UNIVERSIDADE FEDERAL DA FRONTEIRA SUL

CAMPUS CHAPECÓ

CURSO DE CIÊNCIA DA COMPUTAÇÃO

CARLOS GIOVANE NEU NOGUEIRA - 2311100010

MARCO ANTONIO DUZ - 2311100006

TÓPICOS ESPECIAIS EM COMPUTAÇÃO XIII:

Desenvolver uma aplicação de “bate-papo” (chat) com o protocolo MQTT

CHAPECÓ - SC

2025

SUMÁRIO

1. DESCRIÇÃO DO PROJETO

Uma aplicação de bate-papo robusta, desenvolvida em C, que utiliza o protocolo MQTT para comunicação em tempo real. O projeto implementa conversas privadas e em grupo, com foco na entrega das mensagens mesmo com o usuário desconectado.

Utilizando de uma arquitetura multi thread para garantir uma experiência de usuário.

1.1 FUNCIONALIDADES

1.1.2 COMUNICAÇÃO SEGURA

Todas as mensagens trocadas são criptografadas de ponta a ponta utilizando a biblioteca openssl em C, que possui a encriptação AES, as mensagens postadas nos tópicos são encriptadas.

Utiliza um sistema de solicitação de conversas ou seja o usuário só pode iniciar uma conversa com o outro após o aceite da solicitação.

1.1.2 CONVERSAS PRIVADAS E EM GRUPO

É possível criar tanto conversas privadas com um usuário quanto criar grupos de uma ou mais pessoas.

1.1.3 GERENCIAMENTO DE GRUPOS

Todos os grupos têm opções de adicionar participantes ao criar e gerenciar convites. Cada grupo possui um líder com a permissão para aceitar novos membros.

1.1.4 STATUS DE USUÁRIO

Visualizar quais usuários estão online ou offline em tempo real.

1.1.5 PERSISTÊNCIA DE DADOS

Mensagens e informações de usuários/grupos são salvas em arquivos locais, garantindo que os dados não sejam perdidos ao reiniciar a aplicação.

1.1.6 INTERFACE DE LINHA DE COMANDO

Menu interativo, simplificado montado direto no terminal, para facilitar na utilização das funcionalidades do chat.

1.1.7 ARQUITETURA MULTITHREAD

O sistema utiliza threads para gerenciar a interface do usuário e as tarefas de rede de forma independente, evitando bloqueios e garantindo que o programa continue operando.

2. ARQUITETURA DO SISTEMA

2.1 ESTRUTURA DAS PASTAS

As pastas foram divididas em funcionalidades, seguindo o formato abaixo:

A pasta *AES* fica responsável por todas as funções que envolvem a biblioteca openssl, utilizada na criptografia das mensagens.

No diretório *chat* está tudo relacionado ao chat, desde a listagem de chats quanto a conversa em si.

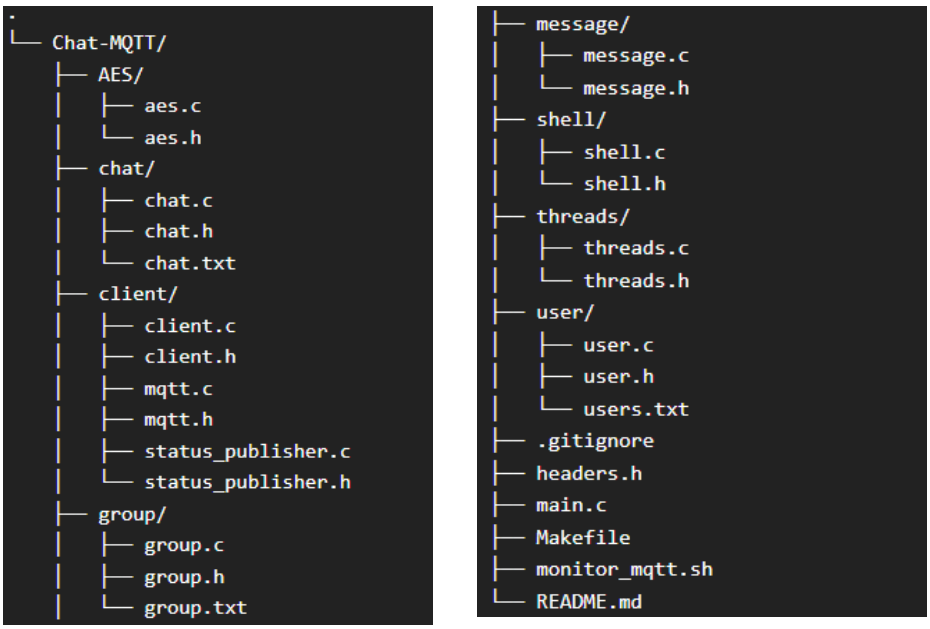
Na pasta *client* contém tudo o que precisa para o MQTT funcionar e suas derivações, a partir de suas funções básicas, como a de se conectar até a parte de enviar atualizações de conexões do usuário ao chat.

