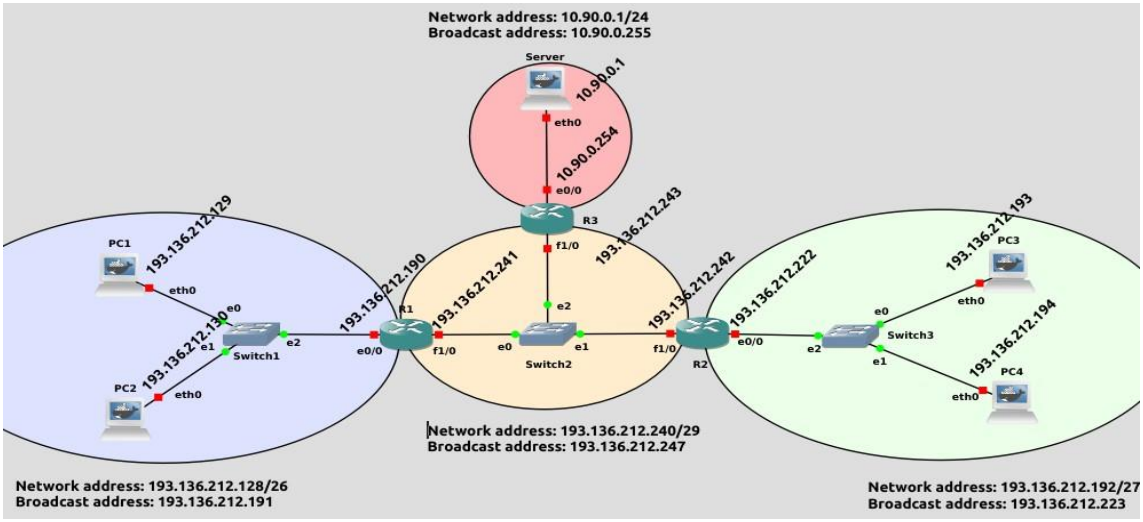
Relatório do projeto “Sistema de messaging”
Disciplina de Redes de Comunicação (2020-2021)

Diagrama das redes



Atribuição de IPs

Exemplo de atribuição para router (R1):

```
R1#config terminal
R1#interface Ethernet0/0
R1l(config-if)#ip address 193.136.212.190 255.255.255.192
R1(config-if)#no shutdown
R1(config-if)#exit
R1#interface FastEthernet1/0
R1l(config-if)#ip address 193.136.212.241 255.255.255.248
R1(config-if)#no shutdown
R1(config)#exit
```

Exemplo de atribuição para container (PC1):

Com o container desligado, clique direito no container e “Edit Config”.

```
auto eth0
iface eth0 inet static
       address 193.136.212.129
       netmask 255.255.255.192
       gateway 193.136.212.190
```

IP Routes

Uma vez que a rede 10.90.0.0/24 é uma rede DMZ, não serão feitos ip routes para essa rede. A ligação será feita através da configuração do NAT no router R3.

R1	`R1#ip route 193.136.212.192 255.255.255.224 193.136.212.242`
R2	`R2#ip route 193.136.212.128 255.255.255.128 193.136.212.241`
R3	`R3#ip route 193.136.212.192 255.255.255.224 193.136.212.242` `R3#ip route 193.136.212.128 255.255.255.128 193.136.212.241`

Alexy Damien Tomé Julien Denis de Almeida Nº2019192123 PL1
Joana Maria Silva Simões Nº2019217013 PL1

