UNIVERSIDADE D
COIMBRA

Redes de Comunicação: Relatório Trabalho Prático

Turmas online

Mariana Sousa & Rui Oliveira

Licenciatura em Engenharia Informática
Faculdade de Ciências e Tecnologia da Universidade de Coimbra

May 17, 2024

1 Introdução

Dado que na primeira meta foi realizado as configurações necessárias nos dispositivos do GNS3, implementámos a parte corresponde ao servidor -UDP, a parte do servidor -TCP e do cliente – TCP agora, para a meta final, implementamos o *Multicast*, dado que é um fator indispensável para o sucesso deste projeto.

2 Servidor

O *servidor.c* recebe os argumentos de execução, cria a *Shared Memory* que contém dois arrays, um com todos os utilizadores e outro com as turmas. Para manter a sincronização dos dados ao longo da execução, devido ao uso de processos, utilizamos dois semáforos, um para cada array da *Shared Memory*.

Seguidamente, a partir do ficheiro de texto fornecido como argumento, carregamos todos os dados do mesmo para o array dos utilizadores que se encontra na *Shared Memory* e executamos a função *"process_client"*, através de um fork, quando encontramos uma conexão TCP, e também chamamos a função "udp_server_function()".

3 Cliente

O nosso ficheiro cliente_tcp.c tem como função a comunicação com o servidor através da conexão TCP. Para a execução do mesmo, temos de fornecer o IP do servidor (193.137.100.1) bem como a porta a ser utilizada. Estas informações são obrigatórias para a criação do socket responsável pela comunicação entre ambos.

Posteriormente, o utilizador recebe a mensagem do pedido de Login, sem o mesmo bem sucedido o cliente não consegue aceder aos comandos disponíveis. Caso o cliente (aluno ou professor) inicie sessão corretamente, tem acesso aos comandos "LIST_CLASSES", "LIST_SUBSCRIBED" e "SUSBCRIBE_CLASS". No entanto, caso o utilizador seja do tipo "professor" tem acesso a dois comandos exclusivos, *"CREATE_CLASS"* e *"SEND"*.

No caso do comando utilizado ser *"SUBSCRIBE_CLASS"*, o programa cria uma *thread* responsável por juntar o cliente ao grupo *Multicast*, aguardando a leitora de mensagens enviadas pelo professor que criou a turma. De notar que antes da *thread* ser criada é elaborada a estrutura do UDP e *Multicast*, sendo estes preenchidos com as estruturas necessárias como o IP dado pelo servidor e a porta 5000. Deste modo, temos vários alunos a ouvir do mesmo IP *Multicast*, usamos a função *setsockopt()* para reutilizar as portas.

4 UDP

No server UDP, para acesso do admininstrador é criado o socket UDP com a porta fornecida e o bind. De seguida, é pedido ao administrador para realizar o LOGIN e enquanto este não for bem sucedido (seja por erro nas credenciais ou por não ser administrador), o utilizador é notificado. Se for bem sucedido, o administrador terá acesso aos comandos, *"ADD_USER", "DEL", "LIST"* e *"QUIT_SERVER"*. Caso este último seja selecionado, fecha o servidor e o programa escreve na consola uma última mensagem e limpa todos os recursos.

5 TCP

No nosso servidor TCP, gerido através de um processo filho, é criado o socket TCP com a porta fornecida. Para evitar o erro de binding, permitimos a reutilização de endereços com a função *setsockopt()*. Em seguida, realizamos o *bind()* e o *listen()*. Criamos também o socket para o *Multicast* onde utilizamos a mesma porta mencionada no cliente, que é a 5000.

No servidor, o endereço do *Multicast* é especificado na estrutura, sendo configurado apenas quando é criado uma nova turma.

Aumentamos o valor do TTL, permitindo que as mensagens de *Multicast* alcancem destinos mais distantes na rede, no nosso caso, chegar ao destino no destinatário do GNS3. O processo de LOGIN é essencialmente o mesmo que no servidor UDP. No loop while, os comandos esperados são *"LOGIN", "LIST_CLASSES", "LIST_SUBSCRIBED", "SUBSCRIBE_CLASS", "CREATE_CLASS" e "SEND"*.

Caso um comando seja inválido, uma mensagem de erro é enviada ao cliente indicando que o comando não existe. Os comandos *"CREATE_CLASS" e "SEND"* só podem ser executados por um usuário com permissões de professor. O papel de utilizador (administrador, aluno ou professor) é verificado e armazenado durante o login.

Ao criar uma turma com o comando *"CREATE_CLASS"*, a função *create_class()* é executada. Esta função recebe o ID e o nome da turma, e utiliza um endereço base em hexadecimal para gerar um novo endereço *Multicast*t exclusivo para cada turma. O endereço base para *Multicast* é definido como 239.0.0.0 (0xEF000000). Para cada nova turma criada, o ID da turma é adicionado ao endereço base, garantindo um endereço *Multicast* único e válido dentro do intervalo 239.0.0.0 a 239.255.255.255.

A função *send_cont()* é responsável por enviar mensagens a cada turma associada ao ID fornecido. O socket de *Multicast*, a estrutura do grupo *Multicast* e o socket do cliente são usados para enviar as mensagens através da função *sendto()*.

Dessa forma, o servidor gere de forma eficiente a comunicação *Multicast*, garantindo que cada turma tem um endereço exclusivo para o envio de mensagens, o que facilita a gestão das turmas e a disseminação de informações específicas para cada turma de alunos.

6 Conclusão

Em conclusão, este trabalho permitiu-nos explorar e aplicar diferentes técnicas de comunicação e com recurso aos protocolos da pilha TCP/IP na implementação de um sistema de turmas online para difusão de conteúdos. Através das duas fases do trabalho, conseguimos desenvolver funcionalidades para administradores, professores e alunos, demonstrando o conhecimento adquirido e as habilidades de programação em rede.