Faculdade de Engenharia da Universidade do Porto

2.º projeto laboratorial

Configuração e análise de uma rede

Redes de Computadores (L.EIC025) 2024/2025

Licenciatura em Engenharia Informática e Computação

Manuel Alberto Pereira Ricardo (Co-regente da unidade curricular)

Rui Pedro de Magalhães Claro Prior (Co-regente da unidade curricular)

Eduardo Nuno Moreira Soares de Almeida (Professor das aulas laboratoriais)

Turma 2:

Guilherme Duarte Silva Matos up202208755@up.pt

João Vítor da Costa Ferreira up202208393@up.pt

Índice

Resumo

O segundo projeto laboratorial de Redes de Computadores consiste na configuração e estudo de uma rede interna que permite a comunicação com servidores FTP dentro da rede externa do laboratório.

Para tal, é usado um conjunto de computadores Linux (tux), um switch e um router, todos interligados por rotas e IPs manualmente definidos para permitir que qualquer tux consiga comunicar com o servidor FTP do laboratório usando um cliente também desenvolvido durante este projeto na linguagem C.

Os temas abordados neste projeto incluem:

- Uso de "sockets" em C;
- Protocolo FTP;
- Configuração de rotas em computadores Linux e routers MikroTik;
- Configuração de "bridges" em switches "MikroTik";
- "Network address translation";
- Estrutura dos pacotes TCP;
- Servidores DNS e configuração de "hosts";
- Fases e desempenho de conexões TCP.

1. Introdução

Este relatório visa documentar o segundo projeto prático realizado durante as aulas laboratoriais. Foi desenvolvido uma aplicação de transferência (2.) usada para baixar um ficheiro de um servidor FTP, incluindo a arquitetura do código e um exemplo de execução. No apêndice E está disponível o código-fonte da aplicação. Também foi configurado e estudado uma rede interna (3.), tal como sugerido pelas seis experiências propostas. Para cada uma delas apresenta-se:

- Um esquema da estrutura de rede;
- Os objetivos compreendidos com as instruções dadas;
- Os novos comandos introduzidos, bem como o seu significado;
- As conclusões e explicações retiradas dos registos do "Wireshark".

Também é apresentado nos apêndices mais informações sobre cada experiência:

- (A.) Os comandos executados durante a mesma;
- (B.) Os registos capturados pelo "Wireshark", tal como mencionado nas instruções;
- (C.) A resolução das questões propostas no guião.

No apêndice D é disponibilizado o guião usado durante a demonstração, incluindo os "scripts" executados para agilizar a configuração da rede.

2. Aplicação de transferência

2.1. Arquitetura

A aplicação desenvolvida no âmbito deste projeto é dividida em 2 secções:

- "Parser" - responsável por tratar da "string" do URL, dividindo-a em componentes necessários a uma conexão FTP;
- "FTP Client" - estabelece a ligação com o servidor, analisa os pacotes de FTP recebidos e envia pacotes relevantes para realizar a transferência.

O cliente é dividido em três funções relevantes: *ftpConnect*, *ftpRetrieve*, *ftpDisconnect*. A primeira realiza o "login" ao servidor. A segunda realiza a transferência estabelecendo a conexão passiva. A última, fecha a conexão com o servidor, terminando o ciclo da aplicação. Estas são executadas pela ordem descrita.

2.2. Exemplo de uma transferência com sucesso

Realizou-se, como exemplo, a conexão aos “mirrors” da Universidade do Porto:

No.	Time	Source	Destination	Protocol	Lengtl	Info
2277	3.038667572	192.168.1.220	193.137.29.15	FTP	82	Request: USER anonymous
2297	3.056402960	193.137.29.15	192.168.1.220	FTP	139	Response: 220-Welcome to the University of Porto's mirror archive (mirrors.up.pt)
2300	3.057467991	193.137.29.15	192.168.1.220	FTP	141	Response: 220----------
2301	3.057468018	193.137.29.15	192.168.1.220	FTP	304	Response: 220-All connections and transfers are logged. The max number of connections is 200.
2302	3.057468125	193.137.29.15	192.168.1.220	FTP	72	Response: 220
2303	3.057468142	193.137.29.15	192.168.1.220	FTP	100	Response: 331 Please specify the password.
2308	3.057621534	192.168.1.220	193.137.29.15	FTP	82	Request: PASS anonymous
2323	3.077240529	193.137.29.15	192.168.1.220	FTP	89	Response: 230 Login successful.
2324	3.077317661	192.168.1.220	193.137.29.15	FTP	72	Request: PASV
2331	3.094367270	193.137.29.15	192.168.1.220	FTP	118	Response: 227 Entering Passive Mode (193,137,29,15,199,125).
2348	3.111159255	192.168.1.220	193.137.29.15	FTP	113	Request: RETR pub/gnu/emacs/elisp-manual-21-2.8.tar.gz
2369	3.129125388	193.137.29.15	192.168.1.220	FTP	169	Response: 150 Opening BINARY mode data connection for pub/gnu/emacs/elisp-manual-21-2.8.tar.gz (2455995 bytes).
4116	4.081593303	193.137.29.15	192.168.1.220	FTP	90	Response: 226 Transfer complete.
4240	4.248204787	192.168.1.220	193.137.29.15	FTP	72	Request: QUIT
4263	4.265381664	193.137.29.15	192.168.1.220	FTP	80	Response: 221 Goodbye.

Um *trace* da realização de uma transferência, no Wireshark.

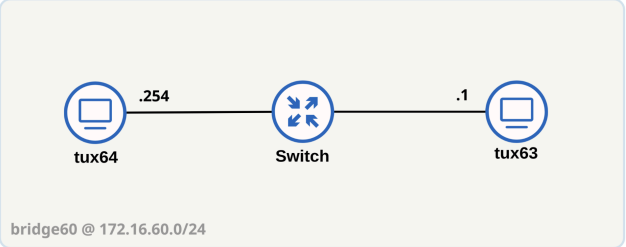
```
Unset
$ ./bin/download ftp://ftp.up.pt/pub/gnu/emacs/elisp-manual-21-2.8.tar.gz
      FTP URL:
      User: anonymous
      Password: anonymous
      Host: ftp.up.pt
      IP: 193.137.29.15:21
      URL path: pub/gnu/emacs/elisp-manual-21-2.8.tar.gz
        USER anonymous
      220-Welcome to the University of Porto's mirror archive (mirrors.up.pt)
      220----------------------------------------------------------------
      220-
      220-All connections and transfers are logged. The max number of connections is 200.
      220-
      220-For more information please visit our website: http://mirrors.up.pt/
      220-Questions and comments can be sent to mirrors@uporto.pt
      220-
      220-
      220
      331 Please specify the password.
        PASS anonymous
      230 Login successful.
        Authentication successful!
        PASV
      227 Entering Passive Mode (193,137,29,15,199,125).
        Trying to establish connection...
        Established passive connection.
        RETR pub/gnu/emacs/elisp-manual-21-2.8.tar.gz
      150 Opening BINARY mode data connection for pub/gnu/emacs/elisp-manual-21-2.8.tar.gz (2455995 bytes).
        Received preliminary message.
        Started passive download.
        Finished passive download.
      226 Transfer complete.
      221 Goodbye.
```

O *output* da aplicação aquando de uma transferência.

3. Configuração e análise da rede

Todas as experiências foram realizadas na bancada n.º 6.

3.1. Configurar uma rede IP



Objetivos da experiência

- Ligar dois computadores numa rede local usando um switch;
- Testar uma conexão usando ping;
- Configurar uma interface de rede de um computador usando ifconfig;
- Compreender as tabelas de rotas e de ARP;
- Capturar pacotes usando Wireshark;

Comandos principais de configuração

- `ifconfig <eth> <IP>`

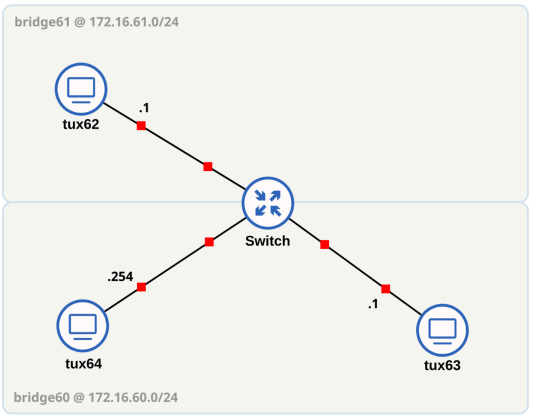
  Configura uma interface de rede <eth> para um dado <IP>, netmask e broadcast.

Análise dos registos

Quando o tux63 faz ping ao tux64, os seguintes pacotes são enviados:

1. Tux63 envia "ARP Request" para toda a rede local, já que não sabe o endereço MAC do tux64, apenas o seu endereço IP;
2. Tux64 responde com "ARP Reply", a indicar o seu endereço MAC;
3. Tux63, agora conhecedor do endereço MAC, envia "ICMP Echo Request" para o tux64, tal como instruído pelo comando ping;
4. Tux64 responde com "ICMP Echo Reply", assegurando que a conexão entre os dois é funcional.

3.2. Implementação de duas “bridges” num “switch”



Objetivos da experiência

- Configuração de um switch “MikroTik”;
- “Bridges” e o seu impacto na rede.

Comandos principais de configuração

- `/interface bridge add name=<bridge_name>`:
  Cria uma nova “bridge” vazia;
- `/interface bridge port remove`:
  Remove portas (mencionadas a partir do seu ID) da “bridge” associada;
- `/interface bridge port add bridge=<bridge_name> interface=<ether>`:
  Adiciona uma porta a uma “bridge”.

