Computer Networks

File Transfer Protocol Project Report

Trabalho realizado por:

Turma 3

- Guilherme Almeida - 202006137
- Pedro Nunes - 202004714

Índice

Parte 1 - Desenvolvimento da aplicação

Para a primeira parte deste trabalho prático, desenvolvemos uma aplicação, com o objetivo de fazer *download* de um único ficheiro arbitrário, implementando o protocolo FTP como descrito no RFC959, com recurso a ligações TCP (Transmission Control Protocol) a partir de *sockets*. O argumento passado à aplicação deve seguir a sintaxe URL, descrita no RFC1738.

Modo de utilização:

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

A especificação da aplicação desenvolvida tem por base os seguintes passos:

- Validação de argumentos e subsequente inicialização de parâmetros relevantes. (***parse_url***)
- Estabelecimento de uma ligação com o servidor através da porta de controlo FTP. (***open_connection***)
- *Login* (***login***)
- Pedido ao servidor para utilização do modo passivo na transferência de dados, e leitura dos valores para identificação da porta a ser usada para transferência de dados. (***download***)
- Estabelecimento de uma ligação com o servidor, através da porta indicada, para transferência de dados. (***open_connection***)
- Pedido do ficheiro indicado e respetiva transferência. (***download***)
- Encerramento das ligações estabelecidas. (***download*** e ***app***)

A comunicação com o servidor é feita através de *sockets* TCP, usando a função ***gethostbyname()*** para obter o IP quando necessário, a porta 21 (porta de controlo FTP), bem como a porta indicada pelo servidor para a transferência.
O *login* é feito usando os parâmetros fornecidos ou, caso estes não sejam fornecidos, é usada uma ligação anónima, de acordo com o protocolo.
Para interpretar as respostas do servidor, são avaliados os códigos (primeiros 3 dígitos da mensagem enviada pelo servidor), havendo apenas a necessidade acrescida de ler também os valores referentes ao IP e porta nas respostas com o código **227**. O ficheiro transferido é guardado localmente com o mesmo nome do ficheiro no servidor.

É possível encontrar em anexo um exemplo de utilização (com sucesso) desta aplicação, com uma ligação anónima. (Figura 17)

Parte 2 - Configuração e estudo de uma rede

Experiência 1

Q1: O que são os pacotes ARP e para que servem?

O protocolo ARP (*Address Resolution Protocol*) é usado para mapear o endereço IP de uma máquina ao seu endereço físico (MAC Address).

Quando determinado computador, numa rede local, tenta enviar pacotes a outro (na mesma rede) e não existem entradas na tabela ARP para o IP destino, será enviado um pacote ARP em modo broadcast para todas as máquinas pertencentes a esta rede, de forma a descobrir qual das máquinas tem o endereço MAC correspondente ao endereço IP do destinatário.

Por fim, a máquina de destino enviará um novo pacote ARP o qual indicará à máquina que envia qual é o seu endereço MAC. Após isto, poderá ser realizada a transmissão de pacotes.

Q2: Quais são os endereços MAC e IP dos pacotes ARP e por quê?

Quando o tux13 tenta enviar um pacote para o tux14, e não conhece o seu MAC Address, isto é, não existe uma entrada na tabela ARP referente ao tux14, será enviado um pacote ARP em broadcast para toda a rede local, o qual contém o endereço IP do tux13 (172.16.10.1 neste caso) e o seu endereço MAC (00:21:5a:61:2d:ef).

Como o endereço MAC do tux14 é desconhecido, será usado o valor genérico 00:00:00:00:00:00.
Podemos ver isto exemplificado nas figuras 1 e 2.

Q3: Que pacotes são gerados pelo comando ping?

Por consequência do comando ping, são inicialmente gerados pacotes ARP para descobrir o endereço MAC do destinatário (caso este seja desconhecido). Após isto, gera pacotes ICMP (Internet Control Message Protocol).

Q4: Quais são os endereços MAC e IP dos pacotes de ping?

Os endereços MAC e IP podem ser encontrados nos pacotes Ping Request e Reply. No nosso caso os endereços encontrados foram:

- Ping Request (Figura 3):
    - Origem: IP - 172.16.10.1, MAC - 00:21:5a:61:2d:ef
    - Destino: IP - 172.16.10.254, MAC - 00:21:5a:61:2f:24
- Ping Reply (Figura 4):
    - Origem: IP - 172.16.10.254, MAC - 00:21:5a:61:2f:24
    - Destino: IP - 172.16.10.1, MAC - 00:21:5a:61:2d:ef

Q5: Como se determina se uma trama Ethernet recebida é ARP, IP, ICMP?

Para determinarmos o tipo da trama recebida, temos de analisar o cabeçalho ethernet (*ethernet header*) da trama.
Nesta trama (Figura 5), os últimos 2 bytes do MAC Header representam o *EtherType*:
Caso este tenha o valor 0x0800, significa que a trama é do tipo IP (como é o caso demonstrado na foto).
No caso de ter o valor 0x0806, então a trama seria do tipo ARP.
No caso de a trama ser do tipo IP, podemos analisar o seu IP header. Caso este tome o valor 1 (Figura 6), então o protocolo usado é ICMP:

Q6: Como determinar o comprimento de uma trama recebida?

O comprimento de uma trama pode ser visualizado de forma direta no Wireshark, através do campo frame.len (ver Figura 7):

Q7: O que é a interface de loopback e qual é a sua importância?

A interface de loopback é uma interface virtual que possibilita à máquina receber respostas de si próprio, seja isto para fins de teste ou para testar a correta configuração da rede.

Experiência 2

Q1: Como configurar a bridgeY0?

De modo a configurar a rede, foram feitas as seguintes configurações:

tux13 >> ifconfig eth0 up
tux13 >> ifconfig eth0 172.16.10.1/24

tux14 >> ifconfig eth0 up
tux14 >> ifconfig eth0 172.16.10.254/24

Configurações no Switch:
Considerações: tux13 e tux14 foram ligados às porta 8 e 9 do switch, respetivamente

Criação da bridge10:
> /interface bridge add name=bridge10

Remoção das portas do tux13 e tux14 conectados à default bridge:
> /interface bridge port remove [find interface=ether8]
> /interface bridge port remove [find interface=ether9]

Adicionar as portas às bridges correspondentes:
> /interface bridge port add bridge=bridge10 interface=ether8
> /interface bridge port add bridge=bridge10 interface=ether9

Q2: Quantos domínios de broadcast existem? Como se conclui isso a partir dos logs?

Existem 2 domínios de broadcast, visto que ao fazer ping em broadcast apenas são abrangidas as portas pertencentes a uma determinada bridge. Assim quando enviamos um ping do tux13 para o endereço de broadcast (172.16.10.255) os pacotes são visualizados nos tux13 e no tux14 não aparecendo os pacotes no tux12 (Ver figura 8). Quando o ping é dado na tux12 para o endereço de broadcast (172.16.11.255) os pacotes apenas são encontrados no tux12 não aparecendo no tux13 nem no tux14, pois estes encontram-se noutro domínio (ver figura 9).

