Redes de Computadores

2º Trabalho Laboratorial
"Redes de Computadores"

António Cunha Seco Fernandes de Almeida - up201505836@fe.up.pt
Diogo Luis Rey Torres - up201506428@fe.up.pt
João Paulo Madureira Damas - up201504088@fe.up.pt

Turma 1 / Bancada 5

Data da Demonstração: 15 de dezembro de 2017
Data de Entrega: 22 de dezembro de 2017

Sumário

Este projeto teve como objetivo o desenvolvimento de uma aplicação de download FTP e a configuração e estudo de uma rede. Assim, foram realizadas diversos experimentos na rede de acordo com o abordado nas aulas teóricas (Mac Sublayer, Network Layer e Transport Layer) de forma a concretizar o objetivo esperado.

As experiências acima referidas baseiam-se na configuração de um IP de rede, de um router em Linux, de um router comercial e do DNS, e na implementação de duas LAN's virtuais no switch e do NAT e ainda num teste com a aplicação de download desenvolvida para a verificação de um bom funcionamento nas ligações TCP.

A realização deste projeto/relatório permitiu-nos aplicar os conhecimentos teóricos previamente adquiridos numa vertente mais prática, e assim possibilitou uma melhor aprendizagem por parte do grupo nestes conteúdos.

Índice

Introdução

Este segundo trabalho laboratorial encontra-se divididos em duas partes: a primeira corresponde ao desenvolvimento de uma aplicação de download e a segunda refere-se à configuração e estudo de uma rede de computadores.

A primeira parte tem como objetivos o download de um ficheiro a partir desta aplicação aplicando um protocolo FTP baseado no RFC959 e adotando uma sintaxe de URL descrita no RFC1738.

Na segunda parte, os objetivos consistiam em configurar uma rede de computadores. Assim, ao longo das aulas laboratoriais realizaram-se várias experiências relacionadas com a configuração de um IP de rede, a implementação de duas LANs virtuais num switch, a configuração de um router em Linux, a configuração de um router comercial e a implementação de uma NAT, a configuração do DNS e por fim o estabelecimento de conexões TCP.

Este relatório encontra-se dividido em diversas secções retratando diferentes partes do projeto:

●**Aplicação de Download**, onde são apresentados a arquitetura e o protocolo implementados da aplicação assim como os resultados de uma transferência com sucesso

●**Análise e Configuração da rede**, referindo a arquitetura da rede, os objetivos principais assim como alguns conceitos necessários para o entendimento das experiências, os comandos usados na configuração e por fim, a análise dos logs retirados em cada experiência

●**Conclusões**, elaborando uma síntese das secções apresentadas e uma breve reflexão sobre os objectivos alcançados

Parte 1 - Aplicação de Download

Arquitetura

Para uma melhor estruturação, a aplicação foi dividida em duas partes: parser, que recebe um URL como especificado no RFC1738, protocolo://[username:password@]host:path, e recolhe as diferentes componentes fornecidas no mesmo, e client, que implementa o protocolo FTP segundo o RFC959. O código fonte de cada camada encontra-se no Anexo I.

Parser

A camada parser foi implementada segundo uma função: parseURL(URLInfo* url, const char* urlStr). O primeiro argumento é uma struct onde serão guardadas as informações obtidas ao longo da função. urlStr representa o url como introduzido na linha de comandos.

O URL é primeiramente testado com uma simples expressão regular que determina se este se encontra no formato especificado pelo RFC correspondente e é, então, válido. Apenas depois é feito o seu processamento, cujo mecanismo se baseia num tamanho acumulado (cumulativeLength, como presente no código) ao qual é somado o tamanho da próxima componente a ser lida, com o intuito de depois obter a substring do URL a começar nesse índice e terminada pelo delimitador do elemento atual (ex: no caso do username, :, para password, @, …). Este processo é repetido até todos os componentes tiverem sido obtidos.

Client

Na camada client foram implementadas as chamadas necessárias de forma a poder transferir um ficheiro pelo protocolo FTP. Na figura abaixo encontram-se os cabeçalhos das funções definidas.

```
int connectSocket(const char *ip, int port);
int connectServer(FTPInfo* client, const char* ip, int port);
int login(FTPInfo* client, const char* username, const char* password);
int download(FTPInfo* client, const char* fileName);
int disconnectServer(FTPInfo* client);

int readMessage(FTPInfo* client, char* message, size_t messageSize);
int sendMessage(FTPInfo* client, const char* message, size_t messageSize);
int handleFTPReplyCode(char* reply_code);
int parseFTPReplyCode(char* reply);
int sendCommand(FTPInfo* client, char* command);

int setCWD(FTPInfo* client, const char* path);
int setPassiveMode(FTPInfo* client);
int requestRETR(FTPInfo* client, const char* fileName);
```

Figura 1 - Cabeçalhos das funções na camada client

Na função main (disponível no código fonte, Anexo I), após o parser ser executado, é executada a sequência de chamadas connectServer, login, setCWD, setPassiveMode, download, disconnectServer, correspondendo à sequência de chamadas necessárias para transferir um ficheiro do servidor especificado.

Resultado do download de um ficheiro com sucesso

Durante a apresentação, realizamos o download de 4 ficheiros de tipos diferentes (png, txt e zip), utilizando o modo anónimo e com user e password demonstrando assim que a nossa aplicação é versátil e consegue lidar com a generalidade dos downloads via FTP.

Para efeitos de relatório vamos só referir um dos downloads, deixando alguns prints de downloads com sucesso na parte dos anexos.

Para efectuar o download corremos o seguinte comando: ./download ftp://demo:password@test.rebex.net/pub/example/mail-editor.png. Assim, mal é executada a aplicação através da função **connectServer** é aberta uma conexão de controlo recorrendo ao endereço IP (obtido através do URL na componente parser.c) e à porta 21(default). Posteriormente, é chamada a função **login** que envia os comandos USER “demo” e a PASS “password” para o servidor. De seguida, é mudado a current working directory para o path onde se encontra o ficheiro que se deseja fazer o download, usando a função **setCWD**. Consequentemente, é chamada a função **setPassiveMode** que ativa o modo passivo da ligação FTP e como tal, é necessário que o cliente abra uma ligação de dados TCP existindo apenas ligações TCP de saída(dados e controlo), esta função envia o comando PASV e é retornado o endereço IP e a porta a utilizar. Após a interpretação do que é retornado, é estabelecida a ligação e começa-se o **download**. O cliente inicia a transferência enviando o comando RETR, começando de seguida o download e o respetivo armazenamento do ficheiro. Terminado o download, é fechada a conexão de dados TCP. Por último, o cliente procede ao fecho da

sessão FTP usando a função **disconnectServer**, a qual envia o comando QUIT e procede ao fecho da conexão de controlo.

O sistema FTP funciona com base em que para cada comando enviado por parte do cliente, é obtida sempre uma resposta. Na nossa aplicação, também lidamos com respostas positivas e negativas, e são efectuados os mecanismos necessários para proceder ao seu tratamento.

```
                                $ ./download ftp://demo:password@test.rebex.net/p
ub/example/mail-editor.png

The IP received to test.rebex.net was 195.144.107.198
220 Microsoft FTP Service
Info: USER demo

331 Password required for demo.
Info: PASS password

230 User logged in.
Info: CWD pub/example

250 CWD command successful.
Info: PASV

227 Entering Passive Mode (195,144,107,198,4,3).
IP: 195.144.107.198
PORT: 1027
Info: RETR mail-editor.png

150 Opening ASCII mode data connection.
Info: QUIT

226 Transfer complete.
```

Figura 2 - registo capturado após o download de um ficheiro.

Parte 2 - Configuração da rede e análise

Experiência 1 - Configurar um IP de rede

Esta experiência teve como objetivo a compreensão do funcionamento de uma simples rede de computadores ligados entre si bem como o tipo de dados enviado entre ambos.

Primeiro foi necessário ligar os computadores um ao outro via switch, após este procedimento em ambos os computadores foram configurados os respetivos IPs das portas eth0 e routes necessários para se comunicarem entre eles. Após verificar que existia ligação entre os dois computadores e que as forwarding e ARP tables estavam corretas, foi efectuado um ping entre os dois computadores e capturado os packets transferidos/recebidos via wireshark pela porta eth0. (ver Anexo II - Exp.1 Commands)

O que são pacotes ARP e qual é a sua utilidade?

Ao longo da experiência e durante a análise dos registos guardados, verificamos o envio de pacotes ARP, utilizados para mapear um endereço de rede para um endereço físico - neste caso o endereço MAC - e associá-lo a um endereço IP.

```
     7 10.323928    G-ProCom_8b:e4:a7   Broadcast           ARP      42 Who has 172.16.50.254? Tell 172.16.50.1
     8 10.324050    HewlettP_c3:78:70   G-ProCom_8b:e4:a7   ARP      60 172.16.50.254 is at 00:21:5a:c3:78:70
     9 10.324060    172.16.50.1         172.16.50.254       ICMP     98 Echo (ping) request  id=0x662d, seq=1/256, ttl=64 (reply in 10)
    10 10.324191    172.16.50.254       172.16.50.1         ICMP     98 Echo (ping) reply    id=0x662d, seq=1/256, ttl=64 (request in 9)
```

Figura 3 - registo capturado através do *Wireshark* durante a experiência.

Quais são os endereços MAC e IP de um pacote ARP?

Desta forma, existem dois pacotes ARP: o primeiro é necessário para associar o IP que se pretende alcançar a um determinado endereço MAC, daí que este pacote é enviado a toda a rede do computador de origem. E o segundo é utilizado quando o computador que possui o IP em questão, envia um pacote ARP contendo o seu endereço MAC.

Que pacotes são gerados pelo comando *ping*?

O comando *ping*, após obter o endereço MAC através dos pacotes ARP, gera pacotes de protocolo ICMP que podem ser do tipo *reply* ou *request*.

Figura 4 - exemplo de resultado de um comando *ping*.

Quais são os endereços MAC e IP dos pacotes *ping*?

Os pacotes do tipo *request* têm como endereço IP e MAC de origem a máquina 1, e de destino os valores de IP e MAC da máquina 4. Os pacotes de tipo *reply* têm os valores contrários.

Como se deteta se uma trama Ethernet recebida é ARP, IP ou ICMP?

No cabeçalho de uma trama Ethernet, o campo EtherType indica o protocolo que o conteúdo da trama encapsula. Os valores do mesmo para ARP e IP são respectivamente 0x0806, 0x0800. Uma trama ICMP é identificada analisando de forma semelhante o cabeçalho de uma trama IP.

Como se obtém o tamanho de uma trama recebida?

O campo EtherType mencionado acima pode ter uma interpretação alternativa. Valores iguais ou inferiores a 1500 indicam que este campo do cabeçalho representa o comprimento do campo de dados da trama (entre 46 e 1500 octetos).

O que é a interface loopback e qual é a sua importância?

A interface loopback é uma interface de rede virtual que o computador utiliza para comunicar consigo próprio, com o objetivo de realizar testes de diagnóstico, ou aceder a servidores na própria máquina, como se de um cliente se tratasse. Assim, uma interface loopback permite a existência de um endereço IP no router, que está sempre ativo, em vez de ser dependente de uma interface física.

Experiência 2 - Implementar duas LANs virtuais num switch

Nesta experiência o principal foco era criar e configurar duas LANs virtuais no switch: a primeira constituída pelas máquinas 1 e 4, e a segunda apenas pela máquina 2. Com esta configuração, a máquina 2 deixaria de ter acesso às máquinas 1 e 4, uma vez que se encontrariam em sub-redes diferentes.

Inicialmente configurou-se a máquina 2 de forma semelhante às outra máquinas na experiência anterior. De seguida, utilizando o terminal no switch da Cisco, foram criada duas vlans - *vlan50* e *vlan51* - às quais foram adicionadas as portas dos computadores correspondentes - *tux51* e *tux54*, e *tux52*, respetivamente. Após todas as configurações estarem efectuadas, foi realizado um *ping* da máquina 1 para as máquinas 4 e 2, e guardados os registos dos mesmos. Por fim, guardou-se os dados obtidos nas 3 máquinas após a realização do *ping* para o broadcast da máquina 1 e de um broadcast a partir da máquina 2. (ver Anexo II - Exp.2 Commands)

Como se configura a vlany0?

A configuração de cada *vlan5x* , sendo x 0 ou 1 , é realizada no switch da Cisco. Em primeiro lugar é necessário criar a vlan usando-se os comandos 'configure terminal', 'vlan5x ' e 'end'. Posteriormente, de modo a configurar a rede de cada vlan, são adicionadas as portas correspondentes, recorrendo para isso a uma série de comandos: 'configure terminal', 'interface fastethernet 0/i' (onde i é o número da porta), 'switchport mode access', 'switchport access vlan 5x (onde x é 0 ou 1, dependendo da vlan a que se quer adicionar a porta i). Conforme as configurações exigidas no guião, o grupo adicionou a porta 1 e 4 à *vlan50* e a porta 2 à *vlan51*.

