Faculdade de Engenharia da Universidade do Porto

Rede de Computadores

T02G01
António Costa Rego
Daniel Cabral Bernardo

Relatório de Projeto realizado no âmbito da
Licenciatura em Engenharia Informática e Computação
Para a cadeira de Redes de Computadores

Porto, 19 de dezembro de 2023

Sumário

Este projeto foi realizado no âmbito da unidade curricular Redes de Computadores com o objetivo de implementar um programa de transferência de dados através do protocolo FTP e configurar uma rede de computadores.

A implementação utilizada requereu explorar conceitos lecionados nas aulas teóricas, e foi crucial para perceber a comunicação realizada numa rede e o estabelecimento da mesma.

Introdução

O trabalho realizado procura criar um programa que, através do protocolo FTP, obtém e transfere ficheiros de um servidor na *internet*. Este trabalho também visa conectar dois computadores em redes diferentes por meio de um computador comum, servindo neste caso de *router*. O relatório possui as seguintes secções:

- **Introdução:** Breve explicação dos objetivos do projeto;
- **Aplicação de *Download*:** Arquitetura da aplicação de *download* e demonstração;
- **Configuração da rede e análise:** Configuração da rede faseado em seis etapas, e análise da informação obtida em cada experiência realizada;
- **Conclusões:** Breve conclusão sobre a implementação e conhecimentos utilizados no desenvolvimento do projeto.

Aplicação de *Download*

Configuração da rede e análise

Experiência 1

Nesta experiência, começa-se a configuração da rede através da conexão dos PCs *tux13* e *tux14* ao *switch* em portas abritrárias através do *port eth0* de cada um dos PCs. Após isto, configura-se o IP do *tux13* com o comando "*ifconfig eth0 172.16.10.1/24*", o que gera o MAC *address* 00:21:5a:5a:7d:16 e automaticamente configura a *gateway* do sistema para que IPs da gama 172.16.10.0, por causa da máscara de 24 bits, sejam redirecionados para 0.0.0.0, ou seja, para o PC que consiga aceitá-lo, o que evita a necessidade de a configurar com o comando *route*, sendo que esta *gateway* já permitirá identificar a gama de IPs onde poderá haver comunicação. No *tux14*, faz-se o mesmo com o comando "*ifconfig eth0 172.16.10.254/24*", o que gera um MAC *address* 00:c0:df:25:13:65. Assim, estando os dois PCs na mesma rede e com *gateways* que permitem conectar-se, o comando "*ping 172.16.10.254*" no *tux13* ou "*ping* 172.16.10.1" no *tux14* sucedem, havendo comunicação entre os dois PCs através do protocolo ICMP, que troca mensagens de *request* e *reply* entre dois computadores para garantir a comunicação.

A tabela ARP é utilizada para associar um IP ao seu respetivo endereço MAC sem necessitar que haja um pedido de *broadcast* primeiro, portanto, ao limpar a tabela ARP, o protocolo ARP será utilizado através do endereço de *broadcast* para identificar o endereço MAC do IP de destino, que é o que acontece na linha 2-3 e 19-20 da captura, uma vez para o *tux13* identificar o *tux14* e outra para o *tux14* saber como responder ao *tux13*, sendo o restante pacotes ICMP, como é identificável na quinta coluna da captura.

Anexos Fig.1 – Captura *Wireshark ping* do *tux14* a partir do *tux13*

O pedido ARP é feito em duas etapas, a primeira que tem como IP e MAC de origem os mesmos do *tux13* e MAC de destino 00:00:00:00:00:00 e IP de destino 172.16.10.254, que seria o chamado na função *ping* e, neste caso, o do *tux14*, sendo daqui redirecionado para o IP de *broadcast*. A partir daqui, há um pedido de *broadcast* na rede para que o PC com aquele IP se identifique, portanto, o *tux14* obtém o pacote ARP e inverte o sentido dos IPs e MACs, ou seja, o destino passa a ser a origem e vice-versa, no entanto, altera o MAC *address*, que seria 00:00:00:00:00:00 ainda, para o seu próprio, e envia-lo para o *tux13*, podendo agora haver comunicação entre os dois PCs.

A qualquer momento, pode-se *pingar* o IP 127.0.0.1/localhost, que seria a *loopback interface*, sendo que este IP redireciona qualquer pedido ao próprio PC, podendo ser utilizado para diagnosticar e resolver problemas de rede, mas é especialmente utilizado para correr servidores na máquina local.

Por fim, um *frame* é constituído por um *header* de 14 bytes, uma sequência de dados de até 1500 bytes e uma *Frame Check Sequence* de 4 bytes, portanto, estes *frames* podem ter até 1518 bytes. No próprio *header* há um campo *length* que determina este tamanho, sendo este utilizado para saber o tamanho exato do *frame*.

Experiência 2

Nesta experiência, conecta-se o *tux12* ao *switch* com a mesma configuração que na primeira experiência, com IP 172.16.11.1/24 e MAC 00:21:5a:5a:7e:51, portanto, este PC irá pertencer a uma rede diferente da previamente definida, mas poderá ainda pedir ao *switch* para encontrar os PCs da outra rede pois estão em gamas diferentes, mas no mesmo *switch*, portanto o *switch* poderá fazer o redirecionamento. Portanto, removeu-se as portas do *switch* que correspondem às conexões aos *tuxes* e adicionou-se o *tux13* e o *tux14* à *bridge10* e o *tux12* à *bridge11*, simulando assim dois *switches* diferentes e redes realmente separadas. Para este fim, conectou-se à consola do *switch* e utilizou-se o comando "*/interface bridge add name=bridge10*" e "*/interface bridge add name=bridge11*" para criar as *bridges*, "*/interface bridge port print*" para identificar os IDs das portas que estão conectadas aos *tuxes*, "*/interface bridge port remove numbers=0,1,9*" para remover as portas da *bridge default* e, por fim, adicionou-se as portas às suas *bridges*, "*/interface bridge port add bridge=bridge10 interface=ether1*", "*/interface bridge port add bridge=bridge10 interface=ether2*", e "*/interface bridge port add bridge=bridge11 interface=ether10*".

A partir do *tux13 pingou-se* o *tux14* e o *tux14*, no entanto, na captura seguinte, só se observa a troca de pacotes com o *tux14*, no caso do *tux12* nem há o início da transferência pois as rotas do *tux13* permitem identificar que não há como chegar à rede do *tux14*.

Anexos Fig.2 – Captura *Wireshark* do *tux13* a *pingar* o *tux14* e, de seguida, o *tux12*

Devido à presença das duas redes, haverão dois domínios de *broadcast*, isto pois ao *pingar* o domínio de *broadcast* da *bridge10*, com o comando "*ping -b 172.16.10.255*", só o *tux13* e *tux14* são afetados por esta chamada. No *tux12* o seu domínio de *broadcast*, atingido através do comando "*ping -b 172.16.11.255*", não permite comunicar com ninguém, portanto, existem estes dois domínios de *broadcast* individuais.