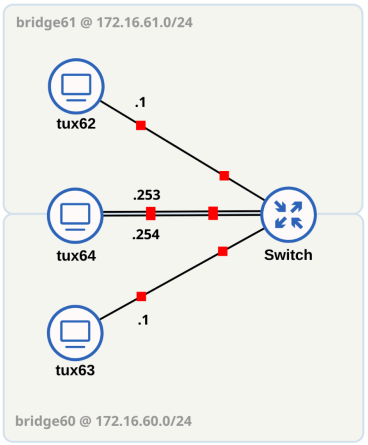
Análise dos registos

A partir dos pacotes “ICMP Echo”, comprova-se se existe ou não uma conexão entre dois computadores. Assim, os registos do Wireshark mostram que:

- O tux63 está conectado ao tux64, mas não ao tux62;
- Quando o tux63 faz ping ao endereço de broadcast, apenas o tux64 responde;
- Quando o tux62 faz ping ao endereço de broadcast, ninguém responde.

Tudo isto é devido à forma como as “bridges” foram configuradas anteriormente.

3.3. Configurar um router em Linux



Objetivos da experiência

- Tornar um computador Linux num router;
- Configurar e observar rotas em computadores Linux;

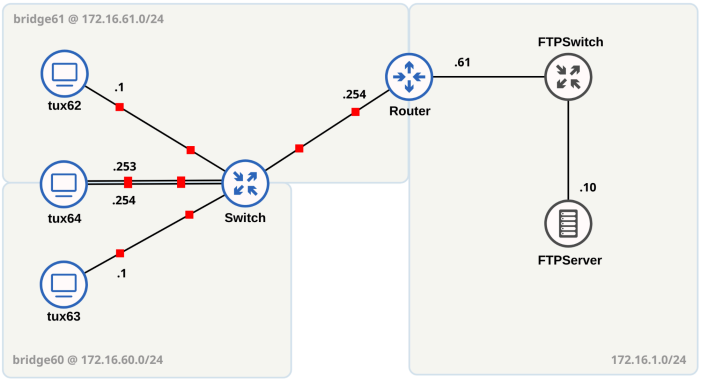
Comandos principais de configuração

- `route add -net <packetIP> gw <gatewayIP>`:
  Sempre que um computador receber um pacote com o destino <packetIP>, envia em vez para o endereço <gatewayIP>.

Análise dos registos

- Comprova-se a correta conexão de rede entre o tux63 e todas as outras interfaces de rede ao observar os "ICMP Echo Replys" emitidos;
- Ao realizar um ping do tux63 ao tux62 com as tabelas ARP apagadas, observa-se um pacote "ARP" em cada interface do tux64. Isto porque realiza a rota tux63 eth1 → tux64 eth1 → tux64 eth2 → tux62 eth1 e o tux64 tem conhecimento dos seus próprios endereços MAC. Mais especificamente, no eth1 do tux64 observa-se um "ARP Request" do tux63 e no eth2 um "ARP Reply" do tux64.

3.4. Configurar um router comercial e implementar NAT



Objetivos da experiência

- Interligar redes locais e configurar rotas usando um router MikroTik comercial;
- Observar as rotas que um pacote segue dependendo da configuração da rede;
- Compreender os pacotes "ICMP Redirect" enviados pelo router e como podem ajudar no desempenho da rede;
- Compreender a importância da Network Address Translation (NAT) dentro do router e o que acontece quando esta se encontra desligada.

Comandos principais de configuração

- `/ip address add address=172.16.1.61/24 interface=ether1`
  Associa um endereço IP a uma porta do router.
- `/ip route add dst-address=<packetIP> gateway=<gatewayIP>`
  Sempre que o router receber um pacote com o destino <packetIP>, envia em vez para o endereço <gatewayIP>.
- `traceroute -n <IP>`
  Mostra todos os nós em que o pacote tem de passar para chegar a <IP>.
- `/ip firewall nat disable 0`
  Desliga a Network Address Translation (NAT).

Análise dos registos

- Após a configuração do router e de todas as rotas, o tuxY3 consegue aceder às interfaces tuxY2eth1, tuxY4eth2, tuxY4eth1 e RCeth2;
- Ao realizar um ping ao tuxY3 a partir do tuxY2, os endereços MAC comprovam que a rota percorrida pelo pacote é tuxY2 → Routereth2 → tuxY4eth2 → tuxY3.

Isto também é comprovado pelo “traceroute”:

```
Unset
root@tux62:~# traceroute -n 172.16.60.1
traceroute to 172.16.60.1 (172.16.60.1), 30 hops max, 60 byte packets
1  172.16.61.254  0.214 ms  0.200 ms  0.201 ms
2  172.16.61.253  0.345 ms  0.336 ms  0.331 ms
3  172.16.60.1  2.449 ms  2.455 ms  2.403 ms
```

- Ao alterar manualmente a rota para usar o tuxY4 em vez do router, o “traceroute” corresponderá à mesma informação dada pelos pacotes ICMP Redirect enviados pelo router: a rota mais eficiente:

```
Unset
root@tux62:~# traceroute -n 172.16.60.1
traceroute to 172.16.60.1 (172.16.60.1), 30 hops max, 60 byte packets
1  172.16.61.253  0.186 ms  0.158 ms  0.149 ms
2  172.16.60.1  0.348 ms  0.338 ms  0.319 ms
```

- O tuxY3 consegue aceder ao servidor FTP com sucesso pela rota tuxY3 → tuxY4eth1 → Routereth2 → FTPServer caso o NAT esteja ativo. Caso contrário, o router não tem como saber para qual endereço IP deve ser enviado a resposta do ICMP.

3.5. DNS

Objetivos da experiência

- Traduzir endereços alfanuméricos para endereços IP usando um servidor DNS;
- Configurar computadores para comunicarem com um servidor DNS.

Análise dos registos

Ao realizar um ping para um domínio em vez de um endereço IP, o tuxY3 envia um “DNS Request” para o servidor DNS (neste caso, 10.227.20.3) a perguntar pelo endereço IP, isto é, a resolução do domínio. A seguir, o servidor DNS envia um pacote “DNS Response”, que inclui o endereço IP em questão. Isto tudo acontece antes de trocarem os pacotes ICMP.

3.6. Conexões TCP

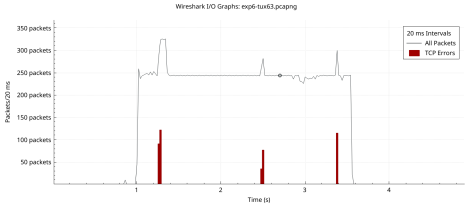
Objetivos da experiência

- Estabelecer a conexão com um servidor FTP e baixar um ficheiro usando a aplicação desenvolvida anteriormente;
- Compreender os pacotes FTP e as conexões TCP por detrás de um pedido a um servidor FTP.

Análise dos registos

Quando o tuxT3 pede um ficheiro ao servidor FTP usando o cliente desenvolvido:

- A primeira conexão TCP é estabelecida usando um "three-way handshake";
- A conexão começa a receber dados, em que usa uma variação do "Go-Back-N" com "sliding window" para o mecanismo ARQ, isto é, só pode enviar até N mensagens sem receber a confirmação da mensagem inicial;
- Para maximizar a capacidade, o mecanismo de controlo de congestionamento do TCP aumenta o tamanho da janela a cada mensagem até começar a perder pacotes, em que reduz em metade o tamanho, tal como se pode observar neste gráfico, em que o número de pacotes diminui na presença de erros:



- Numa perspetiva mais alto nível, a aplicação comunica com o servidor FTP, enviando os comandos USER, PASS, PASV, RETR e QUIT;
- As conexões TCP são terminadas depois do ficheiro ser transferido e a mensagem de saída enviada;

Veja as respostas às questões desta experiência para mais detalhes sobre como estes processos funcionam.

4. Conclusões

Todos os objetivos que eram pretendidos alcançar com o desenvolvimento deste projeto foram alcançados, desde a compreensão do funcionamento de distribuição de pacotes do protocolo IP (*internet protocol*), bem como das ferramentas que permitem a interação com o mesmo.

Apêndices

A. Comandos de configuração

A.1. Configurar uma rede IP

```
Unset
root@tux64:~# ifconfig eth1 up
root@tux64:~# ifconfig eth1 172.16.60.254/24
root@tux63:~# ifconfig eth1 up
root@tux63:~# ifconfig eth1 172.16.60.1/24
```

A.2. Implementação de duas “bridges” num “switch”

```
Unset
root@tux62:~# ifconfig eth1 up
root@tux62:~# ifconfig eth1 172.16.61.1/24

[admin@MikroTik] > /interface bridge add name=bridge60
[admin@MikroTik] > /interface bridge add name=bridge61

[admin@MikroTik] > /interface bridge port remove
numbers: 1 2 3
[admin@MikroTik] > /interface bridge port add bridge=bridge61
interface=ether2
[admin@MikroTik] > /interface bridge port add bridge=bridge60
interface=ether3
[admin@MikroTik] > /interface bridge port add bridge=bridge60
interface=ether4
```

A.3. Configurar um router em Linux

```
Unset
root@tux64:~# ifconfig eth2 up
root@tux64:~# ifconfig eth2 172.16.61.253/24

[admin@MikroTik] > /interface bridge port remove
```

```
numbers: 10
[admin@MikroTik] > /interface bridge port add bridge=bridge61
interface=ether14

root@tux64:~# sysctl net.ipv4.ip_forward=1
root@tux64:~# sysctl net.ipv4.icmp_echo_ignore_broadcasts=0

root@tux63:-# route add -net 172.16.61.0/24 gw 172.16.60.254
root@tux62:-# route add -net 172.16.60.0/24 gw 172.16.61.253

# Delete all ARP tables (demonstration only)
root@tux62:~# ip neigh flush dev eth1
root@tux63:~# ip neigh flush dev eth1
root@tux64:~# ip neigh flush dev eth1
root@tux64:~# ip neigh flush dev eth2
```

