Redes de Computadores

Relatório Final

Mestrado Integrado em Engenharia Informática e Computação

Redes de Computadores

Turma 5 - Grupo:
Ana Cláudia Fonseca Santos - 200700742
Eduardo de Mendonça Rodrigues Salgado Ramos - 201505779
José Nuno Amaro Freixo - 201504988
Rúben José da Silva Torres - 201405612

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

22 de Dezembro de 2017

Índice

1 Sumário **3**

2 Introdução **4**

3 Aplicação de Download **5**

4 Configuração e Análise da rede **5**
4.1 Configurar uma rede de IP's 5
4.2 Implementar duas VLANs num switch 6
4.3 Configurar um router em Linux 6
4.4 Configurar um router comercial e implementar NAT 7
4.5 DNS 9
4.6 Ligações TCP 9
4.7 Implementar NAT em Linux 10

5 Conclusão **11**

6 Anexos **12**
6.1 Imagens 12
6.2 Logs das experiências 13
6.2.1 Experiência 1 13
6.2.2 Experiência 2 14
6.2.3 Experiência 3 14
6.2.4 Experiência 4 16
6.2.5 Experiência 5 16
6.2.6 Experiência 6 17
6.3 Código Desenvolvido 17

1 Sumário

No âmbito da unidade curricular de Redes de Computadores foi-nos proposto elaborar, ao longo de vários passos intermédios, uma rede de computadores que no final é composta por duas redes locais virtuais ligadas uma à outra através de um computador que tem função de *router*. Cada rede local é composta por um computador, as redes são separadas utilizando um *switch* configurado para o efeito. Numa das redes é ainda ligado um *router* comercial onde é implementada a técnica de NAT. Ambos os computadores devem ser capazes de aceder à Internet e através de um programa, desenvolvido pelo grupo, descarregar um ficheiro utilizando o protocolo FTP. Num passo opcional foi ainda implementada a técnica de NAT no *router* que conecta as duas redes virtuais.

O desenvolvimento deste trabalho permitiu uma melhor compreensão dos protocolos que permitem computadores comunicarem entre si mesmo não se conhecendo diretamente, bem como o encaminhamento de pacotes através de uma rede com e sem técnica de NAT.

2 Introdução

O trabalho tem como objetivo o desenvolvimento de uma aplicação de download através do protocolo FTP com as especificações fornecidas pelos docentes. De forma a testar esta aplicação foi configurada uma rede de computadores que permitisse comunicar com a Internet utilizando nomes de domínios.

Nas especificações descritas no guião é indicado que a aplicação deve ser capaz de fazer download de um ficheiro ao receber um URL do tipo "ftp://[<username>:<password>@]<hostname>/<file-path>". Deve ser estabelecida uma ligação FTP com o servidor de nome "hostname", é depois feito o login com os valores de "username"e "password". É então enviado um pedido para um servidor entrar em modo passivo, o servidor devolve um endereço ip com uma porta na qual deve ser estabelecida uma nova ligação através da qual será recebido o ficheiro. Pode nesta altura ser executado o comando *RETR* com o argumento "file-path". A informação recebida deve ser guardada para poder ser analisada mais tarde.

No que toca à configuração da rede esta foi feita em seis etapas:

- Primeira Etapa: Rede com os computadores 1 e 4 ligados a um switch na rede 0;
- Segunda Etapa: Configuração de uma segunda rede virtual no switch, verificação que nenhum dos computadores na rede 0 comunica com o computador 2 ligado à rede 1;
- Terceira Etapa: Ligação do segundo interface do computador 4 à rede 1, utilização do computador 4 como router e verificação de comunicação entre os computadores 1 e 2;
- Quarta Etapa: Ligação da rede 1 a um router comercial e comunicação exterior com a técnica NAT dos computadores 1 e 2.
- Quinta Etapa: Configuração de DNS para os computadores 1 e 2.
- Sexta Etapa: Configuração do computador 4 como router que implementa a técnica NAT.

3 Aplicação de Download

A primeira parte deste trabalho consistiu no desenvolvimento de uma aplicação de download de ficheiros. Para isso foi utilizado o protocolo ***FTP (File Transfer Protocol)***, assim como a sintaxe de endereços url especificada no RFC1738. Para a utilização da aplicação é necessário compilar e correr o código em anexo, passando-lhe um dos seguintes argumentos:

- ftp://<username>:<password>@<hostname>/<file-path>
- ftp://<hostname>/<file-path>

A aplicação começa por validar o argumento utilizando expressões regulares. De seguida, através de uma máquina de estados faz o *parse* do host, do caminho do ficheiro a descarregar e no caso de lhe ser fornecido, do utilizador e da password, caso contrário, estes são assumidos como *anonymous* e *clear*.

Após esta informação ser processada, é efetuado um pedido para descobrir o endereço de IP do respetivo host e estabelecida conexão entre o cliente FTP e o servidor através de um *socket*. Depois do login, é enviado o comando PASV, declarando então o modo passivo, uma vez que é o servidor a decidir para que porta será enviado o ficheiro. Todas as respostas do servidor aos pedidos enviados são processadas numa máquina de estados. Por fim, depois de recebida a informação sobre a nova porta é aberto um novo socket, o *sockfd_client*, para a receção do ficheiro. Concluída a transferência resta fechar os dois sockets e libertar a memória alocada.

4 Configuração e Análise da rede

4.1 Configurar uma rede de IP's

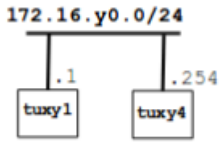

Figura 1: Experiência 1

A configuração da rede consiste em dois computadores e um switch, de modo que os computadores consigam comunicar entre si. Para esta configuração foi utilizado o comando *ifconfig*<interface><endereço ip> que atribui ao interface o endereço ip. O objetivo desta experiência é entender o funcionamento e a importância dos pacotes ARP por isso foram eliminadas de ambos os computador as entradas da tabela ARP utilizando o comando *arp* –d <endereço ip>.

É possível verificar a comunicação entre os dois computadores através do comando *ping <endereço ip>]*. O comando ping vai analisar a tabela ARP que não vai conter entradas por isso vai enviar um pedido *broadcast* do tipo ARP com o objetivo de ficar a conhecer o endereço

que foi passado como argumento do comando ping. O computador com este endereço identifica-se e fica também a conhecer o endereço do computador que enviou o pacote ARP. A resposta do computador que tem o endereço procurado vai conter o seu endereço MAC. Após analisar esta resposta o computador que executou o ping vai preencher na sua tabela ARP o endereço IP e o endereço MAC do computador que descobriu na rede. Depois de conhecer o endereço MAC do destino vai então enviar pacotes ICMP com tamanho apresentado num campo da trama. O interface de *loopback*, descoberto ao analisar os interfaces do computador, serve para fazer testes a interfaces utilizando o próprio computador enviando pacotes para si próprio sem necessidade de ligações externas.

4.2 Implementar duas VLANs num switch

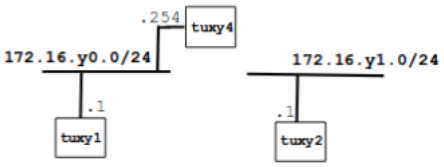

Figura 2: Experiência 2

A configuração da rede consiste na mesma da experiência anterior com a adição de uma VLAN no switch que liga apenas um computador. Para criar uma VLAN é necessário aceder à consola de configuração do switch através da porta série e executar o comando *vlan* <ID> sendo ID o identificador da VLAN.

Com duas VLANs criadas foram adicionadas portas do switch às respetivas VLANs para completar a configuração das sub-redes individuais. Para este fim foram utilizados os comandos: *interface .fastethernet 0/<número da porta>*, *switchport* mode access e *switchport* access vlan <ID >.

Após concluída a configuração das VLANs, verificamos que, utilizando o comando *ping*, a comunicação do computador adicionado com os restantes falhou e os computadores da experiência anterior mantém a comunicação. O facto de um pedido *broadcast* não chegar a nenhum computador explica o porquê de existirem dois domínios, um domínio na rede que liga o computador 2 e outro domínio que liga o computador 1 e 4.

4.3 Configurar um router em Linux

A configuração da rede mantém-se da experiência anterior com a adição de uma interface no computador 4 que passa a estar ligado às duas VLANs, uma em cada interface, tendo como finalidade a função de *router*. O objetivo desta experiência é entender a importância das rotas no encaminhamento dos pacotes recebidos e enviados por um computador.

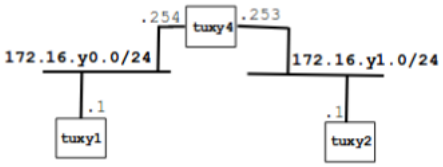

Figura 3: Experiência 3

Cada computador guarda uma tabela de rotas que utiliza para direcionar pacotes que recebe e envia. Uma entrada desta tabela contém um endereço ip utilizado juntamente com uma máscara para determinar os pacotes que serão direcionados para o terceiro valor na tabela, um endereço ip para o qual os pacotes que passam na restrição são dirigidos.

