Faculdade de Engenharia da Universidade do Porto
Mestrado Integrado em Engenharia Informática e Computação
Unidade Curricular: Redes de Computadores
Docente: Manuel Alberto Pereira Ricardo

Redes de Computadores

António Alexandre de Almeida Martins, *201404422*
Filipe Oliveira e Sousa Ferreira de Lemos, *201200689*
Frederico Portugal Pinho Rocha, *201408030*

22 de Dezembro de 2017

Conteúdo

1 Sumário **3**

2 Introdução **3**

3 Part 1 – Desenvolvimento da aplicação para download **3**

4 Part 2 – Configuração de uma rede de computadores **5**

4.1 1ª Experiência - Configuração de uma rede IP 5

4.2 2ª Experiência - Implementação de duas LAN's virtuais num switch 6

4.3 3ª Experiência - Configurar um router em LINUX 7

4.4 4ª Experiência - Configuração de um router comercial e implementação do NAT 9

4.5 5ª Experiência - Configuração de um servidor DNS 10

4.6 6ª Experiência - Estabelecimento de conexão TCP 11

5 Conclusões **12**

1 SUMÁRIO

Este relatório serve para apresentar o desenvolvimento de uma aplicação e a configuração de uma rede de computadores, enquadrado da unidade curricular de Redes de Computadores. Uma das principais conclusões deste trabalho é que dois computadores que não estejam na mesma Local Area Network(LAN), não são capazes de comunicar entre si. Para tal resolver este problema, é necessário um router que ligue as duas LAN's entre si.

As experiências laboratoriais foram feitas com recurso ao programa Wireshark, em ambiente Linux. A aplicação foi desenvolvida na linguagem de programação C.

2 INTRODUÇÃO

Este trabalho laboratorial é constituído por duas vertentes complementares que serão analisadas sem qualquer liga- ção entre as duas. A primeira parte diz respeito ao desenvolvimento em detalha da aplicação para download de ficheiro utilizando o protocolo FTP. A segunda parte é a construção e estudo de uma rede de computadores, desde os IP's dos computadores que fazem parte de rede, das VLAN's em que se dividem os computadores e o router que irá ligar as VLAN's. Para ajudar, são utilizados logs de Wireshark, capturados no laboratório em conformidade com o guião.

3 PART 1 – DESENVOLVIMENTO DA APLICAÇÃO PARA DOWNLOAD

Nesta parte inicial do trabalho, foi elaborada uma aplicação em C para download de um ficheiro, implementando o protocolo FTP descrito no RFC959, com sintaxe descrita no RFC1738.

O código está divido em quatro fases:

- Estabelecimento da ligação - converte o nome do servidor no seu endereço IP e abre a socket que irá realizar a ligação;
- Autenticação, processo de autenticação pedido pelo servidor. O username e password são os primeiros parâmetros inseridos no URL, seguindo a sintaxe proveniente do RFC1738;

- Transferência de dados - é aberta uma socket para a transferência onde são lidos os dados do servidor e escritos no ficheiro de destino;
- Fim de ligação - as sockets de controle e transferência de dados são fechadas;

Tabela 3.1: Funções

Função	**Descrição**
int computePortNumber(int socket fd)	Obtem a porta que será usada para a transferência de dados
char * getHostIp (char * url)	Descobre o IP do endereço da web passado por parâmetro
int connect_ftp (char * url , int porta)	Estabelece uma nova ligação FTP
void sendCommand(int port , char *cmd)	Envia um comando (passado por parâmetro) ao servidor
int saveFile (int sock , char * path , int sockControl)	Guarda o ficheiro com o conteúdo do download
int verifyCode(int code)	Verifica o código FTP recebido. Se o código não estiver dentro dos previstos para esta aplicação, significa que ocorreu um erro ftp e o programa deve terminar.
void printAnswer(int fd)	Imprime a mensagem do servidor. É nesta função que é chamada a função verifiyCode().
int getParameter(char limit,char * argv, char * res,int i)	Serve para retirar os diferentes parâmetros (username, password, host, path) do url inserido.
int endReached(char * rbuffer)	Verifica se é o últmo pacote da resposta ao comando enviado.

A tabela apresentada em cima descreve as funções utilizadas no desenvolvimento do código.