A.4. Configurar um router comercial e implementar NAT

```
Unset
# Switch
[admin@MikroTik] > /interface bridge port remove
numbers: 7
[admin@MikroTik]    >    /interface    bridge    port    add    bridge=bridge61
interface=ether11

# Router
[admin@MikroTik] > /ip address add address=172.16.1.61/24 interface=ether1
[admin@MikroTik] > /ip address add address=172.16.61.254/24 interface=ether2

root@tux62:~# route add -net 172.16.60.0/24 gw 172.16.61.253       #  Already
exists :/
root@tux62:~# route add -net 172.16.1.0/24 gw 172.16.61.254
root@tux63:~# route add -net 172.16.61.0/24 gw 172.16.60.254
root@tux63:~# route add -net 172.16.1.0/24 gw 172.16.60.254
root@tux64:~# route add -net 172.16.1.0/24 gw 172.16.61.254
[admin@MikroTik]    >    /ip    route    add    dst-address=172.16.60.0/24
gateway=172.16.61.253
```

```
root@tux62:~# sysctl net.ipv4.conf.eth1.accept_redirects=0
root@tux62:~# sysctl net.ipv4.conf.all.accept_redirects=0
root@tux62:~# route del -net 172.16.60.0/24 gw 172.16.61.253
root@tux62:~# route add -net 172.16.60.0/24 gw 172.16.61.254

# Demonstration purposes only (no configuration from here)
root@tux62:~# traceroute -n 172.16.60.1
root@tux62:~# route del -net 172.16.60.0/24 gw 172.16.61.254
root@tux62:~# route add -net 172.16.60.0/24 gw 172.16.61.253
root@tux62:~# traceroute -n 172.16.60.1
root@tux62:~# sysctl net.ipv4.conf.eth1.accept_redirects=1
root@tux62:~# sysctl net.ipv4.conf.all.accept_redirects=1
root@tux62:~# traceroute 172.16.60.1
[admin@MikroTik] > /ip firewall nat disable 0
```

A.5. DNS

```
Unset
root@tux62:~# echo "nameserver 10.227.20.3" >> /etc/resolv.conf
root@tux63:~# echo "nameserver 10.227.20.3" >> /etc/resolv.conf
root@tux64:~# echo "nameserver 10.227.20.3" >> /etc/resolv.conf
```

A.6. Conexões TCP

```
Unset
# Download the source code from Moodle (for instructors) or from GitLab (for
students)
root@tux63:~# git clone gitlab.up.pt/up202208393/rcom
root@tux63:~# cd rcom/Lab2
root@tux63:~# make
root@tux63:~# ./bin/download ftp://rcom:rcom@ftp.netlab.fe.up.pt/pipe.txt
```

B. Registos capturados

Todos os registos capturados a partir do Wireshark durante as aulas laboratoriais estão presentes nesta pasta partilhada. Em baixo estão disponíveis imagens filtradas dos registos para os objetivos de cada experiência:

B.1. Configurar uma rede IP

exp1.pcapng

	Time	Source	Destination	Protocol	Lengtl	Info
20	24.060599264	KYE_25:40:66	Broadcast	ARP	42	Who has 172.16.60.254? Tell 172.16.60.1
21	24.060694667	3Com_a1:35:69	KYE_25:40:66	ARP	60	172.16.60.254 is at 00:01:02:a1:35:69
22	24.060703118	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) request id=0x3e54, seq=1/256, ttl=64 (reply in 23)
23	24.060804528	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x3e54, seq=1/256, ttl=64 (request in 22)
24	25.091032726	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) request id=0x3e54, seq=2/512, ttl=64 (reply in 25)
25	25.091146427	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x3e54, seq=2/512, ttl=64 (request in 24)

B.2. Implementação de duas “bridges” num “switch

exp2-tux63-ping.pcapng

No.	Time	Source	Destination	Protocol	Lengtl	Info
10	16.348514519	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) request id=0x7a32, seq=1/256, ttl=64 (reply in 11)
11	16.348678715	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x7a32, seq=1/256, ttl=64 (request in 10)
12	17.358055028	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) request id=0x7a32, seq=2/512, ttl=64 (reply in 13)
13	17.358204278	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x7a32, seq=2/512, ttl=64 (request in 12)

No.	Time	Source	Destination	Protocol	Lengtl	Info
48	34.030046845	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x7a3c, seq=3/768, ttl=64 (no response found!)
49	35.054046548	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x7a3c, seq=4/1024, ttl=64 (no response found!)

exp2-tux62-broadcast3.pcapng

No.	Time	Source	Destination	Protocol	Lengtl	Info
1	0.000000000	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e4 Cost = 0 Port = 0x0001
2	0.956158582	0.0.0.0	255.255.255.255	MNDP	182	5678 → 5678 Len=140
3	0.956221790	Routerboardc_1c:8b:…	CDP/VTP/DTP/PAgP/UD…	CDP	116	Device ID: MikroTik Port ID: bridge61/ether2
4	0.956322224	Routerboardc_1c:8b:…	LLDP_Multicast	LLDP	133	MA/c4:ad:34:1c:8b:e3 IN/bridge61/ether2 120 SysN=MikroTik SysD=MikroTik RouterOS 7.1.1 (stable) Dec/21/2021 11:53:05 CRS326-24G-2S+
5	2.001871834	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e4 Cost = 0 Port = 0x0001

exp2-tux63-broadcast3.pcapng

No.	Time	Source	Destination	Protocol	Lengtl	Info
26	48.749734882	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x7bf4, seq=1/256, ttl=64 (no response found!)
27	48.749990989	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x7bf4, seq=1/256, ttl=64
28	49.754990084	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x7bf4, seq=2/512, ttl=64 (no response found!)
29	49.755203868	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x7bf4, seq=2/512, ttl=64

exp2-tux64-broadcast3.pcapng

No.	Time	Source	Destination	Protocol	Lengtl	Info
23	38.742759843	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x7bf4, seq=2/512, ttl=64 (no response found!)
24	38.742800491	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x7bf4, seq=2/512, ttl=64
25	39.766770270	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x7bf4, seq=3/768, ttl=64 (no response found!)
26	39.766803026	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x7bf4, seq=3/768, ttl=64

exp2-tux62-broadcast2.pcapng

No.	Time	Source	Destination	Protocol	Lengtl	Info
9	6.542226150	Netronix_b5:8c:8e	Broadcast	ARP	42	Who has 172.16.61.253? Tell 172.16.61.1
10	7.566221278	Netronix_b5:8c:8e	Broadcast	ARP	42	Who has 172.16.61.253? Tell 172.16.61.1
11	8.008874085	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e4 Cost = 0 Port = 0x0001
12	8.590378998	Netronix_b5:8c:8e	Broadcast	ARP	42	Who has 172.16.61.253? Tell 172.16.61.1
13	9.614232554	Netronix_b5:8c:8e	Broadcast	ARP	42	Who has 172.16.61.253? Tell 172.16.61.1

exp2-tux63-broadcast2.pcapng

No.	Time	Source	Destination	Protocol	Lengtl	Info
1	0.000000000	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e5 Cost = 0 Port = 0x0001
2	2.002347034	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e5 Cost = 0 Port = 0x0001
3	4.004681985	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e5 Cost = 0 Port = 0x0001
4	6.006543414	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e5 Cost = 0 Port = 0x0001
5	8.008888353	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e5 Cost = 0 Port = 0x0001

exp2-tux64-broadcast2.pcapng

No.	Time	Source	Destination	Protocol	Lengtl	Info
1	0.000000000	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e5 Cost = 0 Port = 0x0002
2	2.002346977	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e5 Cost = 0 Port = 0x0002
3	4.004694304	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e5 Cost = 0 Port = 0x0002
4	6.007033599	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e5 Cost = 0 Port = 0x0002
5	8.008871004	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e5 Cost = 0 Port = 0x0002

B.3. Configurar um router em Linux

exp3-tux63.pcapng

No.	Time	Source	Destination	Protocol	Lengtl	Info
11	14.300460052	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) request id=0x08e2, seq=2/512, ttl=64 (reply in 12)
12	14.300606719	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x08e2, seq=2/512, ttl=64 (request in 11)
13	15.324459995	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) request id=0x08e2, seq=3/768, ttl=64 (reply in 14)
14	15.324607988	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x08e2, seq=3/768, ttl=64 (request in 13)

No.	Time	Source	Destination	Protocol	Lengtl	Info
31	26.806778037	172.16.60.1	172.16.61.253	ICMP	98	Echo (ping) request id=0x08ed, seq=1/256, ttl=64 (reply in 32)
32	26.806955992	172.16.61.253	172.16.60.1	ICMP	98	Echo (ping) reply id=0x08ed, seq=1/256, ttl=64 (request in 31)
33	27.836457318	172.16.60.1	172.16.61.253	ICMP	98	Echo (ping) request id=0x08ed, seq=2/512, ttl=64 (reply in 34)
34	27.836612225	172.16.61.253	172.16.60.1	ICMP	98	Echo (ping) reply id=0x08ed, seq=2/512, ttl=64 (request in 33)

