Redes de Computadores

Ângela Cardoso e Bruno Madeira

20 de Dezembro de 2015

Sumário

Este relatório tem como objectivo reportar o segundo trabalho prático relativo a Redes de Computadores da Licenciatura com Mestrado em Engenharia Informátia e Computação.

Conteúdo

Capítulo 1

Introdução

Capítulo 2

Aplicação

A aplicação desenvolvida realiza o download de um ficheiro fazendo uso do protocolo FTP segundo o RFC959. Para tal são usadas duas sockets, uma para comandos e outra para dados, de acordo com o modelo descrito na secção 2.3 do RFC959. Os comandos usados podem ser verificados na secção 4 (páginas 25 a 34) do RFC959 e na página 47. É usado o comando PASV sendo que o servidor não usa a porta default para os dados (porta 20) e fica à espera que o cliente estabeleça a ligação.

Todas as funcionalidades desenvolvidas ligadas ao protocolo FTP podem ser verificadas no ficheiro ftp.c e ftp.h dispoíveis nos anexos D.3 e D.2. Apesar de existir uma função denominada ftp_abort esta não envia um comando ABORT (embora esta tenha sido a funcionalidade inicialmente pensada para o mesmo). Esta função apenas fecha as sockets em caso de erro.

Para efectuar ligação ao servidor a aplicação deve também receber um URL no formato estabelecido no RFC1738. Não consideramos utilizadores anónimos como é referido na secção 3.2.1. do RFC1738. No downloader.c (ver anexo D.1) é realizado o parsing do url ficando guardado numa estrutura o nome de utilizador, password, nome do host, caminho até ao ficheiro e o nome do ficheiro.

Uma vez realizado o parsing tenta-se obter o ip do destino e cria-se uma ligação TCP para a porta 21 do servidor a fim de enviar os comandos para pedir a recepção do ficheiro. As funções usadas para obter o ip e para estabelecer são as disponibilizadas nos exemplos do moodle da disciplina. A conexão é realizada com a função connect e não o bind uma vez que a aplicação está do lado do cliente. É utilizada a função gethostbyname para obter o ip, que funciona mas está depreciada segundo o Beej's Guide to Network Programming.

Em termos de estrutura foram desenvolvidos apenas 4 modulos que apresentamos seguidamente.

- downloader - Onde se encontra a função main da aplicação. Também é responsável pelo parsing e por obter o ip destino.
- ftp - Implementa e disponibiliza comandos do protocolo ftp. Os file descriptors das sockets também se encontram neste módulo.
- socket - Apenas disponibiliza uma função para conectar sockets.
- utilities - Apenas disponibiliza auxiliares para debug.

Capítulo 3

Experiências

3.1 Experiência 1 - Configurar uma Rede IP

Nesta experiência criou-se uma LAN com o tux1 e o tux4 na mesma rede e configurados os seus endereços ip. Usando o comando ping na etapa 7, pudemos verificar o envio de um comando ARP em broadcast pelo tux1 que procurava o endereço físico do tux4, necessário ao protocolo ethernet usado para poder comunicar dentro de uma mesma rede local. Seguidamente vericou-se a resposta do tux4 e foi realizado o ping com sucesso.

Atentando nos pacotes capturados com o wireshark do anexo C.1 é possível verificar que os pacotes ARP são identificáveis pelo cabeçalho Ethernet x0806 e os IP pelo x0800. As mensagens de ping podem ser identificadas pelo cabeçalho Ethernet correspondente ao protocolo IP e pelo cabeçalho de IP x01 que corresponde ao protocolo ICMP.

...
TODO frame length

Na lista de pacotes recebidos existe também pacotes do tipo loopback. Este são pacotes que são redireccionados para a máquina que os emitiu com a finalidade (tipicamente) de verificar se esta se encontra em estado operacional. Neste caso, os pacotes recebidos aparentam ser do switch, tendo como endereço de origem e destino o CiscoInc_3a:f1:03.

3.2 Experiência 2 - Implementar 2 LANs num switch

Foram criadas duas LANs uma com o tux1 e o tux4 na rede 172.16.60.0 outra com o tux2 na 172.16.61.0 (com máscara de 24 bits) atribuindoendereços ip às máquinas relativos à rede em que se deviam encontrar e configurando o switch de modo a funcionarem como 2 redes distintas. Constatou-se que apenas computadores que se encontravam na mesma rede virtual local podiam comunicar entre si. Nos anexos C.2.2 e C.2.3 verifica-se que pings realizados do tux1 em broadcast (alignea 7 do trabalho prático) chegam ao tux4 mas não ao tux2. Similarmente, não foi possível encontrar pacotes de ICMP no tux1 e no tux4 quando realizado ping apartir do tux2 como se pode observar nos anexos seguintes.

3.3 Experiência 3 - Configurar um Router em Linux

...

3.4 Experiência 4 - Configurar um Router Comercial e Implementar NAT

...

3.5 Experiência 5 - DNS

3.6 Experiência 6 - Conexões TCP

Nesta experiência usámos a aplicação desenvolvida para realizar o downlad de um ficheiro.

Como esperado, devido ao protocolo FTP, podemos verificar o triple hand shake de duas conexões TCP. O primeiro relativo à ligação usada para envio de comandos e o segundo relativo a de envio de dados que podem ser verificados respectivamente nos anexo XXX e XXX.

TODO...

