UTFPR

Projeto Cidades Inteligentes

Documentação v1.0.0

Apresentação do projeto, protocolos, convenções e justificativas.

Elias Biondo
Julio Cesar Garcia Ribeiro
Prof. Dr. Fernando Barreto
Prof. Dr. Luiz Fernando Carvalho

Sumário

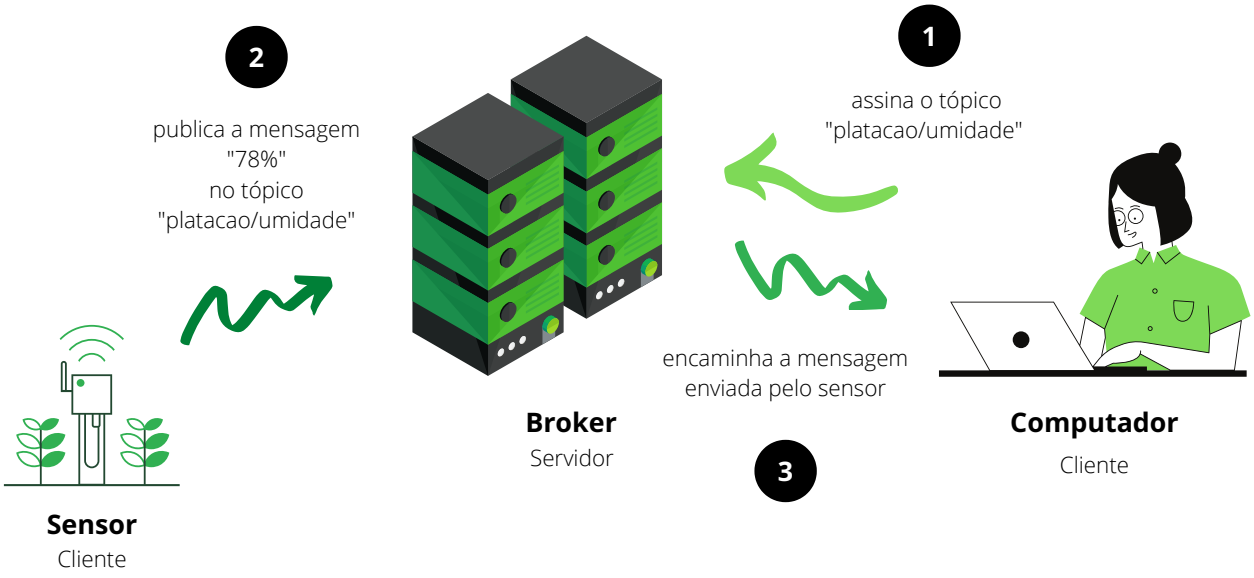
1. INTRODUÇÃO

O projeto "Cidades Inteligentes" visa, por meio da tecnologia, propor soluções inteligentes que objetivam causar impacto no desenvolvimento urbano e na qualidade de vida da população local, dando maior autonomia às decisões públicas através de estudos especializados. Alinhado ao poder público, esse projeto pode promover um aumento expressivo na qualidade de vida da população local: o desenvolvimento de aplicações no formato Internet das Coisas pode auxiliar no avanço econômico, na educação, na saúde, na segurança, na mobilidade e sustentabilidade de uma cidade. Um estudo feito pela divisão de pesquisa da multinacional francesa Capgemini aponta que 73% dos estrevistados que usaram iniciativas de cidades inteligentes estão mais felizes com sua qualidade de vida, sobretudo em relação a questões de saúde, como a qualidade do ar e segurança pública. Esse resultado vai ao encontro da crença de 60% dos cidadãos entrevistados, que acreditam que cidades inteligentes são sinônimo de sustentabilidade e melhores serviços. Abaixo seguem algumas das aplicações de soluções inteligentes que podem ser implementadas em uma cidade objetivando o seu desenvolvimento:

- **Iluminação pública inteligente**, capaz de se adequar às condições de iluminação natural, além utilizar dados de movimentação de pedestres e veículos nas vias.
- **Estacionamento público inteligente**, capaz de detectar vagas disponíveis em grandes centros e diminuir a emissão de carbono pelos carros e o stress diário da população ao procurar uma vaga.
- **Agendamento online de atendimento para serviços públicos**, disponível à população, de modo a facilitar os trâmites públicos e diminuir a burocracia envolvida no processo. Agendamento desde serviços relacionados à prefeitura, como também à unidades básicas de saúde ou emergências clínicas de nível azul ou verde (não urgentes ou pouco urgentes).
- **Monitoramento urbano**, voltado à segurança pública, de modo a responder à acidentes, furtos e roubos de maneira rápida e eficiente.
- **Mobilidade urbana**, com ônibus inteligentes com rotas monitoráveis e compartilháveis e que por isso, incentivam o uso do transporte público, diminuindo o fluxo de automóveis particulares e a poluição emitida por eles.
- **Publicidade urbana**, através de totens digitais espalhados pela cidade que serviriam também, como ponto de acesso a internet gratuita e à serviços públicos de modo a democratizar o acesso à tecnologia por parte das populações mais carentes.

2. O PROTOCOLO MQTT

O protocolo MQTT, acrônimo para Message Queuing Telemetry Transport (transporte de telemetria de enfileiramento de mensagens) é um protocolo de comunicação máquina para máquina desenvolvido, inicialmente, para conectar sistemas de telemetria de oleodutos via satélite. Esse protocolo atua no transporte de mensagens cliente/servidor de forma assíncrona no formato publicação/assinatura. Uma das principais características desse protocolo é que as transmissões das mensagens são feitas em pacotes de dados extremamente pequenos, desacoplando o emissor do receptor através de um agente intermediário chamado broker, sendo portanto, muito escalável em infraestruturas de comunicação não confiáveis ou instáveis. Voltado às entregas das mensagens em quaisquer que sejam as condições, esse protocolo atua através de níveis de serviço, o que poupa recursos e garante a entrega dos pacotes de dados. O protocolo MQTT atua sobre o protocolo TCP/IP e outros protocolos de rede que forneçam conexões bidirecionais ordenadas sem perdas. Por possuir essas características, muitos profissionais da área de internet das coisas optam por trabalhar com essa tecnologia, que auxilia no desenvolvimento e na conexão de aplicações inteligentes com a Internet. Segue abaixo um diagrama que ilustra uma comunicação feita através do protocolo MQTT:



2.1 PRINCIPAIS CONCEITOS

Nesta seção serão definidos os principais conceitos e abstrações que regem o supracitado protocolo.

Cliente (client)

Publicante ou assinante que envia ou recebe mensagens do servidor.

Servidor (server)

Máquina sobre qual roda o broker, responsável por proporcionar a comunicação entre os clientes.

Agente (broker)

Agente responsável por administrar a comunicação entre os clientes; aplicação no servidor responsável por receber e distribuir as mensagens dos publicantes aos assinantes.

Publicante (publisher)

Cliente que atua no envio de dados.

Assinante (subscriber)

Cliente que atua no recebimento de dados.

Mensagem (message)

Pacote de dados trafegado entre cliente e broker.

Conteúdo (payload)

Conteúdo da mensagem; propriamente os dados enviados nela.

Tópico (topic)

Repositório de dados único responsável por receber as mensagens, acessível aos clientes, instantâneo e sem histórico.

Nível do serviço (QoS)

Acordo feito entre rementente e destinatário qual define a garantia de entrega de uma mensagem.

Última vontade (will message)

Mensagem definida pelo cliente no momento de sua conexão a ser enviada pelo broker quando o mesmo se desconectar involuntariamente.

Tempo de sobrevivência (keep alive)

Intervalo de tempo máximo permitido, definido pelo cliente no momento de sua conexão, de não-comunicação entre cliente e servidor.

Sessão (session)

Intervalo de tempo no qual o cliente encontra-se conectado ao broker.

Sessão limpa (clean session)

Tipo de sessão definida pelo usuário no momento de sua conexão que não exige o recebimento de possíveis mensagens enviadas pelo broker enquanto estava desconectado.

Sessão persistida (persistent session)

Tipo de sessão definida pelo usuário no momento de sua conexão que exige o recebimento de possíveis mensagens enviadas pelo broker enquanto estava desconectado.