Anexos Fig.3 – Captura *Wireshark* do *ping broadcast 172.16.10.255* apartir do *tux13* no *tux13*
Anexos Fig.4 – Captura *Wireshark* do *ping broadcast 172.16.10.255* apartir do *tux13* no *tux14*
Anexos Fig.5 – Captura *Wireshark* do *ping broadcast 172.16.10.255* apartir do *tux13* no *tux12*

Com estas capturas, embora no *tux13* e *tux14* não haja resposta por algum erro imprevisto, talvez por alguma restrição da rede ou *firewall*, pode-se observar mesmo assim que o *tux13* e *tux14* estão na mesma rede e pertencem ao mesmo domínio de *broadcast*, enquanto que o *tux12* não recebe qualquer *ping*. Esta conclusão é ainda mais consolidada pelo seguinte teste em que se *pinga* o domínio *broadcast* 172.16.11.255 a partir do *tux12*, resultando na identificação dos *pings* no *tux12* e nada no *tux13* e *tux14*.

Anexos Fig.6 - Captura *Wireshark* do *ping broadcast 172.16.11.255* apartir do *tux12* no *tux13*
Anexos Fig.7 - Captura *Wireshark* do *ping broadcast 172.16.11.255* apartir do *tux12* no *tux14*
Anexos Fig.8 - Captura *Wireshark* do *ping broadcast 172.16.11.255* apartir do *tux12* no *tux12*

Experiência 3

Nesta experiência configura-se o *tux14* como um *router* afim de permitir que haja conexão entre os PCs da *bridge10* e *bridge11*. Para este fim, conectou-se o *eth1* do *tux14* à porta 11 do *switch* com o IP 172.16.11.253/24 e MAC 00:c0:df:25:13:65, sendo estes diferentes do *eth0* do *tux14*. Após isto, removeu-se a porta 11 do *switch* da *bridge default* com o comando "*/interface bridge port remove numbers=7*" e adicionou-se à *bridge11* com o comando "*/interface bridge port add bridge=bridge11 interface=ether11*".

Para que este PC possa agir como um *router*, é preciso que se permita o IP *forwarding* e o *broadcasting* de *echoes* ICMP com os comandos "*sysctl net.ipv4.ip_foward=1*" e "*sysctl net.ipv4.icmp_echo_ignore_broadcasts=0*".

Agora, o *tux14* agirá como *router*, no entanto, o *tux12* e *tux13* não o utilizarão como *router*, para isto, terá de haver uma rota que, quando houver um pacote com IP de destino 172.16.10.X no *tux12*, ou 172.16.11.X no *tux13*, o PC saiba enviá-lo para o *tux14*, onde fará o redirecionamento. Para este fim, usou-se o comando "*route add -net 172.16.10.0 netmask 255.255.255.0 gw 172.16.11.253*" no *tux12* e "*route add -net 172.16.11.0 netmask 255.255.255.0 gw 172.16.10.254*" no *tux13*. Poderia também utilizar-se nesta tarefa a *default gateway* para o mesmo fim, mas este método permite garantir que só há este redirecionamento e todo o restante tráfego mantém-se igual.

Anexos Fig.9 – Captura *Wireshark* de, respetivamente, *ping 172.16.10.254*, *ping 172.16.11.253*, *ping 172.16.11.1* a partir do *tux13*

Na figura acima, observa-se então que há conexão do *tux13* a todas as interfaces estabelecidas. A seguir, limpou-se as ARP *tables* e *pingou-se* o *tux12* a partir do *tux13* para observar os pacotes no *tux14*.
Anexos Fig.10 – Captura *Wireshark* do *tux14.eth0* e *tux14.eth1* ao *pingar* o *tux12* a partir do *tux13*

Nesta figura, observa-se então que o *router* utiliza o ARP no *eth0* para informar ao *tux13* como encontrar o *router/tux14.eth0* e, ao próprio *router/tux14.eth0*, como encontrar o *tux13*. No *eth1*, o *router* faz o mesmo, só que para o *tux12*, informando-lhe como encontrar o *tux14/router*, e ao *router/tux14.eth1* como encontrar o *tux12*. Assim, nunca há comunicação direta entre o *tux12* e o *tux13*, mas sim entre o *tux13* ao *tux14* ao *tux12*, o que comprova o redirecionamento realizado pelo *router*, sendo que os próprios MAC *addresses* de destino e origem dos pacotes ICMP são sempre entre o PC e o *router* na própria rede, e os IPs serão os dos dois *tuxes* para que o *router* saiba encontrar o MAC do destino.

Estes pedidos ARP são então guardados na tabela ARP, onde é associado o IP ao MAC da máquina afim de reduzir a necessidade destes pedidos.

Experiência 4

Na tarefa 4, finalmente haverá a conexão à *internet* para todos os *tuxes* através da conexão do *ether1* do *router MikroTik* ao P1.1, que conecta à *internet*, e do *ether2* à porta 12 na *bridge11*. Esta parte foi realizada tal como nas últimas duas experiências, removendo a porta 12 da *bridge default* e adicionando-lhe à *bridge11*. Para alterar os IPs do *ether1* e *ether2*, utilizou-se a consola do *router MikroTik* com os comandos "*/ip address add address=172.16.2.19/24 interface=ether1*" e "*/ip address add address=172.16.11.254/24 interface=ether2*".

De seguida, é necessário definir as *routes* deste *router* para que ao receber qualquer IP, este consiga redirecioná-lo para a *internet*, para este fim usa-se o comando "*/ip route add dst-address=0.0.0.0/0 gateway=172.16.2.254*". No entanto, este *router* só sabe encontrar os PCs na sua gama, portanto, se receber algum pedido à net do *tux13*, não o conseguirá encontrar para responder, portanto, será necessário mais uma rota que permite que o *router* redirecione qualquer pedido a 172.16.10.X para o *tux14*, que sabe encontrar os PCs desta gama. Isto é feito com o comando "*/ip route add dst-address=172.16.10.0/24 gateway=172.16.11.253*". Assim, está terminada a configuração do *router MikroTik*. Porém, os PCs da rede ainda não sabem que os pedidos a IPs desconhecidos como 1.0.0.1 devem ser redirecionados para o *router*.

Portanto, no *tux12* e *tux14* é adicionado uma *default gateway* que, ao receber qualquer IP de destino que não saiba encontrar, ou seja, não esteja nas redes 172.16.10.X ou 172.16.11.X, irá mandar o pedido para o *router MikroTik* e, consequentemente, para a *internet*. Isto é feito com o comando "*route add default gw 172.16.11.254*" nos dois *tuxes*. No caso do *tux13*, não se pode adicionar uma rota diretamente para o *router MikroTik*, mas o *tux14* sabe encontrar este *router*, e é atingível a partir do *tux13*, portanto, no *tux13* será adicionada uma *default gateway* para o *tux14*, removendo a necessidade da rota direta definida previamente. Isto é feito com o comando "*route add default gw 172.16.10.254*" no *tux13*.

