Redes de Computadores - 2º Projeto

Segundo projeto da unidade curricular Redes de Computadores (RCOM)

Autores

Francisco Pires da Ana (up202108762) & João Torre Pereira (up202108848)

Introdução

Este projeto teve como objetivo o desenvolvimento e teste de um programa de *download*, usando o protocolo FTP, para aplicação numa rede configurada e estudada durante as aulas práticas. Em suma, no fim deste trabalho devemos ser capazes de transferir um ficheiro da internet utilizando uma rede configurada utilizando um programa desenvolvido por nós.

Parte 1 - Aplicação de Download

A primeira parte do projeto consiste no desenvolvimento, na linguagem C, de uma aplicação muito simples que faz *download* de um ficheiro via protocolo FTP. É nos desafiado a explorar este protocolo através da leitura do RFC959, que inclui toda a documentação necessária para sua utilização.

O programa recebe um argumento, um URL no seguinte formato:

```
ftp://[:@][:]/
```

Este URL indica, segundo o RFC1738, todas as informações necessárias bem como opcionais, para transferir o ficheiro. Os dois únicos parametros obrigatórios são o host e o url-path, este último sendo o caminho para o ficheiro pretendido. Os restantes parametros são valores padrão que o usuário tem a liberdade de adicionar neste URL.

Arquitetura

O programa segue o seguinte fluxo para que consiga transferir o ficheiro solicitado:

1. Análise do URL passado como argumento

```
typedef struct {
    char *username;
    char *password;
    char *host;
    char *port;
    char *path;
} ParsedURL;

ParsedURL components = parse_url(argv[1]);
```

Esta função é responsável pela extração das informações necessárias para preencher o struct `ParsedURL`. Se não for fornecido nenhum username, será atribuído o valor de `anonymous` e se nenhuma porta for especificada, terá o valor `21`, o padrão do protocolo FTP. A palavra-passe padrão é uma string vazia.

2. Conectar ao servidor remoto e autenticar.

```
int connection_fd = connect_to_host(components.host, components.port);
login(connection_fd, components.username, components.password);
```

Pega-se nas informações anteriormente extraidas e conecta-se ao servidor remoto, no entanto, é ainda necessário conhecer o IP do hostname. Utiliza-se então a função `getaddrinfo`, uma espécie de sistema de resolução de DNS incluído na API do POSIX, que retorna uma lista de IPs a partir de um hostname. O uso de `getaddrinfo` revela-se ser mais flexivel que outros como `gethostbyname`.

A autenticação no servidor é feita através dos comandos `USER` e `PASS`. Todas as mensagens recebidas incluiem um código de 3 dígitos que é lido e utilzado para a confirmação do seguimento do fluxo pretendido. Por exemplo, espera-se

ler o código `230` assim que a autenticação seja bem sucessida.

3. Entrar no modo passivo

```
enter_passive_mode(connection_fd, passive_host, passive_port);
```

Entra-se no modo passivo através do comando `PASV` que, em caso de sucesso, responde com a informação necessária para construir o host e a porta a utilizar para a transferência no formato `(h1, h2, h3, h4, p1, p2)`. No final, temos:

- host: `h1.h2.h3.h4`
- porta: `p1 * 256 + p2`

4. Conectar ao segundo host para a receber o ficheiro e iniciar a transferência.

```
int passive_connection_fd = connect_to_host(passive_host, passive_port);
start_transfer_command(connection_fd, components.path) < 0);
```

Conecta-se ao host que será utilizado para a troca de informação relativas do ficheiro e inicia-se a transferência com o comando `RETR` seguido do *path* para o ficheiro pretendido enviado no primeira conexão.

5. Receção e escrita do ficheiro e fecho das conexões.

```
receive_file(passive_connection_fd, components.path);
close(connection_fd);
close(passive_connection_fd);
```

Ao longo da leitura dos bytes da conexão passiva, escreve-ve para um ficheiro no disco com o mesmo nome que o ficheiro requisitado. No fim da transferênia fecham-se as duas conexões TCP com o função `close`.