Na realização da última alínea pudemos verificar que a recepção de dados quando usada uma segunda ligação no TUX2 era afectada. Pode observar-se nos anexos C.6.1 e C.6.2 que a recepção tende para um plateu máximo no TUX1 que é quebrado devido à ligação estabelecida pelo TUX2. Observando o gráfico relativo ao TUX2 podemos ver que este atinge um plateu máximo perto do final da sua ligação que ocorre devido ao TUX1 já ter terminado o download. Além deste plateu máximo podemos verificar que os gráficos são complementares no sentido em que a soma das funções dos dois gráficos, alinhando-os consoante os seus pontos mínimos e máximos dado que as leituras em wireshark não foram iniciadas em simultâneo, resulta aproximadamente numa função constante que apresenta uma recepção entre 10000 e 12000 packets por segundo.

3.7 Experiência 7 - Implementar NAT em Linux

Capítulo 4

Conclusões

Capítulo 5

Esclarecimentos

Apesar de deste relatório referir muitas vezes o TUX2, da experiência 4 até à 7, qualquer referência ao TUX2 corresponde na realidade ao TUX3 uma vez que o TUX2 deixou de estar disponivel apartir de dada altura. Para que o relatório respeite os nomes referidos no guião e usados nos anexos, mantendo a continuidade entre experiências, decidimos continuar a referir-nos ao terceiro computador usado na rede como sendo o TUX2.

Anexos

Anexo A

Enderaços MAC

- TUX1 eth0: 00:0f:fe:8c:af:71
- TUX2 eth0: 00:21:5a:5a:7d:9c
- TUX3 eth0: 00:21:5a:61:2f:4e
- TUX4 eth0: 00:21:5a:c5:61:bb
- TUX4 eth1: 00:c0:df:04:20:8c

Anexo B

Console logs

B.1 Ex4 align.4

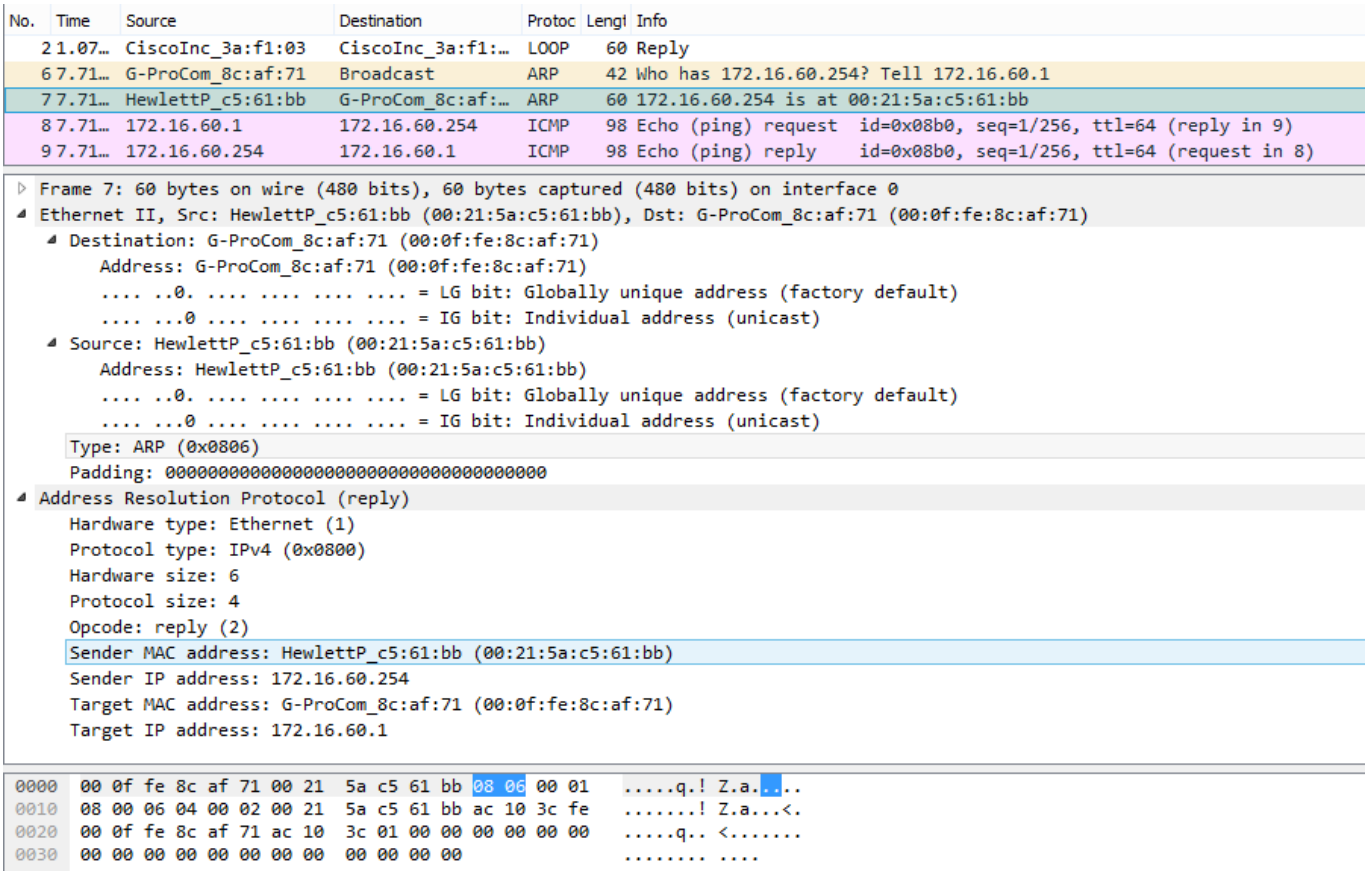
```
tux63:~/Desktop/RCOM/scripts# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use
    Iface
0.0.0.0         172.16.61.254   0.0.0.0         UG    0      0        0
    eth0
172.16.61.0     172.16.61.254   255.255.255.0   UG    0      0        0
    eth0
172.16.61.0     0.0.0.0         255.255.255.0   U     0      0        0
    eth0
tux63:~/Desktop/RCOM/scripts# traceroute 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 (172.16.61.254)  0.498 ms  0.548 ms  0.587 ms
 2  172.16.61.253 (172.16.61.253)  0.873 ms  0.500 ms  0.506 ms
 3  172.16.60.1 (172.16.60.1)  0.799 ms  0.792 ms  0.784 ms
tux63:~/Desktop/RCOM/scripts# ping 172.16.60.1
PING 172.16.60.1 (172.16.60.1) 56(84) bytes of data.
64 bytes from 172.16.60.1: icmp_seq=1 ttl=62 time=0.629 ms
64 bytes from 172.16.60.1: icmp_seq=2 ttl=62 time=0.594 ms
64 bytes from 172.16.60.1: icmp_seq=3 ttl=62 time=0.587 ms
64 bytes from 172.16.60.1: icmp_seq=4 ttl=62 time=0.569 ms
64 bytes from 172.16.60.1: icmp_seq=5 ttl=62 time=0.623 ms
^C
--- 172.16.60.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4000ms
rtt min/avg/max/mdev = 0.569/0.600/0.629/0.031 ms
tux63:~/Desktop/RCOM/scripts# traceroute 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 (172.16.61.253)  0.465 ms  0.343 ms  0.344 ms
 2  172.16.60.1 (172.16.60.1)  0.666 ms  0.662 ms  0.654 ms
tux63:~/Desktop/RCOM/scripts# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use
    Iface
0.0.0.0         172.16.61.254   0.0.0.0         UG    0      0        0
    eth0
172.16.61.0     172.16.61.254   255.255.255.0   UG    0      0        0
    eth0
172.16.61.0     0.0.0.0         255.255.255.0   U     0      0        0
    eth0
tux63:~/Desktop/RCOM/scripts#
```