No computador 1 foi necessário definir uma rota para a rede do computador 2 que indica que os pacotes com destino na rede do computador 2 devem ser enviados para o computador 4. No computador 2 foi necessário fazer o mesmo que no computador 1 mas agora com endereços diferentes seguindo o trajeto oposto. No computador 4 não é necessário definir qualquer rota pois por defeito são definidas rotas ao configurar o endereço ip de cada interface.

Utilizando o comando *ping* podemos verificar a conexão entre computadores através de pacotes ICMP. É de notar os endereços MAC e IP de origem e destino dos pacotes. Os pacotes que saem do computador 1 com endereço ip de destino do computador 2 mantêm o endereço ip de destino. No entanto, o seu endereço MAC de destino não vai ser o do computador 2 uma vez que o computador 1 não está na mesma rede que o computador 2 e por isso não tem possibilidade de conhecer o seu endereço MAC. O endereço MAC de destino destes pacotes será o do computador 4 pois após análise da tabela de rotas é obtida informação sobre o computador da mesma rede que o computador 1 para o qual os pacotes devem ser enviados de forma a chegarem ao computador 2. No caso geral todos os pacotes terão endereço MAC de destino do computador do próximo *hop*, ou seja, do próximo nó do grafo de rede para o qual serão dirigidos e origem do computador que os envia. Os endereços ip não sofrem alterações desde a origem até ao destino final.

4.4 Configurar um router comercial e implementar NAT

A rede configurada consiste num *router* comercial que conecta a VLAN 0 e 1 à Internet. A VLAN 1 liga o computador 2 e um *router*, este segundo *router* é o computador 4 que corre sistema operativo Linux e foi configurado para o efeito. No segundo interface deste *router* está configurada a VLAN 0 que o liga ao computador 1.

Para configurar as rotas estáticas do *router* comercial é necessário utilizar o comando ”ip route”que nos casos da experiência recebe 3 argumentos o primeiro é o endereço IP prefixo, o segundo é a máscara e o terceiro é o endereço IP para o qual serão dirigidos os pacotes (gateway). No caso de definir um rota por defeito nos dois primeiros parâmetros podem ser passados os valores 0.0.0.0 e 0.0.0.0 . O objetivo desta experiência é compreender o funcionamento da técnica NAT num *router* comercial e a sua importância.

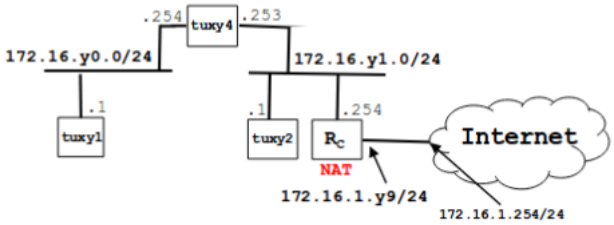

Figura 4: Experiência 4

Nesta experiência foram capturados 2 logs importantes, o primeiro foi da comunicação entre o computador 1 e 2 sem que o segundo computador tenha uma rota para o computador 1 e que a sua rota por defeito seja para o *router* comercial. O computador 2 foi ainda configurado para não aceitar pacotes de redirecionamento. Os pacotes enviados pelo computador 2 para o computador 1 são enviados para o router, isto acontece pois não existe rota definida para o computador 1 logo é tomada a rota por defeito. O *router* tem uma rota para o computador 1 por isso envia os pacotes para este e o computador 2 recebe a resposta do computador 1, o *router* envia também um pacote ICMP com informação sobre o redirecionamento que fez para informar o computador 2 da rota que este deve estabelecer, no entanto este pacote não é aceite pelo computador 2 pelo facto de ter sido desligada a aceitação destes pacotes e portanto o computador 2 continua a enviar todos os pacotes ICMP através do *router* recebendo também a cada pacote um pacote de redirecionamento. Se for executado o comando *traceroute* é possível verificar a rota que os pacotes tomam para chegar ao computador 1 e verifica-se esta rota Router->Computador4 ->Computador1. O segundo log realizado foi depois de ativar a aceitação de pacotes de redirecionamento no computador 2. O que se verificou foi que o computador 2 recebe apenas um pacote de redirecionamento, pois após receber este pacote é adicionada uma rota para o Computador 1 através do computador 4. Se for executado o comando *traceroute* novamente a rota tomada é Computador4 ->Computador1.

Na segunda parte da experiência é necessário configurar NAT no router, isto é feito acedendo à linha de comandos do *router*. Os comandos e ser executados são na configuração de cada interface *ip nat <tipo>* onde tipo indica se é o interface para a rede interior ou exterior podendo alternar entre os valores *inside* e *outside*. É também necessário configurar uma tabela de tradução de endereços com permissões. Para este efeito são utilizados os comandos ”ip nat pool”para indicar os endereços para os quais será possível traduzir endereços interiores. O comando ”ip nat inside source list”é utilizado para criar uma lista de permissões que fica ligada a uma *pool* criada com o comando anterior e significa que os endereços interiores que satisfação as permissões da lista podem ser traduzidos utilizando a *pool* definida. Para adicionar endereços à lista de permissões o comando a utilizar é *access-list <ID da lista> permit <endereço IP><máscara >*.

4.5 DNS

O objetivo desta experiência é verificar a utilização de pacotes DNS e como configurar a utilização de um serviço DNS. Para configurar o serviço DNS é necessário indicar no ficheiro ”/etc/resolv.conf”qual o endereço IP do servidor que disponibiliza o serviço. Isto pode ser feito acrescentando no ficheiro uma linha do tipo ”nameserver <endereço IP>”.

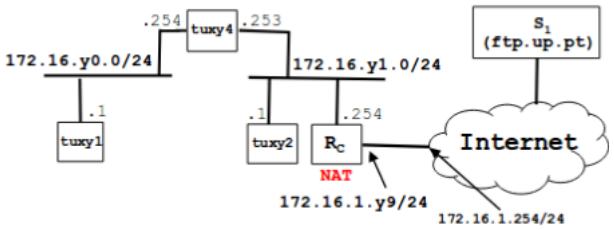

Figura 5: Experiência 5

4.6 Ligações TCP

A configuração da rede manteve-se igual à experiência anterior e testou-se a aplicação download, descrita na Secção 3, no computador 1 e no computador 2. Para tal usamos o servidor ftp da UP de onde transferimos vários ficheiros. Verificada a integridade dos ficheiros foi possível concluir, não só que a rede foi configurada corretamente, como também que a aplicação funciona sem quaisquer erros.

Utilizando os logs capturados foi possível verificar que são abertas 2 ligações FTP, uma de controlo e outra de dados. A primeira ligação a ser aberta funcionará como ligação de controlo, é através desta ligação que são enviados comandos e recebidas respostas para cada comando. É também por esta ligação que é recebida informação sobre outras ligações a serem estabelecidas.

As conexões TCP são separadas em 3 fases distintas, a primeira é o estabelecimento de uma ligação estável do tipo cliente-servidor que recorre ao método *3way-handhsake* para iniciar a ligação. O método referido consiste no cliente enviar um pacote com a flag *SYN* o servidor responder com um pacote com as flags *SYN, ACK* e o cliente responder com um pacote com as flags *ACK*. A segunda fase da conexão TCP é a fase de transmissão de dados e a terceira fase é a terminação da ligação em que o cliente envia um pacote com a flag *FIN* e espera por um pacote com as flags *ACK, FIN* do servidor ao qual deve responder com um pacote com a flag *ACK*.

O protocolo TCP utiliza o mecanismo *Selective Repeat ARQ*, uma variação do *Go-Back-N*, sendo a diferença que o recetor, quando deteta um erro na transmissão, continua a processar as tramas recebidas, enviando nos pacotes ACK o número da trama em que ocorreu o primeiro erro. No final da transmissão, o emissor verifica os pacotes ACK e reenvia as tramas que falharam na primeira transmissão.

O cabeçalho de um pacote TCP indica informação importante sobre a ligação e o controlo de fluxo. São indicadas as portas de origem e de destino do envio, o número de sequência que identifica a posição de sequencia do primeiro byte do campo de dados, o número de *Ackno-*

wledgement que indica a posição, na sequência, do próximo byte que o recetor espera ler e o tamanho da janela usado para o controlar o fluxo de transmissão. É também possível verificar o comprimento do cabeçalho TCP bem como as flags que indicam o tipo da trama e o código de verificação de erros (*Checksum*).

O TCP executa o algoritmo de janela deslizante: a cada envio, são enviados até um número máximo de pacotes, definido pelo tamanho da janela sem que haja um *ACK*. O controlo de fluxo deste protocolo começa por enviar poucos dados de cada vez (janela de congestionamento pequena). Cada vez que é recebido um ACK o tamanho da janela aumenta. Com isto, a quantidade de dados transmitida também aumenta. Na nossa experiência verificou-se um aumento inicial progressivo da janela, que posteriormente estabilizou (figura 11). Em consequência a quantidade de dados transmitida, o *throughput*, teve um comportamento semelhante (figuras 8 e 9).

