpa SOA cruz.ns.cloudflare.com
93	25.007313966	172.16.60.1	172.16.2.1	DNS	85	Standard query 0x6e58 PTR 98.149.17.52.in-addr.arpa
94	25.009706933	172.16.2.1	172.16.60.1	DNS	368	Standard query response 0x6e58 PTR 98.149.17.52.in-addr.arpa PTR ec2-52-17-149-98.eu-west-1.compute.amazonaws.com NS x4.amazon…
95	25.552494511	172.16.60.1	8.8.8.8	ICMP	98	Echo (ping) request id=0x0df3, seq=8/2048, ttl=64 (reply in 96)
96	25.567291630	8.8.8.8	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0df3, seq=8/2048, ttl=112 (request in 95)

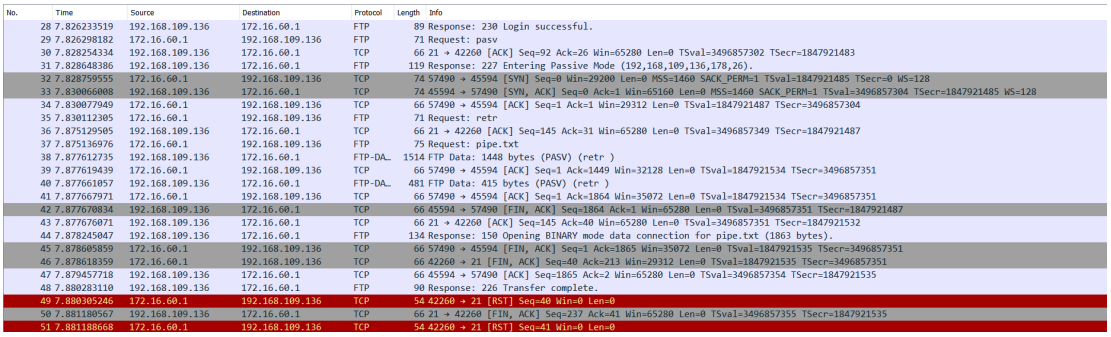
Figura 17: Ping 8.8.8.8

Experiência 6



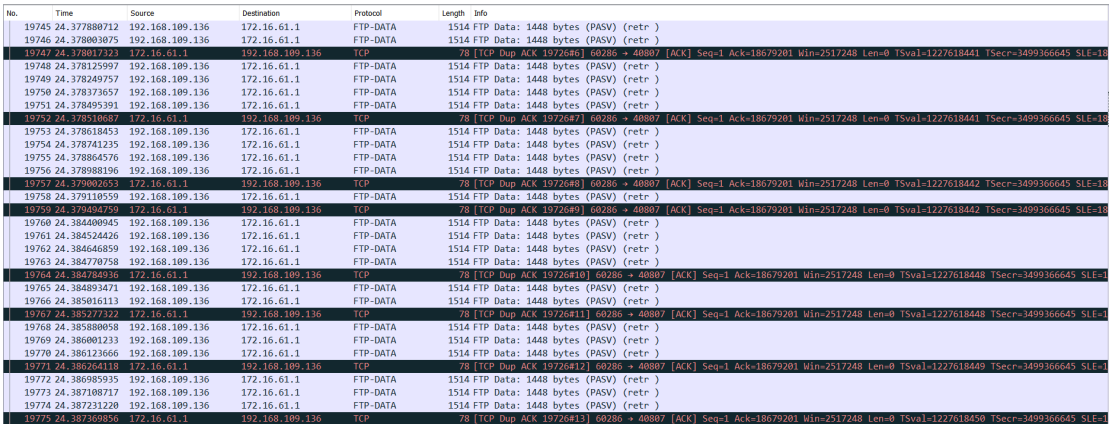
No.	Time	Source	Destination	Protocol	Length	Info
12	7.800703333	172.16.60.1	172.16.2.1	DNS	76	Standard query 0x734d A netlab1.fe.up.pt
13	7.803296895	172.16.2.1	172.16.60.1	DNS	252	Standard query response 0x734d A netlab1.fe.up.pt A 192.168.109.136 NS ns2.fe.up.pt NS ns1.fe.up.pt NS magoo.fe.up.pt A 193.13
14	7.803448357	172.16.60.1	192.168.109.136	TCP	74	42260 → 21 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1847921460 TSecr=0 WS=128
15	7.805653803	192.168.109.136	172.16.60.1	TCP	74	21 → 42260 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3496857280 TSecr=1847921460 WS=128
16	7.805671889	172.16.60.1	192.168.109.136	TCP	66	42260 → 21 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1847921462 TSecr=3496857280
17	7.808999854	192.168.109.136	172.16.60.1	FTP	100	Response: 220 Welcome to netlab-FTP server
18	7.809009561	172.16.60.1	192.168.109.136	TCP	66	42260 → 21 [ACK] Seq=1 Ack=35 Win=29312 Len=0 TSval=1847921466 TSecr=3496857283
19	7.809096779	172.16.60.1	192.168.109.136	FTP	71	Request: user
20	7.811232604	192.168.109.136	172.16.60.1	TCP	66	21 → 42260 [ACK] Seq=35 Ack=6 Win=65280 Len=0 TSval=3496857285 TSecr=1847921466
21	7.811240425	172.16.60.1	192.168.109.136	FTP	71	Request: rcom
22	7.813454600	192.168.109.136	172.16.60.1	TCP	66	21 → 42260 [ACK] Seq=35 Ack=11 Win=65280 Len=0 TSval=3496857287 TSecr=1847921468
23	7.813769604	192.168.109.136	172.16.60.1	FTP	100	Response: 331 Please specify the password.
24	7.813843414	172.16.60.1	192.168.109.136	FTP	71	Request: pass