Anexos Fig.11 – Captura *Wireshark* dos *pings* do *tux13* ao *tux12*, *tux14.eth1*, *tux14.eth0*, *MikroTik Router eth1*, *MikroTik Router eth2*

Do *tux12*, ao remover a *route* de *tux12* para 172.16.10.X via *tux14*, e utilizando o comando "*traceroute 172.16.10.1*" obtém-se o caminho esperado de *tux12->MikroTik->tux14->tux13*, no entanto, ao correr os comandos "*sysctl net.ipv4.conf.eth0.accept_redirects=0*" e "*sysctl* *net.ipv4.conf.all.accept_redirects=0*", o *tux12* perde a capacidade de conectar-se ao *tux13*, isto pois estes dois comandos definem que o *tux12* utilizará só as suas próprias rotas para chegar ao destino.

Por fim, *pinga-se* o *router* da sala de IP 172.16.2.254 e reconhece-se a conexão, no entanto, se desligar-se a funcionalidade NAT no *router MikroTik* com o comando "*/ip firewall nat disable 0*" e voltar-se a fazer este teste, não haverá resposta. Isto pois o NAT trata de transformar os IPs locais à rede em IPs públicos, portanto, se não houver esta conversão, o *router* não saberá encontrar a máquina de origem pois esta máquina está identificada pelo IP público na *internet*, não pelo IP local que recebeu. Por fim, pode-se voltar a configurar o NAT com o comando "*/ip firewall nat enable 0*" ou adicionando uma nova regra NAT "*/ip firewall nat add chain=srcnat action=masquerade out-interface=ether1*".

Experiência 5

Nesta experiência é necessário configurar um serviço DNS nas máquinas para que estas consigam resolver *hostnames* como *google.com* sem precisar de utilizar diretamente o IP do endereço. Para isto, usa-se o comando "*nano /etc/resolv.conf*" e insere-se "*nameserver 172.16.2.1*" no ficheiro, tornando-se este o domínio que tratará de resolver os *hostnames* e fornecer o IP respetivo.

Anexos Fig.12 – Captura *Wireshark* dos pacotes DNS na conexão ao *domain google.com*

Observa-se então que o PC faz um pedido ao servidor DNS, que é encontrado por meio do *MikroTik Router*, pelo IPv4 e IPv6 do domínio *google.com*, e é retornado o IPv4 e IPv6 do servidor pelo protocolo para que o *ping* possa ocorrer.

Experiência 6

Nesta experiência, ir-se-á confirmar a estabilidade da rede criada ao longo destas experiências e a efetividade da aplicação de *download* FTP descrita e desenvolvida no capítulo anterior.

Anexos Fig.13 - Captura *Wireshark* da transferência do URL ftp://rcom:rcom@netlab1.fe.up.pt/pipe.txt

Anexos Fig. 14 – Captura *Wireshark* da transferência do URL ftp://rcom:rcom@netlab1.fe.up.pt/files/crab.mp4

Após uma *query* DNS para identificar o IP do servidor e um protocolo ARP para identificar o PC que servirá de intermédio entre o *tux13* e a *internet*, nas linhas 7-9 da figura 13 ocorre o *three-way handshake*, em que o cliente envia um pacote SYN, o servidor responde com SYN-ACK, e por fim o cliente responde com o ACK, o que estabelece a conexão. De seguida, existe uma troca de pacotes *FTP Data* que, semelhantemente ao primeiro trabalho, segue um número de sequência incremental para garantir que não há perda de pacotes durante a transferência. No fim há uma troca de pacotes FIN-ACK e assim termina a conexão.

A aplicação utiliza duas ligações TCP, uma na porta 21 para o envio de comandos como "*RETR pipe.txt*" e outra conexão na porta 22 para receber os pacotes do ficheiro requisitado. Este protocolo, com o auxílio do protocolo ARQ, que é o responsável pela troca de mensagens ACK, FIN e SYN, garante a transferência de dados *lossless* com o controlo de fluxo e congestionamento. Estas mensagens de controlo, para além de garantir o sucesso da transferência de cada pacote e do ínicio e fim da transferência, também permitem calcular o tempo de receção de cada pacote, ou seja, o *Round Trip Time*, que poderá ser utilizado para calcular o *timeout* com *Adaptive Retransmission*.

Os pacotes TCP também vêm acompanhados de campos que auxiliam a transferência, como *window size*, que permite ao emissor saber quanta informação pode mandar antes de esperar por um ACK, uma *checksum* para verificar a integridade do pacote, os números de sequência já referidos, utilizados para que o recetor saiba em que ordem reconstruir os dados, e os números de *acknowledgement*, para que o emissor saiba a que pacote a resposta se corresponde.

O controlo de congestionamento é realizado por técnicas como *Additive Increase*, que aumenta a velocidade de transferência linearmente até encontrar congestão, ou seja, três ACKs seguidos ou um *timeout*, e quando isto ocorre, a velocidade de transferência é reduzida exponencialmente. A técnica *Slow Start* permite também encontrar o "*sweet spot*" para quantidade de pacotes transferidos com o aumento gradual e exponencial da quantidade até encontrar congestão, sendo que começa com muitos poucos pacotes a serem transferidos, o dito *Slow Start*. Quando a congestão é atingida por qualquer um destes métodos, a *congestion window* e o *throughput* são reduzidos.

Anexos Fig. 15 – Gráfico de quantidade de pacotes transferidos em função do tempo relativo à figura 14

O gráfico da figura 15 ajuda a confirmar a presença deste controlo, sendo que há um grande pico de velocidade no início da transferência, e logo de seguida há uma caída drástica, sendo que começou a haver congestionamento. Durante o restante gráfico, nota-se estas descidas em casos menores também, mas o mesmo princípio do efeito das técnicas referidas mantém-se, havendo grandes descidas após um pico que gera congestionamento.

No caso de iniciar uma segunda transferência num PC diferente, que seria suposto ser ilustrado pela figura 14 e 15, é esperado que o *throughput* desça drasticamente, pois os dois PCs irão encontrar congestionamento devido ao aumento de tráfego na rede e terão de, através dos métodos referidos, diminuir a velocidade de transferência. Nestas figuras não é notável essa mudança pois o ficheiro utilizado no *tux12* para a transferência não foi grande o suficiente para que haja um congestionamento duradouro.

Conclusões

Este projeto consolidou o conhecimento do protocolo FTP e os restantes protocolos que permitem esta transferência ocorrer propriamente, tal como permitiu perceber como a *network layer*, *data link layer* e a *physical layer* se relacionam através do estabelecimento de uma rede de computadores e a perceção das suas rotas.