Experiência 3

Q1: Que rotas existem nos tuxes? Qual o significado destas rotas?

Algumas rotas são geradas de forma automática ao ligar as máquinas à bridge: o tux13 tem uma rota para a bridge10 (própria bridge) e o tux12, da mesma forma, para a bridge11. O tux14 tem uma rota para ambas, visto que tem interfaces de rede configuradas nas 2 bridges (a gateway para todas estas rotas é 0.0.0.0).

Para além destas rotas geradas automaticamente, adicionamos manualmente (usando o comando `route add -net <subnet> gw <gateway>`) as rotas:

tux13: rota para a subnet 172.16.11.0/24 através da gateway 172.16.10.254, usada para quando o tux13 quer enviar pacotes para a bridge11, usando como gateway o router (neste caso, o tux14).

tux12: rota para a subnet 172.16.10.0/24 através da gateway 172.16.11.253, usada para quando o tux12 quer enviar pacotes para a bridge10, usando como gateway o router (neste caso, o tux14).

Q2: Quais as informações que contém uma entrada das tabelas de encaminhamento?

Na tabela de encaminhamento (*forwarding table*), cada entrada possui os campos **Destination** (rede/host destino), **Gateway** (o endereço da gateway, ou '*' caso nao haja nenhum) e a **IFace** (Interface de rede, por exemplo: eth0, eth1, ...)

Para além disso, existem outras informações disponíveis em cada entrada da tabela:

- **Genmask** - quantidade de endereços contidos na rede (ex.: a máscara 255.255.255.255 seria apenas para aceder a 1 host, enquanto que a 0.0.0.0 seria a default route)
- **Flags** - Informação sobre a rota (U - up, G - gateway, etc.)
- **Metric** - Custo da rota (em hops)
- **Use** - Número de lookups para a rota
- **Ref** - Número de referências para determinada rota (não usada nos sistemas Linux)

Q3: Quais mensagens ARP e endereços MAC associados são observados e porquê?

Como as tabelas ARP foram limpas, quando o tux13 faz um ping para o tux12, este não conhece o endereço MAC destino.

Utilizando o wireshark, podemos observar que a interface eth0 do tux14 recebe um pedido de broadcast ARP do tux13 para averiguar qual é o MAC Address da interface que tem o IP 172.16.10.254. Sendo esta a interface a eth0, esta dirá ao tux13 o seu MAC Address. O mesmo irá verificar-se no sentido contrário, visto que a interface eth0 também não tem conhecimento do endereço MAC do tux13, como podemos ver na figura 10.

Assim como isto acontece com a interface eth0, e o tux13, também acontecerá com a interface eth1 e o tux12 (como podemos ver na figura 11), visto que será necessário percorrer estes dois "troços" para realizar a conexão entre o tux13 e o tux12.

Q4: Que pacotes ICMP são observados e porquê?

Podemos observar todos os pacotes ICMP do tipo Request e Reply, visto que, estando todas estas rotas adicionadas, todos os tuxes são reconhecíveis entre si, como podemos ver na figura 12. Se este não fosse o caso, os pacotes ICMP enviados iriam conter a mensagem "Host Unreachable".

Q5: Quais são os endereços IP e MAC associados aos pacotes ICMP e porquê?

Os endereços IP e MAC de origem e destino associados aos pacotes ICMP são os das máquinas/interfaces que recebem/enviam os pacotes. Quando um pacote ICMP é enviado do tux13 para a interface eth0 do tux14 (ambos se encontram na mesma subrede), os endereços MAC e IP serão ambos do tux13 (emissor) e os os endereços IP e MAC de destino serão do tux14 (recetor). O mesmo se aplica no caso do tux12 com a interface eth1 do tux14.

Por outro lado, quando isto acontece entre máquinas que não se encontram na mesma rede, que é o caso do tux13 ao enviar um pacote ICMP para o tux12, não é possível realizar esta conexão de forma direta sem que o pacote seja modificado ao passar por uma outra máquina. Neste caso em particular o que acontece é que o pacote que o tux13 envia para o tux12 terá primeiro de passar pela interface eth0 do tux14, visto que o tux14 está ligado a ambas as bridges e consegue interagir de forma direta com o tux12. Os endereços IP serão os do tux13 (172.16.10.1) e do tux12 (172.16.11.1) (origem e destino, respectivamente) e os endereços MAC serão os do tux13 (00:21:5a:61:2d:ef) e da interface eth0 do tux14, como podemos ver na figura 13. Quando recebe este pacote, o tux14 irá enviá-lo para o tux12 mantendo o endereço IP do pacote, mas alterando os endereços MAC, sendo que a origem será a interface eth1 do tux14 e o destino será o tux12.

Experiência 4

Q1: Como configurar uma rota estática num *router* comercial.

Inicialmente é necessário ligar o cabo série de S0 de qualquer um dos tuxes à entrada de configuração do router. Posteriormente, para aceder à interface do router é necessário aceder ao gtkterm, mudar a *baudrate* para 115200 e fazer login. Para configurar uma rota estática devemos executar o seguinte comando:

`/ip route add dst-address=172.16.10.0/24 gateway=172.16.11.253` (Neste caso foi criada uma rota estática para a rede 172.16.10.0/24 através da gateway 172.16.11.253)

Q2: Quais os caminhos seguidos pelos pacotes nas experiências e porquê?

Todos os pacotes que são enviados para as redes que não a 172.16.10.0/24 e 172.16.11.0/24 são encaminhados para o router Rc, uma vez que a default gateway do tux14 e do tux12 é um IP que se encontra no router (172.16.11.254) e a default gateway do tux13 é um IP do tux14 que, por sua vez, encaminhará o tráfego para o router.

No caso do Router ou do tux12 quererem aceder à rede 172.16.10.0/24, fá-lo-ão através de uma rota em que a gateway é 172.16.11.253 e se encontra no tux14.

No caso do tux13 querer aceder à rede 172.16.11.0/24, poderá fazê-lo através da gateway 10.112.10.254 que se encontra no tux14.

Q3: Como configurar NAT num *router* comercial?

De modo a ligar/desligar a NAT num router comercial, necessitamos de conectar a consola do router à porta série do tux e correr o comando:

```
/ip firewall nat <enable/disable> 0
```

Para adicionar regras NAT, podemos correr o seguinte comando:

```
/ip firewall nat add chain=srcnat action=masquerade out-interface=ether1
```

- **Chain** - Especifica a cadeia de regras de firewall na qual a regra de NAT será adicionada. Neste caso, o valor "srcnat" indica que a regra será adicionada na cadeia de NAT de origem, que é responsável por modificar o endereço IP de origem de pacotes enviados pela rede privada para o endereço IP público do router.
- **Action** - Especifica a ação a ser executada pela firewall quando um pacote corresponde à regra de NAT. Neste caso, o valor "masquerade" indica que o endereço IP de origem dos pacotes será modificado para o endereço IP público do router e a porta de origem será alterada para uma porta aleatória disponível.
- **Out-interface** - Especifica a interface de saída pelo qual os pacotes serão enviados. Neste caso, será a interface ether1.

