Universidade De Évora
Departamento de Informática
Redes de Computadores
Ano letivo 2019 - 2020

Trabalho Prático de Redes

Alunos:
Rúben Bravo - 37548
Rúben Teimas - 39868

Docentes:
José Saias
Pedro Patinho

12 de Janeiro de 2020

1 Introdução

Este trabalho consiste na implementação de um conjunto de aplicações que permita captar, processar e disponibilizar dados sobre a qualidade do ar.

As aplicações implementadas foram:
- O *broker* que é a aplicação mediadora de todo o sistema;
- O *sensor* que permite enviar dados sobre o ar ao *broker*;
- O *Public Client* que será utilizada pela população geral;
- O *Admin Client* que será utilizada pelos fabricantes dos sensores.

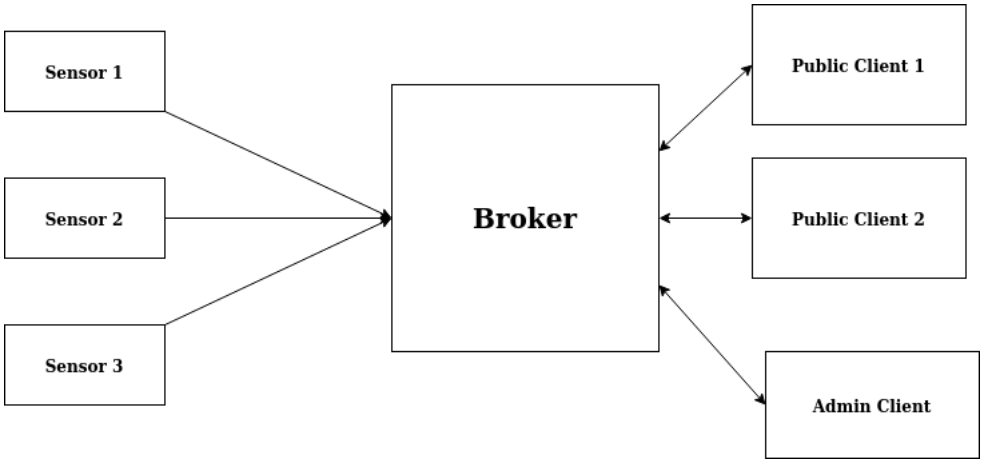

Diagram.png

Representação do Conjunto de Aplicações

2 Aplicações

2.1 Broker

Para guardar os sensores utilizamos dicionários devido à sua complexidade constante no acesso, não precisando assim de percorrer toda uma lista de sensores.

Utilizamos ainda outro dicionário, *LOCATIONLASTENTRY* em que a cada localização correspondia um tuplo com a sua última leitura, a lista de sensores que subscreveram esse local e um boleano, o que viria a facilitar a implementação de outra aplicação, onde iriamos obter a última leitura de um local.
No loop principal do broker utilizámos o código facultado no moodle para escolher o socket apropriado, dependendo da aplicação que estivesse a comunicar com o broker.
O broker, ao receber mensagens dos diversos clientes , irá verificar qual a primeira posição do tuplo recebido e encaminhar a informação recebida para a respetiva função.

No final do loop o broker irá verificar se existe alguma leitura nova num dos lugares subscritos pelos clientes, e caso exista, reencaminhar essa informação para os mesmos.

2.2 Sensor

O sensor irá inciar-se junto do broker enviando a informação necessária e após comuniar com o broker irá periódicamente, de 10 em 10 segundos, enviar-lhe informação sobre as leituras do ar.

Para o sensor optamos por criar 2 objetos com as respetivas informações. Um deles com a informação do sensor, o outro com a amostra de ar enviada pelo sensor ao broker.

2.2.1 Registar um sensor

O sensor irá enviar os seus dados para o broker, o broker ao receber o objeto serializado com o módulo pickle, irá descodificá-lo e caso o objeto seja do tipo *Sensor* e ainda não exista no dicionário de sensores irá guardá-lo.

Para além disto, o broker irá verficar se o localização do sensor com o respetivo ID existe no dicionário de localizações. Se não existir irá adicioná-la, caso contrário, não faz nada. Se já existir no dicionário dos sensores um objeto com esse ID, não acontecerá nada.

2.2.2 Enviar Periódicamente uma leitura

O sensor irá enviar de 10 em 10 segundos, leituras do ar ao broker. Ao recebê-los, o broker irá verificar se são do tipo *SensorData*. Se o sensor com o respetivo ID não existir, a leitura não será guardada. Caso exista, será guardada na Queue do respetivo sensor.