Quantos são os domínios de broadcast? Como se define esse valor a partir de registos?

Um domínio de broadcast é uma divisão lógica da rede em que qualquer node consegue alcançar qualquer outro node pertencente a esse broadcast (i.e ping). Como tal , a criação e respetiva configuração de uma vlan essencialmente cria o seu próprio domínio de broadcast. Então, devido à arquitetura da rede podemos concluir que existem dois domínios broadcast, um para a *vlan50* e outra *vlan51*.

A análise dos registos obtidos ao longo da experiência permite confirmar a correta implementação da arquitetura de rede especificada no guião uma vez que o ping da máquina 4 a partir da máquina 1 funciona devido a pertencerem ao mesmo domínio. Por outro lado a máquina 1 não consegue alcançar a máquina 2 por não estarem na mesma sub-rede.

Ainda na análise do outro registo em que foi realizado um ping pelo broadcast, ao realizar broadcast a partir da máquina 2, como esta se encontra sozinha na mesma sub-rede apenas consegue dar ping a si próprio não alcançando a máquina 1 nem a 4. (Ver anexo III - Exp2-Part7)

Experiência 3 - Configurar um Router em Linux

O objetivo da experiência 3 era configurar a máquina 4 como router entre as duas sub-redes criadas na experiência 2.

Para realizar esta tarefa, foi necessário configurar a porta *eth1* da máquina 4, assim como permitir IP Forwarding e desativar o ICMP echo-ignore-broadcast. Após esta configuração inicial, as máquinas 1 e 2 foram reconfiguradas para ser possível estabelecer ligação entre as mesmas. Para isso, foram adicionadas rotas em ambas as máquinas com a outra sub-rede como IP de destino, através da

máquina 4. Verificadas as rotas, a partir da máquina 1 foram capturados os pacotes enviados pelo comando *ping* para à porta eth0 e eth1 da máquina 4 assim como a máquina 2.

Verificada a conexão entre todas as máquinas, foi realizado um procedimento para limpar as ARP Tables de todas as máquinas e pingar da máquina 1 para a máquina 2. Desta forma, verificou-se que o router consegue detectar qual a rota a utilizar para efetuar o *ping*. (ver Anexo II - Exp.3 Commands)

Que rotas existem nos tuxes? O que significam?

Na máquina 1, através do comando "route add -net 172.16.51.0/24 gw 172.16.50.254" foi adicionada uma rota que possibilita a comunicação com as máquinas da sub-rede 172.16.51.0/24, utilizando como *gateway* o endereço 172.16.50.254, ou seja, os pacotes enviados para essa sub-rede são encaminhados para a máquina com o IP identificado na *gateway*, a máquina 4.

Da mesma forma, na máquina 2, através do comando "route add -net 172.16.50.0/24 gw 172.16.51.253" foi adicionada a rota que permite o envio de pacotes a partir da máquina para a sub-rede com endereço 172.16.50.0/24 passando pelo *gateway* 172.16.51.253, novamente a máquina 4.

Assim a máquina 1 pode comunicar com a máquina 2, sendo que as interfaces existentes na máquina 4 servem de intermediador da ligação, ou seja, funcionam como *router*.

```
tux51:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.50.254   0.0.0.0         UG    0      0        0 eth0
172.16.50.0     172.16.50.254   255.255.255.0   UG    0      0        0 eth0
172.16.50.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
172.16.51.0     172.16.50.254   255.255.255.0   UG    0      0        0 eth0
tux51:~#
```

Figura 5 - rotas no *tux51*

```
tux52:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.51.253   0.0.0.0         UG    0      0        0 eth0
172.16.50.0     172.16.51.253   255.255.255.0   UG    0      0        0 eth0
172.16.51.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
tux52:~#
```

Figura 6 - rotas no *tux52*

```
tux54:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.50.0     172.16.50.1     255.255.255.0   UG    0      0        0 eth0
172.16.50.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
172.16.51.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
tux54:~#
```

Figura 7 - rotas no *tux54*

Que informação consta numa tabela de *forwarding*?

As tabelas de *forwarding* são obtidas através do comando "route -n" e listam as rotas através das quais os pacotes devem ser encaminhados para chegarem ao destino. São constituídas pelo IP de destino do pacote, o IP para onde dever encaminhado para atingir esse destino (*gateway*), a máscara e a interface utilizada no reencaminhamento do pacote.

Que mensagens ARP e respetivos endereços MAC são observados e porquê?

Ao pingar a máquina 2 a partir da 1 é possível observar a sequência de reencaminhamento existente. Um pedido efetuado para a máquina com IP 172.16.51.1 é reencaminhado para o *router*, ou seja, a máquina 4, através da interface *eth0*, sendo que a máquina obtém o endereço MAC da interface

em questão (172.16.50.254). Os dados são então enviados para a máquina 2, já que está conectada à máquina 4 através da interface *eth1*. Neste último caso, a máquina 4 obtém o endereço MAC da máquina 2. Em ambos os casos em que é necessário obter os endereços MAC são utilizados pacotes ARP.

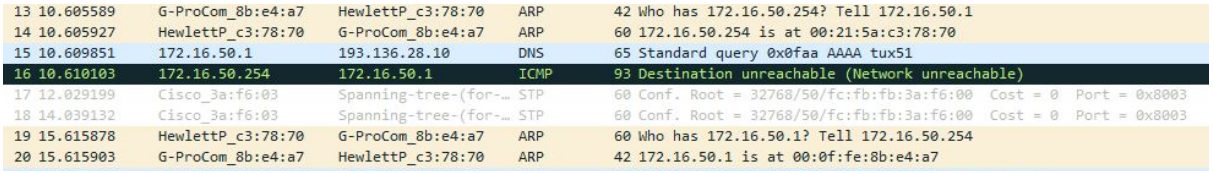


Figura 8 - Mensagem ARP e endereços MAC das máquinas 4 e 2

Que pacotes ICMP são observados e porquê?

Os pacotes ICMP observados no *Wireshark* nas *requests* correspondem aos valores IP e MAC da máquina 1 e 2, sendo respectivamente a origem e o destino, enquanto nas *replies* verifica-se a ordem contrária.

Quais são os endereços IP e MAC associados a pacotes ICMP e porquê?

Sendo possível alcançar a máquina de destino e estando o endereço MAC mapeado, são atribuídos como IPs dos pacotes ICMP o valor das respetivas máquinas de origem e destino, sendo que a rota efectuada e os respectivos encaminhamentos são da responsabilidade do *router*. (Ver Anexos - Exp3-5 para exemplos observados)

Experiência 4 - Configurar um Router Comercial e implementar a NAT

A experiência 4 consiste em configurar um *router* comercial e implementar NAT (*Network Address Translation*) no mesmo, com o objetivo de ter acesso à internet em todas as máquinas.

A implementação da NAT possibilita a comunicação entre os computadores da rede interna, com redes externas. Por se tratar de uma rede privada, os IPs da mesma não seriam reconhecidos fora da rede. Para contornar isso utiliza-se uma técnica que permite reescrever os IPs de origem de uma rede interna para que possam aceder a uma rede externa, e vice-versa.

Inicialmente, configurou-se o *router* comercial e ligou-se o mesmo à rede do laboratório. De seguida, verificou-se se as rotas nas máquinas estavam corretas: na máquina 4 o *router* por omissão é a máquina 1; o *router* comercial tem por omissão encaminhar para as máquinas 2 e 4; uma rota para o endereço 172.16.50.0/24 a partir da máquina 2 e do *router* comercial.

Segundo esta configuração, através da máquina 1 é possível dar *ping* às máquinas 2 e 4 e *router* comercial, o que se confirmou com os testes efetuados.

Por último, foi adicionada a funcionalidade NAT ao *router* comercial, seguido de testes para confirmar se um *ping* da máquina 1 ao *router* era possível. (ver Anexo II - Exp.4 Commands)

Como se configura uma rota estática num *router* comercial?

Para configurar uma rota estática num router comercial é necessário correr a seguinte sequência de comandos no terminal do *router*:

```
conf t
ip route 0.0.0.0 0.0.0.0. 172.16.1.254
ip route 172.16.50.0 255.255.255.0 172.16.51.253
end
```

Quais são as rotas seguidas pelos pacotes nos testes efetuados e porquê?

Executando pela primeira vez um *ping* da máquina 2 para a máquina 1, os pacotes foram enviados para o *router*, que por sua vez encaminhou para a rede 172.16.50.0. Realizando o mesmo

processo novamente, o caminho seguido pelos pacotes é diferente, indo da máquina 2 directamente para a rede 172.16.50.0. O último teste fez o reencaminhamento de forma diferente porque o *router*, através do primeiro pacote, registou as informações de redirecionamento relativas a endereços com o destino utilizado.

Como se configura a NAT num router comercial?

Para a configuração do NAT, executaram-se as seguintes sequências de comandos:

```
conf t
interface gigabitethernet 0/0
ip address 172.16.51.254 255.255.255.0
no shutdown *assegura que as configurações mantêm-se após shutdown
ip nat inside
exit
interface gigabitethernet 0/1
ip address 172.16.1.59 255.255.255.0
no shutdown
ip nat outside
exit
```

Criou-se uma lista de permissões de acesso dos pacotes que iriam ser enviados/recebidos, isto para todas as sub-redes. Foi utilizada a máscara 0.0.0.255 para deixar que todas as subredes tivessem acesso à internet.

```
ip natpool ovrld 172.16.1.59 172.16.1.59 prefix 24
ip nat inside source list 1 pool ovrld overload *desta forma é
mantida a gama de endereços
access-list 1 permit 172.16.50.0 0.0.0.255
access-list 1 permit 172.16.51.0 0.0.0.255
```

Por último, adicionaram-se rotas que redirecionam os pacotes com destino o IP 172.16.50.0 para 172.16.51.253.

```
ip route 0.0.0.0 0.0.0.0 172.16.1.254
ip route 172.16.50.0 255.255.255.0 172.16.51.253
end.
```

O que faz uma NAT?

Uma NAT (*Network Address Translation*) permite que um dispositivo - neste caso *router* Cisco - aja como um intermediador entre a Internet e uma rede local - rede do laboratório. Na prática, todos os computadores pertencentes à rede local são representados através de um endereço IP único. A NAT oferece então segurança adicional na medida em que toda a rede de computadores é representada por um único endereço.

Experiência 5 - Configurar um servidor DNS

Sendo o objetivo geral das experiências conseguir aceder a redes externas mantendo a configuração de rede interna desenvolvida até ao momento, conseguindo desta forma aceder à Internet, o passo lógico seguinte foi configurar o DNS.

Como se configura o serviço de DNS num *host*?

Para configurar o DNS, em todos os *hosts* da rede, edita-se o ficheiro */etc/resolv.conf*. Este ficheiro é lido cada vez que são invocadas rotinas que requerem acesso à Internet. Neste caso, o ficheiro foi editado colocando "domain netlab.fe.up.pt nameserver 172.16.1.1", que se trata do endereço IP do servidor que deve ser acedido.

Figura 9 - Ficheiro resolv.conf com a configuração necessária para a experiência

Que pacotes são trocados por DNS e qual é a informação transportada?

Para testar esta experiência, foi executado um *ping* a http://google.pt. Nos *logs*, consequentemente, verificou-se que o DNS pergunta qual a informação contida num dado *domain name* e este responde com o tempo de vida e o tamanho do pacote de dados.

As queries possuem parâmetros *name*, *type* e *class*, enquanto as respostas, para além dos campos que existem nas *queries*, também possuem *time to live*, *data length* e *address*.

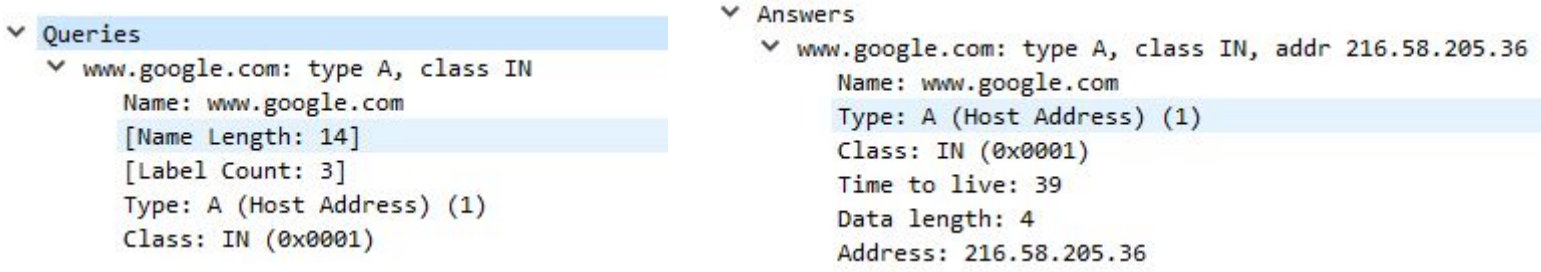


Figura 10 - Query e resposta do comando ping google.com

Experiência 6 - Estabelecer uma ligação TCP

Durante a experiência 6, compilou-se e executou-se a aplicação desenvolvida e descrita na primeira parte do relatório, com a rede descrita nas experiências anteriores configurada.

