# Escalabilidade do MQTT

**Bruno Menegaz Acerbi**

[1]Universidade Federal do Espírito Santo (UFES)

[2]Centro Tecnológico
Departamento de Informática

***Resumo.*** *Para esse trabalho, foi proposto a construção de uma série de testes para avaliarmos a capacidade de escalabilidade que o protocolo de comunicação MQTT pode fornecer aos usuários. Ao final, foram construídos gráficos informativos para a visualização dos gastos de memória e CPU durante os testes de estresse.*

## 1. Introdução

Com o intuito de medir a capacidade de escalabilidade que o protocolo MQTT pode oferecer aos seus usuários, construiu-se uma serie de testes a fim de criar um ambiente de estresse sob um broker Mosquitto [Light 2017] e medir o consumo de CPU e memória utilizada pelo processo ao longo do teste.

A princípio, o projeto iniciou-se com o intuito de medir a capacidade de *payload* oferecida pelo serviço, realizando um aumento gradativo do tamanho da mensagem publicada na rede até que o limite máximo fosse atingido.

Em sequência, visando monitorar os gastos computacionais do processo gerado pelo broker, iniciou-se uma simulação na qual há um aumento gradativo de clientes se conectando ao broker e publicando mensagens. Por fim, realizou-se outro comparativo considerando também a inscrição gradativa de usuários no tópico que ocorriam as publicações.

## 2. Metodologia utilizada

Os testes foram realizados em um ambiente com um processador Intel i7-11390H 3.40GHz com 16 GB de memória RAM disponíveis.

### 2.1. Teste do Payload

A princípio foi construido um código capaz de se conectar a um broker local e entrar em um looping infinito que, a cada iteração, o tamanho da mensagem a ser publicada será incrementado em 100 caracteres e contabilizado para a visualização do usuário.

### 2.2. Teste de Publisher

Para capturar os gastos de CPU e memória durante a execução da simulação, foi desenvolvido um script para monitorar a utilização desses recursos pelo processo do broker, registrando os valores obtidos em um arquivo de log local a cada meio segundo.

Após implementar o serviço de monitoramento, foi necessário desenvolver uma aplicação para realizar a escrita de mensagens. Para isso, foi criado um script que simula

o cadastro crescente de novos usuários utilizando multi-threading. O ambiente foi configurado para instanciar uma nova thread com um novo usuário para o broker a cada 0,1 segundos de execução, com cada usuário publicando mensagens em um loop a cada 0,1 segundos.

### 2.3. Teste de Publisher e Subscriber

Para realizar o teste de publisher e subscriber, utilizou-se o mesmo script de monitoramento descrito anteriormente.

Para gerar um processo de estresse semelhante, foram feitas atualizações na aplicação, acrescentando a criação de um usuário inscrito no tópico a ser publicado e ajustando a frequência de criação de threads. Desta vez, a cada 0,5 segundos, são iniciadas duas novas threads: uma para instanciar um cliente que publica no tópico de mensagens a cada 0,5 segundos e outra para instanciar um usuário que se inscreve no tópico da mensagens.

## 3. Resultados obtidos

Para o primeiro experimento, o script desenvolvido performou como esperado, incrementando o payload de cada mensagens por iteração, porém quando valores maiores eram atingidos, o broker acabava escalando o consumo de memória da máquina de tal forma que ocorria um travamento do dispositivo antes que o limite fosse atingido. Dessa forma, para atendermos a pergunta proposta nesse tópico, foi realizado uma consulta a documentação do Mosquitto, na qual é informado que o limite de payload é de 268.435.455 bytes, sendo assim uma mensagem pode conter até 268.435.455 caracteres.

