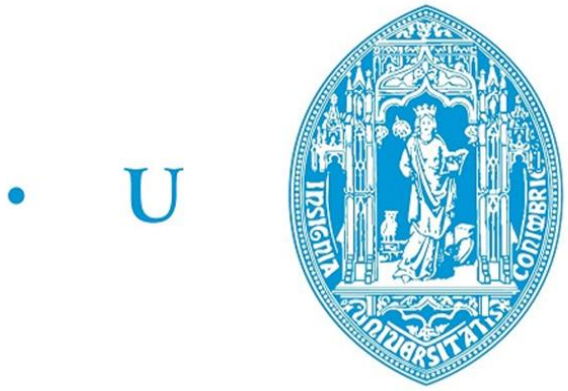

FCTUC FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Redes de Comunicação

Relatório

João de Macedo: 2021220627

Johnny Fernandes: 2021190668

20 de maio de 2023

Introdução

O presente relatório tem como objetivo descrever de maneira clara as opções tomadas na construção da solução e o modo de funcionamento do trabalho prático realizado no âmbito da disciplina de Redes de Comunicação. Neste trabalho, desenvolvemos um sistema de disseminação de notícias, utilizando técnicas de comunicação e os protocolos da pilha TCP/IP, nomeadamente os protocolos UDP e TCP, além das comunicações *multicast*.

Ao utilizar o GNS3, foi possível criar uma topologia de rede personalizada, com dispositivos virtuais configurados de acordo com os requisitos do nosso projeto.

