Faculdade de Engenharia da Universidade do Porto

2º Trabalho Laboratorial

Rede de Computadores

2017/2018 -- Mestrado Integrado em Engenharia Informática e Computação:

Turma 3:

Afonso Bernardino da Silva Pinto

(up201503316@fe.up.pt)

Filipe Miguel Leitao Ribeiro

(ei11141@fe.up.pt)

Tomás Sousa Oliveira

(up201504746@fe.up.pt)

Docentes:

Manuel Ricardo mricardo@fe.up.pt Maria Teresa Andrade mandrade@fe.up.pt

Sumário

Criado no âmbito da disciplina Rede de Computadores leccionada no 3º ano do Mestrado Integrado em Engenharia Informática e Computação, este trabalho tem como objetivos a implementação de um cliente FTP que permita realizar a transferência de ficheiros segundo o protocolo descrito em RFC959, bem como a análise das experiências realizadas nas aulas laboratoriais (a saber: configuração de um IP de rede, de um router em Linux, de um router comercial e do DNS, implementação de LAN's virtuais e do NAT).

Palavras-Chave

FTP; Protocolo; Transferência de Ficheiros; RFC959; RFC1738; Sockets; IP; Router; NAT; Linux; C; Rede de Computadores; FEUP;

Agradecimentos

Este projeto foi resultado de diversas contribuições e colaborações, dada de forma direta e indireta, mas todas elas essenciais à sua realização. Gostaríamos assim de expressar os nossos sinceros agradecimentos a todos os que tornaram possível este trabalho, especialmente à professora Maria Teresa Andrade pela orientação dada e valioso acompanhamento constante durante o desenvolvimento do trabalho.

Índice

1. Introdução

Este projeto foi realizado no âmbito da unidade curricular Redes de Computadores, do 3º ano do Mestrado Integrado em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto.

O primeiro objetivo deste projeto é implementar um cliente FTP (*File Transfer Protocol)* que permita realizar a transferência de ficheiros segundo o protocolo descrito em RFC959.
O endereçamento dos ficheiros a transferir deve adotar as normas de sintaxe descritas em RFC1738.

O segundo objetivo é configurar e analisar uma rede que permita a execução da aplicação supracitada a partir dos conhecimentos adquiridos nas experiências realizadas nas aulas práticas da unidade curricular.

O presente relatório servirá para explorar as funcionalidades deste projeto e fornecer detalhes sobre a sua implementação.

2. Aplicação de Download

2.1. Arquitetura

A aplicação está dividida em 2 módulos: o de processamento do URL e o do cliente FTP propriamente dito.

O processamento do URL consiste essencialmente na criação e preenchimento da seguinte estrutura:

```
typedef struct {
    char* user;
    char* password;
    char* host;
    char* ip;
    char* path;
    char* filename;
    unsigned int port;
} Url;
```

Estrutura Url

A string de input correspondente ao url do ficheiro a transferir é validada com recurso a **expressões regulares** seguindo a norma RFC1738.

Em caso de validade os parâmetros **user**, **password** e **host** são extraídos diretamente dessa *string*, **path** e **filename** são adquiridos através das funções *dirname*() e *basename*() e o **ip** é obtido com recurso à função *getIpByHost*(); o atributo **port** é 21 (número típico da porta de controlo do protocolo FTP).

```
void initURL(Url* url);
int parseURL(const char* urlStr, Url* url);
int processURL(const char* urlStr, Url* url);
int getIpByHost(Url* url);
void showURL(Url* url);
```

Funções do módulo Url

Segue-se uma breve descrição de cada uma das funções utilizadas no módulo Url:

- **initURL** - instância a estrutura e aloca memória para os seus atributos.
- **parseURL** - processa a string de input correspondente ao url com recurso a expressões regulares seguindo a norma RFC1738.
- **processURL** - Preenche a estrutura url com os valores processados.
- **getIpByHost** - obtém o IP a partir do hostname com recurso às funções *gethostbyname*() e *inet_ntoa*().
- **showURL** - Mostra no ecrã o conteúdo da estrutura url.

No que diz respeito ao módulo do cliente FTP é também criada uma estrutura para armazenar os descritores dos sockets utilizados.

```
typedef struct
{
    int sockfd;
    int datafd;
} Ftp;
```

Estrutura Ftp

sockfd corresponderá ao descritor do socket de controlo e **datafd** ao descritor do socket de dados.

```
int downloadLayer(const Url url);
int openConnection(Ftp* ftp, const char* ip, int port);
int connectSocket(const char* ip, int port);
int ftpLogin(Ftp* ftp, const char* user, const char* password);
int ftpCWD(Ftp* ftp, const char* path);
int ftpPassiveMode(Ftp* ftp);
int ftpRetrieve(Ftp* ftp, const char* filename);
int ftpDisconnect(Ftp* ftp);
int socketDownload(Ftp* ftp, const char* filename);
int sendSocket(Ftp* ftp, const char *command);
int recvSocket(Ftp* ftp, char* command, size_t size);
int ftpValidateCode(const char* answer, int expected);
```

