Faculdade de Engenharia da Universidade do Porto

2º Trabalho Laboratorial

Rede de Computadores

Licenciatura em Engenharia Informática e Computação

Turma 5 - Grupo 8

Ana Rita Baptista de Oliveira - up202004155

Diogo Alexandre da Costa Melo Moreira da Fonte - up202004175

Índice

1. Sumário

Este relatório foi realizado no âmbito da UC Redes de Computadores do 3º ano da LEIC da FEUP. O relatório é relativo ao 2º trabalho laboratorial, que tem como objetivo o desenvolvimento de uma aplicação de download e a configuração e estudo de uma rede de computadores. O local de realização das experiências foi a bancada 3 do laboratório I320 do DEI da FEUP.

Concluímos com sucesso a implementação da aplicação de download na linguagem C e a própria configuração da rede no laboratório. Como foi possível observar na demonstração, os downloads foram executados sem qualquer tipo de problema e os ficheiros foram descarregados totalmente e sem perdas, ou seja, a aplicação é robusta à ocorrência de erros e inputs inválidos.

2. Introdução

Neste trabalho laboratorial executamos as 6 experiências presentes no guião dado para documentar a aprendizagem da configuração de uma rede com um tux que conecta/conhece duas redes e faz a ligação entre estas e um router que disponibiliza o acesso à Internet.

Neste relatório abordamos as duas partes do trabalho. A primeira parte aborda o desenvolvimento de uma aplicação de download por FTP. A segunda incide nas 6 experiências, em que cada uma representa uma etapa da configuração da rede. Vamos responder às perguntas do guião, tendo em conta os resultados e os logs capturados no laboratório, durante as experiências.

3. Parte 1 - Aplicação de download

Nesta secção vai ser explicada a arquitetura da aplicação de download de ficheiros e os resultados que obtivemos com ela. A aplicação utiliza o protocolo FTP (File Transfer Protocol) descrito no RFC959 e possui ligações TCP (Transmission Control Protocol) a partir de sockets. O tipo de input esperado pela aplicação está de acordo com a sintaxe url descrita no RCF1738 e é o seguinte:

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

3.1. Arquitetura

```
struct connection{
    char* user;
    char* password;
    char* host;
    char* urlPath;
    char* filename;
    int annonymous;
    int ctrlfd;
    int datafd;
};

void initConnection (struct connection *connection);
int parseInput(struct connection *connection, char* input);
int getIP(char* ip, char* host);
int createConnection(char* addr, int port);
int checkResponse(int sockfd, char * expectedResponse, char* response);
int login(int sockfd, char * username, char * password, int annonymous);
int sendCommand(int sockfd, char * command, char* args, int hasArg);
int enterPassiveMode(int sockfd, struct connection *connection);
int transfer(int sockfd, char * name);
```

Possuímos uma struct para armazenar os dados da ligação para download. Assim conseguimos aceder aos elementos que pretendemos de forma mais simples.

A função **initConnection()** apenas serve para inicializar os atributos da estrutura e na função **parseInput()** procedemos ao parsing do input, ou seja, processamos o url que é passado no terminal ao correr a aplicação e guardamos os dados nos respetivos atributos da estrutura.

Posto isto, fazemos uma chamada à função **getIP()**, que foi fornecida pelos professores, e passamos o nome do host e um array para guardar o IP. Esta função retorna 0 se bem sucedida e -1 caso contrário e de seguida o endereço IP é enviado, juntamente com a porta host, na chamada à função **createConnection()** para fazer a ligação ao socket.

Com o resultado da função anterior (resposta do socket ou -1 em caso de erro),

executamos uma chamada a **checkResponse()** e verificamos se é a que pretendemos para podermos avançar no processo. Esta função é chamada várias vezes ao longo do programa para verificar as respostas do servidor. Assim, sempre que é recebida uma resposta do socket, assume-se que é verificada através deste processo.

A seguir é executado o login com a função **login()**, em que são executados os comandos USER e PASS com a função a seguir descrita. Com **enterPassiveMode()**, enviasse o comando PASV, entra-se em modo passivo para executar o download e a função retorna a porta para abrir o socket de dados para receber o ficheiro. Nesta situação o cliente fica responsável por abrir a ligação TCP para os dados. A função **sendCommand()** serve para enviar comandos e receber respostas do socket, ou seja, comandos para interagir com o sistema, tal como pasv para entrar em modo passivo e help para pedir ajuda sobre comandos possíveis de executar.