25	7.815701106	192.168.109.136	172.16.60.1	TCP	66	21 → 42260 [ACK] Seq=69 Ack=16 Win=65280 Len=0 TSval=3496857290 TSecr=1847921470
26	7.815708438	172.16.60.1	192.168.109.136	FTP	71	Request: rcom
27	7.820899752	192.168.109.136	172.16.60.1	TCP	66	21 → 42260 [ACK] Seq=69 Ack=21 Win=65280 Len=0 TSval=3496857291 TSecr=1847921472
28	7.826233519	192.168.109.136	172.16.60.1	FTP	89	Response: 230 Login successful.
29	7.826298182	172.16.60.1	192.168.109.136	FTP	71	Request: pasv
30	7.828254334	192.168.109.136	172.16.60.1	TCP	66	21 → 42260 [ACK] Seq=92 Ack=26 Win=65280 Len=0 TSval=3496857302 TSecr=1847921483
31	7.828648386	192.168.109.136	172.16.60.1	FTP	119	Response: 227 Entering Passive Mode (192,168,109,136,178,26).
32	7.828759555	172.16.60.1	192.168.109.136	TCP	74	57490 → 45594 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1847921485 TSecr=0 WS=128
33	7.830066008	192.168.109.136	172.16.60.1	TCP	74	45594 → 57490 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3496857304 TSecr=1847921485 WS=128
34	7.830077949	172.16.60.1	192.168.109.136	TCP	66	57490 → 45594 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1847921487 TSecr=3496857304
35	7.830112305	172.16.60.1	192.168.109.136	FTP	71	Request: retr
36	7.875129505	192.168.109.136	172.16.60.1	TCP	66	21 → 42260 [ACK] Seq=145 Ack=31 Win=65280 Len=0 TSval=3496857349 TSecr=1847921487
37	7.875136976	172.16.60.1	192.168.109.136	FTP	75	Request: pipe.txt
38	7.877612735	192.168.109.136	172.16.60.1	FTP-DA…	1514	FTP Data: 1448 bytes (PASV) (retr)
39	7.877619439	172.16.60.1	192.168.109.136	TCP	66	57490 → 45594 [ACK] Seq=1 Ack=1449 Win=32128 Len=0 TSval=1847921534 TSecr=3496857351
40	7.877661057	192.168.109.136	172.16.60.1	FTP-DA…	481	FTP Data: 415 bytes (PASV) (retr)
41	7.877667971	172.16.60.1	192.168.109.136	TCP	66	57490 → 45594 [ACK] Seq=1 Ack=1864 Win=35072 Len=0 TSval=1847921534 TSecr=3496857351
42	7.877670834	192.168.109.136	172.16.60.1	TCP	66	45594 → 57490 [FIN, ACK] Seq=1864 Ack=1 Win=65280 Len=0 TSval=3496857351 TSecr=1847921487
43	7.877676071	192.168.109.136	172.16.60.1	TCP	66	21 → 42260 [ACK] Seq=145 Ack=40 Win=65280 Len=0 TSval=3496857351 TSecr=1847921532
44	7.878245047	192.168.109.136	172.16.60.1	FTP	134	Response: 150 Opening BINARY mode data connection for pipe.txt (1863 bytes).