No.	Time	Source	Destination	Protocol	Lengtl	Info
48	36.711011066	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x08f4, seq=1/256, ttl=64 (reply in 49)
49	36.711333104	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x08f4, seq=1/256, ttl=63 (request in 48)
50	37.724453347	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x08f4, seq=2/512, ttl=64 (reply in 51)
51	37.724709593	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x08f4, seq=2/512, ttl=63 (request in 50)

exp3-tux64-eth1.pcapng

No.	Time	Source	Destination	Protocol	Lengtl	Info
27	44.648961395	KYE_25:40:66	Broadcast	ARP	60	Who has 172.16.60.254? Tell 172.16.60.1
28	44.648985421	3Com_a1:35:69	KYE_25:40:66	ARP	42	172.16.60.254 is at 00:01:02:a1:35:69
29	44.649094935	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0ce2, seq=1/256, ttl=64 (reply in 30)
30	44.649377519	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0ce2, seq=1/256, ttl=63 (request in 29)
31	45.673251446	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0ce2, seq=2/512, ttl=64 (reply in 32)
32	45.673402376	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0ce2, seq=2/512, ttl=63 (request in 31)
33	46.050755673	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e5 Cost = 0 Port = 0x0002
34	46.697247598	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0ce2, seq=3/768, ttl=64 (reply in 35)
35	46.697394687	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0ce2, seq=3/768, ttl=63 (request in 34)
36	47.721268404	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0ce2, seq=4/1024, ttl=64 (reply in 37)
37	47.721418357	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0ce2, seq=4/1024, ttl=63 (request in 36)
38	48.053120404	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e5 Cost = 0 Port = 0x0002
39	48.745246746	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0ce2, seq=5/1280, ttl=64 (reply in 40)
40	48.745397398	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0ce2, seq=5/1280, ttl=63 (request in 39)
41	49.723395530	3Com_a1:35:69	KYE_25:40:66	ARP	42	Who has 172.16.60.1? Tell 172.16.60.254
42	49.723516358	KYE_25:40:66	3Com_a1:35:69	ARP	60	172.16.60.1 is at 00:c0:df:25:40:66
43	49.769232701	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0ce2, seq=6/1536, ttl=64 (reply in 44)
44	49.769398927	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0ce2, seq=6/1536, ttl=63 (request in 43)

exp3-tux64-eth2.pcapng

No.	Time	Source	Destination	Protocol	Lengtl	Info
20	31.546481121	fe80::2e0:7dff:feb5…	ff02::2	ICMPv6	70	Router Solicitation from 00:e0:7d:b5:8c:8e
21	32.035428713	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e4 Cost = 0 Port = 0x0002
22	34.037806994	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e4 Cost = 0 Port = 0x0002
23	36.040184925	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e4 Cost = 0 Port = 0x0002
24	38.042564603	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e4 Cost = 0 Port = 0x0002
25	38.642719560	KYE_04:20:8c	Broadcast	ARP	42	Who has 172.16.61.1? Tell 172.16.61.253
26	38.642854916	Netronix_b5:8c:8e	KYE_04:20:8c	ARP	60	172.16.61.1 is at 00:e0:7d:b5:8c:8e
27	38.642861202	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0ce2, seq=1/256, ttl=63 (reply in 28)
28	38.642981611	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0ce2, seq=1/256, ttl=64 (request in 27)
29	39.666884662	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0ce2, seq=2/512, ttl=63 (reply in 30)
30	39.667003954	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0ce2, seq=2/512, ttl=64 (request in 29)
31	40.044324075	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e4 Cost = 0 Port = 0x0002
32	40.690885843	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0ce2, seq=3/768, ttl=63 (reply in 33)
33	40.690995985	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0ce2, seq=3/768, ttl=64 (request in 32)
34	41.714906510	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0ce2, seq=4/1024, ttl=63 (reply in 35)
35	41.715018957	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0ce2, seq=4/1024, ttl=64 (request in 34)
36	42.046688318	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:e4 Cost = 0 Port = 0x0002
37	42.738885690	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0ce2, seq=5/1280, ttl=63 (reply in 38)
38	42.738997928	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0ce2, seq=5/1280, ttl=64 (request in 37)
39	43.762859562	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0ce2, seq=6/1536, ttl=63 (reply in 40)
40	43.763002600	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0ce2, seq=6/1536, ttl=64 (request in 39)
41	43.834386056	Netronix_b5:8c:8e	KYE_04:20:8c	ARP	60	Who has 172.16.61.253? Tell 172.16.61.1
42	43.834391433	KYE_04:20:8c	Netronix_b5:8c:8e	ARP	42	172.16.61.253 is at 00:c0:df:04:20:8c

B.4. Configurar um router comercial e implementar NAT

exp4-tux62.pcapng

No.	Time	Source	Destination	Protocol	Lengtl	Info
7	2.007743991	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) request id=0x227b, seq=2/512, ttl=64 (reply in 9)
8	2.007894222	172.16.61.254	172.16.61.1	ICMP	126	Redirect (Redirect for host)
9	2.008094042	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) reply id=0x227b, seq=2/512, ttl=63 (request in 7)
10	3.031749646	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) request id=0x227b, seq=3/768, ttl=64 (reply in 12)
11	3.031904556	172.16.61.254	172.16.61.1	ICMP	126	Redirect (Redirect for host)
12	3.032111569	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) reply id=0x227b, seq=3/768, ttl=63 (request in 10)
13	4.005111163	Routerboardc_1c:8b:…	Spanning-tree-(for-…	STP	60	RST. Root = 32768/0/74:4d:28:eb:23:fc Cost = 10 Port = 0x0001
14	4.055746993	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) request id=0x227b, seq=4/1024, ttl=64 (reply in 16)
15	4.055904348	172.16.61.254	172.16.61.1	ICMP	126	Redirect (Redirect for host)
16	4.056085589	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) reply id=0x227b, seq=4/1024, ttl=63 (request in 14)
17	5.079751400	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) request id=0x227b, seq=5/1280, ttl=64 (reply in 19)
18	5.079912596	172.16.61.254	172.16.61.1	ICMP	126	Redirect (Redirect for host)
19	5.080096421	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) reply id=0x227b, seq=5/1280, ttl=63 (request in 17)

exp4-tux63.pcapng

No.	Time	Source	Destination	Protocol	Lengtl	Info
2	0.969441206	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x2199, seq=1/256, ttl=64 (reply in 3)
3	0.969760170	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x2199, seq=1/256, ttl=63 (request in 2)
4	1.971376514	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x2199, seq=2/512, ttl=64 (reply in 5)
5	1.971661605	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x2199, seq=2/512, ttl=63 (request in 4)
17	6.163347378	KYE_25:40:66	3Com_a1:35:69	ARP	42	Who has 172.16.60.254? Tell 172.16.60.1
18	6.163463803	3Com_a1:35:69	KYE_25:40:66	ARP	60	172.16.60.254 is at 00:01:02:a1:35:69
19	6.616871559	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) request id=0x219d, seq=1/256, ttl=64 (reply in 20)
20	6.617015781	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x219d, seq=1/256, ttl=64 (request in 19)
21	7.635421632	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) request id=0x219d, seq=2/512, ttl=64 (reply in 22)
22	7.635562572	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x219d, seq=2/512, ttl=64 (request in 21)
34	12.360672703	172.16.60.1	172.16.61.253	ICMP	98	Echo (ping) request id=0x21a1, seq=1/256, ttl=64 (reply in 35)
35	12.360810709	172.16.61.253	172.16.60.1	ICMP	98	Echo (ping) reply id=0x21a1, seq=1/256, ttl=64 (request in 34)
36	13.363376123	172.16.60.1	172.16.61.253	ICMP	98	Echo (ping) request id=0x21a1, seq=2/512, ttl=64 (reply in 37)
37	13.363510358	172.16.61.253	172.16.60.1	ICMP	98	Echo (ping) reply id=0x21a1, seq=2/512, ttl=64 (request in 36)
61	22.291378305	172.16.60.1	172.16.61.254	ICMP	98	Echo (ping) request id=0x21a5, seq=5/1280, ttl=64 (reply in 62)
62	22.291668495	172.16.61.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x21a5, seq=5/1280, ttl=63 (request in 61)
63	23.315375561	172.16.60.1	172.16.61.254	ICMP	98	Echo (ping) request id=0x21a5, seq=6/1536, ttl=64 (reply in 64)
64	23.315639840	172.16.61.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x21a5, seq=6/1536, ttl=63 (request in 63)
68	24.979374813	172.16.60.1	172.16.1.61	ICMP	98	Echo (ping) request id=0x21a9, seq=2/512, ttl=64 (reply in 69)
69	24.979689097	172.16.1.61	172.16.60.1	ICMP	98	Echo (ping) reply id=0x21a9, seq=2/512, ttl=63 (request in 68)
70	26.003386526	172.16.60.1	172.16.1.61	ICMP	98	Echo (ping) request id=0x21a9, seq=3/768, ttl=64 (reply in 71)
71	26.003664074	172.16.1.61	172.16.60.1	ICMP	98	Echo (ping) reply id=0x21a9, seq=3/768, ttl=63 (request in 70)

B.5. DNS

exp5-tux63.pcapng

No.	Time	Source	Destination	Protocol	Lengtl	Info
44	3.212746032	10.227.20.63	10.227.20.3	DNS	70	Standard query 0x7936 A google.com
45	3.212757556	10.227.20.63	10.227.20.3	DNS	70	Standard query 0xe13f AAAA google.com
46	3.213275778	10.227.20.3	10.227.20.63	DNS	86	Standard query response 0x7936 A google.com A 142.250.200.142
47	3.213295892	10.227.20.3	10.227.20.63	DNS	98	Standard query response 0xe13f AAAA google.com AAAA 2a00:1450:4003:80f::200e
48	3.213710400	10.227.20.63	142.250.200.142	ICMP	98	Echo (ping) request id=0x10df, seq=1/256, ttl=64 (reply in 51)
51	3.229060383	142.250.200.142	10.227.20.63	ICMP	98	Echo (ping) reply id=0x10df, seq=1/256, ttl=112 (request in 48)
52	3.229174015	10.227.20.63	10.227.20.3	DNS	88	Standard query 0xdd09 PTR 142.200.250.142.in-addr.arpa
53	3.229466091	10.227.20.3	10.227.20.63	DNS	127	Standard query response 0xdd09 PTR 142.200.250.142.in-addr.arpa PTR mad41s14-in-f14.1e100.net
56	3.990395993	10.227.20.158	255.255.255.255	DNS	132	Standard query 0x1ea6 A pnp-connect-production-1950043792.us-east-1.elb.amazonaws.com OPT
57	4.215555087	10.227.20.63	142.250.200.142	ICMP	98	Echo (ping) request id=0x10df, seq=2/512, ttl=64 (reply in 58)
58	4.230873991	142.250.200.142	10.227.20.63	ICMP	98	Echo (ping) reply id=0x10df, seq=2/512, ttl=112 (request in 57)

B.6. Conexões TCP

exp6-tux63.pcapng

No.	Time	Source	Destination	Protocol	Lengtl	Info
5	0.863142928	172.16.66.1	172.16.1.10	FTP	77	Request: USER rcom
7	0.868380892	172.16.1.10	172.16.66.1	FTP	116	Response: 220 ProFTPD Server (Debian) [::ffff:172.16.1.10]
9	0.869215846	172.16.1.10	172.16.66.1	FTP	98	Response: 331 Password required for rcom
11	0.869303217	172.16.66.1	172.16.1.10	FTP	77	Request: PASS rcom
13	1.014923832	172.16.1.10	172.16.66.1	FTP	112	Response: 230-Welcome, archive user rcom@172.16.1.61 !
14	1.014950442	172.16.1.10	172.16.66.1	FTP	115	Response:
15	1.015067216	172.16.1.10	172.16.66.1	FTP	233	Response:
16	1.015068683	172.16.66.1	172.16.1.10	FTP	72	Request: PASV
18	1.015900564	172.16.1.10	172.16.66.1	FTP	115	Response: 227 Entering Passive Mode (172,16,1,10,165,93).
23	1.016388406	172.16.66.1	172.16.1.10	FTP	87	Request: RETR files/crab.mp4
24	1.017437423	172.16.1.10	172.16.66.1	FTP	142	Response: 150 Opening ASCII mode data connection for files/crab.mp4 (29803194 bytes)
31482	14.304308686	172.16.1.10	172.16.66.1	FTP	89	Response: 226 Transfer complete
31484	14.304428743	172.16.66.1	172.16.1.10	FTP	72	Request: QUIT
31485	14.304889696	172.16.1.10	172.16.66.1	FTP	80	Response: 221 Goodbye.