Funções do módulo Ftp

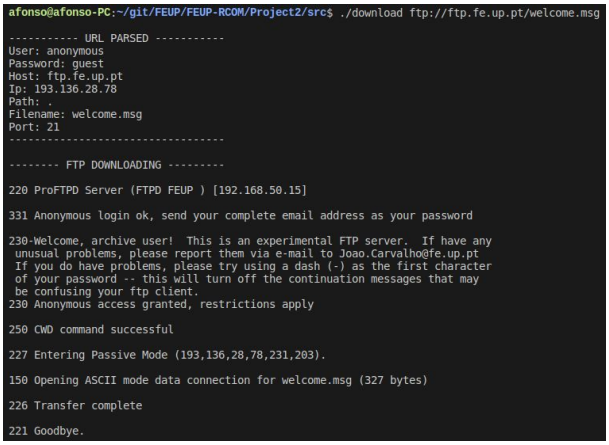
Segue-se uma breve descrição de cada uma das funções utilizadas no módulo Ftp:

- **downloadLayer** - instância a estrutura e é o responsável por gerir que comandos enviar e quando enviar.
- **openConnection** - inicia a comunicação entre o cliente e servidor através da criação de um socket de controlo.
- **connectSocket** - cria e conecta um socket.
- **ftpLogin** - envia os comandos *USER* e *PASS,* com os respetivos dados armazenados na estrutura Url, para o servidor e valida as respostas recebidas.
- **ftpCWD** - envia o comando *CWD,* com path armazenado na estrutura Url, para o servidor e valida a resposta recebida.
- **ftpPassiveMode** - envia o comando *PASV* para o servidor de forma a permitir uma comunicação mútua entre servidor e cliente e valida a resposta recebida. Em caso de sucesso cria um novo socket para transferência de dados.
- **ftpRetrieve** - envia os comando *RETR,* com o filename armazenado na estrutura Url, para o servidor e valida as respostas recebidas.
- **ftpDisconnect** - envia o comando *QUIT* para o servidor de forma terminar a ligação de forma correta e valida a resposta recebida.
- **socketDownload** - responsável pela a transferência do ficheiro propriamente dita.

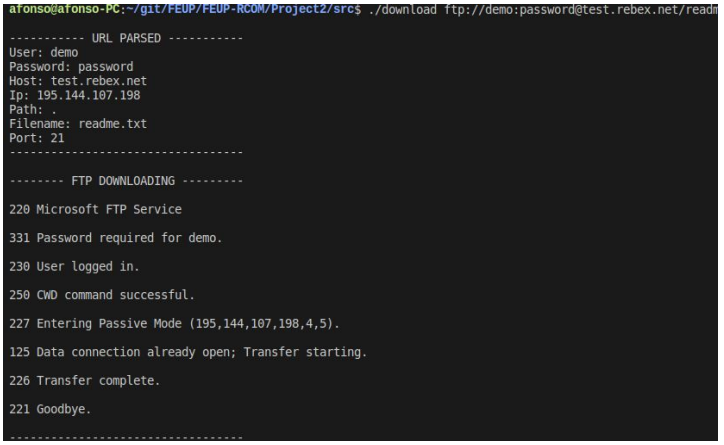
- **sendSocket** - envia para o servidor o conteúdo do buffer que recebe.
- **recvSocket** - lê, linha a linha, as informações que o servidor envia.
- **ftpValidateCode** - verifica se a resposta do servidor está de acordo com o expectável.

2.2. Resultados

A aplicação foi testada com diversos ficheiros, tanto em modo anónimo como em modo autenticado:



Modo Anónimo



Modo Autenticado

Em caso de erro, a aplicação termina com o log do local onde ele ocorreu.

Log em Caso de Erro

3. Configuração e Análise de Rede

3.1. Experiência 1

Para a primeira experiência foi-nos pedido para configurar e estabelecer uma comunicação entre dois computadores, (tux31 e tux34), usando os comandos *ifconfig* e *route*. Estes teriam como endereço de IP, 172.16.30.1 e 172.16.30.254, respetivamente. Logo foram atribuídos esses endereços e ativamos as portas eth0 pelo comando *ifconfig eth0 up*.

Em seguida usamos o comando *ping* para verificar a conetividade entre ambos os computadores, o qual gerou pacotes ICMP.

Usámos a ferramenta do Wireshark para começar a capturar os packets no computador tux31. Ao verificar o log dessa ferramenta, podemos verificar que o pacote ARP (pacotes usados para obter o endereço MAC - identificador único de uma placa de rede - de um IP - etiqueta numérica associada a cada dispositivo de uma dada rede de computadores que segue o Protocolo da Internet) era enviado em mensagens de broadcast aos restantes computadores.

Podemos determinar se um pacote é ARP, IP ou ICMP a partir dos headers, a saber: Ethernet Header: ARP (0x0806), IP (0x0800) e IP Header: ICMP (1).

O tamanho de uma frame é verificado no campo *length* do frame recebido.

A interface LOOP é importante pois envia de 10 em 10 segundos um pacote que determina se a ligação e o sistema estão ativos.

3.2. Experiência 2

Nesta segunda experiência configuramos um novo computador (tux32) e criámos duas LANs virtuais (VLANs).