Figura 18: FTP download - Part 1



No.	Time	Source	Destination	Protocol	Length	Info
28	7.826233519	192.168.109.136	172.16.60.1	FTP	89	Response: 230 Login successful.
29	7.826298182	172.16.60.1	192.168.109.136	FTP	71	Request: pasv
30	7.828254334	192.168.109.136	172.16.60.1	TCP	66	21 → 42260 [ACK] Seq=92 Ack=26 Win=65280 Len=0 TSval=3496857302 TSecr=1847921483
31	7.828648386	192.168.109.136	172.16.60.1	FTP	119	Response: 227 Entering Passive Mode (192,168,109,136,178,26).
32	7.828759555	172.16.60.1	192.168.109.136	TCP	74	57490 → 45594 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1847921485 TSecr=0 WS=128
33	7.830066008	192.168.109.136	172.16.60.1	TCP	74	45594 → 57490 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3496857304 TSecr=1847921485 WS=128
34	7.830077949	172.16.60.1	192.168.109.136	TCP	66	57490 → 45594 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1847921487 TSecr=3496857304
35	7.830112305	172.16.60.1	192.168.109.136	FTP	71	Request: retr
36	7.875129505	192.168.109.136	172.16.60.1	TCP	66	21 → 42260 [ACK] Seq=145 Ack=31 Win=65280 Len=0 TSval=3496857349 TSecr=1847921487
37	7.875136976	172.16.60.1	192.168.109.136	FTP	75	Request: pipe.txt
38	7.877612735	192.168.109.136	172.16.60.1	FTP-DA…	1514	FTP Data: 1448 bytes (PASV) (retr)
39	7.877619439	172.16.60.1	192.168.109.136	TCP	66	57490 → 45594 [ACK] Seq=1 Ack=1449 Win=32128 Len=0 TSval=1847921534 TSecr=3496857351
40	7.877661057	192.168.109.136	172.16.60.1	FTP-DA…	481	FTP Data: 415 bytes (PASV) (retr)
41	7.877667971	172.16.60.1	192.168.109.136	TCP	66	57490 → 45594 [ACK] Seq=1 Ack=1864 Win=35072 Len=0 TSval=1847921534 TSecr=3496857351
42	7.877670834	192.168.109.136	172.16.60.1	TCP	66	45594 → 57490 [FIN, ACK] Seq=1864 Ack=1 Win=65280 Len=0 TSval=3496857351 TSecr=1847921487
43	7.877676071	192.168.109.136	172.16.60.1	TCP	66	21 → 42260 [ACK] Seq=145 Ack=40 Win=65280 Len=0 TSval=3496857351 TSecr=1847921532
44	7.878245047	192.168.109.136	172.16.60.1	FTP	134	Response: 150 Opening BINARY mode data connection for pipe.txt (1863 bytes).
45	7.878605859	172.16.60.1	192.168.109.136	TCP	66	57490 → 45594 [FIN, ACK] Seq=1 Ack=1865 Win=35072 Len=0 TSval=1847921535 TSecr=3496857351
46	7.878618359	172.16.60.1	192.168.109.136	TCP	66	42260 → 21 [FIN, ACK] Seq=40 Ack=213 Win=29312 Len=0 TSval=1847921535 TSecr=3496857351
47	7.879457718	192.168.109.136	172.16.60.1	TCP	66	45594 → 57490 [ACK] Seq=1865 Ack=2 Win=65280 Len=0 TSval=3496857354 TSecr=1847921535
48	7.880283110	192.168.109.136	172.16.60.1	FTP	90	Response: 226 Transfer complete.
49	7.880305246	172.16.60.1	192.168.109.136	TCP	54	42260 → 21 [RST] Seq=40 Win=0 Len=0
50	7.881180567	192.168.109.136	172.16.60.1	TCP	66	21 → 42260 [FIN, ACK] Seq=237 Ack=41 Win=65280 Len=0 TSval=3496857355 TSecr=1847921535
51	7.881188668	172.16.60.1	192.168.109.136	TCP	54	42260 → 21 [RST] Seq=41 Win=0 Len=0