C. Resolução das questões propostas

C.1. Configurar uma rede IP

- **O que são os pacotes ARP e para que são utilizados?**
  Os pacotes ARP são usados no "Address Resolution Protocol" para descobrir o endereço MAC de uma conexão a partir do seu endereço IP.

- **Quais são os endereços MAC e IP dos pacotes ARP e por quê?**

Pacote	Endereço da fonte		Endereço de destino	
	MAC	IP	MAC	IP
ARP Request (tux63 → tux64)	00:c0:df: 25:40:66	172.16.60.1	FF:FF:FF: FF:FF:FF	172.16.60.254
ARP Reply (tux64 → tux63)	00:21:5a: c5:61:bb	172.16.60.254	00:c0:df: 25:40:66	172.16.60.1

  O endereço MAC de destino de um ARP Request é FF:FF:FF:FF:FF:FF (broadcast), já que o pedido é enviado a todos os dispositivos da rede local.

- **Que pacotes o comando `ping` gera?**
  "ICMP Echo Request" e "ICMP Echo Reply".

- **Quais são os endereços MAC e IP dos pacotes gerados pelo `ping`?**
  Os endereços IP e MAC de fonte de um pacote "Request" correspondem aos do tux63 (quem executou o ping) e de destino correspondem aos do tux64, e vice-versa para o caso dos pacotes "Reply".

- **Como determinar se uma trama Ethernet a receber é ARP, IP, ICMP?**
  Ao analisar o parâmetro de tipo numa trama: um EtherType de 0x0806 corresponde a um pacote ARP e 0x0800 corresponde a um pacote IP. Caso o pacote seja do tipo IP, o parâmetro de protocolo do datagrama dirá 1 se corresponder a uma mensagem ICMP.

- **Como determinar o comprimento de uma trama a receber?**
  Caso o parâmetro de tipo/tamanho da trama seja menor que 1500, basta sumar 18 para obter o comprimento da trama. Caso contrário, é necessário observar o

parâmetro do protocolo em questão. Por exemplo, um pacote IP tem um parâmetro de 16 bits, chamado tamanho total onde esse valor está armazenado.

- **O que é a "interface" de "loopback" e por que é importante?**
  Uma "loopback interface" é uma interface de rede virtual que permite um computador aceder serviços e aplicações a correr localmente sem a necessidade de uma rede física.

C.2. Implementação de duas "bridges" num "switch"

- **Como configurar o bridgeY0?**
  É necessário criar a "bridge", remover todas as portas candidatas das "bridges" antigas e adicioná-las uma a uma à nova "bridge". Veja os comandos desta experiência para configurar um switch "MikroTik" em concreto.

- **Quantos domínios de "broadcast" existem? Como pode concluir isso a partir dos registos?**
  Existem 2 domínios de "broadcast" (172.16.60.255 e 172.16.61.255), um por cada "bridge". Os logs do Wireshark demonstram isso mesmo, já que o "broadcast" do tux63 é visível para o tux64, mas o "broadcast" do tux62 não é acessível pelos computadores da bridge60.

C.3. Configurar um router em Linux

- **Que rotas existem nos "tuxes"? Qual o seu significado?**

Rota*	Destination	Gateway	Genmask	Flags	Iface
Comum	0.0.0.0	10.227.20.254	0.0.0.0	UG	eth0
	Se todas as outras rotas falharem, enviar pacote pelo RLab.				
	10.227.20.0	0.0.0.0	255.255.255.0	U	eth0
	Pacotes para o router devem ser enviados diretamente ao RLab.				
tux62	172.16.60.0	172.16.61.253	255.255.255.0	UG	eth1
	Pacotes para a bridge60 devem passar primeiro pelo tux64.				
	172.16.61.0	0.0.0.0	255.255.255.0	U	eth1
	Pacotes devem ser enviados diretamente ao destinatário.				

tux63	172.16.61.0	172.16.60.254	255.255.255.0	UG	eth1
	Pacotes para a bridge61 devem passar primeiro pelo tux64.				
	172.16.60.0	0.0.0.0	255.255.255.0	U	eth1
	Pacotes devem ser enviados diretamente ao destinatário.				
tux64	172.16.60.0	0.0.0.0	255.255.255.0	U	eth1
	Pacotes devem ser enviados diretamente para a bridge60.				
	172.16.61.0	0.0.0.0	255.255.255.0	U	eth2
	Pacotes devem ser enviados diretamente para a bridge61.				

*Metric = 0; Ref = 0; Use = 0

- **Que informações contém uma entrada da tabela de encaminhamento?**
  Cada entrada indica para um determinado conjunto de destinos, para que endereço IP e interface de rede deve ser redirecionado.

- **Que mensagens ARP e endereços MAC associados são observados e por quê?**
  Um ping do tux63 para o tux62 faz a rota tux63 eth1 → tux64 eth1 → tux64 eth2 → tux62 eth1. Como o tux64 conhece os seus endereços MACs internos, só serão observados "ARP Requests" e "ARP Replies" antes dos pacotes "ICMP Echo" para os IPs cujos MACs ainda são desconhecidos:

Tuxs	Pacote	MAC da fonte	MAC de destino
tux63 ↔ (1) tux64	ARP Request	00:c0:df:25:40:66	FF:FF:FF:FF:FF:FF
	ARP Reply	00:01:02:a1:35:69 (eth1)	00:c0:df:25:40:66
tux64 ↔ tux62	ARP Request	00:c0:df:04:20:8c	FF:FF:FF:FF:FF:FF
	ARP Reply	00:e0:7d:b5:8c:8e	00:c0:df:04:20:8c

- **Que pacotes ICMP são observados e por quê?**
  São observados os pacotes "ICMP Echo Request" (enviados pelo tux63) e "ICMP Echo Reply" (recebidos pelo tux63) graças à execução do ping pelo lado do tux63.

- **Quais são os endereços IP e MAC associados aos pacotes ICMP e por**

quê?

Os endereços IP e MAC de fonte de um pacote "Request" correspondem aos do tux63 (quem executou o ping) e de destino correspondem às outras interfaces (tux64 eth1, tux64 eth2 ou tux63 eth1), e vice-versa para o caso dos pacotes "Reply".

C.4. Configurar um router comercial e implementar NAT

- **Como configurar uma rota estática num router comercial?**

  No caso de um router MikroTik, execute `/ip route add dst-address=<packetIP> gateway=<gatewayIP>`.

- **Quais os caminhos seguidos pelos pacotes, com e sem redirecionamento ICMP habilitado, nas experiências realizadas e por quê?**

  Sem redirecionamento ICMP, a rota é tuxY2 → Routereth2 → tuxY4eth2 → tuxY3, tal como configurado na experiência, no entanto, com ICMP habilitado a rota passa a ser a mais eficiente, isto é, tuxY2 → tuxY4eth2 → tuxY3. Isto deve-se aos pacotes ICMP Redirect enviados pelo router para o tuxY2 que indicam que existe um caminho mais curto que o computador devia seguir.

- **Como configurar o NAT num router comercial?**

  No caso de um router MikroTik, execute `/ip firewall nat disable 0` ou `/ip firewall nat enable 0` para desativar ou ativar o NAT padrão, respetivamente.

- **O que faz o NAT?**

  Uma "Network Address Table" dentro de um router permite mapear vários endereços IP da rede privada para um único endereço público, o que permite ofuscar a configuração da rede privada e preservar o número limitado de IPs da Internet.

- **O que acontece quando o tuxY3 faz ping ao servidor FTP com o NAT desativado? Por quê?**

  O tuxY3 consegue enviar com sucesso o pacote ICMP Request para o servidor FTP, no entanto, o servidor FTP envia um ICMP Reply para o router e o mesmo

não consegue perceber quem é que devia ser o destinatário deste pacote, pelo que o tuxY3 não recebe a resposta.

C.5. DNS

- **Como configurar o serviço DNS num "host"?**
  Para configurar o DNS num computador Linux, modifique o ficheiro /etc/resolv.conf para incluir o IP do servidor DNS. Por exemplo, nesta experiência execute "echo "nameserver 10.227.20.3" >> /etc/resolv.conf " num terminal Bash.

- **Que pacotes são trocados pelo DNS e que informação é transportada?**
  Um computador envia um "DNS Query" para o servidor com o domínio a ser resolvido e, a seguir, o servidor envia um "DNS Response" com o endereço IP associado ao domínio.

C.6. Conexões TCP

- **Quantas ligações TCP são abertas pela sua aplicação FTP?**
  O cliente FTP abre duas conexões TCP: uma para comunicar comandos para o servidor e outra para baixar os conteúdos do ficheiro já solicitado.

