Projecto 1 - Message Broker

Pedro Bastos – 93150 - P1

Eduardo Santos – 93107 - P1

Protocolo

Foi implementado um protocolo **PubSub** com 3 tipos de serialização (JSON, XML e Pickle) e com N producers e N consumers. O protocolo (TCP, porto 8000) faz uso de sockets com mensagens de 'accepted' e 'closing'. Posteriormente, tem mensagens de SUB de um consumer e de PUB de um producer. As mensagens de listagem de tópicos e de cancelamento de tópico também estão implementadas.

Estruturas de dados

No broker, são utilizados 3 dicionários. Primeiramente, o dicionário 'protocolCon' guarda pares de {conn, protocol}, isto é, para cada identidade (conn) guarda o seu mecanismo de serialização (JSON, XML ou Pickle).

Depois, o dicionário 'dicConsumers' guarda pares de {conn, topic}, isto é, para cada identidade guarda o seu tópico mais específico subscrito, ou seja, se o consumer estiver subscrito a weather, guarda {conn, weather}, se estiver subscrito a weather/temperature guarda {conn, temperature}. Está feito desta maneira porque posteriormente, ao ser publicado algo, percorremos os consumers e os tópicos em que foi publicada a mensagem. Ou seja, se for publicado no tópico weather/temperature, vai ser enviada para quem subscreveu weather e weather/temperature.

Finalmente, o dicionário 'content' guarda pares {topic, message}, isto é, para cada tópico guarda apenas a última mensagem que foi publicada. Assim, quando é publicado algo, guardamos a mensagem nos tópicos correspondentes e enviamos imediatamente a seguir, poupando espaço.

Fluxo/Arquitetura de software

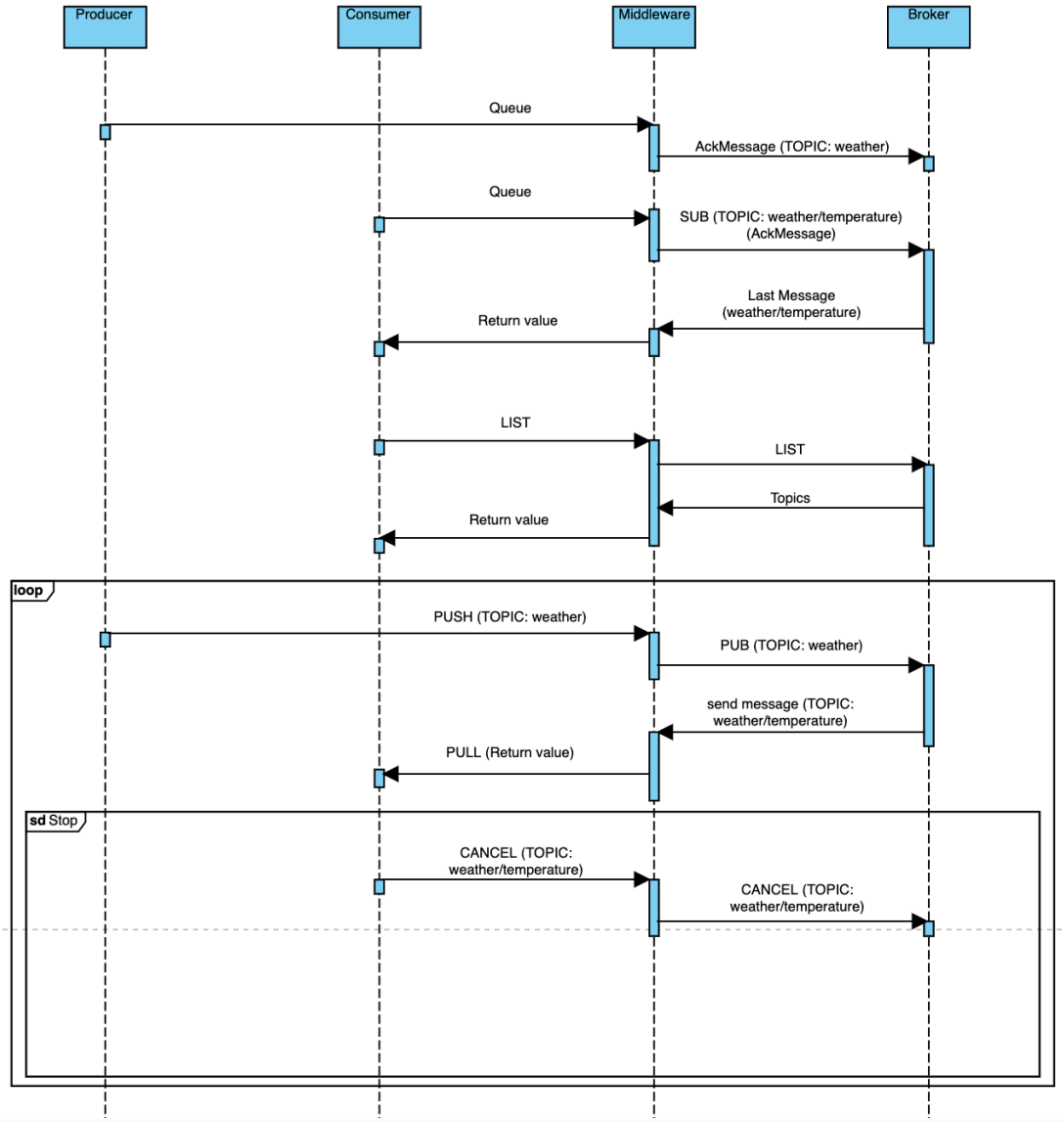
Quando um producer se liga, é enviada uma mensagem para o broker que dá a conhecer o seu tópico onde vai publicar e o seu tipo de serialização (JSON, XML ou Pickle). Quando um consumer se liga, é enviada uma mensagem idêntica para o broker. Alem disso, se houver, é enviada de volta para o consumer a última mensagem do tópico que se subscreveu. Está feito desta maneira pois assim o broker fica logo a conhecer as informações das entidades e pode logo guardá-las. Estas mensagens iniciais trocadas entre o middleware e o broker são feitas todas em JSON.

