Transmissão de Dados – Turma A – Trabalho 1
Professor: Marcos F. Caetano <mfcaetano@unb.br>
Monitores: Matheus Stauffer <matheusvostauffer@gmail.com> e Rafael Amaral Soares <amarals16@hotmail.com>
Resumo: Implementação de uma "rede de sensores" utilizando o protocolo MQTT.

1 Introdução

Para os dispositivos de Internet das Coisas (IoT), a conexão com a Internet é um requisito importante. A conexão com a Internet permite que os dispositivos trabalhem entre si e com serviços de *backend*. O protocolo de rede subjacente da Internet é o TCP/IP. Desenvolvido com base na pilha TCP/IP, o MQTT (*Message Queue Telemetry Transport*) tornou-se o padrão para comunicações de IoT.

O MQTT foi proposto e implementado, inicialmente, pela IBM no final dos anos 90. Sua aplicação original era vincular sensores em pipelines de petróleo a satélites. Como seu nome sugere, ele é um protocolo de mensagem com suporte para a comunicação assíncrona entre as partes. Um protocolo de sistema de mensagens assíncrono desacopla o emissor e o receptor da mensagem tanto no espaço quanto no tempo e, portanto, é escalável em ambientes de rede que não são confiáveis. Apesar de seu nome, ele não tem nada a ver com filas de mensagens, na verdade, ele usa um modelo de publicação e assinatura. No final de 2014, ele se tornou oficialmente um padrão aberto OASIS, com suporte nas linguagens de programação populares, usando diversas implementações de software livre.

O protocolo MQTT possui diversos tipos de mensagens que viabilizam a comunicação entre cliente e servidor. Em sua versão 5.0, o protocolo MQTT define 15 mensagens distintas. Em [1] podemos encontrar uma descrição detalhada do protocolo. Utilizando o padrão *publisher-subscriber* não é necessário um conhecimento de quem receberá as mensagens. Neste tipo de arquitetura, *Publisher* refere-se à transmissores de mensagens que não conhecem previamente os nós quem irão consumir esta informação. As mensagens são categorizadas, o que facilita aos *subscribers* o consumo desta informação. Os *subscribers* podem se interessar pelo tópico registrado e se inscrever para receber apenas informações sobre determinada classe (categoria). É importante destacar que um *publisher* não precisa conhecer um *subscriber* para que aconteça a comunicação. Da mesma forma, um *publisher* pode publicar uma mensagem, sobre determinado tópico, e nenhum *subscriber* receber essas informações (por não haver *subscriber* registrado nesta categoria). Outro ponto a ser observado é que um nó pode desempenhar tanto o papel de *publisher* quanto de *subscriber*.

De uma maneira mais informal, o padrão utilizado equivale ao seu despertador quando é necessário acordar cedo. Pode-se fazer isso de duas maneiras: você pode acordar a cada minuto, checar o horário e voltar a dormir caso não seja a hora correta (algo que certamente muitos alunos devem odiar) ou programar o seu celular para lhe dizer o momento correto de levantar. Nesse último caso, todos os *assinantes* (os integrantes do mesmo quarto) irão receber o sinal de alerta.

2 Descrição e Implementação

O objetivo deste trabalho é aplicar o conhecimento adquirido em sala sobre o funcionamento de redes de computadores através da construção de uma arquitetura cliente servidor utilizando o protocolo MQTT. Para tal utilizaremos um cenário (virtual) de IOT. Aqueles que desejarem utilizar algum sensor como o DHT11[2] podem fazê-lo desde que apontem no relatório o sensor e o código utilizado para implementação. Vale lembrar que não é permitida a utilização de bibliotecas prontas. Outro ponto é que o servidor deve permitir a publicação de vários tópicos por vários clientes. Portanto, faz-se necessário o uso de mais alguns sensores. Além disso, não será necessário algum mecanismo

de autenticação ou cifragem de informação. Estes tópicos não foram abordados na disciplina e estão fora do escopo deste projeto.