Nas execuções da aplicação foram utilizados diversos servidores FTP, de forma a testar os diferentes modos suportados pelo protocolo FTP e efetuados os respetivos *downloads* de ficheiros. Todos os *downloads* foram executados com sucesso, o que demonstrou que a rede estava corretamente configurada, não trazendo qualquer problema no acesso por protocolo FTP, assim como à utilização de um servidor exterior à rede.

Quantas conexões TCP são abertas pela aplicação FTP?

A aplicação FTP realiza 2 conexões TCP, a conexão de controlo e a conexão de dados.

Em qual das conexões é transportada a informação de controlo FTP?

A informação de controlo FTP, tal como a identificação do utilizador, *password* ou comandos para mudar diretórios remotos é transportada pela conexão de controlo.

Quais são as fases de uma conexão TCP?

Uma conexão TCP é caracterizada por três fases distintas. Em primeiro lugar, para estabelecer a ligação, o cliente envia a mensagem SYN com informação relativa ao número de sequência, e recebe a resposta do servidor. Esta resposta inclui SYN e ACK (*acknowledgment*) da mensagem anteriormente enviada. Para finalizar esta primeira etapa, o cliente envia uma mensagem de ACK.

Com a ligação estabelecida, ocorre o processo de transferência dos dados pretendidos pelo cliente. Terminando a transferência, para encerrar a ligação, é repetido um processo semelhante ao descrito para iniciar a mesma.

Como funciona o mecanismo ARC TCP? Quais são os campos TCP relevantes? Que informação relevante pode ser observadas nos *logs*?

O ARQ do TCP recorre ao mecanismo de *Sliding Window*, uma variante do Go-Back-N. Este permite ao receptor receber *N bytes* sem ter de esperar pela confirmação do ACK. O receptor envia um ACK que inclui o número de sequência esperado do próximo *byte*, ou seja, permite ao recetor receber *N bytes*, começando pelo *byte* com o número de sequência esperado.

Tanto o receptor como o servidor possuem uma "janela" com um tamanho menor ou igual ao tamanho do *buffer*. O servidor envia todos os *bytes* permitidos pelo tamanho da janela e fica à espera de um ACK. Õ receptor desloca-se na janela para o número de sequência correspondente, indicando os *bytes* que podem ser enviados pelo servidor que o *buffer* da nova posição da janela permite. O servidor recebe o ACK com essa informação, voltando a enviar os *bytes*.

Como se pode concluir pela estrutura da *Sliding Window*, um dos elementos mais importantes neste mecanismo é o número de sequência.

Como funciona o mecanismo de controlo de congestão TCP? Quais são os campos relevantes? Como evolui o fluxo dos dados? Funciona de acordo com o mecanismo de controlo de congestão TCP?

O mecanismo de controlo de congestão do TCP comporta-se em modo "serra". O número de *bytes* transmitidos por segmento aumenta com o tempo, até ser detectado que um segmento foi perdido, fazendo com que o número de *bytes* transmitidos diminua e se entre no modo *congestion avoidance*. Através dos *logs* pode-se verificar o número de *bytes* enviados por segmento.

```
36 3.115570   172.16.50.1     90.130.70.73    TCP       66 59377 → 21 [ACK] Seq=58 Ack=235 Win=29312 Len=0 TSval=4660198 TSecr=1462130689
37 3.115765   90.130.70.73    172.16.50.1     FTP-DA… 1514 FTP Data: 1448 bytes
38 3.115788   172.16.50.1     90.130.70.73    TCP       66 49673 → 24952 [ACK] Seq=1 Ack=1449 Win=32128 Len=0 TSval=4660198 TSecr=1462130689
39 3.116016   90.130.70.73    172.16.50.1     FTP-DA… 2962 FTP Data: 2896 bytes
40 3.116162   172.16.50.1     90.130.70.73    TCP       66 49673 → 24952 [ACK] Seq=1 Ack=4345 Win=37888 Len=0 TSval=4660199 TSecr=1462130689
41 3.116266   90.130.70.73    172.16.50.1     FTP-DA… 4410 FTP Data: 4344 bytes
```

Figura 11 - Excerto do log capturado demonstrando o crescente tamanho de dados transferido por segmento

Os resultados estão de acordo com o mecanismo, uma vez que começam com um valor baixo, sobem e voltam a diminuir - quando é detectado que um segmento foi perdido - criando o efeito "serra".

O fluxo de dados TCP é afetado pela existência de uma segunda conexão TCP? Como?

Analisando os *logs* recolhidos, pode-se concluir - utilizando o I/O Graph - que a conexão de dados foi afectada por sido iniciada a transferência de um ficheiro noutra máquina, ou seja, pelo aparecimento de uma segunda conexão de dados.

Inicialmente começou-se um *download* na máquina 1, que atingiu uma velocidade de aproximadamente 3800 pacotes por segundo. No entanto, devido ao aparecimento de uma nova conexão de dados - na máquina 2 -, a velocidade diminuiu para menos de metade (800 pacotes por segundo), sendo esta também a velocidade a que a segunda transferência começou. Quando o *download* na máquina 2 terminou, a velocidade na máquina 1 subiu para 7500 pacotes por segundo, até a transferência ter terminado.

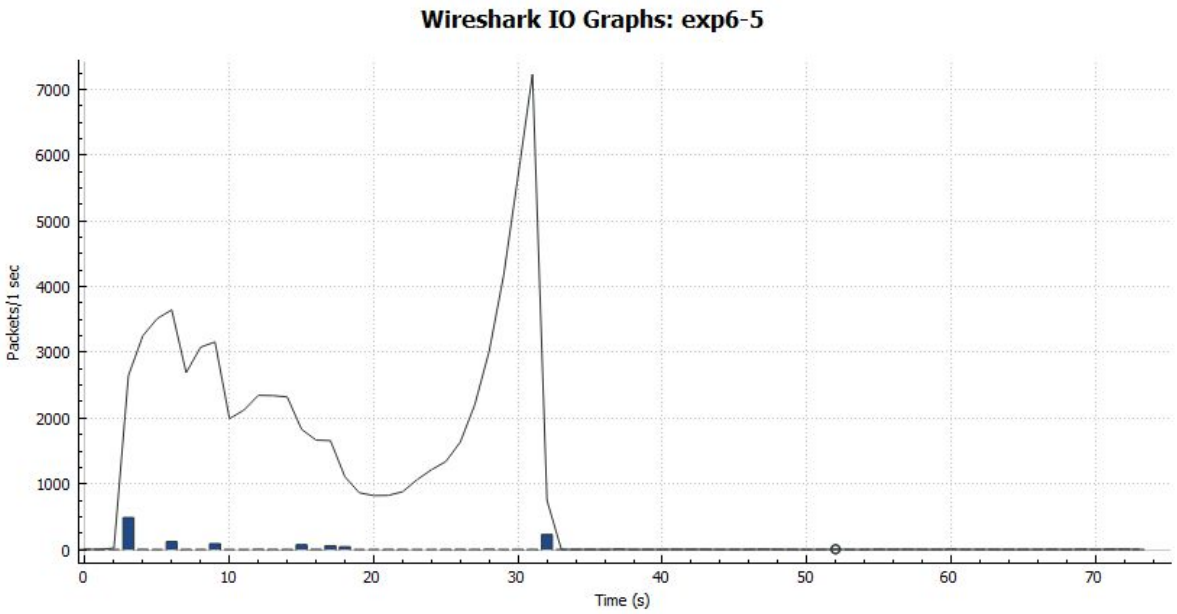


Figura 12 - Evolução do número de pacotes transmitidos por segundo na máquina 1

Em suma, conclui-se que o fluxo de dados FTP é afetado pelo aparecimento de uma segunda conexão de dados.

Conclusão

Com este trabalho foi possível compreender com maior pormenor como funciona uma rede de computadores e como são feitas as comunicações na mesma, através da realização da sua configuração. Através do desenvolvimento da aplicação de download houve também a oportunidade de passar a conhecer em maior detalhe um dos protocolos mais importantes e mais usados na transferência de ficheiros entre um cliente e um servidor, o FTP. No final, todas as experiências planeadas foram executadas com sucesso e o protocolo foi de igual forma implementado, pelo que o grupo considera que o resultado final é positivo.

Anexo I - Código Fonte da Aplicação de Download

parser.h

```
#pragma once

#include <string.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <regex.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>

#define MAX_SIZE 256

typedef struct {
    char user[MAX_SIZE];
    char password[MAX_SIZE];
    char host[MAX_SIZE];
    char ip[MAX_SIZE];
    char path[MAX_SIZE];
    char filename[MAX_SIZE];
    int port;
} URLInfo;

int parseURL(URLInfo* url, const char* str);
int getIpByHost(URLInfo* url);
int getSubstring(const char* str, int start, char delimiter, int* substringLength);
```

parser.c

```
#include "parser.h"

#define BASE_INDEX_AFTER_PROTOCOL_IDENTIFIER 6

int parseURL(URLInfo* url, const char* urlStr) {
    // Building regular expression to check if given URL is valid
    const char* regularExpression =
"ftp://([a-z0-9]*:[a-z0-9]*@)?[a-z0-9.~-]+/[a-z0-9/~._-]+";
    regex_t* regex = (regex_t*) malloc(sizeof(regex_t));
    size_t nmatch = strlen(urlStr);
    regmatch_t pmatch[nmatch];

    if (regcomp(regex, regularExpression, REG_EXTENDED | REG_ICASE) != 0) {
        perror("Bad regex");
        return -1;
```

```
    }

    if (regexec(regex, urlStr, nmatch, pmatch, REG_EXTENDED | REG_ICASE) != 0) {
        perror("Invalid URL format");
        return -1;
    }

    free(regex);

    int credentialsGiven = 0;
    if (memchr(urlStr, '@', strlen(urlStr))) //If '@' is present, it is because username
and password were provided
        credentialsGiven = 1;

    int cumulativeLength = BASE_INDEX_AFTER_PROTOCOL_IDENTIFIER;
    int substringLength = 0;

    if (credentialsGiven) {
        //Retrieve username
        getSubstring(urlStr, cumulativeLength, ':', &substringLength);
        memcpy(url->user, urlStr+cumulativeLength, substringLength);

        //printf("Got username, is %s\n", url->user);

        cumulativeLength += substringLength + 1;

        //Retrieve password
        getSubstring(urlStr, cumulativeLength, '@', &substringLength);
        memcpy(url->password, urlStr+cumulativeLength, substringLength);

        //printf("Got password, is %s\n", url->password);

        cumulativeLength += substringLength + 1;
    }

    //Retrieve host
    getSubstring(urlStr, cumulativeLength, '/', &substringLength);
    memcpy(url->host, urlStr+cumulativeLength, substringLength);

    cumulativeLength += substringLength + 1;

    //printf("Got host, is %s\n", url->host);

    char* lastSlash = strrchr(urlStr, '/');
    int pathSize = lastSlash - (urlStr + cumulativeLength);

    if(pathSize > 0){
        memcpy(url->path, urlStr+cumulativeLength, pathSize);
        cumulativeLength += pathSize + 1;
    }

    //printf("Got path, is %s\n", url->path);

    //Retrieve filename
```

```
    int filenameSize = strlen(urlStr) - cumulativeLength;
    memcpy(url->filename, urlStr+cumulativeLength, filenameSize);

    //printf("Got filename, is %s\n", url->filename);

    return 0;
}

int getIpByHost(URLInfo* url) {
    struct hostent* h;

    if ((h = gethostbyname(url->host)) == NULL) {
        herror("gethostbyname");
        return -1;
    }

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

    char* ip = inet_ntoa(*((struct in_addr *) h->h_addr));
    strcpy(url->ip, ip);

    return 0;
}

int getSubstring(const char* str, int start, char delimiter, int* substringLength) {
    int length = strlen(str+start);
    char* end = memchr(str+start, delimiter, length);
    if(end == NULL)
        return -1;

    *substringLength = end - (str + start);
    return 0;
}
```

client.h

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

#define R1XX                1
                                        //FTP Reply code hundreds digit
#define R2XX                2
                                        //FTP Reply code hundreds digit
```