Sucesso do *Download*

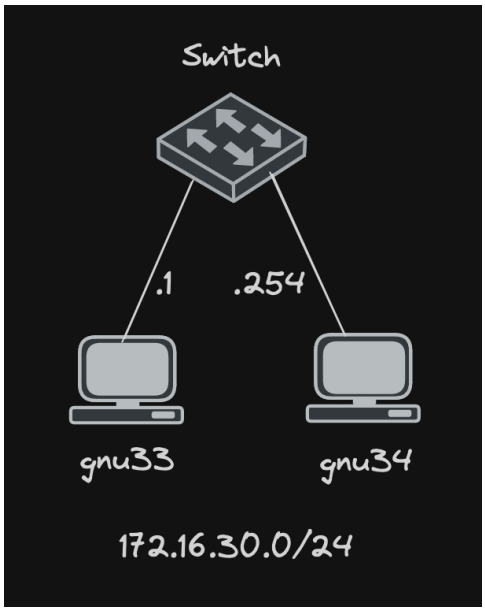
Testámos o funcionamento deste programa transferindo vários ficheiros, de diferetes tamanhos. O programa mostrou-se funcional, correto e consistente. Mostramos, seguidamente, a execução bem sucedida de uma transferência:

```
❯ make run
./bin//main "ftp://ftp.up.pt/pub/kodi/apt/pre-release/ios/Release"
username: anonymous
password:
host: ftp.up.pt
port: 21
path: pub/kodi/apt/pre-release/ios/Release
Connected to 193.137.29.15
passive_host: 193.137.29.15
passive_port: 51456
Connected to 193.137.29.15
Sending file with size: 179
File named 'Release' received
Bytes received: 179
❯ cat Release
Origin: teamXBMC-pre
Label: teamXBMC-pre
Suite: pre-release
Version: 1.0
Codename: XBMC
Architectures: iphoneos-arm
Components: main
Description: The Official teamXBMC Repository!%
```

Parte 2 - Configuração e estudo de uma rede

Experiência 1 - Configurar uma rede IP

Arquitetura da rede



Existem os dispositivos `gnu33` e `gnu34` conectados a um switch.

Objetivos da Experiência

Esta experiência tem como objetivo a configuração dos endereços IP dos dois computadores - gnu33 `(172.16.30.1)` e gnu34 `(172.16.30.254)` - ligados a um *switch*. Pretende-se ainda analisar o funcionamento do protocolo ARP.

Principais comandos de configuração

```
ifconfig eth0 172.16.30.1/24 #gnu33
ifconfig eth0 172.16.30.254/24 #gnu34
```

Análise dos logs

Tratou-se depois de verificar a conectividade entre as duas máquinas, gerando sinais do `gnu33` para o `gnu34`

```
ping 172.16.30.254 #gnu33
```

> `ping <ip>` gera pacotes de 64 bytes enviados da máquina local para a máquina identificada pelo ip especificado.

Consultámos as tabelas ARP, de onde podemos retirar os endereços MAC associados a cada máquina.

```
arp -a
```

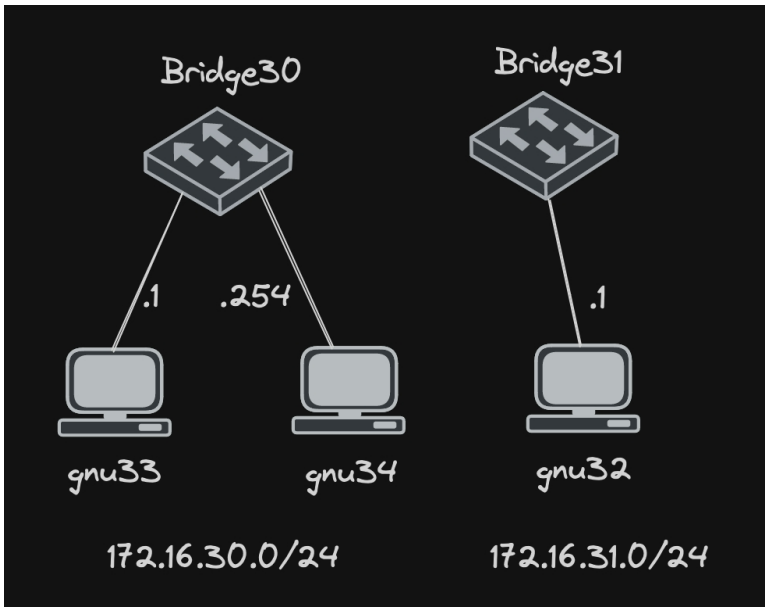
> Os endereços MAC identificam uma placa de rede numa rede local (ao nível da ligação/hardware) enquanto o endereço IP identifica o dispositivo (ao nível da rede).
>
> Os pacotes ARP (ARP packets) são mensagens de solicitação e resposta trocadas entre dispositivos em uma rede local para proceder ao mapeamento de andereços MAC e endereços IP.