Pedimos o ficheiro com o comando RETR e com a chamada a **transfer()** executamos a transferência do ficheiro pretendido e caso não tenha sido possível é retornado -1. No final do programa fechamos a ligação ao servidor e libertamos as estruturas de dados utilizadas que já não são necessárias.

3.2. Resultados

Como foi possível observar na demonstração, implementamos com sucesso a aplicação pretendida. As transferências foram executadas sem qualquer tipo de problema e os ficheiros foram descarregados a 100%. A aplicação é robusta à ocorrência de erros e inputs inválidos.

4. Parte 2 - Configuração e Estudo de uma Rede

4.1. Experiência 1

Objetivos

O objetivo desta experiência foi configurar duas máquinas,neste caso o tux33 e o tux34, incluindo a atribuição dos endereços de IP das mesmas, e depois ligá-las através de uma rede de computadores de forma a ser possível enviar mensagens entre elas.

Comandos Relevantes

- Para configurar os IPs do tux33 e do tux34, respetivamente:
    - ifconfig eth0 172.16.30.1/24
    - ifconfig eth0 172.16.30.254/24

Questões

1. O que são os pacotes ARP e para que servem?

Address Resolution Protocol é usado para obter os endereços MAC associados a um certo endereço IP. Na experiência 1 foi possível verificar este protocolo em ação quando no tux33 apagamos as entradas da tabela ARP e depois tentamos dar ping ao tux34. Ao tentarmos dar ping ao tux34 através do tux33, como não há nenhuma entrada para o IP do tux34, é enviado um pacote ARP, por broadcast, que vai para todas as máquinas da rede local a perguntar qual o endereço MAC de quem têm o IP do destinatário. O destinatário, neste caso o tux34, envia outro pacote ARP à máquina que emitiu o pacote ARP inicial, o tux33, a indicar o seu endereço MAC. Após esta interação, o tux 33 irá guardar na sua tabela ARP a informação do endereço MAC para o IP do destinatário e a transferência de pacotes é efetuada.

2. O que são endereços MAC e IP de pacotes ARP (e porquê)?

Quando o tux33 tenta enviar uma mensagem ao tux34 e não há a entrada na tabela ARP, será necessário efetuar o processo mencionado na questão anterior.

O pacote ARP enviado por broadcast do tux33 tem os seus endereços de IP e MAC. Também contém o endereço de IP do destinatário, porém, o endereço MAC do destinatário é deixado em branco, 00:00:00:00:00:00, pois este não é conhecido.

O pacote ARP enviado de resposta pelo tux34 tem os endereços IP e MAC do destinatário, o tux33, como também os endereços IP e MAC dele próprio.

3. Que pacotes são gerados pelo comando ping?

O comando ping gera pacotes do tipo ICMP (*Internet Control Message Protocol*).

4. O que são endereços MAC e IP de pacotes ping?

Quando é efetuado um ping, do tux33 para o tux34, por exemplo, os pacotes enviados têm os endereços IP e MAC relativos das máquinas de origem, tux33, e de destino, tux34. Os pacotes de resposta a este ping têm os endereços IP e MAC também de origem, agora o tux34, e destino, tux33, isto é, invertidos.

5. Como determinar se uma trama Ethernet é ARP, IP ou ICMP?

Durante as experiências utilizamos o Wireshark para verificar isso, mas isto é possível ver através de um dos octetos no campo EtherType no header trama Ethernet. Caso o valor seja 0x0806, trata-se de um ARP, e se for 0x0800, trata-se de uma trama IPv4. Para verificar se é uma trama ICMP, o valor no header da trama Ethernet tem de ser 0x0800 e o número de protocolo no header do IP tem de ser 1.

6. Como determinar o tamanho de uma trama recebida?

O outro octeto no EtherType do header da trama Ethernet permite-nos descobrir isto. Caso o valor seja igual ou inferior a 1500 (0x05dc), o valor indica o tamanho da payload.

7. O que é a interface loopback e por que razão é importante?

Uma interface de loopback é uma interface de rede virtual que permite que um cliente e um servidor no mesmo host se comunicarem entre si usando a stack de protocolos TCP/IP. Se um pacote com destino a uma interface de loopback for recebido numa interface, sem que tenha sido originado no mesmo host, deve ser descartado, pois pode se tratar de um pacote malicioso. Isto é importante para testar a configuração e também para *troubleshooting*.