Q4: O que faz uma NAT?

Uma NAT (Network Address Translation) é um mecanismo que permite que dispositivos com endereços IP privados possam comunicar com a Internet ou outras redes públicas através de um IP mascarado, sem a necessidade de ter um endereço IP público individual para cada dispositivo, mas sim através de um IP mascarado.

Ela faz isso ao substituir o endereço IP privado do dispositivo por um endereço IP público fornecido pela sua conexão com a Internet, permitindo que os dispositivos na rede interna comuniquem com dispositivos na Internet de forma transparente. Além disso, a NAT também pode ser usada para ocultar os endereços IP privados da rede interna de dispositivos externos, oferecendo uma camada adicional de segurança para a rede. Esta é muito frequentemente usada em redes domésticas ou empresariais para permitir que vários dispositivos compartilhem uma única conexão com a Internet.

No caso particular deste projeto, a NAT opera num router, o qual estabelece um ponto de ligação entre a rede local e a Internet/ Rede Pública, ou seja, no caso de uma das máquinas querer enviar um pacote para a Internet, como foi feito na experiência, o pacote é enviado para o router e este modifica o endereço “source” para um endereço exterior (“mascarado”), que neste caso é 172.16.1.19.

Experiência 5

Q1: Como configurar o serviço DNS num host?

Para configurar o serviço DNS (Domain Name System) num sistema operativo Linux, é necessário editar o ficheiro de configuração "resolv.conf" no diretório /etc/ do respetivo tux. Deve-se depois adicionar os endereços IP dos servidores DNS que se pretende utilizar na secção "nameserver". Podemos adicionar múltiplos endereços IP, separados por um espaço. Depois devem-se adicionar os endereços IP dos servidores DNS, guardar o ficheiro e reiniciar o serviço DNS com o comando "service network-manager restart" para que as alterações entrem em vigor. Para verificar que o serviço DNS está a funcionar corretamente deve-se tentar aceder a um site na Internet. Se o acesso tiver sucesso, isso significa que o serviço DNS está configurado corretamente.

Exemplo: `echo "nameserver 172.16.1.1" > /etc/resolv.conf`

Q2: Que pacotes são trocados pelo DNS e que informação que informação é que estes contém?

O host envia para o servidor (172.16.1.1) um pacote com o hostname (no nosso caso google.com) , ficando à espera que este retorne o seu IP. O servidor DNS recebe o pacote e verifica na sua base de dados se possui o registro do hostname solicitado. Se o registro existir, o servidor retorna um pacote com o endereço IP correspondente (no caso do trabalho laboratorial foi 142.250.200.78, como mostra a figura 14). Se o registro não existir, o servidor pode consultar outros servidores DNS recursivamente até encontrar o registro ou determinar que o nome não pode ser resolvido. Quando o servidor obtém a resposta, ele retorna um pacote para o host com o endereço IP encontrado ou um código de erro indicando que o nome não pôde ser resolvido.

Os pacotes DNS contém o **nome do domínio** que está a ser consultado, o **tipo de registo** (ex.: A, CNAME, etc.), o **classificador de recurso** (ex.: IN para Internet), o **endereço de IP** correspondente ao nome de domínio, caso encontrado, o **código de erro**, caso o nome não pôde ser resolvido e o **tempo de vida do registo**. Podem também conter outras informações, dependendo do tipo de consulta e da configuração do servidor DNS.

Experiência 6

Q1: Quantas ligações TCP são abertas pelo sua aplicação ftp?

São abertas duas ligações TCP pela nossa aplicação: uma ao entrar em contacto com o servidor, através da qual é efetuado o envio e receção de comandos de modo a preparar a

transferência (*login*, modo passivo, pedido do ficheiro) e outra dedicada à transferência de dados do ficheiro.

Q2: Em que ligação é transportada a informação de controlo do FTP?

A informação de controlo do FTP é transportada através da primeira ligação mencionada, dedicada ao envio e receção de comandos.

Q3: Quais as fases de uma conexão TCP?

Uma conexão TCP pode ser dividida em 3 fases distintas. Primeiro, a conexão é estabelecida, depois são transmitidos os dados e, por fim, a conexão é encerrada.

Q4: Como funciona o mecanismo ARQ TCP? Quais são os campos TCP relevantes? Que informação relevante pode ser observada nos logs?

De modo a lidar com erros na transmissão de dados, o protocolo TCP utiliza um mecanismo de retransmissão de dados ARQ, que funciona através de um método denominado de "janela deslizante". Para este efeito são utilizados números sequenciais, que indicam o número da trama a ser enviada, números *ACK (acknowledge)* , que indicam o correto recebimento da trama, e, implicitamente, das tramas anteriores, e tamanho da janela, que controla o fluxo de dados na rede. Estes valores podem ser encontrados nos logs (SEQ, ACK, WS) como ilustrado na figura 15.

Q5: Como funciona o mecanismo de controlo de congestão do TCP? Quais são os campos relevantes? Como é que o *throughput* de dados evoluiu ao longo do tempo ? Este fluxo está de acordo com o mecanismo de controlo de congestão TCP?

O mecanismo de controlo de congestão TCP tem por base os números *ACK* recebidos na transmissão de pacotes, funcionando estes como *source clock* da transmissão. De modo a regular a janela de transmissão de pacotes, tendo em conta a congestão da conexão, é necessário introduzir uma nova variável por conexão, denominada *CongestionWindow*. Esta variável é regulada, incrementando quando a congestão diminui e decrementando quando a congestão aumenta. Isto é habitualmente feito com recurso a um algoritmo *Additive Increase/Multiplicative Decrease*, da seguinte forma: por cada RTT (*Round Trip Time*) *CongestionWindow* é incrementado em 1, e, quando se verifica que um pacote é perdido, o seu valor passa a metade. Pode haver, ainda, uma fase de *slow start* no início da conexão para determinar rapidamente a capacidade disponível.

Ao iniciar o download, verificamos que a taxa de transmissão aumenta rapidamente, tendo atingido um pico pouco tempo depois. (Figura 16)

Q6: O throughput de uma conexão de dados TCP é afetada pelo aparecimento de uma segunda conexão? Como?

Sim, visto que a taxa de transmissão de pacotes da conexão TCP (a qual já tinha sido iniciada) diminui, dado que o mesmo canal de transmissão de pacotes será partilhado pelas duas conexões, o que implica que a capacidade para transmitir dados da primeira conexão diminua.

Conclusão

Sentimos que todos os objetivos propostos neste projeto foram concluídos com sucesso, sendo que consideramos que a participação de ambos os membros do grupo foi igual, tendo sido todo o projeto realizado em sessões síncronas e durante as aulas laboratoriais.

Para efeito de contagem de páginas, consideramos o limite apenas para o conteúdo do relatório em si, isto é, as respostas às questões das experiências.

Tanto a forma gradual como a configuração da rede foi feita, bem como a repetição da experiência a cada aula, permitiu uma mais fácil compreensão do funcionamento da rede implementada ao longo das aulas práticas.

Por fim, consideramos que este projeto foi extremamente enriquecedor para aprofundar os nossos conhecimentos de redes e que nos deu as bases necessárias para futuramente aprofundarmos os mesmos.