Para concluir a experiência, aos 4 segundos do download no computador 1 foi iniciado também um download no computador 2 o que levou a uma diminuição clara no *throughput* do primeiro download, como verificável na figura 12 dos anexos. Em ambos os casos mesmo num estado estável é possível verificar um comportamento do tipo ”dentes de serra”. Este comportamento é observável mesmo num estado estável pois a qualquer altura ocorrem tentativas de acelerar a transferência de dados, este é o comportamento esperado por estar intrinsecamente ligado ao método de controlo de congestionamento do protocolo TCP.

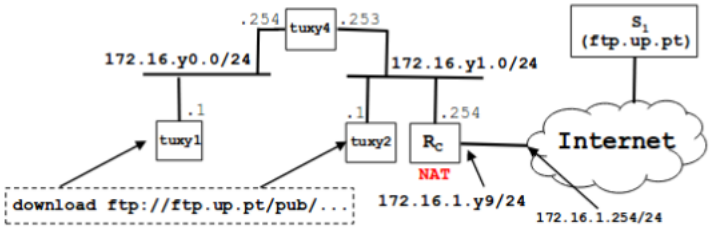

Figura 6: Experiência 6

4.7 Implementar NAT em Linux

O objetivo desta experiência é visualizar o efeito que a técnica NAT tem nos pacotes que são dirigidos por um *router* que implemente esta técnica. A rede configurada consiste num *router* comercial que conecta uma VLAN à Internet, a VLAN liga um computador e um *router*, este segundo *router* é um computador que corre sistema operativo Linux e foi configurado para funcionar como *router* e aplicar a técnica NAT. Num segundo interface deste *router* está configurada uma segunda VLAN que o liga a um computador. Os comandos utilizados para configurar o computador para implementar a técnica NAT foram:

- iptables –t nat –A POSTROUTING -o eth1 –j MASQUERADE
- iptables –A FORWARD –i eth1 –m state –state NEW,INVALID –j DROP

- iptables –L
- iptables –t nat –L

Foram analisados logs capturados em ambos os interfaces do *router* em comunicações do tipo TCP, UDP e ICMP. Para todos os tipos de comunicações foram verificados os mesmos efeitos nos endereços de destino e origem. Um pacote sai do computador com o seu endereço ip de origem e vai para o *router* com endereço de destino do servidor com que deve comunicar. No *router* o endereço de origem deixa de ser o endereço do computador e passa a ser o endereço do interface do *router* que comunica com a rede exterior, o endereço ip de destino não sofre alterações. Os pacotes que chegam ao *router* pelo lado exterior vêm com o endereço de origem do servidor que os enviou e como destino o endereço do interface exterior do *router* nesta altura é verificada a porta em que chegou o pacote e o endereço de destino é traduzido, utilizando uma tabela porta:endereço, no endereço o computador de destino. Este processo é útil para prevenir comunicações indesejadas a partir do exterior e permitir que utilizando apenas um endereço ip muitos computadores consigam comunicar com redes exteriores. Uma observação feita também no decorrer desta experiência foi a melhor compreensão dos endereços *Wildcard* utilizadas na configuração das listas de acesso no *router* comercial que pertence à rede da experiência.

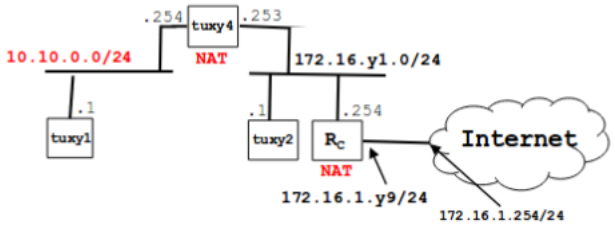

Figura 7: Experiência 7

5 Conclusão

Após a conclusão de todas as experiências propostas pelo guião e, tendo obtido os resultados previstos em todas elas, podemos dar por concluídos com sucesso o desenvolvimento da aplicação de download e a configuração e estudo de uma rede de computadores.

Foram encontradas algumas dificuldade a configurar a rede para as experiências iniciais. Os comandos necessários para configurar a rede eram desconhecidos ao grupo e foi necessário descobrir o seu funcionamento e capacidade de configuração para o objetivo do trabalho. Após um estudo dos comandos necessários, não surgiram quaisquer obstáculos de maior gravidade durante o desenvolvimento das restantes experiências.

Outro ponto que gostaríamos de salientar como ponto de alguma dificuldade foi o desenvolvimento deste relatório. No entanto, retroceder às primeiras experiências e percorrê-las de novo ajudou a solidificar alguns conceitos sobre a configuração de redes.

6 Anexos

6.1 Imagens

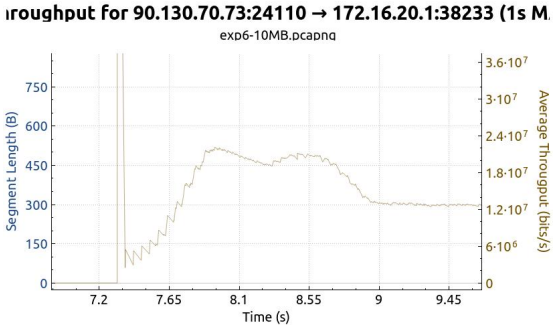

Figura 8: Throughput do TCP no ínicio (experiência 6)

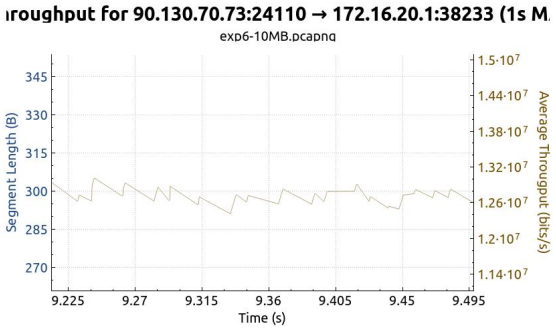

Figura 9: Throughput do TCP depois de estabilizar (experiência 6)

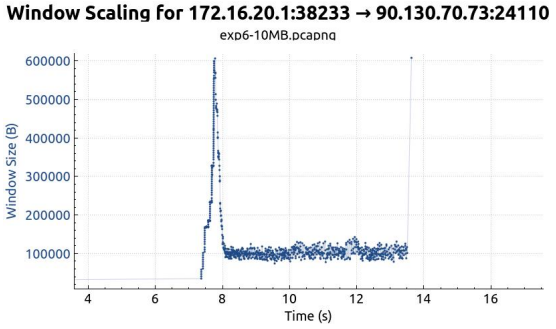

Figura 10: TCP window scaling (experiência 6)

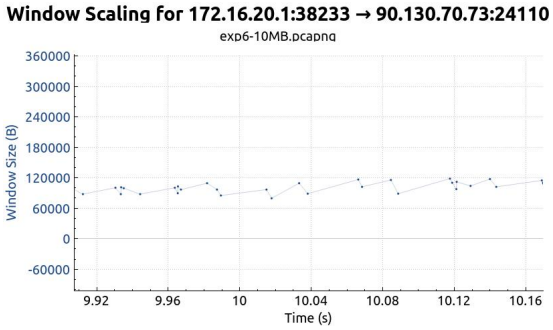

Figura 11: TCP pormenor do window scaling (experiência 6)

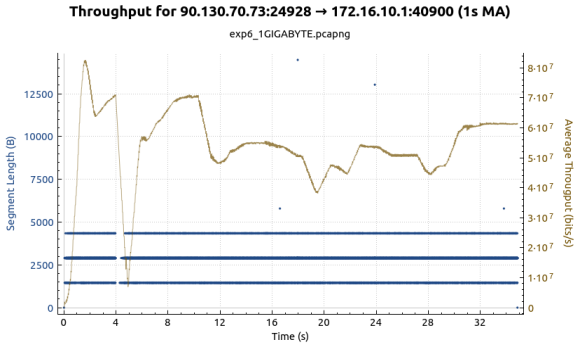

Figura 12: Evolução da transferência de dados no tux1 (experiência 6)

