Mestrado Integrado de Engenharia Informática e Computação
Rede de Computadores
23 de Dezembro de 2014

Rede de Computadores

Relatório

Bruno Moreira ei12012@fe.up.pt
Paulo Faria ei12135@fe.up.pt
Ricardo Coutinho ei12161@fe.up.pt
Pedro Moura up201306843@fe.up.pt

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Sumário

O seguinte relatório foi realizado no âmbito do segundo trabalho laboratorial da unidade curricular de Redes de Computadores. O relatório está dividido em 2 partes: o desenvolvimento da aplicação de transferência de ficheiros, e a configuração da rede de computadores.

Índice

Introdução

Os objetivos deste 2º trabalho são a utilização dos conhecimentos cliente-servidor e suas particularidades em TCP/IP e a criação de uma aplicação FTP, recorrendo a sockets. Com o relatório, pretendemos demonstrar como fizemos o trabalho e com que objetivos.

Neste relatório, está disponível a seguinte informação:

- Arquitetura da aplicação de download do nosso programa, que inclui os blocos funcionais e interfaces;
- Visualização com sucesso do download.

Para cada experiência (1-6):

- Arquitetura da rede;
- Objetivos da experiência;
- Principais comandos de configuração;
- Análise aos logs obtidos a partir do Wireshark (se necessário);
- Resposta às questões colocadas.

Parte 1: A aplicação download

A aplicação recebe o argumento URL, que tem o seguinte formato:
ftp://username:password@host/path

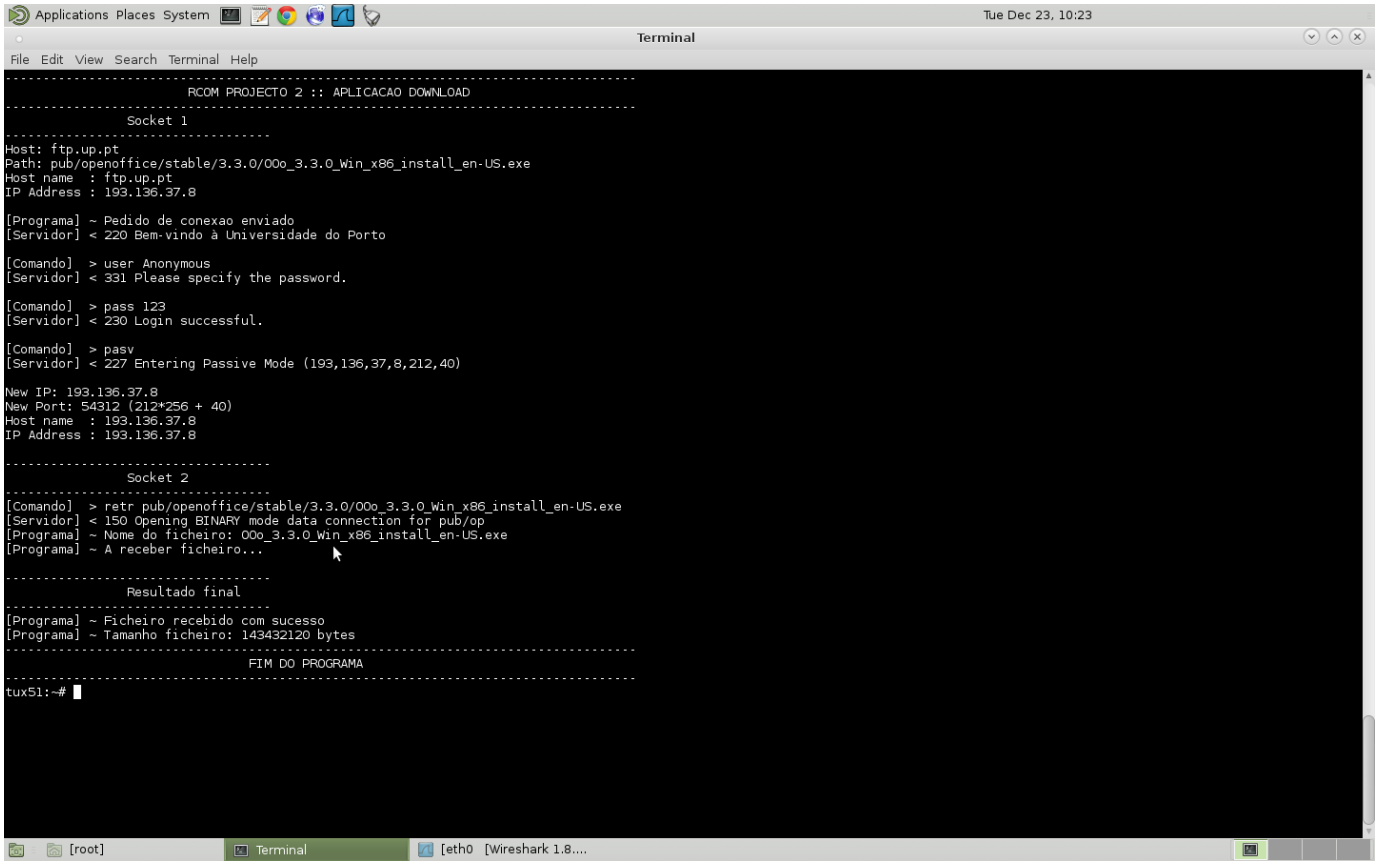
Verificado o formato, são processados os campos requeridos para a ligação ao servidor: username, password, host e path. Se username e password não estão especificadas, o programa assume o utilizador Anonymous (cuja palavra-chave é ‘123’).