Para configurar cada uma, usamos os seguintes comandos:

```
configure terminal
vlan 30
end

configure terminal
interface fastethernet 0/1
switchport mode access
switchport access vlan 30
end
```

(comandos utilizados para configurar a vlan 30)

Esta primeira rede virtual (vlan30) é constituída pelos computadores tux31 e tux34 e a segunda (vlan31) pelo computador tux32.

Enviando um comando ping do tux31 tanto para o tux34 como para o tux32, podemos concluir que existem duas sub redes (uma constituida pelos tux31 e tux34 e outra pelo tux32) e por conseguinte dois domínios broadcast distintos, visto que apenas o tux34 recebe o ping.

3.3. Experiência 3

Nesta terceira experiência era pretendido transformar o computador tux34 num router com o propósito de o ligar à vlan30 e vlan31.

Para esse efeito foi necessário ligar a interface eth1 do tux34 e adicioná-la à sub rede do tux32.

De seguida, adicionamos uma rota ao tux31 e outra ao tux32, para que pudessem enviar informações, através do tux34, entre eles.

No final conseguimos comunicar entre qualquer um dos 3 computadores intervenientes.

Durante estas comunicações, pudemos observar pacotes ICMP que continham os pings de cada tux (identificados pelo seu IP e endereço MAC) do ou para o switch (identificado pelo seu endereço MAC).

3.4. Experiência 4

Na quarta experiência era pedido que fosse configurado um router com NAT (Network Address Translation) implementado. O objetivo desta experiência era habilitar a comunicação entre a rede interna anteriormente criada e redes externas. No entanto, como se trata de uma rede privada, os IP's não são reconhecidos fora da mesma, o que levou a uma necessidade de reescrever estes IP's para que pudessem aceder a redes externas.

Para configurar o router, foi necessário configurar a interface interna no processo NAT. Para tal, usámos os seguintes comandos para configurar a interface gigabitethernet 0/0 do router.

```
interface gigabitethernet 0/0
ip address 172.16.31.254 255.255.255.0
no shutdown
ip nat inside
exit
```

Depois foi necessário configurar a interface externa, utilizando-se os seguintes comandos:

```
interface gigabitethernet 0/1
ip address 172.16.1.39 255.255.255.0
no shutdown
ip nat outside
exit
```

Numa fase seguinte, foi necessário garantir a gama de endereços, sendo utilizados os seguintes comandos:

```
ip nat pool ovrld 172.16.1.39 172.16.1.39 prefix 24
ip nat inside source list 1 pool ovrld overload
```

Depois foi necessário criar uma lista de acessos e permissões dos pacotes, para as duas sub-redes, utilizando os comandos:

```
access-list 1 permit 172.16.30.0 0.0.0.7
access-list 1 permit 172.16.31.0 0.0.0.7
```

Por fim, foi necessário definir as rotas internas e rotas externas, para que o router saiba para qual IP deve redirecionar os pacotes. Para tal, foram usados os seguintes comandos:

```
ip route 0.0.0.0 0.0.0.0 172.16.1.254
ip route 172.16.30.0 255.255.255.0 172.16.31.253
```

3.5. Experiência 5

O objetivo desta experiência era habilitar o acesso a redes externas através de nomes de domínio em detrimento dos endereços de IP previamente utilizados.
Para tal, foi necessário configurar o DNS. Para isso, editaram-se os ficheiros “resolv.conf” dos hosts da rede interna com os seguintes comandos:

```
cp /etc/resolv.conf /etc/resolv.conf.backup
echo "search netlab.fe.up.pt" > /etc/resolv.conf
echo "nameserver 172.16.1.1" >> /etc/resolv.conf
```

Para testar os efeitos da experiência, foi enviado com sucesso o comando ping para www.google.com.

3.6. Experiência 6

Nesta experiência, era pedido que se executasse a aplicação desenvolvida e descrita na primeira parte do relatório, utilizando o setup das experiências anteriores.
Foi utilizado o servidor FTP da FEUP com o objetivo de efetuar o download de um ficheiro, o que foi conseguido, demonstrando que as redes interna e externa estavam bem configuradas.

Verificamos que tivemos 2 conexões TCP abertas pelo servidor, na qual a primeira delas (conexão de controlo) transportava a informação do servidor FTP. As fases da conexão são o estabelecimento, envio de dados e terminação.

Para os mecanismos ARQ TCP, sempre que o receptor receberia a informação, era enviada de volta uma mensagem de sucesso. O receptor usado na conexão TCP foi o *Selective Process ARQ* que permite que vários pacotes sejam enviados sem ser usados.

4. Conclusões

Com a realização deste projeto pudemos familiarizar-mos-nos com o protocolo FTP bem como com configurações genéricas da rede.

Concluímos com sucesso as experiências que nos foram propostas o que contribuiu para aprofundar os conhecimentos do grupo nos conteúdos lecionados na cadeira.

A implementação do cliente FTP foi também realizada com sucesso. A realização desta parte do projeto foi vista com grande interesse por todos os elementos do grupo e revelou-se um excelente método para tomar conhecimento de como é implementado e usado um protocolo oficial.