Analisaram-se os pacotes de request assim como os respetivos acks de reposta e conclui-se que a partir de dispositvo `gnu33` é possível chegar ao `gnu34`, dado estes dois estarem conectados no switch.

No processo do comando *ping*, o `gnu33` envia primeiramente um pedido ARP para descobrir o endereço MAC do dispositivo identificado no endereço IP do comando. Só após receber o endereço MAC do `gnu34` são mandados os pactos *ping*.

Experiência 2 - Implementar duas *bridges* num *switch*

Arquitetura da rede



No mesmo switch existem duas subnetworks separadas, a `bridge30` e a `bridge31`, os dispositivos anteriormente conectaods pertencem à primeira e liga-se um novo dispositivo `gnu32` à segunda subnetwork `bridge31`.

Objetivos da Experiência

Esta experiência tem como objetivo a configuração de duas LANs (*Local Area Network*), implementando duas *bridges* no *switch* - `bridge30` e `bridge31`. À primeira deverão estar ligadas as máquinas `gnu33` e `gnu34` com os endereços IP configurados na experiência anterior, ao passo que à segunda *bridge* deverá estar ligado o computador gnu32, com o endereço IP `172.16.31.1`.

Principais comandos de configuração

```
# gnu32
ifconfig eth0 172.16.31.1/24

# Consola do switch

# Criar as bridges
/interface bridge add name=bridge30
/interface bridge add name=bridge31

# Remover bridges default dos ethers conectados (9, 13, 14)
/interface bridge port remove [find interface = etherX]

# Conexão dos dispositivos às bridges
/interface bridge port add bridge=bidge30 interface=ether9
/interface bridge port add bridge=bidge30 interface=ether13
/interface bridge port add bridge=bidge31 interface=ether14
```

Análise dos logs

Depois da configuração verificámos que há 2 domínios de *broadcast*. A partir do `gnu33` conseguimos dar *ping* ao `gnu34` mas não ao `gnu32` (sendo inclusivamente reportada uma reposta negativa).

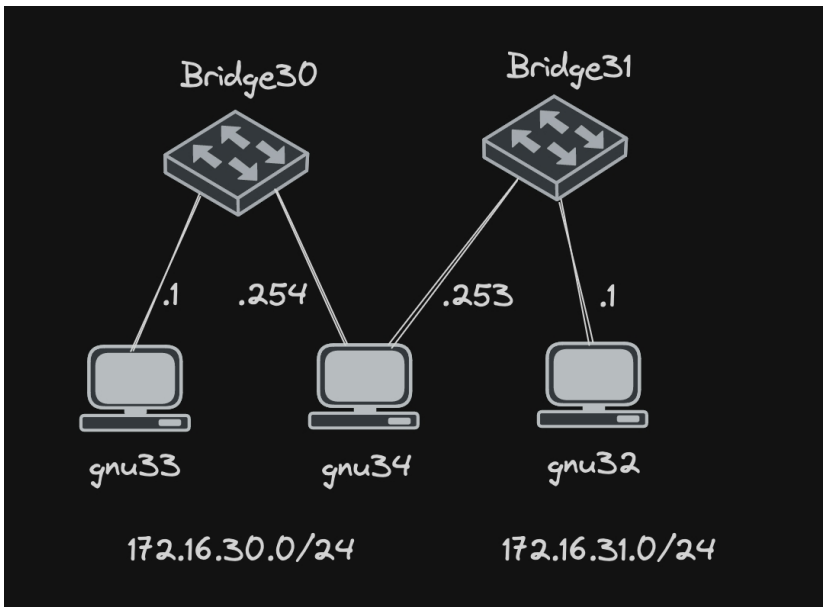
```
# gnu33
ping 172.13.30.254     # para gnu34, funciona
ping 172.13.31.1       # para gnu32, "unreacheable network"
```