Após a filtragem dos valores recebidos como argumento (URL descrita) por expressões regulares, é aberta uma ligação TCP de controlo através de um socket, para se poder enviar comandos e receber respostas entre o cliente (aplicação) e o servidor (host).

Após a ligação ao servidor estiver estabelecida, são enviados comandos que contem o username e a password selecionados descritos em cima, e o comando "pasv" para se pedir ao servidor FTP para transferir dados em modo passivo onde será posteriormente aberta outra ligação TCP (ligação de dados) através de um socket com o IP e a porta recebidas após o comando em causa.

Estando esta ligação de dados aberta, é enviado a partir da ligação de controlo um ultimo comado "retr" seguido do path do ficheiro pretendido, para que depois o servidor FTP envie o ficheiro através da ligação de dados para se poder guardar no computador onde a aplicação FTP é executada.

Depois da receção do ficheiro, os sockets são fechados e a aplicação termina, mostrado se recebeu o ficheiro com sucesso ou não.

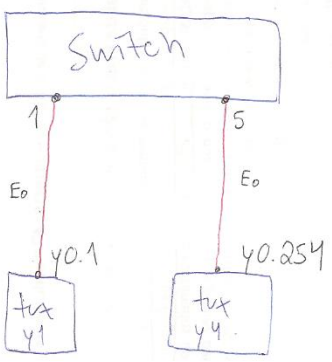


Exemplo de um ficheiro transferido com sucesso

Parte 2: Configuração da rede de computadores

Experiência 1

Arquitetura da rede



Respostas às questões

Os pacotes ARP são pacotes transmitidos na Ethernet, a perguntar quem possui um determinado endereço IP. Esta transmissão chega a todas as máquinas das CS Ethernet, e cada uma verificará esse endereço IP.

O protocolo utilizado por estes pacotes é o ARP (Address Resolution Protocol), que permite associar um endereço IP a um endereço MAC, usando ARP request e ARP reply.

O endereço MAC é um ID físico de uma interface de comunicação (ex: placa de rede) que conecta um dispositivo a uma rede. Este endereço é único por cada interface; tipicamente, este é constituído por 6 pares de dígitos em hexadecimal.
O endereço IP é um ID de um dispositivo numa dada rede. Cada computador tem um IP único, que é o meio no qual as máquinas usam para se comunicarem na rede.
Quando um dispositivo transmite pacotes através da rede (pacotes ARP), estes pacotes contêm o endereço IP e MAC do dispositivo que emitiu e tem o endereço IP do dispositivo a receber os pacotes. Os endereços IP são utilizados para descobrir o endereço MAC do dispositivo a que o transmissor está a enviar pacotes de dados, utilizando para isso ARP.

No computador origem, este gera pacotes ICMP echo request para o dispositivo destino, medindo o tempo entre a transmissão e a receção, e regista qualquer pacote perdido durante a transmissão. No computador destino, este gera pacotes ICMP echo reply para o dispositivo origem.

O endereço IP e o endereço MAC, num pacote IP, corresponde aos endereços de rede e da placa de rede (respetivamente) dos dispositivos transmissor (ou aquele que executa o ping) e recetor (aquele que responde aos pacotes ping).

A partir do cabeçalho de qualquer pacote a transmitir: IP (0x0800) e ARP (0x0806). Dado que o cabeçalho de um pacote IP é constituído por 12 conjuntos de bits (ou categorias) numa dada ordem, o 9º conjunto especifica o protocolo utilizado. Este conjunto contem 8-bits, e quando os 8-bits valerem 0000 0001 (ou seja, o número do protocolo IP for 1), corresponde a pacotes ICMP.

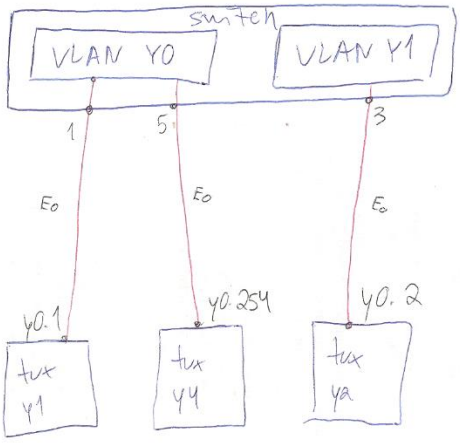
Analisando o cabeçalho de um pacote IP no seu 4º conjunto de bits. Existem 2 bytes reservados para a indicação do comprimento da trama.

Interface que obriga o tráfego enviado por um dispositivo a ser reendereçado para o mesmo computador (ciclo, ou loop). Esta interface é importante pois serve como mecanismo de teste da transmissão ou do transporte da mesma.

