RELATÓRIO PROJETO

Redes de Comunicação

Síntese

Este projeto tem como objetivo implementar um sistema de turmas online para difusão de conteúdos, recorrendo a diversas técnicas de comunicação e com recurso aos protocolos da pilha TCP/IP. Em particular, iremos fazer uso dos protocolos UDP e TCP, bem como das comunicações IP multicast.

Henrique Oliveira
2022211169
Miguel Serra
2022218245

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Introdução

Este projeto propõe a implementação de um sistema de turmas online para a disseminação de conteúdos educacionais, fazendo uso de diversas técnicas de comunicação e os protocolos da pilha TCP/IP. A aplicação compreende duas fases de implementação.

Na primeira fase, o foco está na configuração da infraestrutura de rede que suportará a aplicação. São utilizados 3 routers e 2 switches, configurados para permitir a comunicação entre clientes (PCs) e servidor. O servidor atua como ponto central, responsável por autenticar e aceitar conexões de clientes, além de gerir turmas e usuários. As comunicações entre clientes são realizadas via multicast, enquanto as comunicações com o servidor ocorrem por TCP. O servidor também suporta comunicações UDP com uma consola de administração.

Na segunda fase, a ênfase é na construção da aplicação de gestão e difusão de conteúdos. O servidor é responsável por gerir turmas, usuários e permissões, além de permitir a interação com a consola de administração através de comandos UDP. Os clientes podem se autenticar, listar turmas, inscrever-se em turmas e receber conteúdos associados a elas. Professores têm a capacidade adicional de criar turmas e enviar conteúdos para os alunos nelas inscritos.

A arquitetura da aplicação é dividida entre servidor, consola de administração e clientes, cada um com suas funcionalidades específicas. O servidor é o núcleo da aplicação, responsável por coordenar a interação entre os clientes e gerenciar os recursos do sistema. A consola de administração oferece ferramentas para gerir users e configurar a aplicação, enquanto os clientes permitem que os users participem nas turmas e recebam conteúdos educacionais.

Servidor

O núcleo do programa encontra-se implementado num servidor para gestão de turmas e utilizadores, oferecendo funcionalidades específicas para administradores e professores. A estrutura baseia-se em sockets para comunicação com clientes e administradores, além de usar funções para manipulação de utilizadores e turmas. Abaixo, explico as principais opções tomadas na construção da solução e o modo de funcionamento:

Funções Principais:

- erro(): Trata erros de execução, exibindo uma mensagem e encerrando o programa.
- main(): Configura o servidor, inicializa sockets para clientes e administradores, e gere a aceitação de conexões e autenticação das mesmas.
- Funções como add_user(), delete_user(), list_users(), authenticate_client(), e authenticate_admin() permitem adicionar, remover, listar utilizadores e autenticar clientes e administradores.
- load_users_from_file() e save_users_to_file() carregam e salvam dados de utilizadores de/para um ficheiro, garantindo persistência.
- process_client(): Trata comandos dos clientes autenticados, como inscrição em turmas (subscribe_class()) e listagem de turmas (list_classes()), assim como os comandos exclusivos a professores.
- handle_admin_commands(): Lida com comandos do administrador via UDP, como adicionar, remover utilizadores e listar utilizadores.
- Funções como sanitize_input() garantem a limpeza e validação das entradas dos utilizadores, prevenindo possíveis alterações de comandos.

Comunicação e Funcionamento

Utiliza sockets TCP para comunicação com clientes e UDP para comunicação com administradores, tal como requirido.

A consola de administração permite a interação com um servidor através de UDP. Vou explicar brevemente as funcionalidades da consola de administração e como são implementadas no código:

- Autenticação de Administrador:
  Quando o programa é executado, inicialmente é solicitado que o administrador insira seu nome de usuário e senha. Estas credenciais são enviadas para o servidor para autenticação.
  Se as credenciais forem corretas, o servidor responde com "OK" e o administrador é autenticado, permitindo o acesso às funcionalidades adicionais.
  Caso contrário, o servidor responde com uma mensagem de erro e o administrador é solicitado a tentar novamente.
- Comandos Disponíveis:
  Após a autenticação bem-sucedida, o administrador tem acesso a uma série de comandos disponíveis:
  ADD_USER <username> <password> <user_type>: Adiciona um novo usuário com o nome de usuário, senha e tipo de usuário especificados.
  DELETE_USER <username>: Exclui o usuário especificado.
  LIST: Lista todos os usuários registrados.
  QUIT_SERVER: Encerra o servidor.
  BYE: Encerra a sessão na consola de administração e sai do programa.
- Envio de Comandos para o Servidor:
  O administrador insere o comando desejado na consola de administração.
  O comando é enviado para o servidor através de uma conexão UDP.
  O servidor processa o comando e responde com uma mensagem indicando o resultado da operação.
- Encerramento da Conexão:
  Quando o administrador digita "BYE" na consola de administração, o programa encerra a conexão com o servidor e sai do loop de execução.

Essas funcionalidades permitem que o administrador gerencie usuários e operações do servidor de forma remota e conveniente, através de uma interface de linha de comando simples e direta.

Cliente

O cliente liga-se ao servidor via TCP e realiza a autenticação ao enviar o seu nome de usuário e senha. Após a autenticação bem-sucedida, o cliente pode enviar comandos ao servidor, como listar turmas e inscrever-se nelas. Se recebr o comando específico, o cliente inicia a recepção de mensagens multicast da turma correspondente.

O código foi organizado para facilitar a manutenção e a adição de novas funcionalidades:

- Função erro: Para exibir mensagens de erro e encerrar o programa em caso de falha.
- Função receive_multicast: Gere a recepção de mensagens multicast.
- Função main: Gere a conexão com o servidor, autenticação do usuário e envio de comandos.
- Inicialização: O cliente é executado com o endereço do servidor e a porta como argumentos.
- Autenticação: O cliente liga-se ao servidor e realiza a autenticação ao enviar o nome de usuário e senha.
- Envio de Comandos: Após a autenticação, o cliente pode enviar comandos para listar turmas, inscrever-se nelas e receber conteúdos. Caso seja professor, também pode realizar os comandos reservados aos mesmos.
- Receção de Conteúdos: Se o cliente se inscrever em uma turma, ele recebe o endereço multicast e inicia a função receive_multicast para receber os conteúdos enviados para aquela turma.

Implementação GNS3

Após a implementação dos códigos para os clientes, a consola de administrador e o servidor e as funcionalidades para cada um dos códigos, estes foram adicionados ás pastas dos Dockers para que dentro do gns3 e para o cenário implementado na 1º meta deste projeto seja possível a sua execução.

Para a execução dos códigos temos que abrir as consolas dos Dockers e executar os comandos para permitir a seua execução:

- No caso do servidor primeiro compilamos o ficheiro .c do seguinte modo: “gcc servidormeta2.c -o servidormeta2”. Para a execução do código executamos “./servidormeta2 9999 9900 config.txt”.
- No caso dos clientes primeiro compilamos o ficheiro .c do seguinte modo: “gcc clientemeta2.c -o clientemeta2”. Para a execução do código executamos “./clientemeta2 193.137.100.1 9999”.
- No caso da consola de adminisstrador primeiro compilamos o ficheiro .c do seguinte modo: “gcc consola_admin.c -o consola”. Para a execução do código executamos “./consola 193.137.100.1 9900”.