6.2 Logs das experiências

6.2.1 Experiência 1

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_d4:1c:03	Spanning-tree-…	STP	60	Conf. Root = 32768/1/30:37:a6:d4:1c:00 Cost = 0 Port = 0x8003
2	0.620971	Cisco_d4:1c:03	CDP/VTP/DTP/PA…	CDP	453	Device ID: tux-sw4 Port ID: FastEthernet0/1
3	0.749117	G-ProCom_8c:af:af	Broadcast	ARP	42	Who has 172.16.40.254? Tell 172.16.40.1
4	0.749334	HewlettP_5a:7b:ea	G-ProCom_8c:af…	ARP	60	172.16.40.254 is at 00:21:5a:5a:7b:ea
5	0.749353	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x576e, seq=1/256, ttl=64 (reply in 6)
6	0.749611	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x576e, seq=1/256, ttl=64 (request in 5)
7	1.748260	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x576e, seq=2/512, ttl=64 (reply in 8)
8	1.748495	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x576e, seq=2/512, ttl=64 (request in 7)
9	2.004767	Cisco_d4:1c:03	Spanning-tree-…	STP	60	Conf. Root = 32768/1/30:37:a6:d4:1c:00 Cost = 0 Port = 0x8003
10	2.748283	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x576e, seq=3/768, ttl=64 (reply in 11)
11	2.748488	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x576e, seq=3/768, ttl=64 (request in 10)
12	3.748268	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x576e, seq=4/1024, ttl=64 (reply in 13)
13	3.748500	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x576e, seq=4/1024, ttl=64 (request in 12)
14	4.009672	Cisco_d4:1c:03	Spanning-tree-…	STP	60	Conf. Root = 32768/1/30:37:a6:d4:1c:00 Cost = 0 Port = 0x8003
15	4.748265	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x576e, seq=5/1280, ttl=64 (reply in 16)
16	4.748470	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x576e, seq=5/1280, ttl=64 (request in 15)
17	5.748273	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x576e, seq=6/1536, ttl=64 (reply in 18)
18	5.748482	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x576e, seq=6/1536, ttl=64 (request in 17)
19	5.760755	HewlettP_5a:7b:ea	G-ProCom_8c:af…	ARP	60	Who has 172.16.40.1? Tell 172.16.40.254
20	5.760772	G-ProCom_8c:af:af	HewlettP_5a:7b…	ARP	42	172.16.40.1 is at 00:0f:fe:8c:af:af
21	6.014489	Cisco_d4:1c:03	Spanning-tree-…	STP	60	Conf. Root = 32768/1/30:37:a6:d4:1c:00 Cost = 0 Port = 0x8003
22	6.748259	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x576e, seq=7/1792, ttl=64 (reply in 23)
23	6.748464	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x576e, seq=7/1792, ttl=64 (request in 22)
24	7.748268	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x576e, seq=8/2048, ttl=64 (reply in 25)
25	7.748610	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x576e, seq=8/2048, ttl=64 (request in 24)
26	7.969722	Cisco_d4:1c:03	Cisco_d4:1c:03	LOOP	60	Reply
27	8.019415	Cisco_d4:1c:03	Spanning-tree-…	STP	60	Conf. Root = 32768/1/30:37:a6:d4:1c:00 Cost = 0 Port = 0x8003
28	8.748273	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x576e, seq=9/2304, ttl=64 (reply in 29)
29	8.748536	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x576e, seq=9/2304, ttl=64 (request in 28)
30	9.748257	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x576e, seq=10/2560, ttl=64 (reply in 31)
31	9.748493	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x576e, seq=10/2560, ttl=64 (request in 30)
32	10.024217	Cisco_d4:1c:03	Spanning-tree-…	STP	60	Conf. Root = 32768/1/30:37:a6:d4:1c:00 Cost = 0 Port = 0x8003
33	10.748263	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x576e, seq=11/2816, ttl=64 (reply in 34)
34	10.748472	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x576e, seq=11/2816, ttl=64 (request in 33)
35	11.748260	172.16.40.1	172.16.40.254	ICMP	98	Echo (ping) request id=0x576e, seq=12/3072, ttl=64 (reply in 36)
36	11.748496	172.16.40.254	172.16.40.1	ICMP	98	Echo (ping) reply id=0x576e, seq=12/3072, ttl=64 (request in 35)
37	12.028996	Cisco_d4:1c:03	Spanning-tree-…	STP	60	Conf. Root = 32768/1/30:37:a6:d4:1c:00 Cost = 0 Port = 0x8003

Figura 13: Log experiência 1

6.2.2 Experiência 2

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
2	1.747570	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x68c0, seq=1/256, ttl=64 (no response found!)
3	1.747942	172.16.10.254	172.16.10.1	ICMP	126	Destination unreachable (Network unreachable)
4	1.778665	Cisco_3a:fc:03	Cisco_3a:fc:03	LOOP	60	Reply
5	2.004795	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
6	2.747060	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x68c0, seq=2/512, ttl=64 (no response found!)
7	2.747217	172.16.10.254	172.16.10.1	ICMP	126	Destination unreachable (Network unreachable)
8	3.747053	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x68c0, seq=3/768, ttl=64 (no response found!)
9	3.747288	172.16.10.254	172.16.10.1	ICMP	126	Destination unreachable (Network unreachable)
10	3.917660	Cisco_3a:fc:03	CDP/VTP/DTP/PA…	CDP	435	Device ID: tux-sw1 Port ID: FastEthernet0/1
11	4.014970	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
12	4.747067	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x68c0, seq=4/1024, ttl=64 (no response found!)
13	4.747226	172.16.10.254	172.16.10.1	ICMP	126	Destination unreachable (Network unreachable)
14	5.747075	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x68c0, seq=5/1280, ttl=64 (no response found!)
15	6.014418	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
16	6.747059	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x68c0, seq=6/1536, ttl=64 (no response found!)
17	6.748802	HewlettP_a6:a4:f8	G-ProCom_8b:e4…	ARP	60	Who has 172.16.10.1? Tell 172.16.10.254
18	6.748831	G-ProCom_8b:e4:ef	HewlettP_a6:a4…	ARP	42	172.16.10.1 is at 00:0f:fe:8b:e4:ef
19	8.019266	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
20	10.029372	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
21	11.785913	Cisco_3a:fc:03	Cisco_3a:fc:03	LOOP	60	Reply
22	12.029085	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003

Figura 14: Log experiência 2.5 (tux1-2)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
2	0.372616	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x685c, seq=1/256, ttl=64 (reply in 3)
3	0.372980	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x685c, seq=1/256, ttl=64 (request in 2)
4	1.372092	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x685c, seq=2/512, ttl=64 (reply in 5)
5	1.372288	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x685c, seq=2/512, ttl=64 (request in 4)
6	2.004899	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
7	2.372089	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x685c, seq=3/768, ttl=64 (reply in 8)
8	2.372429	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x685c, seq=3/768, ttl=64 (request in 7)
9	3.372088	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x685c, seq=4/1024, ttl=64 (reply in 10)
10	3.372334	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x685c, seq=4/1024, ttl=64 (request in 9)
11	4.014926	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
12	4.026776	Cisco_3a:fc:03	Cisco_3a:fc:03	LOOP	60	Reply
13	4.372087	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x685c, seq=5/1280, ttl=64 (reply in 14)
14	4.372450	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x685c, seq=5/1280, ttl=64 (request in 13)
15	5.372086	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x685c, seq=6/1536, ttl=64 (reply in 16)
16	5.372334	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x685c, seq=6/1536, ttl=64 (request in 15)
17	5.375811	HewlettP_a6:a4:f8	G-ProCom_8b:e4…	ARP	60	Who has 172.16.10.1? Tell 172.16.10.254
18	5.375831	G-ProCom_8b:e4:ef	HewlettP_a6:a4…	ARP	42	172.16.10.1 is at 00:0f:fe:8b:e4:ef
19	6.014947	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
20	6.372093	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x685c, seq=7/1792, ttl=64 (reply in 21)
21	6.372439	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x685c, seq=7/1792, ttl=64 (request in 20)
22	8.019424	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
23	10.029392	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003

Figura 15: Log experiência 2.5 (tux1-tux4)

6.2.3 Experiência 3

No.	Time	Source	Destination	Protocol	Length	Info
4	2.004774	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
5	2.365754	Cisco_3a:fc:03	Cisco_3a:fc:03	LOOP	60	Reply
6	2.485692	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x04df, seq=1/256, ttl=64 (reply in 7)
7	2.485847	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x04df, seq=1/256, ttl=64 (request in 6)
8	2.900543	HewlettP_a6:a4:f8	Broadcast	ARP	60	Who has 172.16.10.3? Tell 172.16.10.254
9	3.484799	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x04df, seq=2/512, ttl=64 (reply in 10)
10	3.485046	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x04df, seq=2/512, ttl=64 (request in 9)
11	3.900764	HewlettP_a6:a4:f8	Broadcast	ARP	60	Who has 172.16.10.3? Tell 172.16.10.254
12	4.009734	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
13	4.484800	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x04df, seq=3/768, ttl=64 (reply in 14)
14	4.484958	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x04df, seq=3/768, ttl=64 (request in 13)
15	4.900526	HewlettP_a6:a4:f8	Broadcast	ARP	60	Who has 172.16.10.3? Tell 172.16.10.254
16	5.484800	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x04df, seq=4/1024, ttl=64 (reply in 17)
17	5.484986	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x04df, seq=4/1024, ttl=64 (request in 16)
18	5.900716	HewlettP_a6:a4:f8	Broadcast	ARP	60	Who has 172.16.10.3? Tell 172.16.10.254
19	6.014470	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
20	6.484801	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x04df, seq=5/1280, ttl=64 (reply in 21)
21	6.484958	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x04df, seq=5/1280, ttl=64 (request in 20)
22	7.484798	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x04df, seq=6/1536, ttl=64 (reply in 23)
23	7.485028	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x04df, seq=6/1536, ttl=64 (request in 22)
24	7.496673	HewlettP_a6:a4:f8	G-ProCom_8b:e4…	ARP	60	Who has 172.16.10.1? Tell 172.16.10.254
25	7.496696	G-ProCom_8b:e4:ef	HewlettP_a6:a4…	ARP	42	172.16.10.1 is at 00:0f:fe:8b:e4:ef
26	8.019349	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
27	8.484807	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x04df, seq=7/1792, ttl=64 (reply in 28)
28	8.485153	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x04df, seq=7/1792, ttl=64 (request in 27)
29	9.484798	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x04df, seq=8/2048, ttl=64 (reply in 30)
30	9.485031	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x04df, seq=8/2048, ttl=64 (request in 29)
31	10.034194	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
32	10.484794	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x04df, seq=9/2304, ttl=64 (reply in 33)
33	10.485145	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x04df, seq=9/2304, ttl=64 (request in 32)
34	11.484799	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x04df, seq=10/2560, ttl=64 (reply in 35)
35	11.485008	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x04df, seq=10/2560, ttl=64 (request in 34)
36	12.029028	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
37	12.373119	Cisco_3a:fc:03	Cisco_3a:fc:03	LOOP	60	Reply
38	12.484798	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x04df, seq=11/2816, ttl=64 (reply in 39)
39	12.484944	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x04df, seq=11/2816, ttl=64 (request in 38)
40	13.484798	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x04df, seq=12/3072, ttl=64 (reply in 41)