Experiência 2

Nota: solicitou-se por parte dos docentes que tux2 tivesse endereço IP 172.16.y0.2

Arquitetura da rede



Respostas às questões

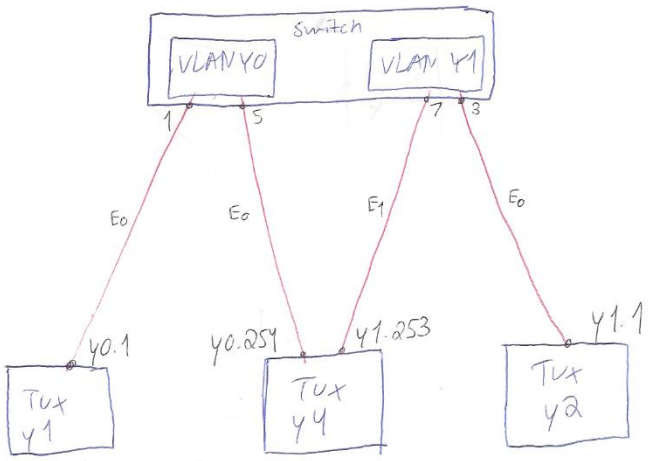
Para configurar vlany0:

```
configure terminal

interface fastethernet 0/1
switchport mode access
switchport access vlan 50
exit

interface fastethernet 0/5
switchport mode access
switchport access vlan 50
exit

end

show vlan brief
```

Conclui-se que apenas existe um domínio: y0.0, mesmo com os computadores com o mesma rede (y0.0) e pertencerem a duas LANs distintas. Por esse motivo, tuxy2 não consegue comunicar nem o tuxy1 nem tuxy4.

Experiência 3

Arquitetura da rede



Respostas às questões

Tuxy1

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
172.16.50.0	0.0.0.0	(24)	U	0	0	0	Eth0
172.16.51.0	172.16.50.254	(24)	UG	0	0	0	Eth0

Tuxy2

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
172.16.50.0	172.16.51.253	(24)	UG	0	0	0	Eth0
172.16.50.0	0.0.0.0	(24)	U	0	0	0	Eth0

Tuxy4

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
172.16.50.0	0.0.0.0	(24)	U	0	0	0	Eth0
172.16.51.0	0.0.0.0	(24)	U	0	0	0	Eth1

Contem o endereço IP da rede no qual os nodos pertencem. É utilizada para reconhecer pacotes IP destinado a outros computadores que estão ligados à mesma rede que o próprio transmisor.

```
nº1(request):
Hardware type: Ethernet (1)
Protocol type: IP(0x0800)
Hardware size: 6
Protocol size:4
Opcode: request (1)
[Us gratuitous: False]
Sender MAC address: G-ProCOm_8c:af:71 (00:0f:fe:8c:af:71)
Sender IP address: 172.16.60.1(172.16.60.1)
Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
Target IP address: 172.16.60.254 (172,16,60.254)

nº1(reply):
Hardware type: Ethernet (1)
Protocol type: IP(0x0800)
Hardware size: 6
Protocol size:4
Opcode: reply (2)
[Us gratuitous: False]
Sender MAC address: Hewlett-_c5:61:bb (00:21:5a:c5:61:bb)
Sender IP address: 172.16.60.254 (172.16.60.254)
Target MAC address: G-ProCom_8c:af:71 (00:0f:fe:8c:af:71)
Target IP address: 172.16.60.1(172.16.60.1)

nº2(request):
Hardware type: Ethernet (1)
Protocol type: IP (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: request(1)
[Is gratuitous: False]
Sender Mac address: Hewlett-_c5:61:bb (00:21:5a:c5:61:bb)
Sender IP address: 172.16.60.254 (172.16.60.254)
Target Mac address: 00:00:00_00:00:00 (00:00:00:00:00:00)
Target IP address: 172.16.60.1(172.16.60.1)

nº2(reply)
Hardware type: Ethernet (1)
Protocol type: IP (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: reply(2)
[Is gratuitous: False]
Sender Mac address: G-ProCOm_8c:af:71 (00:0f:fe:8c:af:71)
Sender IP address: 172.16.60.1 (172.16.60.1)
Target Mac address: Hewlett-_c5:61:bb (00:21:5a:c5:61:bb)
Target IP address: 172.16.60.254(172.16.60.254)
```

```
Type: 8 (Echo (ping) request)
Code: 0
Checksum: 0x1a23 [correct]
Identifier (BE): 9288 (0x2448)
Identifier (LE): 18468 (0x4824)
Sequence number (BE): 1 (0x0001)
Sequence number (LE): 256 (0x0100)
Response In: 10

Type: 0 (Echo (ping) reply)
Code: 0
Checksum: 0x2223 [correct]
```