Ao executar o comando responsável por dar *ping* a todos os dispositivos associados a uma *broadcast* concluímos que os dispoitivos associados à `bridge30` estão efetivamente isolados dos dispositivos associados à`bridge31`. Isto verifica-se analisando os ficheiros, porque o comando `ping -b 172.16.30.255` executado no gnu33 gera frames do gnu34 para o gnu33 mas não gera nenhum tipo de sinal para o gnu32. Já o comando `ping -b 172.13.31.255` executado no `gnu32` gera sinais no mesmo (sozinho na sua *broadcast*) mas não tem qualquer interferência com os `gnu33` e `gnu34`.

Verificamos assim que se configuraram 2 sub-redes diferentes.

Experiência 3 - Configurar um *router* em Linux

Arquitetura da rede



Partindo da configuração anterior, foi conectado o gnu34 à `Bridge31`. No final existe um rede de duas subnetworks com um dispositivo conectado nas duas.

Objetivos da Experiência

Esta experiência tem como objetivo a configuração de um dispositivo em duas redes de tal forma que seja possível a troca de informação entre as duas redes. No final, será possível através do `gnu33` chegar ao `gnu32` através do `gnu34`.

Principais comandos de configuração

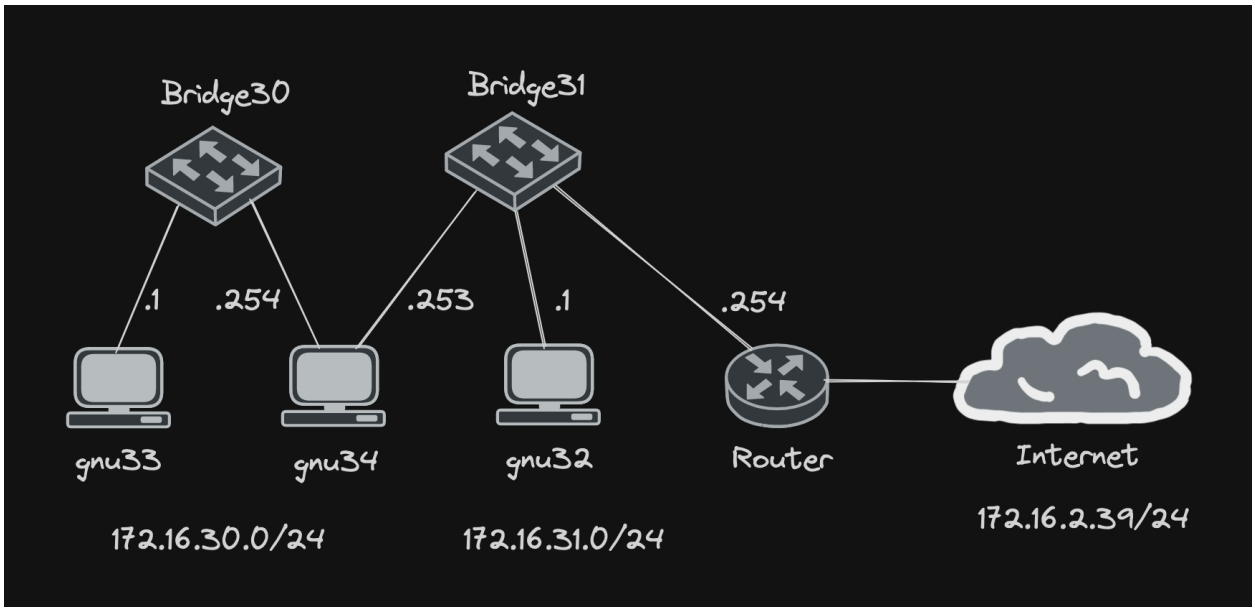
```
route add -net 172.16.30.0/24 gw 172.16.31.253 #gnu32
route add -net 172.16.31.0/24 gw 172.16.30.254 #gnu33

sysctl net.ipv4.ip_forward=1 #gnu34
sysctl net.ipv4.icmp_echo_ignore_broadcasts=0 #gnu34
```