Referências

Durante a realização deste projeto foram consultados os slides das aulas teóricas, o guião das aulas laboratoriais, as man pages de alguns comandos utilizados e RFC's.

Anexos

No.	Time	Source	Destination	Protocol	Length	Info
34	17.982922188	HewlettP_61:2d:ef	HewlettP_61:2f:24	ARP	42	Who has 172.16.10.254? Tell 172.16.10.1
35	17.983021292	HewlettP_61:2f:24	HewlettP_61:2d:ef	ARP	60	172.16.10.254 is at 00:21:5a:61:2f:24

Figura 1 - ARP Request and Reply

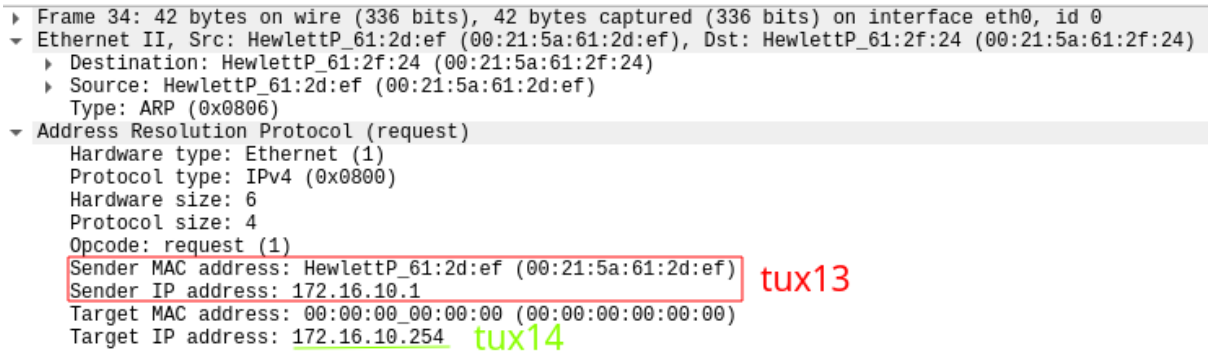

Figura 2 - ARP Request

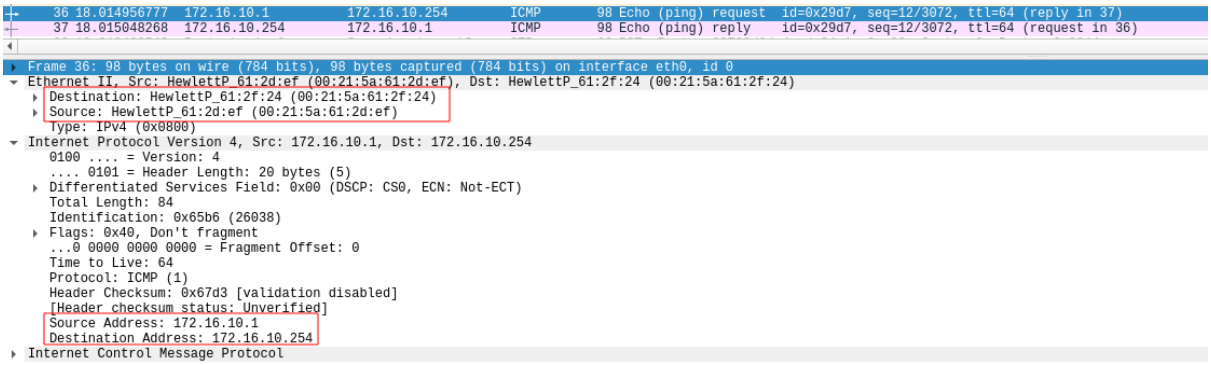

Figura 3 - ARP Request