Depois do Acknowledge feito pelo broker, está pronto a funcionar. Cada vez que o producer publica algo, é enviada uma mensagem do middleware para o broker (METHOD='PUB') e este encontra os consumers subscritos naquele tópico e possiveis tópicos pais, enviando uma mensagem para os

consumers (no middleware) encontrados com a publicação do producer. O consumer, que está em loop a espera, recebe o data retornado pelo middleware da função ‘pull’.

As funcionalidades de listagem e cancelamento de tópicos estão apenas implementadas no middleware e no broker, não estando a ser usadas pelo consumer. Contudo, estão completas, fazendo o método ‘LIST’, é feito o print de uma mensagem com a lista de tópicos e fazendo o método ‘CANCEL’, é cancelada a subscrição do consumer e é feito o print do mesmo.

Depois do Acknowledge, todas estas mensagens trocadas entre o middleware e o broker têm verificação prévia de qual o mecanismo de serialização utilizado pelo consumer e producer, para ser feito os encodes e decodes nas linguagens certas.



Interface da listagem e cancelamento de tópicos

Estão implementadas duas funções (‘listTopics()’ e ‘cancelTopic()’) no middleware. A primeira, caso seja chamada, envia para o broker uma mensagem com o método ‘LIST’ e este devolve ao consumer a lista de todos os tópicos existentes. Na segunda, o middleware envia uma mensagem ao broker com o método ‘CANCEL’ e este retira a subscrição do consumer nesse tópico.

Estratégia de validação da solução

Foram feitos inúmeros testes à solução gerada com objetivo de nos certificarmos que estava funcional. Como a solução está feita para N consumers, N producers, 3 mecanismos de serialização e tópicos hierárquicos, testamos de várias maneiras.

Primeiramente, testámos todas as possibilidades de combinação dos mecanismos de serialização, ou seja, alternámos entre JSONQueue, XMLQueue e PickleQueue nos consumers e nos producers.

Depois de verificado isso, testámos os tópicos não hierárquicos (‘temp’ e ‘msg’) primeiro com 2 producers e 3 consumers (verificando que o consumer quando se liga recebe a ultima mensagem do tópico correspondente) e depois ao contrário, com 3 consumers e 2 producers (verificando que os consumers recebem todas as mensagens produzidas pelo producer correspondente).

Finalmente testámos os tópicos hierárquicos (weather). Foi adicionado no consumer e producer um tópico extra (weather/temperature/teste) para confirmar que a solução suporta profundidade crescente. Verificámos que publicações em weather/temperature/teste recebem os consumers que estão subscritos em weather, em weather/temperature e em weather/temperature/teste. Publicações em weather/temperature, apenas recebem os consumers weather e weather/temperature. E publicações em weather apenas consumers weather recebem. Assim, consumers weather recebem weather, weather/temperature, weather/humidity, weather/pressure e weather/temperature/teste. Por outro lado, consumers weather/temperature apenas recebem weather/temperature e weather/temperature/teste.

As mensagens de listagem e de cancelamento de tópicos foram testadas e estão funcionais. Contudo, não temos implementado nos consumers pois não nos cabe a nós essa parte, visto que o código dos producers e consumers vai ser mudado. O cancelamento de um tópico é também feito quando um consumer se desliga.