```
#define R3XX                3
                                        //FTP Reply code hundreds digit
#define R4XX                4
                                        //FTP Reply code hundreds digit
#define R5XX                5
                                        //FTP Reply code hundreds digit

#define WAIT_REPLY 10
#define CONTINUE   11
#define REPEAT        12
#define ABORT         13

typedef struct {
    int controlSocketFd; // file descriptor to control socket
    int dataSocketFd; // file descriptor to data socket
} FTPInfo;

int connectSocket(const char *ip, int port);
int connectServer(FTPInfo* client, const char* ip, int port);
int login(FTPInfo* client, const char* username, const char* password);
int download(FTPInfo* client, const char* fileName);
int disconnectServer(FTPInfo* client);

int readMessage(FTPInfo* client, char* message, size_t messageSize);
int sendMessage(FTPInfo* client, const char* message, size_t messageSize);
int handleFTPReplyCode(char* reply_code);
int parseFTPReplyCode(char* reply);
int sendCommand(FTPInfo* client, char* command);

int setCWD(FTPInfo* client, const char* path);
int setPassiveMode(FTPInfo* client);
int requestRETR(FTPInfo* client, const char* fileName);
```

client.c

```
#include "client.h"

int connectSocket(const char* ip, int port) {
    int sockfd;
    struct sockaddr_in server_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(ip); /*32 bit Internet address network byte
ordered*/
    server_addr.sin_port = htons(port); /*server TCP port must be network byte ordered */

    // open an TCP socket
    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket()");
        return -1;
    }
```

```
    // connect to the server
    if (connect(sockfd, (struct sockaddr *) &server_addr, sizeof(server_addr))< 0) {
        perror("connect()");
        return -1;
    }

    return sockfd;
}

/*
 * Connect socket (control socket) to ftp server
 * Receive a reply from the ftp server (code : 220).
 */
int connectServer(FTPInfo* client, const char* ip, int port) {
    char response[1024];

    if ((client->controlSocketFd = connectSocket(ip, port)) < 0) {
        printf("ERROR: Cannot connect socket.\n");
        return -1;
    }

    client->dataSocketFd = 0;

    if (readMessage(client, response, sizeof(response)) == -1) {
        printf("ERROR: readMessage failed.\n");
        return -1;
    }

    int reply_state = handleFTPReplyCode(response);

    if(reply_state != CONTINUE) {
        printf("ERROR: Server is not ready.\n");
        return -1;
    }

    return 0;
}

/*
 * Send login to the ftp server using the command USER and wait for confirmation (331)
 * Send password using the command PASS and wait for confirmation (230).
 */
int login(FTPInfo* client, const char* username, const char* password) {
    char message[1024];

    // Sends username
    sprintf(message, "USER %s\r\n", username); //'\r' is to simulate 'ENTER'

    if(sendCommand(client, message) != 0) {
        printf("ERROR: USER command received 5XX code.\n");
        return -1;
    }
```

```
    // cleaning buffer
    memset(message, 0, sizeof(message));

    // Sends password
    sprintf(message, "PASS %s\r\n", password);

    if(sendCommand(client, message) != 0) {
        printf("ERROR: PASS command received 5XX code.\n");
        return -1;
    }

    return 0;
}


int download(FTPInfo* client, const char* fileName) {

    if(requestRETR(client, fileName) == -1) {
        printf("ERROR: sendRETR failed.\n");
        return -1;
    }

    FILE* file = fopen(fileName, "wb");

    if (file == NULL) {
        printf("ERROR: Could not create the file.\n");
        return -1;
    }

    char fileBuf[1024];
    int nBytesRead = 0;

    while ((nBytesRead = read(client->dataSocketFd, fileBuf, sizeof(fileBuf)))) {
        if (nBytesRead < 0) {
                printf("ERROR: Nothing was received from data socket fd.\n");
                return -1;
        }

        if ((nBytesRead = fwrite(fileBuf, nBytesRead, 1, file)) < 0) {
                printf("ERROR: Cannot write data in file.\n");
                return -1;
        }
    }

    fclose(file);
    close(client->dataSocketFd);

    return 0;
}

/*
 * Send the QUIT command and wait for reply (221)
 */
int disconnectServer(FTPInfo* client) {
```

```
    char message[1024];

    sprintf(message, "QUIT\r\n");

    if(sendCommand(client, message) != 0) {
        printf("ERROR: QUIT command received 5XX code.\n");
        return -1;
    }

    close(client->controlSocketFd);

    return 0;
}

int readMessage(FTPInfo* client, char* message, size_t messageSize) {
    FILE* fd = fdopen(client->controlSocketFd, "r");

    do {
        memset(message, 0, messageSize);
        message = fgets(message, messageSize, fd);
        if(message == NULL){
                printf("ERROR: Cannot read the message.\n");
                return -1;
        }
        printf("%s", message);
    } while (!('1' <= message[0] && message[0] <= '5') || message[3] != ' ');

    return 0;
}

int sendMessage(FTPInfo* client, const char* message, size_t messageSize) {
    int nBytesRead = 0;

    if ((nBytesRead = write(client->controlSocketFd, message, messageSize)) <= 0) {
        printf("ERROR: Message wasn't send.\n");
        return -1;
    }

    //printf("Bytes send: %d\n", nBytesRead);
    printf("Info: %s\n", message);

    return 0;
}

int setCWD(FTPInfo* client, const char* path) {
    char message[1024];

    if(strlen(path) <= 0)
        return 0;

    sprintf(message, "CWD %s\r\n", path);

    if(sendCommand(client, message) != 0) {
        printf("download: CWD command received 5XX code.\n");
```

```
        return -1;
    }

    return 0;
}

/*
 * Send command PASV, and waits for a reply that gives an IP address and a port
 * Parse this message,
 * Connect a second socket (a data socket) with the given configuration.
 */
int setPassiveMode(FTPInfo* client) {
    char message[1024] = "PASV\r\n";

    if(sendCommand(client, message) != 0) {
        printf("ERROR: PASV command received 5XX code.\n");
        return -1;
    }

    // Parses the message received
    int ipPart1, ipPart2, ipPart3, ipPart4, portPart1, portPart2;

    if ((sscanf(message, "227 Entering Passive Mode (%d,%d,%d,%d,%d,%d)",
                &ipPart1, &ipPart2, &ipPart3, &ipPart4, &portPart1, &portPart2)) < 0) {
        printf("ERROR: Cannot process information to calculating port.\n");
        return -1;
    }

    char ip[1024];

    // Builds IP
    if (sprintf(ip, "%d.%d.%d.%d", ipPart1, ipPart2, ipPart3, ipPart4) < 0) {
        printf("ERROR: Cannot form ip address.\n");
        return -1;
    }

    // Builds Port
    int port = portPart1 * 256 + portPart2;

    printf("IP: %s\nPORT: %d\n", ip, port);

    if ((client->dataSocketFd = connectSocket(ip, port)) < 0) {
        printf("ERROR: Cannot connect to socket on PASV.\n");
        return -1;
    }

    return 0;
}

/*
 * Send the RETR command with the file path
 */
int requestRETR(FTPInfo* client, const char* fileName) {
    char message[1024];
```

```
    char response[1024];

    sprintf(message, "RETR %s\r\n", fileName);

    if (sendMessage(client, message, strlen(message)) == -1) {
        printf("ERROR: sendMessage failed.\n");
        return -1;
    }

    if (readMessage(client, response, sizeof(response)) == -1) {
        printf("ERROR: readMessage failed.\n");
        return -1;
    }

    return 0;
}

int sendCommand(FTPInfo* client, char* command) {

    int reply_state;
    char response[1024];

    /* Repeat until reply code signals completion */
    do {
        /* Write command and read response */
        if (sendMessage(client, command, strlen(command)) == -1) {
                printf("ERROR: sendMessage failed.\n");
                return -1;
        }

        if (readMessage(client, response, sizeof(response)) == -1) {
                printf("ERROR: readMessage failed.\n");
                return -1;
        }

        reply_state = handleFTPReplyCode(response);

        /* Handle reply code */
        if(reply_state == ABORT) return -1;
        else if(reply_state == WAIT_REPLY) {
                if (readMessage(client, response, sizeof(response)) == -1) {
                        printf("ERROR: readMessage failed.\n");
                        return -1;
                }
                reply_state = handleFTPReplyCode(response);
                if(reply_state == ABORT) return -1;
        }

    } while(reply_state != CONTINUE);

    memcpy(command, response, sizeof(response));
    return 0;
}
```

```
int handleFTPReplyCode(char* reply_code) {

    int result = parseFTPReplyCode(reply_code);
    int hundreds = (result / 100) % 100;

    switch(hundreds) {
    case R1XX:
        return WAIT_REPLY;
        break;
    case R2XX:
        return CONTINUE;
        break;
    case R3XX:
        return CONTINUE;
        break;
    case R4XX:
        return REPEAT;
        break;
    case R5XX:
        return ABORT;
        break;
    default:
        return ABORT;
        break;
    }
}

int parseFTPReplyCode(char* reply) {

    /* Copy first 3 values of reply message which should be the reply code */
    char code[4];
    strncpy(code, reply, 3);
    code[3] = '\0';

    unsigned long number = strtoul(code, NULL, 10);

    return number;
}
```

main.c

```
#include <stdio.h>
#include "parser.h"
#include "client.h"

static void printUsage(char* argv0);

int main(int argc, char** argv) {
    if (argc != 2) {
        printf("WARNING: Wrong number of arguments.\n");
        printUsage(argv[0]);
        return -1;
    }
```

```
////////////// URL PROCESS //////////////
URLInfo* url = (URLInfo*) malloc(sizeof(URLInfo));
url->port = 21;

// start parsing argv[1] to URL components
if (parseURL(url, argv[1]))
    return -1;

// edit url ip by hostname
if (getIpByHost(url)) {
    printf("ERROR: Cannot find ip to hostname %s.\n", url->host);
    return -1;
}

printf("\nThe IP received to %s was %s\n", url->host, url->ip);

////////////// FTP CLIENT PROCESS //////////////

FTPInfo* ftp = (FTPInfo*) malloc(sizeof(FTPInfo));;
if(connectServer(ftp, url->ip, url->port) < 0){
    printf("ERROR: Failure connecting to FTP server control port.\n");
    return -1;
}

// Verifying username
const char* user = strlen(url->user) ? url->user : "anonymous";

// Verifying password
char* password;

if (strlen(url->password)) {
    password = url->password;
} else {
    printf("You are now entering in anonymous mode.\n");
    password = "random";
}

// Sending credentials to server
if (login(ftp, user, password)) {
    printf("ERROR: Cannot login user %s\n", user);
    return -1;
}

// Changing directory
if (setCWD(ftp, url->path)) {
    printf("ERROR: Cannot change directory to the folder of %s\n",url->filename);
    return -1;
}

// Entry in passive mode
if (setPassiveMode(ftp)) {
    printf("ERROR: Cannot entry in passive mode\n");
    return -1;
```

```
    }

    // Starting file transfer
    if(download(ftp, url->filename) < 0){
        printf("ERROR: Cannot download the file\n");
        return -1;
    }

    // Disconnecting from server
    if(disconnectServer(ftp) < 0){
        printf("ERROR: Cannot disconnect from server\n");
        return -1;
    }

    return 0;
}

void printUsage(char* argv0) {
    printf("\nUsage1 Normal: %s ftp://[<user>:<password>@]<host>/<url-path>\n",argv0);
    printf("Usage2 Anonymous: %s ftp://<host>/<url-path>\n\n", argv0);
}
```

Anexo II - Comandos de Configuração

Experiência 1

Tux1

/etc/init.d/networking restart
ifconfig eth0 up
ifconfig
ifconfig eth0 172.16.50.1/24
route add -net 172.16.50.0/24 gw 172.16.50.254
route -n
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

Tux4

/etc/init.d/networking restart
ifconfig eth0 up
ifconfig
ifconfig eth0 172.16.50.254/24
route add -net 172.16.50.0/24 gw 172.16.50.1
route -n
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

Experiência 2

Tux1

/etc/init.d/networking restart
ifconfig eth0 up
ifconfig
ifconfig eth0 172.16.50.1/24
route add -net 172.16.50.0/24 gw 172.16.50.254
route -n
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

Tux2

/etc/init.d/networking restart
ifconfig eth0 up
ifconfig
ifconfig eth0 172.16.51.1/24
route add default gw 172.16.51.254
route -n
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

Tux4

/etc/init.d/networking restart
ifconfig eth0 up
ifconfig
ifconfig eth0 172.16.50.254/24
route add -net 172.16.50.0/24 gw 172.16.50.1
route -n
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

VLANS

// Create vlan50 e vlan51 (where y is equal to 0 or 1)
//# Means the number of port in switch correspondig tuxy1, tux4(eth0) para vlan50
//# Means the number of port in switch correspondig tuxy2, tux4(eth1) para vlan51

configure terminal
vlan 5y
end
show vlan id 5y

//Add Corresponding ports
//configure port1
configure terminal
interface fastethernet 0/#
switchport mode access
switchport access vlan5y
end

Experiência 3

Tux1