4.2. Experiência 2

Objetivos

O objetivo desta experiência foi criar duas bridges no switch, configurá-las e testar a troca de mensagens entre máquinas através delas. Especificamente ligar o tux33 e tux34 à bridge30 e o tux32 à bridge31.

Comandos Relevantes

- Para dar reset ao Mikrotik switch:
    - /system reset-configuration
- Para configurar o tux32:
    - ifconfig eth0 172.16.31.1/24
- Para criar e configurar as bridges: (mencionado na questão 1 para a bridge30, semelhante para bridge31, para mais detalhe ver o anexo …)

Questões

1. Como configurar bridgeY0?

Antes de iniciar a configuração, precisamos de ligar um cabo do RS232 -> Cisco à entrada do S0 de um dos tux, neste caso, do tux4, e um cabo do Cisco -> RS232 à entrada Console do switch. Agora no GTKTerm, após colocar a Baudrate a 115200, podemos fazer reset do switch e inserir os comandos abaixo mencionados.

Para configurar a bridge30, primeiro temos de a criar com o comando **/interface bridge add name=bridge30**. Antes de adicionarmos as portas 3 e 4 à bridge30, precisamos de as remover com o comando **/interface bridge port remove [find interface = etherX]**, em que **X** é o número da porta. Estes números poderiam ser outros dependendo das portas escolhidas para ligar os cabos. Depois de removermos, precisamos de adicionar as portas mencionadas à bridge30 com o comando **/interface bridge port add bridge=bridge30 interface=etherX**.

2. Quantos broadcast domains existem? Como concluir isso a partir dos logs?

Existem 2 broadcast domains. Quando fazemos ping com a flag **-b** de broadcast para

172.16.30.255 através do tux33, podemos verificar nos logs do Wireshark que o ping só alcança o tux34. Isto deve-se a existir outro broadcast domain para a bridge31 em que o tux32 se encontra.

4.3. Experiência 3

Objetivos

O objetivo desta experiência foi configurar o tux34 de modo a funcionar como um router ligado às duas bridges, permitindo a comunicação entre máquinas em diferentes bridges.

Comandos Relevantes

- Para configurar o tux34:
    - ifconfig eth1 172.16.31.253/24
    - Enable IP Forwarding:
        - echo 1 > /proc/sys/net/ipv4/ip_forward
    - Disable Ignore Broadcast:
        - echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
- Para adicionar routes do tux33 para o tux32 e vice-versa, respetivamente:
    - route add -net 172.16.31.0/24 gw 172.16.30.254
    - route add -net 172.16.30.0/24 gw 172.16.31.253

Questões

1. Que routes existem nos tuxes? Quais são os seus significados?

- tux32:
    - Route para a própria sub rede (Destino: 172.16.31.0; Gateway: 0.0.0.0) que permite comunicar com qualquer máquina nessa subrede.
    - Route para a sub rede 172.16.30.0 através da gateway 172.16.31.253 que permite comunicar com máquinas nessa sub rede (tux33), através do tux34 que foi configurado para funcionar como um router.
- tux33:
    - Route para a própria subrede (Destino: 172.16.30.0; Gateway: 0.0.0.0) que permite comunicar com qualquer máquina nessa subrede.
    - Route para a sub rede 172.16.31.0 através da gateway 172.16.30.254 que permite comunicar com máquinas nessa sub rede (tux32), através do tux34.
- tux34:
    - Route para a sub rede 172.16.31.0 através da gateway 172.16.30.254 que

permite comunicar com máquinas nessa sub rede.
    - Route para a sub rede 172.16.30.0 através da gateway 172.16.31.253 que permite comunicar com máquinas nessa sub rede.

2. **Que informação contém uma entrada da forwarding table?**
    - **Destination** - IP da rede de destino
    - **Gateway** - IP da máquina para onde serão enviados os pacotes para serem encaminhados para o destino
    - **Interface** - Placa de rede usada para comunicar com a gateway
    - Outros campos menos relevantes para as experiências como **Genmask**, **Flags**, **Metric**, **Ref**, **Use**.

3. **Que mensagens ARP, e endereços MAC associados, são observadas e porquê?**

Para o caso de captura de logs no tux34 quando se dá ping do tux32 no tux33, no eth0, o tux33 envia por broadcast um pacote ARP, com endereço MAC de origem do tux33 e endereço MAC de destino 00:00:00:00:00, a pedir o endereço MAC do tux34. O tux34 envia o pacote ARP de resposta,com endereço MAC de origem do tux34 e de destino do tux33, para o tux33. Após isto, o ping inicia.

