1 2 9 0 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE D COIMBRA

REDES DE COMUNICAÇÃO

2022/2023

Relatório do trabalho prático

Serviço de notícias

Trabalho realizado por:

Pedro Ramalho nº 2019248594

Raquel Cardoso nº 2021240498

Índice

Introdução

Para melhor compreender o funcionamento da difusão de informação – a sua partilha e receção - através da internet, foi proposto o desenvolvimento de um trabalho prático que visava, sobretudo, a utilização de ligações TCP e UDP e a criação e gestão de grupos multicast. Desta forma, os alunos teriam de colocar em prática e consolidar os conhecimentos adquiridos na unidade curricular de Redes de comunicação.

O presente relatório destina-se a explicar o desenvolvimento do trabalho prático em questão, representando um mecanismo auxiliar à avaliação do código e da simulação desenvolvidos.

De referir que o trabalho foi desenvolvido e testado dentro de uma máquina virtual com sistema operativo Linux e com recurso à aplicação GNS3.

Descrição do código

O código divide-se em três ficheiros:

- "server.c", onde constam todas as funções que o servidor é responsável por executar;
- "cliente.c", onde constam todas as funções responsáveis por gerir a interação com o utilizador;
- "configFileRC.txt", onde constam as credenciais necessárias para inicializar o programa com alguns utilizadores de base e que pode apenas ser alterado pelos utilizadores com permissão de administrador também nele constantes.

Server.c

O ficheiro server.c é executado através do comando *$news_server {PORTO_NOTICIAS} {PORTO_CONFIG} {ficheiro de configuração}*, pelo que o programa se inicia verificando se foram recebidos valores suficientes e se os mesmos estão corretos.

Um dos critérios essenciais para o desenvolvimento do servidor era que este aceitasse ligações de vários clientes ao mesmo tempo, bem como dois tipos de ligação diferentes em simultâneo – UDP e TCP, uma vez que a consola de administração apenas seria acedida por UDP. Para solucionar esta questão, foram criados processos para lidar com cada cliente que acedesse por TCP – um processo "filho" por cada acesso TCP – e um processo que ficaria responsável por receber apenas ligações UDP permitindo o acesso à consola de administração.

No processo principal do servidor, foi criado um socket TCP que irá esperar por ligações de clientes. Uma vez detetada uma ligação, a função "connect" estabelece a ligação e passa a mesma para um processo filho que permitirá continuar a comunicação servidor-cliente enquanto o processo principal recebe mais ligações processando-as da mesma maneira.

Através do processo principal, foi, também, criado um processo secundário onde se cria o socket UDP e se espera a receção de uma mensagem que deve ser um nome de utilizador de um possível administrador. Uma vez verificadas as credenciais do utilizador para confirmar se se trata de um administrador, a comunicação com a consola de administração é estabelecida.

A nível de gestão de dados, foram criadas duas estruturas para guardar as informações dos utilizadores e dos tópicos de notícias, respetivamente. Estas estruturas foram guardadas numa memória partilhada, criada para permitir a comunicação entre processos, dado que os vários utilizadores precisam de aceder a informações atualizadas e este representava o mais rápido e eficaz mecanismo de comunicação inter-processos.

A criação de estruturas permitiu, ainda, a associação de um socket UDP especifico a cada tópico de noticias, permitindo o funcionamento das comunicações por multicast.

Para efeitos de gestão da memória partilhada e para facilitar o uso de ponteiros ao longo do programa, foram estabelecidos limites máximos de utilizadores registados, tópicos e tamanhos de buffers de comunicação.

Client.c

O ficheiro client.c dever ser executado através do comando *news_client {endereço do server} {PORTO_NOTICIAS},* iniciando-se o programa, da mesma maneira que o anterior, por verificar se as variáveis.