Mensagem retida (retained message)

Tipo de mensagem única enviada por um cliente que deve ser retida pelo broker e enviada a outros clientes imediatamente após a sua conexão.

2.2 MOSQUITTO

Eclipse Mosquito™ é um broker MQTT open-source e gratuito. Atualmente, o mais pupular do mercado. "[...] O Mosquitto é leve e adequado para uso em todos os dispositivos, desde computadores de placa única de baixa potência até servidores completos" (MOSQUITTO, 2021).

2.2.1 INSTALAÇÃO

O Mosquitto pode ser instalado em diferentes sistemas operacionais e distribuições. Nesse documento abordaremos a instalação em distribuições linux que utilizam o sistema Debian como base. Para um guia completo, acesse https://mosquitto.org/download.

Em um terminal, execute os seguintes comandos:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install mosquitto
sudo apt-get install mosquitto-clients
```

Para testar o sucesso da instalação, (1) em um terminal, inicie o servidor

```
mosquitto
```

(2) em um outro terminal, inscreva-se em um tópico de teste

```
mosquitto_sub -t 'topico_de_teste'
```

(3) em um outro terminal, publique no tópico de teste

```
mosquitto_pub -t 'topico_de_teste' -m 'Hello, World!'
```

Se tudo estiver funcionando bem, a mensagem 'Hello, World!' aparecerá no terminal inscrito no tópico 'topico_de_teste'.

2.2.2 ARQUIVO DE CONFIGURAÇÃO

O Mosquitto, em distribuições linux com base no sistema Debian, por padrão, é instalado no diretório '/etc/mosquitto'. Nessa pasta é possível encontrar um arquivo denominado **mosquitto.conf**. Esse arquivo é responsável por definir todas as configurações que regem o referido serviço. É importante destacar também que ele não necessariamente precisa estar na pasta de instalação do Mosquitto. Por padrão, o Mosquitto não necessita de um arquivo de configuração. Para uma documentação completa, acesse https://mosquitto.org/man/mosquitto-conf-5.html.

Todas as linhas iniciadas com o caractere '#' são tratadas como comentários, e as linhas de configuração iniciam com o nome de uma variável seguido pelo seu valor separado por um espaço em branco.

2.2.3 PRINCIPAIS CONFIGURAÇÕES

Nesta seção serão definidas as principais configurações do Mosquitto.

acl_file file_path

propriedade que determina o caminho da lista de controle de acesso.

allow_anonymous [true | false]

valor booleano que determina se os clientes que solicitam conexão sem fornecer um nome de usuário têm permissão para se conectar.

password_file file_path

propriedade que determina o caminho do arquivo de senha; se a propriedade allow_anonymous for definida como false, apenas os usuários definidos neste arquivo poderão se conectar.

max_keepalive value

propriedade que sobrescreve o valor máximo de keep alive setado pelo cliente no momento de conexão; são aceitos valores entre 10 e 65535 segundos.

persistent_client_expiration duration

propriedade que permite que clientes persistentes (aqueles com sessão limpa definida como falsa) sejam removidos se eles não se reconectarem dentro de um determinado período de tempo. O período de expiração deve ser um inteiro seguido por 'h', 'd', 'w', 'm' ou 'y' para hora, dia, semana, mês e ano, respectivamente.

retain_available [true | false]

valor booleano que determina o suporte à mensagens retidas; se definido como falso, os clientes que enviarem uma mensagem com a bandeira 'retain' ativa serão desconectados.

2.2.4 AUTENTICAÇÃO

Existem três métodos diferentes para fazer a configuração de autenticação no Mosquitto, são eles: **arquivo de senha**, **plugin de autenticação** e **acesso anônimo**. Para fazer a leitura da documentação completa e na íntegra, acesse https://mosquitto.org/documentation/authentication-methods/. Neste documento trataremos sobre a autenticação anônima e a autenticação com arquivo de senha. Você também deve considerar alguma forma de controle de acesso para determinar quais clientes podem acessar quais tópicos.

2.2.4.1 ANÔNIMA

Para ativar a autenticação anônima, ou seja, permitir que qualquer usuário conecte-se sem fornecer um usuário ou senha, deve-se localizar o arquivo de configuração do Mosquitto (mosquitto.conf), geralmente em '/etc/mosquitto', e definir a propriedade '**allow_anonymous**' como '**true**'. O seu arquivo de configuração deve conter a seguinte linha:

```
allow_anonymous true
```

2.2.4.1 COM ARQUIVO DE SENHA

Os arquivos de senha são um mecanismo simples de armazenamento de usuários e senhas em um arquivo único. Eles são bons se você tiver um número relativamente pequeno de usuários razoavelmente estáticos. Para ativar a autenticação com arquivo de senha, ou seja, permitir que somente usuários cadastrados conectem-se, deve-se localizar o arquivo de configuração do Mosquitto (mosquitto.conf), usualmente em '/etc/mosquitto', e definir a propriedade 'allow_anonymous' como 'false' e a propriedade 'password_file' com o diretório que aponta para o arquivo de senha, criado em seguida. O seu arquivo de configuração deve conter as seguintes linhas:

```
allow_anonymous false
password_file /etc/mosquitto/password_file
```

Após isso, deve-se criar um arquivo de senha utilizando o utilitário **mosquitto_passwd** conforme a sintaxe abaixo:

```
mosquitto_passwd -c <password_file> <username>
```

A propriedade **-c** indica que um novo arquivo de senha será criado no diretório atual com o nome indicado na propriedade **password_file**. Caso já exista um arquivo com o mesmo nome no mesmo diretório, ele será sobrescrito. Após o envio do comando, será necessário preencher a senha que deseja-se associar ao nome de usuário definido na propriedade **username**.

Caso deseje-se adicionar um outro usuário a um arquivo já existente, deve-se utilizar a seguinte linha de comando:

```
mosquitto_passwd <password file> <username>
```

Caso deseje-se remover um usuário existente, deve-se utilizar a seguinte linha de comando:

```
mosquitto_passwd -D <password file> <username>
```

2.2.5 CORINGAS

Coringas são caracteres especiais utilizados para generalizar o acesso a um grupo de tópicos específicos. Nessa seção abordaremos dois desses coringas.

Coringa de nível único (+)

Como o próprio nome sugere, um curinga de nível único substitui um nível de assinatura em um tópico. O símbolo '+' representa um curinga de nível único em um tópico. Imagine que se tenha a seguinte estrutura de tópicos em uma casa inteligente:

```
casa/sala/sensor
casa/quarto/sensor
casa/cozinha/sensor
jardim/irrigador
jardim/lampada/status
jardim/sensor/umidade/valor
```

Poderíamos, por exemplo, para assinar todos os sensores da casa, utilizar o coringa '+', ao invés de assinar tópico a tópico, conforme a sintaxe abaixo:

```
casa/+/sensor
```

Coringa de múltiplos níveis (#)

Diferentemente do curinga de nível único, o coringa de múltiplos níveis substitui todos os níveis de assinatura em um tópico que seguem a sua declaração. Por isso, ele deve ser sempre utilizado no final de uma especificação de acesso. Utilizando o mesmo exemplo de estruturas de tópicos do item anterior, imagine que se queira assinar todos os tópicos existentes na região do jardim. Isso poderia ser feito tópico a tópico, mas daria um trabalho maior do que assinar os mesmos tópicos utilizando o coringa de múltiplos níveis confome abaixo:

```
jardim/#
```

2.2.6 CONTROLE DE ACESSO

Para além da autenticação, também é possível controlar quais usuários podem se inscrever ou publicar em determinados tópicos. Isso é feito através de um arquivo denominado de **lista controle de acesso** (ACL), cujo diretório é definido pela propriedade 'acl_file' no arquivo de configuração do Mosquitto conforme explicitado na seção 2.2.3 do presente documento. Abaixo segue um exemplo de uma lista de controle de acesso.