Figura 16: Log da experência 3 (1)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
2	0.566401	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x0583, seq=1/256, ttl=64 (reply in 3)
3	0.566596	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0583, seq=1/256, ttl=64 (request in 2)
4	1.490756	172.16.10.1	172.16.1.1	DNS	65	Standard query 0xd030 AAAA tux11
5	1.491016	172.16.10.254	172.16.10.1	ICMP	93	Destination unreachable (Network unreachable)
6	1.566176	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x0583, seq=2/512, ttl=64 (reply in 7)
7	1.566373	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0583, seq=2/512, ttl=64 (request in 6)
8	2.004829	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
9	2.565198	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x0583, seq=3/768, ttl=64 (reply in 10)
10	2.565442	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0583, seq=3/768, ttl=64 (request in 9)
11	3.565053	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x0583, seq=4/1024, ttl=64 (reply in 12)
12	3.565285	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0583, seq=4/1024, ttl=64 (request in 11)
13	4.069051	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
14	4.565055	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x0583, seq=5/1280, ttl=64 (reply in 15)
15	4.565300	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0583, seq=5/1280, ttl=64 (request in 14)
16	5.565056	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x0583, seq=6/1536, ttl=64 (reply in 17)
17	5.565288	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0583, seq=6/1536, ttl=64 (request in 16)
18	6.014542	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
19	6.495858	172.16.10.1	193.136.28.10	DNS	65	Standard query 0xd030 AAAA tux11
20	6.496133	172.16.10.254	172.16.10.1	ICMP	93	Destination unreachable (Network unreachable)
21	6.564273	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x0583, seq=7/1792, ttl=64 (reply in 22)
22	6.564472	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0583, seq=7/1792, ttl=64 (request in 21)
23	7.563295	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x0583, seq=8/2048, ttl=64 (reply in 24)
24	7.563528	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0583, seq=8/2048, ttl=64 (request in 23)
25	7.952436	Cisco_3a:fc:03	Cisco_3a:fc:03	LOOP	60	Reply
26	8.019577	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
27	8.562297	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x0583, seq=9/2304, ttl=64 (reply in 28)
28	8.562489	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0583, seq=9/2304, ttl=64 (request in 27)
29	9.185733	Cisco_3a:fc:03	CDP/VTP/DTP/PA…	CDP	435	Device ID: tux-sw1 Port ID: FastEthernet0/1
30	9.561297	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x0583, seq=10/2560, ttl=64 (reply in 31)
31	9.561634	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0583, seq=10/2560, ttl=64 (request in 30)
32	10.024204	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
33	10.561069	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x0583, seq=11/2816, ttl=64 (reply in 34)
34	10.561260	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0583, seq=11/2816, ttl=64 (request in 33)
35	11.500924	172.16.10.1	172.16.1.1	DNS	65	Standard query 0xd030 AAAA tux11
36	11.501180	172.16.10.254	172.16.10.1	ICMP	93	Destination unreachable (Network unreachable)
37	11.560328	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x0583, seq=12/3072, ttl=64 (reply in 38)

Figura 17: Log da experência 3 (2)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
2	2.004928	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
3	2.111797	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0683, seq=1/256, ttl=64 (reply in 4)
4	2.112301	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0683, seq=1/256, ttl=63 (request in 3)
5	3.110791	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0683, seq=2/512, ttl=64 (reply in 6)
6	3.111268	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0683, seq=2/512, ttl=63 (request in 5)
7	4.009774	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
8	4.110744	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0683, seq=3/768, ttl=64 (reply in 9)
9	4.111193	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0683, seq=3/768, ttl=63 (request in 8)
10	5.110742	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0683, seq=4/1024, ttl=64 (reply in 11)
11	5.110975	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0683, seq=4/1024, ttl=63 (request in 10)
12	6.013586	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
13	6.110744	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0683, seq=5/1280, ttl=64 (reply in 14)
14	6.111194	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0683, seq=5/1280, ttl=63 (request in 13)
15	7.110741	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0683, seq=6/1536, ttl=64 (reply in 16)
16	7.111231	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0683, seq=6/1536, ttl=63 (request in 15)
17	7.123381	HewlettP_a6:a4:f8	G-ProCom_8b:e4…	ARP	60	Who has 172.16.10.1? Tell 172.16.10.254
18	7.123397	G-ProCom_8b:e4:ef	HewlettP_a6:a4…	ARP	42	172.16.10.1 is at 00:0f:fe:8b:e4:ef
19	7.512989	Cisco_3a:fc:03	Cisco_3a:fc:03	LOOP	60	Reply
20	8.019589	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
21	8.110749	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0683, seq=7/1792, ttl=64 (reply in 22)
22	8.111200	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0683, seq=7/1792, ttl=63 (request in 21)
23	9.110748	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0683, seq=8/2048, ttl=64 (reply in 24)
24	9.110988	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0683, seq=8/2048, ttl=63 (request in 23)
25	10.024308	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
26	12.029100	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
27	14.030056	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
28	16.038818	Cisco_3a:fc:03	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003

Figura 18: Log da experência 3 (3)

No.	Time	Source	Destination	Protocol	Length	Info
7	6.034354	G-ProCom_8b:e4:ef	HewlettP_a6:a4…	ARP	60	172.16.10.1 is at 00:0f:fe:8b:e4:ef
8	6.034365	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=1/256, ttl=63 (request in 5)
9	6.556655	Cisco_3a:fc:05	Cisco_3a:fc:05	LOOP	60	Reply
10	7.034780	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=2/512, ttl=64 (reply in 11)
11	7.034919	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=2/512, ttl=63 (request in 10)
12	8.019555	Cisco_3a:fc:05	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8005
13	8.033793	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=3/768, ttl=64 (reply in 14)
14	8.033946	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=3/768, ttl=63 (request in 13)
15	9.032813	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=4/1024, ttl=64 (reply in 16)
16	9.032943	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=4/1024, ttl=63 (request in 15)
17	10.021657	Cisco_3a:fc:05	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8005
18	10.032656	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=5/1280, ttl=64 (reply in 19)
19	10.032813	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=5/1280, ttl=63 (request in 18)
20	11.032684	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=6/1536, ttl=64 (reply in 21)
21	11.032837	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=6/1536, ttl=63 (request in 20)
22	12.029294	Cisco_3a:fc:05	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8005
23	12.032691	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=7/1792, ttl=64 (reply in 24)
24	12.032842	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=7/1792, ttl=63 (request in 23)
25	13.032713	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=8/2048, ttl=64 (reply in 26)
26	13.032851	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=8/2048, ttl=63 (request in 25)
27	14.032727	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=9/2304, ttl=64 (reply in 28)
28	14.032868	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=9/2304, ttl=63 (request in 27)
29	14.034092	Cisco_3a:fc:05	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8005
30	15.032745	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=10/2560, ttl=64 (reply in 31)
31	15.032884	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=10/2560, ttl=63 (request in 30)
32	16.032766	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=11/2816, ttl=64 (reply in 33)
33	16.032918	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=11/2816, ttl=63 (request in 32)
34	16.038975	Cisco_3a:fc:05	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8005
35	16.564272	Cisco_3a:fc:05	Cisco_3a:fc:05	LOOP	60	Reply
36	17.032782	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=12/3072, ttl=64 (reply in 37)
37	17.032913	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=12/3072, ttl=63 (request in 36)
38	18.032803	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=13/3328, ttl=64 (reply in 39)
39	18.032940	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=13/3328, ttl=63 (request in 38)
40	18.044385	Cisco_3a:fc:05	Spanning-tree-…	STP	60	Conf. Root = 32768/10/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8005
41	19.032816	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=14/3584, ttl=64 (reply in 42)
42	19.032951	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=14/3584, ttl=63 (request in 41)
43	20.032848	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=15/3840, ttl=64 (reply in 44)