```
/etc/init.d/networking restart
ifconfig eth0 up
ifconfig
ifconfig eth0 172.16.50.1/24
route add -net 172.16.50.0/24 gw 172.16.50.254
route -n
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 >
/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

Tux2

```
/etc/init.d/networking restart
ifconfig eth0 up
ifconfig
ifconfig eth0 172.16.51.1/24
route add default gw 172.16.51.254
route add -net 172.16.51.0/24 gw 172.16.51.253
route -n
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 >
/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

Tux4

```
/etc/init.d/networking restart
ifconfig eth0 up
ifconfig
ifconfig eth0 172.16.50.254/24
ifconfig eth1 up
ifconfig
ifconfig eth1 172.16.51.253/24
route add -net 172.16.50.0/24 gw 172.16.50.1
route add -net 172.16.51.0/24 gw 172.16.51.253
route -n
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

Experiência 4

Tux1

```
#!/bin/bash
/etc/init.d/networking restart
ifconfig eth0 up
ifconfig
ifconfig eth0 172.16.50.1/24
route add default gw 172.16.50.254
route -n
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 >
/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

Tux2

```
/etc/init.d/networking restart
ifconfig eth0 up
ifconfig
ifconfig eth0 172.16.51.1/24
route add -net 172.16.50.0/24 gw 172.16.51.253
route add default gw 172.16.51.254
route -n
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 >
/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
echo 1 >
/proc/sys/net/ipv4/conf/eth0/accept_redirects
echo 1 > /proc/sys/net/ipv4/conf/all/accept_redirects
```

Tux4

```
/etc/init.d/networking restart
ifconfig eth0 up
ifconfig
ifconfig eth0 172.16.50.254/24
ifconfig eth1 up
ifconfig
ifconfig eth1 172.16.51.253/24
route add default gw 172.16.51.254
route -n
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 >
/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

Configurar Router

```
conf t
interface gigabitethernet 0/0
ip address 172.16.51.254 255.255.255.0
no shutdown
ip nat inside
exit
interface gigabitethernet 0/1
ip address 172.16.1.59 255.255.255.0
no shutdown
ip nat outside
exit
ip nat pool ovrld 172.16.1.59 172.16.1.59 prex 24
ip nat inside source list 1 pool ovrld overload
access-list 1 permit 172.16.50.0 0.0.0.255
access-list 1 permit 172.16.51.0 0.0.0.255
ip route 0.0.0.0 0.0.0.0 172.16.1.254
ip route 172.16.50.0 255.255.255.0 172.16.51.253
end
```

Anexo III - Logs Capturados

Exp1 - Part 3

```
tux54:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:21:5a:c3:78:70
          inet addr:172.16.50.254  Bcast:172.16.50.255  Mask:255.255.255.0
          inet6 addr: fe80::221:5aff:fec3:7870/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:63 errors:0 dropped:0 overruns:0 frame:0
          TX packets:132 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6160 (6.0 KiB)  TX bytes:15089 (14.7 KiB)
          Interrupt:17

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:291 errors:0 dropped:0 overruns:0 frame:0
          TX packets:291 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:31803 (31.0 KiB)  TX bytes:31803 (31.0 KiB)

tux54:~#
```

```
tux51:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0f:fe:8b:e4:a7
          inet addr:172.16.50.1  Bcast:172.16.50.255  Mask:255.255.255.0
          inet6 addr: fe80::20f:feff:fe8b:e4a7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6689 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7301 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:676067 (660.2 KiB)  TX bytes:563053 (549.8 KiB)
          Interrupt:19 Memory:f0500000-f0520000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:6266 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6266 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1964951 (1.8 MiB)  TX bytes:1964951 (1.8 MiB)

tux51:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.50.0     172.16.50.254   255.255.255.0   UG    0      0        0 eth0
172.16.50.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
tux51:~#
```

Exp1 - Part 7

7	10.323928	G-ProCom_8b:e4:a7	Broadcast	ARP	42 Who has 172.16.50.254? Tell 172.16.50.1
8	10.324050	HewlettP_c3:78:70	G-ProCom_8b:e4:a7	ARP	60 172.16.50.254 is at 00:21:5a:c3:78:70
9	10.324060	172.16.50.1	172.16.50.254	ICMP	98 Echo (ping) request id=0x662d, seq=1/256, ttl=64 (reply in 10)
10	10.324191	172.16.50.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x662d, seq=1/256, ttl=64 (request in 9)
11	11.322925	172.16.50.1	172.16.50.254	ICMP	98 Echo (ping) request id=0x662d, seq=2/512, ttl=64 (reply in 12)
12	11.323064	172.16.50.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x662d, seq=2/512, ttl=64 (request in 11)
13	12.029121	Cisco_3a:f6:01	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/1/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8001
14	12.321982	172.16.50.1	172.16.50.254	ICMP	98 Echo (ping) request id=0x662d, seq=3/768, ttl=64 (reply in 15)
15	12.322106	172.16.50.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x662d, seq=3/768, ttl=64 (request in 14)
16	13.245497	Cisco_3a:f6:01	CDP/VTP/DTP/PAgP/UD…	DTP	60 Dynamic Trunk Protocol
17	13.245584	Cisco_3a:f6:01	CDP/VTP/DTP/PAgP/UD…	DTP	90 Dynamic Trunk Protocol
18	13.321974	172.16.50.1	172.16.50.254	ICMP	98 Echo (ping) request id=0x662d, seq=4/1024, ttl=64 (reply in 19)
19	13.322095	172.16.50.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x662d, seq=4/1024, ttl=64 (request in 18)
20	14.033977	Cisco_3a:f6:01	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/1/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8001
21	14.321958	172.16.50.1	172.16.50.254	ICMP	98 Echo (ping) request id=0x662d, seq=5/1280, ttl=64 (reply in 22)
22	14.322078	172.16.50.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x662d, seq=5/1280, ttl=64 (request in 21)
23	15.321976	172.16.50.1	172.16.50.254	ICMP	98 Echo (ping) request id=0x662d, seq=6/1536, ttl=64 (reply in 24)
24	15.322127	172.16.50.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x662d, seq=6/1536, ttl=64 (request in 23)
25	15.332846	HewlettP_c3:78:70	G-ProCom_8b:e4:a7	ARP	60 Who has 172.16.50.1? Tell 172.16.50.254
26	15.332857	G-ProCom_8b:e4:a7	HewlettP_c3:78:70	ARP	42 172.16.50.1 is at 00:0f:fe:8b:e4:a7
27	16.038840	Cisco_3a:f6:01	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/1/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8001
28	16.321957	172.16.50.1	172.16.50.254	ICMP	98 Echo (ping) request id=0x662d, seq=7/1792, ttl=64 (reply in 29)
29	16.322077	172.16.50.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x662d, seq=7/1792, ttl=64 (request in 28)
30	17.321968	172.16.50.1	172.16.50.254	ICMP	98 Echo (ping) request id=0x662d, seq=8/2048, ttl=64 (reply in 31)
31	17.322100	172.16.50.254	172.16.50.1	ICMP	98 Echo (ping) reply id=0x662d, seq=8/2048, ttl=64 (request in 30)

Exp2 - Part 1

```
tux52:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:21:5a:61:2f:d6
          inet addr:172.16.51.1  Bcast:172.16.51.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:4 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:17

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:3779 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3779 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:391021 (381.8 KiB)  TX bytes:391021 (381.8 KiB)

tux52:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.51.254   0.0.0.0         UG    0      0        0 eth0
172.16.51.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
tux52:~#
```

Exp2 - Part 4

```
11 15.479864   172.16.50.1     172.16.50.254   ICMP   98 Echo (ping) request  id=0x666c, seq=1/256, ttl=64 (reply in 12)
12 15.480134   172.16.50.254   172.16.50.1     ICMP   98 Echo (ping) reply    id=0x666c, seq=1/256, ttl=64 (request in 11)
13 16.061346   Cisco_3a:f6:03  Spanning-tree-(for-… STP   60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00  Cost = 0  Port = 0x8003
14 16.478865   172.16.50.1     172.16.50.254   ICMP   98 Echo (ping) request  id=0x666c, seq=2/512, ttl=64 (reply in 15)
15 16.479248   172.16.50.254   172.16.50.1     ICMP   98 Echo (ping) reply    id=0x666c, seq=2/512, ttl=64 (request in 14)
16 17.477868   172.16.50.1     172.16.50.254   ICMP   98 Echo (ping) request  id=0x666c, seq=3/768, ttl=64 (reply in 17)
17 17.478103   172.16.50.254   172.16.50.1     ICMP   98 Echo (ping) reply    id=0x666c, seq=3/768, ttl=64 (request in 16)
```

Exp2 - Part7-1

Tux1

```
 7 10.323928   G-ProCom_8b:e4:a7  Broadcast           ARP    42 Who has 172.16.50.254? Tell 172.16.50.1
 8 10.324050   HewlettP_c3:78:70  G-ProCom_8b:e4:a7  ARP    60 172.16.50.254 is at 00:21:5a:c3:78:70
 9 10.324060   172.16.50.1        172.16.50.254      ICMP   98 Echo (ping) request  id=0x662d, seq=1/256, ttl=64 (reply in 10)
10 10.324191   172.16.50.254      172.16.50.1        ICMP   98 Echo (ping) reply    id=0x662d, seq=1/256, ttl=64 (request in 9)
11 11.322925   172.16.50.1        172.16.50.254      ICMP   98 Echo (ping) request  id=0x662d, seq=2/512, ttl=64 (reply in 12)
12 11.323064   172.16.50.254      172.16.50.1        ICMP   98 Echo (ping) reply    id=0x662d, seq=2/512, ttl=64 (request in 11)
```

Tux4

```
22 34.538958   172.16.50.1     172.16.50.255   ICMP   98 Echo (ping) request  id=0x67b2, seq=1/256, ttl=64 (no response found!)
23 34.539007   172.16.50.254   172.16.50.1     ICMP   98 Echo (ping) reply    id=0x67b2, seq=1/256, ttl=64
24 35.537938   172.16.50.1     172.16.50.255   ICMP   98 Echo (ping) request  id=0x67b2, seq=2/512, ttl=64 (no response found!)
25 35.537979   172.16.50.254   172.16.50.1     ICMP   98 Echo (ping) reply    id=0x67b2, seq=2/512, ttl=64
26 36.062064   Cisco_3a:f6:06  Cisco_3a:f6:06  LOOP   60 Reply
27 36.087145   Cisco_3a:f6:06  Spanning-tree-(for-… STP   60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00  Cost = 0  Port = 0x8006
28 36.536925   172.16.50.1     172.16.50.255   ICMP   98 Echo (ping) request  id=0x67b2, seq=3/768, ttl=64 (no response found!)
29 36.536965   172.16.50.254   172.16.50.1     ICMP   98 Echo (ping) reply    id=0x67b2, seq=3/768, ttl=64
30 37.536554   172.16.50.1     172.16.50.255   ICMP   98 Echo (ping) request  id=0x67b2, seq=4/1024, ttl=64 (no response found!)
31 37.536586   172.16.50.254   172.16.50.1     ICMP   98 Echo (ping) reply    id=0x67b2, seq=4/1024, ttl=64
```

Tux2

```
22 32.078203   Cisco_3a:f6:04  Spanning-tree-(for-… STP   60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00  Cost = 0  Port = 0x8004
23 34.083096   Cisco_3a:f6:04  Spanning-tree-(for-… STP   60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00  Cost = 0  Port = 0x8004
24 36.087972   Cisco_3a:f6:04  Spanning-tree-(for-… STP   60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00  Cost = 0  Port = 0x8004
25 38.092842   Cisco_3a:f6:04  Spanning-tree-(for-… STP   60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00  Cost = 0  Port = 0x8004
26 40.102869   Cisco_3a:f6:04  Spanning-tree-(for-… STP   60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00  Cost = 0  Port = 0x8004
27 41.238707   Cisco_3a:f6:04  Cisco_3a:f6:04  LOOP   60 Reply
28 42.102609   Cisco_3a:f6:04  Spanning-tree-(for-… STP   60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00  Cost = 0  Port = 0x8004
29 44.107462   Cisco_3a:f6:04  Spanning-tree-(for-… STP   60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00  Cost = 0  Port = 0x8004
30 46.117556   Cisco_3a:f6:04  Spanning-tree-(for-… STP   60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00  Cost = 0  Port = 0x8004
31 48.117333   Cisco_3a:f6:04  Spanning-tree-(for-… STP   60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00  Cost = 0  Port = 0x8004
32 50.122162   Cisco_3a:f6:04  Spanning-tree-(for-… STP   60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00  Cost = 0  Port = 0x8004
```