Em *group* estão as funções com as ações do grupo, o mesmo acontece para o diretório *user*, que contém as funções relacionados ao usuário.

Dentro do *message* pode-se dizer que está o *core* da aplicação, é a parte que tem as funções mais complexas relacionadas com o recebimento de mensagem, e o processamento das mesmas como a separação das mensagens de dados e controle e a execução de ações a depender do seu tipo.

Na *threads* seria os métodos de auxílio relacionado às threads, como a criação delas e dos mutex necessários para evitar a utilização de regiões críticas simultaneamente pelas threads.

E por fim a pasta *shell*, onde estão as funções de controle da interface, ou seja o recebimento de entradas do usuário direcionando para a ação solicitada, e a apresentação visual do menu e das conversas. Abaixo está uma imagem que representa a estrutura citada anteriormente:



2.2 TÓPICOS

2.2.1 USERS

Esse tópico tem como única e exclusiva função publicar o status do usuário, se ele está online ou offline, para controlar seu status.

2.2.2 GROUPS

Esse tópico tem como foco mandar os grupos criados, seus líderes(um por grupo) e seus participantes, mas também postar quem foi aceito ao grupo e quem foi recusado.

2.2.3 ID_CONTROL

É para receber as mensagens de controle específicas do usuário, ou seja, cada usuário tem o seu próprio tópico de montado com o *id_control*, onde o id é nome do usuário.

Ele tem como função receber os convites para grupos (quando o grupo foi criado ou quando um usuário pede para entrar no grupo, contanto que seja o líder), mas também tudo relacionados ao chat, o pedido de para iniciar o chat, se foi aceito ou recusado o pedido e a criação do tópico em si para se criar o objeto do chat.

2.2.4 CHAT PRIVADO

O tópico do chat privado é montado utilizando o nome do usuário que fez a solicitação, mais o usuário que aceitou e o timestamp do aceite, ou seja, User1_User2_timestam.

2.2.5 CHAT EM GRUPO

Esse tópico é montado com o nome do grupo indicado pelo usuário na hora de sua criação.

2.3 ESTRUTURA DAS MENSAGENS

A estrutura das mensagens é composta por duas fases. Inicialmente, o nome de usuário, o carimbo de data/hora (timestamp) e o conteúdo da mensagem são concatenados, utilizando o caractere '-' como separador, resultando no formato: *'nome_do_usuário-timestamp-mensagem*'. Na segunda fase, toda essa string, desde o nome do usuário até a mensagem, é criptografada. Em seguida, o prefixo 'ENC:' é adicionado antes da string criptografada.

Já a estrutura em si da parte da mensagem tem diversas variações, dependendo do tópico, a qual será enviada.

No tópico ***users***, a estrutura da mensagem possui duas variações: a ‘connected’ e a ‘disconnected’. Que servem para informar quando o usuário foi conectado ou desconectado do *chat*.