Anexo C

Wireshark logs and statistics

C.1 Ex1

C.1.1 Captura no TUX1 - ARP



C.1.2 Captura no TUX1 - ICMP

No.	Time	Source	Destination	Protoc	Lengt	Info
2	1.07…	CiscoInc_3a:f1:03	CiscoInc_3a:f1:…	LOOP	60	Reply
6	7.71…	G-ProCom_8c:af:71	Broadcast	ARP	42	Who has 172.16.60.254? Tell 172.16.60.1
7	7.71…	HewlettP_c5:61:bb	G-ProCom_8c:af:…	ARP	60	172.16.60.254 is at 00:21:5a:c5:61:bb
→ 8	7.71…	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) request id=0x08b0, seq=1/256
← 9	7.71…	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x08b0, seq=1/256

```
Fragment offset: 0
Time to live: 64
Protocol: ICMP (1)
Header checksum: 0x095f [validation disabled]
    [Good: False]
```

```
0000  00 21 5a c5 61 bb 00 0f  fe 8c af 71 08 00 45 00   .!Z.a... ...q..E.
0010  00 54 60 2a 40 00 40 01  09 5f ac 10 3c 01 ac 10   .T`*@.@. ._..<...
0020  3c fe 08 00 82 ca 08 b0  00 01 8b 6d 55 56 a0 bd   <....... ...mUV..
0030  00 00 08 09 0a 0b 0c 0d  0e 0f 10 11 12 13 14 15   ........ ........
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ........ .. !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                             67
```

C.2 Ex2

C.2.1 Alínea 7 - Captura no TUX1

No.	Time	Source	Destination	Protoc	Length	Info
5	6.659…	CiscoInc_3a:f1:03	CiscoInc_3a:f1:03	LOOP	60	Reply
	16.66…	CiscoInc_3a:f1:03	CiscoInc_3a:f1:03	LOOP	60	Reply
	22.72…	CiscoInc_3a:f1:03	CDP/VTP/DTP/PAgP/U…	CDP	453	Device ID: tux-sw6 Port ID: FastEthernet0/1
	26.67…	CiscoInc_3a:f1:03	CiscoInc_3a:f1:03	LOOP	60	Reply
	36.67…	CiscoInc_3a:f1:03	CiscoInc_3a:f1:03	LOOP	60	Reply
	37.69…	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x1031, seq=1/256, ttl=64 (no response found!)
	37.69…	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1031, seq=1/256, ttl=64
	38.69…	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x1031, seq=2/512, ttl=64 (no response found!)
	38.69…	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1031, seq=2/512, ttl=64
	39.69…	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x1031, seq=3/768, ttl=64 (no response found!)
	39.69…	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1031, seq=3/768, ttl=64
	40.69…	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x1031, seq=4/1024, ttl=64 (no response found!)
	40.69…	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1031, seq=4/1024, ttl=64
	41.69…	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x1031, seq=5/1280, ttl=64 (no response found!)
	41.69…	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1031, seq=5/1280, ttl=64
	42.69…	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x1031, seq=6/1536, ttl=64 (no response found!)
	42.69…	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1031, seq=6/1536, ttl=64
	42.70…	HewlettP_c5:61:bb	G-ProCom_8c:af:71	ARP	60	Who has 172.16.60.1? Tell 172.16.60.254
	42.70…	G-ProCom_8c:af:71	HewlettP_c5:61:bb	ARP	42	172.16.60.1 is at 00:0f:fe:8c:af:71
	43.69…	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x1031, seq=7/1792, ttl=64 (no response found!)
	43.69…	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x1031, seq=7/1792, ttl=64