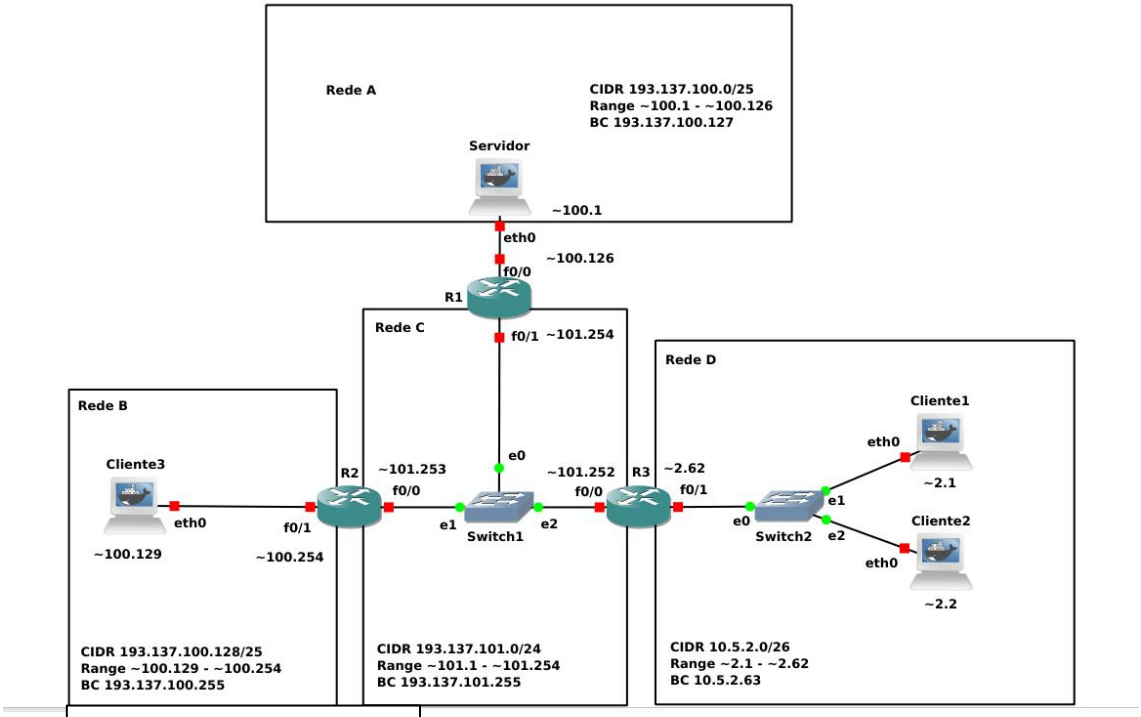

Figura 1-Modelo da Rede no Gns3

Desenvolvimento

No decorrer deste projeto, foi desenvolvido um servidor capaz de lidar com conexões tanto TCP quanto UDP. O servidor TCP tem a finalidade de atender às solicitações dos clientes (jornalistas e leitores). Por sua vez, o servidor UDP é responsável por permitir que os administradores façam a gestão utilizadores e configurem a aplicação, nomeadamente adicionar, remover e listar utilizadores, e desligar o servidor.

Ao criar um servidor que suporta tanto TCP quanto UDP, garantimos uma abordagem completa e abrangente para a comunicação e administração da aplicação. Essa estrutura permite uma integração fluída entre os clientes, que se beneficiam da estabilidade e confiabilidade do TCP, e os administradores, que têm a flexibilidade e eficiência do UDP para suas tarefas de configuração e gestão.

O servidor é composto pelos seguintes arquivos: sv_main.c; sv_main.h; sv_shm.c; sv_shm.h; sv_tcp.c; sv_tcp.h; sv_udp.c; sv_udp.h.

Cada ficheiro "*header*" corresponde ao ficheiro ".c" com o mesmo nome de prefixo, contendo as declarações das funções e estruturas de dados usadas no mesmo.

Neste projeto foi feita a utilização de processos e *threads*. No caso dos processos, é aberto um processo para cada conexão TCP feita por um cliente, por forma a conseguir uma escuta e escrita independente e multiprocesso simultânea com outros clientes conectados. Nessa medida, surgiu a necessidade de criação de uma *shared memory* para a comunicação entre processos e acesso à listagem dos grupos *multicast*, motivo pelo qual temos um arquivo sv_shm.c que será abordado adiante. Além disso, as *threads* são responsáveis cada uma por um protocolo (o TCP e o UDP). O uso de ficheiros na abordagem do desenvolvimento do programa foi apenas considerado para o carregamento e registo de contas de utilizadores.

sv_main.c:

Este ficheiro de código tem como propósito principal iniciar o programa e coordenar suas funcionalidades principais. Realiza as seguintes tarefas:

- Configuração dos sinais (para encerramento com Ctrl+C);
- Verificação e obtenção dos argumentos para inicializar;
- Verificação da validade e *binding* das portas;
- Carregamento de utilizadores;
- Inicialização de *threads* para TCP e UDP;

sv_tcp.c:

Para receber as ligações TCP criamos um ficheiro chamado de "sv_tcp.c".

Ao ligar o servidor, é criada uma *thread* para ficar constantemente à escuta de novas conexões de clientes. Esta realiza várias etapas importantes, como a criação e configuração do *socket* TCP, a vinculação do servidor a uma porta específica e a configuração de opções adicionais.

Uma vez configurado, o servidor entra em modo de escuta e espera por novas conexões dos clientes. Quando uma conexão é estabelecida, o servidor aceita o *socket* do cliente e cria um processo para lidar com essa conexão.

sv_udp.c:

Para receber as ligações UDP implementamos um ficheiro chamado de "sv_udp.c".

Ao ligar o servidor, a função em questão é responsável por criar uma *thread* que trata das comunicações UDP no servidor. Esta configura e vincula um *socket* UDP à porta especificada. Em seguida, aguarda a receção de mensagens dos clientes. Quando uma mensagem é recebida, verifica-se se o cliente já está autenticado ou não. Caso não esteja na lista de clientes, o servidor adiciona-o e envia uma mensagem de login para o cliente. A mensagem recebida é validada e autenticada de acordo com o comando e argumentos. Em seguida, a resposta apropriada é preparada e enviada ao cliente.

sv_shm.c:

As funções neste ficheiro são utilizadas para manipular uma estrutura de memória partilhada - multicast_shm_t. Essa estrutura é utilizada para armazenar informações sobre grupos *multicast*, como ID, tópico, endereço IP e porta.

