Ambiente de experimentação para avaliação protocolos de mensagem para IoT na Fog

Wesley R. Bezerra, Carlos B. Westphall

[1]INE – Universidade Federal do Santa Catarina (UFSC)
Florianópolis – SC – Brazil

{wesleybez,carlosbwestphall}@gmail.com

Abstract. *Choose a message protocol to use in an IoT application involves several factors and a number of experiments. This work aims to share information about setting up an experimentation environment for IoT message protocols. It provides a description of the technologies used in programming, such as brokers and support for the generation of graphics. It also provides a narrative on challenges and choices in creating the experimentation environment. At the end it is possible to use the experience brought here to help in the creation of an experimentation environment, with some challenges already overcome.*

Resumo. *A escolha de qual protocolo de mensagem usar em uma aplicação IoT envolve vários fatores e certo número de experimentações. Este trabalho visa compartilhar informações sobre a montagem de uma ambiente de experimentação para protocolos de mensagem em IoT. Traz um descrição das tecnologias utilizadas na programação, como brokers e suporte a geração de gráficos. Também traz um relato sobre os desafios e escolhas na criação do ambiente de experimentação. Ao fim é possível utilizar-se a experiência aqui trazida para ajudar na criação de um ambiente de experimentação, já com alguns desafios vencidos.*

1. Introdução

Internet of Things (IoT) é um tema que tem ganhado muita atenção dos pesquisadores. Suas tecnologias estão fragmentadas em diversas áreas do conhecimento, o que pode ser um desafio[Dizdarević et al. 2019]. Aliado este tema, temos paradigmas emergentes de computação como a computação em nevoeiro (FG-*Fog Computing*) que traz a proposta de aproximar recursos computacionais, armazenamento dos dispositivos finais (da borda)[Iorga et al. 2018], não sendo um substituto para a *cloud*, mas um complemento[Ahmed and Rehmani 2017]. A computação de borda é um novo foro para implantação de aplicações IoT novas e compreensíveis[Sha et al. 2019].

Devido a ainda ser uma área em desenvolvimento, existem poucos simuladores e ambientes de experimentação para soluções de IoT. Um desenvolvedor de aplicações que anseie validar uma tecnologia, protocolo ou biblioteca tem a tarefa árdua de implementar seu experimento do zero, sem contar com a experiência de outras desenvolvedores que já tiveram problemas parecidos.

Este trabalho tem dois objetivos:

i. descrição das tecnologias utilizadas no ambiente de experimentação;

ii o relato da experiência e desafios encontrados no ambiente criado.

O artigo organiza-se da seguinte maneira. A segunda seção trás um descritivo sobre as tecnologias utilizadas para a criação do ambiente de experimentação. Na terceira seção temos um relato sobre a experiência de criação do ambiente e seus desafios. Por fim, quarta seção, são expostas uma conclusão do artigo e trazidas propostas futuras de trabalho.

2. Ambiente de experimentação

Foram realizadas quatro fases de desenvolvimento das análises nos protocolos de mensagens listados neste trabalho (ver Figura 1). Na primeira, fase (a), foram elegidas e configuradas as tecnologias e ferramentas utilizadas, Tabela 1. Depois, na fase (b), foram construidos os *scripts* de avaliação e de geração dos gráficos. Na fase (c), foram gerado os *datasets* com o resultado dos *scripts* executados de avaliação. Por último, na fase (d), foram gerados os gráficos a partir dos *datasets*. Os arquivos estão disponíveis em um repositório no GitHub[1].

Figura 1. Fases do desenvolvimento

Quanto a fase de experimentação, foram propostos dois casos de testes a serem exceutados para cada protocolo. Os casos diferenciam-se entre o uso de autenticação ou sem autenticação, e com ou sem uso de SSL. Os casos são:

i caso sem autenticação e sem SSL;
ii caso com autenticação e sem SSL;
iii caso sem autenticação e com SSL;
iv caso com autenticação e com SSL;

Em alguns caso é possível a utilização de *containers* de micro serviços para execução das ferramentas. Foram avaliadas diversas ferramentas de *container*: Tomato, DDWRT e Docker. Esse último foi o escolhido para fazer parte do conjunto de tecnologias do experimento. Como sistema operacional da imagem a ser executada, foi selecionado o OpenWRT que é um *firmware open source*[Pa et al. 2015] para roteadores e permite atualizações *on-the-fly*[Palazzi et al. 2010]. O OpenWRT tem uma *interface web* que permite sua administração, assim como uma *interface* ssh. Através de seu utilitário foi possível instalar pacotes necessários para o experimento. A escolha de um *firmware* de roteadores é devida ao seu alinhamento com a computação de borda.