2.1 Funcionamento Básico

De maneira geral, a arquitetura básica a ser implementada é descrita pela Figura 1. De acordo com esta arquitetura, *Broker* é um *middleware* que trata as conexões e recebe as mensagens encaminhadas pelos dispositivos. Um dispositivo pode ser um sensor reportando as medições efetuadas, quanto um cliente se registrando para receber notificações de mensagens enviadas pelos sensores. O *broker* é responsável por encaminhar aos clientes (*subcribers*) registrados, todas as mensagens encaminhadas pelos sensores a uma categoria de informação específica. O *broker* é uma peça fundamental para o funcionamento do protocolo e é por onde recomenda-se começar o trabalho devido a sua complexidade.

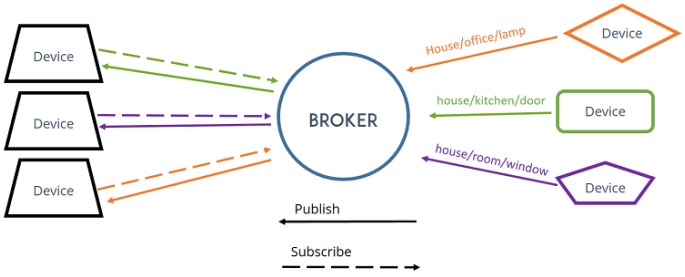

Figura 1: Arquitetura simples do protocolo MQTT.

A Figura 2 apresenta um exemplo da troca de mensagens utilizando o protocolo MQTT. Na figura temos um *broker* e dois clientes. O cliente 1 apresenta o comportamento de um *subscriber*, enquanto que o cliente 2 o comportamento de um sensor (*publisher*). Quando um cliente (cliente 1) deseja se registrar a um *Broker*, ele transmite uma mensagem do tipo CONNECT[1]. Essa mensagem contém informações que são necessárias para a identificação única do cliente. Após receber uma confirmação do tipo CONNACK, o cliente pode se inscrever em determinado tópico, utilizando uma mensagem do tipo SUBSCRIBE. Ao confirmar a inscrição com uma mensagem do tipo SUBACK, o cliente passará a receber todas as mensagens encaminhadas ao *broker*, encaminhadas pelo sensor (cliente 2) e que estejam relacionadas ao tópico registrado pelo cliente.

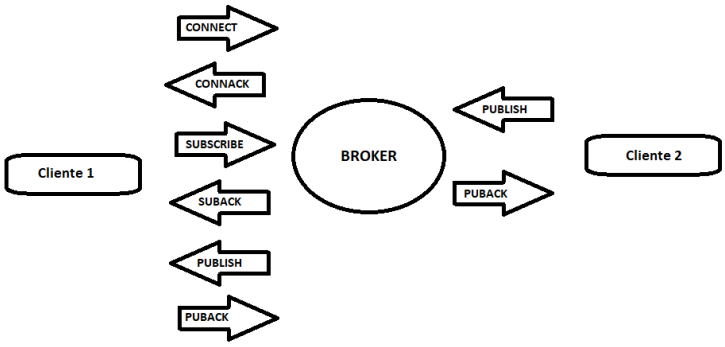

Figura 2: Passos básicos para o funcionamento do trabalho.

É interessante destacar que a frequência na troca de mensagens entre um cliente e o *broker* pode gerar *timeout* na conexão TCP. Para resolver este problema, é previsto o envio de mensagens de controle periódicas, definida como PINGREG e PINGRESP (Figura 3). Tais mensagens servem para os seguintes propósitos: indicar que o cliente está ativo quando não ocorrem troca de mensagens;

[1]É importante destacar que o protocolo MQTT utiliza os protocolos TCP e IP nas camadas inferiores.

indicar que o servidor (*broker*) ainda está ativo e manter a conexão de rede ativa. **Ao implementar o seu broker, utilize esse recurso para manter a troca de mensagens ativa.**

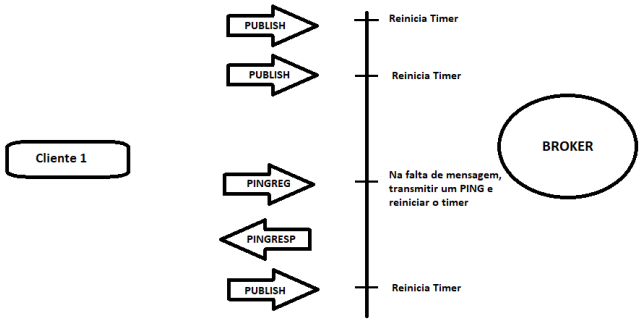

Figura 3: Exemplo no uso de mensagens do tipo PINGREG e PINGRESP.