A memória partilhada é necessária porque o servidor abre processos separados para lidar com as conexões TCP dos clientes. No entanto, a lista de grupos *multicast* precisa ser acedida por todos esses processos de maneira igual. Como cada processo tem o seu próprio espaço de endereçamento, é necessário um espaço de memória partilhada para permitir o acesso simultâneo a esses dados. Desta forma, e porque a lista dos grupos *multicast* tem que estar a todo o instante sincronizada para que os diferentes clientes possam receber e enviar a informação de forma fidedigna e ainda como – conforme mencionado anteriormente – não fazemos o uso de ficheiros para guardar esse tipo de informação, a solução mais apropriada foi o uso de uma memória partilhada.

Clientes:

Para ligações de clientes administradores usamos o *Netcat* que usam comunicações *UDP*. Já para ligações de clientes leitores e jornalistas que usam comunicações *TCP*, criamos um ficheiro client.c (com o seu respetivo *header*) que origina um executável *client*.

O código em questão implementa um programa cliente TCP em C. Ele tem como objetivo estabelecer a conexão com um servidor e permitir a troca de mensagens entre o cliente e o servidor. O programa possui duas *threads* principais: uma responsável por ler os dados recebidos do servidor e exibi-los no ecrã, e outra responsável por ler os comandos enviados pelo utilizador e enviá-los para o servidor. Estas *threads* permitem que o cliente receba mensagens e envie comandos de forma assíncrona, sem bloquear a interação com o servidor. O programa também trata sinais, como o SIGINT, para permitir que o utilizador encerre a execução do cliente de maneira adequada.

Execução:

Para execução do servidor será necessário usar o comando na pasta raiz 'Aplicação' ./server/server <porta_news> <porta_config> <ficheiro_de_utilizadores> como por exemplo, ./server/server 9000 9001 credentials – conforme os ficheiros fornecidos juntamente.

Por outro lado, para a execução de um cliente, deverá ser executado da mesma forma, na pasta raiz 'Aplicação' com o comando ./client/cliente <ip_do_servidor> <porta_news> como por exemplo ./client/client 193.137.100.1 9000.

Por fim, para acesso à área de configuração do servidor, através de uma conta admin, será necessário usar o *Netcat* com o comando nc -u <ip_do_servidor> <porta_config> como por exemplo nc -u 193.137.100.1 9001. Desta forma o cliente estará conectado, contudo, para o servidor detetar conexão é necessário enviar alguma informação.

Convém ainda deixar nota que um administrador nunca poderá aceder através do terminal de cliente da mesma forma que um jornalista ou um leitor não poderá aceder através do *Netcat*, para comunicação com o sistema de administração do servidor.

Conclusão

O trabalho prático realizado no âmbito da disciplina de Redes de Comunicação abordou a implementação de um servidor capaz de lidar com conexões *TCP* e *UDP*, permitindo uma comunicação eficiente entre os clientes (jornalistas e leitores) e os administradores.

Ao criar um servidor que suporta ambas as modalidades de comunicação, *TCP* e *UDP*, obtivemos uma abordagem completa e abrangente para satisfazer as necessidades de diferentes utilizadores. Os clientes beneficiaram da estabilidade e fiabilidade do *TCP*, enquanto os administradores puderam aproveitar a flexibilidade e eficiência do *UDP* para as suas tarefas de configuração e gestão.

Por fim, a implementação do cliente permitiu a interação com o servidor, possibilitando a subscrição em grupos de *multicast*, o envio de mensagens e a participação em comunicações relevantes.

Em resumo, este trabalho prático proporcionou uma compreensão mais aprofundada dos conceitos de redes de comunicação, demonstrando a aplicação dos protocolos *TCP* e *UDP*, assim como a importância da comunicação eficiente na construção de sistemas distribuídos. Através da análise das funcionalidades implementadas no servidor e cliente, foi possível explorar e consolidar conhecimentos relacionados com a arquitetura de rede e troca de informações entre diferentes entidades num ambiente distribuído.