Avaliação de desempenho de protocolos de mensagens com arquitetura *publish/subscribe* no ambiente de computação em nevoeiro: um estudo sobre desempenho do MQTT, AMQP e STOMP

Wesley R. Bezerra, Carlos B. Westphall

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

{wesleybez,carlosbwestphall}@gmail.com

Abstract. *Secure solutions for IoT and agribusiness are critical. However, it is necessary to adopt the appropriate technology for the development of systems in this area. With a focus on pub/sub message protocols, our work provides an overview of the message protocols AQMP, MQTT and STOMP, a use case description of security application in agribusiness and an experiment to evaluate the temporal performance of the mentioned protocols. Finally, the results are discussed and it is verified which protocol is the most suitable for situations analogous to the mentioned use cases.*

Resumo. *Soluções seguras para IoT e agronegócios são primordiais. Entretanto se faz necessária a adoção da tecnologia adequada para o desenvolvimento de sistemas nesta área. Com foco nos protocolos de mensagem pub/sub, nosso trabalho traz uma visão geral dos protocolos de mensagem AQMP, MQTT e STOMP, uma descrição de caso de uso de aplicação de segurança no agronegócio e uma experimentação para avaliação do desempenho temporal dos protocolos citados. Ao final, são discutidos os resultados e avaalido qual protocolo é o mais indicado para as situações análogas aos casos de uso citados.*

1. Introdução

O crescimento da adoção da *Internet* das Coisas (IoT - *Internet of Things*), no dia-a-dia das pessoas, tem exposto tanto os benefícios de tal tecnologia quanto os problemas de segurança [Yi et al. 2015, Aazam and Huh 2014, Bertin et al. 2019, Dizdarević et al. 2019] em diversas áreas. Uma dessas áreas é o agronegócio, onde a IoT tem sido utilizada para com várias finalidades, como: melhoria da gestão e rastreabilidade no setor agropecuário, melhor utilização de recursos no setor de agricultura de precisão, na gestão de estufas, entre outros. Tais aplicações demandam tecnologias seguras e que garantam a integridade dos dados transmitidos. Também deve garantir a proveniência dos dados a partir de sensores reais e pertencentes ao sistema.

Um desenvolvedor de soluções que integram IoT e agronegócios deve levar em conta fatores de segurança nos protocolos utilizados para mensagens de dados. Os protocolos de mensagem podem ser utilizados para comunicar tanto dispositivo-dispositivo, como dispositivo-*fog node*. Se faz relevante uma análise dos protocolos existentes, suas características e tempo de processamento de cada passo da sua utilização. Este último

item é de suma importância devido aos dispositivos utilizados serem restritos quando a energia e transmissão de dados[Deep et al. 2019].

Os protocolos de mensagem analisados são três: AMQP[1], MQTT [2] e STOMP [3]. Com um escopo reduzido para protocolos que tenham uma arquitetura *publish/subscribe* e com o *broker* centralizado, nosso trabalho foca nos três mais proemientes protocolos de mensagens. Existem atualmente algumas pesquisas comparando protocolos de mensagem [Zorkany et al. 2019, Naik 2017, Zhou and Zhang 2014, Luzuriaga et al. 2015, Aloufi and Alhazmi 2020, Sueda et al. 2019], entretanto, tais pesquisas tendem a concentrar esforços MQTT, AMQP e CoAP[4], excluindo o protocolo STOMP.

Dentro de tal perspectiva, este trabalho tem como três objetivos: (i) apresentar o cenário e casos de uso com vulnerabilidades no agronegócio, (ii) apresentar uma visão geral dos protocolos citados e (iii) avaliar o desempenho temporal em três diferentes protocolos de mensagens, utilizando diferentes configurações de segurança.

Na seção dois são apresentados o cenário e os casos de uso em agronegócios. Na terceira seção tem-se uma visão geral dos protocolos de mensagens. Logo após, seção quarta, apresenta a experimentação e seus resultados. Já na seção cinco, propõe-se uma discussão dos resultados obtidos. Por último, temos a conclusão, resultados atingidos e propostas futuras.

2. Casos de uso e ambiente de experimentação

Analisemos um cenário aplicável a agropecuária, especificamente a pecuária intensiva (confinamento). Soluções para alimentação automática em agropecuária são comuns e utilizam-se de balanças para pesagem do animal, comedouro, *dispenser* e brinco RFID para identificação do animal. Tais ferramentas liberam a quantidade de alimento adequada para cada indivíduo e de acordo com a dieta desejada. Esta automação considera restrições alimentares devido a qualquer tratamento veterinário.