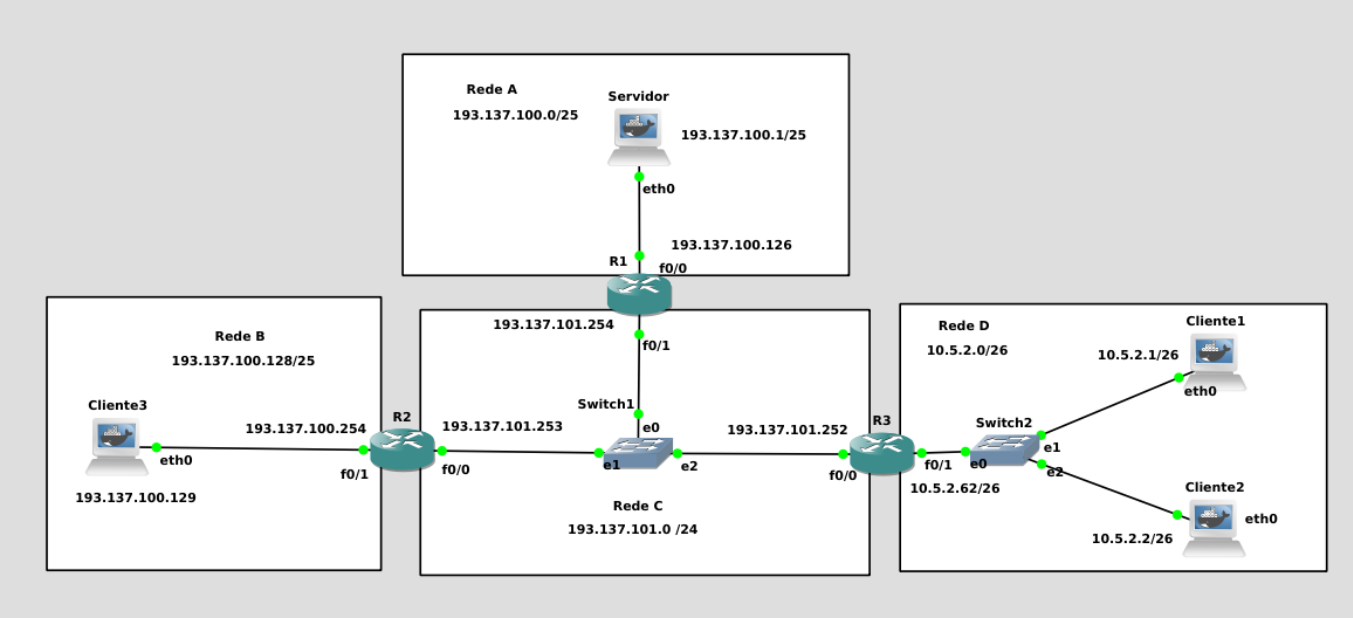
O cliente.c estabelece a comunicação direta com o utilizador e envia os dados recebidos ao servidor para os processar e satisfazer os pedidos dos clientes, através de ligação TCP, sujo socket foi criado na função principal.

Para permitir a comunicação com os grupos multicast, foi criado um socket UDP que será associado aos vários grupos multicast.

De forma permitir a leitura continua das notícias enviadas para os grupos multicast, foi criada uma thread que vai estar continuamente à espera de receber ligações UDP com noticias e as vai imprimir na consola do utilizador.

Simulação GNS3

Captura de ecrã do projeto no gns3.



Configuração do servidor:

```
auto eth0
    iface eth0 inet static
    address 193.137.100.1
```

netmask 255.255.255.128
gateway 193.137.100.126

Configuração dos clientes:

Cliente1:

auto eth0
iface eth0 inet static
address 10.5.2.1
netmask 255.255.255.192
gateway 10.5.2.62

Cliente2:

auto eth0
iface eth0 inet static
address 10.5.2.2
netmask 255.255.255.192
gateway 10.5.2.62

Cliente3:

auto eth0
iface eth0 inet static
address 193.137.100.129
netmask 255.255.255.128
gateway 193.137.100.254

Configurações dos routers:

Router_1:
enable
config t
int fa0/0
ip add 193.137.100.126 255.255.255.128
no shut
exit
int fa0/1
ip add 193.137.101.254 255.255.255.0
no shut
exit
ip route 193.137.100.128 255.255.255.128 193.137.101.253
exit
copy running-config startup-config
Router_2:
enable
config t
int fa0/0
ip add 193.137.101.253 255.255.255.0

```
no shut
exit
int fa0/1
ip add 193.137.100.254 255.255.255.128
no shut
exit
ip route 193.137.100.0 255.255.255.128 193.137.101.254
exit
copy running-config startup-config
```

Router_3:

```
enable

config t

int fa0/0

ip add 193.137.101.252 255.255.255.0

no shut

exit

int fa0/1

ip add 10.5.2.62 255.255.255.192

no shut

exit

ip route 193.137.100.128 255.255.255.128 193.137.101.253

ip route 193.137.100.0 255.255.255.128 193.137.101.254

access-list 30 permit 10.5.2.0 0.0.0.63

ip nat inside source list 30 interface FastEthernet0/0 overload

int fa0/1

ip nat inside

no shut

exit

int fa0/0

ip nat outside

no shut

exit

end

copy running-config startup-config
```