C.2.2 Alínea 7 - Captura no TUX2

No.	Time	Source	Destination	Protoc	Length	Info
2	0.461…	CiscoInc_3a:f1:04	CDP/VTP/DTP/PAgP/U…	CDP	453	Device
3	0.881…	CiscoInc_3a:f1:04	CiscoInc_3a:f1:04	LOOP	60	Reply
9	10.88…	CiscoInc_3a:f1:04	CiscoInc_3a:f1:04	LOOP	60	Reply
	20.89…	CiscoInc_3a:f1:04	CiscoInc_3a:f1:04	LOOP	60	Reply
	30.89…	CiscoInc_3a:f1:04	CiscoInc_3a:f1:04	LOOP	60	Reply
	40.90…	CiscoInc_3a:f1:04	CiscoInc_3a:f1:04	LOOP	60	Reply
	50.90…	CiscoInc_3a:f1:04	CiscoInc_3a:f1:04	LOOP	60	Reply
	60.46…	CiscoInc_3a:f1:04	CDP/VTP/DTP/PAgP/U…	CDP	453	Device
	60.91…	CiscoInc_3a:f1:04	CiscoInc_3a:f1:04	LOOP	60	Reply

C.2.3 Alínea 7 - Captura no TUX4

No.	Time	Source	Destination	Protoc	Length	Info
3	2.608…	CiscoInc_3a:f1:06	CiscoInc_3a:f1:06	LOOP	60	Reply
9	12.61…	CiscoInc_3a:f1:06	CiscoInc_3a:f1:06	LOOP	60	Reply
	13.62…	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request
	13.62…	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply
	14.62…	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request
	14.62…	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply
	15.62…	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request
	15.62…	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply
	16.62…	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request
	16.62…	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply
	17.62…	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request

C.2.4 Alínea10 - Captura no TUX1

No.	Time	Source	Destination	Protoc	Length	Info
3	2.102…	CiscoInc_3a:f1:03	CiscoInc_3a:f1:03	LOOP	60	Reply
9	12.11…	CiscoInc_3a:f1:03	CiscoInc_3a:f1:03	LOOP	60	Reply
	22.11…	CiscoInc_3a:f1:03	CiscoInc_3a:f1:03	LOOP	60	Reply
	32.11…	CiscoInc_3a:f1:03	CiscoInc_3a:f1:03	LOOP	60	Reply
	37.99…	CiscoInc_3a:f1:03	CDP/VTP/DTP/PAgP/U…	CDP	453	Device ID
	42.11…	CiscoInc_3a:f1:03	CiscoInc_3a:f1:03	LOOP	60	Reply
	52.13…	CiscoInc_3a:f1:03	CiscoInc_3a:f1:03	LOOP	60	Reply
	62.13…	CiscoInc_3a:f1:03	CiscoInc_3a:f1:03	LOOP	60	Reply
	72.13…	CiscoInc_3a:f1:03	CiscoInc_3a:f1:03	LOOP	60	Reply

C.2.5 Alínea10 - Captura no TUX2

No.	Time	Source	Destination	Protoc	Length	Info
3	2.344…	CiscoInc_3a:f1:04	CiscoInc_3a:f1:04	LOOP	60	Reply
8	11.75…	CiscoInc_3a:f1:04	CDP/VTP/DTP/PAgP/U…	CDP	453	Device ID: tux-sw6 Port ID: FastEthernet0/2
	12.34…	CiscoInc_3a:f1:04	CiscoInc_3a:f1:04	LOOP	60	Reply
	19.31…	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x11b6, seq=1/256, ttl=64 (no response found!)
	20.31…	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x11b6, seq=2/512, ttl=64 (no response found!)
	21.31…	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x11b6, seq=3/768, ttl=64 (no response found!)
	22.31…	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x11b6, seq=4/1024, ttl=64 (no response found!)
	22.35…	CiscoInc_3a:f1:04	CiscoInc_3a:f1:04	LOOP	60	Reply
	23.31…	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x11b6, seq=5/1280, ttl=64 (no response found!)
	24.31…	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x11b6, seq=6/1536, ttl=64 (no response found!)
	25.31…	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x11b6, seq=7/1792, ttl=64 (no response found!)
	26.31…	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x11b6, seq=8/2048, ttl=64 (no response found!)
	27.31…	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x11b6, seq=9/2304, ttl=64 (no response found!)

