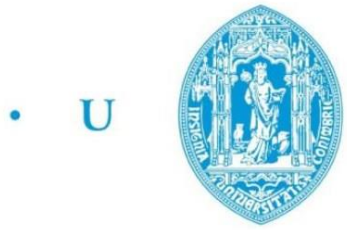

FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

PROJETO 2ª Meta

REDES DE COMUNICAÇÃO

LICENCIATURA EM ENGENHARIA INFORMÁTICA

2022/2023

Francisco Henrique Baltazar Querido - 2021221158

1- Introdução:

Este relatório descreve a implementação de um sistema de difusão de notícias. O objetivo deste trabalho prático foi desenvolver uma aplicação que faz uso dos protocolos UDP e TCP, bem como das comunicações IP multicast. O trabalho compreendeu duas fases distintas, cada uma avaliada separadamente.

1. Cenário de Comunicações: O cenário de rede utilizado para suportar a aplicação consistiu em uma configuração com 3 routers e 2 switches. Os equipamentos foram devidamente configurados para permitir a comunicação entre os clientes (PCs) e o servidor. Todos os PCs (clientes e servidor) utilizaram sistemas operacionais Linux, baseados em imagens docker disponíveis no GNS3.
2. Rede de Comunicação de Suporte à Aplicação: A configuração da rede de comunicação envolveu os seguintes aspectos relacionados ao endereçamento IPv4:

- A rede onde estão localizados os clientes Cliente1 e Cliente2 (Rede D) utilizou endereços privados, sendo implementado o NAT (Network Address Translation) no router R3. Foi utilizada a rede 10.5.2.0/26 para endereçar todas as interfaces na Rede D.
- As redes A, B e C foram endereçadas utilizando a faixa de endereços 193.137.100.0/23. A distribuição de endereços foi realizada de forma a garantir que a Rede C possuísse mais endereços disponíveis do que as restantes, e que a Rede A ficasse com os endereços mais baixos.
- Todos os equipamentos foram atribuídos endereços IP apropriados, de acordo com a gama de sub-rede convencionada.

Além da configuração da rede, o router R3 foi configurado para suportar SNAT (Source NAT), garantindo que a rede dos clientes Cliente1 e Cliente2 fosse a rede interna.

2- Cenário GNS3:

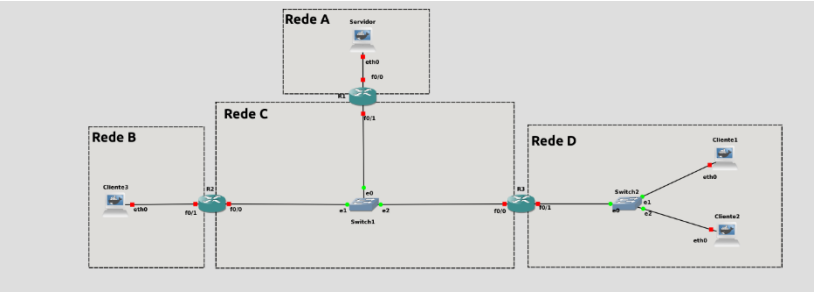

Figura 1-Cenário GNS3

Para a configuração deste cenário de redes, foram consideradas 4 redes como ilustrado na figura 1. Para a endereçar das Redes A, B e C foi usada a rede 193.137.100.0/23 dividida da seguinte forma:

-Rede A:

- 193.137.100.0/25

-Rede B:

- 193.137.100.128/25

-Rede C:

- 193.137.101.0/24

Desta forma foi garantido que a Rede C fica com mais endereços disponíveis que as restantes e que a Rede A fica com os endereços mais baixos, não sendo desperdiçados quaisquer endereços.

A Rede D utiliza endereços privados, foi utilizada a rede 10.5.2.0/26 para endereçar todas as interfaces na Rede D.

2.1 - Configuração dos Dispositivos:

2.1.1- Routers:

Configuração do Router1:

```
config t
interface f0/0
ip address 193.137.100.126 255.255.255.128
no shutdown
exit
interface f0/1
ip address 193.137.101.2 255.255.255.0
no shutdown
exit
ip route 193.137.100.128 255.255.255.128
193.137.101.1
ip route 10.5.2.0 255.255.255.192
193.137.101.3
exit
copy running-config startup-config
```

Configuração do Router2:

```
config t
interface f0/0
ip address 193.137.101.1 255.255.255.0
no shutdown
exit
interface f0/1
ip address 193.137.100.254 255.255.255.128
no shutdown
exit
ip route 193.137.100.0 255.255.255.128
193.137.101.2
ip route 10.5.2.0 255.255.255.192
193.137.101.3
exit
copy running-config startup-config
```