Para o tópico ***groups***:

- Criação/Atualização de Grupo: Group:nome_do_grupo;Leader:nome_do_lider;Participants:[participante_1;participante_2;];
    - Usado para anunciar grupos e seus participantes.
- Aceitação de Convite: 1;nome_do_grupo;veio_do_pedir_para_entrar;
    - veio_do_pedir_para_entrar = 0: O usuário que enviou a mensagem aceitou o convite do grupo.
    - veio_do_pedir_para_entrar = 1: O líder do grupo aceitou o usuário.
- Recusa de Convite: 2;nome_do_grupo;
    - Indica que o usuário que enviou a mensagem recusou o convite do grupo.

Tópico **id_control:**

- Convite de Grupo: 1;nome_do_grupo;
    - Exibe um convite para o grupo X.
- Convite de Conversa: 2;
    - Exibe um convite para iniciar uma conversa com o usuário do tópico id_control.
- Aceitação de Convite de Conversa: 3;
    - Notifica que o usuário aceitou o convite de conversa.
- Rejeição de Convite de Conversa: 4;
    - Notifica que o usuário rejeitou o convite de conversa.
- Solicitação para Entrar no Grupo: 5;nome_do_grupo;
    - Notifica o líder do grupo que um usuário deseja entrar no grupo X.
- Compartilhamento de Tópico de Conversa: 6;topico_da_conversa;
    - Compartilha o tópico da conversa com o usuário que aceitou o convite.

2.4 FUNCIONALIDADES IMPLEMENTADOS

2.4.1 GERENCIAMENTO DE PERSISTÊNCIA

O repositório possui persistência de dados tanto da biblioteca paho quanto do nosso “banco de dados” em arquivos do tipo .txt.

A persistência de dados utilizado da biblioteca paho seria o *cleanseasion = 0*, esse comando faz com que o usuário que tenha se inscrito em um tópico ao se reconectar no broker ele receba as mensagens que foram enviadas no tópico no período que o usuário foi desconectado.

Já a persistência usando arquivos do tipo .txt é dividida em duas, a parte da leitura e a escrita, a leitura somente acontece quando o usuário se conecta no broker, a qual ela lê os arquivos chat.txt, grup.txt e users.txt, esses arquivos são usados para serem adicionados nos objetos *Chat*, *Users*, *Group* e *Participant*. Já na parte da escrita, ela é usada sempre quando é criado um dos quatro objetos citados acima, as funções criam o objeto e logo em seguida são registradas em seus devidos arquivos .txt.

2.4.2 SEGURANÇA ENTRE MENSAGENS

O projeto tem como função secundária criptografar as mensagens enviadas para os tópicos e descriptografar ao recebê-las. Para fazer essa função foi necessário a utilização da biblioteca openssl que possui os métodos de criptografia e descriptografia.

O gerenciamento de segurança é centrado nas funções *aes_encrypt* e *aes_decrypt*, que implementam o algoritmo AES-256-CBC.

A função *aes_encrypt* utiliza uma chave secreta – *AES_KEY* – e um Vetor de Inicialização – *AES_IV* – predefinidos para converter as mensagens legíveis em dados criptografados.

Já a *aes_decrypt* usa exatamente a mesma *AES_KEY* e *AES_IV* para reverter os dados criptografados ao seu formato original. Esta função também verifica a integridade dos dados durante a finalização, garantindo que apenas mensagens válidas sejam lidas.

Todo o sistema de segurança se baseia no fato de que todos os clientes partilham essa mesma *AES_KEY* e *AES_IV* , que estão fixos no código, permitindo que leiam as mensagens uns dos outros.

2.4.3 GERENCIAMENTO DE CHAT

Para iniciar uma conversa, um usuário X deve primeiro enviar um convite (IDCONTROL_CHAT_INVITATION) para o tópico de controle privado do usuário Y (usuarioY_CONTROL). O Usuário Y recebe esta solicitação em seu menu "2 - Controle", onde pode aceitá-la ou rejeitá-la.

Somente após o aceite, um tópico de chat secreto e dinâmico é gerado (usuarioX_usuarioY_timestamp) pelo usuário X. O nome deste novo tópico é enviado de volta ao usuário Y, e ambos os clientes se inscrevem nele para começar a trocar mensagens.

2.4.4 GERENCIAMENTO DE GRUPOS

O gerenciamento de grupos é centralizado na figura do "Líder" do grupo, que é o usuário que o criou. O nome do grupo (com espaços substituídos por underscores) torna-se o tópico MQTT para a comunicação do grupo.