C.2.6 Alínea10 - Captura no TUX4

No.	Time	Source	Destination	Protoc	Length	Info
3	2.085…	CiscoInc_3a:f1:06	CiscoInc_3a:f1:06	LOOP	60	Reply
9	12.08…	CiscoInc_3a:f1:06	CiscoInc_3a:f1:06	LOOP	60	Reply
	22.08…	CiscoInc_3a:f1:06	CiscoInc_3a:f1:06	LOOP	60	Reply
	23.71…	CiscoInc_3a:f1:06	CDP/VTP/DTP/PAgP/U…	CDP	453	Device
	32.09…	CiscoInc_3a:f1:06	CiscoInc_3a:f1:06	LOOP	60	Reply
	42.10…	CiscoInc_3a:f1:06	CiscoInc_3a:f1:06	LOOP	60	Reply
	52.11…	CiscoInc_3a:f1:06	CiscoInc_3a:f1:06	LOOP	60	Reply
	62.10…	CiscoInc_3a:f1:06	CiscoInc_3a:f1:06	LOOP	60	Reply
	72.11…	CiscoInc_3a:f1:06	CiscoInc_3a:f1:06	LOOP	60	Reply

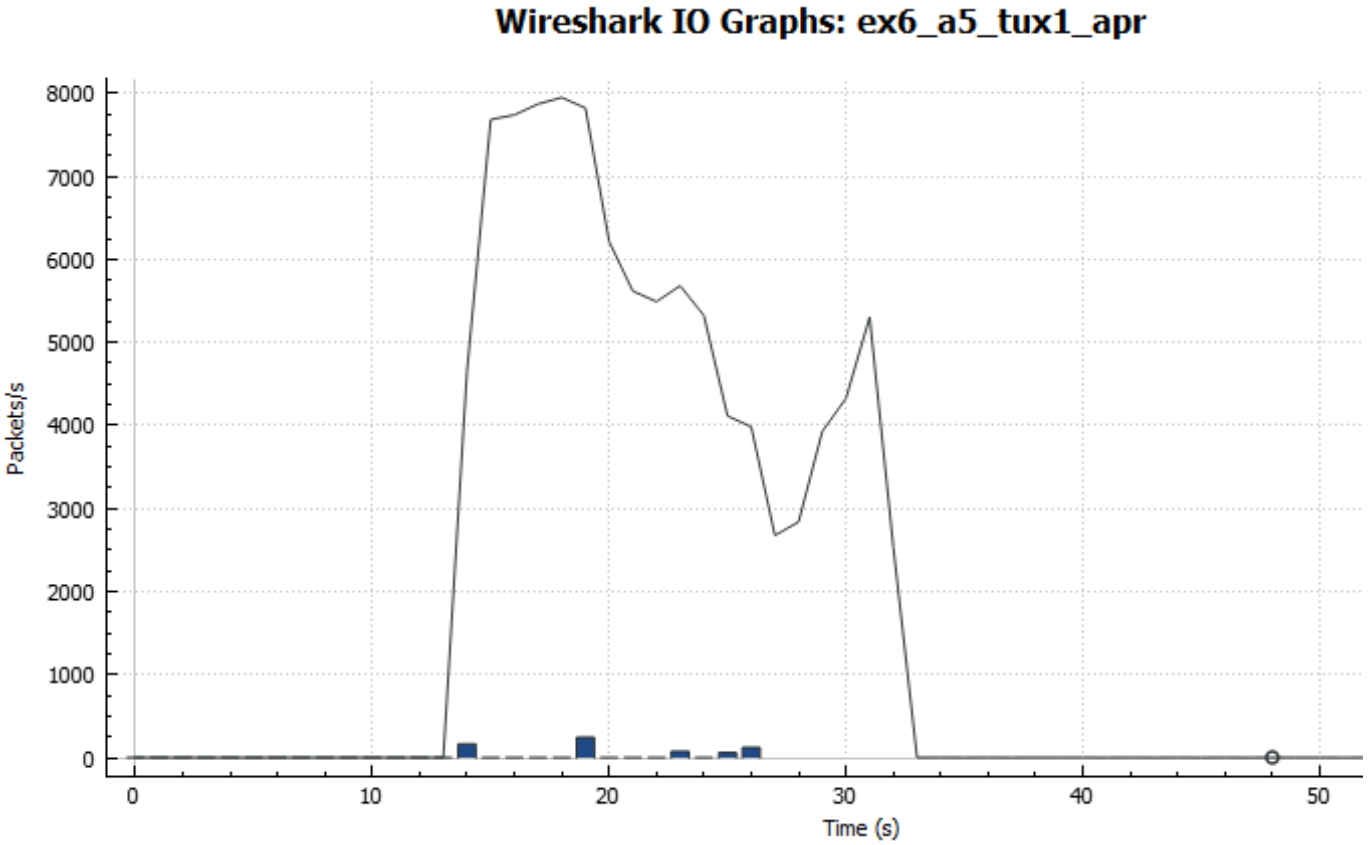
C.3 Ex3

C.4 Ex4

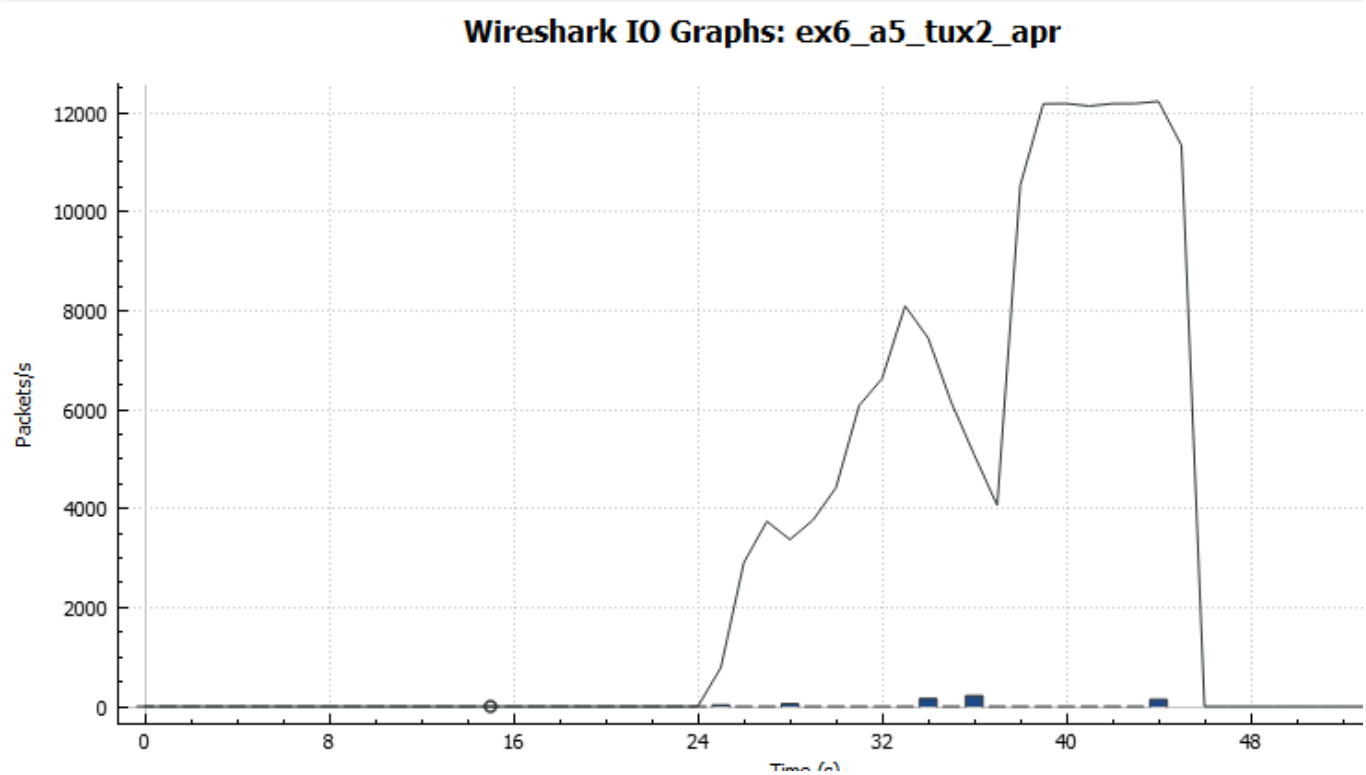
C.5 Ex5

C.6 Ex6

C.6.1 Alínea 5 - Captura no TUX1



C.6.2 Alínea 5 - Captura no TUX2



C.7 Ex7

Anexo D

Código Fonte

D.1 downloader.c

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

//VARS AND STRUCTS
   ------------------------------------------------------------------------------

#define FTP_PORT    21
#define MAX_STRING_SIZE 200
struct /*???*/Info{
  char username[MAX_STRING_SIZE];
  char password[MAX_STRING_SIZE];
  char host_name[MAX_STRING_SIZE];
  char url_path[MAX_STRING_SIZE];
  char filename[MAX_STRING_SIZE];
  char ip[MAX_STRING_SIZE];
};

//AUX FUNCS CODE
   ------------------------------------------------------------------------------

int parse(char *str, struct Info* info) {

  //http://docs.roxen.com/pike/7.0/tutorial/strings/sscanf.xml
    if(4 != sscanf(str, "ftp://[%[^:]:%[^@]@]%[^/]/%s\n", info->