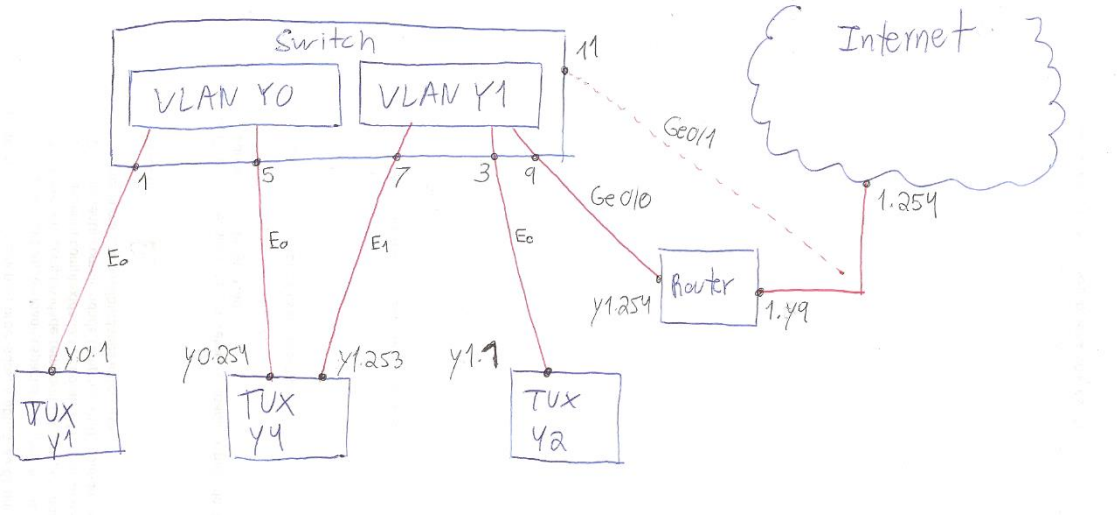
```
Identifier (BE): 9288 (0x2448)
Identifier (LE): 18468 (0x4824)
Sequence number (BE): 1 (0x0001)
Sequence number (LE): 256 (0x0100)
Sequence number (LE): 256 (0x0100)
Response To: 9

Type: 8 (Echo (ping) request)
Code: 0
Checksum: 0x0026 [correct]
Identifier (BE): 9288 (0x2448)
Identifier (LE): 18468 (0x4824)
Sequence number (BE): 2 (0x0002)
Sequence number (LE): 512 (0x0200)
Response In: 12

Type: 0 (Echo (ping) reply)
Code: 0
Checksum: 0x0826 [correct]
Identifier (BE): 9288 (0x2448)
Identifier (LE): 18468 (0x4824)
Sequence number (BE): 2 (0x0002)
Sequence number (LE): 512 (0x0200)
Response To: 11
```

```
Destination:
Hewlett-_c5:61:bb (00:21:5a:c5:61:bb)
Address: Hewlett-_c5:61:bb (00:21:5a:c5:61:bb)
Source:
G-ProCom_8c:af:71 (00:0f:fe:8c:af:71)
Address: G-ProCom_8c:af:71 (00:0f:fe:8c:af:71)

Destination:
G-ProCom_8c:af:71 (00:0f:fe:8c:af:71)
Address: G-ProCom_8c:af:71 (00:0f:fe:8c:af:71)
Source:
Hewlett-_c5:61:bb (00:21:5a:c5:61:bb)
Address: Hewlett-_c5:61:bb (00:21:5a:c5:61:bb)

Destination:
 Hewlett-_c5:61:bb (00:21:5a:c5:61:bb)
Address: Hewlett-_c5:61:bb (00:21:5a:c5:61:bb)
Source:
 G-ProCom_8c:af:71 (00:0f:fe:8c:af:71)
Address: G-ProCom_8c:af:71 (00:0f:fe:8c:af:71)

Destination:
G-ProCom_8c:af:71 (00:0f:fe:8c:af:71)
Address: G-ProCom_8c:af:71 (00:0f:fe:8c:af:71)
Source:
Hewlett-_c5:61:bb (00:21:5a:c5:61:bb)
Address: Hewlett-_c5:61:bb (00:21:5a:c5:61:bb)
```

Em suma, são observados os endereços MAC e os endereços IP das placas de rede que os computadores tuxy2 e tuxy4 estão conectados (os que respondem ao ping) e igualmente para o que o tuxy1 está conectado (o que executa o ping). Isto por que os pacotes IP e ICMP contêm os endereços MAC e IP das placas de rede a que os computadores estão ligados.

Experiência 4

Arquitetura da rede



Respostas às questões

(Ver configuração do router efetuada nos anexos. Tanto serve para a primeira questão como para a terceira questão).

Após apagar a rota 172.16.y0.0 via tuxy4, os pacotes são enviados pela seguinte ordem:

- Tuxy1 (172.16.y0.1) -> Tuxy4 (172.16.y0.254)
- Tuxy4 (172.16.y1.253) -> Tuxy2 (172.16.y1.1) ← não é possível
- Tuxy4 (172.16.y1.253) -> RC (172.16.y1.254)
- RC (172.16.y1.254) -> Tuxy2 (172.16.y1.1)

Isto porque a rota deixou de existir. Se existisse, não havia necessidade de ir ao router.

NAT permite um dado router saber qual o computador que deve retransmitir os dados obtidos da rede exterior.
Se um computador quer fazer conexão ao Google, o pacote de dados a enviar desse computador para o Google é efetuada com sucesso, mas a resposta do Google (ou seja, o pacote de dados) será impossível, por que dentro da rede não sabe qual o computador que pediu a resposta. O router, com o NAT implementado, reconhece essa resposta e retransmite-a para o devido computador.

