

# Redes de Computadores

2.º Trabalho Prático - Rede de Computadores

Gonçalo Teixeira e Gonçalo Alves

Mestrado Integrado em Engenharia  
Informática e Computação



**FEUP** FACULDADE DE ENGENHARIA  
UNIVERSIDADE DO PORTO

20 de dezembro de 2020

Porto

## Sumário

Sumário aqui -> dizer para que é que serve este relatório

# Conteúdo

Sumário . . . . .	1
Introdução . . . . .	3
Parte I - Aplicação Download . . . . .	4
Arquitetura . . . . .	4
Resultados . . . . .	4
Parte II - Configuração e Análise de Rede . . . . .	5
Experiência 1 - Configuração IP de Rede . . . . .	5
Conclusão . . . . .	7
Referências . . . . .	8
Anexos . . . . .	9
Imagens . . . . .	9
Aplicação Download . . . . .	11

## **Introdução**

Sumário aqui -> introduzir o tema

Dizer do que se trata a aplicação download

Dizer do que se trata a configurar uma rede de computadores

## **Parte I - Aplicação Download**

Pequeno texto acerca da aplicação download

### **Arquitetura**

Arquitetura da Aplicação Download

### **Resultados**

Pequeno texto acerca de como foi testada a aplicação e um log de resultados

## Parte II - Configuração e Análise de Rede

Pequeno texto acerca da rede a configurar

### Experiência 1 - Configuração IP de Rede

Explicar sucintamente o objetivo desta experiência.

#### 1. O que são os pacotes ARP e para que são usados?

O ARP (*Address Resolution Protocol*) é um protocolo de comunicação que serve para descobrir o endereço da camada de ligação associado ao endereço IP numa LAN (*Local Area Network*). O endereço da camada de ligação é também conhecido por Endereço MAC (*Media Access Control*).

#### 2. Quais são os endereços MAC e IP dos pacotes ARP e porquê?

Executando o comando *ping* do tux3 para o tux4, o tux3 envia uma pergunta para saber qual é o endereço MAC associado ao IP do tux4. A "pergunta" é feita através de um pacote ARP que contém o endereço IP e MAC do tux3 (172.16.30.1 e 00:21:5a:5a:7d:74) e o endereço IP do tux4 (172.16.1.254), uma vez que se quer descobrir o endereço MAC do tux4, o campo dedicado a esse efeito está a 00:00:00:00:00:00. De seguida é enviada uma resposta, também sob a forma de um pacote ARP, do tux4 para o tux3, indicando o seu endereço MAC (00:21:5a:5a:7d:74). Figura 1

#### 3. Quais os pacotes gerados pelo comando ping?

Primeiro o comando *ping* gera pacotes ARP para fazer a relação entre endereços IP e MAC, de seguida gera pacotes ICMP (*Internet Control Message Protocol*).

#### 4. Quais são os endereços MAC e IP dos pacotes ping?

Quando se executa o comando *ping* no tux3 para o tux4, os endereços (IP e MAC) vão ser os endereços dos tux. Podemos ver de seguida os endereços registados nos pacotes de pedido e resposta, respetivamente.

	tux	MAC	IP
Origem	3	00:21:5a:61:24:92	172.16.30.1
Destino	4	00:21:5a:5a:7d:74	172.16.30.254

Tabela 1: Pacote de Pedido

	tux	MAC	IP
Origem	4	00:21:5a:5a:7d:74	172.16.30.254
Destino	3	00:21:5a:61:24:92	172.16.30.1

Tabela 2: Pacote de Resposta

Devem-se consultar as figuras 2 e 3 para referência.

## **5. Como determinar a trama recetora Ethernet é ARP, IP, ICMP?**

O *Ethernet Header* de um pacote contém a informação acerca do tipo da trama. Para as tramas IP, o valor do tipo será 0x0800, se o *IP Header* tiver o valor 1 então o tipo de protocolo é ICMP. Para as tramas ARP o valor do tipo será 0x0806.

Para referência devem-se consultar as figuras [4](#) e [5](#).

## **6. Como determinar o comprimento de uma trama recetora?**

O comprimento de uma trama recetora pode ser determinado inspecionando a entrada no registo do *Wireshark*, tal como se pode observar na figura [6](#).