Análise dos logs

Após configurar o `gnu34` para que esteja incluído em ambas as redes dos `gnu32` e `gnu33`, verificamos que estes conseguem enviar sinais *ping* de um para o outro e, portanto, a configuração foi bem sucedida. Verificamos também que o `gnu34` está configurado em ambas as máquinas como *default gateway* para endereços de redes que não pertencem à sua. Os pacotes ARP e ICMP, computados no `gnu34`, contêm o endereço IP da máquina de destino mas o endereço MAC do `gnu34`, uma vez que este trata do redirecionamento da informação entre as duas redes criadas. As tabelas de encaminhamento geradas através da criação das rotas garante que por cada IP de destino, existe outro endereço IP (*gateway*) para onde a máquina de origem deve reencaminhar a informação.

Experiência 4 - Configurar um *router* comercial implementando NAT

Arquitetura da rede



Na arquitetura anterior adiciona-se um novo dispositivo, `Router` na subrede `bridge31`. Este *router* está conectado à internet do laboratório.

Objetivos da Experiência

Esta experiência tem como objetivo a adição de conexão à internet na rede configurada anteriormente, através de um *router* comercial.

Principais comandos de configuração

```
route add default gw 172.16.31.254 #gnu32
route add default gw 172.16.30.254 #gnu33
route add default gw 172.16.31.254 #gnu34

# Router
/ip route add dst-address=172.16.30.0/24 gateway=172.16.31.253
/ip route add dst-address=0.0.0.0/0 gateway=172.16.2.254
```

Análise dos logs

Testámos a comunicação do `gnu32` para o `gnu33` em duas situações diferentes: sem e com o `gnu34` como intermediário. Na primeira situação, tendo desativado os redirecionamentos ICMP, a comunicação é feita pela rota *default*, sendo os pacotes reencaminhados pelo *router* configurado na `bridge31`. Assim, o seu percurso é: `gnu32`→`router`→`gnu34`→`gnu33`. Já na segunda situação, reativando os redirecionamentos ICMP, a comunicação é feita de novo usando o `gnu34` imediatamente como intermediário, sendo portanto mais curta (sem recurso à rota *default*). O percurso é então:`gnu32`→`gnu34`→`gnu33`.

Depois tentámos enviar pacotes *ping* do `gnu33` para o `router`, sendo apenas possível se este tiver ativado o NAT.

> O NAT (*Network Adress Translation*) é um processo de mapeamento de endereços IP privados em endereços IP públicos, permitindo que dispositivos de uma LAN se liguem à rede externa sem revelar o seu endereço privado. A comunicação entre a rede interna e a rede externa é sempre feita usando essa interface entre endereços públicos e privados.

Se um dispositivo da rede privada enviar um pedido para a Internet, este é feito com o endereço público mapeado no *router* por meio do NAT e a reposta terá como destino o mesmo endereço, que depois é encaminhado para o endereço privado mapeado.

Como NAT ativado, verificamos então que existe acesso à internet na rede que temos vindo a configurar.

Experiência 5 - DNS

Arquitetura da rede

A arquitetura mantem-se igual à da experiência anterior.

Objetivos da Experiência

Esta experiência tem como objetivo a configuração do DNS (Domain Name Service) nos dispositivos `gnu32`, `gnu33` e `gnu34`. Espera-se que depois da configuração do DNS, através da adição de um sistema de DNS, seja possível conhecer os IPs de determinados domínios.