Após receber a leitura irá dar update da leitura no dicionário de localizações e colocar a flag dessa localização a True, de forma a confirmar que existe uma leitura ainda não lida.

2.3 Public Client

O Public Client utiliza 2 sockets, um para o *publish subscribe* e outro para as restantes operações. No loop principal o Public Client faz um uso do *select* para decidir qual dos sockets receber a informação.

2.3.1 Listar Locais onde Existem Sensores do tipo X

O broker irá percorrer o dicionário de sensores e verificar se o tipo do sensor é o mesmo do parâmetro passado pelo *Public Client*, se for, dá append numa lista auxiliar. No final, irá enviar essa lista ao *Public Client*.

2.3.2 Obter a Última Leitura de Um Local

O broker irá receber um local vindo do *Public Client* e irá obter a última leitura realizada nesse local, acedendo ao diconário de locais com a respetiva chave(localização enviada pelo *Public Client*). Essa leitura será enviada ao *Public Client* que a irá imprimir.

2.3.3 Modo Publish-Subscribe

Sempre que o broker receber uma nova leitura irá retornar o local dessa leitura e procurar pela mesma no diconário de locais. Irá aceder à lista de sensores subscritos dessa localização e enviar a leitura aos respetivos sensores da lista.

No *Public Client* existe uma flag SUBSCRIBED que verifica se o cliente já subscreveu uma localização ou não.

2.4 Admin Client

O Admin Client irá enviar ao broker um tuplo com 3 elementos, em que o primeiro elemento é sempre a letra "A"que representa as operações do admin.

2.4.1 Obter Última Leitura do Sensor X

O broker irá verificar se o ID do sensor existe no dicionário de sensores, caso exista irá aceder à Queue de leituras do mesmo e enviar a última leitura.

2.4.2 Listar Todos os Sensores

O broker irá enviar o dicionário de Sensores.

Ao receber a lista de todos os sensores, o tamanho da mensagem irá crescer e ficará maior do que a capacidade do buffer do socket, como tal, existe no *Admin Client* um loop para que toda a informação seja recebida. No socket colocámos um timeout para que após 0.1 segundos, se o buffernão receber mais informação, comece a tratar a mensagem e a imprimir todos o sensores existentes.

2.4.3 Enviar Ficheiro de Firmware

O broker irá receber o conteudo de um ficheiro com a versão de firmware. Irá de segudia percorrer o dicionário de Sensores e verificar se o respetivo sensor é do tipo lido do ficheiro e se a versão de firmware existente no sensor é menor do que a recebida. Caso estas condições se verifquem, a versão de firmware do sensor será atualizada e será criado um ficheiro com o mesmo nome que o ID do sensor e como conteúdo o tipo de sensor e a versão de firmware.

2.4.4 Desativar Sensores

O broker verificar se existe algum sensor no dicionário de sensores com o ID recebido e caso exista, irá removê-lo do dicionário

3 Decisões

Criámos um ficheiro *common.py* que contém os objetos e estruturas por nós criados e as constantes utilizadas ao longo do programa como o *HOST* e a *PORT*.

Para além dos objetos *Sensor* e *SensorData* implementámos um objeto *subscribedLocation* que contém uma leitura, uma lista de sockets que subscreveram esse local e um boleano.

Além dos objetos criados e das estruturas do Python implementámos ainda uma simples Queue de 10 elementos. Embora pudessemos ter usado uma das queues do Python, achámos que tal não se justificava.

Para serializar objetos e estruturas optamos por utilizar o módulo pickle.

4 Balanço

Por implementar ficou a *obtenção de uma leitura para uma data e hora*. Não totalmente funcional ficou também o *envio de ficheiro de firmware para o sensor*. Conseguimos encaminhar o ficheiro para o broker e estando lá, achámos que apenas seria preciso selecionar o socket (como fizemos no *publish-subscribe*), contudo, se precisássemos de encaminhar o ficheiro para mais que um sensor obtinhamos um erro e por isso optámos por eliminar esse pedaço de código. O outro "erro"que se destaca é na verdade um descuído, pois ainda que eliminemos um sensor e não guardemos mais leituras do mesmo, nunca chegamos a "matar"o socket, continuando ele a mandar informação que será ignorada.
Inicialmente pensámos que se o socket mandasse informação para um sensor que não existia no dicionário, iriamos enviar uma mensagem de volta ao sensor para que ele saísse do loop e encerrasse, mas acabamos por não implementá-lo com sucesso.

Ainda que com um melhor planeamento o trabalho pudesse ter uma melhor estrutura, achamos que o objetivo foi parcialmente conclúido com sucesso.