## Conclusão

Conclusão aqui



## Referências

Referências aqui

## Anexos

### Imagens

16	7.426228675	HewlettP_61:24:92	HewlettP_5a:7d:74	ARP	42 Who has 172.16.30.254? Tell 172.16.30.1
17	7.426352225	HewlettP_5a:7d:74	HewlettP_61:24:92	ARP	60 172.16.30.254 is at 00:21:5a:5a:7d:74
18	7.426352433	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x1a46, seq=9/2304

Figura 1: Pacotes ARP

→	26	10.562260390	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x1a46, seq=9/2304, ttl=64
←	27	10.562398118	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x1a46, seq=9/2304, ttl=64
< >						
> Frame 26: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth0, id 0						
> Ethernet II, Src: HewlettP_61:24:92 (00:21:5a:61:24:92), Dst: HewlettP_5a:7d:74 (00:21:5a:5a:7d:74)						
> Internet Protocol Version 4, Src: 172.16.30.1, Dst: 172.16.30.254						
> Internet Control Message Protocol						

Figura 2: Pacote de Pedido

→	26	10.562260390	172.16.30.1	172.16.30.254	ICMP	98 Echo (ping) request id=0x1a46, seq=9/2304, ttl=64
←	27	10.562398118	172.16.30.254	172.16.30.1	ICMP	98 Echo (ping) reply id=0x1a46, seq=9/2304, ttl=64
< >						
> Frame 27: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth0, id 0						
> Ethernet II, Src: HewlettP_5a:7d:74 (00:21:5a:5a:7d:74), Dst: HewlettP_61:24:92 (00:21:5a:61:24:92)						
> Internet Protocol Version 4, Src: 172.16.30.254, Dst: 172.16.30.1						
> Internet Control Message Protocol						

Figura 3: Pacote de Resposta

```

v Ethernet II, Src: HewlettP_61:24:92 (00:21:5a:61:24:92), Dst: HewlettP_5a:7d:74 (00:21:5a:5a:7d:74)
  > Destination: HewlettP_5a:7d:74 (00:21:5a:5a:7d:74)
  > Source: HewlettP_61:24:92 (00:21:5a:61:24:92)
  Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 172.16.30.1, Dst: 172.16.30.254
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 84
  Identification: 0x9a46 (39494)
  > Flags: 0x40, Don't fragment
  Fragment Offset: 0
  Time to Live: 64
  Protocol: ICMP (1)

```

Figura 4: Campo Type Pacote ICMP

```

v Ethernet II, Src: HewlettP_61:24:92 (00:21:5a:61:24:92), Dst: HewlettP_5a:7d:74 (00:21:5a:5a:7d:74)
  > Destination: HewlettP_5a:7d:74 (00:21:5a:5a:7d:74)
  > Source: HewlettP_61:24:92 (00:21:5a:61:24:92)
  Type: ARP (0x0806)

```

Figura 5: Campo Type Pacote ARP

```

v Frame 9: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth0, id 0
  > Interface id: 0 (eth0)
  Encapsulation type: Ethernet (1)
  Arrival Time: Nov 24, 2020 15:33:40.912061718 Hora padrão de GMT
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1606232020.912061718 seconds
  [Time delta from previous captured frame: 0.408640354 seconds]
  [Time delta from previous displayed frame: 0.408640354 seconds]
  [Time since reference or first frame: 4.418256468 seconds]
  Frame Number: 9
  Frame Length: 98 bytes (784 bits)
  Capture Length: 98 bytes (784 bits)