Figura 19: Log da experência 3 (eth0)

No.	Time	Source	Destination	Protocol	Length	Info
4	5.102466	Kye_04:20:99	Broadcast	ARP	42	Who has 172.16.11.1? Tell 172.16.11.253
5	5.102584	HewlettP_61:2e:c3	Kye_04:20:99	ARP	60	172.16.11.1 is at 00:21:5a:61:2e:c3
6	5.102599	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=1/256, ttl=63 (reply in 7)
7	5.102729	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=1/256, ttl=64 (request in 6)
8	5.625656	Cisco_3a:fc:09	Cisco_3a:fc:09	LOOP	60	Reply
9	6.914470	Cisco_3a:fc:09	Spanning-tree-…	STP	60	Conf. Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8009
10	6.103521	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=2/512, ttl=63 (reply in 11)
11	6.103639	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=2/512, ttl=64 (request in 10)
12	7.102541	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=3/768, ttl=63 (reply in 13)
13	7.102657	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=3/768, ttl=64 (request in 12)
14	8.024432	Cisco_3a:fc:09	Spanning-tree-…	STP	60	Conf. Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8009
15	8.101555	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=4/1024, ttl=63 (reply in 16)
16	8.101662	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=4/1024, ttl=64 (request in 15)
17	9.101397	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=5/1280, ttl=63 (reply in 18)
18	9.101523	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=5/1280, ttl=64 (request in 17)
19	10.024192	Cisco_3a:fc:09	Spanning-tree-…	STP	60	Conf. Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8009
20	10.101425	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=6/1536, ttl=63 (reply in 21)
21	10.101557	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=6/1536, ttl=64 (request in 20)
22	10.105723	HewlettP_61:2e:c3	Kye_04:20:99	ARP	60	Who has 172.16.11.253? Tell 172.16.11.1
23	10.105732	Kye_04:20:99	HewlettP_61:2e…	ARP	42	172.16.11.253 is at 00:c0:df:04:20:99
24	11.101436	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=7/1792, ttl=63 (reply in 25)
25	11.101552	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=7/1792, ttl=64 (request in 24)
26	12.029165	Cisco_3a:fc:09	Spanning-tree-…	STP	60	Conf. Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8009
27	12.101457	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=8/2048, ttl=63 (reply in 28)
28	12.101570	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=8/2048, ttl=64 (request in 27)
29	13.101471	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=9/2304, ttl=63 (reply in 30)
30	13.101587	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=9/2304, ttl=64 (request in 29)
31	14.030074	Cisco_3a:fc:09	Spanning-tree-…	STP	60	Conf. Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8009
32	14.101489	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=10/2560, ttl=63 (reply in 33)
33	14.101604	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=10/2560, ttl=64 (request in 32)
34	15.101509	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=11/2816, ttl=63 (reply in 35)
35	15.101637	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=11/2816, ttl=64 (request in 34)
36	15.633269	Cisco_3a:fc:09	Cisco_3a:fc:09	LOOP	60	Reply
37	16.039009	Cisco_3a:fc:09	Spanning-tree-…	STP	60	Conf. Root = 32768/11/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8009
38	16.101525	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=12/3072, ttl=63 (reply in 39)
39	16.101634	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x0a54, seq=12/3072, ttl=64 (request in 38)
40	17.101547	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x0a54, seq=13/3328, ttl=63 (reply in 41)

Figura 20: Log da experência 3 (eth1)

6.2.4 Experiência 4

No.	Time	Source	Destination	Protocol	Length	Info
7	7.135940	172.16.51.254	172.16.51.1	ICMP	70	Redirect (Redirect for host)
8	7.136313	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x140f, seq=1/256, ttl=63 (request in 6)
9	8.019230	Cisco_7b:d5:02	Spanning-tree-…	STP	60	Conf. Root = 32768/51/00:1e:14:7b:d5:00 Cost = 0 Port = 0x8002
10	8.136384	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x140f, seq=2/512, ttl=64 (reply in 12)
11	8.137067	172.16.51.254	172.16.51.1	ICMP	70	Redirect (Redirect for host)
12	8.137403	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x140f, seq=2/512, ttl=63 (request in 10)
13	9.137497	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x140f, seq=3/768, ttl=64 (reply in 15)
14	9.138186	172.16.51.254	172.16.51.1	ICMP	70	Redirect (Redirect for host)
15	9.138497	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x140f, seq=3/768, ttl=63 (request in 13)
16	10.024812	Cisco_7b:d5:02	Spanning-tree-…	STP	60	Conf. Root = 32768/51/00:1e:14:7b:d5:00 Cost = 0 Port = 0x8002
17	10.138559	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x140f, seq=4/1024, ttl=64 (reply in 19)
18	10.139230	172.16.51.254	172.16.51.1	ICMP	70	Redirect (Redirect for host)
19	10.139565	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x140f, seq=4/1024, ttl=63 (request in 17)
20	10.820743	Cisco_7b:d5:02	Cisco_7b:d5:02	LOOP	60	Reply
21	11.139628	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x140f, seq=5/1280, ttl=64 (reply in 23)
22	11.140311	172.16.51.254	172.16.51.1	ICMP	70	Redirect (Redirect for host)
23	11.140646	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x140f, seq=5/1280, ttl=63 (request in 21)
24	12.032777	Cisco_7b:d5:02	Spanning-tree-…	STP	60	Conf. Root = 32768/51/00:1e:14:7b:d5:00 Cost = 0 Port = 0x8002
25	12.140715	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x140f, seq=6/1536, ttl=64 (reply in 27)
26	12.141408	172.16.51.254	172.16.51.1	ICMP	70	Redirect (Redirect for host)
27	12.141721	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x140f, seq=6/1536, ttl=63 (request in 25)
28	12.147376	BbnBoltB_a0:ac:80	HewlettP_5a:7c…	ARP	60	Who has 172.16.51.1? Tell 172.16.51.253
29	12.147386	HewlettP_5a:7c:e7	BbnBoltB_a0:ac…	ARP	42	172.16.51.1 is at 00:21:5a:5a:7c:e7
30	13.141785	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x140f, seq=7/1792, ttl=64 (reply in 32)
31	13.142406	172.16.51.254	172.16.51.1	ICMP	70	Redirect (Redirect for host)
32	13.142801	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x140f, seq=7/1792, ttl=63 (request in 30)
33	14.033571	Cisco_7b:d5:02	Spanning-tree-…	STP	60	Conf. Root = 32768/51/00:1e:14:7b:d5:00 Cost = 0 Port = 0x8002
34	14.142863	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x140f, seq=8/2048, ttl=64 (reply in 36)
35	14.143570	172.16.51.254	172.16.51.1	ICMP	70	Redirect (Redirect for host)
36	14.143883	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x140f, seq=8/2048, ttl=63 (request in 34)
37	15.143948	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x140f, seq=9/2304, ttl=64 (reply in 39)
38	15.144630	172.16.51.254	172.16.51.1	ICMP	70	Redirect (Redirect for host)
39	15.144963	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x140f, seq=9/2304, ttl=63 (request in 37)
40	16.038515	Cisco_7b:d5:02	Spanning-tree-…	STP	60	Conf. Root = 32768/51/00:1e:14:7b:d5:00 Cost = 0 Port = 0x8002
41	16.145026	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) request id=0x140f, seq=10/2560, ttl=64 (reply in 43)
42	16.145707	172.16.51.254	172.16.51.1	ICMP	70	Redirect (Redirect for host)
43	16.146031	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) reply id=0x140f, seq=10/2560, ttl=63 (request in 41)

Figura 21: Log da experência 4 (tux2-tux1)

6.2.5 Experiência 5

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_3a:fc:08	Cisco_3a:fc:08	LOOP	60	Reply
2	0.639153	172.16.11.1	172.16.1.1	DNS	68	Standard query 0x353f A fe.up.pt
3	0.640321	172.16.1.1	172.16.11.1	DNS	244	Standard query response 0x353f A fe.up.pt A 10.227.240.205 NS ns2.fe.up.pt NS ns1…
4	0.640443	172.16.11.1	10.227.240.205	ICMP	98	Echo (ping) request id=0x6e83, seq=1/256, ttl=64 (reply in 5)
5	0.641347	10.227.240.205	172.16.11.1	ICMP	98	Echo (ping) reply id=0x6e83, seq=1/256, ttl=61 (request in 4)
6	0.641452	172.16.11.1	172.16.1.1	DNS	87	Standard query 0x9131 PTR 205.240.227.10.in-addr.arpa
7	0.642216	172.16.1.1	172.16.11.1	DNS	237	Standard query response 0x9131 PTR 205.240.227.10.in-addr.arpa PTR www.fe.up.pt N…
8	0.754815	Cisco_3a:fc:08	Spanning-tree-…	STP	60	Conf. Root = 32768/1/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8008
9	1.642302	172.16.11.1	10.227.240.205	ICMP	98	Echo (ping) request id=0x6e83, seq=2/512, ttl=64 (reply in 10)
10	1.643024	10.227.240.205	172.16.11.1	ICMP	98	Echo (ping) reply id=0x6e83, seq=2/512, ttl=61 (request in 9)
11	2.641299	172.16.11.1	10.227.240.205	ICMP	98	Echo (ping) request id=0x6e83, seq=3/768, ttl=64 (reply in 12)
12	2.642001	10.227.240.205	172.16.11.1	ICMP	98	Echo (ping) reply id=0x6e83, seq=3/768, ttl=61 (request in 11)
13	2.754398	Cisco_3a:fc:08	Spanning-tree-…	STP	60	Conf. Root = 32768/1/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8008
14	3.640304	172.16.11.1	10.227.240.205	ICMP	98	Echo (ping) request id=0x6e83, seq=4/1024, ttl=64 (reply in 15)
15	3.641035	10.227.240.205	172.16.11.1	ICMP	98	Echo (ping) reply id=0x6e83, seq=4/1024, ttl=61 (request in 14)
16	4.759296	Cisco_3a:fc:08	Spanning-tree-…	STP	60	Conf. Root = 32768/1/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8008