Figura 19: FTP download - Part 2



No.	Time	Source	Destination	Protocol	Length	Info
19745	24.377880712	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19746	24.378003075	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19747	24.378017323	172.16.61.1	192.168.109.136	TCP	78	[TCP Dup ACK 19726#6] 60286 → 40807 [ACK] Seq=1 Ack=18679201 Win=2517248 Len=0 TSval=1227618441 TSecr=3499366645 SLE=18
19748	24.378125997	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19749	24.378249757	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19750	24.378373657	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19751	24.378495391	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19752	24.378510687	172.16.61.1	192.168.109.136	TCP	78	[TCP Dup ACK 19726#7] 60286 → 40807 [ACK] Seq=1 Ack=18679201 Win=2517248 Len=0 TSval=1227618441 TSecr=3499366645 SLE=18
19753	24.378618453	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19754	24.378741235	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19755	24.378864576	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19756	24.378988196	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19757	24.379002653	172.16.61.1	192.168.109.136	TCP	78	[TCP Dup ACK 19726#8] 60286 → 40807 [ACK] Seq=1 Ack=18679201 Win=2517248 Len=0 TSval=1227618442 TSecr=3499366645 SLE=18
19758	24.379110559	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19759	24.379494759	172.16.61.1	192.168.109.136	TCP	78	[TCP Dup ACK 19726#9] 60286 → 40807 [ACK] Seq=1 Ack=18679201 Win=2517248 Len=0 TSval=1227618442 TSecr=3499366645 SLE=18
19760	24.384400945	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19761	24.384524426	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19762	24.384646859	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19763	24.384770758	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19764	24.384784936	172.16.61.1	192.168.109.136	TCP	78	[TCP Dup ACK 19726#10] 60286 → 40807 [ACK] Seq=1 Ack=18679201 Win=2517248 Len=0 TSval=1227618448 TSecr=3499366645 SLE=1
19765	24.384893471	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19766	24.385016113	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19767	24.385277322	172.16.61.1	192.168.109.136	TCP	78	[TCP Dup ACK 19726#11] 60286 → 40807 [ACK] Seq=1 Ack=18679201 Win=2517248 Len=0 TSval=1227618448 TSecr=3499366645 SLE=1
19768	24.385880058	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19769	24.386001233	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19770	24.386123666	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19771	24.386264118	172.16.61.1	192.168.109.136	TCP	78	[TCP Dup ACK 19726#12] 60286 → 40807 [ACK] Seq=1 Ack=18679201 Win=2517248 Len=0 TSval=1227618449 TSecr=3499366645 SLE=1
19772	24.386985935	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19773	24.387108717	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19774	24.387231220	192.168.109.136	172.16.61.1	FTP-DATA	1514	FTP Data: 1448 bytes (PASV) (retr)
19775	24.387369856	172.16.61.1	192.168.109.136	TCP	78	[TCP Dup ACK 19726#13] 60286 → 40807 [ACK] Seq=1 Ack=18679201 Win=2517248 Len=0 TSval=1227618450 TSecr=3499366645 SLE=1