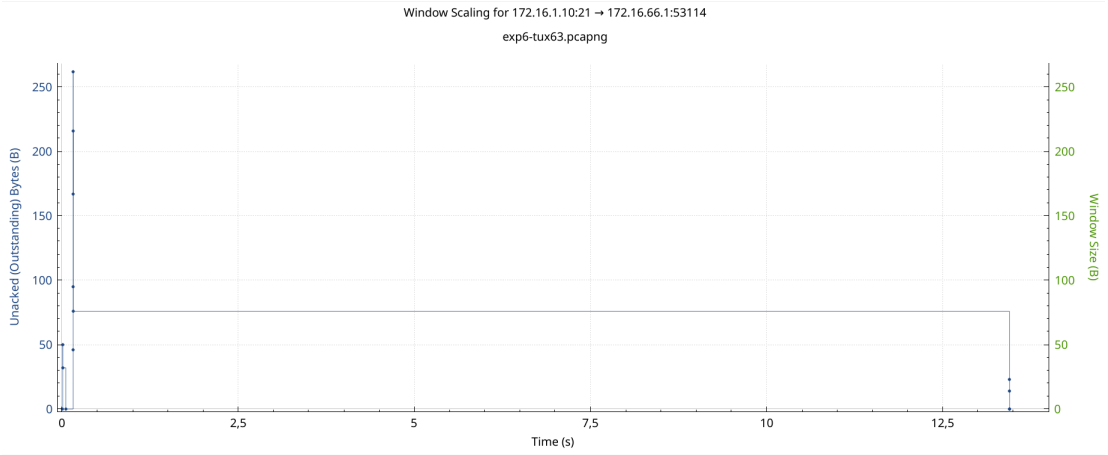
- **Em que conexão é transportada a informação de controlo do FTP?**
  Na primeira conexão TPC aberta pelo cliente FTP é transportada os comandos para comunicação com o servidor.

- **Quais as fases de uma conexão TCP?**
  - "Three-Way Handshake" [SYN, SYN/ACK, ACK];
  - A transferência de dados em si;
  - O fecho da conexão [FIN/ACK, ACK].

- **Como funciona o mecanismo ARQ TCP? Quais são os campos TCP relevantes? Que informação relevante pode ser observada nos logs?**
  O mecanismo de controlo de erros (ARQ) no TCP é uma variação do "Go-Back-N" com "sliding window", em que:
  - Um dos parâmetros estabelecidos em cada pacote é o tamanho da janela, isto é, a capacidade de comunicação;

- Cada pacote ACK contém um número de sequência;
- Um reconhecimento (ACK) aplica-se a todos os bytes com um número de sequência menor que o atual.

Assim, nos registos é relevante observar, por exemplo, a variação do tamanho da janela ao longo da transferência, que se manteve constante durante a maioria da transferência, já que houve poucos pacotes TCP perdidos.



- **Como funciona o mecanismo de controlo de congestionamento do TCP? Quais são os relevantes campos? Como evoluiu o rendimento ("through put") da ligação de dados ao longo do tempo? Está conforme o mecanismo de controlo de congestionamento TCP?**

  O mecanismo de controlo de congestionamento do TCP permite rapidamente se adaptar à variável capacidade da rede. Para tal, o início da conexão começa com o tamanho da janela em 1 MSS ("maximum segment size") e vai dobrar o seu tamanho até perder um pacote (detetado por timeout), em que a janela é reduzida em metade. Isto é o "Slow Start". A partir daí, entra-se no modo "Congestion Avoidance", em que aumenta por um MSS a cada RTT ("round trip time") até perder um pacote (detetado por receção de 3 ACKs duplicados) em que a janela é reduzida em metade.

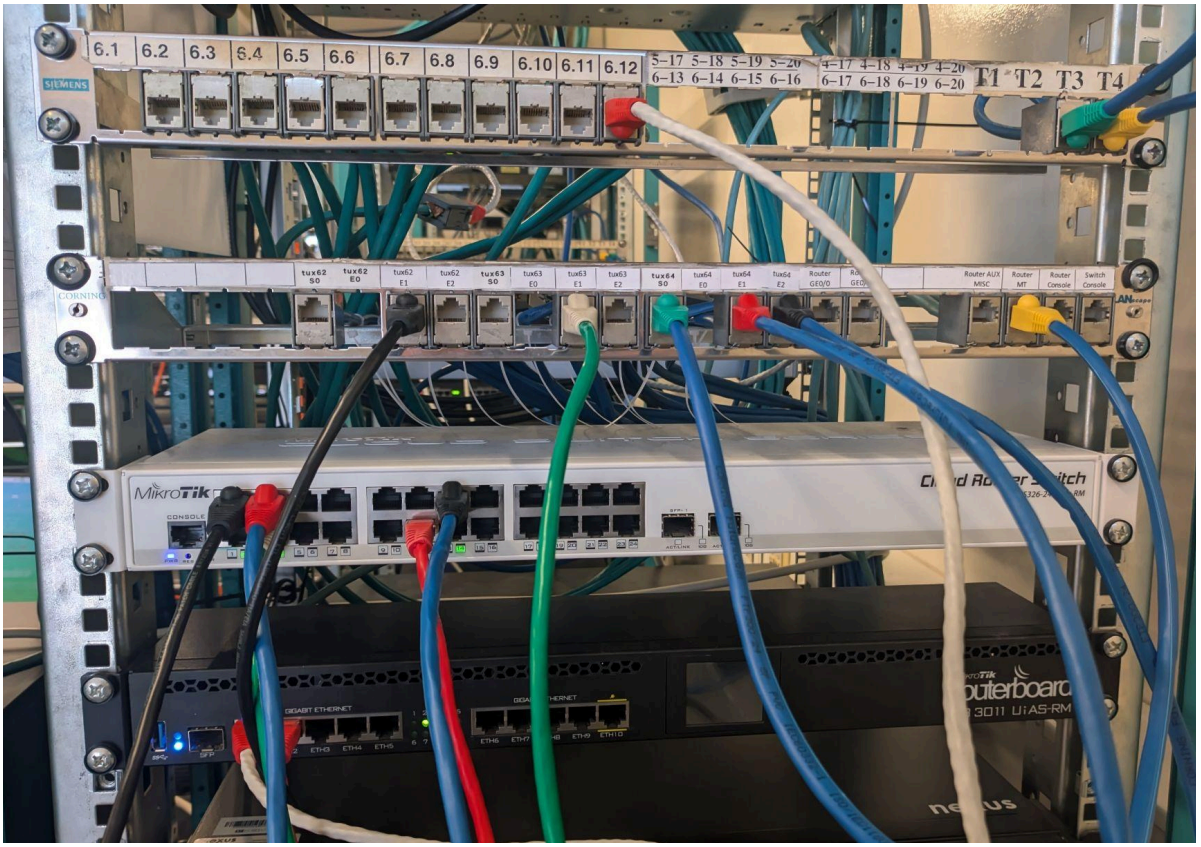
- **O rendimento ("through put") das ligações de dados TCP é perturbada pelo aparecimento de uma segunda ligação TCP? Como?**

  Sim, o "through put" será perturbado negativamente para as duas ligações, já que a largura de banda será dividida entre elas.

D. Guião para demonstração

Este apêndice relata o processo estipulado para configurar a rede segundos as seis experiências o mais rápido e sistemático possível. Este processo foi usado durante a demonstração na última aula prática.

D.1. Configuração das conexões



D.2. Limpar máquinas do laboratório

- ☐ `systemctl restart networking && reboot`
  (Executar em cada um dos 3 TUXs)
- ☐ Conectar a consola do switch ao tux64S0 (ou tux63S0, …)
- ☐ Abrir o GTKTerm e configurar a porta para baudrate de 115200
- ☐ Realizar login (se necessário):
  Utilizador: admin
  Palavra-passe: <branco> (apenas clique Enter)
- ☐ No switch, executar `/system reset-configuration` e confirmar com Y.

- [ ] Conectar a consola do router (RouterMT) ao tux64S0 (ou …)
- [ ] Realizar login (se necessário):

  Utilizador: admin

  Palavra-passe: <branco> (apenas clique Enter)
- [ ] No switch, executar `/system reset-configuration` e confirmar com Y.

D.3. Executar guiões de configuração

- tuxY2.sh

```
Unset
# Script for RCOM demonstration: tux62
# Authors:
# Guilherme Duarte Silva Matos (up202208755@up.pt)
# João Vítor da Costa Ferreira (up202208393@up.pt)

# IPs

bridge60=172.16.66.0/24
tux63eth1=172.16.66.1
tux64eth1=172.16.66.254

bridge61=172.16.67.0/24
tux62eth1=172.16.67.1
tux64eth2=172.16.67.253
routereth2=172.16.67.254

bridgeFTP=172.16.1.0/24
routereth1=172.16.1.61
ftpserver=172.16.1.10



# Reset configuration
systemctl restart networking

# Experience #1
# N/A

# Experience #2
```

```
ifconfig eth1 up
ifconfig eth1 ${tux62eth1}/24

# Experience #3
route add -net ${bridge60} gw ${tux64eth2}

# Experience #4
# route add -net $bridge60 gw $tux64eth2
route add -net ${bridgeFTP} gw ${routereth2}
sysctl net.ipv4.conf.eth1.accept_redirects=0
sysctl net.ipv4.conf.all.accept_redirects=0

# Experience #5
echo "nameserver 10.227.20.3" >> /etc/resolv.conf
```

- tuxY3.sh

```
Unset
# Script for RCOM demonstration: tux63
# Authors:
# Guilherme Duarte Silva Matos (up202208755@up.pt)
# João Vítor da Costa Ferreira (up202208393@up.pt)

# IPs

bridge60=172.16.66.0/24
tux63eth1=172.16.66.1
tux64eth1=172.16.66.254

bridge61=172.16.67.0/24
tux62eth1=172.16.67.1
tux64eth2=172.16.67.253
routereth2=172.16.67.254

bridgeFTP=172.16.1.0/24
routereth1=172.16.1.61
ftpserver=172.16.1.10
```

```
# Reset configuration
systemctl restart networking

# Experience #1
ifconfig eth1 up
ifconfig eth1 ${tux63eth1}/24

# Experience #2
# N/A

# Experience #3
route add -net ${bridge61} gw ${tux64eth1}

# Experience #4
route add -net ${bridge61} gw ${tux64eth1}
route add -net ${bridgeFTP} gw ${tux64eth1}

# Experience #5
echo "nameserver 10.227.20.3" >> /etc/resolv.conf
```