Já no eth1, o tux34 envia por broadcast um pacote ARP, com endereço MAC de origem do tux34 e endereço MAC de destino 00:00:00:00:00, a pedir o endereço MAC do tux32. O tux32 envia o pacote ARP de resposta,com endereço MAC de origem do tux32 e de destino do tux34, para o tux33.

4. **Que pacotes ICMP são observados e porquê?**

É possível verificar pacotes ICMP do tipo *request* e *reply*. Os pacotes de *request* tem origem no tux33, a máquina que realiza o ping, e destino no tux32, a máquina que está ser *pinged*. Os de *reply* são o inverso, com origem no tux32 e destino no tux33. Essencialmente, o comando ping faz o tux33 mandar pacotes ICMP para o destino, tux32, e o tux32 envia pacotes ICMP a indicar se está ou não disponível.

5. **O que são os endereços IP e MAC associados aos pacotes ICMP e porquê?**

Para pacotes ICMP de *request*, os endereços IP e MAC de origem são os do tux33 e os de destino são os do tux32. Já para os pacotes ICMP de *reply*, os endereços IP e MAC de origem são os do tux32 e os de destino são os do tux33.

4.4. Experiência 4

Objetivos

O objetivo desta experiência foi configurar o router RC e adicioná-lo na rede, conectando ether1 à rede do laboratório em P3.1 (no nosso caso) e ether2 à bridge31. Foi também configurado de forma a utilizar NAT. Assim é possível ter acesso à Internet na rede local criada.

Comandos Relevantes

- Para configurar os endereços IP do router no GKTerm:
  - /ip address add address=172.16.2.39/24 interface=ether1
  - /ip address add address=172.16.31.254/24 interface=ether2
- Para adicionar tux34 como default route para do tux33:
  - route add default gw 172.16.30.254
- Para adicionar Rc como default route para o tux34 e tux32:
  - route add default gw 172.16.31.254
- Para adicionar a route no Rc para 172.16.30.0/24:
  - /ip route add dst-address=172.16.30.0/24 gateway=172.16.31.253
- echo 0 > /proc/sys/net/ipv4/conf/eth0/accept_redirects
- echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects

Questões

1. Como configurar uma route estática num router comercial?

Uma route estática é uma route que é inserida na routing table manualmente. Para fazê-lo é necessário primeiro ligar o cabo que se encontrava na entrada "*Console*" do switch e ligar à entrada "*Router MTik*". Depois disso, para configurar uma route estática no RC usa se o comando acima referido, sendo que neste exemplo estamos a adicionar uma route que vai para a rede 172.16.30.0/24 através da gateway 172.16.31.253 (tux34).

2. Quais são os caminhos percorridos pelos pacotes nas experiências realizadas e porquê?

No passo 4 desta experiência, quando desativamos os redirects no tux32 e apagamos a route para a rede 172.16.30.0/24 via o tux34, o tux32 vai ter de enviar os pacotes primeiro para o router e depois o router envia para a rede 172.16.30.0/24. Isto acontece porque o router é a route default do tux32. Os pacotes seguintes irão seguir esta rota.

No entanto, quando ativamos os redirects, depois da primeira vez que o caminho anteriormente descrito ocorre, é enviado um pacote ICMP de redirect que serve para adicionar na forwarding table do tux32 uma entrada que diz que para chegar ao destino pretendido, pode usar a gateway 172.16.31.253 (tux34).

3. **Como configurar NAT num router comercial?**

Para configurar o NAT é necessário ativá-lo (nas experiências estava ativado por default). Para funcionar como esperado na experiência é também necessário definir o router RC como default router para o tux32 e tux34 e adicionar a route do RC para a rede 172.16.30.0/24.

4. **Qual o objetivo da NAT?**

A NAT é a tradução de um endereço privado num endereço público e vice-versa, e o seu objectivo é poupar o espaço de endereçamento público, usando endereços IP privados.

4.5. Experiência 5

Objetivos

O objetivo desta experiência é configurar o servidor DNS na nossa rede de computadores.