Os testes foram executados em um ambiente simulado e os códigos implementandos em Python 3.7 e 2.7 utilizando-se a IDE Eclipse. Foram utilizandos tanto a versão 3.7, quanto a versão 2.7 do Python devido à compatibilidade de algumas bibliotecas utilizadas na implementação. A versão do Eclipse 2019-03 foi utilizada, rodando em um

[1]https://github.com/wesleybez/wpeif2020

notebook com processador Intel Core i7-4510U e 16GB de memória RAM com o sistema operacional Ubuntu 18.04.03 LTS.

As bibliotecas Python utilizadas para execução dos protocolos AMQP, MQTT e STOMP foram respectivamente Pika, Paho e Stomp.py, as quais foram instaladas diretamente pelo utilitário PIP[2], não sendo recompiladas ou mesmo customizadas na sua utilização. Se faz importante comentar que todas as bibliotecas utilizadas são *open source* e estão disponíveis para *download*.

Como *broker* foi utilizado o RabbitMQ para o AMQP que é uma ferramenta com versão comunitária. O Mosquitto que é um *broker open source* muito utilizado tanto pela comunidade acadêmica como na execução de projetos MQTT reais. Por último, para o STOMP foi utilizado o *broker* CoilMQ que é um servidor de código aberto e implementado em Python, mesma linguagem do código de testes. Todas as opções são acessíveis e disponíveis para utilização na replicação deste experimento.

Tecnologia	Tipo	Protocolo	Versão
Python	Linguagem	NA	2.7 e 3.7
Eclipse	IDE	NA	2019-03
Ubuntu	SO	NA	18.04.03 LTS
Pika	Biblioteca	AMQP	1.1.0
Paho	-	MQTT	1.5.0
Stomp.py	-	STOMP	4.1.22
RabbitMQ	*Broker*	AMQP	3.8.2
Mosquitto	-	MQTT	1.6.2
CoilMQ	-	STOMP	1.0.1

Tabela 1. Versões de bibliotecas e *brokers* utilizados nos casos de testes

A autenticação utilizada em cada caso é a baseada em usuário/senha. Tal autenticação demanda menos processamento por parte das ferramentas de *broker* durante a verificação da existência do usuário e da corretude dos dados. A ferramenta RabbitMQ fornece um ambiente *web* para manutenção de algumas configurações, incluindo a autenticação; foram criados dez usuários no mesmo padrão dos demais *brokers* e com acesso à publicação no *virtual host* "\". O *broker* Mosquitto foi instalado na máquina citada e as configurações puderam ser feitas de maneira mais facilitada: foi criado um arquivo com dez usuários/senhas, para as senhas foram gerados *hashs* (através do utilitário `mosquitto_passwd` [3] . Para o CoilMQ foram utilizados os mesmos usuários e senhas gerados, no entanto o *broker* STOMP citado não utilizava *hash* para as senhas.

Cada execução de um caso de testes gerou um arquivo de dados onde cada linha representa uma *thread* de execução, com seus dados separados por vírgulas. O arquivo gerado foi no padrão CSV[4] e com a codificação ASCII[5]. Isto foi importante para garantir a compatibilidade e tratabilidade dos dados em qualquer ferramenta que permita abrir esses tipos de padrões. Sendo tais padrões clássicos e bem conhecidos na comunidade

[2]instalador de pacotes Python

[3]é o utilitário de manutenção de senhas do Mosquitto

[4]*Comma Separeted Value* - valores separados por vírgula (tradução livre)

[5]*American Standard Code for Information Interchange* - código americano para troca de informações

acadêmica e no tratamento de dados, foi possível eleger de maneira mais facilitada as ferramentas utilizadas no tratamento dos dados.