Referências

- Documentação RFC 959 - https://www.rfc-editor.org/rfc/rfc959
- Manual Router MikroTik - https://wiki.mikrotik.com/wiki/Manual:TOC

Anexos

1	0.000000000	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:74 Cost = 0 Port = 0x8001
2	0.985720112	HewlettPacka_5a:7d:…	Broadcast	ARP	42	Who has 172.16.10.254? Tell 172.16.10.1
3	0.985878093	KYE_25:13:65	HewlettPacka_5a:7d:…	ARP	60	172.16.10.254 is at 00:c0:df:25:13:65
4	0.985896391	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x18db, seq=1/256, ttl=64 (reply in 5)
5	0.986017147	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x18db, seq=1/256, ttl=64 (request in 4)
6	2.002307975	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:74 Cost = 0 Port = 0x8001
7	2.006761405	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x18db, seq=2/512, ttl=64 (reply in 8)
8	2.006879646	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x18db, seq=2/512, ttl=64 (request in 7)
9	3.030761035	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x18db, seq=3/768, ttl=64 (reply in 10)
10	3.030880394	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x18db, seq=3/768, ttl=64 (request in 9)
11	4.004609106	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:74 Cost = 0 Port = 0x8001
12	4.054773237	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x18db, seq=4/1024, ttl=64 (reply in 13)
13	4.054892456	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x18db, seq=4/1024, ttl=64 (request in 12)
14	5.078759737	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x18db, seq=5/1280, ttl=64 (reply in 15)
15	5.078881540	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x18db, seq=5/1280, ttl=64 (request in 14)
16	5.996924609	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:74 Cost = 0 Port = 0x8001
17	6.102758110	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x18db, seq=6/1536, ttl=64 (reply in 18)
18	6.102904567	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x18db, seq=6/1536, ttl=64 (request in 17)
19	6.145046378	KYE_25:13:65	HewlettPacka_5a:7d:…	ARP	60	Who has 172.16.10.1? Tell 172.16.10.254
20	6.145067889	HewlettPacka_5a:7d:…	KYE_25:13:65	ARP	42	172.16.10.1 is at 00:21:5a:5a:7d:16
21	7.126759625	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x18db, seq=7/1792, ttl=64 (reply in 22)
22	7.126881219	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x18db, seq=7/1792, ttl=64 (request in 21)
23	7.999210654	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:74 Cost = 0 Port = 0x8001
24	8.150759395	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x18db, seq=8/2048, ttl=64 (reply in 25)
25	8.150877567	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x18db, seq=8/2048, ttl=64 (request in 24)
26	9.174773762	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x18db, seq=9/2304, ttl=64 (reply in 27)
27	9.174926225	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x18db, seq=9/2304, ttl=64 (request in 26)

Fig.1 - Captura *Wireshark ping* do *tux14* a partir do *tux13*

32	10.198762497	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x18db, seq=10/2560, ttl=64 (reply in 33)
33	10.198904973	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x18db, seq=10/2560, ttl=64 (request in 32)
34	11.222758565	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x18db, seq=11/2816, ttl=64 (reply in 35)
35	11.222909213	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x18db, seq=11/2816, ttl=64 (request in 34)
36	12.003818992	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:74 Cost = 0 Port = 0x8001
37	12.246866170	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x18db, seq=12/3072, ttl=64 (reply in 38)
38	12.247000614	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x18db, seq=12/3072, ttl=64 (request in 37)
39	13.270760479	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x18db, seq=13/3328, ttl=64 (reply in 40)
40	13.270887102	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x18db, seq=13/3328, ttl=64 (request in 39)
41	14.006113697	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:74 Cost = 0 Port = 0x8001
42	15.998401823	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:74 Cost = 0 Port = 0x8001
43	18.000696388	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:74 Cost = 0 Port = 0x8001
44	20.002988020	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:74 Cost = 0 Port = 0x8001
45	22.005302630	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:74 Cost = 0 Port = 0x8001
46	24.007602015	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:74 Cost = 0 Port = 0x8001
47	26.009898257	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:74 Cost = 0 Port = 0x8001

Fig.2 - Captura *Wireshark* do *tux13* a *pingar* o *tux14* e, de seguida, o *tux12*

1	0.000000000	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x205e, seq=8/2048, ttl=64 (no response found!)
2	0.494284148	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8002
3	1.024012056	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x205e, seq=9/2304, ttl=64 (no response found!)
4	2.047999457	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x205e, seq=10/2560, ttl=64 (no response found!)
5	2.496776673	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8002
6	3.072003970	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x205e, seq=11/2816, ttl=64 (no response found!)
7	4.096007994	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x205e, seq=12/3072, ttl=64 (no response found!)
8	4.498628464	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8002
9	5.120008106	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x205e, seq=13/3328, ttl=64 (no response found!)
10	6.144007032	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x205e, seq=14/3584, ttl=64 (no response found!)

Fig.3 - Captura *Wireshark* do *ping broadcast 172.16.10.255* apartir do *tux13* no *tux13*

18	33.356170223	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x205e, seq=1/256, ttl=64 (no response found!)
19	34.002419931	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8001
20	34.362973392	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x205e, seq=2/512, ttl=64 (no response found!)
21	35.386952280	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x205e, seq=3/768, ttl=64 (no response found!)
22	36.004852992	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8001
23	36.410928094	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x205e, seq=4/1024, ttl=64 (no response found!)
24	37.434895388	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x205e, seq=5/1280, ttl=64 (no response found!)
25	37.997272908	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8001
26	38.458860097	172.16.10.1	172.16.10.255	ICMP	98	Echo (ping) request id=0x205e, seq=6/1536, ttl=64 (no response found!)

Fig.4 - Captura *Wireshark* do *ping broadcast 172.16.10.255* apartir do *tux13* no *tux14*

1	0.000000000	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7f Cost = 0 Port = 0x8001
2	2.002535638	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7f Cost = 0 Port = 0x8001
3	4.004971891	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7f Cost = 0 Port = 0x8001
4	6.007427979	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7f Cost = 0 Port = 0x8001
5	8.009902086	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7f Cost = 0 Port = 0x8001
6	10.012363203	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7f Cost = 0 Port = 0x8001
7	12.014834517	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7f Cost = 0 Port = 0x8001
8	14.017285296	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7f Cost = 0 Port = 0x8001
9	16.019749766	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7f Cost = 0 Port = 0x8001
10	18.022224711	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7f Cost = 0 Port = 0x8001

Fig.5 – Captura *Wireshark* do *ping broadcast 172.16.10.255* apartir do *tux13* no *tux12*

15	18.003317771	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8002
16	20.005879650	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8002
17	22.008423788	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8002
18	24.010963527	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8002
19	26.013519679	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8002
20	28.016083792	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8002
21	30.018651258	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8002
22	32.021590565	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8002
23	34.014075094	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8002
24	36.016515936	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8002
25	38.018943647	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8002

Fig.6 - Captura *Wireshark* do *ping broadcast 172.16.11.255* apartir do *tux12* no *tux13*

9	12.005239214	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8001
10	14.007719488	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8001
11	16.010179298	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8001
12	18.012657336	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8001
13	20.015149204	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8001
14	22.017638277	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8001
15	24.020504424	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8001
16	26.022996431	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8001

Fig.7 - Captura *Wireshark* do *ping broadcast 172.16.11.255* apartir do *tux12* no *tux14*