Experiência 5

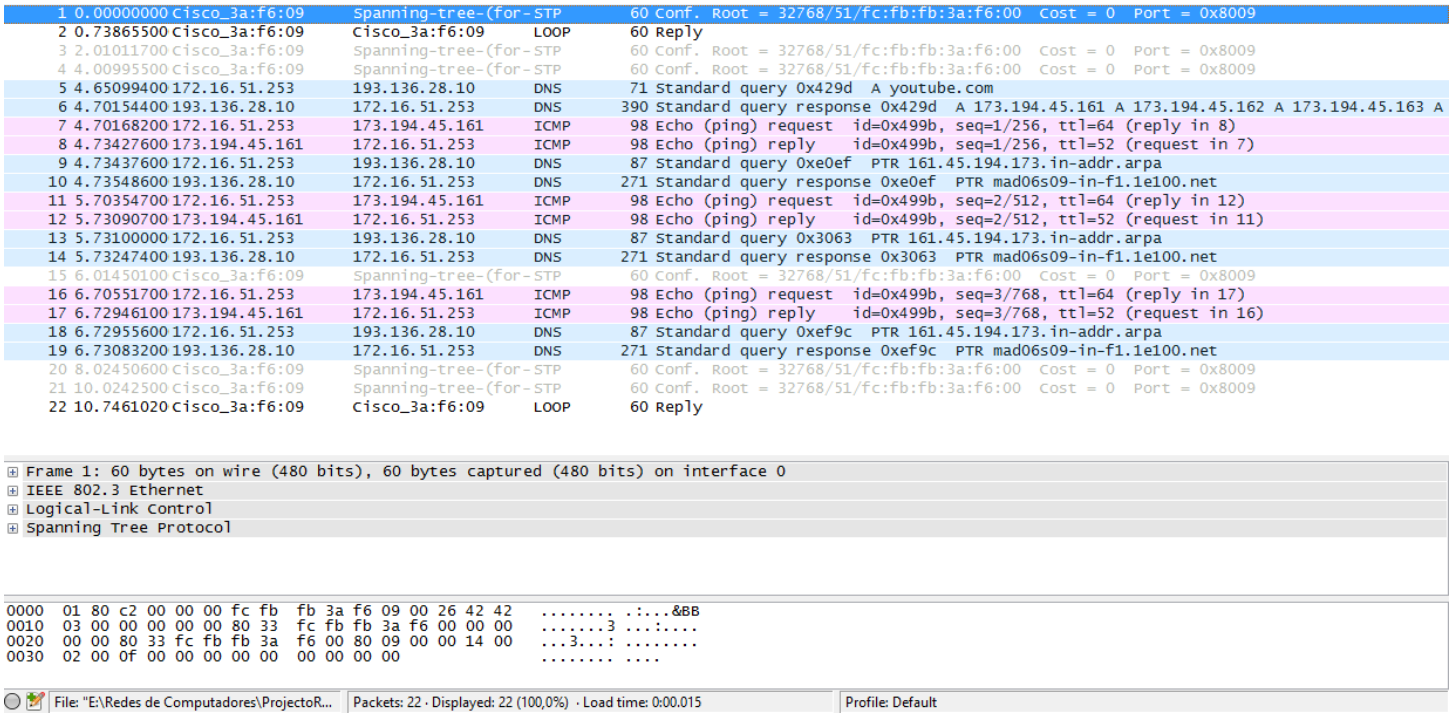
Arquitetura da rede

A mesma que na experiência 4.

Respostas às questões

Para configurar o serviço DNS, é preciso para cada computador efectuar o seguinte:

```
search netlab.fe.up.pt
nameserver 172.16.1.1
```



Transporta o endereço IP e MAC do URL a que o computador executa o ping.

Experiência 6

Arquitetura da rede

A mesma que na experiência 4.

Respostas às questões

Existem duas conexões TCP: a conexão de controlo e a conexão de dados.

A informação de controlo FTP está contida na conexão de controlo.

3 fases da conexão TCP: estabelecimento da conexão, troca de dados e encerramento da conexão.

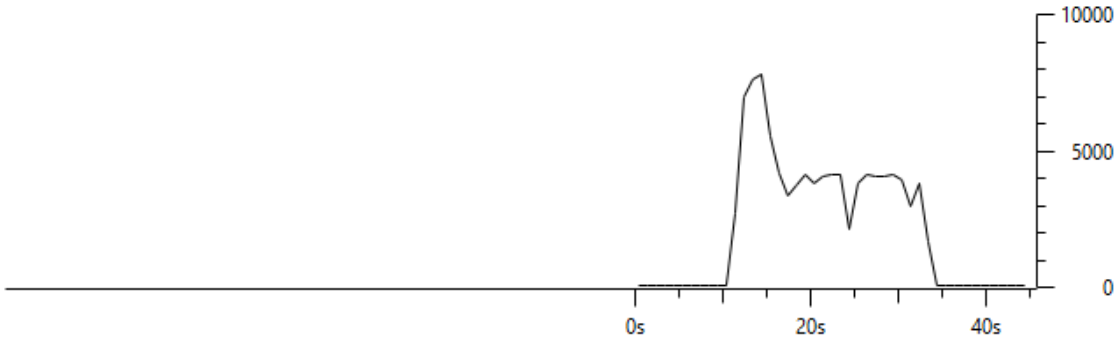
<u>How does the ARQ TCP mechanism work?</u>
O ARQ TCP inicia a ligação através de um socket e envia pacotes de dados mediante um timeout ou falha de envio (através de uma resposta) o pacote é reenviado
o ARQ TCP pode operar sobre os seguintes modelos: Stop-and-wait, Go-Back-N, Selective Repeat.

<u>What are the revevant TCP fields?</u>
Source port, destination port, sequence number, acknowledgement number,
data offset, reserved, control bits(flags),
window size, checksum, urgent pointer, padding.