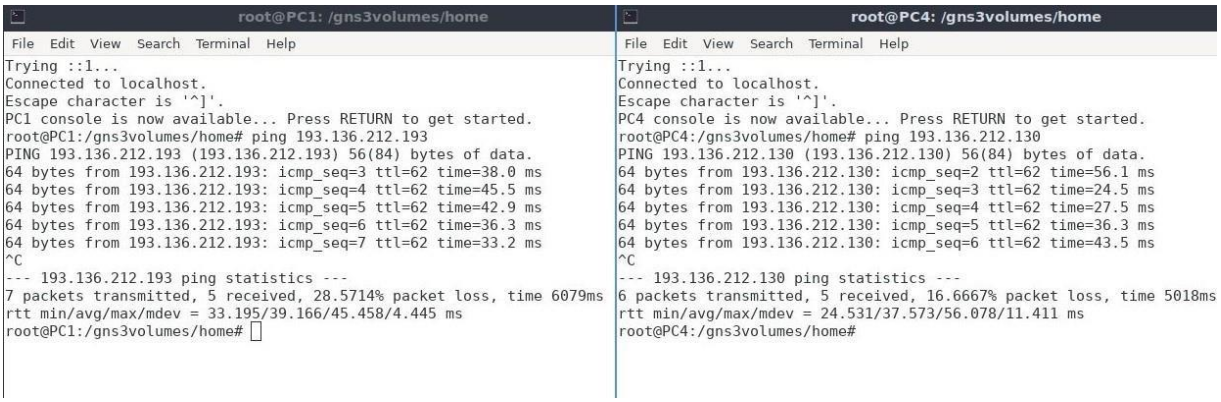
Configuração do NAT

É feita a configuração do router R3 para suportar modos SNAT e DNAT. O router R3 efetua SNAT para comunicações com origem na rede do servidor e destinadas às redes externas e efetua DNAT para as comunicações com origem nas redes externas e destinadas ao servidor, sendo que as comunicações com destino ao endereço da interface externa desse router são redirecionadas para o servidor.

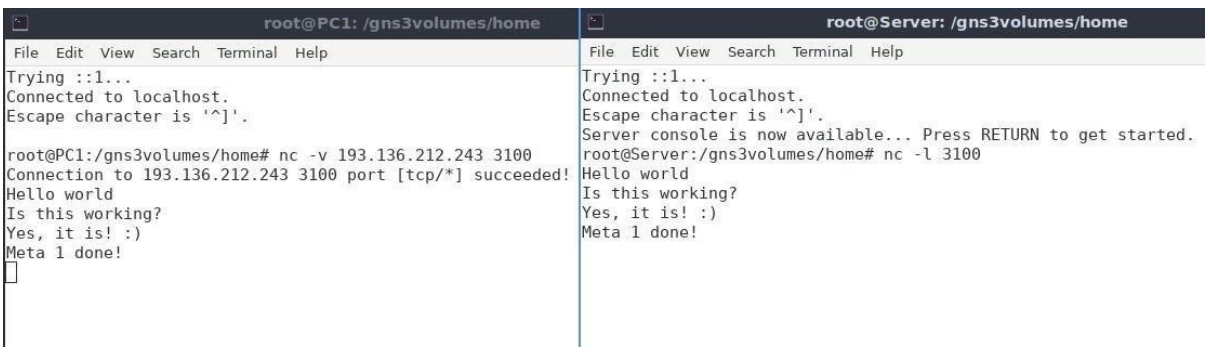
SNAT	DNAT
router# config terminal router(config)# access-list 30 permit 10.90.0.0 0.0.0.255 router(config)# ip nat inside source list 30 interface Ethernet0/0 overload router(config)#interface FastEthernet0 router(config-if)# ip nat outside router(config-if)# exit router(config)# interface Ethernet0 router(config-if)# ip nat inside router(config-if)# end	router# config terminal router(config)# ip nat inside source static tcp 10.90.0.1 3100 router(config)# ip nat inside source static udp 10.90.0.1 3100 193.136.212.243 3100 router(config)#interface FastEthernet0 router(config-if)# ip nat outside router(config-if)# exit router(config)# interface Ethernet0 router(config-if)# ip nat inside router(config-if)# end

Testes de ligação

Pings PC1->PC3 e PC4->PC2



Licação do PC1 ao Server através do netcat



Estrutura do programa

O programa está dividido numa pasta client, numa pasta server e numa série de documentos relevantes ao projeto, nomeadamente o diagrama de redes ou o projeto do GNS3 (não entregues com o código fonte). Seguidamente, as duas pastas estão também divididas da seguinte forma:

client: admin.c, client.c, client.h, functions.c, functions.h, global.h, makefile.

server: server.c, server_clients.c, server_config.c, server.h, file.c, file.h, global.h, makefile, clients.bin, groups.bin.

Ficheiros

Os dois ficheiros binários, clients.bin e groups.bin, são usados para manter o registo dos utilizadores e dos grupos inseridos pelo administrador e pelos clientes, respetivamente. Uma vez que são ficheiros binários, mesmo que o servidor feche a sessão, o conteúdo escrito para estes (quer seja informação de clientes ou de grupos) permanece após o fim da sessão, ou seja, ao iniciar novamente o servidor estes continuam com a informação.

No entanto, um defeito destes ficheiros binários prende-se com o facto de estarmos sempre a abrir e fechá-los para leitura e escrita.