       username, info->password, info->host_name, info->url_path)) {
        return 1;
    }

  //get filename http://stackoverflow.com/questions/32822988/get-the-
     last-token-of-a-string-in-c
      char *last = strrchr(info->url_path, '/') ;
      if(last!=NULL)
      {
    memcpy(info->filename, last+1, strlen(last)+1);
    memset(last,0,strlen(last)+1);
      }
      else {
    strcpy(info->filename,info->url_path);
    memset(info->url_path,0,sizeof(info->url_path));
      }

    return 0;
}
```

```
int get_ip(struct Info* info) {
    struct hostent* host;

    if ((host = gethostbyname(info->host_name)) == NULL) {
        perror("gethostbyname");
        return 1;
    }

    char* ip = inet_ntoa(*((struct in_addr *)host->h_addr));
    strcpy(info->ip, ip);

    printf("Host name  : %s\n", host->h_name);
    printf("IP Address : %s\n", info->ip);

    return 0;
}


//MAIN
   ------------------------------------------------------------------------------

#define DEBUG_ALL 1
int main(int argc,char **argv)
{
  struct Info info;

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

    // ---- URL stuff ----

    //parse
    if(parse(argv[1],&info)!=OK)
    {
      printf("\nINVALID ARGUMENT! couldn't be parsed properly.\n");
      return 1;
    }
    DEBUG_SECTION(DEBUG_ALL,
    printf("\nuser:%s\n",info.username);
    printf("pass:%s\n",info.password);
    printf("host:%s\n",info.host_name);
    printf("urlpath:%s\n",info.url_path);
    printf("filename:%s\n",info.filename);
    );