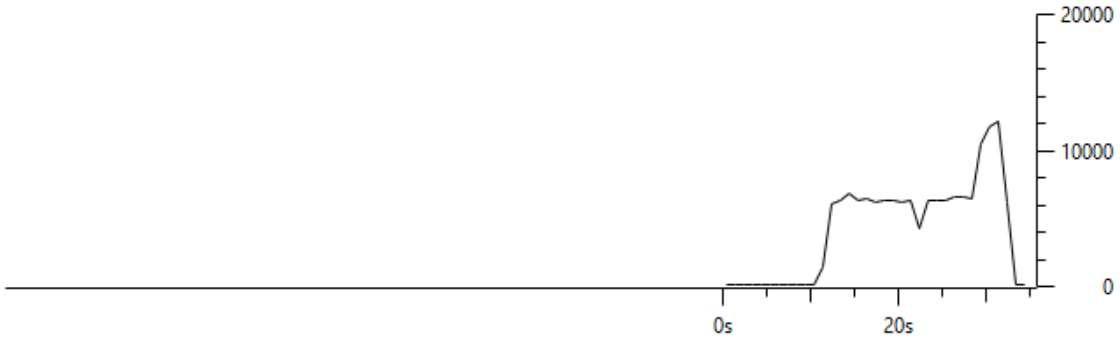
What relevant information can be observed in the logs?
Dos logs de uma ligação ARQ TCP, sabe-se a cada instancia se foi transferido um pacote com sucesso ou não (controlo/pacote de dados).

Sim, é disturbado apesar do TCP usar vários mecanismos para melhorar a performance e evitar o colapso por congestionamento. Tais mecanismos mantêm a "data flow" em níveis aceitáveis. O conhecimento da informação enviada ou a falta dele, é usada para aferir do estado da condição da rede entre o TCP que envia e o que recebe, adequando assim o comportamento da "flow" de data.

Tux 1:



Tux 2:



Referências

http://moodle.up.pt/course/view.php?id=2562
http://technet.microsoft.com/pt-pt/library/cc787920%28v=ws.10%29.aspx

Anexos

Scripts executados nos computadores locais

Experiência 1

Tuxy1

```
/etc/init.d/networking restart
ifconfig eth0 down

ifconfig eth0 up
ifconfig eth0 172.16.50.1/24

echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

ping 172.16.50.254

route –n
arp –a
arp –d 172.16.50.254
```

Tuxy4

```
/etc/init.d/networking restart
ifconfig eth0 down

ifconfig eth0 up
ifconfig eth0 172.16.50.254/24

echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

Experiência 2

Tuxy1

```
/etc/init.d/networking restart
ifconfig eth0 down

ifconfig eth0 up
ifconfig eth0 172.16.50.1/24

echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

ping 172.16.50.254
ping 172.16.50.2
ping –b 172.16.50.255
```

Tuxy2

```
/etc/init.d/networking restart
ifconfig eth0 down

ifconfig eth0 up
ifconfig eth0 172.16.50.2/24

echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

ping –b 172.16.50.255
```

Tuxy4

```
/etc/init.d/networking restart
ifconfig eth0 down

ifconfig eth0 up
ifconfig eth0 172.16.50.254/24

echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

Experiência 3

Tuxy1

```
/etc/init.d/networking restart
ifconfig eth0 down
ifconfig eth1 down

ifconfig eth0 up
ifconfig eth0 172.16.50.1/24
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

route add -net 172.16.51.0/24 gw 172.16.50.254
```

Tuxy2

```
/etc/init.d/networking restart
ifconfig eth0 down
ifconfig eth1 down

ifconfig eth0 up
ifconfig eth0 172.16.51.1/24
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

route add -net 172.16.50.0/24 gw 172.16.51.253
```

Tuxy4

```
/etc/init.d/networking restart
ifconfig eth0 down
ifconfig eth1 down

ifconfig eth0 up
ifconfig eth0 172.16.50.254/24
ifconfig eth1 up
ifconfig eth1 172.16.51.253/24

echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Experiência 4

Tuxy1

```
/etc/init.d/networking restart
ifconfig eth0 down
ifconfig eth1 down

ifconfig eth0 up
ifconfig eth0 172.16.50.1/24
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

route add -net 172.16.51.0/24 gw 172.16.50.254
route add default gw 172.16.50.254
```

Tuxy2

```
/etc/init.d/networking restart
ifconfig eth0 down
ifconfig eth1 down

ifconfig eth0 up
ifconfig eth0 172.16.51.1/24
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

route add -net 172.16.50.0/24 gw 172.16.51.253
route add default gw 172.16.51.254
```

Tuxy4

```
/etc/init.d/networking restart
ifconfig eth0 down
ifconfig eth1 down

ifconfig eth0 up
ifconfig eth0 172.16.50.254/24
ifconfig eth1 up
ifconfig eth1 172.16.51.253/24

echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
echo 1 > /proc/sys/net/ipv4/ip_forward

route add default gw 172.16.51.254
```

Experiência 5

Mesmos scripts que na experiência 4.

Experiência 6

Mesmos scripts que na experiência 4.

Script executado no switch

```
configure terminal

interface fastethernet 0/1
switchport mode access
switchport access vlan 50
exit

interface fastethernet 0/5
switchport mode access
switchport access vlan 50
exit

interface fastethernet 0/3
switchport mode access
switchport access vlan 51
exit

interface fastethernet 0/7
switchport mode access
switchport access vlan 51
exit

interface fastethernet 0/9
switchport mode access
switchport access vlan 51
exit

interface fastethernet 0/11
switchport mode access
switchport access vlan 1
end

show vlan brief
```

Script executado no router

```
configure terminal

access-list 1 permit 172.16.50.0 0.0.0.255
access-list 1 permit 172.16.51.0 0.0.0.255

interface gigabitethernet 0/0
ip address 172.16.51.254 255.255.255.0
no shutdown
ip nat inside
exit

interface gigabitethernet 0/1
ip address 172.16.1.59 255.255.255.0
no shutdown
ip nat outside
exit

ip route 0.0.0.0 0.0.0.0 172.16.1.254
ip route 172.16.50.0 255.255.255.0 172.16.51.253