7	10.035252776	172.16.11.1	172.16.11.255	ICMP	98	Echo (ping) request id=0x75bb, seq=1/256, ttl=64 (no response found!)
8	11.057423469	172.16.11.1	172.16.11.255	ICMP	98	Echo (ping) request id=0x75bb, seq=2/512, ttl=64 (no response found!)
9	12.005041152	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7f Cost = 0 Port = 0x8001
10	12.081400913	172.16.11.1	172.16.11.255	ICMP	98	Echo (ping) request id=0x75bb, seq=3/768, ttl=64 (no response found!)
11	13.105434440	172.16.11.1	172.16.11.255	ICMP	98	Echo (ping) request id=0x75bb, seq=4/1024, ttl=64 (no response found!)
12	13.997557416	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7f Cost = 0 Port = 0x8001
13	14.129407345	172.16.11.1	172.16.11.255	ICMP	98	Echo (ping) request id=0x75bb, seq=5/1280, ttl=64 (no response found!)
14	15.153427113	172.16.11.1	172.16.11.255	ICMP	98	Echo (ping) request id=0x75bb, seq=6/1536, ttl=64 (no response found!)
15	16.000087117	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7f Cost = 0 Port = 0x8001
16	16.177394011	172.16.11.1	172.16.11.255	ICMP	98	Echo (ping) request id=0x75bb, seq=7/1792, ttl=64 (no response found!)
17	17.201392688	172.16.11.1	172.16.11.255	ICMP	98	Echo (ping) request id=0x75bb, seq=8/2048, ttl=64 (no response found!)

Fig.8 - Captura *Wireshark* do *ping broadcast 172.16.11.255* apartir do *tux12* no *tux12*

5	7.999501773	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x2b9b, seq=1/256, ttl=64 (reply in 6)
6	7.999866700	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2b9b, seq=1/256, ttl=64 (request in 5)
7	8.009093439	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8002
8	9.027081518	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x2b9b, seq=2/512, ttl=64 (reply in 9)
9	9.027425422	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2b9b, seq=2/512, ttl=64 (request in 8)
36	25.903882479	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x2ba8, seq=1/256, ttl=64 (reply in 37)
37	25.904244821	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2ba8, seq=1/256, ttl=64 (request in 36)
38	26.028872993	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:7a Cost = 0 Port = 0x8002
39	26.915084391	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x2ba8, seq=2/512, ttl=64 (reply in 40)
40	26.915445616	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2ba8, seq=2/512, ttl=64 (request in 39)
41	27.939083945	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x2ba8, seq=3/768, ttl=64 (reply in 42)
42	27.939443075	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2ba8, seq=3/768, ttl=64 (request in 41)
63	40.904093850	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2bb2, seq=1/256, ttl=64 (reply in 64)
64	40.904599090	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2bb2, seq=1/256, ttl=63 (request in 63)
65	41.923082421	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2bb2, seq=2/512, ttl=64 (reply in 66)
66	41.923592410	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2bb2, seq=2/512, ttl=63 (request in 65)

Fig.9 - Captura *Wireshark* de, respetivamente, *ping 172.16.10.254*, *ping 172.16.11.253*, *ping 172.16.11.1* a partir do *tux13*

No.	Time	Source	Destination	Protocol	Length	Info	
41	22.971030395	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2c7f, seq=5/1280, ttl=64 (reply in 42)	
42	22.971201366	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c7f, seq=5/1280, ttl=63 (request in 41)	
43	23.994987911	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2c7f, seq=6/1536, ttl=64 (reply in 44)	
44	23.995128222	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c7f, seq=6/1536, ttl=63 (request in 43)	
45	24.058944647	HewlettPacka_5a:7d:…	KYE_25:13:65	ARP	60	Who has 172.16.10.254? Tell 172.16.10.1	
46	24.058950584	KYE_25:13:65	HewlettPacka_5a:7d:…	ARP	42	172.16.10.254 is at 00:c0:df:25:13:65	
71	31.105049882	KYE_25:13:65	HewlettPacka_5a:7d:…	ARP	42	Who has 172.16.10.1? Tell 172.16.10.254	eth0
72	31.105316326	HewlettPacka_5a:7d:…	KYE_25:13:65	ARP	60	172.16.10.1 is at 00:21:5a:5a:7d:16	
73	31.130733106	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2c7f, seq=13/3328, ttl=64 (reply in 74)	
74	31.130914903	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c7f, seq=13/3328, ttl=63 (request in 73)	
75	32.154707384	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2c7f, seq=14/3584, ttl=64 (reply in 76)	
76	32.154862501	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c7f, seq=14/3584, ttl=63 (request in 75)	
13	22.702693548	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2c7f, seq=1/256, ttl=63 (reply in 16)	
14	22.702872342	HewlettPacka_5a:7e:…	Broadcast	ARP	60	Who has 172.16.11.253? Tell 172.16.11.1	
15	22.702881631	HewlettPacka_5a:7b:…	HewlettPacka_5a:7e:…	ARP	42	172.16.11.253 is at 00:21:5a:7b:3f	
16	22.702968513	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c7f, seq=1/256, ttl=64 (request in 13)	
17	23.722631628	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2c7f, seq=2/512, ttl=63 (reply in 18)	
18	23.722758809	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c7f, seq=2/512, ttl=64 (request in 17)	eth1
25	26.762489211	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2c7f, seq=5/1280, ttl=63 (reply in 26)	
26	26.762630988	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c7f, seq=5/1280, ttl=64 (request in 25)	
27	27.728467915	HewlettPacka_5a:7b:…	HewlettPacka_5a:7e:…	ARP	42	Who has 172.16.11.1? Tell 172.16.11.253	
28	27.728596214	HewlettPacka_5a:7e:…	HewlettPacka_5a:7b:…	ARP	60	172.16.11.1 is at 00:21:5a:5a:7e:51	
29	27.786437368	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2c7f, seq=6/1536, ttl=63 (reply in 30)	
30	27.786557984	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2c7f, seq=6/1536, ttl=64 (request in 29)	

Fig.10 – Captura *Wireshark* do *tux14.eth0* e *tux14.eth1* ao *pingar* o *tux12* a partir do *tux13*