Client

O cliente tenta estabelecer ligação ao servidor, sendo pedido o endereço e o porto do mesmo. Após isso, é pedido que insira um username e uma password, sendo que o cliente apenas pode conectar-se ao servidor se esses dois dados fizeram parte do ficheiro de registos, adicionados pelo administrador. Se a autentificação for bem-sucedida, é enviado ao cliente uma lista de modos de comunicação.

Existe uma função **is_error()** que verifica se a mensagem recebida do servidor é uma mensagem de erro. Caso seja, o programa do cliente para. Foi também criada uma thread *chat(), usada em cada modo de comunicação, que permite ficar à escuta de mensagens que o cliente possa receber de outros clientes ou do servidor, enquanto este também escreve as suas próprias mensagens.

Em cada modo de comunicação, se o utilizador pretender sair bastará escrever “EXIT!” na linha de comandos.

Client/Server

É desde logo criada a thread *chat() até o utilizador decidir sair do modo de comunicação. Dentro de um loop, o utilizador irá sempre escrever o user e a mensagem que pretende enviar-lhe.

P2P

O cliente envia ao servidor o username ao qual pretende enviar mensagens e o servidor envia de volta o endereço IP do username. Após isso, é criada a thread *chat() e é criada uma nova estrutura do socket com o IP do username de destino. Dentro de um loop, o utilizador vai sempre enviando mensagens a esse username de destino, sem passar pelo servidor.

Group

O utilizador recebe duas escolhas: criar um grupo ou juntar-se a um grupo já existente. Na primeira opção, o cliente envia o nome do grupo que pretende criar e o servidor verifica se esse grupo já existe. Caso não exista, devolve ao cliente o endereço multicast do grupo. Na segunda opção, é feito exatamente o mesmo processo que na primeira opção, acrescendo a criação da thread *chat() e o início de comunicação ao grupo através do endereço IP recebido. Como acontece no P2P, é criada uma nova estrutura do socket com o endereço multicast do grupo e, dentro de um loop, o utilizador vai enviando

continuamente mensagens até decidir sair.

Server

O server encontra-se dividido em duas partes: uma para administração e outra para comunicação com os clientes. A primeira (**server_config.c**) utiliza TCP, enquanto a segunda (**server_clients.c**) utiliza UDP. É usado o mesmo porto para as comunicações TCP e UDP, visto serem dois tipos de comunicação diferentes.

server_config

Recebe do administrador uma série de comandos, sendo eles QUIT, LIST, ADD e DEL, permitindo listar, adicionar ou apagar users que se encontram no ficheiro de registos. É também usado um *unnamed semaphore* (mutex_registers) para controlar o acceso à lista de clientes. A partir dos comandos recebidos e verificados, o servidor altera o ficheiro binário.

server_clients

Tem uma lista de endereços multicasts que se possam endereçar para os grupos que forem criados. Recebe do cliente uma série de comandos, sendo eles pedidos de autentificação, pedidos de modos de comunicação e envios de mensagens para outros utilizadores/grupos. Em cada pedido são feitas verificações e, caso um dos requisitos não seja preenchido, envia uma mensagem de erro ao cliente. O servidor também verifica se o cliente está apto a utilizar determinados modos de comunicação consoante os dados inseridos pelo administrador no ficheiro de registos.

Admin

O administrador tenta primeiro conectar-se ao servidor enviando um pedido com o IP do servidor e o seu porto. Através da linha de comandos, insere comandos ADD, LIST, DEL ou QUIT, que permitem adicionar, listar ou apagar utilizadores no ficheiro de registos, sendo o administrador quem faz o pedido de alteração desse ficheiro ao servidor.

Atuais problemas do projeto e trabalho futuro

No seu estado atual, apesar do projeto executar e ser possível desempenhar várias funções, este encontra-se ainda com alguns problemas, nomeadamente:

1. Na parte do administrador não são feitas verificações acerca do input do username e password, uma vez que no código tem gerado vários conflitos e interrompido o uso de várias funções;
2. Os ficheiros binários são sempre abertos e fechados para leitura e escrita;
3. Users pertencentes a um grupo não conseguem receber as mensagens dos outros users desse mesmo grupo, apenas os do próprio user, apesar de ser possível criar grupos.

Um trabalho futuro consistiria em resolver esses problemas, que se prendem sobretudo com código em linguagem C e ligações multicast.