4 Part 2 – Configuração de uma rede de computadores

4.1 1ª Experiência - Configuração de uma rede IP

Nesta experiência, foram configurados o tux1 e o tux4 com os IP's pedidos, seguidamente foram ligados ao switch sem se criar VLANs. Como originalmente todas as portas do switch estão ligadas na mesma VLAN foi possível determinar que os dois computadores conseguem comunicar. Cada interface de rede possui um endereço IP e um MAC que a caracterizam, sendo que o MAC é o endereço que caracteriza o hardware da interface, ou seja, é único a cada interface, e o IP é usado como uma etiqueta numérica que identifica cada dispositivo que pertence à rede de computadores.

Para haver conexão os computadores precisam de saber os endereços MAC do destinatário, para isso foram utilizados pacotes ARP que perguntam à rede qual é o MAC que corresponde ao IP a conectar. Para testar a conexão é utilizado o comando ping, este gera pacotes Internet Control Message Protocol (ICMP), que envia ICMP request e espera por um ICMP echo reply. Estes pacotes contem os endereços MAC e os endereços IP dos dois computadores, o destinatário e o de origem. Para determinar o tipo de pacote de Ethernet recebido, é lido o cabeçalho Ethernet desse mesmo pacote. Um pacote tipo IP terá um cabeçalho 0x0800 enquanto um pacote tipo ARP terá um cabeçalho 0x0806. A distinção entre o tipo de pacote IP é feita através do cabeçalho IP, por exemplo, o pacote ICMP terá um cabeçalho de valor 1 enquanto que um TCP seria de valor 6.

O tamanho destes pacotes é determinado por um campo no cabeçalho do pacote Ethernet. Este indica o tamanho do pacote em octetos e não pode exceder 1500B. Antes de podermos fazer comunicação podemos testar a placa de rede, para isso existe uma interface própria, o Loopback, que é o próprio computador. Em seguida usando o comando ipconfig podemos configurar o IP para uma placa de rede do computador, como dito anteriormente, para o identificar numa rede, por exemplo para o tux1 usamos:

> ifconfig eth0 172.16.30.1

Depois, para testar a conexão entre os dois Tux's faz-se ping e confirma-se que há conexão, uma vez que existe resposta do recetor. Para verificar os pacotes que estavam a ser criados e a ser recebidos foi recorremos ao uso do Wireshark, este programa captura todos os pacotes que entram numa dada interface de rede. Analisando o log desta primeira experiência gerado pelo Wireshark (figura 4.1) reparamos que recebemos três tipos de pacotes, um STP criado pelo switch, este é usado na criação da Spanning-tree, os pacotes ICMP usados pelos pings e ainda os ARP já explicado anteriormente. No Wireshark é possível também verificar mais coisas, como o comprimento do pacote, o emissor e recetor e o conteúdo dos pacotes.

15 6.014440	Cisco_3a:fa:83	Spanning-tree-(for-…	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cost = 0 Port = 0x8003
16 6.724790	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x3b84, seq=6/1536, ttl=64 (reply in 17)
17 6.725130	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x3b84, seq=6/1536, ttl=64 (request in 16)
18 6.738808	HewlettP_5a:7d:74	G-ProCom_8b:e4:4d	ARP	60	Who has 172.16.30.1? Tell 172.16.30.254
19 6.738821	G-ProCom_8b:e4:4d	HewlettP_5a:7d:74	ARP	42	172.16.30.1 is at 00:0f:fe:8b:e4:4d

Figura 4.1: Experiência 1 - Captura dos pacotes via Wireshark

4.2 2ª Experiência - Implementação de duas LAN's virtuais num switch

Uma rede VLAN é uma rede logicamente independente. Várias VLAN's podem co-existir no mesmo switch de forma a partilhar uma rede física. O objetivo desta experiência foi criar duas LAN's virtuais no switch e criar a ligação entre dois tuxs que se encontram na mesma LAN virtual.

Para a criação da VLAN30 foi feita a ligação de um dos computadores (Tux1, Tux2 ou Tux4) ao switch da Cisco via porta de série. Depois de entrar no switch com as credenciais dadas, foi aberto o modo de edição com o comando "conf t". A seguinte sequencia de comandos cria a Vlan pretendida, sem portas adicionadas ainda:

> vlan 30

> end

Para adicionar uma porta, é necessária a abertura do modo de configuração com o comando "conf t"e a execução dos seguintes comandos para configurar a porta com fast ethernet na porta, neste caso 1:

> interface fastethernet 0/1

> switchport mode access

> switchport access vlan 30

> end

Foi criada a VLAN30 e a VLAN31 e adicionadas as portas necessárias(porta 1 e 3 na VLAN30 e porta 2 na VLAN31, Figura 4.2) Ao ser realizado ping do tux1 para o tux4 recebemos uma resposta, uma vez que se encontram na mesma LAN virtual (Figura 4.3) mas não é possível ping do tux1 para o tux2 pois encontram-se em VLAN's distintas e, a este ponto do trabalho, não existe forma de comunicarem entre si. Isto pode ser visível pelo resultado obtido na linha de comandos (Figura 4.4).