Neste caso o adversário pode ser modelado como o criador de gado concorrente e que deseja que seus animais tenham melhor aspecto para participação de feiras e leilões. Através de ataques de segurança ao sistema de alimentação, o adversário pode trazer problemas para a engorda da propriedade detentora do sistema. Caso um adversário queria atacar o confinamento pode utilizar-se o ataque MITM[5]. Para um melhor entendimento foram desenvolvidos casos de uso que podem denotar o quão prejudicial é a falta de segurança ou o baixo desempenho de um protocolo nas dadas situações:

1. **Alterando o conteúdo das mensagens enviadas**, o atacante faz que o sistema registre a medição para o animal errado, ou ainda, faz que o sistema libere menor quantidade de alimento, o que levaria o animal subnutrição;
2. Ainda com MITM, é possível realizar o ***replay*** da mensagem e liberar alimento no coxo sem que este tenha nenhum animal a espera, desperdiçando o alimento;

[1] *Advanced Message Queuing Protocol*
[2] *Message Queuing Telemetry Transport*
[3] *Simple Text Oriented Messaging Protocol*
[4] *Constrained Application Protocol*
[5] *Man-In-The-Middle*

3. Por último, um **atraso na decisão de liberação** do alimento é igualmente problemático, tendo em vista que o sistema deve atender vários animais e que por vezes passam de centenas de unidades.

A autenticação é uma das soluções para o problema de MITM acima citado, mas o impacto de sua implantação de segurança afeta o desempenho do sistema (caso de uso 3). Para análise de qual o melhor protocolo para a resolução de problemas como o acima citado, foi desenvolvido o experimento. Neste experimento, foram propostos dois casos de testes (com autenticação e sem autenticação) a serem executados para cada protocolo, totalizando seis casos de testes. Para simulação foram utilizadas bibliotecas que encapsulam cada protocolo de mesagem [Stomp.py , Paho-mqtt , Pika], a linguagem escolhida para desenvolvimento foi Python, o ambiente de desenvolvimento foi o Eclipse e a geração dos gráficos foi feita com o Gnuplot. Os *datasets* e *scripts* para geração de gráficos estão disponíveis no GitHub (https://github.com/wesleybez/wpeif2020).

3. Visão geral sobre protocolos de mensagem de arquitetura *pub/sub*

Com dois níveis de QoS, *unsettle format* e *settle format* - que são respectivamente o confiável e o não confiável, o AMQP vem ganhando espaço dentro dos protocolos de mensagens em IoT [Naik 2017], tendo sua origem em uma empresa do mercado financeiro, a JPMorgan em 2003. O AMQP é o mais novo dentre os protocolos abordados neste estudo. Pode utilizar a arquitetura de *pub/sub* ou a requisição/resposta para publicação e consumo de dados. O AMQP tem três tipos de troca de mensagens: *fan-out*, tópicos e baseado em cabeçalhos. A segurança no AMQP acontece tanto no transporte com SSL/TLS, quanto na autenticação com o uso de SASL (*Simple Authentication and Security Layer*[6]). O SASL permite suporte a diferentes métodos de autenticação, assim como também adiciona suporte à integridade de dados.

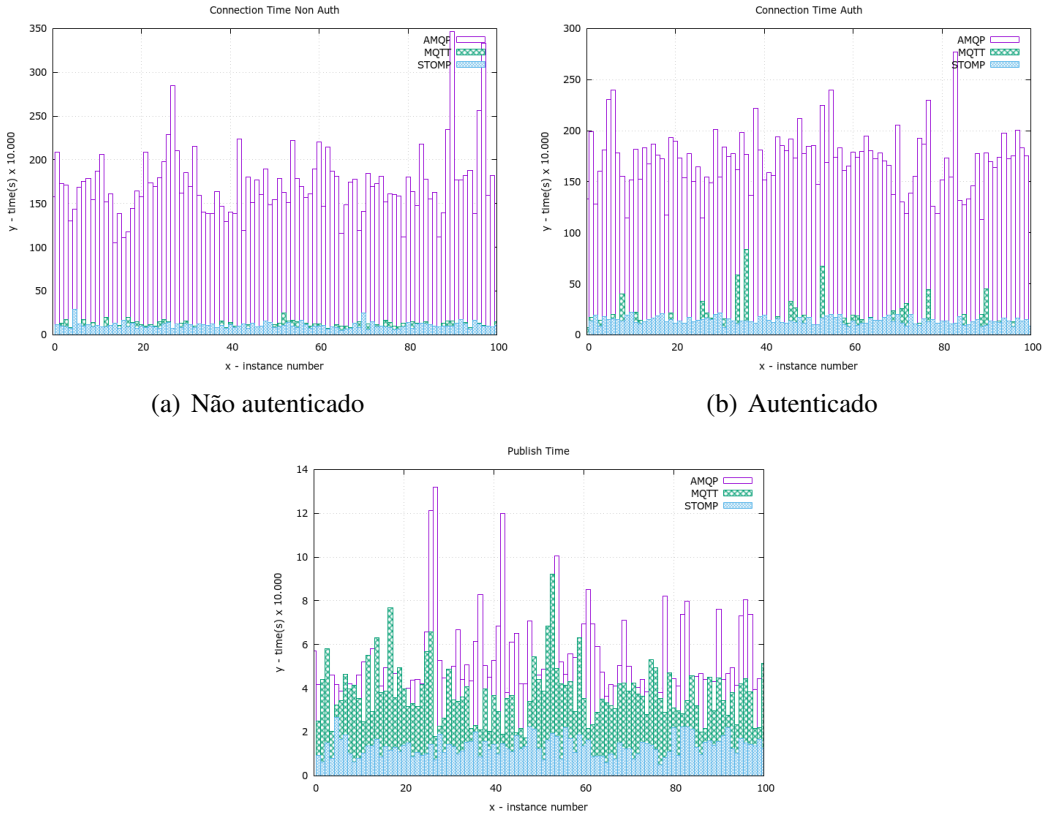
O MQTT é um protocolo para comunicação máquina-a-máquina utilizado para telemetria e apresenta-se como um protocolo leve de mensagens [Soni and Makwana 2017]. É um dos protocolos mais antigos, disponível desde 1999, e tem grande utilização pela industria, assim como grande suporte de organizações [Naik 2017]. Ele está na versão atual 5.0, entretanto o experimento foi realizado com a versão 3.1, suportada tanto pela biblioteca utilizada quanto pelo *broker*. Seu protocolo é binário, implementa QoS em três níveis e requer um componente centralizado, o *broker*.