Outra funcionalidade a ser implementada, é a possibilidade que vários clientes podem conectar ao *broker*. Para isso, Utilize o campo **Client ID** para identificação única. As informações a serem coletadas e enviadas ao *broker*, pelos sensores, deverão ser carregadas no seu programa como um parâmetro de entrada. Este parâmetro de entrada será fornecido no formato de um arquivo texto. O tempo entre duas publicações sucessivas deve ser passado como parâmetro no momento da criação de um *publisher*.

Outro tópico a ser abordado na implementação é utilização de mensagens do tipo DISCONNECT para desvincular um cliente do *broker*. Esse mecanismo deve ser implementado tanto no *broker*, abrindo possibilidade para desconectar mais de um assinante, tanto no cliente. A qualquer momento deve ser possível desconectar um cliente. Investigue em [1] como esse mecanismo pode ser implementado utilizando um *Reason Code* apropriado.

OBSERVAÇÃO:

É importante destacar que o trabalho consiste na implementação do *broker*, dos clientes (*subscribers*) e dos sensores (*publishers*). Os *subscribers* irão se registrar no *broker* em tópicos específicos. As medições feitas pelos sensores (*publishers*) e enviadas ao *broker*, serão encaminhadas aos *subscribers* segundo o registro dos tópicos específicos efetivados.

Não será aceito o uso de código pronto de *broker*, *subscriber* ou *publisher*. Somente o uso de biblioteca que implemente a troca de mensagens no formato MQTT será permitida (Protocolo Camada de Aplicação). Dúvidas, sobre o que pode ou não utilizar, devem ser enviadas diretamente aos monitores da disciplina, via fórum do Moodle.

3 Relatório

Um relatório final do projeto deve ser apresentado. Este relatório deve conter:

- Apresentação teórica sobre o protocolo MQTT, incluindo comparação com HTTP. Exemplos deverão ser apresentados;
- Documento apresentando a arquitetura do sistema desenvolvido;
- Explicação da arquitetura produzida e da relação entre seus principais componentes;
- Documentação de todo o código desenvolvido;
- Instruções para compilação/execução do código e a relação de todas as bibliotecas utilizadas;
- *Screenshots* e explicação do funcionamento das funcionalidades implementadas;

4 Avaliação

A avaliação consiste em 2 etapas:

- Código e funcionamento do projeto
  - O trabalho deve ser desenvolvido **OBRIGATORIAMENTE** utilizando-se de uma ferramenta de versionamento *online* (GitHub, BitBucket, etc). Um dos monitores da disciplina deverá ser adicionado ao repositório do trabalho quando o mesmo estiver concluído;
  - O código e o relatório deverão ser enviados pelo Moodle até o **dia 01/12/2019**, às 23:55h;
    - Este é um prazo firme, **não serão aceitos trabalhos enviados fora do prazo**;
  - A apresentação do trabalho deverá ser agendada pelo fórum da disciplina, respeitando os horários que serão disponibilizados;
    - Os dias 03/12 e 05/12, no horário da aula, estão reservados para apresentação dos trabalhos;
    - Trabalhos não apresentados não serão considerados;
- Nota do Relatório

5 Observações

As seguintes observações deverão ser consideradas pelos alunos que forem fazer o trabalho:

- O trabalho deve rodar **OBRIGATORIAMENTE** na plataforma GNU/Linux;
- O programa pode ser feito em qualquer linguagem;
- Códigos copiados serão considerados "cola"e todos os alunos envolvidos ganharão nota zero;
- Dúvidas sobre o trabalho deverão preferencialmente ser tiradas **utilizando a lista de email da disciplina**. Desta forma, dúvidas comuns e esclarecimentos poderão ser respondidos uma única vez para toda a turma.

Referências

[1] "MQTT Version 5.0"; OASIS Standard.. Disponível em: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html. Acesso em 27 de outubro de 2019.

[2] "Sensor de umidade e temperatura DHT11"; Arduíno e CIA. Disponível em https://www.arduinoecia.com.br/sensor-de-umidade-e-temperatura-dht11/. Acesso em 27 de outubro de 2019.