Figura 4.2: Experiência 2 - Correspondência entre cada VLAN e as suas portas

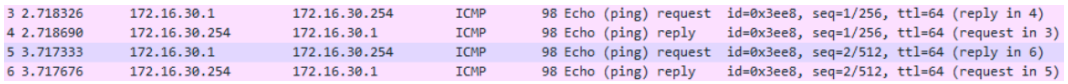

Figura 4.3: Experiência 2 - Teste de comunicação entre tux1 e tux4 (usando Wireshark)

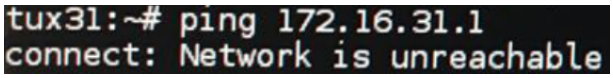

Figura 4.4: Experiência 2 - Teste de comunicação entre tux1 e tux2 (no terminal do tux1)

Existem portanto dois domínios de broadcast. O 172.16.30.255 que pertence à rede 172.16.30.0 (VLAN30) e 172.16.31.255 que pertence à rede 172.16.31.0 (VLAN31). Podemos verificar isto uma vez que apenas é possível a comunicação entre computadores no mesmo domínio.

4.3 3ª Experiência - Configurar um router em LINUX

Esta experiencia tinha como objetivo configurar o tux4 como um router, de modo a permitir a comunicação entre a duas VLANs. Para tal, foi necessário criar uma interface ethernet1 no tux4, com o IP dentro da mesma gama que o tux2, e adicionar a sua porta do switch à

VLAN1. Depois ativamos o IP Forwarding que permite a determinação do path a seguir por um pacote assim como a desactivação do echo-ignore-broadcast através das linhas:

> echo 1 > /proc/sys/net/ipv4/ip_foward

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

De seguida, foram criadas rotas, de modo a permitir a conexão do tux1 com a VLAN1 e o tux2 com a VLAN0. No terminal do tux1, executou-se o comando:

> route add -net 172.16.31.0/24 gw 172.16.30.254

Este cria uma rota entre o tux1 e a VLAN1, usando o eth0 do tux4 como gateway. Fez-se o mesmo no tux2, exceto com o comando:

> route add -net 172.16.30.0/24 gw 172.16.31.253

Foram adicionadas outras rotas para permitir a esta comunicação ficando assim formada a tabela de rotas (Figura 4.5).

Destination	Gateway	Genmask	Iface	Destination	Gateway	Genmask	Iface	Destination	Gateway	Genmask	Iface
172.16.30.0	0.0.0.0	255.255.255.0	eth0	172.16.30.0	172.16.31.253	255.255.255.0	eth0	172.16.30.0	0.0.0.0	255.255.255.0	eth0
172.16.31.0	172.16.30.254	255.255.255.0	eth0	172.16.31.0	0.0.0.0	255.255.255.0	eth0	172.16.31.0	0.0.0.0	255.255.255.0	eth1

Figura 4.5: Experiência 3 - Tabelas de rotas

Finalmente, é possível fazer ping entre os tux1 e tux2, usando o tux4 como router. Com uma análise aos logs é possivel determinar que através de um ping to tux1 para o tux2, o pacote ICMP tem o endereço MAC do tux4 como endereço de destino.

Source	Destination	Protocol	Length	Info
172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) request

Figura 4.6: Experiência 3 - Pacote do tux1, com destino para o tux4 onde será reencaminhado

Source	Destination	Protocol	Length	Info
172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request
172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply

Figura 4.7: Experiência 3 - Pacotes request e reply no tux4, mostrando o reencaminhamento

NOTA: Foram usados endereços IP com formato xxx.xxx.6x.xxx porque os testes foram executados numa bancada diferente da que foi inicialmente utilizada, cujos endereços IP seriam de formato xxx.xxx.3x.xxx.

Na resposta do tux2, vê-se que o endereço de origem do pacote é o endereço MAC do tux4. Concluindo, então, que o pacote passa pelo tux4 com sucesso, como era esperado.