O STOMP é um acrônimo para, em português, protocolo de mensagem orientado a texto simples [Stomp b]. Está na versão 1.2, que foi lançada em 2012 e adicionou modificações no cabeçalho, conexões, entre outros. É um protocolo modelado em *frames* HTTP [Stomp a] e que mesmo sendo um protocolo textual, possibilita o envio de *frames* conteúdo binário [Szydło et al. 2013]. Implementa *frames* como CONNECT, SEND, MESSAGE, entre outros. Este protocolo tem por filosofia a simplicidade e interoperabilidade [Stomp a]. Devido à proposta de simplicidade, a sua implementação em diversas linguagens de programação é fácil, tanto a implementação do servidor, quanto a implementação do cliente.

[6]Camada de segurança e autenticação simples, tradução livre

4. Experimentação com protocolos de mensagens

Em cada execução foram iniciadas 100 instâncias do código-fonte das quais foram tiradas medições de tempo. A cada execução a instância conecta, autentica(variável), publica e desconecta do *broker*. As medições de tempo foram feitas de forma a medir-se: tempo de conexão e tempo de publicação.



(a) Não autenticado

(b) Autenticado

(c) Publicação

Figura 1. Comparação de tempo entre protocolos de mensagens

Como podemos notar na Figura 1-a, tanto o protocolo STOMP quando o MQTT têm um tempo de conexão muito inferior ao AMQP, o qual tem um tempo de execução vinte vezes maior que que os demais protocolos. Entretanto, isto é esperado, pois esse é um protocolo mais seguro que os outros dois, tendo sido proposto para troca de mensagens não somente em dispostivos IoT. Os demais protocolos têm um desempenho superior, conectando-se em milisegundos, mesmo o protocolo baseado em texto como o STOMP. Observa-se ainda que o melhor desempenho vem de um protocolo com cabeçalho pequeno (de tamanho mínimo de 2 *bytes*), o MQTT. Sendo este um protocolo concebido para a telemetria, é de se esperar que o mesmo tenha um grande desempenho em relação aos demais. Outro importante fato é que este é um protocolo binário, diferente do protocolo textual STOMP que teve o segundo melhor desempenho.

No caso onde os clientes foram autenticados, Figura 1-b, o pior desempenho fica com o MQTT. Tal fato pode estar relacionado a ser ferramenta com mais funcionalidades que a outra. Entretanto, sendo o Mosquitto a ferramenta mais utilizada como *broker* MQTT atualmente, tal comparação se faz válida. Poucos milissegundos de diferença

fazem o AMQP do segundo melhor em desempenho. Ficando o melhor desempenho com o STOMP neste item.

Para publicação de dados o que executa mais rapidamente o processo é o AMQP, seguido pelo STOMP, Figura 1-c. Mesmo com a diferença de poucos milissegundos o protocolo STOMP efetivamente teve um melhor desempenho neste item. Sendo o menor desempenho do protocolo MQTT, que não possue envio de dados no modo *full duplex*, a não ser mediante o uso de WebSockets. O desempenho melhor do AMQP nesta função pode ser decorrente da sua capacidade de transmissão *full duplex*, não presente em nenhum dos outros protocolos estudados.

