Nícolas Ruwer Hackenhar

Implementação de aplicação de chat utilizando o protocolo MQTT

Docente: Profº Drº Marco Aurélio Spohn

Universidade Federal da Fronteira Sul

DESCRIÇÃO

O projeto consiste na implementação de uma aplicação de chat, utilizando o protocolo MQTT. Para a construção desta aplicação utilizou-se a linguagem de programação *Python* e a biblioteca *paho-mqtt,* conectando-se ao broker *mosquitto* em execução na mesma máquina (*localhost)*. A arquitetura do código consiste em um arquivo *main.py* que é o ponto de entrada e controlará as chamadas das demais classes para o funcionamento da aplicação, além do fornecimento das funcionalidades ao usuário através de um menu de seleção numérica. A conexão com o broker e a publicação e assinatura dos *tópicos* é gerenciada pela classe *Client*, que realiza a chamada das classes *StatusHandler*, responsável por lidar com as mensagens referentes ao controle de status dos usuários, *GroupHandler*, responsável por lidar com as mensagens referente ao controle de grupos, e *ChatHandler*, responsável por lidar com as mensagens relacionadas as conversas. Para que o usuário participante de uma conversa (um a um ou em grupo) receba as mensagens enquanto estiver *offline*, as mensagens do tópico em questão são publicadas com QOS 2, assim como a assinatura nestes tópicos é feita utilizando o mesmo mecanismo.

A conexão do *client* com o *broker* é efetuada utilizando a versão 5 do protocolo MQTT, definindo como *client_id* o nome informado pelo usuário, a *flag* de *reconnect_on_failure* como *False* para que em caso de desconexão não seja feito tentativas de reconexão (tratativa para que 2 usuários não utilizem o mesmo *client_id* simultâneamente em execuções paralelas da aplicação) e a flag *clean_start* como *False*, para que a sessão seja persistida e o usuário receba as mensagens publicadas com QOS 2 enquanto estava offline.

Com relação aos tópicos:

Ao logar, e deslogar no sistema o *client* publica uma mensagem a ser retida no tópico **USERS/{username}** informando o seu status, onde username representa o nome de usuário informado pelo usuário do chat. Para que seja possível obter os usuários e seus respectivos status, o *client* assina o tópico **USERS/#** de forma que sempre que o sistema é iniciado ele obtém as mensagens retidas de todos os usuários já registrados e adiciona a uma lista da classe *StatusHandler*, que é iterada para exibir os usuários e seus respectivos status quando essa funcionalidade é acessada através do menu.

Ao criar um grupo, o *client* publica uma mensagem a ser retida no tópico **GROUPS/{name}/owner**, identificando que o grupo foi criado e que seu dono é o usuário que está criando este grupo, onde *name* representa o nome do grupo que está sendo criado. Além disso, ao criar o grupo e sempre que um usuário é adicionado a esse grupo ou solicita o ingresso, é realizada uma publicação que

ficará retida no tópico **GROUPS/{name}/members/{member}**, onde *name* representa o nome do grupo e *member* o nome do usuário . No caso da solicitação de ingresso, a mensagem da publicação é efetuada com o campo *status* Pendente. Quando o administrador do grupo aceita essa solicitação, é feita uma nova publicação nesse mesmo tópico porém com o *status* Aprovado.

Para visualizar os grupos, seu dono e os demais membros, o *client* assina com QOS 2 o tópico **GROUPS/#** e como as mensagens são sempre publicadas como retidas, toda vez que o sistema é iniciado se obtém uma lista atualizada das informações, que são armazenadas em uma lista da classe *GroupHandler* para que possa ser iterada para visualização. A visualização é tratada conforme a funcionalidade que está sendo executada, por exemplo, se o usuário deseja gerenciar um grupo, são exibidos apenas os grupos dos quais ele é administrador e somente os membros que estão com *status* Pendente. Já na visualização de todos os grupos, são exibidos todos os grupos que estão cadastrados e somente os usuários com *status* Aprovado.

As mensagens trocadas entre os grupos são publicadas no tópico **GROUPS/{name}/messages** com QOS 2 e levam as informações do usuário que enviou, do conteúdo da mensagem e quando ela foi enviada. Ao receber uma mensagem deste tópico, a aplicação trata a mesma para validar se o usuário logado no chat é membro do grupo em questão e caso seja, as mensagens são atribuídas em uma lista para que possam ser iteradas e exibidas. Se o usuário está na funcionalidade de conversa (seja em grupo ou um a um) e a mensagem recebida é do grupo/usuário que está ativo no momento, as mensagens são exibidas assim que recebidas, limpando o console e imprimindo a lista novamente sempre que uma mensagem é recebida.

Para solicitar o início de uma conversa um a um, o *client* realiza uma publicação de mensagem retida para o tópico **CONTROL/{user_target}_Control/chat_requests/{user}** informando que deseja iniciar uma conversa com determinado usuário, onde *user_target* representa o usuário com o qual deseja se conectar e user o usuário que está efetuando a solicitação. Para aceitar uma solicitação, o *client* realiza duas publicações com a mesma mensagem retida para os tópicos **CONTROL/{user_target}_Control/chat_requests/{user}** e **CONTROL/{user}_Control/chat_requests/{user_target}** de forma que as mensagens retidas sejam atualizadas tanto para o usuário que solicitou o início da conversa quanto o usuário que aceitou a solicitação, e na mensagem enviada um campo *topic* define o tópico onde as mensagens dos usuários serão trocadas, seguindo a estrutura **X_Y_timestamp**. Para poder visualizar e tratar as solicitações de conversa e conversas ativas, o *client* assina o tópico **CONTROL/{user}_Control/chat_requests/#**, onde user representa o nome de usuário que o usuário informou ao logar no sistema. As mensagens trocadas entre

os usuários são publicadas no tópico definido ao ser aceita a solicitação de conversa, com QOS 2.

O tratamento das conversas ativas e das solicitações de conversas é feito por meio de um campo *status* nas mensagens publicadas. Ao iniciar a aplicação, pelo fato das mensagens serem publicadas com a *flag retain True*, o *client* recebe as mensagens e assina com QOS 2 os devidos tópicos que definem onde as mensagens trocadas entre os usuários serão publicadas, isso também é feito durante a execução da aplicação caso uma solicitação de conversa seja aprovada. Ao receber uma mensagem de algum tópico de conversa entre usuários, as mesmas são atribuídas em uma lista e exibidas conforme a lógica de chat descrita anteriormente.

Para executar a aplicação é necessário baixar a biblioteca do python, que pode ser feita através do comando *pip install requirements.txt*, no diretório do projeto. Com a biblioteca instalada e o serviço *mosquitto* em execução local na porta padrão, basta executar o comando *python main.py*. Ao iniciar a aplicação, será solicitado o nome de usuário que deseja se conectar ao chat e após informado, a conexão será estabelecida e terá um pequeno *delay* para que possam ser recebidas e tratadas as mensagens de controle. Após isso, o menu será exibido e o usuário poderá selecionar as opções informando o número da opção desejada. Conforme a lógica de cada opção, mensagens de instrução serão exibidas para obter as entradas do usuário.