No.	Time	Source	Destination	Protocol	Length	Info	
7	6.196527995	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x4f27, seq=1/256, ttl=64 (reply in 8)	tux12
8	6.196822236	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x4f27, seq=1/256, ttl=63 (request in 7)	
9	7.216978710	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x4f27, seq=2/512, ttl=64 (reply in 10)	
10	7.217228392	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x4f27, seq=2/512, ttl=63 (request in 9)	
70	43.088932051	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x4f3e, seq=3/768, ttl=64 (reply in 71)	
71	43.089056578	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x4f3e, seq=3/768, ttl=64 (request in 70)	
72	44.040824048	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:78 Cost = 0 Port = 0x8002	tux14.eth1
73	44.112931332	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x4f3e, seq=4/1024, ttl=64 (reply in 74)	
74	44.113053484	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x4f3e, seq=4/1024, ttl=64 (request in 73)	
75	45.136924886	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x4f3e, seq=5/1280, ttl=64 (reply in 76)	
76	45.137088873	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x4f3e, seq=5/1280, ttl=64 (request in 75)	
98	54.928931746	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x4f45, seq=4/1024, ttl=64 (reply in 99)	
99	54.929092870	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x4f45, seq=4/1024, ttl=64 (request in 98)	
100	55.952892894	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x4f45, seq=5/1280, ttl=64 (reply in 101)	tux14.eth0
101	55.953020843	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x4f45, seq=5/1280, ttl=64 (request in 100)	
117	68.684517984	172.16.10.1	172.16.11.254	ICMP	98	Echo (ping) request id=0x4f4f, seq=1/256, ttl=64 (reply in 118)	MikroTik
118	68.684815089	172.16.11.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x4f4f, seq=1/256, ttl=63 (request in 117)	Router
119	69.712950371	172.16.10.1	172.16.11.254	ICMP	98	Echo (ping) request id=0x4f4f, seq=2/512, ttl=64 (reply in 120)	
120	69.713212485	172.16.11.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x4f4f, seq=2/512, ttl=63 (request in 119)	eth1
148	80.956715378	172.16.10.1	172.16.2.19	ICMP	98	Echo (ping) request id=0x4f59, seq=1/256, ttl=64 (reply in 149)	MikroTik
149	80.957045238	172.16.2.19	172.16.10.1	ICMP	98	Echo (ping) reply id=0x4f59, seq=1/256, ttl=63 (request in 148)	Router
150	81.968931736	172.16.10.1	172.16.2.19	ICMP	98	Echo (ping) request id=0x4f59, seq=2/512, ttl=64 (reply in 151)	
151	81.969197622	172.16.2.19	172.16.10.1	ICMP	98	Echo (ping) reply id=0x4f59, seq=2/512, ttl=63 (request in 150)	eth2

Fig.11 – Captura *Wireshark* dos *pings* do *tux13* ao *tux12*, *tux14.eth1*, *tux14.eth0*, *MikroTik Router eth1*, *MikroTik Router eth2*

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/74:4d:28:eb:24:28 Cost = 10 Port = 0x8002
2	0.398689465	172.16.11.1	172.16.2.1	DNS	70	Standard query 0xc6a2 A google.com
3	0.398699592	172.16.11.1	172.16.2.1	DNS	70	Standard query 0xc4ab AAAA google.com
4	0.399440059	172.16.2.1	172.16.11.1	DNS	86	Standard query response 0xc6a2 A google.com A 142.250.184.174
5	0.399490485	172.16.2.1	172.16.11.1	DNS	98	Standard query response 0xc4ab AAAA google.com AAAA 2a00:1450:4003:80c::200e
6	0.399767060	172.16.11.1	142.250.184.174	ICMP	98	Echo (ping) request id=0x1161, seq=1/256, ttl=64 (reply in 7)
7	0.414465142	142.250.184.174	172.16.11.1	ICMP	98	Echo (ping) reply id=0x1161, seq=1/256, ttl=108 (request in 6)
8	0.414573118	172.16.11.1	172.16.2.1	DNS	88	Standard query 0x35a5 PTR 174.184.250.142.in-addr.arpa
9	0.415211546	172.16.2.1	172.16.11.1	DNS	127	Standard query response 0x35a5 PTR 174.184.250.142.in-addr.arpa PTR mad07s23-in-f14.1e100.net
10	1.401360120	172.16.11.1	142.250.184.174	ICMP	98	Echo (ping) request id=0x1161, seq=2/512, ttl=64 (reply in 11)
11	1.415456721	142.250.184.174	172.16.11.1	ICMP	98	Echo (ping) reply id=0x1161, seq=2/512, ttl=108 (request in 10)
12	1.914715186	172.16.11.1	172.16.2.1	DNS	86	Standard query 0x9fe1 PTR 93.243.107.34.in-addr.arpa
13	1.915351169	172.16.2.1	172.16.11.1	DNS	138	Standard query response 0x9fe1 PTR 93.243.107.34.in-addr.arpa PTR 93.243.107.34.bc.googleusercontent.com
14	2.002116137	Routerboardc_2b:84:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/74:4d:28:eb:24:28 Cost = 10 Port = 0x8002
15	2.402310841	172.16.11.1	142.250.184.174	ICMP	98	Echo (ping) request id=0x1161, seq=3/768, ttl=64 (reply in 16)
16	2.416444039	142.250.184.174	172.16.11.1	ICMP	98	Echo (ping) reply id=0x1161, seq=3/768, ttl=108 (request in 15)
17	3.403723428	172.16.11.1	142.250.184.174	ICMP	98	Echo (ping) request id=0x1161, seq=4/1024, ttl=64 (reply in 18)
18	3.417465929	142.250.184.174	172.16.11.1	ICMP	98	Echo (ping) reply id=0x1161, seq=4/1024, ttl=108 (request in 17)