Configuração do Router3:

```
config t
access-list 1 permit 10.5.2.0 0.0.0.63
ip nat inside source list 1 interface FastEthernet0/0 overload
interface f0/0
ip address 193.137.101.3 255.255.255.0
ip nat outside
no shutdown
exit
interface f0/1
ip address 10.5.2.62 255.255.255.192
ip nat inside
no shutdown
exit
ip route 193.137.100.0 255.255.255.128 193.137.101.2
ip route 193.137.100.128 255.255.255.128 193.137.101.1
exit
copy running-config startup-config
```

2.1.2- Clientes e Servidores:

Servidor:

```
auto eth0
iface eth0 inet static
address 193.137.100.1
netmask 255.255.255.128
gateway 193.137.100.126
up echo nameserver 193.137.100.126 > /etc/resolv.conf
```

Cliente 1:

```
auto eth0
iface eth0 inet static
address 10.5.2.1
netmask 255.255.255.192
gateway 10.5.2.62
up echo nameserver 10.5.2.62 > /etc/resolv.conf
```

Cliente 2:

```
auto eth0
iface eth0 inet static
address 10.5.2.2
netmask 255.255.255.192
gateway 10.5.2.62
up echo nameserver 10.5.2.62 > /etc/resolv.conf
```

Cliente 3:

```
auto eth0
iface eth0 inet static
address 193.137.100.129
netmask 255.255.255.128
gateway 193.137.100.254
up echo nameserver 193.137.100.254 > /etc/resolv.conf
```

3- Funcionamento Da Aplicação

3.1-Servidor

O servidor é um programa em linguagem C que implementa um servidor de notícias com uma consola de administração. Aqui está uma descrição sucinta do seu funcionamento:

1. O código inclui várias bibliotecas padrão e define constantes, estruturas de dados e variáveis globais.
2. A função init é responsável por inicializar o programa. Ela lê um arquivo de configuração, armazena os dados dos usuários em uma estrutura de dados compartilhada e cria um semáforo para controle de acesso.
3. A função admin_console é responsável por lidar com os comandos da consola de administração. Ela cria um socket UDP e aguarda conexões dos administradores. Após a autenticação, os administradores podem executar comandos como adicionar usuários, remover usuários, listar usuários, sair da consola ou encerrar o servidor.
4. A função main é o ponto de entrada do programa. Ela recebe argumentos da linha de comando, inicializa o programa e cria um processo filho para executar a consola de administração e um processo pai para lidar com as solicitações de clientes.
5. O processo filho (consola de administração) executa a função admin_console passando a porta de configuração como argumento.
6. O processo pai (lidar com clientes) executa a função handleClient para lidar com as solicitações de clientes. Essa função cria um socket TPC, aguarda conexões de clientes, quando recebe uma conexão, cria um processo filho que processa suas solicitações de acordo com as suas permissões.
7. Por fim o programa termina esperando o final da execução de todos os processos filhos, libertando os recursos criados e escrevendo no ficheiro de configurações os dados atualizados dos utilizadores da aplicação.

O código implementa recursos como autenticação de administradores, adição e remoção de usuários, listagem de usuários, encerramento do servidor, criar tópicos de notícias, listar os mesmos, subescrever um tópico e enviar noticias para um grupo multicast de clientes subscritos ao mesmo. Também faz uso de memória compartilhada e semáforos para garantir a consistência dos dados e o acesso exclusivo a recursos críticos.

3.2-Cliente

Este código é um programa em C que implementa um cliente para se comunicar com um servidor através de sockets TCP. O cliente permite que um usuário se autentique no servidor e envie comandos para interagir com o mesmo.

A função autenticar() é responsável por autenticar o usuário no servidor. Ela solicita o nome de usuário e senha, envia essas informações ao servidor e aguarda a resposta. Se as credenciais forem corretas, o usuário é autenticado e pode começar a interagir com o servidor.

Dentro do loop principal de interação com o servidor, o usuário pode digitar que são enviados ao servidor através do socket TCP e as respostas do servidor são recebidas e exibidas na tela.

Se o comando "SUBSCRIBE_TOPIC" for enviado, o programa cria um processo filho usando fork(). O processo filho chama a função subscribe() e passa o endereço IP do tópico como argumento. A função subscribe() configura um novo socket para receber mensagens multicast do tópico e imprime as mensagens recebidas na tela.

O programa também define as funções cleanup_child() e cleanup() para lidar com a limpeza de recursos quando o programa é encerrado. Essas funções são chamadas quando o usuário pressiona Ctrl+C ou quando um processo filho é encerrado.

No final do programa, o socket é fechado e o programa é encerrado.