    //- - - - - -
    get_ip(&info);

    // ---- FTP stuff -----

printf("\n connecting... \n");

    if(ftp_connect(info.ip, FTP_PORT)!=OK)
{ftp_abort(); return 1;}

printf("\n logging in... \n");

    if(ftp_login(info.username, info.password)!=OK)// Send user n pass
{ftp_abort(); return 1;}



    if(strlen(info.url_path)>0) {
      printf("\n changing dir... \n");

      if(ftp_changedir(info.url_path)!=OK)// change directory
      {ftp_abort(); return 1;}
    }
```

```
printf("\n passive mode... \n");

    if(ftp_pasv()!=OK)// passive mode
{ftp_abort(); return 1;}

printf("\n asking for file... \n");

    if(ftp_retr(info.filename)!=OK)// ask to receive file
{ftp_abort(); return 1;}

printf("\n downloading file... \n");

    if(ftp_download(info.filename)!=OK)// receive file
{ftp_abort(); return 1;}

printf("\n disconecting... \n");

    if(ftp_disconnect()!=OK)// disconnect from server
{ftp_abort(); return 1;}

printf("\n downloader terminated ok! \n");

    return 0;
}
```

D.2 ftp.h

```
#ifndef FTP
#define FTP

int ftp_connect( const char* ip, int port);
int ftp_disconnect();

int ftp_login( const char* user, const char* password);
int ftp_changedir( const char* path);
int ftp_pasv();
int ftp_retr( const char* filename);
int ftp_download( const char* filename);

void ftp_abort();

#endif
```

D.3 ftp.c

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>

#include <sys/types.h>
#include <sys/socket.h>


#include "ftp.h"
#include "socket.h"
#include "utilities.h"

#define MAX_STRING_SIZE 500

int control_socket_fd;
int data_socket_fd;


//
   ------------------------------------------------------------------------
```

```
// READ AND SEND
#if 1

int ftp_read(char* str,unsigned long str_total_size)
{
    int bytes = 0;
    if( (bytes = recv(control_socket_fd,str,str_total_size,0)) < 0  )
      {
    perror("ftp_read: recv failed\n");
    return -1;
      }
    return bytes;
}

int ftp_send( const char* str,unsigned long str_size)
{
        int bytes = 0;
    if( (bytes = send(control_socket_fd,str,str_size,0)) < 0  )
      {
    perror("ftp_read: recv failed\n");
    return -1;
      }
    return bytes;
}

#endif


//
   ------------------------------------------------------------------------

// CONECT AND DISCONECT
#if 1

int ftp_connect( const char* ip, int port) {

    int socket_fd;
    char read_bytes[MAX_STRING_SIZE];

    //open control socket
    if ((socket_fd = connect_socket_TCP(ip, port)) < 0)
    {
        printf("ftp_connect: Failed to connect socket\n");
        return 1;
    }

    control_socket_fd = socket_fd;
    data_socket_fd    = 0;

    //Try to read with control socket
    if (ftp_read(read_bytes, sizeof(read_bytes))<0)
    {
        printf("ftp_connect: Failed to read\n");
        return 1;
    }

    return 0;
}

int ftp_disconnect() {
    char aux[MAX_STRING_SIZE];

    //read discnnect
        if (ftp_read(aux, sizeof(aux))<0) {
        printf("ftp_disconnect: Failed to disconnect\n");
        return 1;
    }
    //send disconnect
    sprintf(aux, "QUIT\r\n");
```

```
    if (ftp_send(aux, strlen(aux))<0) {
        printf("ftp_disconnect: Failed to output QUIT");
        return 1;
    }

    close(control_socket_fd);

    return 0;
}

#endif

//
   ------------------------------------------------------------------------

// MAIN OPERATIONS
#if 1

int ftp_login( const char* user, const char* password) {

    char aux[MAX_STRING_SIZE];

    //send username
    sprintf(aux, "user %s\r\n", user);
    if (ftp_send( aux, strlen(aux))< 0) {
        printf("ftp_login: ftp_send failure.\n");
        return 1;
    }
    //receive answer to username
    if (ftp_read( aux, sizeof(aux))<0) {
        printf( "ftp_login:Bad response to user\n");
        return 1;
    }

    //send password
    memset(aux, 0, sizeof(aux));//reuse 2send
    sprintf(aux, "pass %s\r\n", password);
    if (ftp_send( aux, strlen(aux))< 0) {
        printf("ftp_login: failed to send password.\n");
        return 1;
    }
    //receive answer to password
    if (ftp_read( aux, sizeof(aux))<0)
    {
        printf( "ftp_login:Bad response to pass\n");
        return 1;
    }

    return 0;
}

int ftp_changedir(const char* path) {

    char aux[MAX_STRING_SIZE];

    //send cwd command
    sprintf(aux, "CWD %s\r\n", path);
    if (ftp_send(aux, strlen(aux))< 0) {
        printf("ftp_changedir:Failed to send\n");
        return 1;
    }

    //get response
    if (ftp_read(aux, sizeof(aux))< 0) {
        printf("ftp_changedir:Failed to get a valid response\n");
        return 1;
    }

    return 0;
```

```
}

#define DEBUG_PASV 1
int ftp_pasv() {

    char aux[MAX_STRING_SIZE] = "PASV\r\n";

    //send pasv msg
    if (ftp_send(aux, strlen(aux))< 0) {
        printf("ftp_pasv: Failed to enter in passive mode\n");
        return 1;
    }

    //receive response
    if (ftp_read(aux, sizeof(aux))<0) {
        printf("ftp_pasv: Failed to receive information to enter
           passive mode\n");
        return 1;
    }

        DEBUG_SECTION(DEBUG_PASV,printf("pasv():received:%s\n",aux);
    );

    // info was received. scan it
    int ip_bytes[4];
    int ports[2];

    if ((sscanf(aux, "%*[^(](%d,%d,%d,%d,%d,%d)",
    ip_bytes,&ip_bytes[1], &ip_bytes[2], &ip_bytes[3], ports, &ports
       [1]))
        !=6 )
    {
        printf("ftp_pasv: Cannot process received data, must receive 6
           bytes\n");
        return 1;
    }

    // reuse aux and get ip
    memset(aux, 0, sizeof(aux));
    if ((sprintf(aux, "%d.%d.%d.%d",
    ip_bytes[0], ip_bytes[1], ip_bytes[2], ip_bytes[3]))
        <7)
    {
        printf("ftp_pasv: Cannot compose ip address\n");
        return 1;
    }

        DEBUG_SECTION(DEBUG_PASV,printf("pasv():ip:%s\n",aux);
    );

    // calculate port
    int portResult = ports[0] * 256 + ports[1];

    printf("IP: %s\n", aux);
    printf("PORT: %d\n", portResult);

    if ((data_socket_fd = connect_socket_TCP(aux, portResult)) < 0) {
        printf( "ftp_pasv: Failed to connect data socket\n");
        return 1;
    }

    return 0;
}

#define DEBUG_RETR 1
int ftp_retr(const char* filename) {
    char aux[MAX_STRING_SIZE];

    //send retr
```

```
    sprintf(aux, "RETR %s\r\n", filename);
    //sprintf(aux, "LIST %s\r\n", "");
    if (ftp_send(aux, strlen(aux))< 0) {
        printf("ftp_retr: Failed to send \n");
        return 1;
    }

    //get respones
    if (ftp_read(aux, sizeof(aux))< 0) {
        printf("ftp_retr: Failed to get response\n");
        return 1;
    }

    DEBUG_SECTION(DEBUG_PASV,printf("ftp_retr_debug_1:%s\n",aux););

    return 0;
}

#define DEBUG_DOWNLOAD 0
int ftp_download(const char* filename) {

  printf("\ndata_%d__cont_%d\n",data_socket_fd,  control_socket_fd);

    FILE* file;
    int bytes;

    //create n open file
    if (!(file = fopen(filename, "w"))) {
        printf("ftp_download: Failed to create/open file\n");
        return 1;
    }


    char buf[MAX_STRING_SIZE];
    while ((bytes = recv(data_socket_fd,buf,MAX_STRING_SIZE,0))>0) {
        if (bytes < 0) {
            perror("ftp_download: Failed to receive from data socket\n"
               );
            fclose(file);
            return 1;
        }

        DEBUG_SECTION(DEBUG_DOWNLOAD,
                  printf("bytes:%d\n",bytes);
                      printf("rec:%s\n",buf);
                 );

        //output received bytes to file
        if ((bytes = fwrite(buf, bytes, 1, file)) < 0) {
            perror("ftp_download: Failed to write data in file\n");
            return 1;
        }
    }

    //close file and data socket
    fclose(file);
    close(data_socket_fd);

    return 0;
}

void ftp_abort()
{
    printf("\n ABORTED! \n");
    if(data_socket_fd) close(data_socket_fd);
    if(control_socket_fd) close(control_socket_fd);

}
```

```
#endif
```