5. Discussão e resultados

O STOMP teve bom resultado nas situações de conexão com autenticação e segundo melhor resultado na publicação de dados. Sendo este um protocolo textual, se torna mais indicado a sua utilização em tais situações. Sendo o protocolo que obteve melhor desempenho temporal geral, este protocolo seria uma escolha apropriada para implementação de segurança nos casos de uso citados. Através de sua autenticação é possível evitar o caso de alteração de mensagem (caso de uso 1) e o *replay* de mensagens (caso de uso 2), em adição, seu desempenho adequado na conexão e publicação de mensagens o torna a opção ideal para resolução da questão da latência para tomada de decisão no *dispenser* de alimento (caso de uso 3).

É importante notar-se que o protcolo STOMP, mesmo que com menor visibilidade dentre os demais utilizados neste estudo, apresentou bons resultados e trás características relevantes para a decisão do protocolo a ser utilizando em um projeto. Este trabalho apresentou dados sobre o protocolo, que suportam a escolha do STOMP neste tipo de projeto. *A priori* não imaginava-se que esse protocolo atingiria um resultado interessante, devido a sua pouca expressividade na área acadêmica.

6. Conclusão e trabalhos futuros

Através deste trabalho foi possível expor o cenário e casos de uso, uma visão geral dos protocolos de mensagem e o experimento com seus resultados. Tais resultados levaram a escolha do STOMP como solução, a qual é melhor comentada na seção de discussão. Desta forma pode-se notar que os objetivos deste trabalho foram atingidos de maneira plena.

Como futuras contribuições anotamos a necessidade de ampliação do escopo de testes e ampliação do número de protocolos experimentados. Sendo que a amplicação do número de protocolos testados permitiria trazer a comparação para além do escopo somente da arquitetura *pub/sub*.

Referências

Aazam, M. and Huh, E.-N. (2014). Fog computing and smart gateway based communication for cloud of things. In *2014 International Conference on Future Internet of Things and Cloud*, pages 464–470. IEEE.

Aloufi, K. and Alhazmi, O. (2020). Secure iot resources with access control over restful web services. *Jordan Journal of Electrical Engineering. All rights reserved-Volume*, 6(1):64.

Bertin, E., Hussein, D., Sengul, C., and Frey, V. (2019). Access control in the internet of things: a survey of existing approaches and open research questions. *Annals of Telecommunications*, pages 1–14.

Deep, S., Zheng, X., and Hamey, L. (2019). A survey of security and privacy issues in the internet of things from the layered context. *arXiv preprint arXiv:1903.00846*.

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.

Luzuriaga, J. E., Perez, M., Boronat, P., Cano, J. C., Calafate, C., and Manzoni, P. (2015). A comparative evaluation of amqp and mqtt protocols over unstable and mobile networks. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 931–936. IEEE.

Naik, N. (2017). Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. In *2017 IEEE international systems engineering symposium (ISSE)*, pages 1–7. IEEE.

Paho-mqtt. Paho-mqtt - pypi. `https://pypi.org/project/paho-mqtt/`. Accessed: 2019-11-22.

Pika. Pika pure python rabbitmq/amqp 0-9-1 client library. `https://github.com/pika/pika`. Accessed: 2019-11-22.

Soni, D. and Makwana, A. (2017). A survey on mqtt: a protocol of internet of things (iot). In *International Conference On Telecommunication, Power Analysis And Computing Techniques (ICTPACT-2017)*.

Stomp. STOMP Protocol Specification version 1.2. `http://stomp.github.io/stomp-specification-1.2.html`. Accessed: 2019-10-04.

Stomp. Stomp the simple text oriented messaging protocol. `https://stomp.github.io/`. Accessed: 2019-10-03.

Stomp.py. Stomp.py - pypi. `https://pypi.org/project/stomp.py/`. Accessed: 2019-11-22.

Sueda, Y., Sato, M., and Hasuike, K. (2019). Evaluation of message protocols for iot. In *2019 IEEE International Conference on Big Data, Cloud Computing, Data Science & Engineering (BCD)*, pages 172–175. IEEE.

Szydło, T., Suder, P., and Bibro, J. (2013). Message-oriented communication for ipv6-enabled pervasive devices. *Computer Science*, 14.

Yi, S., Hao, Z., Qin, Z., and Li, Q. (2015). Fog computing: Platform and applications. In *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, pages 73–78. IEEE.

Zhou, C. and Zhang, X. (2014). Toward the internet of things application and management: A practical approach. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pages 1–6. IEEE.

Zorkany, M., Fahmy, K., and Yahya, A. (2019). Performance evaluation of iot messaging protocol implementation for e-health systems. *Performance Evaluation*, 10(11).