AVALIAÇÃO DA ESCALABILIDADE DO PROTOCOLO MQTT NO CONTEXTO DA INTERNET DAS COISAS

DIOGO MACIEL DA CUNHA [1], MARCO AURÉLIO SPOHN[2]

1 Introdução

O MQTT é um protocolo da camada de aplicação, muito utilizado em redes IoT por sua simplicidade e garantias. Por isso, é objeto de diversas pesquisas como as focadas na escalabilidade do protocolo (Spohn, 2022). Estas, têm como objetivo resolver o problema do servidor MQTT como gargalo e ponto único de falha (Spohn, 2022). Assim, buscam meios para atenuar ou resolver esse problema através dos processos de escalabilidade vertical (Genero, 2022) e horizontal (Ribas, 2022).

Desse modo, para avaliar o desempenho dessas soluções é necessário a utilização de ferramentas específicas que consigam simular um tráfego de dados em uma rede que utilize o protocolo. Entretanto, há um número limitado de ferramentas para testes de redes MQTT de código aberto disponíveis na internet (Spohn, 2022), com a maioria descontinuada. Outro problema, é a dificuldade na geração de experimentos com elas, por não possuírem formas de disparo e coleta automatizados. Assim, uma grande carga de trabalho é imposta aos interessados pelo tema por precisarem adaptar uma ferramenta para os seus testes e realizar os disparos e coletas de resultados de forma manual.

2 Objetivos

Criar uma interface para o *MQTTLoader*, a ferramenta de código aberto mais completa para testes de escalabilidade, que facilite o disparo e a coleta de experimentos com clientes e servidores MQTT hospedados em qualquer topologia de rede definida pelo usuário.

3 Metodologia

Foi desenvolvido uma interface que permite a automatização do processo de disparo de

[1] Acadêmico de Ciência da Computação, Universidade Federal da Fronteira Sul, *campus Chapecó*, contato: diogomaciel.cunha@gmail.com
[2] Doutor, Universidade Federal da Fronteira Sul, **Orientador(a)**.

experimentos e coleta dos resultados. Ela utiliza uma estrutura de orquestrador e trabalhador, com a comunicação intermediada pelo MQTT, e a interface com o usuário é dada por uma *API REST* que retorna JSON como resultado. Foi desenvolvida em Go utilizando a versão 3.1.1 do MQTT para a comunicação entre o orquestrador e os trabalhadores.

Para realizar os experimentos, o orquestrador recebe via HTTP um JSON com a definição do experimento requisitado, que é composto por: trabalhadores responsáveis, quantidade de tentativas (em caso de falha) e os parâmetros do *MQTTLoader*. Após isso, ele requisita os experimentos para os trabalhadores escolhidos e espera o retorno dos dados, também em formato JSON. Ao receber o retorno de todos os envolvidos, ele retorna ao usuário uma lista com os resultados dos experimentos realizados separados por trabalhador. Os trabalhadores executam localmente um processo da ferramenta de teste e retornam ao orquestrador todos os resultados gerados por ela: métricas, arquivos adicionais de log e erros.

Para a comunicação entre orquestrador e trabalhadores, foi utilizado o protocolo MQTT com um sistema de tópicos hierárquicos separados com identificadores únicos dos clientes. Desta forma, existem tópicos de controle de estado dos clientes, de requisição/resposta e identificação. Para o controle de estado, foram utilizados tópicos de *ping* para avaliar o estado atual dos trabalhadores do ponto de vista do orquestrador. Já para o disparo e coleta de experimentos, escolheu-se adicionar um tópico para disparo e outro para coleta por trabalhador. Para identificar os trabalhadores, foi utilizado um identificador único, ID, fornecido pelo orquestrador para cada trabalhador. Estes, requisitam um ID para o orquestrador que envia o valor para o trabalhador, que passa a utilizá-lo como identificador do cliente, para o MQTT, e parte dos seus tópicos de ping e requisição/resposta.

Para a validação, foi utilizado uma rede de containers *Docker* para testes em máquina local, duas máquinas com *Intel Core* i7-3770 8GB para simular uma comunicação em rede local e brokers públicos para simular um ambiente real. Para cada cenário, foram instanciados três trabalhadores, cada um com 10, 100 e 1000 clientes trocando 10 mensagens de tamanho 10 bytes entre si. Como métricas, foi observado como o orquestrador se comportou em cada um dos cenários no que diz respeito a quantidade de clientes por experimento, controle de trabalhadores e coleta de resultados.