Fig.12 – Captura *Wireshark* dos pacotes DNS na conexão ao *domain google.com*

3	3.307093239	172.16.10.1	193.136.28.10	DNS	76	Standard query 0xa7f8 A netlab1.fe.up.pt
4	3.307907914	KYE_25:13:65	Broadcast	ARP	60	Who has 172.16.10.1? Tell 172.16.10.254
5	3.307929494	HewlettPacka_5a:7d:16	KYE_25:13:65	ARP	42	172.16.10.1 is at 00:21:5a:5a:7d:16
6	3.308060163	193.136.28.10	172.16.10.1	DNS	286	Standard query response 0xa7f8 A netlab1.fe.up.pt A 192.168.109.136 NS cns2.fe.up.pt NS ns1.fe.up.pt NS cns1.fe.up.pt NS ns2.fe.up.pt A 19…
7	3.308180636	172.16.10.1	192.168.109.136	TCP	74	59870 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2549964606 TSecr=0 WS=128
8	3.309055582	192.168.109.136	172.16.10.1	TCP	74	21 → 59870 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2364162017 TSecr=2549964606 WS=128
9	3.309073880	172.16.10.1	192.168.109.136	TCP	66	59870 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2549964607 TSecr=2364162017
10	3.327200238	192.168.109.136	172.16.10.1	FTP	100	Response: 220 Welcome to netlab-FTP server
11	3.327225939	172.16.10.1	192.168.109.136	TCP	66	59870 → 21 [ACK] Seq=1 Ack=35 Win=64256 Len=0 TSval=2549964625 TSecr=2364162035
12	3.327420930	172.16.10.1	192.168.109.136	FTP	77	Request: USER rcom
13	3.328019732	192.168.109.136	172.16.10.1	TCP	66	21 → 59870 [ACK] Seq=35 Ack=12 Win=65280 Len=0 TSval=2364162036 TSecr=2549964625
14	3.328091387	192.168.109.136	172.16.10.1	FTP	100	Response: 331 Please specify the password.
15	3.328172819	172.16.10.1	192.168.109.136	FTP	77	Request: PASS rcom
16	3.328752834	192.168.109.136	172.16.10.1	TCP	66	21 → 59870 [ACK] Seq=69 Ack=23 Win=65280 Len=0 TSval=2364162037 TSecr=2549964626
17	3.338314107	192.168.109.136	172.16.10.1	FTP	89	Response: 230 Login successful.
18	3.338382060	172.16.10.1	192.168.109.136	FTP	72	Request: PASV
19	3.338919194	192.168.109.136	172.16.10.1	TCP	66	21 → 59870 [ACK] Seq=92 Ack=29 Win=65280 Len=0 TSval=2364162047 TSecr=2549964636
20	3.339057546	192.168.109.136	172.16.10.1	FTP	120	Response: 227 Entering Passive Mode (192,168,109,136,169,110).
21	3.339538040	172.16.10.1	192.168.109.136	TCP	74	35672 → 43374 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2549964638 TSecr=0 WS=128
22	3.340127832	192.168.109.136	172.16.10.1	TCP	74	43374 → 35672 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2364162048 TSecr=2549964638 WS=128
23	3.340151298	172.16.10.1	192.168.109.136	TCP	66	35672 → 43374 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2549964638 TSecr=2364162048
24	3.340309414	172.16.10.1	192.168.109.136	FTP	81	Request: RETR pipe.txt
25	3.340808625	192.168.109.136	172.16.10.1	TCP	66	21 → 59870 [ACK] Seq=146 Ack=44 Win=65280 Len=0 TSval=2364162049 TSecr=2549964638
26	3.340951516	192.168.109.136	172.16.10.1	FTP	134	Response: 150 Opening BINARY mode data connection for pipe.txt (1863 bytes).
27	3.341452892	192.168.109.136	172.16.10.1	FTP-DA…	1929	FTP Data: 1863 bytes (PASV) (RETR pipe.txt)
28	3.341477825	172.16.10.1	192.168.109.136	TCP	66	35672 → 43374 [ACK] Seq=1 Ack=1864 Win=63488 Len=0 TSval=2549964639 TSecr=2364162049
29	3.341484110	192.168.109.136	172.16.10.1	TCP	66	43374 → 35672 [FIN, ACK] Seq=1864 Ack=1 Win=65280 Len=0 TSval=2364162049 TSecr=2549964638
30	3.341787143	172.16.10.1	192.168.109.136	TCP	66	35672 → 43374 [FIN, ACK] Seq=1 Ack=1865 Win=64128 Len=0 TSval=2549964640 TSecr=2364162049
31	3.342371907	192.168.109.136	172.16.10.1	TCP	66	43374 → 35672 [ACK] Seq=1865 Ack=2 Win=65280 Len=0 TSval=2364162050 TSecr=2549964640
32	3.342484278	192.168.109.136	172.16.10.1	FTP	90	Response: 226 Transfer complete.
33	3.342640508	172.16.10.1	192.168.109.136	TCP	66	59870 → 21 [ACK] Seq=44 Ack=238 Win=64256 Len=0 TSval=2549964641 TSecr=2364162049
34	3.342667606	172.16.10.1	192.168.109.136	FTP	72	Request: QUIT
35	3.343158925	192.168.109.136	172.16.10.1	TCP	66	21 → 59870 [ACK] Seq=238 Ack=50 Win=65280 Len=0 TSval=2364162051 TSecr=2549964641
36	3.343203692	192.168.109.136	172.16.10.1	FTP	80	Response: 221 Goodbye.
37	3.343236447	192.168.109.136	172.16.10.1	TCP	66	21 → 59870 [FIN, ACK] Seq=252 Ack=50 Win=65280 Len=0 TSval=2364162051 TSecr=2549964641
38	3.343587040	172.16.10.1	192.168.109.136	TCP	66	59870 → 21 [FIN, ACK] Seq=50 Ack=253 Win=64256 Len=0 TSval=2549964642 TSecr=2364162051
39	3.344015504	192.168.109.136	172.16.10.1	TCP	66	21 → 59870 [ACK] Seq=253 Ack=51 Win=65280 Len=0 TSval=2364162052 TSecr=2549964642
40	4.004089842	Routerboardc_2b:84:75	Spanning-tree-(for-b…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:74 Cost = 0 Port = 0x8002
41	6.006181216	Routerboardc_2b:84:75	Spanning-tree-(for-b…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:74 Cost = 0 Port = 0x8002
42	8.008217908	Routerboardc_2b:84:75	Spanning-tree-(for-b…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:84:74 Cost = 0 Port = 0x8002
43	8.471085829	HewlettPacka_5a:7d:16	KYE_25:13:65	ARP	42	Who has 172.16.10.254? Tell 172.16.10.1
44	8.471213007	KYE_25:13:65	HewlettPacka_5a:7d:16	ARP	60	172.16.10.254 is at 00:c0:df:25:13:65

Fig.13 – Captura *Wireshark* da transferência do URL ftp://rcom:rcom@netlab1.fe.up.pt/pipe.txt