- tuxY4.sh

```
Unset
# Script for RCOM demonstration: tux64
# Authors:
# Guilherme Duarte Silva Matos (up202208755@up.pt)
# João Vítor da Costa Ferreira (up202208393@up.pt)

# IPs
bridge60=172.16.66.0/24
tux63eth1=172.16.66.1
tux64eth1=172.16.66.254

bridge61=172.16.67.0/24
tux62eth1=172.16.67.1
tux64eth2=172.16.67.253
```

```
routereth2=172.16.67.254

bridgeFTP=172.16.1.0/24
routereth1=172.16.1.61
ftpserver=172.16.1.10



# Reset configuration
systemctl restart networking

# Experience #1
ifconfig eth1 up
ifconfig eth1 ${tux64eth1}/24

# Experience #2
# N/A

# Experience #3
ifconfig eth2 up
ifconfig eth2 ${tux64eth2}/24
sysctl net.ipv4.ip_forward=1
sysctl net.ipv4.icmp_echo_ignore_broadcasts=0

# Experience #4
route add -net ${bridgeFTP} gw ${routereth2}

# Experience #5
echo "nameserver 10.227.20.3" >> /etc/resolv.conf
```

- switch.txt (Não automatizado)

```
Unset
# "Script" for RCOM demonstration: MikroTik Switch
# Authors:
# Guilherme Duarte Silva Matos (up202208755@up.pt)
# João Vítor da Costa Ferreira (up202208393@up.pt)
```

```
# Note: Do not execute this script. It is not automated :/

# Reset configuration
[admin@MikroTik] > /system reset-configuration

# Experience #1
# N/A

# Experience #2
[admin@MikroTik] > /interface bridge add name=bridge60
[admin@MikroTik] > /interface bridge add name=bridge61
[admin@MikroTik] > /interface bridge port remove
numbers: 1
[admin@MikroTik] > /interface bridge port remove
numbers: 2
[admin@MikroTik] > /interface bridge port remove
numbers: 3
[admin@MikroTik]    >    /interface    bridge    port    add    bridge=bridge61
interface=ether2
[admin@MikroTik]    >    /interface    bridge    port    add    bridge=bridge60
interface=ether3
[admin@MikroTik]    >    /interface    bridge    port    add    bridge=bridge60
interface=ether4

# Experience #3
[admin@MikroTik] > /interface bridge port remove
numbers: 10
[admin@MikroTik]    >    /interface    bridge    port    add    bridge=bridge61
interface=ether14

# Experience #4
[admin@MikroTik] > /interface bridge port remove
numbers: 7
[admin@MikroTik]    >    /interface    bridge    port    add    bridge=bridge61
interface=ether11

# Experience #5
# N/A
```

- router.txt (Não automatizado)

```
Unset
# "Script" for RCOM demonstration: MikroTik Router
# Authors:
# Guilherme Duarte Silva Matos (up202208755@up.pt)
# João Vítor da Costa Ferreira (up202208393@up.pt)

# Note: Do not execute this script. It is not automated :/

# Reset configuration
[admin@MikroTik] > /system reset-configuration

# Experience #1
# N/A

# Experience #2
# N/A

# Experience #3
# N/A

# Experience #4
[admin@MikroTik] > /ip address add address=172.16.1.Y1/24 interface=ether1
# Router eth1
[admin@MikroTik] > /ip address add address=172.16.Y1.254/24 interface=ether2
# Router eth2
[admin@MikroTik]    >    /ip    route    add    dst-address=172.16.Y0.0/24
gateway=172.16.Y1.253
# Router → tuxY4eth2 ⇉ bridgeY0

# Experience #5
# N/A
```

E. Código-fonte

parser.h

```C/C++
/**
 * @file parser.h
 * @brief FTP URL parser
 * @authors Guilherme Matos, João Ferreira
 */

#ifndef _PARSER_H_
#define _PARSER_H_

#include <stdbool.h>
#include <arpa/inet.h>

#define MAX_INPUT_SIZE 60

typedef struct
{
  char user[MAX_INPUT_SIZE];
  char password[MAX_INPUT_SIZE];
  char host[MAX_INPUT_SIZE];
  struct sockaddr_in server_addr;
  char url_path[MAX_INPUT_SIZE];
} FTP_URL;

void parseFTPURL(char *url, FTP_URL *ftp);
void printFTP(FTP_URL *ftp);

#endif // _PARSER_H_
```

parser.c

```C/C++
/**
 * @file parsec.c
 * @brief FTP URL parser implementation
```

```
 * @authors Guilherme Matos, João Ferreira
 */

#include "../include/parser.h"
#include "../include/debug.h"
#include <stdio.h>
#include <string.h>
#include <netdb.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define SERVER_PORT 21

void getIPFromHost(char *host, struct sockaddr_in *addr)
{
  struct hostent *h;

  if ((h = gethostbyname(host)) == NULL)
error("Failed to get IP from host");

  char *ip = inet_ntoa(*((struct in_addr *)h->h_addr_list[0]));
  addr->sin_family = AF_INET;
  addr->sin_addr.s_addr = inet_addr(ip);
  addr->sin_port = htons(SERVER_PORT);
}

/**
  * @warning The use of sscanf without field width specifiers is unsafe
(without AddressSanitizer)!
 * If the field is larger than MAX_INPUT_SIZE, an buffer overflow will occur.
 * AddressSanitizer will detect this error.
 */
void parseFTPURL(char *url, FTP_URL *ftp)
{
  // With user and password: ftp://[<user>:<password>@]<host>/<url-path>
    if (sscanf(url, "ftp://%[^:]:%[^@]@%[^/]/%s", ftp->user, ftp->password,
ftp->host, ftp->url_path) == 4)
  {
getIPFromHost(ftp->host, &ftp->server_addr);
```

```
return;
  }

  // With only host and path: ftp://<host>/<url-path>
  if (sscanf(url, "ftp://%[^/]/%s", ftp->host, ftp->url_path) == 2)
  {
strcpy(ftp->user, "anonymous");
strcpy(ftp->password, "anonymous");
getIPFromHost(ftp->host, &ftp->server_addr);
return;
  }

  error("Invalid FTP URL");
}

void printFTP(FTP_URL *ftp)
{
  printf("FTP URL:\n");
  printf("User: %s\n", ftp->user);
  printf("Password: %s\n", ftp->password);
  printf("Host: %s\n", ftp->host);
          printf("IP:     %s:%d\n",     inet_ntoa(ftp->server_addr.sin_addr),
ntohs(ftp->server_addr.sin_port));
  printf("URL path: %s\n", ftp->url_path);
}
```

ftp.h

```
C/C++
/**
 * @file ftp.h
 * @brief (Incomplete) API for communication with FTP servers
 * @authors Guilherme Matos, João Ferreira
 */

#ifndef _FTP_H_
#define _FTP_H_
```

```c
#include "parser.h"
#include <arpa/inet.h>

typedef struct ftpc FTPConn;

FTPConn *ftpConnect(FTP_URL *ftp);
void ftpRetrieve(FTPConn *ftp, char *path);
void ftpDisconnect(FTPConn *ftp);

#endif // _FTP_H_
```