```

Figura 6: Tamanho de uma trama Recetora

## Aplicação Download

### download.c

```
1  /*      (C)2000 FEUP      */
2
3  #include "connection.h"
4  #include "getip.h"
5
6  #define SERVER_PORT 6000
7  #define SERVER_ADDR "192.168.28.96"
8
9
10 int main(int argc, char **argv) {
11
12     if (argc != 2) {
13         printf("usage: download ftp://[<user>:<password>@]<host>/<url-path>\n");
14         return 1;
15     }
16
17     char user[1024], password[1024], host[1024], url_path[1024], *ip;
18
19     parseArg(argv[1], user, password, host, url_path);
20     printArg(user, password, host, url_path);
21     ip = getIP(host);
22     printf("IP - %s\n", ip);
23     if (ftp_init_connection(ip) == -1) return -1;
24     if (ftp_login(user, password) == -1) return -1;
25     if (ftp_download(url_path) == -1) return -1;
26
27     return 0;
28 }
29
30 void printArg(char *user, char *password, char *host, char *url_path) {
31     printf("User - %s\n", user);
32     printf("Password - %s\n", password);
33     printf("Host - %s\n", host);
34     printf("URL - %s\n", url_path);
35 }
36
37 // ./download ftp://user:1234@sftp.up.pt/pub/ficheiro.zip
38 void parseArg(char *arg, char *user, char *password, char *host, char *url_path) {
39
40
41     char *args = strtok(arg, "/");
42     args = strtok(NULL, "/*");
43     strcpy(user, args);
44
45     args = strtok(NULL, "@");
46     strcpy(password, args);
47
48     args = strtok(NULL, "/");
49     if (args == NULL) {
50         printf("No User\nSetting Default - anonymous\n");
51         strcpy(host, user);
52         strcpy(user, "anonymous");
53     } else {
54         strcpy(host, args);
55     }
56
57     args = strtok(NULL, "\\0");
58     if (args == NULL) {
```

```

59     printf("No Password\nSetting Default- 1234\n");
60     strcpy(url_path, password);
61     strcpy(password, "1234");
62 } else {
63     strcpy(url_path, args);
64 }
65
66 }

```

## getip.h

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <errno.h>
4 #include <netdb.h>
5 #include <sys/types.h>
6 #include <netinet/in.h>
7 #include <arpa/inet.h>
8
9 char *getIP(char host[]);

```

## getip.c

```

1
2 #include "getip.h"
3
4 char *getIP(char host[]) {
5     struct hostent *h;
6
7     /*
8     struct hostent {
9         char    *h_name;   Official name of the host.
10        char    **h_aliases; A NULL-terminated array of alternate names for the host.
11        int     h_addrtype; The type of address being returned; usually AF_INET.
12        int     h_length;   The length of the address in bytes.
13        char    **h_addr_list; A zero-terminated array of network addresses for the host.
14        Host addresses are in Network Byte Order.
15    };
16
17    #define h_addr h_addr_list[0] The first address in h_addr_list.
18    */
19    if ((h = gethostbyname(host)) == NULL) {
20        perror("gethostbyname");
21        exit(1);
22    }
23    char *IP = inet_ntoa(*((struct in_addr *) h->h_addr));
24    printf("Host name   : %s\n", h->h_name);
25    printf("IP Address  : %s\n", inet_ntoa(*((struct in_addr *) h->h_addr)));
26
27    return IP;
28 }

```

## connection.h

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <errno.h>
4 #include <netdb.h>
5 #include <sys/types.h>
6 #include <netinet/in.h>
7 #include <arpa/inet.h>
8
9 char *getIP(char host[]);

```

## connection.c

```
1
2 #include "getip.h"
3
4 char *getIP(char host[]) {
5     struct hostent *h;
6
7     /*
8     struct hostent {
9         char      *h_name;   Official name of the host.
10        char      **h_aliases; A NULL-terminated array of alternate names for the host.
11        int        h_addrtype; The type of address being returned; usually AF_INET.
12        int        h_length;   The length of the address in bytes.
13        char      **h_addr_list; A zero-terminated array of network addresses for the host.
14                                Host addresses are in Network Byte Order.
15    };
16
17    #define h_addr h_addr_list[0] The first address in h_addr_list.
18    */
19    if ((h = gethostbyname(host)) == NULL) {
20        perror("gethostbyname");
21        exit(1);
22    }
23    char *IP = inet_ntoa(*((struct in_addr *) h->h_addr));
24    printf("Host name   : %s\n", h->h_name);
25    printf("IP Address  : %s\n", inet_ntoa(*((struct in_addr *) h->h_addr)));
26
27    return IP;
28 }
```

## Makefile

```
1 CC = gcc
2 CFLAGS = -Wall -pthread -Wno-pointer-sign -g
3 DEPS = connection.h getip.h
4 OBJ = connection.o getip.o
5 TARGETS = download
6
7 all: download
8
9 %.o: %.c $(DEPS)
10     @$(CC) $(CFLAGS) -c -o $@ $<
11     @echo $@
12
13 download: $(OBJ)
14     @$(CC) $(CFLAGS) -o $@ $@.c $(OBJ) -lm
15     @echo $@
16
17 clean:
18     @rm *.o $(TARGETS)
```