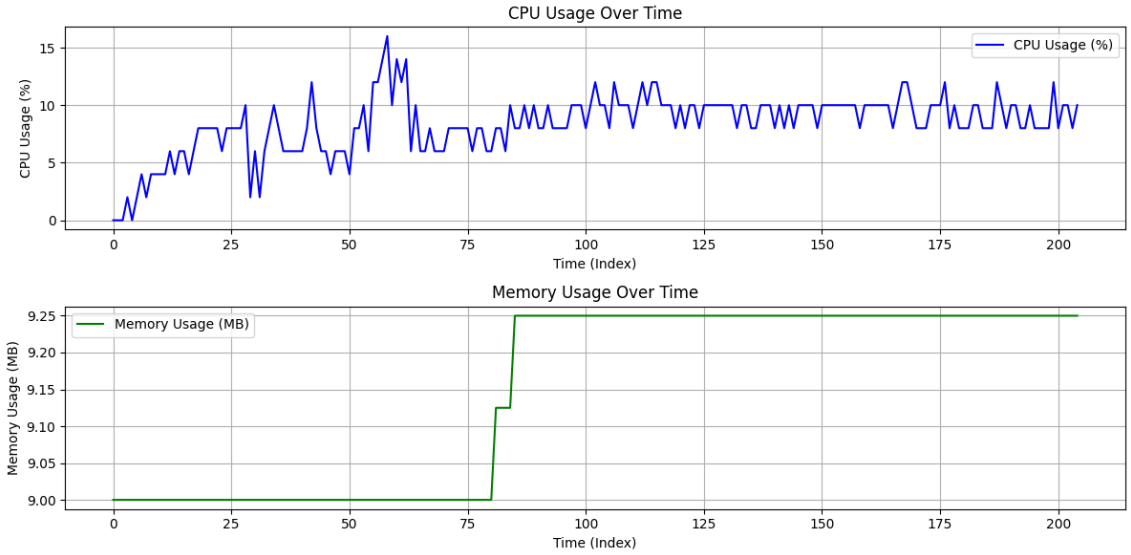


**Figura 1. Teste com publicadores**

Dando prosseguimento, o teste de consumo de CPU e memória com o incremento de publicadores em um broker apresentou os resultados exibidos na imagem 1. Notou-se que os valores não apresentaram uma grande variação de acordo com o incremento de clientes publicadores, apresentando uma estabilização com cerca de 500 clientes publicando uma mensagem na rede cada 0,1 segundo, cenário esse que gerou um gasto médio de 10%

de CPU e um aumento de 0,25 MB de memória ao compararmos com o consumo base do broker.

Por fim, o experimento final apresentou resultados mais interessantes. Pode-se visualizar na imagem 2 que o consumo de CPU e memória escalou muito mais rápido, mesmo com uma frequência de utilização bem inferior comparada ao teste anterior.

Nota-se que houve uma estabilização na marca de 200 leituras de tempo. Considerando que a cada leitura, um publicador e um leitor eram instanciados na rede, nessa marca existiam cerca de 400 clientes na rede, 200 leitores e 200 publicadores que escreviam no tópico assinalado a cada 0,5 segundos. Nesse cenário, pode-se registrar um consumo de mais de 50% de CPU e um incremento de aproximadamente 16 MB de RAM ao valor base consumido pelo broker sem demanda.

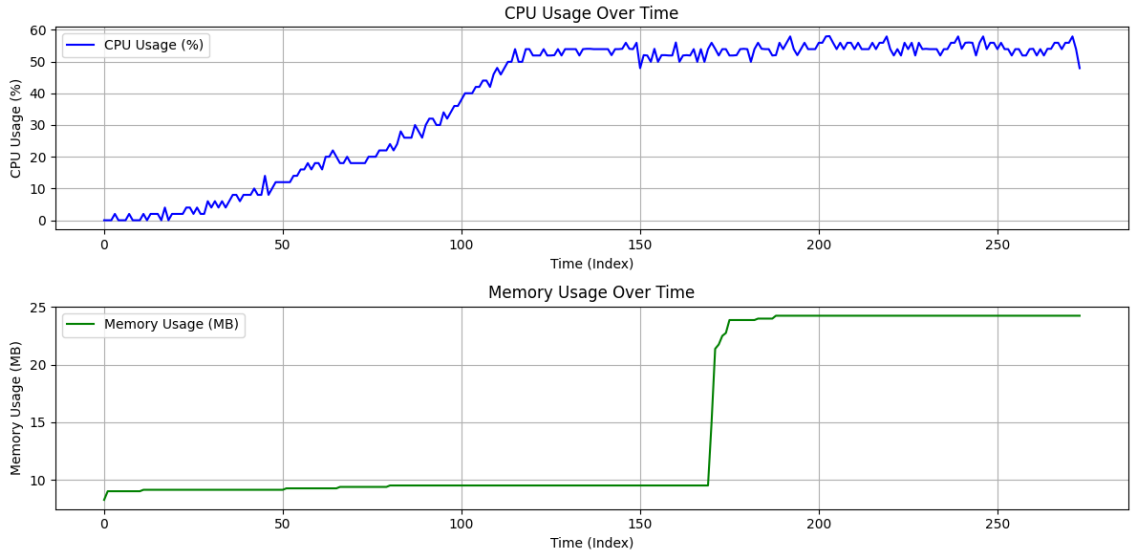


**Figura 2. Teste com assinadores e publicadores**

## Referências

Light, R. A. (2017). Mosquitto: server and client implementation of the mqtt protocol. *The Journal of Open Source Software*, 2.