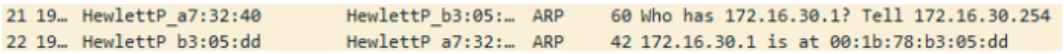

Figura 4.8: Experiência 3 - ARP no Tux1 que procura saber o destino do ping

4.4 4ª Experiência - Configuração de um router comercial e implementação do NAT

O objetivo desta experiência foi configurar um router comercial Rc, com NAT. Este router possibilita a comunicação entre os computadores da rede como redes externas. Para configurar o router, ligamos a "router console"a uma porta de série de um dos tux. De seguida foi preciso criar duas interfaces gigabitethernet (0/0 e 0/1). A interface interna 0/0 tem um IP da mesma gama que o tux2, neste caso 172.16.31.254. A porta desta interface é ligada ao switch. A interface externa 0/1 é configurada com o IP 172.16.1.39, de modo a estar ligado ao router do laboratório. Foram usados os seguintes comandos para efetuar a configuração da interface interna 0/0:

> interface gigabitethernet 0/0

> ip address 172.16.31.254 255.255.255.0

> no shutdown

> ip nat inside (para a interface 0/1 seria usado ip nat outside)

O tux4 é definido como default gateway do tux1, e o router como default gateway do tux2 e do tux4. Deste modo, os pacotes vindos do tux1 são reencaminhados para o router através do tux4.

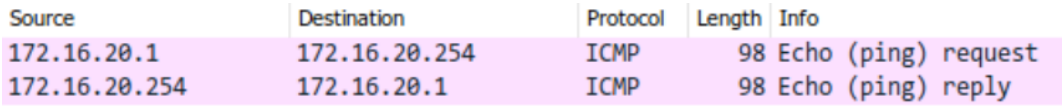

Figura 4.9: Experiência 4 - Logs no tux1, mostrando como o tux4 serve de default gateway e reencaminha os pacotes entre o tux1 e o comercial router

NOTA: Foram usados endereços IP com formato xxx.xxx.2x.xxx porque os testes foram executados numa bancada diferente da que foi inicialmente utilizada, cujos endereços IP seriam de formato xxx.xxx.3x.xxx.

De seguida, implementa-se o NAT, usando os comandos de nat pool e uma access-list.

> ip nat pool ovrld 172.16.1.39 172.16.1.39 prefix 24

> ip nat inside source list 1 pool ovrld overload

> access-list 1 permit 172.16.30.0 0.0.0.7

> access-list 1 permit 172.16.31.0 0.0.0.7

> 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

Esta access-list deve ser dada permissões para cada sub-rede. O Network Address Translation é um método que possibilita a comunicação de uma rede privada de computadores a uma rede publica, usando apenas um IP address.

4.5 5ª Experiência - Configuração de um servidor DNS

Neste ponto do trabalho foi pedido para configurar um servidor DNS. Para tal foi apenas necessário verificar se o ficheiro “resolv.conf” continha a informação:

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

De forma a testar o DNS foi efectuado um ping ao host "www.fe.up.pt"sendo feito um pedido ao servidor DNS da sala como podemos observar na seguinte captura do wireshark:

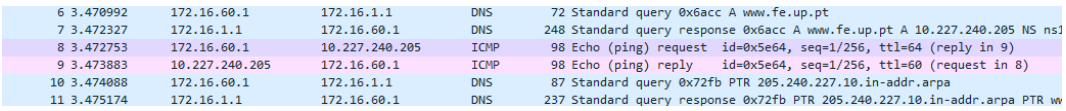

Figura 4.10: Experiência 5 - Captura do Wireshark

NOTA: Foram usados endereços IP com formato xxx.xxx.6x.xxx porque os testes foram executados numa bancada diferente da que foi inicialmente utilizada, cujos endereços IP seriam de formato xxx.xxx.3x.xxx.

Olhando para a figura podemos observar que o tux faz um pedido ao servidor DNS 172.16.1.1. Neste pedido o tux pede o IP deste host e recebe a informação relativo a esta (10.227.240.205). Sabendo agora o IP do host é feito a troca de pacotes ICMP, agora entre o tux e o host.

4.6 6ª Experiência - Estabelecimento de conexão TCP

Durante esta fase do trabalho laboratorial, verificamos que a aplicação FTP abre duas conexões TCP com o servidor: uma de controlo entre o tux1 e o servidor e, outra de dados entre o tux1 e o servidor. Desta forma, a informação de controlo ftp vai na primeira ligação.