Figura 22: Log da experência 5

6.2.6 Experiência 6

No.	Time	Source	Destination	Protocol	Length	Info
8	2.377416	172.16.10.1	193.137.29.15	TCP	66	50665 → 21 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=49926477 TSecr=309581877
9	2.381718	193.137.29.15	172.16.10.1	FTP	139	Response: 220-Welcome to the University of Porto's mirror archive (mirrors.up.pt)
10	2.381730	193.137.29.15	172.16.10.1	FTP	135	Response: 220----------
11	2.381755	172.16.10.1	193.137.29.15	TCP	66	50665 → 21 [ACK] Seq=1 Ack=74 Win=29312 Len=0 TSval=49926478 TSecr=309581878
12	2.381767	172.16.10.1	193.137.29.15	TCP	66	50665 → 21 [ACK] Seq=1 Ack=143 Win=29312 Len=0 TSval=49926478 TSecr=309581878
13	2.382005	193.137.29.15	172.16.10.1	FTP	72	Response: 220-
14	2.382018	172.16.10.1	193.137.29.15	TCP	66	50665 → 21 [ACK] Seq=1 Ack=149 Win=29312 Len=0 TSval=49926478 TSecr=309581878
15	2.382023	193.137.29.15	172.16.10.1	FTP	151	Response: 220-All connections and transfers are logged. The max number of connections is 200.
16	2.382032	172.16.10.1	193.137.29.15	TCP	66	50665 → 21 [ACK] Seq=1 Ack=234 Win=29312 Len=0 TSval=49926478 TSecr=309581878
17	2.382036	193.137.29.15	172.16.10.1	FTP	72	Response: 220-
18	2.382044	172.16.10.1	193.137.29.15	TCP	66	50665 → 21 [ACK] Seq=1 Ack=240 Win=29312 Len=0 TSval=49926478 TSecr=309581878
19	2.382049	193.137.29.15	172.16.10.1	FTP	140	Response: 220-For more information please visit our website: http://mirrors.up.pt/
20	2.382057	172.16.10.1	193.137.29.15	TCP	66	50665 → 21 [ACK] Seq=1 Ack=314 Win=29312 Len=0 TSval=49926478 TSecr=309581878
21	2.382270	193.137.29.15	172.16.10.1	FTP	127	Response: 220-Questions and comments can be sent to mirrors@uporto.pt
22	2.382282	172.16.10.1	193.137.29.15	TCP	66	50665 → 21 [ACK] Seq=1 Ack=375 Win=29312 Len=0 TSval=49926478 TSecr=309581878
23	2.382286	193.137.29.15	172.16.10.1	FTP	72	Response: 220-
24	2.382294	172.16.10.1	193.137.29.15	TCP	66	50665 → 21 [ACK] Seq=1 Ack=381 Win=29312 Len=0 TSval=49926478 TSecr=309581878
25	2.382299	193.137.29.15	172.16.10.1	FTP	72	Response: 220-
26	2.382307	172.16.10.1	193.137.29.15	TCP	66	50665 → 21 [ACK] Seq=1 Ack=387 Win=29312 Len=0 TSval=49926478 TSecr=309581878
27	2.382890	193.137.29.15	172.16.10.1	FTP	72	Response: 220
28	2.382904	172.16.10.1	193.137.29.15	TCP	66	50665 → 21 [ACK] Seq=1 Ack=393 Win=29312 Len=0 TSval=49926478 TSecr=309581878
29	2.386332	172.16.10.1	193.137.29.15	FTP	71	Request: user
30	2.388133	193.137.29.15	172.16.10.1	TCP	66	21 → 50665 [ACK] Seq=393 Ack=6 Win=29056 Len=0 TSval=309581880 TSecr=49926479
31	2.388152	172.16.10.1	193.137.29.15	FTP	76	Request: anonymous
32	2.389934	193.137.29.15	172.16.10.1	TCP	66	21 → 50665 [ACK] Seq=393 Ack=16 Win=29056 Len=0 TSval=309581880 TSecr=49926480
33	2.389946	193.137.29.15	172.16.10.1	FTP	100	Response: 331 Please specify the password.
34	2.391983	172.16.10.1	193.137.29.15	FTP	71	Request: pass
35	2.431750	193.137.29.15	172.16.10.1	TCP	66	21 → 50665 [ACK] Seq=427 Ack=21 Win=29056 Len=0 TSval=309581891 TSecr=49926481
36	2.431793	172.16.10.1	193.137.29.15	FTP	72	Request: clear
37	2.433631	193.137.29.15	172.16.10.1	TCP	66	21 → 50665 [ACK] Seq=427 Ack=27 Win=29056 Len=0 TSval=309581891 TSecr=49926491
38	2.565651	193.137.29.15	172.16.10.1	FTP	89	Response: 230 Login successful.
39	2.566006	172.16.10.1	193.137.29.15	FTP	71	Request: pasv
40	2.567791	193.137.29.15	172.16.10.1	TCP	66	21 → 50665 [ACK] Seq=450 Ack=32 Win=29056 Len=0 TSval=309581925 TSecr=49926524
41	2.568264	193.137.29.15	172.16.10.1	FTP	117	Response: 227 Entering Passive Mode (193,137,29,15,199,91).
42	2.568523	172.16.10.1	193.137.29.15	TCP	74	54127 → 51035 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=49926525 TSecr=0 WS=128
43	2.570367	193.137.29.15	172.16.10.1	TCP	74	51035 → 54127 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1380 SACK_PERM=1 TSval=309581925 TSecr=49926525 WS=128
44	2.570409	172.16.10.1	193.137.29.15	TCP	66	54127 → 51035 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=49926525 TSecr=309581925
45	2.570441	172.16.10.1	193.137.29.15	FTP	71	Request: retr
46	2.615844	193.137.29.15	172.16.10.1	TCP	66	21 → 50665 [ACK] Seq=501 Ack=37 Win=29056 Len=0 TSval=309581937 TSecr=49926525
47	2.615893	172.16.10.1	193.137.29.15	FTP	85	Request: parrot/README.html
48	2.617659	193.137.29.15	172.16.10.1	TCP	66	21 → 50665 [ACK] Seq=501 Ack=56 Win=29056 Len=0 TSval=309581937 TSecr=49926537
49	2.618787	193.137.29.15	172.16.10.1	FTP-DATA	2802	FTP Data: 2736 bytes
50	2.618808	172.16.10.1	193.137.29.15	TCP	66	54127 → 51035 [ACK] Seq=1 Ack=2737 Win=34688 Len=0 TSval=49926537 TSecr=309581937
51	2.619035	193.137.29.15	172.16.10.1	FTP-DATA	2628	FTP Data: 2562 bytes
52	2.619052	172.16.10.1	193.137.29.15	TCP	66	54127 → 51035 [ACK] Seq=1 Ack=5299 Win=39808 Len=0 TSval=49926537 TSecr=309581937
53	2.619057	193.137.29.15	172.16.10.1	TCP	66	51035 → 54127 [FIN, ACK] Seq=5299 Ack=1 Win=29056 Len=0 TSval=309581937 TSecr=49926525
54	2.619065	193.137.29.15	172.16.10.1	FTP	144	Response: 150 Opening BINARY mode data connection for parrot/README.html (5298 bytes).
55	2.626084	172.16.10.1	193.137.29.15	TCP	66	50665 → 21 [FIN, ACK] Seq=56 Ack=579 Win=29312 Len=0 TSval=49926539 TSecr=309581937
56	2.626114	172.16.10.1	193.137.29.15	TCP	66	54127 → 51035 [FIN, ACK] Seq=1 Ack=5300 Win=39808 Len=0 TSval=49926539 TSecr=309581937