Part7-2

Tux1

4	3.775052	Cisco_3a:f6:03	CDP/VTP/DTP/PAgP/UD…	CDP	435 Device ID: tux-sw5 Port ID: FastEthernet0/1
5	4.244606	Cisco_3a:f6:03	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
6	6.254472	Cisco_3a:f6:03	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
7	8.254218	Cisco_3a:f6:03	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
8	10.007589	Cisco_3a:f6:03	Cisco_3a:f6:03	LOOP	60 Reply
9	10.259091	Cisco_3a:f6:03	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
10	12.268990	Cisco_3a:f6:03	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
11	14.268775	Cisco_3a:f6:03	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
12	16.273659	Cisco_3a:f6:03	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
13	18.278448	Cisco_3a:f6:03	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003

Tux4

7	9.297806	Cisco_3a:f6:06	CDP/VTP/DTP/PAgP/UD…	CDP	435 Device ID: tux-sw5 Port ID: FastEthernet0/4
8	10.020509	Cisco_3a:f6:06	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8006
9	12.025144	Cisco_3a:f6:06	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8006
10	14.034097	Cisco_3a:f6:06	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8006
11	16.035040	Cisco_3a:f6:06	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8006
12	17.745073	Cisco_3a:f6:06	Cisco_3a:f6:06	LOOP	60 Reply
13	18.039964	Cisco_3a:f6:06	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8006
14	20.049162	Cisco_3a:f6:06	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8006
15	22.048012	Cisco_3a:f6:06	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8006
16	24.052999	Cisco_3a:f6:06	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8006
17	26.064019	Cisco_3a:f6:06	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8006

Tux2

23	36.087935	Cisco_3a:f6:04	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0
24	36.178877	172.16.51.1	172.16.51.255	ICMP	98 Echo (ping) request id=0x7e58, seq=1/256, ttl=64 (no respo
25	37.177883	172.16.51.1	172.16.51.255	ICMP	98 Echo (ping) request id=0x7e58, seq=2/512, ttl=64 (no respo
26	38.092818	Cisco_3a:f6:04	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0
27	38.176881	172.16.51.1	172.16.51.255	ICMP	98 Echo (ping) request id=0x7e58, seq=3/768, ttl=64 (no respo
28	39.176017	172.16.51.1	172.16.51.255	ICMP	98 Echo (ping) request id=0x7e58, seq=4/1024, ttl=64 (no resp

Exp3- 4

```
tux51:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.50.254   0.0.0.0         UG    0      0        0 eth0
172.16.50.0     172.16.50.254   255.255.255.0   UG    0      0        0 eth0
172.16.50.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
172.16.51.0     172.16.50.254   255.255.255.0   UG    0      0        0 eth0
tux51:~#
```

```
tux52:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.51.253   0.0.0.0         UG    0      0        0 eth0
172.16.50.0     172.16.51.253   255.255.255.0   UG    0      0        0 eth0
172.16.51.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
tux52:~#
```

```
tux54:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.50.0     172.16.50.1     255.255.255.0   UG    0      0        0 eth0
172.16.50.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
172.16.51.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
tux54:~#
```

Exp3 - 5

13 10.605589	G-ProCom_8b:e4:a7	HewlettP_c3:78:70	ARP	42	Who has 172.16.50.254? Tell 172.16.50.1
14 10.605927	HewlettP_c3:78:70	G-ProCom_8b:e4:a7	ARP	60	172.16.50.254 is at 00:21:5a:c3:78:70
15 10.609851	172.16.50.1	193.136.28.10	DNS	65	Standard query 0x0faa AAAA tux51
16 10.610103	172.16.50.254	172.16.50.1	ICMP	93	Destination unreachable (Network unreachable)
17 12.029199	Cisco_3a:f6:03	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
18 14.039132	Cisco_3a:f6:03	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
19 15.615878	HewlettP_c3:78:70	G-ProCom_8b:e4:a7	ARP	60	Who has 172.16.50.1? Tell 172.16.50.254
20 15.615903	G-ProCom_8b:e4:a7	HewlettP_c3:78:70	ARP	42	172.16.50.1 is at 00:0f:fe:8b:e4:a7
21 15.717015	172.16.50.1	172.16.1.1	DNS	81	Standard query 0x72be AAAA tux51.netlab.fe.up.pt
22 15.717235	172.16.50.254	172.16.50.1	ICMP	109	Destination unreachable (Network unreachable)
23 16.038704	Cisco_3a:f6:03	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
24 18.043803	Cisco_3a:f6:03	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
25 20.053703	Cisco_3a:f6:03	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
26 20.308670	Cisco_3a:f6:03	Cisco_3a:f6:03	LOOP	60	Reply
27 20.722134	172.16.50.1	193.136.28.10	DNS	81	Standard query 0x72be AAAA tux51.netlab.fe.up.pt
28 20.722507	172.16.50.254	172.16.50.1	ICMP	109	Destination unreachable (Network unreachable)
29 22.053357	Cisco_3a:f6:03	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
30 23.848824	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x7579, seq=1/256, ttl=64 (reply in 31)
31 23.849206	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x7579, seq=1/256, ttl=64 (request in 30)
32 24.058306	Cisco_3a:f6:03	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
33 24.847831	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x7579, seq=2/512, ttl=64 (reply in 34)
34 24.848181	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x7579, seq=2/512, ttl=64 (request in 33)
…	…	…	…	…	…
71 40.185482	172.16.50.1	172.16.51.253	ICMP	98	Echo (ping) request id=0x758a, seq=1/256, ttl=64 (reply in 72)
72 40.185698	172.16.51.253	172.16.50.1	ICMP	98	Echo (ping) reply id=0x758a, seq=1/256, ttl=64 (request in 71)
73 40.328715	Cisco_3a:f6:03	Cisco_3a:f6:03	LOOP	60	Reply
74 40.742438	172.16.50.1	193.136.28.10	DNS	65	Standard query 0xfa69 AAAA tux51
75 40.742783	172.16.50.254	172.16.50.1	ICMP	93	Destination unreachable (Network unreachable)
76 41.185300	172.16.50.1	172.16.51.253	ICMP	98	Echo (ping) request id=0x758a, seq=2/512, ttl=64 (reply in 77)
77 41.185647	172.16.51.253	172.16.50.1	ICMP	98	Echo (ping) reply id=0x758a, seq=2/512, ttl=64 (request in 76)

104 50.642379	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x7592, seq=2/512, ttl=64 (reply in 105)
105 50.642844	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x7592, seq=2/512, ttl=63 (request in 104)
106 50.752531	172.16.50.1	193.136.28.10	DNS	65	Standard query 0xfa69 AAAA tux51
107 50.752736	172.16.50.254	172.16.50.1	ICMP	93	Destination unreachable (Network unreachable)
108 51.200181	HewlettP_c3:78:70	G-ProCom_8b:e4:a7	ARP	60	Who has 172.16.50.1? Tell 172.16.50.254
109 51.200202	G-ProCom_8b:e4:a7	HewlettP_c3:78:70	ARP	42	172.16.50.1 is at 00:0f:fe:8b:e4:a7
110 51.643706	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x7592, seq=3/768, ttl=64 (reply in 111)
111 51.644186	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x7592, seq=3/768, ttl=63 (request in 110)

Exp3 - 8

Eth0

322 221.262076	172.16.50.1	193.136.28.10	DNS	81	Standard query 0xb196 AAAA tux51.netlab.fe.up.pt
323 221.262125	172.16.50.254	172.16.50.1	ICMP	109	Destination unreachable (Network unreachable)
324 221.497056	Cisco_3a:f6:06	Cisco_3a:f6:06	LOOP	60	Reply
325 222.562407	Cisco_3a:f6:06	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8006
326 224.567013	Cisco_3a:f6:06	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8006
327 226.259940	172.16.50.1	192.168.16.39	TCP	74	51242 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=18757346 TSecr=0 WS=128
328 226.259986	172.16.50.254	172.16.50.1	ICMP	102	Destination unreachable (Network unreachable)
329 226.267134	172.16.50.1	172.16.1.1	DNS	65	Standard query 0x7ff3 AAAA tux51
330 226.267182	172.16.50.254	172.16.50.1	ICMP	93	Destination unreachable (Network unreachable)
331 226.269256	172.16.50.1	172.16.1.1	DNS	81	Standard query 0xd209 AAAA tux51.netlab.fe.up.pt
332 226.571848	Cisco_3a:f6:06	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8006
333 227.220099	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x791e, seq=1/256, ttl=64 (reply in 334)
334 227.220399	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x791e, seq=1/256, ttl=63 (request in 333)
335 228.220382	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x791e, seq=2/512, ttl=64 (reply in 336)
336 228.220575	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x791e, seq=2/512, ttl=63 (request in 335)
337 228.576698	Cisco_3a:f6:06	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8006

Eth1

102 111.768741	Cisco_3a:f6:0f	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x800f
103 112.153793	172.16.51.1	172.16.1.1	DNS	65	Standard query 0x64d4 AAAA tux52
104 112.153836	172.16.51.253	172.16.51.1	ICMP	93	Destination unreachable (Network unreachable)
105 113.713955	Cisco_3a:f6:0f	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x800f
106 115.718555	Cisco_3a:f6:0f	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x800f
107 117.158728	172.16.51.1	193.136.28.10	DNS	65	Standard query 0x64d4 AAAA tux52
108 117.158771	172.16.51.253	172.16.51.1	ICMP	93	Destination unreachable (Network unreachable)
109 117.723379	Cisco_3a:f6:0f	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x800f
110 119.728117	Cisco_3a:f6:0f	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x800f
111 120.094229	Cisco_3a:f6:0f	Cisco_3a:f6:0f	LOOP	60	Reply
112 120.978203	Cisco_3a:f6:0f	CDP/VTP/DTP/PAgP/UD…	CDP	436	Device ID: tux-sw5 Port ID: FastEthernet0/13
113 121.733048	Cisco_3a:f6:0f	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x800f

182	205.926449	172.16.51.1	193.136.28.10	DNS	86 Standard query 0x8e1d PTR 253.51.16.172.in-addr.arpa
183	205.926488	172.16.51.253	172.16.51.1	ICMP	114 Destination unreachable (Network unreachable)
184	205.953200	Cisco_3a:f6:0f	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x800f
185	207.958062	Cisco_3a:f6:0f	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x800f
186	209.968021	Cisco_3a:f6:0f	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x800f
187	210.144368	Cisco_3a:f6:0f	Cisco_3a:f6:0f	LOOP	60 Reply
188	210.931508	172.16.51.1	172.16.1.1	DNS	86 Standard query 0xbf67 PTR 254.51.16.172.in-addr.arpa
189	210.931546	172.16.51.253	172.16.51.1	ICMP	114 Destination unreachable (Network unreachable)
190	211.967819	Cisco_3a:f6:0f	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x800f
191	213.972491	Cisco_3a:f6:0f	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x800f
192	215.982643	Cisco_3a:f6:0f	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x800f
193	217.982391	Cisco_3a:f6:0f	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x800f
194	219.987021	Cisco_3a:f6:0f	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x800f
195	220.138533	Cisco_3a:f6:0f	Cisco_3a:f6:0f	LOOP	60 Reply
196	221.997026	Cisco_3a:f6:0f	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x800f
197	223.996702	Cisco_3a:f6:0f	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x800f
198	225.861289	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request id=0x791e, seq=1/256, ttl=63 (reply in 201)
199	225.861427	HewlettP_61:2f:d6	Broadcast	ARP	60 Who has 172.16.51.253? Tell 172.16.51.1
200	225.861449	Kye_08:d5:b0	HewlettP_61:2f:d6	ARP	42 172.16.51.253 is at 00:c0:df:08:d5:b0
201	225.861558	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply id=0x791e, seq=1/256, ttl=64 (request in 198)
202	226.001574	Cisco_3a:f6:0f	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x800f
203	226.861580	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request id=0x791e, seq=2/512, ttl=63 (reply in 204)
204	226.861716	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply id=0x791e, seq=2/512, ttl=64 (request in 203)
205	227.861567	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request id=0x791e, seq=3/768, ttl=63 (reply in 206)
206	227.861679	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply id=0x791e, seq=3/768, ttl=64 (request in 205)