ip nat pool ovrld 172.16.1.59 172.16.1.59 prefix 24
ip nat inside source list 1 pool ovrld overload

end

show interface gigabitethernet 0/0
show interface gigabitethernet 0/1
show ip route
```

Código da aplicação download

Código do ficheiro ftp.h

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
#include <netdb.h>
#include <signal.h>

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

#include <netinet/in.h>
#include <arpa/inet.h>

#define SERVER_PORT 21

typedef struct sockaddr_in SOCKADDR_IN;
typedef struct hostent HOSTENT;

/**
*       Imprime uma string com mudanca de linha
*/
void msg(char * str)
{
        printf("%s\n", str);
}

/**
*       Imprime uma string, terminando o programa
*/
void msgErro(char * str)
{
        printf("ERROR: %s\n", str);
        exit(0);
}

/**
*       Calcula o tamanho de um ficheiro com o nome 'str'
*       - Devolve -1 se nao existe
*/
int tamanhoFicheiro(char * str)
{
        int r = 0;

        FILE * f;
        f = fopen(str, "rb");

        if (f == NULL)
                return(-1);

        fseek(f, 0L, SEEK_END);
        r = ftell(f);

        fclose(f);
        return(r);
}
/**
*       Obtem um inteiro correspondente a um numero existente numa string
*/
int toInt(char c)
{
        int num = 0;
        switch(c)
        {
                case '0':
                        num = 0;
```

```
                        break;
                case '1':
                        num = 1;
                        break;
                case '2':
                        num = 2;
                        break;
                case '3':
                        num = 3;
                        break;
                case '4':
                        num = 4;
                        break;
                case '5':
                        num = 5;
                        break;
                case '6':
                        num = 6;
                        break;
                case '7':
                        num = 7;
                        break;
                case '8':
                        num = 8;
                        break;
                case '9':
                        num = 9;
                        break;
        }
        return(num);
}

/**
*       Obtem o codigo de resposta por telnet
*/
int getCodigo(char * str)
{
        int r = toInt(str[0])*100 + toInt(str[1])*10 + toInt(str[2]);

        return(r);
}
```