Figura 23: Log da experência 6

6.3 Código Desenvolvido

```
#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 <errno.h>
#include <ctype.h>
#include <regex.h>

#define FTP_PORT  21
#define MAX_STRING_SIZE 256

int verifyURL(char * url);
void getFilename(char *path, char * filename);
char read_reply(int sockfd);
void readArgs(char* args, char* user, char* pass, char* host, char* file, int initState);


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

        struct  hostent *h;
        int             sockfd,sockfd_client;
        struct          sockaddr_in server_addr;
        struct  sockaddr_in client_addr;
```

```
//Verify url with regex
if(verifyURL(argv[1]))
        return -1;

char* user   = malloc(MAX_STRING_SIZE);
char* pass   = malloc(MAX_STRING_SIZE);
char  mode[] = "pasv\n";
char* host   = malloc(MAX_STRING_SIZE);
char* file   = malloc(MAX_STRING_SIZE);


if(strchr(argv[1],'@') != NULL)
        readArgs(argv[1], user, pass, host, file, 0);
else {
        readArgs(argv[1], user, pass, host, file, 2);
        user = "anonymous\n";
        pass = "clear\n";
}

printf("\nUser : %s",user );
printf("pass : %s",pass );
printf("host : %s\n",host );
printf("file : %s\n",file );

setbuf(stdout, NULL);

int        bytes;

if (argc != 2) {
        fprintf(stderr,"usage: getip address\n");
        exit(1);
}

h = gethostbyname(host);

if (h == NULL) {
        herror("gethostbyname");
        exit(1);
}

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

// server address handling
bzero((char*)&server_addr,sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = inet_addr(inet_ntoa(*((struct in_addr
 ↪  *)h->h_addr)));        /*32 bit Internet address network byte ordered*/
server_addr.sin_port = htons(FTP_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(0);
}
// connect to the server
if(connect(sockfd,
        (struct sockaddr *)&server_addr,
        sizeof(server_addr)) < 0){
        perror("connect()");
        exit(0);
}
```

```
    read_reply(sockfd);

// send a user to the server
    write(sockfd,"user ",5);
    bytes = write(sockfd, user, strlen(user));
    printf("Written bytes %d\n", bytes);

    char verify_code = read_reply(sockfd);
    if (verify_code == '4' || verify_code == '5'){
            printf("reply code error.\n");
            close(sockfd);
            return -1;
    }

    // send a pass to the server
    write(sockfd,"pass ",5);
    bytes = write(sockfd, pass, strlen(pass));
    printf("Written bytes %d\n", bytes);

    verify_code = read_reply(sockfd);
    if (verify_code == '4' || verify_code == '5'){
            printf("reply code error.\n");
            close(sockfd);
            return -1;
    }

    // send a mode (pasv) to the server
    bytes = write(sockfd, mode, strlen(mode));
    printf("Written bytes %d\n", bytes);

    char temp = 0;
    int adress = 0;
    char temp_buf[7];
    int index = 0;
    while(read(sockfd,&temp,1)){
            printf("%c", temp);
            if(temp == '.')
                    break;
            if(temp==')')
                    adress=0;
            if (adress >= 4) {
                    temp_buf[index]=temp;
                    index++;
            }
            if(temp == ',')
                    adress++;
    }

    printf("%s\n",temp_buf );

    int second_item = 0;

    for(index = 0; index< strlen(temp_buf); index++){
            if(temp_buf[index]==','){
                    temp_buf[index] = '\0';
                    second_item = index+1;
                    break;
            }
    }

    int second = atoi(&temp_buf[second_item]);
    int first = atoi(&temp_buf[0]);

    printf("First port number = %d\n", first);
    printf("Second port number = %d\n", second);
```

```
        int new_port = 256*first+second;

        bzero((char*)&client_addr,sizeof(client_addr));
        client_addr.sin_family = AF_INET;
        client_addr.sin_addr.s_addr = inet_addr(inet_ntoa(*((struct in_addr
         ↪  *)h->h_addr)));        /*32 bit Internet address network byte ordered*/
        client_addr.sin_port = htons(new_port);


        if ((sockfd_client = socket(AF_INET,SOCK_STREAM,0)) < 0) {
                perror("socket()");
                exit(0);
        }

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

        // send a request to retrieve file to the server
        write(sockfd, "retr ", 5);
        bytes = write(sockfd, file, strlen(file));
        printf("Written bytes %d\n", bytes);


        verify_code = read_reply(sockfd);
        if (verify_code == '4' || verify_code == '5'){
                printf("reply code error.\n");
                close(sockfd);
                close(sockfd_client);
                return -1;
        }

        char filename[MAX_STRING_SIZE];
        getFilename(file, filename);
        FILE *fp;
        fp = fopen(filename,"w");

        temp = 0;
        int read_error = 0;
        while((read_error=read(sockfd_client,&temp,1))){
                fprintf(fp,"%c", temp);
        }

        fclose(fp);
        close(sockfd);
        close(sockfd_client);

        return 0;
}


char read_reply(int sockfd){
        int state=0;
        char temp;
        char reply_code[4];

        while (state!=99) {
                read(sockfd,&temp,1);
                switch(state){
                        case 0://read start, first code char
                        printf("%c",temp);
                        if(isdigit(temp)){
```

```
        reply_code[state]=temp;
        state = 1;
}else{
        if(temp!=13 && temp!=10)
                printf("\nDidn't read a number on first char of reply
                 ↪  code\n");
}
break;

case 1://second code char
printf("%c",temp);
if(isdigit(temp)){
        reply_code[state]=temp;
        state = 2;
}else{
        state=0;
        printf("\nCouldn't read second char of reply code\n");
}
break;

case 2://third code char
printf("%c",temp);
if(isdigit(temp)){
        reply_code[state]=temp;
        state = 3;
}else{
        state=0;
        printf("\nCouldn't read third char of reply code\n");
}
break;

case 3://checking last char of reply code
printf("%c",temp);
if(temp == ' '){
        state = 9;
}else{
        state=4;
}
break;

case 4://waiting for reply code with space
printf("%c",temp);
if(isdigit(temp) && temp == reply_code[state-4]){
        state = 5;
}
break;

case 5://read first reply code char
printf("%c",temp);
if(isdigit(temp) && temp == reply_code[state-4]){
        state = 6;
}else{
        state = 4;
}
break;

case 6://read second reply code char
printf("%c",temp);
if(isdigit(temp) && temp == reply_code[state-4]){
        state = 7;
}else{
        state=4;
}
break;
```

```
                        case 7://read third reply code char
                        printf("%c",temp);
                        if(temp==' '){
                                state = 9;
                        }else{
                                state=4;
                        }
                        break;

                        case 9:
                        printf("%c",temp);
                        if(temp == '\n')
                                state = 99;
                        break;
                }
        }
        return reply_code[0];
}


void readArgs(char* args, char* user, char* pass, char* host, char* file, int initState){

        int state = initState, index = 6, inner_index = 0;
        while(index != strlen(args)) {
                switch (state){
                        case 0:
                        if(args[index] == ':'){
                                user[inner_index] = '\n';
                                user[inner_index + 1] = '\0';
                                state = 1;
                                inner_index = 0;
                        } else {
                                user[inner_index] = args[index];
                                inner_index++;
                        }
                        break;

                        case 1:
                        if(args[index] == '@'){
                                pass[inner_index] = '\n';
                                pass[inner_index + 1] = '\0';
                                state = 2;
                                inner_index = 0;
                        } else {
                                pass[inner_index] = args[index];
                                inner_index++;
                        }
                        break;

                        case 2:
                        if(args[index] == '/'){
                                host[inner_index] = '\0';
                                state = 3;
                                inner_index = 0;
                        } else {
                                host[inner_index] = args[index];
                                inner_index++;
                        }
                        break;
                        case 3:
                        file[inner_index] = args[index];
                        inner_index++;
                        break;
                }
```

```
                index++;
        }

        file[inner_index] = '\n';
        file[inner_index + 1] = '\0';
}


void getFilename(char *path, char * filename){
        int length = strlen(path);
        int i, filenameIndex = 0;
        for(i = 0; i < length; i++) {
                if(path[i] == '/') {
                        memset(filename,0,MAX_STRING_SIZE);
                        filenameIndex = 0;
                }
                else{
                        filename[filenameIndex] = path[i];
                        filenameIndex++;
                }
        }
}

int verifyURL(char * url){
        regex_t regex;
        int reti;
        char msgbuf[100];

        // compile regular expression
        reti = regcomp(&regex, "^ftp://([a-z0-9]+:[a-z0-9]+@)?([\\.a-z0-9-]+)/([\\./a-z0-9-]+)$",
         ↪  REG_EXTENDED|REG_ICASE);
        if (reti) {
                fprintf(stderr, "Could not compile regex\n");
                exit(1);
        }

        // execute regular expression
        reti = regexec(&regex, url , 0, NULL, 0);
        if (!reti) {
                puts("\nValid URL checked!");
                return 0;
        }
        else if (reti == REG_NOMATCH) {
                puts("Please insert a ftp url in the format
                 ↪  ftp://[<username>:<password>@]<hostame>/<file-path>");
                return 1;
        }
        else {
                regerror(reti, &regex, msgbuf, sizeof(msgbuf));
                fprintf(stderr, "Regex match failed: %s\n", msgbuf);
                exit(1);
        }

        // free memory allocated to the pattern buffer by regcomp()
        regfree(&regex);
}
```