Exp4 -4 part1

137	69.001459	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
138	70.001447	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
139	70.779626	Cisco_3a:f6:04	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0
140	71.001470	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
141	71.451148	Cisco_3a:f6:04	CDP/VTP/DTP/PAgP/UD…	CDP	453 Device ID: tux-sw5 Port ID: FastEthernet0/2
142	72.001515	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
143	72.789580	Cisco_3a:f6:04	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0
144	73.001512	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
145	74.001538	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
146	74.789411	Cisco_3a:f6:04	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0
147	74.940709	Cisco_3a:f6:04	Cisco_3a:f6:04	LOOP	60 Reply
148	75.001569	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
149	76.001576	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
150	76.794327	Cisco_3a:f6:04	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0
151	77.001598	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
152	78.001628	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253

Exp4 - part2

301	161.003362	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
302	162.003398	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
303	163.003407	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
304	163.004313	Cisco_3a:f6:04	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
305	164.003440	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
306	164.992826	Cisco_3a:f6:04	Cisco_3a:f6:04	LOOP	60 Reply
307	165.003463	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
308	165.013899	Cisco_3a:f6:04	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
309	166.003474	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
310	167.003497	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253

Exp4 - part3

1	0.000000	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
2	1.000022	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
3	1.250576	Cisco_3a:f6:04	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
4	1.706414	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) request id=0x168f, seq=1/256, ttl=64 (no response found!)
5	2.000034	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
6	2.705747	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) request id=0x168f, seq=2/512, ttl=64 (no response found!)
7	3.000065	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
8	3.255433	Cisco_3a:f6:04	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
9	3.705747	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) request id=0x168f, seq=3/768, ttl=64 (no response found!)
10	4.000075	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
11	4.705737	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) request id=0x168f, seq=4/1024, ttl=64 (no response found!)
12	4.705882	172.16.51.253	172.16.51.1	ICMP	126 Redirect (Redirect for host)
13	4.705913	HewlettP_61:2f:d6	Broadcast	ARP	42 Who has 172.16.51.254? Tell 172.16.51.1
14	5.000097	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
15	5.260305	Cisco_3a:f6:04	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
16	5.705722	HewlettP_61:2f:d6	Broadcast	ARP	42 Who has 172.16.51.254? Tell 172.16.51.1
17	5.705752	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) request id=0x168f, seq=5/1280, ttl=64 (no response found!)
18	6.000138	Kye_08:d5:b0	Broadcast	ARP	60 Who has 172.16.51.254? Tell 172.16.51.253
19	6.705726	HewlettP_61:2f:d6	Broadcast	ARP	42 Who has 172.16.51.254? Tell 172.16.51.1

Exp5

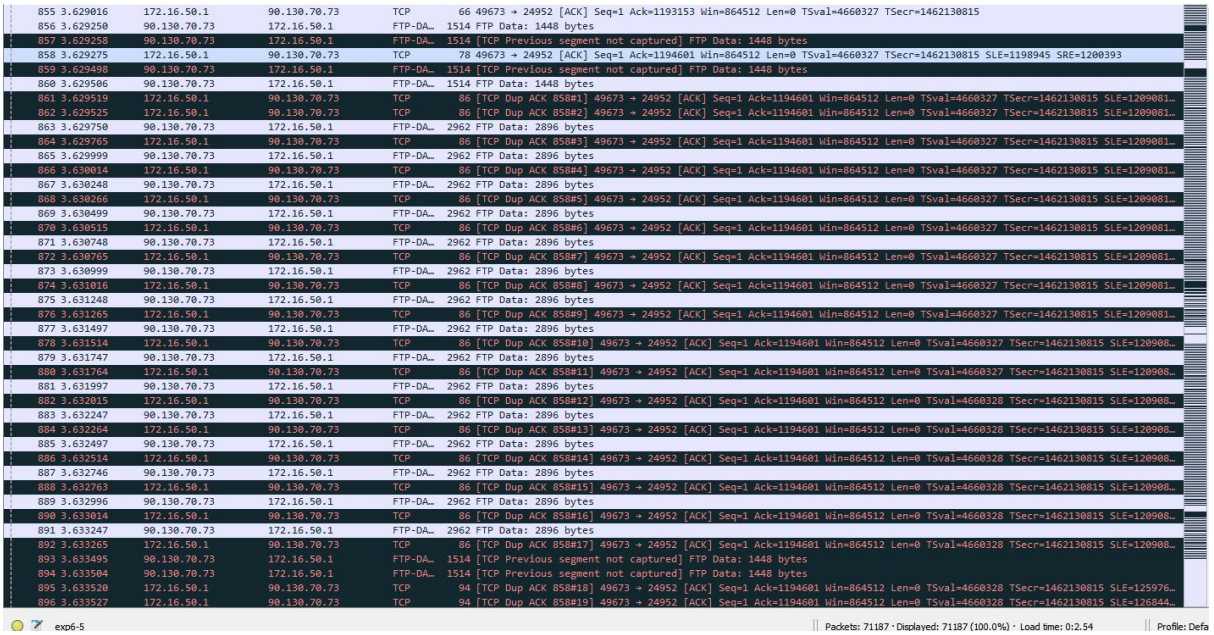
```
tux52:~# traceroute 172.16.50.1
traceroute to 172.16.50.1 (172.16.50.1), 30 hops max, 60 byte packets
```

8	5.001665	HewlettP_a6:a4:f8	G-ProCom_8b:e4:ef	ARP	60 Who has 172.16.50.1? Tell 172.16.50.254
9	5.001691	G-ProCom_8b:e4:ef	HewlettP_a6:a4:f8	ARP	42 172.16.50.1 is at 00:0f:fe:8b:e4:ef
10	5.004896	172.16.50.1	172.16.1.1	DNS	87 Standard query 0x9ab9 PTR 234.211.58.216.in-addr.arpa
11	5.006113	172.16.1.1	172.16.50.1	DNS	303 Standard query response 0x9ab9 PTR 234.211.58.216.in-addr.arpa PTR mad01s24-in-f10.1e100.net PTR mad01s24-in-f234.1e1…
12	5.006404	172.16.50.1	172.16.1.1	DNS	86 Standard query 0x912f PTR 29.220.184.93.in-addr.arpa
13	5.008224	172.16.1.1	172.16.50.1	DNS	157 Standard query response 0x912f No such name PTR 29.220.184.93.in-addr.arpa SOA ns1.edgecastcdn.net
14	5.342013	172.16.50.1	172.16.1.1	DNS	74 Standard query 0xa960 A www.google.com
15	5.343073	172.16.1.1	172.16.50.1	DNS	226 Standard query response 0xa960 A www.google.com A 216.58.205.36 NS ns4.google.com NS ns2.google.com NS ns3.google.com…
16	5.343299	172.16.50.1	216.58.205.36	ICMP	98 Echo (ping) request id=0x14ea, seq=1/256, ttl=64 (reply in 17)
17	5.376567	216.58.205.36	172.16.50.1	ICMP	98 Echo (ping) reply id=0x14ea, seq=1/256, ttl=48 (request in 16)
18	5.376848	172.16.50.1	172.16.1.1	DNS	86 Standard query 0x558c PTR 36.205.58.216.in-addr.arpa
19	5.377932	172.16.1.1	172.16.50.1	DNS	300 Standard query response 0x558c PTR 36.205.58.216.in-addr.arpa PTR mil04s24-in-f4.1e100.net PTR mil04s24-in-f36.1e100.…
20	5.381851	Cisco_3a:fc:03	Cisco_3a:fc:03	LOOP	60 Reply
21	5.815031	172.16.50.1	93.184.220.29	TCP	66 33683 → 80 [ACK] Seq=1 Ack=1 Win=241 Len=0 TSval=528056 TSecr=327801678
22	5.859734	93.184.220.29	172.16.50.1	TCP	66 [TCP ACKed unseen segment] 80 → 33683 [ACK] Seq=1 Ack=2 Win=285 Len=0 TSval=327804190 TSecr=505453
23	6.182325	Cisco_3a:fc:03	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
24	6.345045	172.16.50.1	216.58.205.36	ICMP	98 Echo (ping) request id=0x14ea, seq=2/512, ttl=64 (reply in 25)
25	6.377810	216.58.205.36	172.16.50.1	ICMP	98 Echo (ping) reply id=0x14ea, seq=2/512, ttl=48 (request in 24)
26	7.346908	172.16.50.1	216.58.205.36	ICMP	98 Echo (ping) request id=0x14ea, seq=3/768, ttl=64 (reply in 27)
27	7.379759	216.58.205.36	172.16.50.1	ICMP	98 Echo (ping) reply id=0x14ea, seq=3/768, ttl=48 (request in 26)
28	8.187142	Cisco_3a:fc:03	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:fc:00 Cost = 0 Port = 0x8003
29	8.348885	172.16.50.1	216.58.205.36	ICMP	98 Echo (ping) request id=0x14ea, seq=4/1024, ttl=64 (reply in 30)
30	8.381663	216.58.205.36	172.16.50.1	ICMP	98 Echo (ping) reply id=0x14ea, seq=4/1024, ttl=48 (request in 29)