Uma conexão TCP tem três fases: o estabelecimento de ligação, a transferência de dados e o encerramento da ligação. Na primeira fase, o cliente envia a flag SYN do pacote TCP, ativa, informando o servidor de que deseja iniciar uma transferência. O servir aceita o pedido de ligação através do envio de um pacote TCP com as flags SYN e ACK ativas, informando o cliente de que está pronto para realizar a transferência. Por fim, o cliente envia um pacote TCP com a flag ACK para finalizar esta fase. Esta fase é também denominada de three-way handshake. Na segunda fase, dá se a transferência de dados. A última fase é semelhante à primeira. O cliente informa o servidor sobre a sua intenção de encerrar a ligação através do envio da flag FIN ativa. Por sua vez, o servidor responde ao cliente através do envio de dois pacotes seguidos, tendo o primeiro a flag ACK ativa e, o segundo, a flag FIN ativa. A ligação termina quando o cliente envia um pacote ACK, confirmando que recebeu a resposta do servidor.

Quanto aos mecanismos de controlo de fluxo, o protocolo TCP utiliza o mecanismo ARQ. Este funciona à base de acknowledgements e timeouts, tendo como objectivo assegurar a robustez da transmissão. Cada pacote tem um número de sequência que permite garantir a sua posição na stream de dados. Uma das suas características importantes é a utilização de uma sliding window, que se adapta continuamente e ajusta o tamanho do buffer, fazendo com que o controlo do fluxo seja mais eficiente. Para este mecanismo em concreto, os campos TCP com mais relevo são o tamanho da sliding window, o número da sequência e do acknowledgement, e também as portas do cliente e do servidor. O campo checksum é também importante pois dele depende a resistência aos erros. Estes campos com mais relevo podem

ser observados neste excerto retirado de um log:

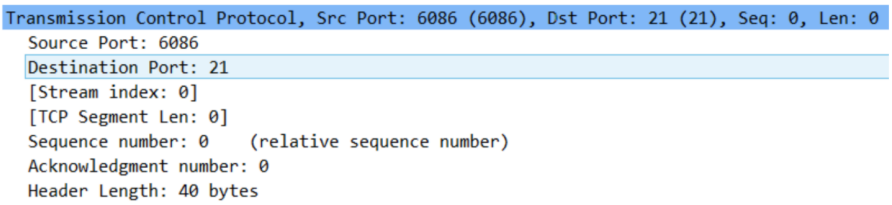

Figura 4.11: Excerto de um log TCP

No protocolo TCP, o mecanismo de controlo de congestionamento funciona à base de vários algoritmos, como o AIMD ou o slow-start, os quais permitem evitar enviar mais dados do que a ligação consiga suportar. Como a evolução do throughput foi exponencial de início, estamos a falar de um algoritmo slow-start. Neste caso, a sliding window também tem um papel relevante, e como tal o seu tamanho é um dos campos mais relevantes. Teve alguns picos e descidas, sendo que o primeiro timeout foi atingido no momento em que se iniciou o download no tux2. Por fim estabilizou, mostrando que a evolução está de acordo com o mecanismo de controlo de congestionamento.

Sabendo que a largura de banda do servidor é limitada e que o canal de comunicação é partilhado, podemos concluir que o aparecimento de uma segunda ligação, reduzirá a capacidade disponível para cada ligação para metade. Desta forma, a velocidade de transferência no computador diminuirá assim que o segundo computador iniciar a transferência.

5 Conclusões

Ao longo deste trabalho laboratorial tivemos contacto com diversas vertentes da configuração de redes, nomeadamente no que diz respeito à ligação de cabos e portas e comandos de interface com o router e o switch. Para além disso, aplicou-se conhecimentos de programação em C, aprendendo ao mesmo tempo a utilizar sockets.

Foi ainda possível consolidar-se alguns aspetos relativos à pilha protocolar da arquitetura de computadores. A camada de aplicação, responsável por ligar o utilizador à rede foi inserida no contexto da primeira parte do trabalho. Nesta, foi possível compreender melhor o

funcionamento do protocolo TCP-IP. Na segunda parte do trabalho foi analizada a camada de Rede. Através da construção da uma rede de computadores, foi possível perceber o funcionamento interno da ligação dos vários tipos de elementos que constituem uma rede de computadores.