D.4 socket.h

```
#ifndef SOCKET
#define SOCKET

/*return socket fd*/
int connect_socket_TCP(const char* ip, int port);

#endif
```

D.5 socket.c

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
//#include <netdb.h>
#include <strings.h>
#include <stdio.h>


#include "socket.h"

int connect_socket_TCP(const char* ip, int port)
{
    //adapted from clientTCP.c

    int socket_fd;
    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 ((socket_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("connect_socket:socket()");
        return -1;
    }

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

    return socket_fd;
}
```

D.6 Utilities.h

```
#ifndef UTILITIES
#define UTILITIES

// section: should be a definition created by the programmer that must
   be equal to zero to avoid running the debug code.

#define DEBUG_SECTION(SECT,CODE) {\
if (SECT != 0)\
{\
CODE\
}\
}

#ifndef TYPEDEF_BOOLEAN_DECLARED_
#define TYPEDEF_BOOLEAN_DECLARED_
typedef int bool;
#endif /* TYPEDEF_BOOLEAN_DECLARED_*/

#define TRUE  1
#define YES   1
#define FALSE 0
#define NO    0
#define OK    0

#define PRINTBYTETOBINARY "%d%d%d%d%d%d%d%d"
#define BYTETOBINARY(byte)\
(byte & 0x80 ? 1 : 0),\
(byte & 0x40 ? 1 : 0),\
(byte & 0x20 ? 1 : 0),\
(byte & 0x10 ? 1 : 0),\
(byte & 0x08 ? 1 : 0),\
(byte & 0x04 ? 1 : 0),\
(byte & 0x02 ? 1 : 0),\
(byte & 0x01 ? 1 : 0)

#endif /* UTILITIES */
```