Comandos Relevantes

- Para editar o ficheiro pretendido:
    - sudo nano /etc/resolv.conf

Questões

1. Como configurar o serviço DNS num host?

Para configurar o serviço DNS nos computadores tux32, tux33 e tux34 necessitamos de limpar o conteúdo do ficheiro /etc/resolv.conf e inserir o nome do IP do servidor DNS. O conteúdo presente no ficheiro referido deve ser o seguinte: “nameserver 172.16.2.1”, no nosso caso.

2. Que pacotes são trocados pelo DNS e que informação é transportada?

O host envia um pacote com o hostname, para receber o endereço IP. O servidor DNS envia um pacote de resposta com o IP do hostname pedido.

4.6. Experiência 6

Objetivos

O objetivo desta experiência foi observar o funcionamento do protocolo TCP em conjunto com a aplicação de download criada de forma a testar a rede configurada.

Comandos Relevantes

- Para testar a aplicação de download
    - make clean && make
    - ./download ftp://ftp.up.pt/pub/kodi/timestamp.txt
    - ./download ftp://rcom:rcom@netlab1.fe.up.pt/files/crab.mp4

Questões

1. **Quantas conexões TCP são abertas pela aplicação ftp?**

A aplicação abre duas conexões TCP: a primeira ligação serve para controlo e para enviar e receber os comandos e respostas necessários para iniciar o download; a segunda serve para realizar a transferência do ficheiro.

2. **Em que conexão é transportada a informação de controlo FTP?**

A informação relativa ao controlo é passada pela primeira conexão referida anteriormente, isto é, na ligação de controlo.

3. **Quais são as fases de uma conexão TCP?**

As três fases de uma conexão TCP são o estabelecimento da ligação, a transferência de dados e a terminação da ligação.

4. **Como funciona o mecanismo ARQ TCP? Quais os campos relevantes do TCP?**

O mecanismo ARQ TCP (Automatic Repeat Request) serve para quando há algum erro na transmissão de pacotes, seja por ser um pacote em falta, um pacote com erros ou existir um pedido automático de retransmissão desse pacote. Os campos relevantes são os seguintes: *Sequence Number*, *Acknowledgement Number* e *Window Size*.

5. **Como funciona o mecanismo de controlo de congestionamento do TCP funciona? Quais os campos relevantes? Como é que o *throughput* da conexão de dados evoluiu ao longo do tempo? Está de acordo com o mecanismo de controlo do TCP?**

Este mecanismo tem como base os acknowledgments recebidos na transmissão de dados e é responsável por controlar a janela de congestionamento para calcular o tamanho máximo do buffer do emissor. A janela é incrementada em 1 se a congestão da rede diminuir e decrementada em 1 se aumentar, a cada RTT - Round Trip Time. Os algoritmos mais conhecidos para gerir a variação da janela são os seguintes: Increase/Multiplicative Decrease, Slow Start e Congestion Avoidance. O throughput da conexão aumentou linearmente até estabilizar, depois diminui linearmente até ao fecho da conexão. Os resultados estão de acordo com o mecanismo referido porque o throughput manteve-se estável durante a transferência de dados, isto porque existia apenas um download a ser executado na rede.

6. **O *throughput* de uma conexão de dados TCP é perturbada pelo aparecimento de uma segunda conexão TCP? De que forma?**

Sim, será afetado. A taxa de transferência terá que ser distribuída de forma igual às duas conexões, daí a diminuição da taxa de transferência da conexão TCP iniciada em primeiro lugar.

5. Conclusões

Este projeto consistiu no desenvolvimento de uma aplicação para transferência de ficheiros, utilizando os protocolos FTP e TCP, e na configuração de uma rede de

computadores para aprendermos o funcionamento de uma rede e como utilizar aparelhos como o switch e o router. Também aprendemos técnicas, como NAT e DNS, protocolos, como ARP e ICMP e estruturas de dados, por exemplo ARP tables e forwarding tables, que são fundamentais para a comunicação entre todas as máquinas utilizadas na rede configurada. Conseguimos atingir os objetivos propostos para este trabalho e aprender com as respostas às perguntas de cada experiência do guião do trabalho laboratorial.

6. Referências

Para desenvolver este trabalho laboratorial fizemos uso do guião disponibilizado pelos professores e dos slides apresentados nas aulas teóricas da unidade curricular.

7. Anexos

Incluímos no zip o código desenvolvido para a aplicação, um ficheiro com os comandos executados e um diagrama que representa a configuração final de cabos na bancada. Os logs estão disponíveis no repositório do gitlab, com uma tag que marca esta entrega.