Principais comandos de configuração

```
echo 'nameserver 193.136.28.10' > /etc/resolv.conf #gnu 32, 33 e 34
```

Análise dos logs

Durante a experiência analisamos os logs resultantes de alguns pings a domínios específicos, como por exemplo `ftp.up.pt`. Foram observados os pacotes responsáveis pela pesquisa de DNS. Isto acontece pois é necessário transcrever o domínio `ftp.up.pt` em um IP de modo a transmitir os pacotes para o destino pretendido.

Experiência 6 - Conexões TCP

Arquitetura da rede

A arquitetura mantem-se igual à da experiência anterior.

Objetivos da Experiência

Esta experiência tem como objetivo a transferência singular ou simultânea de ficheiros através do protocolo FTP a partir de diferentes dispositivos. Será necessário usar a aplicação cliente FTP referida na primeira parte do relatório.

Principais comandos

```
make
make run
```

Análise dos logs

Para realizar a transferência de um ficheiro, são abertas duas ligações TCP: uma ligação de controlo e outra para proceder à efetiva receção do ficheiro. Uma ligação começa com uma procura DNS que visa encontrar o endereço IP do servidor associado ao nome indicado (por meio de pacotes `DNS`). Depois é estabelecido o TCP *handshake* (com pacotes `SYN-ACK`), de forma a indicar que os servidores estão prontos para se comunicar e podem começar a fazê-lo (usando pacotes `DATA` para a receção de tramas de informação). O protocolo TCP usa o mecanismo ARQ (*Automatic Repeat Request*) para monitorizar a correta receção dos pacotes (por meio de mensagens `ACK`) e eventuais falhas no envio dos mesmos e necessidade de retransmissão (por meio de *timeouts*). Pacotes TCP têm dois campos importantes: `"Sequence number"` e `"Sequence number (raw)"`. Estes campos são importantes para controlar a receção de respostas e possíveis falhas no envio de pacotes. Os campos `"Acnowledgement number"` e `"Ackownledgement number (raw)"` indicam de que forma um pacote foi aceite. Um pacote não ser corretamente recebido depois de 3 tentativas de envio é um indício de que a rede está congestionada e a conexão sofre uma redução do número de pacotes transferidos - de modo a reduzir o efeito de congestionamente da rede. Devido ao mecanismo de controlo da congestão, verifica-se que o fluxo de transferência é consideravelmente inferior quando existe uma segunda conexão TCP.

Conclusões

Neste projeto fizemos o desenvolvimento de uma aplicação de download baseada no protocolo FTP e testamos numa rede configurada e estudada durante as aulas práticas.

A primeira parte focou-se no desenvolvimento de um programa em linguagem C capaz de realizar o download de um ficheiro por meio do protocolo FTP, seguindo as especificações do RFC959. A arquitetura do programa foi estruturada para analisar URLs, conectar-se e autenticar com o servidor remoto, entrar no modo passivo e iniciar a transferência do arquivo, concluindo com a escrita do arquivo no disco local.

Na segunda parte, exploramos a configuração e o estudo das redes por meio de diversas experiências. Desde a configuração de redes IP, a análise do funcionamento do protocolo ARP, a implementação de bridges em um switch, a configuração de routers, a configuração do serviço DNS, até a análise das conexões TCP executadas na transferência de ficheiros por FTP.

Conseguimos explorar os conceitos fundamentais das redes de computadores através da análise de vários logs com a ajuda da ferramenta Wireshark.

Em conclusão, este projeto foi fundamental para consolidar o conhecimento teórico adquirido em aulas práticas, proporcionando uma visão prática e detalhada do funcionamento dos componentes em uma rede de computadores.

Referências

RFC959 RFC1738

Anexos

O código da applicação cliente FTP que faz transferência de ficheiros encontra-se em `main.c`, `include/` e `src/`.

Os logs capturados encontram-se divididos por experiências na pasta `logs`.

Foram sequencialmente listados todos os comandos necessários para a configuração da rede de computadores final na pasta `lab-steup`.