Figura 20: DUP ACK durante a transferência

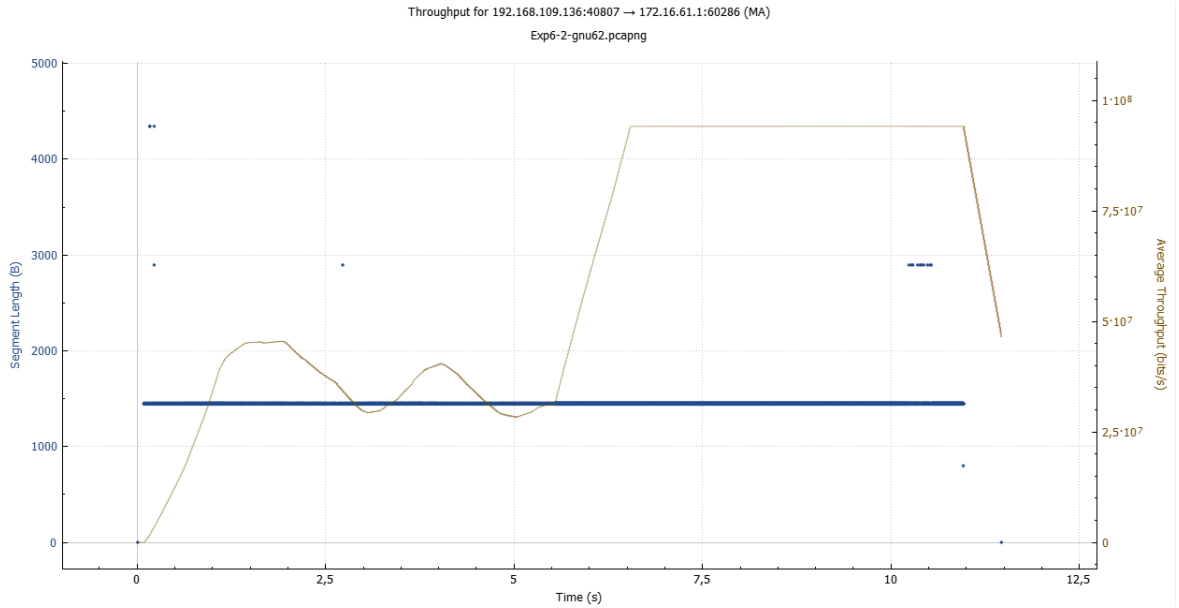


Figura 21: Taxa de transferência no gnu62

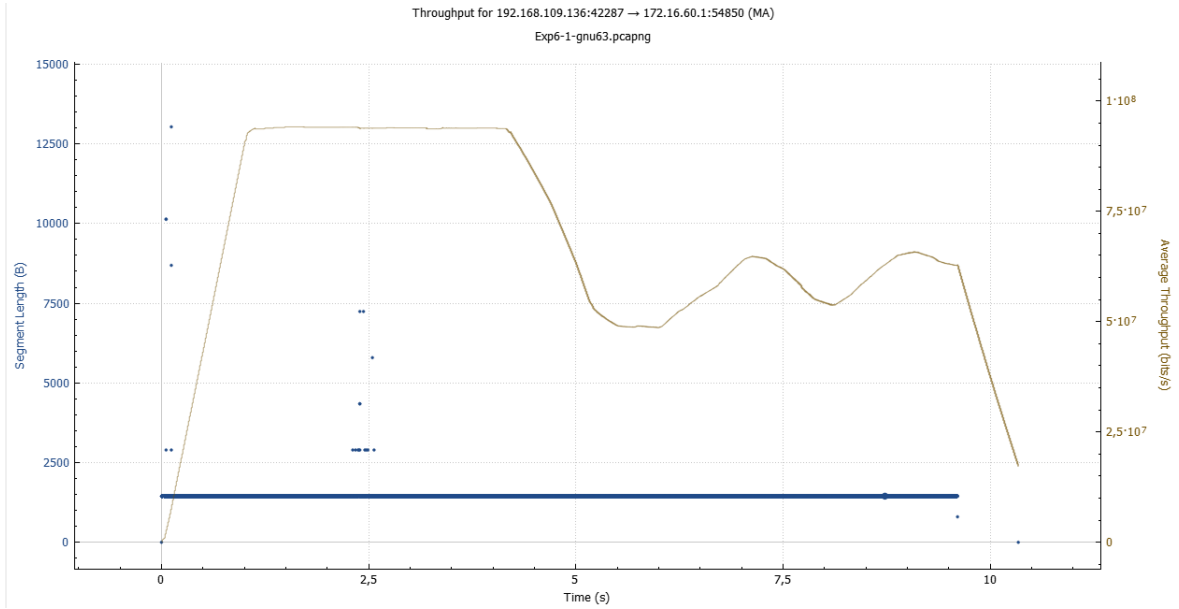


Figura 22: Taxa de transferência no gnu63

5.2 Anexo II - Outras Imagens

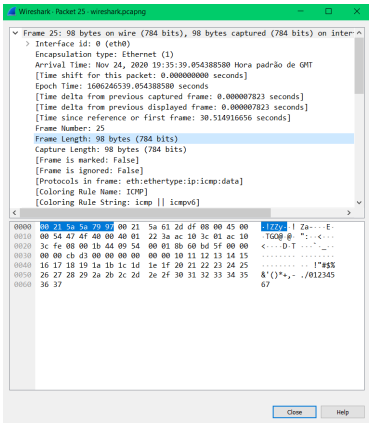


Figura 23: Package Lenght

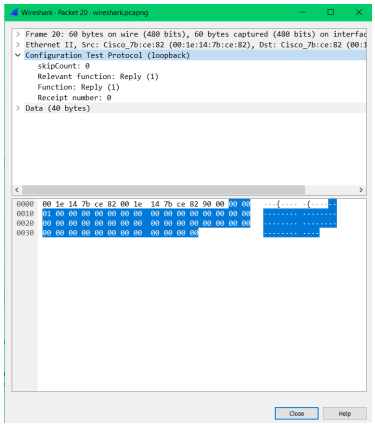


Figura 24: LOOP

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Cisco_7b:ce:82	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/1/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8002
2	2.000888387	Cisco_7b:ce:82	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/1/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8002
3	4.005810600	Cisco_7b:ce:82	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/1/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8002
4	6.010706134	Cisco_7b:ce:82	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/1/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8002
5	6.795088957	Cisco_7b:ce:82	Cisco_7b:ce:82	LOOP	60	Reply
6	8.015711249	Cisco_7b:ce:82	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/1/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8002
7	9.072613155	Cisco_7b:ce:82	CDP/VTP/DTP/PAgP/UD…	DTP	60	Dynamic Trunk Protocol
8	9.072707720	Cisco_7b:ce:82	CDP/VTP/DTP/PAgP/UD…	DTP	90	Dynamic Trunk Protocol
9	10.024528584	Cisco_7b:ce:82	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/1/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8002
10	12.025503644	Cisco_7b:ce:82	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/1/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8002
11	14.030350429	Cisco_7b:ce:82	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/1/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8002
12	16.035258884	Cisco_7b:ce:82	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/1/00:1e:14:7b:ce:80 Cost = 0 Port = 0x8002
13	16.790575907	Cisco_7b:ce:82	Cisco_7b:ce:82	LOOP	60	Reply