No primeiro cenário, foram instanciados cinco contêineres: um servidor MQTT (*Mosquitto*), um orquestrador e três trabalhadores. O servidor foi utilizado para garantir a

comunicação entre os processos da aplicação e como alvo dos experimentos. No segundo cenário, o orquestrador e dois trabalhadores ficaram na máquina que possuía o Mosquitto e o último trabalhador na segunda máquina. Aqui, novamente se utilizou o mesmo servidor para teste e comunicação. E no último ambiente, escolheu-se manter a distribuição anterior de processos e alterar apenas o servidor alvo dos testes. Deste modo, foram escolhidos cinco servidores MQTT públicos disponibilizados na internet: *Mosquitto*, *Eclipse*, *HiveMQ*, *Flux* e *Emqx*.

4 Resultados e Discussão

No primeiro cenário, verificou-se que a latência permaneceu baixa em todos os casos de teste, mas houve um aumento na carga de trabalho maior da ferramenta conforme o número de clientes crescia. Entretanto, não houve problemas nas rotinas especificadas da interface e a ferramenta executou de forma transparente. No segundo ambiente, observou-se uma situação semelhante acompanhada por um aumento da latência, mas o aumento da carga foi menor no trabalhador que estava isolado na segunda máquina. No terceiro, dois dos servidores públicos estavam indisponíveis no momento do teste e os testes foram realizados apenas em três servidores: *Mosquitto*, *HiveMQ* e *Emqx*. Neles, obteve-se a maior latência entre os três casos e as ressalvas sobre os clientes MQTT continuaram.

Ao fim dos testes, percebeu-se um problema na definição do experimento, utilizar um único servidor para comunicação e realização dos testes pode gerar um congestionamento indesejado na rede que dificulta o funcionamento da interface. Também verificou-se que utilizar trabalhadores separados em várias máquinas conectadas por uma topologia de rede, permite a realização de experimentos com mais clientes. Com isso, foi percebido a estabilidade da ferramenta ao realizar sua tarefa principal.

5 Conclusão

Com o crescimento das pesquisas sobre a escalabilidade do protocolo MQTT, houve a necessidade da criação de ferramentas que gerem cenários de testes para a validação das soluções. Entretanto, há um número reduzido de ferramentas ainda recebendo atualizações. Entre elas, o *MQTTLoader* se destaca pelas suas funcionalidades e flexibilidade, mas carece de formas de automatizar o processo de disparo e coleta de resultados.

Por isso, foi implementada uma interface em Go que possibilita a automatização dos disparos e coletas de experimentos. Ela utiliza uma arquitetura de orquestrador e trabalhadores para realizar o disparo de experimentos e coletas de resultados, com o MQTT para a comunicação entre o orquestrador e os trabalhadores.

A interface se mostrou estável em todos os casos de testes e retornou todos os resultados gerados pela ferramenta base, mas utilizar o mesmo servidor MQTT para testes e comunicação não se mostrou uma boa estratégia. Por outro lado, os dados gerados foram satisfatórios mesmo quando os trabalhadores não estavam juntos do orquestrador.

Referências Bibliográficas

RIBAS, Nicolas Kolling. Federação de brokers do protocolo MQTT implementação e análise de desempenho. Trabalho de Conclusão de Curso, Ciência da Computação, Universidade Federal da Fronteira Sul, 2022.

GENERO, Willian Bordignon. Avaliação dos protocolos MQTT e MQTT-SN no contexto da internet das coisas. Trabalho de Conclusão de Curso, Ciência da Computação, Universidade Federal da Fronteira Sul, 2022.

SPOHN, Marco Aurelio. On MQTT Scalability in the Internet of Things: Issues, Solutions, and Future Directions. **Journal of Electronics and Electrical Engineering**, p. 4-4, 2022.

Palavras-chave: Ferramentas; MQTT; MQTTLoader; Experimentos

Nº de Registro no sistema Prisma: PES 2021 - 0471

Financiamento: UFFS