Como ferramenta para geração dos gráficos foi utilizada a ferramenta *open source* GnuPlot. Esta é uma ferramenta que permite a criação de *scripts* para geração dos gráficos, configurações do arquivo de entrada: tipo de arquivo lido, seu separador, entre outras; e configurações dos arquivos de saídas: formatos, legendas e tipos de gráficos gerados. Também é possível através de programação extrair e plotar, se necessário dados estatísticos, como médias, máximos, mínimos, desvio-padrão, entre outros. Permite exportação em diversos formatos, como pdf, svg, png; formatos esses integráveis a ferramentas como Latex [Racine 2006].

3. Discussão e resultados

Nesta seção será apresentado algumas das dificuldades e realizações obervadas durante o processo de criação dos ambiente de experimentação. Foi utilizada uma ilustração do processo e seus componentes, Figura 2, que ilustra a execução de um experimento projetado para este ambiente.

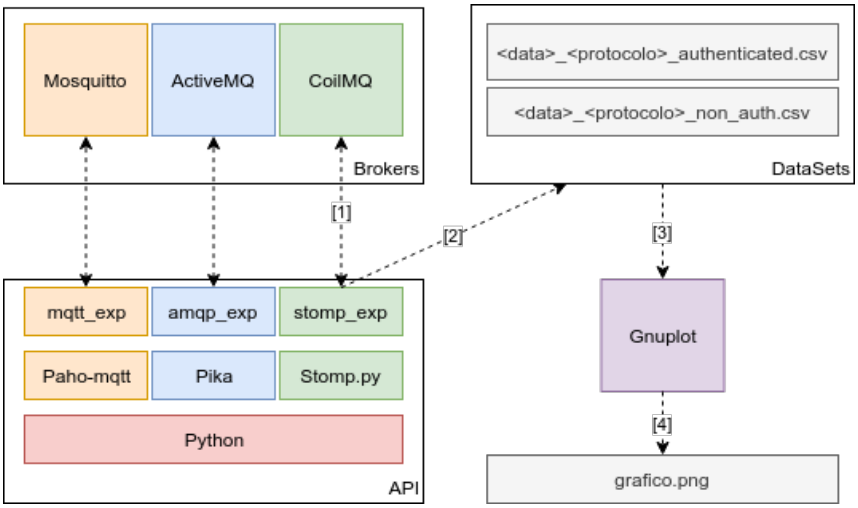

Figura 2. Ambiente de experimentação utilizado

Tomemos com exemplo a execução de um experimento com STOMP com autenticação e sem SSL. Primeiramente, passo (1), o experimento conecta-se com o *broker*, neste caso o CoilMQ. Ao final do conjunto de linhas de execução do experimento, passo (2), os dados são salvos em um *dataset*. Os arquivos gerados, passo (3), servem de entrada para o Gnuplot, o qual é responsável pela geração dos gráficos. Ao gerar os gráficos, passo (4), a ferramenta combina *datasets* para comparação de dados e gera os gráficos no formato PNG.

Durante a comunicação com o *broker*, o experimento executa os seguintes passos, sempre nesta ordem: (i) conexão com o *broker*, (ii) publicação de dados, (iii) desconexão. A conexão é parametrizada utilizando-se um arquivo de configurações dentro do ambiente de experimentação e a cada linha de execução, uma nova conexão é estabelecida e finalizada ao término do *script*. O número de linhas de execução também é parametrizado por configuração, desta forma é possível adequar o número de solicitações ao serviço durante cada execução.

Os *datasets* gerados são nomeados com o seguinte padrão **data_protocolo_authenticated.csv**. Este padrão permite que os arquivos possam ser agrupados por data e protocolo utilizado no experimento. Isto permitiu que, em caso de experimentos com diferentes opções de segurança em um mesmo momento, os arquivos permaneciam próximos e ordenados, facilitando assim sua posterior manipulação.