Figura 25: LOOP

Configuração do NAT

```
# Configurar a interface interna do router
conf t
interface fastethernet 0/0
ip address 172.16.y1.254 255.255.255.0
no shutdown
ip nat inside
exit

# Configurar a interface externa do router
interface fastethernet 0/1
ip address 172.16.2.y9 255.255.255.0
no shutdown
ip nat outside
exit

# NAT
ip nat pool ovrld 172.16.2.y9 172.16.2.y9 prefix 24
ip nat inside source list 1 pool ovrld overload

# Lista de acesso
access-list 1 permit 172.16.y0.0 0.0.0.7
access-list 1 permit 172.16.y1.0 0.0.0.7

# Rotas
ip route 0.0.0.0 0.0.0.0 172.16.2.254
ip route 172.16.y0.0 255.255.255.0 172.16.y1.253
```

5.3 Anexo III - Código

utils.h

```
struct hostent;

enum code_state {start, first_digit, second_digit, third_digit, last_line,
    code_received};

enum pasv_state {pasv_start, h1, h2, h3, h4, p1, p2, pasv_end};

int parseArguments(char* argument, char* user, char* pass, char* host, char* file_path);

struct hostent* getIP(char *host);

void readServerResponse(int sockfd, char *response, char *fullResponse);

int login(int sockfd, char *user, char *pass);

int activatePassiveMode(int sockfd);

int download_file(int sockfd, int sockfd_client, char* file_path);
```

utils.c

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

#include "utils.h"

#define BUFFER_SIZE 1024

int parseArguments(char* argument, char* user, char* pass, char* host, char* file_path) {

  // Parse the initial part of the argument: "ftp://"
  if(strncmp(argument, "ftp://", 6) != 0) {
   fprintf(stderr,"Invalid Argument: it should start with 'ftp://'\n");
      return -1;
  }

  int i = 6, j;

  if(strchr(argument, '@') != NULL) { // User and pass present in input
   // Parse the part of the argument relative to the user
   while (argument[i] != ':') {
     user[i - 6] = argument[i];
```

```
      i++;
    }
    user[i - 6] = '\0';

    // Parse the part of the argument relative to the pass
    i++;
    j = i;
    while (argument[j] != '@') {
      pass[j - i] = argument[j];
      j++;
    }
    pass[j - i] = '\0';
    j++;
  }
  else {
    user[0] = '\0';
    pass[0] = '\0';
    j = 6;
  }

  // Parse the part of the argument relative to the host
  int k = j;
  while (argument[k] != '/') {
    host[k - j] = argument[k];
    k++;
  }
  host[k - j] = '\0';

  // Parse the part of the argument relative to the url path
  k++;
  int l = k;
  while (argument[l] != '\0') {
    file_path[l - k] = argument[l];
    l++;
  }
  file_path[l - k] = '\0';

  return 0;
}

struct hostent* getIP(char *host) {
   struct hostent *h;

  /*
  struct hostent {
     char   *h_name;  Official name of the host.
     char   **h_aliases; A NULL-terminated array of alternate names for the host.
     int    h_addrtype;  The type of address being returned; usually AF_INET.
     int    h_length; The length of the address in bytes.
     char   **h_addr_list;  A zero-terminated array of network addresses for the host.
     Host addresses are in Network Byte Order.
  };