Existem duas formas de um usuário entrar em um grupo: sendo convidado pelo líder ao criar o grupo, ou solicitando acesso através do menu "7 - Pedir acesso ao grupo". No caso em que o usuário é convidado, o usuário é adicionado à lista de participantes com o status "pending" (pendente).

Todas as solicitações de entrada e convites são tratados através dos tópicos de controle – ID_CONTROL – do líder e dos usuários. A entrada no grupo só é finalizada quando o líder aprova o pedido ou o usuário aceita o convite no menu "2 - Controle", mudando seu status para "active".

2.4.5 INTERFACE

Para a Interface do sistema, optamos por utilizar uma interface com comandos por base do console, o que deixou o projeto visualmente menos apresentável. Então para deixar o console com uma estética mais bonita utilizamos inteligência artificial para configurar os nossos prints.

3. DESCRIÇÃO DOS PRINCIPAIS ASPECTOS DA IMPLEMENTAÇÃO

3.1. void on_recv_message() - O Roteador Central

Esta função, localizada em *message.c*, é o cérebro de toda a aplicação. Ela é chamada pelo callback *msgarrvd* , do mqtt.c, após uma mensagem ser recebida e descriptografada.

Sua principal responsabilidade é atuar como um roteador de mensagens. Ela analisa o tópico (topic) em que a mensagem chegou e decide qual módulo deve tratar a informação:

- Tópico **USERS**: A mensagem é um "pulso" de status. Assim é chamado *change_status()*, do user.c, para atualizar a lista de usuários online.
- Tópico **GROUPS**: A mensagem é um anúncio de grupo ou uma resposta de convite. A função chama *add_group_by_message()* ou atualiza os participantes na lista groups.
- Tópico **ID_CONTROL**: A mensagem é uma ação direta para o usuário (como um convite de chat ou grupo). A função não a processa imediatamente; em vez disso, ela a adiciona à fila *control_messages* para que o usuário possa respondê-la mais tarde através do menu "Controle", dizendo se aceita ou recusa a solicitação.
- Tópicos de **Chat:** Se a mensagem é de um chat privado ou em grupo, a função verifica a variável global *selected_chat*:
    - Se o usuário está com o chat aberto *(topic == selected_chat)*, a mensagem é impressa na tela imediatamente *(show_message_from_other)*.
    - Se o usuário está em outro menu, a mensagem é adicionada à fila *unread_messages* para notificação futura.

3.2. int send_message(...) e int msgarrvd(...) (Conexão com a rede)

Estas são as que implementam a comunicação com a rede de forma segura, responsáveis também pela criptografia.

send_message (Saída): Antes de publicar qualquer mensagem, esta função primeiro formata a mensagem concatenando usuário e timestamp com o payload, separando-os por ‘-’, então criptografa o resultado usando *aes_encrypt (AES-256-CBC)* e, por fim, adiciona o prefixo "ENC:" à mensagem criptografada.

msgarrvd (Entrada): Este é o callback da Paho que recebe os dados brutos da rede. Ele imediatamente verifica se a mensagem começa com "ENC:". Se não começar, a mensagem é descartada. Caso contrário, ele remove o prefixo e passa o payload criptografado para *aes_decrypt* antes de entregá-la para *on_recv_message*.

3.3. int create_threads() e void init_mutexes() (Unidades de execução)

Para que a interface não trave enquanto a rede está ouvindo, foi utilizada uma arquitetura multithread. O módulo *threads.c* é quem gerencia isso:

init_mutexes(): Inicializa todos os *mutex* como recursivos. Isso é vital para que as threads possam acessar as listas de dados globais de forma segura *(thread-safe)* sem causar deadlocks ou corrupção de dados.

***create_threads()**:* Esta função que efetivamente inicia cada uma das operações. Ela dispara duas threads de execução paralelas:

- *Thread* do **Shell** *(run_shell):* Responsável por ler a entrada do usuário no terminal (*fgets*).
- *Thread* de **Status** *(run_status_publisher)*: Envia uma mensagem de status "connected" para o tópico USERS a cada 20 segundos. Mostrando que o usuário está ativo
- Uma terceira *thread* (implícita), criada pela própria biblioteca Paho, roda em segundo plano para escutar a rede e disparar o callback *msgarrvd*..