A geração de gráfico com o Gnuplot nos permite incorporar múltiplas fontes de dados. Em nosso caso, foi comparado o desempenho entre todos os protocolos de mensagem em três situações: autenticado sem SSL e não autenticado sem SSL, e durante a publicação de dados. Com somente um *script* foi possível comparar os três protocolos entre sí, nas três situações diferentes. Também foi gerado um gráfico para cada protocolo em cada situação para permitir-se comparar o mesmo protocolo em distintas configurações de segurança.

Sobre a utilização de *microcontainers*, houveram alguns problemas práticos durante a sua adoção. Não foi possível utilizar-se a imagem do OpenWRT para todos os protocolos. O caso que teve mais fácil adoção com Docker/OpenWRT foi o MQTT com Mosquitto para o qual a instalação pode ser feita utilizando-se o utilitário de instalação do próprio OpenWRT. O caso mais difícil e para o qual não conseguimos uma solução adequada ainda foi o AMQP. Com a utilização do RabbitMQ como *broker*, se fez necessário a instalação de algumas ferramentas a quais o OpenWRT não tinha suporte. Também não foi possível achar uma imagem adaptada para rodar RabbitMQ em tempo ábil. Devido a isto, foi utilizada uma imagem do RabbitMQ com Docker, mas não a com OpenWRT.

A criação deste ambiente de experimentação permitiu um maior conhecimento das tecnologias utilizadas na IoT e seus desafios de um ponto de vista prático. Ao testar-se diferentes bibliotecas para os protocolos de mensagem, notou-se que algumas não possuiam suporte as versões mais recentes da linguagem de programação adotada. Também na escolha do *firmware* para utilização com o Docker houve um esclarecimento sobre qual é mais utilizado na prática para soluções IoT e mais aceito pela comunidade de programadores de soluções. Por último, a adoção da ferramenta Gnuplot, permitiu acesso a geração de gráficos a partir de um ponto de vista de programação, evitando-se recorrer a ferramentas de planilhas ou ferramentas estatísticas. Tais ferramentas podem ser um entrave quando não fazem parte da rotina do desenvolvedor do experimento, o que poderia levar a uma curva maior de aprendizagem para criação do experimento.

4. Conclusão e trabalhos futuros

Este estudo conclui-se trazendo a possibilidade de uma pequena empresa na área de IoT, ou uma equipe de programação, ou ainda um pesquisador da área; possa criar um ambiente de experimentação inicial para protocolos IoT. Foi possível uma análise sobre as tecnologias utilizadas na construção e configuração do ambiente proposto. Por último trouxe um relato sobre os desafios e escolhas realizadas durante o processo.

Como trabalhos futuros, propomos um refatoramento no código-fonte para trazer uma melhor manutenibilidade. A criação de uma *interface* para execução dos *scripts* de experimentação e uma maior automação nas partes do processo. Sendo este um trabalho inicial, vemos que existem muitas possibilidades de melhorias a ser implementadas no futuro próximo.

Referências

Ahmed, E. and Rehmani, M. H. (2017). Mobile edge computing: opportunities, solutions, and challenges.

Dizdarević, J., Carpio, F., Jukan, A., and Masip-Bruin, X. (2019). A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration. *ACM Computing Surveys (CSUR)*, 51(6):116.

Iorga, M., Feldman, L., Barton, R., Martin, M. J., Goren, N. S., and Mahmoudi, C. (2018). Fog computing conceptual model. Technical report.

Pa, Y. M. P., Suzuki, S., Yoshioka, K., Matsumoto, T., Kasama, T., and Rossow, C. (2015). Iotpot: analysing the rise of iot compromises. In *9th {USENIX} Workshop on Offensive Technologies ({WOOT} 15)*.

Palazzi, C. E., Brunati, M., and Roccetti, M. (2010). An openwrt solution for future wireless homes. In *2010 IEEE International Conference on Multimedia and Expo*, pages 1701–1706. IEEE.

Racine, J. (2006). gnuplot 4.0: a portable interactive plotting utility. *Journal of Applied Econometrics*, 21(1):133–141.

Sha, K., Yang, T. A., Wei, W., and Davari, S. (2019). A survey of edge computing based designs for iot security. *Digital Communications and Networks*.