  #define h_addr h_addr_list[0] The first address in h_addr_list.
  */
```

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

  return h;
}

void readServerResponse(int sockfd, char *response, char *fullResponse) {
  enum code_state state = start;
  char character;
  int i = 0;

  while(state != code_received) {
   read(sockfd, &character, 1);

   fullResponse[i] = character;
   i++;

   switch(state) {
     case start:
      if (isdigit(character)) {
        state = first_digit;
        response[0] = character;
      }
      break;
     case first_digit:
      if (isdigit(character)) {
        state = second_digit;
        response[1] = character;
      }
      else {
        state = start;
        memset(response,0,sizeof(response));
      }
      break;
     case second_digit:
      if (isdigit(character)) {
        state = third_digit;
        response[2] = character;
      }
      else {
        state = start;
        memset(response,0,sizeof(response));
      }
      break;
     case third_digit:
      if ((character == ' ')) {
        state = last_line;
      }
      else {
        state = start;
        memset(response,0,sizeof(response));
      }
      break;
     case last_line:
```

```
      if ((character == '\n')) {
        state = code_received;
      }
      break;
    }
  }

  fullResponse[i] = '\0';
  printf("\nServer Response:\n%s\n", fullResponse);
}

int login(int sockfd, char *user, char *pass) {
  printf(">> Sending username...\n");

  // Send username
   write(sockfd, "user ", 5);
  write(sockfd, user, strlen(user));
  write(sockfd, "\n", 1);

  char response[3];
  char fullResponse[1024];

  readServerResponse(sockfd, response, fullResponse);

  while(response[0] == '4') {
   // Resend username
   write(sockfd, "user ", 5);
   write(sockfd, user, strlen(user));
   write(sockfd, "\n", 1);
      readServerResponse(sockfd, response, fullResponse);
  }

  if (response[0] == '2') { // Password not requested - Login successful
   return 0;
  }

  if(response[0] != '3') {
   fprintf(stderr,"Username not accepted\n");
      return -1;
  }

  printf(">> Sending password...\n");

  // Send pass
   write(sockfd, "pass ", 5);
  write(sockfd, pass, strlen(pass));
  write(sockfd, "\n", 1);

  memset(response,0,sizeof(response));
  memset(fullResponse,0,sizeof(fullResponse));
  readServerResponse(sockfd, response, fullResponse);

  while(response[0] == '4') {
   // Resend pass
   write(sockfd, "pass ", 5);
   write(sockfd, pass, strlen(pass));
```

```
    write(sockfd, "\n", 1);
       readServerResponse(sockfd, response, fullResponse);
  }

  if(strncmp(response, "230", 3) != 0) {
   fprintf(stderr,"Password not accepted\n");
      return -2;
  }

  return 0;
}

int activatePassiveMode(int sockfd) {
  char response[4];
  char fullResponse[512];
  char number[4];
  int i = 0;
  int pasv_numbers[6];
  int j = 0;
  int k = 0;

  write(sockfd, "pasv\n", 5);

  enum pasv_state state = pasv_start;
  char character;

  memset(fullResponse,0,sizeof(fullResponse));

  while(state != pasv_end) {
   read(sockfd, &character, 1);
   fullResponse[k] = character;
   k++;

   switch(state) {
     case pasv_start:
      if (character == '(') {
        state = h1;
        memset(number,0,sizeof(number));
      }
      break;
     case h1:
      if (isdigit(character)) {
        number[i] = character;
        i++;
      }
      else if (character == ',') {
        state = h2;
        number[3] = '\0';
        pasv_numbers[j] = atoi(number);
        memset(number,0,sizeof(number));
        i = 0;
        j++;
      }
      break;
     case h2:
      if (isdigit(character)) {
```

```
      number[i] = character;
      i++;
    }
    else if (character == ',') {
      state = h3;
      number[3] = '\0';
      pasv_numbers[j] = atoi(number);
      memset(number,0,sizeof(number));
      i = 0;
      j++;
    }
    break;
  case h3:
    if (isdigit(character)) {
      number[i] = character;
      i++;
    }
    else if (character == ',') {
      state = h4;
      number[3] = '\0';
      pasv_numbers[j] = atoi(number);
      memset(number,0,sizeof(number));
      i = 0;
      j++;
    }
    break;
  case h4:
    if (isdigit(character)) {
      number[i] = character;
      i++;
    }
    else if (character == ',') {
      state = p1;
      number[3] = '\0';
      pasv_numbers[j] = atoi(number);
      memset(number,0,sizeof(number));
      i = 0;
      j++;
    }
    break;
  case p1:
    if (isdigit(character)) {
      number[i] = character;
      i++;
    }
    else if (character == ',') {
      state = p2;
      number[3] = '\0';
      pasv_numbers[j] = atoi(number);
      memset(number,0,sizeof(number));
      i = 0;
      j++;
    }
    break;
  case p2:
    if (isdigit(character)) {
```

```
        number[i] = character;
        i++;
      }
      else if (character == '\n') {
        state = pasv_end;
        number[3] = '\0';
        pasv_numbers[j] = atoi(number);
      }
      break;
   }
  }

  int port = pasv_numbers[4]*256 + pasv_numbers[5];

  fullResponse[k] = '\0';
  printf("Server Response: \n%s\n", fullResponse);

  return port;
}

int download_file(int sockfd, int sockfd_client, char* file_path) {
  // Send file request
   write(sockfd, "retr ", 5);
  write(sockfd, file_path, strlen(file_path));
  write(sockfd, "\n", 1);

  char response[3];
  char fullResponse[1024];

  readServerResponse(sockfd, response, fullResponse);

  while(response[0] == '4') {
   // Resend file request
   write(sockfd, "retr ", 5);
   write(sockfd, file_path, strlen(file_path));
   write(sockfd, "\n", 1);
      readServerResponse(sockfd, response, fullResponse);
  }

   if (strncmp(response, "150", 3) != 0) {
      fprintf(stderr,"Couldn't open file\n");
      return -1;
   }

  char* filename;

  filename = basename(file_path);

  FILE *file = fopen(filename, "wb+");

  char file_part[BUFFER_SIZE];
  int bytes_read, elems_written;

  while((bytes_read = read(sockfd_client, file_part, BUFFER_SIZE)) > 0) {
   elems_written = fwrite(file_part, bytes_read, 1, file);
   if (elems_written != 1) {
```

```
      fprintf(stderr,"Error downloading file\n");
        return -2;
    }
   }

   fclose(file);

   return 0;
}
```