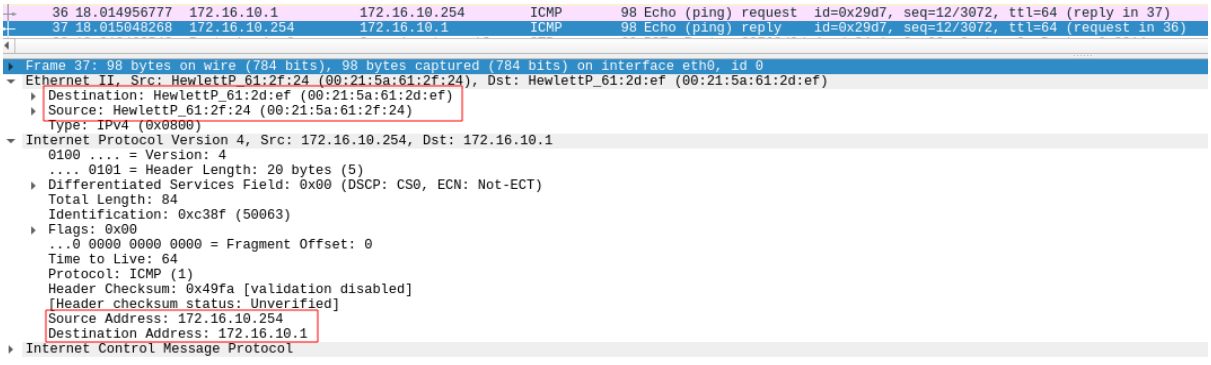

Figura 4 - ARP Reply

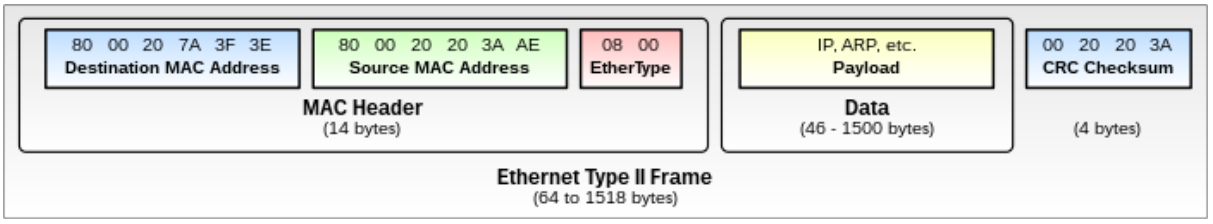

Figura 5 - Trama Ethernet Tipo II

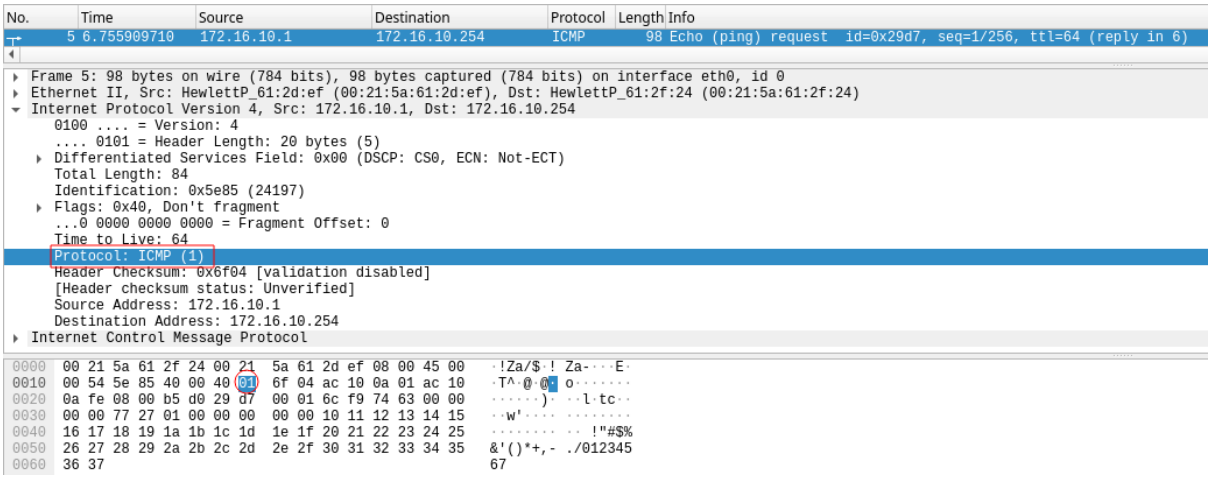

Figura 6 - Trama Ethernet ICMP

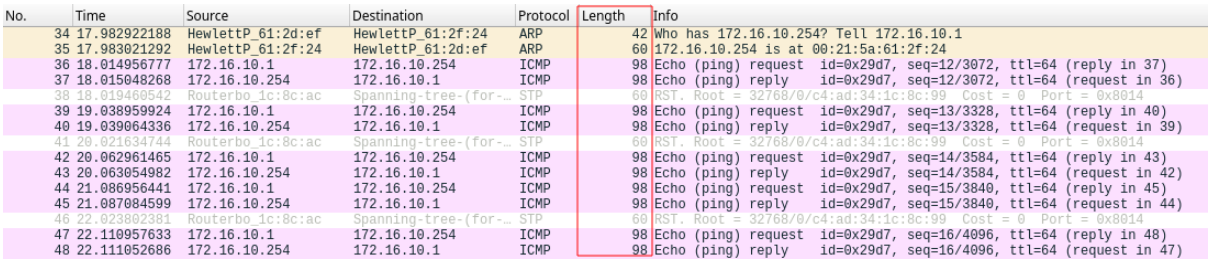

Figura 7 - Tamanho de uma trama recebida

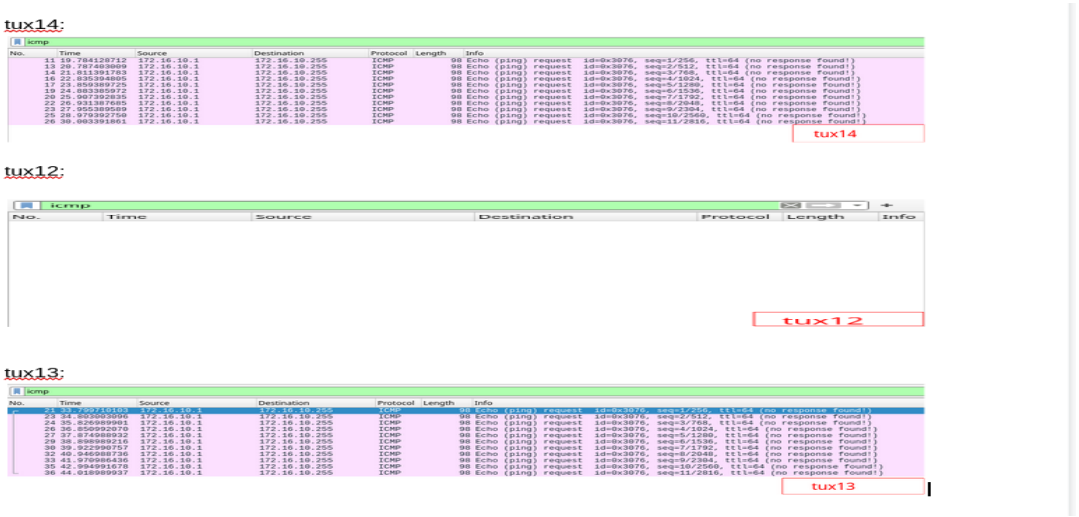

Figura 8 - captura na eth0 dos tuxs, ping enviado de tux13 para o endereço broadcast 172.16.10.255

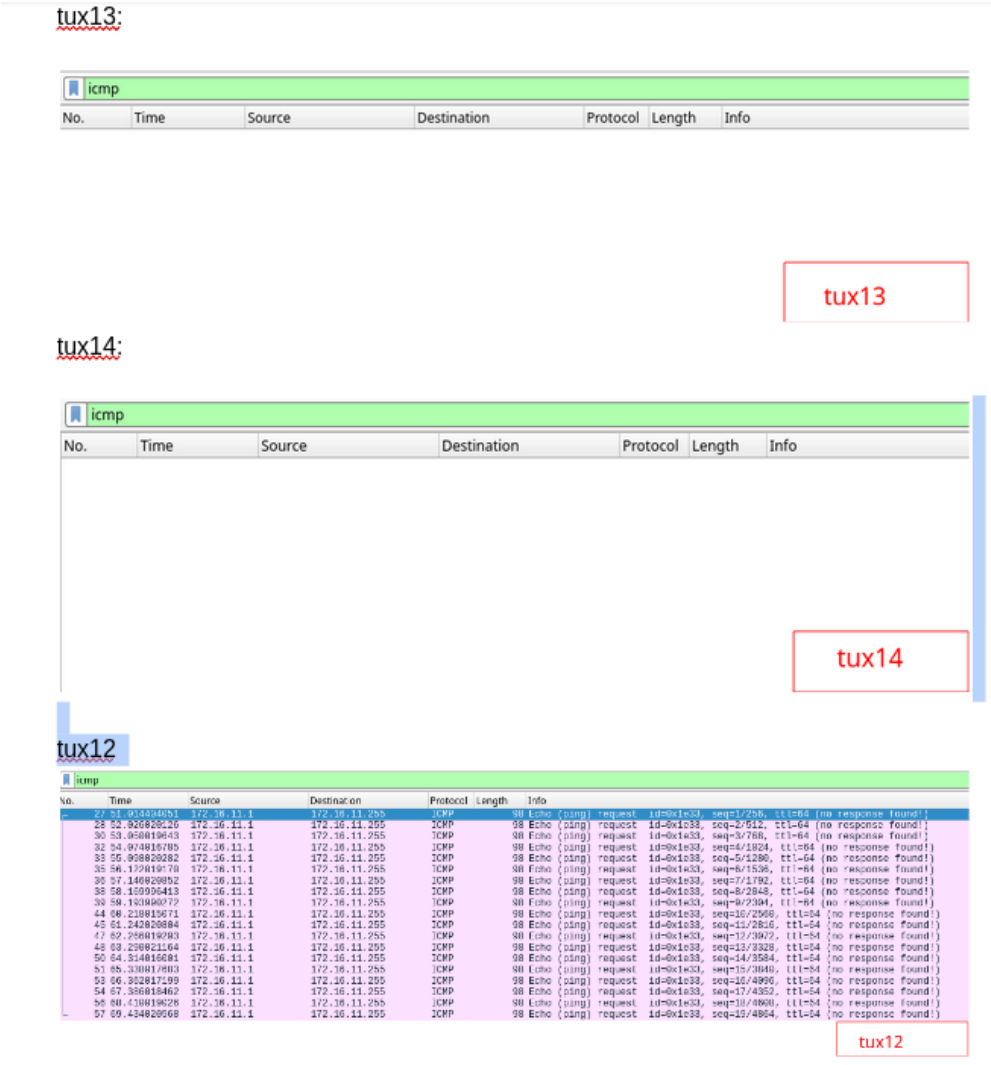

Figura 9 - captura na eth0 dos tuxs, ping enviado de tux12 para o endereço broadcast 172.16.11.255