Código do ficheiro ftp.c

```
/*
*       RCOM 2014/2015
*
*       Turma 3 Bancada 5
*/
#include "ftp.h"

int main(int argc, char** argv)
{
        /*********************************
        *       VARIAVEIS
        *********************************/
        SOCKADDR_IN server_addr;
        SOCKADDR_IN server_addr2;

        int     bytes;
        int     sockfd, sockfd2;

        char user[40];
        char pass[40];
        char host[40];
        char path[2048];

        HOSTENT * h;
        HOSTENT * h2;

        char * SERVER_ADDR;

        int codigo = 0;
        char response[256];
```

```
char userSend[45];
char passSend[45];
char pasvSend[45];

int pasv1, pasv2, pasv3, pasv4, pasv5, pasv6;
int port;
char ip[256];

char retrSend[45];

char buffer[256];
char filename[2048];
int i, j, nb, tam;

/*********************************
*       INICIO DO PROGRAMA
*********************************/
system("clear");
msg("-------------------------------------------------------------------------------");
msg("                  RCOM PROJECTO 2 :: APLICACAO DOWNLOAD");
msg("-------------------------------------------------------------------------------");
msg("           Socket 1");
msg("-----------------------------------");

/**
*       Verifica se o programa e executado com argumentos
*/
if (argc != 2)
{
        msgErro("Assim se usa: ftp ftp://[<user>:<password>@]<host>/<url-path>");
}

if(sscanf(argv[1], "ftp://%[^:]:%[^@]@%[^/]/%s\n", user, pass, host, path) == 4)
{
        printf("User: %s\n", user);
        printf("Pass: %s\n", pass);
        printf("Host: %s\n", host);
        printf("Path: %s\n", path);
}
else if(sscanf(argv[1], "ftp://%[^/]/%s\n", host, path) == 2)
{
        printf("Host: %s\n", host);
        printf("Path: %s\n", path);
        strcpy(user, "Anonymous");
        strcpy(pass, "123");

}
else
{
        fprintf(stderr,"Invalid URL! Usage: ftp ftp://[<user>:<password>@]<host>/<url-path>");
}

/**
*       'Inicializacao/preparacao' do socket
*/
if ((h=gethostbyname(host)) == NULL)
{
        herror("gethostbyname");
        exit(1);
}

SERVER_ADDR = inet_ntoa(*((struct in_addr *)h->h_addr));

printf("Host name  : %s\n", h->h_name);
printf("IP Address : %s\n", 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(SERVER_ADDR);    //32 bit Internet address network byte ordered
server_addr.sin_port = htons(SERVER_PORT);               //server TCP port must be network byte ordered

//open an TCP socket
```

```
        sockfd = socket(AF_INET,SOCK_STREAM,0);
        if (sockfd < 0)
        {
                msgErro("socket()");
        }

        msg("\n[Programa] ~ Pedido de conexao enviado");

        //connect to the server
        if(connect(sockfd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0)
        {
                msgErro("connect()");
        }

    //receive response to connect
        memset(response, 0, 256);
        bytes = read(sockfd, response, 50);

        printf("[Servidor] < %s\n", response);

        /**
        *       Envio de User
        ****************************/
        strcpy(userSend, "user ");
        strcat(userSend, user);
        strcat(userSend, "\n");

        printf("[Comando]  > %s", userSend);

        if(write(sockfd, userSend, strlen(userSend))<0)
        {
                msgErro("ftp.c: line 124");
        }

    /**
        *       Receber resposta de User
        ****************************/
        memset(response, 0, 256);
        bytes = read(sockfd, response, 50);
        printf("[Servidor] < %s\n", response);

        /**
        *       Envio de Pass
        ****************************/
        strcpy(passSend, "pass ");
        strcat(passSend, pass);
        strcat(passSend, "\n");
        printf("[Comando]  > %s", passSend);

        if(write(sockfd, passSend, strlen(passSend))<0)
        {
                perror("");
                exit(0);
        }

        /**
        *       Receber resposta de Pass
        ****************************/
        memset(response, 0, 256);
        bytes = read(sockfd, response, 50);
        printf("[Servidor] < %s\n", response);

        codigo = getCodigo(response);
        if (codigo == 530) //password errada
        {
                msgErro("Palavra-chave errada");
        }

        /**
        *       Envio de pasv
        ****************************/
        strcpy(pasvSend, "pasv");
        strcat(pasvSend, "\n");
        printf("[Comando]  > %s", pasvSend);
```

```
	if(write(sockfd, pasvSend, strlen(pasvSend))<0)
	{
		perror("");
		exit(0);
	}

/**
	*	Receber resposta de pasv
	*****************************/
	memset(response, 0, 256);
	bytes = read(sockfd, response, 50);
	printf("[Servidor] < %s\n", response);

	if(sscanf(response, "%*[^(](%d,%d,%d,%d,%d,%d)\n", &pasv1, &pasv2, &pasv3, &pasv4, &pasv5, &pasv6)!=6)
	{
		perror("pasv");
	}

	/**
	*	Novo IP
	*/
	port = pasv5*256 + pasv6;
	memset(ip, 0, 256),
	sprintf(ip, "%d.%d.%d.%d", pasv1, pasv2, pasv3, pasv4);
	printf("New IP: %s\n", ip);
	printf("New Port: %d (%d*256 + %d)\n", port, pasv5, pasv6);

	/**
	*	'Inicializacao/preparacao' do socket 2
	*/
	if ((h2=gethostbyname(ip)) == NULL)
	{
		herror("gethostbyname");
		exit(1);
	}

	SERVER_ADDR = inet_ntoa(*((struct in_addr *)h2->h_addr));

	printf("Host name  : %s\n", h2->h_name);
	printf("IP Address : %s\n\n", SERVER_ADDR);

	//server address handling
	bzero((char*)&server_addr2,sizeof(server_addr2));
	server_addr2.sin_family = AF_INET;
	server_addr2.sin_addr.s_addr = inet_addr(SERVER_ADDR);  //32 bit Internet address network byte ordered
	server_addr2.sin_port = htons(port);            //server TCP port must be network byte ordered

	msg("----------------------------------");
	msg("           Socket 2");
	msg("----------------------------------");

	//open an TCP socket
	sockfd2 = socket(AF_INET,SOCK_STREAM,0);
	if (sockfd2 < 0)
	{
		perror("socket()");
		exit(0);
	}

	//connect to the server
	if(connect(sockfd2, (struct sockaddr *)&server_addr2, sizeof(server_addr2)) < 0)
	{
		perror("connect()");
		exit(0);
	}

	/**
	*	Envio de Retr ficheiro, para socket 1
	*****************************/
	strcpy(retrSend, "retr ");
	strcat(retrSend, path);
	strcat(retrSend, "\n");
	printf("[Comando]  > %s", retrSend);
```

```
if(write(sockfd, retrSend, strlen(retrSend)) < 0)
{
        msgErro("ftp.c : line 226");
}

/**
*       Receber resposta de retr
*****************************/
memset(response, 0, 256);
bytes = read(sockfd, response, 50);
printf("[Servidor] < %s\n", response);

codigo = getCodigo(response);
if (codigo == 550) // ficheiro nao existe
{
        msgErro("Ficheiro nao existe");
}

/**
*       Ficheiro a fazer transferencia
***********************************/
i = strlen(path)-1;
j = 0;
while(i > 0)
{
        if(path[i] == '/')
        {
                i++;
                break;
        }

        i--;
}

while(i < strlen(path))
{
        filename[j] = path[i];
        i++;
        j++;
}
filename[j] = '\0';

printf("[Programa] ~ Nome do ficheiro: %s\n", filename);
printf("[Programa] ~ A receber ficheiro...\n\n");

/**
*       Obter o ficheiro
*****************************/
FILE* fx = fopen(filename, "wb");
while((nb=read(sockfd2, buffer, 255)) > 0)
{
        buffer[nb] = '\0';
        fwrite(buffer, sizeof(char), nb, fx);
}
fclose(fx);

/**
*       Fechar os sockets
*****************************/
close(sockfd);
close(sockfd2);

/**
*       Conclusoes finais
*****************************/
tam = tamanhoFicheiro(filename);

msg("-----------------------------------");
msg("           Resultado final");
msg("-----------------------------------");

if (tam < 0)
{
```

```
                msgErro("[Programa] ~ Ficheiro recebido com erros ou nao foi recebido");
        }
        else
        {
                printf("[Programa] ~ Ficheiro recebido com sucesso\n");
                printf("[Programa] ~ Tamanho ficheiro: %d bytes\n", tam);
        }

        msg("-------------------------------------------------------------------------------");
        msg("                          FIM DO PROGRAMA");
        msg("-------------------------------------------------------------------------------");

        exit(0);
}
```