clientTCP.c

```
/*      (C)2000 FEUP */

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <netdb.h>
#include <string.h>
#include <libgen.h>
#include <sys/time.h>

#include "utils.h"

#define SERVER_PORT 21
#define SERVER_ADDR "192.168.28.96"

#define MAX_SIZE 256

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

   int   sockfd, sockfd_client;
   struct  sockaddr_in server_addr;
   struct  sockaddr_in server_addr_client;

   char user[MAX_SIZE];
   char pass[MAX_SIZE];
   char host[MAX_SIZE];
   char file_path[MAX_SIZE];

   struct hostent *h;

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

   if (parseArguments(argv[1], user, pass, host, file_path) < 0) {
      fprintf(stderr,"Usage: ./download ftp://[user]:[pass]@[host]/[url-path]\n");
      exit(2);
```

```
}

if (user[0] == '\0' && pass[0] == '\0') {
   strcpy(user, "anonymous");
   strcpy(pass, "anypass");
}

printf("\n-------------------- INPUT DATA --------------------\n");
printf("User: %s\nPassword: %s\nHost: %s\nURL path: %s\n", user, pass, host,
    file_path);

if ((h = getIP(host)) == NULL) {
   fprintf(stderr,"Couldn't get Host IP\n");
   exit(3);
}

printf("\nHost name : %s\n", h->h_name);
printf("IP Address : %s\n",inet_ntoa(*((struct in_addr *)h->h_addr)));
printf("----------------------------------------------------\n");

/*server address handling*/
bzero((char*)&server_addr,sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = inet_addr(inet_ntoa(*((struct in_addr *)h->h_addr)));
    /*32 bit Internet address network byte ordered*/
server_addr.sin_port = htons(SERVER_PORT);  /*server TCP port must be network byte
    ordered */

/*open an TCP socket*/
if ((sockfd = socket(AF_INET,SOCK_STREAM,0)) < 0) {
   perror("socket()");
   exit(4);
}

/*connect to the server*/
if(connect(sockfd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
   perror("connect()");
   exit(4);
}

char serverResponse[3];
char fullResponse[1024];

printf("\n>> Conecting to the server...\n");

readServerResponse(sockfd, serverResponse, fullResponse);

if (strncmp(serverResponse, "220", 3) != 0) {
   fprintf(stderr,"Connection lost\n");
   exit(5);
}

if (login(sockfd, user, pass) < 0) {
   fprintf(stderr,"Couldn't login\n");
   exit(6);
}
```

```
printf(">> Entering passive mode...\n\n");

int port;
if ((port = activatePassiveMode(sockfd)) < 0) {
   fprintf(stderr,"Couldn't enter passive mode\n");
   exit(7);
}

printf(">> Connecting to the client port...\n");

/*server address handling*/
bzero((char*)&server_addr_client,sizeof(server_addr_client));
server_addr_client.sin_family = AF_INET;
server_addr_client.sin_addr.s_addr = inet_addr(inet_ntoa(*((struct in_addr
    *)h->h_addr))); /*32 bit Internet address network byte ordered*/
server_addr_client.sin_port = htons(port);  /*server TCP port must be network byte
    ordered */

/*open an TCP socket*/
if ((sockfd_client = socket(AF_INET,SOCK_STREAM,0)) < 0) {
   perror("socket()");
   exit(4);
}

/*connect to the server*/
if(connect(sockfd_client, (struct sockaddr *)&server_addr_client,
    sizeof(server_addr_client)) < 0) {
   perror("connect()");
   exit(4);
}

printf("<< Client connection successful\n\n");

printf(">> Starting downloading the file\n");

struct timeval init_time;
struct timeval current_time;
gettimeofday(&init_time, 0);

if (download_file(sockfd, sockfd_client, file_path) < 0) {
   fprintf(stderr,"Couldn't download file\n");
   exit(8);
}

gettimeofday(&current_time, 0);
double elapsedTime = (current_time.tv_usec - init_time.tv_usec) / 1000.0 +
          (current_time.tv_sec - init_time.tv_sec) * 1000.0;

char *filename = basename(file_path);

printf("File %s downloaded successfully in %f seconds!\n", filename,
    elapsedTime/1000.0);

if (close(sockfd_client) < 0) {
   perror("Error closing client socket");
```

```
        exit(9);
    }

    if (close(sockfd)) {
        perror("Error closing server socket");
        exit(9);
    }

    exit(0);
}
```