arp

No.	Time	Source	Destination	Protocol	Lengtl	Info
30	51.078806810	HewlettP_61:2d:ef	Broadcast	ARP	60	Who has 172.16.10.254? Tell 172.16.10.1
31	51.078828391	HewlettP_61:2f:24	HewlettP_61:2d:ef	ARP	42	172.16.10.254 is at 00:21:5a:61:2f:24
45	56.158354845	HewlettP_61:2f:24	HewlettP_61:2d:ef	ARP	42	Who has 172.16.10.1? Tell 172.16.10.254
46	56.158466801	HewlettP_61:2d:ef	HewlettP_61:2f:24	ARP	60	172.16.10.1 is at 00:21:5a:61:2d:ef

Figura 10 - captura dos pacotes ARP na interface eth0 do tux14

arp

No.	Time	Source	Destination	Protocol	Lengtl	Info
29	49.076715204	KYE_04:20:99	Broadcast	ARP	42	Who has 172.16.11.1? Tell 172.16.11.253
30	49.076848950	HewlettP_61:2e:c3	KYE_04:20:99	ARP	60	172.16.11.1 is at 00:21:5a:61:2e:c3
44	54.117026196	HewlettP_61:2e:c3	KYE_04:20:99	ARP	60	Who has 172.16.11.253? Tell 172.16.11.1
45	54.117042539	KYE_04:20:99	HewlettP_61:2e:c3	ARP	42	172.16.11.253 is at 00:c0:df:04:20:99

Figura 11 - captura dos pacotes ARP na interface eth1 do tux14

11	12.774262459	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x2f5c, seq=1/256, ttl=64 (reply in 12)
12	12.774422257	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2f5c, seq=1/256, ttl=64 (request in 11)
13	13.792804222	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x2f5c, seq=2/512, ttl=64 (reply in 14)
14	13.792932032	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2f5c, seq=2/512, ttl=64 (request in 13)
16	14.816820325	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x2f5c, seq=3/768, ttl=64 (reply in 17)
17	14.816944503	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2f5c, seq=3/768, ttl=64 (request in 16)
18	15.840816732	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x2f5c, seq=4/1024, ttl=64 (reply in 19)
19	15.840940422	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2f5c, seq=4/1024, ttl=64 (request in 18)
21	16.864812720	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x2f5c, seq=5/1280, ttl=64 (reply in 22)
22	16.864937947	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2f5c, seq=5/1280, ttl=64 (request in 21)
25	17.888796137	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x2f5c, seq=6/1536, ttl=64 (reply in 26)
26	17.888920316	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2f5c, seq=6/1536, ttl=64 (request in 25)
30	18.912805675	172.16.10.1	172.16.10.254	ICMP	98	Echo (ping) request id=0x2f5c, seq=7/1792, ttl=64 (reply in 31)
31	18.912932298	172.16.10.254	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2f5c, seq=7/1792, ttl=64 (request in 30)
35	24.214156522	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x2f63, seq=1/256, ttl=64 (reply in 36)
36	24.214303749	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2f63, seq=1/256, ttl=64 (request in 35)
37	25.216803480	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x2f63, seq=2/512, ttl=64 (reply in 38)
38	25.216930173	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2f63, seq=2/512, ttl=64 (request in 37)
40	26.240799607	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x2f63, seq=3/768, ttl=64 (reply in 41)
41	26.240920085	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2f63, seq=3/768, ttl=64 (request in 40)
42	27.264800624	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x2f63, seq=4/1024, ttl=64 (reply in 43)
43	27.264925851	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2f63, seq=4/1024, ttl=64 (request in 42)
45	28.288792632	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x2f63, seq=5/1280, ttl=64 (reply in 46)
46	28.288940068	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2f63, seq=5/1280, ttl=64 (request in 45)
47	29.312814252	172.16.10.1	172.16.11.253	ICMP	98	Echo (ping) request id=0x2f63, seq=6/1536, ttl=64 (reply in 48)
48	29.312938291	172.16.11.253	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2f63, seq=6/1536, ttl=64 (request in 47)
52	35.070305091	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2f6a, seq=1/256, ttl=64 (reply in 53)
53	35.070601080	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2f6a, seq=1/256, ttl=63 (request in 52)
55	36.096811403	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2f6a, seq=2/512, ttl=64 (reply in 56)
56	36.097052497	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2f6a, seq=2/512, ttl=63 (request in 55)
57	37.120802921	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2f6a, seq=3/768, ttl=64 (reply in 58)
58	37.121055469	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2f6a, seq=3/768, ttl=63 (request in 57)
60	38.144807710	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2f6a, seq=4/1024, ttl=64 (reply in 61)
61	38.145084074	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2f6a, seq=4/1024, ttl=63 (request in 60)
62	39.168802720	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2f6a, seq=5/1280, ttl=64 (reply in 63)
63	39.169077408	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2f6a, seq=5/1280, ttl=63 (request in 62)
65	40.192805832	172.16.10.1	172.16.11.1	ICMP	98	Echo (ping) request id=0x2f6a, seq=6/1536, ttl=64 (reply in 66)
66	40.193027789	172.16.11.1	172.16.10.1	ICMP	98	Echo (ping) reply id=0x2f6a, seq=6/1536, ttl=63 (request in 65)

Figura 12 - captura dos pacotes ICMP no tux13

57 37.120802921 172.16.10.1 172.16.11.1 ICMP 98 Echo (ping) request id=0x2f6a, seq=3/768, ttl=64 (reply in 58)

```
Frame 57: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth0, id 0
Ethernet II, Src: HewlettP_61:2d:ef (00:21:5a:61:2d:ef), Dst: KYE_04:20:99 (00:c0:df:04:20:99)
    Destination: KYE_04:20:99 (00:c0:df:04:20:99)        tux14
    Source: HewlettP_61:2d:ef (00:21:5a:61:2d:ef)
    Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 172.16.10.1, Dst: 172.16.11.1        tux12
Internet Control Message Protocol
```