```
// Isso afeta o controle de acesso de usuários não autenticados
topic read jardim/irrigador

// Isso afeta o controle de acesso de usuários com o nome 'abc'
user abc
topic readwrite casa/+/sensor

// Isso afeta todos os clientes
pattern readwrite casa/%u/#
```

A primeira parte do arquivo é reservada ao controle de acesso de usuários não autenticados. Permissões a usuários específicos são sempre precedidas pelo padrão **user <username>** seguidas das regras de controle de acesso para àquele usuário em específico. Patterns são aplicados a todos os usuários, e suportam o uso de novos coringas: %u (nome do usuário) e %c (id do cliente). O uso dos coringas anteriormente conhecidos é suportado em qualquer estrutura de tópicos na lista de acesso de controle, seja para usuários anônimos, usuários autenticados ou patterns.

Basicamente a sintaxe para instanciar uma regra a um determinado usuário e tópico em específico em uma lista de acesso de controle é dada da seguinte forma:

```
user <username>
topic [read|write|readwrite] <topic>
```

onde <username> representa o nome do usuário alvo, [read|write|readwrite] as permissões concedidas (inscrever-se, publicar, ou ambas, respectivamente) e <topic> representa o tópico alvo da consessão de permissão.

2.2.7 TLS

Nesta seção aprenderemos a como configurar o Mosquitto para a comunicação com suporte ao protocolo de segurança TLS (Transport Layer Security) de encriptação de dados. Para utilizar o TLS entre o broker e o cliente, um conjunto de chaves e certificados deve ser gerado e implantado, junto às definições de configuração do servidor. Para gerar essas chaves e assinar os certificados utilizaremos o pacote **openssl**. Caso a máquina utilizada ainda não possua o supracitado pacote instalado, o download e a instalação do mesmo pode ser feita atraves do seguinte link: https://www.openssl.org/. Com o pacote anteriormente mencionado já devidamente instalado, siga os procedimentos abaixo:

No servidor, em um terminal, execute a seguinte linha de comando:

```
sudo openssl genrsa -des3 -out ca.key 2048
```

Será solicitada uma senha para que o certificado seja gerado com sucesso. Certifique-se de elaborar uma palavra-passe forte e guardá-la em um lugar seguro. Execute então a seguinte linha de comando no mesmo diretório:

```
sudo openssl req -new -x509 -days 1826 -key ca.key -out ca.crt
```

Novamente será solicitada uma senha para que o certificado seja gerado com sucesso. Digite a mesma senha do passo anterior e tecle 'enter' para continuar. Serão então solicitados alguns dados pessoais/organizacionais para a incorporação na requisição do certificado (nome do país, estado, cidade, nome da organização/entidade, unidade da organização, nome comum e endereço de e-mail). Atente-se ao campo nome comum (common name): esse campo deve ser preenchido obrigatoriamente com o endereço IP ou domínio do servidor em questão. Em seguida, execute a seguinte linha de comando:

```
sudo openssl genrsa -out server.key 2048
```

Será então criado um arquivo nomeado como 'server.key'. No mesmo diretório execute a seguinte linha de comando:

```
sudo openssl req -new -out server.csr -key server.key
```

Algumas informações serão solicitadas novamente. Preencha com os mesmos dados pessoais/organizacionais disponibilizados anteriormente (nome do país, estado, cidade, nome da organização/entidade, unidade da organização, nome comum e endereço de e-mail). Dois novos atributos serão solicitados, são eles, uma senha de desafio e um nome de companhia opciciaol. Deixe ambos em branco e pressione a tecla 'enter' para continuar. Em seguida, execute a seguinte linha de comando para assinar e validar o certificado do servidor com a chave da autoridade certificadora:

```
sudo openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt -days 360
```

Movamos então, através do terminal, três dos seis arquivos gerados, conforme abaixo:

```
sudo mv ca.crt ../../etc/mosquitto/ca_certificates
sudo mv server.crt ../../etc/mosquitto/certs
sudo mv server.key ../../etc/mosquitto/certs
```

Por fim, adicionemos as seguintes linhas ao arquivo de configuração do Mosquitto:

```
listener 8883
certfile /etc/mosquitto/certs/server.crt
keyfile /etc/mosquitto/certs/server.key
cafile /etc/mosquitto/ca_certificates/ca.crt
```