3.4. void on_connect(...) (Gerenciador de Sessão)

Esta função de *callback* é chamada pela *Paho* toda vez que o cliente se conecta (ou reconecta) ao *broker*. Ela é essencial para o gerenciamento do estado da sessão.

Sua função é assinar (*subscribe*) dinamicamente todos os tópicos que o usuário precisa ouvir, assina os tópicos globais *USERS* e *GROUPS* e o tópico de controle pessoal (*<user_id>_CONTROL*).

Chama *subscribe_all_chats()*, que lê a lista chats (carregada do chat.txt) e se inscreve em todos os tópicos de chat privado e em grupo dos quais o usuário faz parte.

Isso, combinado com a configuração *cleansession* como 0, que garante que o usuário receba mensagens mesmo que tenha estado offline.

3.5. Funções de Persistência

A persistência de dados é gerenciada por arquivos .txt. Adicionar dados é fácil, mas modificar os existentes é complexo. Para resolver isso, usamos o método "ler-modificar-renomear": o arquivo original é lido, um temporário é criado, o conteúdo é copiado, a linha desejada é alterada no temporário, ambos são fechados, o original é excluído e o temporário é renomeado para o nome original, um exemplo desse necessidade é na hora de trocar um usuário pendente de um grupo para ativo.

4. INSTRUÇÃO PARA COMPILAÇÃO

Para a compilação do chat basta utilizar o comando make dentro da pasta do projeto, projeto disponível no repositório GitHub aqui.

Após a compilação, para rodar efetivamente o chat deve-se rodar o seguinte comando ‘./main nomeDoUsuario’, o nome do usuário pode ser tanto um usuário novo como um que já foi cadastrado anteriormente, para o segundo caso as mensagens e chats já registrados anteriormente serão recuperados.

5. UTILIZAÇÃO DO APLICATIVO

A utilização do chat ocorre por uma interface simplificada pelo próprio terminal, o chat inicia com um menu com 8 opções:

1. Entrar no chat - Esta opção apresentará as conversas privadas com usuários e os chats em grupo. Abaixo, haverá uma outra lista de todos os usuários com os quais ainda não há uma conversa iniciada; para esses, uma solicitação precisará ser enviada para iniciar um chat. Basta então digitar o número correspondente da conversa em que deseja acessar ou solicitar um pedido.
   OBS: ao entrar em uma conversa, para sair da mesma é necessário digitar **/quit**
2. Controle - Nesta opção será apresentada as mensagens de controle enviadas ao usuário, ou seja, solicitações para iniciar conversas ou para entrar em grupos. Para responder basta digitar o número da mensagem e depois pressionar a tecla enter digitar ‘s’, para aceitar solicitação, ou ‘n’, para negar a solicitação.
3. Listar mensagens - Lista as últimas mensagens de conteúdo recebidas.
4. Listar Usuários - Lista todos os usuários e ao lado se o usuário está on-line ou off-line
5. Listar grupos - Esta ação detalha todos os grupos existentes, apresentando o nome de cada grupo, seu líder e a lista de participantes. O status de cada participante é indicado como ‘*pending*’ se o convite para o grupo ainda estiver aguardando aceitação, ou ‘*active*’ se o participante já estiver apto a enviar mensagens.
6. Criar grupo - Para criar um novo grupo, digite o nome desejado. Em seguida, selecione os usuários que você quer convidar, digitando o número correspondente a cada um. Se quiser convidar mais de um usuário, separe os números com um espaço (exemplo: 1 2).
7. Pedir acesso ao grupo - Será listado todos os grupos existentes, então basta digitar o número do grupo que deseja entrar, e a solicitação será enviada ao líder.

0. Sair - opção para desconectar o usuário, em seguida aparece uma opção para reconectar o usuário, digite ‘s’ para reconectar e ‘n’, para então encerrar a aplicação.