Figura 13 - ping entre tux13 e tux12 (captura no tux13)

dns

No.	Time	Source	Destination	Protocol	Length	Info
2	1.723551541	172.16.51.1	172.16.2.1	DNS	70	Standard query 0x66a8 A google.com
3	1.723561738	172.16.51.1	172.16.2.1	DNS	70	Standard query 0xd5b1 AAAA google.com
4	1.724391261	172.16.2.1	172.16.51.1	DNS	86	Standard query response 0x66a8 A google.com A 142.250.200.78
5	1.724405300	172.16.2.1	172.16.51.1	DNS	98	Standard query response 0xd5b1 AAAA google.com AAAA 2a00:1450:4003:80d::200e
8	1.742786891	172.16.51.1	172.16.2.1	DNS	87	Standard query 0x5aac PTR 78.200.250.142.in-addr.arpa
9	1.743386562	172.16.2.1	172.16.51.1	DNS	126	Standard query response 0x5aac PTR 78.200.250.142.in-addr.arpa PTR mad07s24-in-f14.1e100.net

Figura 14 - Pesquisa DNS

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Routerbo_1c:95:ca	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8001
2	2.002077250	Routerbo_1c:95:ca	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:ca Cost = 0 Port = 0x8001
3	3.050076460	172.16.50.1	172.16.2.1	DNS	86	Standard query 0x2cf9 PTR 26.114.82.140.in-addr.arpa
4	3.050816771	172.16.2.1	172.16.50.1	DNS	131	Standard query response 0x2cf9 PTR 26.114.82.140.in-addr.arpa PTR lb-140-82-114-26-iad.github.com
5	3.050967766	172.16.50.1	172.16.2.1	DNS	86	Standard query 0x631f PTR 194.26.161.35.in-addr.arpa
6	3.160669286	172.16.2.1	172.16.50.1	DNS	149	Standard query response 0x631f PTR 194.26.161.35.in-addr.arpa PTR ec2-35-161-26-194.us-west-2.compute.amazonaws.com
7	3.681110511	172.16.50.1	172.16.2.1	DNS	69	Standard query 0x42ac A ftp.up.pt
8	3.681936027	172.16.2.1	172.16.50.1	DNS	107	Standard query response 0x42ac A ftp.up.pt CNAME mirrors.up.pt A 193.137.29.15
9	3.682016693	172.16.50.1	193.137.29.15	TCP	74	36340 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=356591175 TSecr=0 WS=128
10	3.685510822	193.137.29.15	172.16.50.1	TCP	74	21 → 36340 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1380 SACK_PERM=1 TSval=2531393048 TSecr=356591175 WS=128
11	3.685528561	172.16.50.1	193.137.29.15	TCP	66	36340 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=356591179 TSecr=2531393048
12	3.691836151	193.137.29.15	172.16.50.1	FTP	139	Response: 220-Welcome to the University of Porto's mirror archive (mirrors.up.pt)
13	3.691860734	172.16.50.1	193.137.29.15	TCP	66	36340 → 21 [ACK] Seq=1 Ack=74 Win=64256 Len=0 TSval=356591185 TSecr=2531393052
14	3.691879731	193.137.29.15	172.16.50.1	FTP	141	Response: 220---
15	3.691887274	172.16.50.1	193.137.29.15	TCP	66	36340 → 21 [ACK] Seq=1 Ack=149 Win=64256 Len=0 TSval=356591185 TSecr=2531393052
16	3.691952505	193.137.29.15	172.16.50.1	FTP	310	Response: 220-All connections and transfers are logged. The max number of connections is 200.
17	3.691960118	172.16.50.1	193.137.29.15	TCP	66	36340 → 21 [ACK] Seq=1 Ack=393 Win=64128 Len=0 TSval=356591185 TSecr=2531393052
18	3.692742543	172.16.50.1	193.137.29.15	FTP	81	Request: user anonymous
19	3.694545689	193.137.29.15	172.16.50.1	TCP	66	21 → 36340 [ACK] Seq=393 Ack=16 Win=65280 Len=0 TSval=2531393057 TSecr=356591186
20	3.697444565	193.137.29.15	172.16.50.1	FTP	100	Response: 331 Please specify the password.
21	3.697479416	172.16.50.1	193.137.29.15	FTP	72	Request: pass
22	3.701095346	193.137.29.15	172.16.50.1	FTP	89	Response: 230 Login successful.
23	3.701133619	172.16.50.1	193.137.29.15	FTP	71	Request: pasv
24	3.704425349	193.137.29.15	172.16.50.1	FTP	117	Response: 227 Entering Passive Mode (193,137,29,15,225,91).
25	3.704491278	172.16.50.1	193.137.29.15	TCP	74	47600 → 57691 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=356591198 TSecr=0 WS=128
26	3.708141012	193.137.29.15	172.16.50.1	TCP	74	57691 → 47600 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1380 SACK_PERM=1 TSval=2531393070 TSecr=356591198 WS=128
27	3.708161335	172.16.50.1	193.137.29.15	TCP	66	47600 → 57691 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=356591202 TSecr=2531393070
28	3.708182707	172.16.50.1	193.137.29.15	FTP	94	Request: retr pub/kodi/timestamp.txt
29	3.710046544	193.137.29.15	172.16.50.1	FTP-DA…	77	FTP Data: 11 bytes (PASV) (retr pub/kodi/timestamp.txt)
30	3.710070081	172.16.50.1	193.137.29.15	TCP	66	47600 → 57691 [ACK] Seq=1 Ack=12 Win=64256 Len=0 TSval=356591203 TSecr=2531393072
31	3.710075109	193.137.29.15	172.16.50.1	TCP	66	57691 → 47600 [FIN, ACK] Seq=12 Ack=1 Win=65280 Len=0 TSval=2531393072 TSecr=356591202
32	3.710141807	193.137.29.15	172.16.50.1	FTP	146	Response: 150 Opening BINARY mode data connection for pub/kodi/timestamp.txt (11 bytes).
33	3.753627606	172.16.50.1	193.137.29.15	TCP	66	36340 → 21 [ACK] Seq=55 Ack=581 Win=64128 Len=0 TSval=356591247 TSecr=2531393072
34	3.753638012	172.16.50.1	193.137.29.15	TCP	66	47600 → 57691 [ACK] Seq=1 Ack=13 Win=64256 Len=0 TSval=356591247 TSecr=2531393072
35	3.756831197	193.137.29.15	172.16.50.1	FTP	90	Response: 226 Transfer complete.
36	3.756847400	172.16.50.1	193.137.29.15	TCP	66	36340 → 21 [ACK] Seq=55 Ack=605 Win=64128 Len=0 TSval=356591250 TSecr=2531393118
37	3.756902225	172.16.50.1	193.137.29.15	TCP	66	47600 → 57691 [FIN, ACK] Seq=13 Win=64256 Len=0 TSval=356591250 TSecr=2531393072
38	3.756918288	172.16.50.1	193.137.29.15	TCP	66	36340 → 21 [FIN, ACK] Seq=55 Ack=605 Win=64128 Len=0 TSval=356591250 TSecr=2531393118
39	3.760198355	193.137.29.15	172.16.50.1	TCP	66	57691 → 47600 [ACK] Seq=13 Ack=2 Win=65280 Len=0 TSval=2531393122 TSecr=356591250
40	3.760445591	193.137.29.15	172.16.50.1	TCP	66	21 → 36340 [FIN, ACK] Seq=605 Ack=56 Win=65280 Len=0 TSval=2531393122 TSecr=356591250
41	3.760456835	172.16.50.1	193.137.29.15	TCP	66	36340 → 21 [ACK] Seq=56 Ack=606 Win=64128 Len=0 TSval=356591254 TSecr=2531393122