ftp.c

```c
C/C++
/**
 * @file ftp.c
 * @brief (Incomplete) API implementation for FTP communication
 * @authors Guilherme Matos, João Ferreira
 */

#include "../include/ftp.h"
#include "../include/debug.h"
#include <arpa/inet.h>
#include <ctype.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <time.h>
#include <unistd.h>

#define MAX_MSG_LENGTH 30
#define MAX_BUF_LENGHT 1024 * 1024

typedef struct serv_msg {
  unsigned short code;
  union {
```

```
bool head_ending;
struct sockaddr_in addr;
int nbytes;
  };
} ServMsg;

//
================================================================================

void writeToServer(int socketfd, char *msg) {
  int bytes = write(socketfd, msg, strlen(msg));
  if (bytes < 0)
error("Failed to write to the socket");
}

int readMessageFromServer(int socketfd, ServMsg *msg) {
  char header[4] = {0};
  int head = 0;
  int err = 0;
  while (head < 3) {
head += (err = read(socketfd, header + head, (sizeof(header) - 1) - head));
if (err < 0)
       error("Error: could not read from FD: %d", socketfd);
  }
  printf("%s", header);
  msg->code = (((unsigned char)header[0] - '0') * 100) +
              (((unsigned char)header[1] - '0') * 10) +
              (((unsigned char)header[2] - '0') * 1);
  if (msg->code == 227) {
char last = 0;
char curr = 0;
unsigned char addr[6] = {0};
char idx = 0;
bool is_parsing_addr = false;
do {
       if (idx >= 6) {
       info("Tried to read too many bytes.");
       idx = 0;
       }
```

```
        last = curr;
        int err = read(socketfd, &curr, 1);
        if (err < 0)
        error("TODO: I'll think of this message later");
        if (err == 0)
        continue;
        printf("%c", curr);
        if (curr == '(') {
        is_parsing_addr = true;
        } else if (curr == ')') {
        is_parsing_addr = false;
        idx = 0;
        } else if (curr == ',' && last != ',') {
        idx++;
        } else {
        if (!isdigit(curr)) {
        continue;
        }
        addr[idx] *= 10;
        addr[idx] += curr - '0';
        }
} while (curr != '\n' && last != '\r');
if (is_parsing_addr)
        error("Incomplete address parsing");
unsigned short port = (addr[4] << 8) + addr[5];
char ip[21] = {0};
sprintf(ip, "%hhu.%hhu.%hhu.%hhu", addr[0], addr[1], addr[2], addr[3]);
msg->addr.sin_family = AF_INET;
msg->addr.sin_addr.s_addr = inet_addr(ip);
msg->addr.sin_port = htons(port);
  } else if (msg->code == 150) {
char last = 0;
char curr = 0;
bool is_parsing = false;
msg->nbytes = 0;
do {
        last = curr;
        int err = read(socketfd, &curr, 1);
        if (err < 0)
```

```
        error("TODO: I'll think of this message later");
        if (err == 0)
        continue;

        printf("%c", curr);
        if (curr == '(') {
        is_parsing = true;
        } else if (curr == ' ') {
        is_parsing = false;
        } else {
        if (!isdigit(curr)) {
        continue;
        }
        msg->nbytes *= 10;
        msg->nbytes += curr - '0';
        }
} while (curr != '\n' && last != '\r');
  } else {
char last = 0;
char curr = 0;
msg->head_ending = true;
do {
        last = curr;
        int err = read(socketfd, &curr, 1);
        if (err < 0)
        error("TODO: I'll think of this message later");
        if (err == 0)
        continue;
        if (curr != '\r' && curr != '\n') {
        msg->head_ending = false;
        }
        printf("%c", curr);
} while (curr != '\n' && last != '\r');
  }
  if (!isdigit(header[0]) || !isdigit(header[1]) || !isdigit(header[2])) {
msg->code = 230;
  }
  return 0;
}
```

```
//
================================================================================

typedef struct ftpc {
  int socketfd;
  struct sockaddr_in server_addr;
} FTPConn;

FTPConn *ftpConnect(FTP_URL *ftp) {
  FTPConn *auth = (FTPConn *)malloc(sizeof(FTPConn));
  ServMsg msg = {0};
  // Open TCP socket
  auth->socketfd = socket(AF_INET, SOCK_STREAM, 0);
  if (auth->socketfd < 0)
error("Failed to open a socket");

  // Connect to the server
  int err = connect(auth->socketfd, (struct sockaddr *)&ftp->server_addr,
                    sizeof(ftp->server_addr));
  if (err < 0)
error("Failed to connect to the server");

  // Authenticate
  char user[MAX_INPUT_SIZE + 5 + 2 + 1] = {0};
  sprintf(user, "USER %s\r\n", ftp->user);
  writeToServer(auth->socketfd, user);
  info("%s", user);
  // Read Auth user response

  do {
readMessageFromServer(auth->socketfd, &msg);
  } while (msg.code == 220);
  if (msg.code != 331) {
error("Expected a password prompt (331), got (%d)", msg.code);
  }

  char password[MAX_INPUT_SIZE + 5 + 2 + 1] = {0};
  sprintf(password, "PASS %s\r\n", ftp->password);
```

```
  writeToServer(auth->socketfd, password);
  info("%s", password);
  // Read Auth password response
  readMessageFromServer(auth->socketfd, &msg);
  if (msg.code != 230) {
if (msg.code == 530) {
       error("Authentication Rejected. Expected (230), got (%d)", msg.code);
}
error("Authentication failed. Expected (230), got (%d)", msg.code);
  }
  info("Authentication successful!");
  return auth;
}

void  read_from_passive(int  passvdf,  unsigned  int  bufsize,  unsigned  int
bytes_to_read,
                    char *filename) {
  char *buf = calloc(bufsize, sizeof(char));
  FILE *file = fopen(filename, "wb");
  if (file == NULL) {
error("Unable to open file: '%s'", filename);
  }
  int nread;
  while ((nread = read(passvdf, buf, bufsize)) && bytes_to_read > 0) {
if (nread < 0) {
       error("Encountered error while reading file.");
}
bytes_to_read -= nread;
fwrite(buf, 1, nread, file);
  }
  fclose(file);
  free(buf);
  buf = NULL;
  file = NULL;
}

void ftpRetrieve(FTPConn *ftp, char *path) {
  ServMsg msg = {0};
  // DONE: send PASV
```

```
  char passive[] = "PASV\r\n";
  writeToServer(ftp->socketfd, passive);
  info("%s", passive);
  do {

readMessageFromServer(ftp->socketfd, &msg);
  } while (msg.code == 230);
  if (msg.code != 227)
error("Expected to enter passive mode, but received (%d)", msg.code);

  info("Trying to establish connection...");
  // DONE: create socket
  int passvfd = socket(AF_INET, SOCK_STREAM, 0);
  // DONE: connect to socket
  int err = connect(passvfd, (struct sockaddr *)&msg.addr, sizeof(msg.addr));
  if (err < 0)
error("Failed to connect to provided address.");

  info("Established passive connection.");
  // DONE: RETR ftp.path
  char file[MAX_INPUT_SIZE + 5 + 2 + 1] = {0};
  sprintf(file, "RETR %s\r\n", path);
  writeToServer(ftp->socketfd, file);
  info("%s", file);

  readMessageFromServer(ftp->socketfd, &msg);
  if (msg.code < 100 || msg.code >= 200)
error("Preliminary message not in range");
  info("Received preliminary message.");
  int bytes_to_read = msg.nbytes;
  int bufsize = bytes_to_read < MAX_BUF_LENGHT ? bytes_to_read :
MAX_BUF_LENGHT;
  char *filename = strrchr(path, '/');
  if (filename == NULL) {
filename = path;
  } else {
filename++;
  }
  // DONE: Continuously append to file
```

```
  info("Started passive download.");
  read_from_passive(passvfd, bufsize, bytes_to_read, filename);
  info("Finished passive download.");
  // DONE: Receive messages from the server
  do {
readMessageFromServer(ftp->socketfd, &msg);
  } while (msg.code < 200);
}

void ftpDisconnect(FTPConn *ftp) {
  // Send QUIT command
  writeToServer(ftp->socketfd, "QUIT\r\n");
  ServMsg msg = {0};
  // TODO: Read 221 Goodbye

  readMessageFromServer(ftp->socketfd, &msg);
  if (msg.code != 221) {
error("Expected a goodbye message (221), got (%d) (goodvolta?)", msg.code);
  }
  // Close the socket
  if (close(ftp->socketfd) < 0)
error("Failed to close the socket");

  free(ftp);
  ftp = NULL;
}
```

debug.h

```C/C++
/**
 * @file debug.h
 * @brief Auxiliary functions for printing debug messages
 * @authors Guilherme Matos, João Ferreira
 */

#ifndef _DEBUG_H_
#define _DEBUG_H_
```

```
// Comment/uncomment these lines to disable/enable debug messages for each
component
#define DEBUG_FTP

// Print an optional debug message for the FTP server messages
void debug_ftp(const char *msg, ...);
// Print a mandatory error message and exit the program
void error(const char *msg, ...);
// Print a mandatory info message
void info(const char *msg, ...);

#endif // _DEBUG_H_
```

debug.c

```C/C++
/**
 * @file debug.c
 * @brief Implementation of debug messages
 * @authors Guilherme Matos, João Ferreira
 */

#include "../include/debug.h"
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>

void debug_ftp(const char *msg, ...)
{
  #ifdef DEBUG_FTP
  va_list args;
  va_start(args, msg);
  fprintf(stdout, "  ");
  vfprintf(stdout, msg, args);
  fprintf(stdout, "\n");
  va_end(args);
  #endif
```

```
}

void error(const char *msg, ...)
{
  va_list args;
  va_start(args, msg);
  fprintf(stderr, "⚠ ");
  vfprintf(stderr, msg, args);
  fprintf(stderr, "\n");
  va_end(args);
  exit(-1);
}

void info(const char *msg, ...)
{
  va_list args;
  va_start(args, msg);
  fprintf(stdout, "  ");
  vfprintf(stdout, msg, args);
  fprintf(stdout, "\n");
  va_end(args);
}
```

main.c

```C/C++
/**
 * @file main.c
 * @brief Entry point of the FTP client program
 * @authors Guilherme Matos, João Ferreira
 */

#include "include/parser.h"
#include "include/ftp.h"
#include "include/debug.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```c
#include <unistd.h>

#define MAX_URL_SIZE MAX_INPUT_SIZE * 5

int main(int argc, char *argv[])
{
  char ftpURL[MAX_URL_SIZE];
  switch (argc)
  {
  case 1:
    info("No arguments provided; please enter the FTP URL:");
    int size = read(STDIN_FILENO, ftpURL, MAX_URL_SIZE);
    ftpURL[size - 1] = '\0';
    break;

  case 2:
    strcpy(ftpURL, argv[1]);
    break;

  default:
    error("Invalid arguments\n  Usage: download
ftp://[<user>:<password>@]<host>/<url-path>");
  }

  FTP_URL ftp;
  parseFTPURL(ftpURL, &ftp);
  printFTP(&ftp);

  FTPConn *auth = ftpConnect(&ftp);
  ftpRetrieve(auth, ftp.url_path);
  ftpDisconnect(auth);

  return 0;
}
```

Makefile

```C/C++
/**
 * @file main.c
 * @brief Entry point of the FTP client program
 * @authors Guilherme Matos, João Ferreira
 */

#include "include/parser.h"
#include "include/ftp.h"
#include "include/debug.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#define MAX_URL_SIZE MAX_INPUT_SIZE * 5

int main(int argc, char *argv[])
{
  char ftpURL[MAX_URL_SIZE];
  switch (argc)
  {
  case 1:
info("No arguments provided; please enter the FTP URL:");
int size = read(STDIN_FILENO, ftpURL, MAX_URL_SIZE);
ftpURL[size - 1] = '\0';
break;

  case 2:
strcpy(ftpURL, argv[1]);
break;

  default:
error("Invalid arguments\n  Usage: download
ftp://[<user>:<password>@]<host>/<url-path>");
  }

  FTP_URL ftp;
  parseFTPURL(ftpURL, &ftp);
```

```
  printFTP(&ftp);

  FTPConn *auth = ftpConnect(&ftp);
  ftpRetrieve(auth, ftp.url_path);
  ftpDisconnect(auth);

  return 0;
}
```