Exp6 - part 2

1	0.000000	Cisco_3a:f6:03	CDP/VTP/DTP/PAgP/UD…	CDP	435 Device ID: tux-sw5 Port ID: FastEthernet0/1
2	0.363880	172.16.50.1	192.168.16.39	TCP	74 40862 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4422786 TSecr=0 WS=128
3	1.612301	172.16.50.1	192.168.16.39	TCP	74 40872 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4423098 TSecr=0 WS=128
4	1.799864	Cisco_3a:f6:03	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
5	2.611901	172.16.50.1	192.168.16.39	TCP	74 [TCP Retransmission] 40872 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4423348 TSecr=0 WS=128
6	3.799604	Cisco_3a:f6:03	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
7	4.359881	172.16.50.1	216.58.214.163	TCP	66 40942 → 80 [ACK] Seq=1 Ack=1 Win=240 Len=0 TSval=4423785 TSecr=1907988502
8	4.375653	216.58.214.163	172.16.50.1	TCP	66 [TCP ACKed unseen segment] 80 → 40942 [ACK] Seq=1 Ack=2 Win=170 Len=0 TSval=1907998518 TSecr=4421285
9	4.463870	172.16.50.1	216.58.205.67	TCP	66 54975 → 80 [ACK] Seq=1 Ack=1 Win=243 Len=0 TSval=4423811 TSecr=3996578497
10	4.498095	216.58.205.67	172.16.50.1	TCP	66 [TCP ACKed unseen segment] 80 → 54975 [ACK] Seq=1 Ack=2 Win=170 Len=0 TSval=3996588532 TSecr=4421311
11	4.540975	172.16.50.1	172.16.1.1	DNS	86 Standard query 0xc7fa PTR 67.205.58.216.in-addr.arpa
12	4.542262	172.16.1.1	172.16.50.1	DNS	300 Standard query response 0xc7fa PTR 67.205.58.216.in-addr.arpa PTR mil04s25-in-f67.1e100.net PTR mil04s25-in-f3.1e100.…
13	4.542533	172.16.50.1	172.16.1.1	DNS	86 Standard query 0x955c PTR 67.205.58.216.in-addr.arpa
14	4.543623	172.16.1.1	172.16.50.1	DNS	300 Standard query response 0x955c PTR 67.205.58.216.in-addr.arpa PTR mil04s25-in-f3.1e100.net PTR mil04s25-in-f67.1e100.…
15	4.543829	172.16.50.1	172.16.1.1	DNS	87 Standard query 0xd0b8 PTR 163.214.58.216.in-addr.arpa
16	4.544909	172.16.1.1	172.16.50.1	DNS	302 Standard query response 0xd0b8 PTR 163.214.58.216.in-addr.arpa PTR mad01s26-in-f163.1e100.net PTR mad01s26-in-f3.1e10…
17	4.545183	172.16.50.1	172.16.1.1	DNS	87 Standard query 0x9126 PTR 227.211.58.216.in-addr.arpa
18	4.546170	172.16.1.1	172.16.50.1	DNS	302 Standard query response 0x9126 PTR 227.211.58.216.in-addr.arpa PTR mad01s24-in-f227.1e100.net PTR mad01s24-in-f3.1e10…
19	4.546467	172.16.50.1	172.16.1.1	DNS	87 Standard query 0x1319 PTR 174.214.58.216.in-addr.arpa
20	4.547547	172.16.1.1	172.16.50.1	DNS	303 Standard query response 0x1319 PTR 174.214.58.216.in-addr.arpa PTR mad01s26-in-f174.1e100.net PTR mad01s26-in-f14.1e1…
21	4.615884	172.16.50.1	192.168.16.39	TCP	74 [TCP Retransmission] 40872 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4423849 TSecr=0 WS=128
22	4.707888	172.16.50.1	192.168.16.39	TCP	74 40860 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4423872 TSecr=0 WS=128
23	5.804646	Cisco_3a:f6:03	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
24	7.814433	Cisco_3a:f6:03	Spanning-tree-(for-…	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
25	8.136422	172.16.50.1	172.16.1.1	DNS	69 Standard query 0x2d7b A ftp.up.pt
26	8.137827	172.16.1.1	172.16.50.1	DNS	355 Standard query response 0x2d7b A ftp.up.pt CNAME mirrors.up.pt A 193.137.29.15 NS ns3.up.pt NS ns1.up.pt NS ns4.up.pt…
27	8.137987	172.16.50.1	193.137.29.15	TCP	74 46411 → 21 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4424729 TSecr=0 WS=128
28	8.140849	193.137.29.15	172.16.50.1	TCP	74 21 → 46411 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1380 SACK_PERM=1 TSval=590676629 TSecr=4424729 WS=128
29	8.140893	172.16.50.1	193.137.29.15	TCP	66 46411 → 21 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=4424730 TSecr=590676629
30	8.144997	193.137.29.15	172.16.50.1	FTP	139 Response: 220-Welcome to the University of Porto's mirror archive (mirrors.up.pt)
31	8.145016	193.137.29.15	172.16.50.1	FTP	135 Response: 220----------
32	8.145020	193.137.29.15	172.16.50.1	FTP	72 Response: 220-
33	8.145040	172.16.50.1	193.137.29.15	TCP	66 46411 → 21 [ACK] Seq=1 Ack=74 Win=29312 Len=0 TSval=4424731 TSecr=590676630
34	8.145051	172.16.50.1	193.137.29.15	TCP	66 46411 → 21 [ACK] Seq=1 Ack=143 Win=29312 Len=0 TSval=4424731 TSecr=590676630
35	8.145070	172.16.50.1	193.137.29.15	TCP	66 46411 → 21 [ACK] Seq=1 Ack=149 Win=29312 Len=0 TSval=4424731 TSecr=590676630
36	8.145291	193.137.29.15	172.16.50.1	FTP	151 Response: 220-All connections and transfers are logged. The max number of connections is 200.
37	8.145303	193.137.29.15	172.16.50.1	FTP	72 Response: 220-
38	8.145306	193.137.29.15	172.16.50.1	FTP	140 Response: 220-For more information please visit our website: http://mirrors.up.pt/
39	8.145310	193.137.29.15	172.16.50.1	FTP	127 Response: 220-Questions and comments can be sent to mirrors@uporto.pt
40	8.145311	172.16.50.1	193.137.29.15	TCP	66 46411 → 21 [ACK] Seq=1 Ack=234 Win=29312 Len=0 TSval=4424731 TSecr=590676630
58	8.284203	193.137.29.15	172.16.50.1	FTP	103 Response: 250 Directory successfully changed.
59	8.284276	172.16.50.1	193.137.29.15	FTP	72 Request: PASV
60	8.286507	193.137.29.15	172.16.50.1	FTP	118 Response: 227 Entering Passive Mode (193,137,29,15,221,204).
61	8.286628	172.16.50.1	193.137.29.15	TCP	74 60710 → 56780 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4424766 TSecr=0 WS=128
62	8.288354	193.137.29.15	172.16.50.1	TCP	74 56780 → 60710 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1380 SACK_PERM=1 TSval=590676666 TSecr=4424766 WS=128
63	8.288388	172.16.50.1	193.137.29.15	TCP	66 60710 → 56780 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=4424767 TSecr=590676666
64	8.288414	172.16.50.1	193.137.29.15	FTP	82 Request: RETR timestamp
65	8.291362	193.137.29.15	172.16.50.1	FTP	134 Response: 150 Opening BINARY mode data connection for timestamp (186 bytes).
66	8.300661	193.137.29.15	172.16.50.1	FTP-DA…	252 FTP Data: 186 bytes
67	8.300681	193.137.29.15	172.16.50.1	TCP	66 56780 → 60710 [FIN, ACK] Seq=187 Ack=1 Win=29056 Len=0 TSval=590676669 TSecr=4424767
68	8.300708	172.16.50.1	193.137.29.15	TCP	66 60710 → 56780 [ACK] Seq=1 Ack=187 Win=30336 Len=0 TSval=4424770 TSecr=590676669
69	8.327874	172.16.50.1	193.137.29.15	TCP	66 46411 → 21 [ACK] Seq=66 Ack=607 Win=29312 Len=0 TSval=4424777 TSecr=590676666
70	8.339871	172.16.50.1	193.137.29.15	TCP	66 60710 → 56780 [ACK] Seq=1 Ack=188 Win=30336 Len=0 TSval=4424780 TSecr=590676669
71	8.341483	193.137.29.15	172.16.50.1	FTP	90 Response: 226 Transfer complete.
72	8.341551	172.16.50.1	193.137.29.15	TCP	66 46411 → 21 [ACK] Seq=66 Ack=631 Win=29312 Len=0 TSval=4424780 TSecr=590676679
73	8.341606	172.16.50.1	193.137.29.15	TCP	66 60710 → 56780 [FIN, ACK] Seq=1 Ack=188 Win=30336 Len=0 TSval=4424780 TSecr=590676669
74	8.341634	172.16.50.1	193.137.29.15	FTP	72 Request: QUIT
75	8.342955	193.137.29.15	172.16.50.1	TCP	66 56780 → 60710 [ACK] Seq=188 Ack=2 Win=29056 Len=0 TSval=590676679 TSecr=4424780
76	8.343576	193.137.29.15	172.16.50.1	FTP	80 Response: 221 Goodbye.
77	8.343586	193.137.29.15	172.16.50.1	TCP	66 21 → 46411 [FIN, ACK] Seq=645 Ack=72 Win=29056 Len=0 TSval=590676679 TSecr=4424780
78	8.343644	172.16.50.1	193.137.29.15	TCP	66 46411 → 21 [FIN, ACK] Seq=72 Ack=646 Win=29312 Len=0 TSval=4424780 TSecr=590676679
79	8.345461	193.137.29.15	172.16.50.1	TCP	66 21 → 46411 [ACK] Seq=646 Ack=73 Win=29056 Len=0 TSval=590676680 TSecr=4424780
80	8.387892	172.16.50.1	192.168.16.39	TCP	74 [TCP Retransmission] 40862 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4424792 TSecr=0 WS=128
81	8.627885	172.16.50.1	192.168.16.39	TCP	74 [TCP Retransmission] 40872 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4424852 TSecr=0 WS=128

Exp6 - part 5



855	3.629016	172.16.50.1	90.130.70.73	TCP	66 49673 → 24952 [ACK] Seq=1 Ack=1193153 Win=864512 Len=0 TSval=4660327 TSecr=1462130815
856	3.629250	90.130.70.73	172.16.50.1	FTP-DA…	1514 FTP Data: 1448 bytes
857	3.629258	90.130.70.73	172.16.50.1	FTP-DA…	1514 [TCP Previous segment not captured] FTP Data: 1448 bytes
858	3.629275	172.16.50.1	90.130.70.73	TCP	78 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660327 TSecr=1462130815 SLE=1198945 SRE=1200393
859	3.629498	90.130.70.73	172.16.50.1	FTP-DA…	1514 [TCP Previous segment not captured] FTP Data: 1448 bytes
860	3.629506	90.130.70.73	172.16.50.1	FTP-DA…	1514 FTP Data: 1448 bytes
861	3.629519	172.16.50.1	90.130.70.73	TCP	86 [TCP Dup ACK 858#1] 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660327 TSecr=1462130815 SLE=1209081…
862	3.629525	172.16.50.1	90.130.70.73	TCP	86 [TCP Dup ACK 858#2] 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660327 TSecr=1462130815 SLE=1209081…
863	3.629750	90.130.70.73	172.16.50.1	FTP-DA…	2962 FTP Data: 2896 bytes
864	3.629765	172.16.50.1	90.130.70.73	TCP	86 [TCP Dup ACK 858#3] 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660327 TSecr=1462130815 SLE=1209081…
865	3.629999	90.130.70.73	172.16.50.1	FTP-DA…	2962 FTP Data: 2896 bytes
866	3.630014	172.16.50.1	90.130.70.73	TCP	86 [TCP Dup ACK 858#4] 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660327 TSecr=1462130815 SLE=1209081…
867	3.630248	90.130.70.73	172.16.50.1	FTP-DA…	2962 FTP Data: 2896 bytes
868	3.630266	172.16.50.1	90.130.70.73	TCP	86 [TCP Dup ACK 858#5] 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660327 TSecr=1462130815 SLE=1209081…
869	3.630499	90.130.70.73	172.16.50.1	FTP-DA…	2962 FTP Data: 2896 bytes
870	3.630515	172.16.50.1	90.130.70.73	TCP	86 [TCP Dup ACK 858#6] 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660327 TSecr=1462130815 SLE=1209081…
871	3.630748	90.130.70.73	172.16.50.1	FTP-DA…	2962 FTP Data: 2896 bytes
872	3.630765	172.16.50.1	90.130.70.73	TCP	86 [TCP Dup ACK 858#7] 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660327 TSecr=1462130815 SLE=1209081…
873	3.630999	90.130.70.73	172.16.50.1	FTP-DA…	2962 FTP Data: 2896 bytes
874	3.631016	172.16.50.1	90.130.70.73	TCP	86 [TCP Dup ACK 858#8] 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660327 TSecr=1462130815 SLE=1209081…
875	3.631248	90.130.70.73	172.16.50.1	FTP-DA…	2962 FTP Data: 2896 bytes
876	3.631265	172.16.50.1	90.130.70.73	TCP	86 [TCP Dup ACK 858#9] 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660327 TSecr=1462130815 SLE=1209081…
877	3.631497	90.130.70.73	172.16.50.1	FTP-DA…	2962 FTP Data: 2896 bytes
878	3.631514	172.16.50.1	90.130.70.73	TCP	86 [TCP Dup ACK 858#10] 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660327 TSecr=1462130815 SLE=120908…
879	3.631747	90.130.70.73	172.16.50.1	FTP-DA…	2962 FTP Data: 2896 bytes
880	3.631764	172.16.50.1	90.130.70.73	TCP	86 [TCP Dup ACK 858#11] 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660327 TSecr=1462130815 SLE=120908…
881	3.631997	90.130.70.73	172.16.50.1	FTP-DA…	2962 FTP Data: 2896 bytes
882	3.632015	172.16.50.1	90.130.70.73	TCP	86 [TCP Dup ACK 858#12] 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660328 TSecr=1462130815 SLE=120908…
883	3.632247	90.130.70.73	172.16.50.1	FTP-DA…	2962 FTP Data: 2896 bytes
884	3.632264	172.16.50.1	90.130.70.73	TCP	86 [TCP Dup ACK 858#13] 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660328 TSecr=1462130815 SLE=120908…
885	3.632497	90.130.70.73	172.16.50.1	FTP-DA…	2962 FTP Data: 2896 bytes
886	3.632514	172.16.50.1	90.130.70.73	TCP	86 [TCP Dup ACK 858#14] 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660328 TSecr=1462130815 SLE=120908…
887	3.632746	90.130.70.73	172.16.50.1	FTP-DA…	2962 FTP Data: 2896 bytes
888	3.632763	172.16.50.1	90.130.70.73	TCP	86 [TCP Dup ACK 858#15] 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660328 TSecr=1462130815 SLE=120908…
889	3.632996	90.130.70.73	172.16.50.1	FTP-DA…	2962 FTP Data: 2896 bytes
890	3.633014	172.16.50.1	90.130.70.73	TCP	86 [TCP Dup ACK 858#16] 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660328 TSecr=1462130815 SLE=120908…
891	3.633247	90.130.70.73	172.16.50.1	FTP-DA…	2962 FTP Data: 2896 bytes
892	3.633265	172.16.50.1	90.130.70.73	TCP	86 [TCP Dup ACK 858#17] 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660328 TSecr=1462130815 SLE=120908…
893	3.633495	90.130.70.73	172.16.50.1	FTP-DA…	1514 [TCP Previous segment not captured] FTP Data: 1448 bytes
894	3.633504	90.130.70.73	172.16.50.1	FTP-DA…	1514 [TCP Previous segment not captured] FTP Data: 1448 bytes
895	3.633520	172.16.50.1	90.130.70.73	TCP	94 [TCP Dup ACK 858#18] 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660328 TSecr=1462130815 SLE=125976…
896	3.633527	172.16.50.1	90.130.70.73	TCP	94 [TCP Dup ACK 858#19] 49673 → 24952 [ACK] Seq=1 Ack=1194601 Win=864512 Len=0 TSval=4660328 TSecr=1462130815 SLE=126844…

exp6-5 Packets: 71187 · Displayed: 71187 (100.0%) · Load time: 0:2.54 Profile: Defa