Figura 15 - Aplicação de *download (*)*

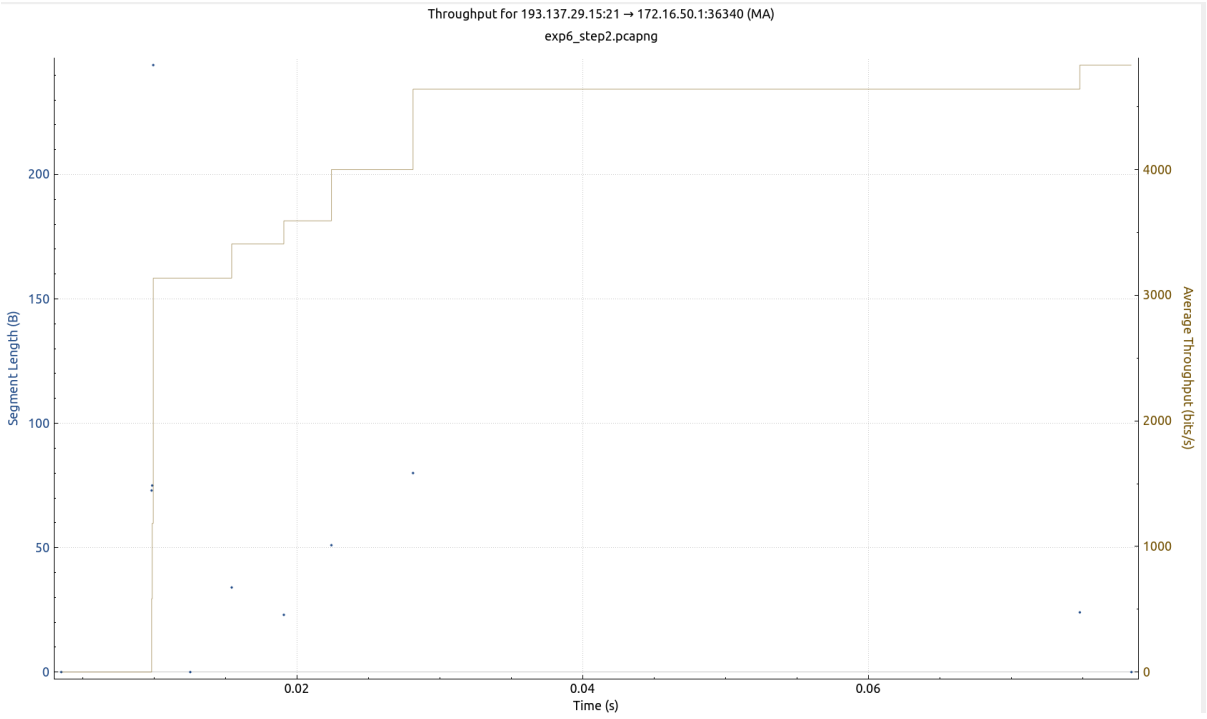

Figura 16 - Taxa de transmissão *

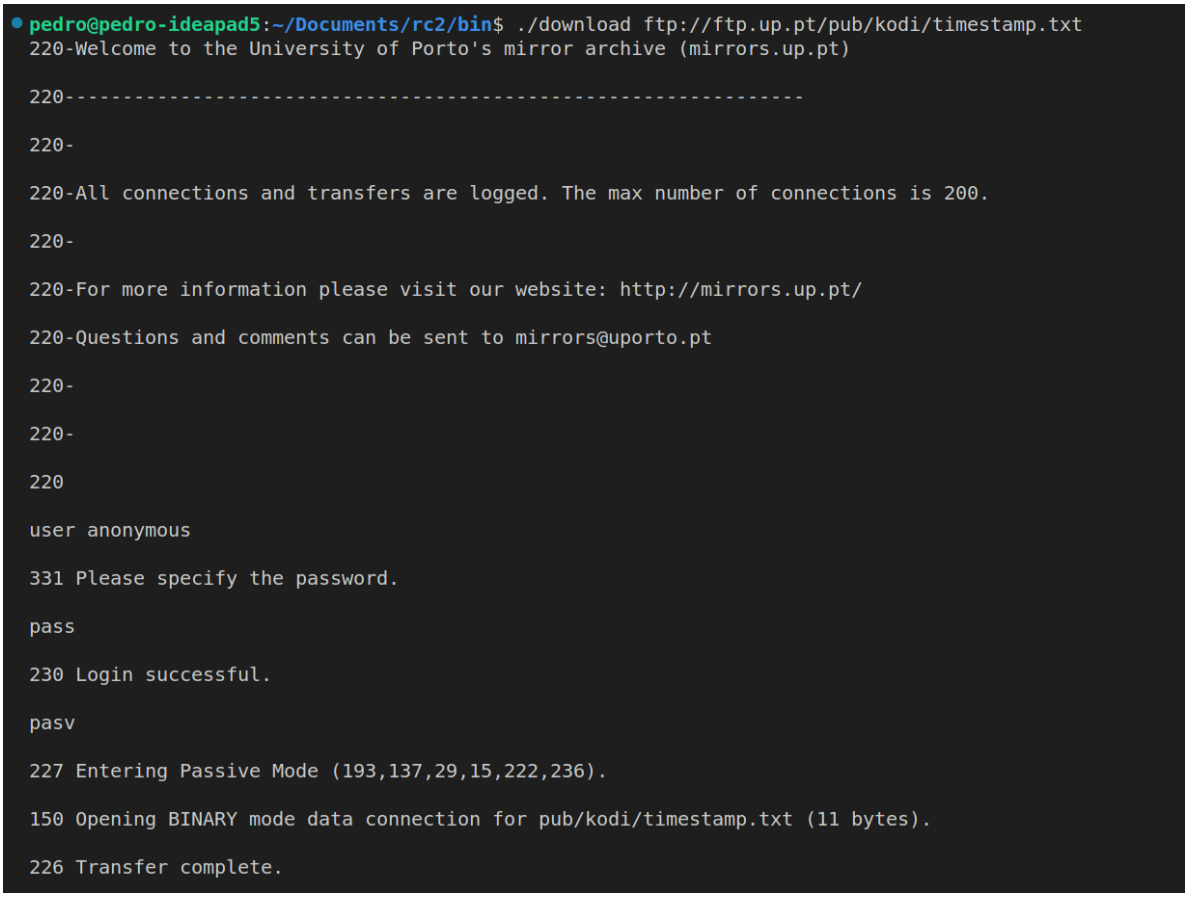

Figura 17 - Exemplo de *download* (com sucesso) usando a aplicação

(*) - Estas capturas foram realizadas na bancada 5. Por motivos de uniformização no relatório optamos por usar sempre os IP's da bancada 1