1	0.000000000	172.16.10.1	193.136.28.10	DNS	76	Standard query 0x075b A netlab1.fe.up.pt
2	0.000805905	193.136.28.10	172.16.10.1	DNS	286	Standard query response 0x075b A netlab1.fe.up.pt A 192.168.109.136 NS ns2.fe.up.pt NS cns2.fe.up.pt NS ns1.fe.up.pt NS cns1.fe.up.pt A 19…
3	0.000881404	172.16.10.1	192.168.109.136	TCP	74	58386 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2550633635 TSecr=0 WS=128
4	0.001447822	192.168.109.136	172.16.10.1	TCP	74	21 → 58386 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2364831045 TSecr=2550633635 WS=128
5	0.001465422	172.16.10.1	192.168.109.136	TCP	66	58386 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2550633635 TSecr=2364831045
6	0.003095182	192.168.109.136	172.16.10.1	FTP	100	Response: 220 Welcome to netlab-FTP server
7	0.003105239	172.16.10.1	192.168.109.136	TCP	66	58386 → 21 [ACK] Seq=1 Ack=35 Win=64256 Len=0 TSval=2550633637 TSecr=2364831046
8	0.003200085	172.16.10.1	192.168.109.136	FTP	77	Request: USER rcom
9	0.003644488	192.168.109.136	172.16.10.1	TCP	66	21 → 58386 [ACK] Seq=35 Ack=12 Win=65280 Len=0 TSval=2364831047 TSecr=2550633637
10	0.003703505	192.168.109.136	172.16.10.1	FTP	100	Response: 331 Please specify the password.
11	0.003780261	172.16.10.1	192.168.109.136	FTP	77	Request: PASS rcom
12	0.004225922	192.168.109.136	172.16.10.1	TCP	66	21 → 58386 [ACK] Seq=69 Ack=23 Win=65280 Len=0 TSval=2364831048 TSecr=2550633638
13	0.013683000	192.168.109.136	172.16.10.1	FTP	89	Response: 230 Login successful.
14	0.013740131	172.16.10.1	192.168.109.136	FTP	72	Request: PASV
15	0.014264713	192.168.109.136	172.16.10.1	TCP	66	21 → 58386 [ACK] Seq=92 Ack=29 Win=65280 Len=0 TSval=2364831058 TSecr=2550633648
16	0.014530671	192.168.109.136	172.16.10.1	FTP	120	Response: 227 Entering Passive Mode (192,168,109,136,177,194).
17	0.014913823	172.16.10.1	192.168.109.136	TCP	74	45788 → 45506 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2550633649 TSecr=0 WS=128
18	0.015404113	192.168.109.136	172.16.10.1	TCP	74	45506 → 45788 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2364831059 TSecr=2550633649 WS=128
19	0.015415846	172.16.10.1	192.168.109.136	TCP	66	45788 → 45506 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2550633649 TSecr=2364831059
20	0.015440570	172.16.10.1	192.168.109.136	FTP	87	Request: RETR files/crab.mp4
21	0.015863253	192.168.109.136	172.16.10.1	TCP	66	21 → 58386 [ACK] Seq=146 Ack=50 Win=65280 Len=0 TSval=2364831059 TSecr=2550633649
22	0.016014880	192.168.109.136	172.16.10.1	FTP	144	Response: 150 Opening BINARY mode data connection for files/crab.mp4 (88123184 bytes).
23	0.016386858	192.168.109.136	172.16.10.1	FTP-DA…	1514	FTP Data: 1448 bytes (PASV) (RETR files/crab.mp4)
24	0.016397544	172.16.10.1	192.168.109.136	TCP	66	45788 → 45506 [ACK] Seq=1 Ack=1449 Win=64128 Len=0 TSval=2550633650 TSecr=2364831059
25	0.016508942	192.168.109.136	172.16.10.1	FTP-DA…	1514	FTP Data: 1448 bytes (PASV) (RETR files/crab.mp4)
26	0.016517392	172.16.10.1	192.168.109.136	TCP	66	45788 → 45506 [ACK] Seq=1 Ack=2897 Win=64128 Len=0 TSval=2550633650 TSecr=2364831059
27	0.016631304	192.168.109.136	172.16.10.1	FTP-DA…	1514	FTP Data: 1448 bytes (PASV) (RETR files/crab.mp4)
28	0.016638498	172.16.10.1	192.168.109.136	TCP	66	45788 → 45506 [ACK] Seq=1 Ack=4345 Win=64128 Len=0 TSval=2550633651 TSecr=2364831059
29	0.016754017	192.168.109.136	172.16.10.1	FTP-DA…	1514	FTP Data: 1448 bytes (PASV) (RETR files/crab.mp4)
30	0.016761141	172.16.10.1	192.168.109.136	TCP	66	45788 → 45506 [ACK] Seq=1 Ack=5793 Win=64128 Len=0 TSval=2550633651 TSecr=2364831059
31	0.016878056	192.168.109.136	172.16.10.1	FTP-DA…	1514	FTP Data: 1448 bytes (PASV) (RETR files/crab.mp4)
32	0.016885180	172.16.10.1	192.168.109.136	TCP	66	45788 → 45506 [ACK] Seq=1 Ack=7241 Win=64128 Len=0 TSval=2550633651 TSecr=2364831059
33	0.017011524	192.168.109.136	172.16.10.1	FTP-DA…	1514	FTP Data: 1448 bytes (PASV) (RETR files/crab.mp4)
34	0.017018159	172.16.10.1	192.168.109.136	TCP	66	45788 → 45506 [ACK] Seq=1 Ack=8689 Win=64128 Len=0 TSval=2550633651 TSecr=2364831059
35	0.017107486	192.168.109.136	172.16.10.1	FTP-DA…	1514	FTP Data: 1448 bytes (PASV) (RETR files/crab.mp4)
36	0.017114191	172.16.10.1	192.168.109.136	TCP	66	45788 → 45506 [ACK] Seq=1 Ack=10137 Win=64128 Len=0 TSval=2550633651 TSecr=2364831059

Fig. 14 – Captura *Wireshark* da transferência do URL ftp://rcom:rcom@netlab1.fe.up.pt/files/crab.mp4

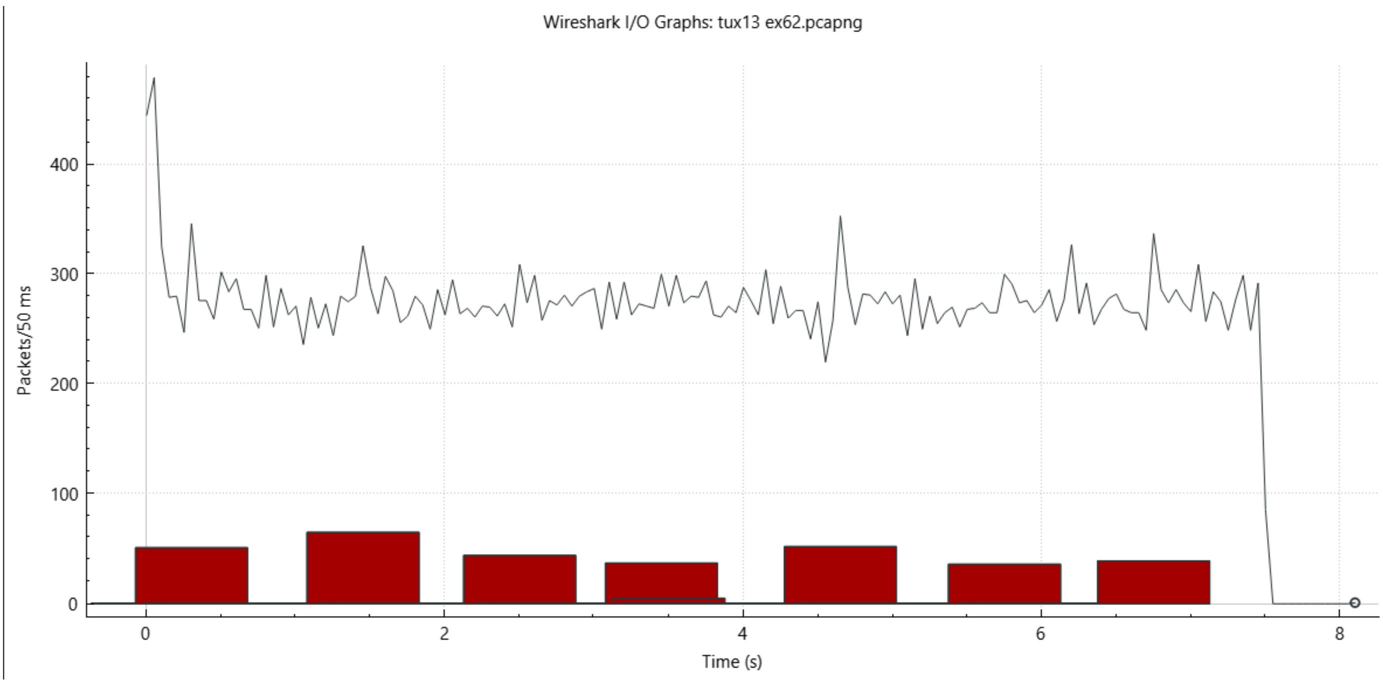

Fig. 15 – Gráfico de quantidade de pacotes transferidos em função do tempo relativo à figura 14