

MAC0352 - Redes de Computadores e Sistemas Distribuídos

EP 1 - Relatório de Implementação e Performance

Arthur Pilone M. da Silva, N°USP 11795450

Implementação

Versão do Protocolo MQTT Utilizada

Trata-se de um “redux” do protocolo MQTT v3.1.1. Várias das funções básicas descritas no protocolo foram implementadas, incluindo:

- Possibilidade de um cliente se inscrever (ou desinscrever) de diferentes tópicos;
- Possibilidade de um cliente publicar a um ou mais tópicos;
- Suporte ao acesso de múltiplos clientes simultaneamente;
- Controle de “tempo de permanência” de uma dada conexão através da passagem de um *keep alive time*;
- Identificação de clientes recorrentes utilizando o *client_id*;
- Gravação das escolhas de inscrições de um cliente entre diferentes sessões;
- Cliente pode escolher por limpar seus dados / utilizar “sessão limpa” (*clean session*);
- Troca de pacotes de pings (*PINGREQ* e *PINGRESP*) entre broker e clientes.

Versão do Protocolo MQTT Utilizada

Apesar da grande maioria dos comandos ser implementada, algumas funcionalidades mais complexas foram deixadas de lado e algumas restrições são impostas:

- Identificação com *usernames* e autenticação com senhas não são permitidas;
- Troca de mensagens com *Quality of Service* (QoS) maior que 0 não é aceita;
- Não há suporte para a identificação e criação de tópicos usando expressões e caracteres coringa (*wildcard characters*);
- Mensagens “*retidas*” (*retained*) a tópicos específicos não são aceitas;
- “Mensagens de testamento” (*Will messages*) para clientes também não são aceitas;
- Os pacotes recebidos pelo *broker* têm tamanho máximo de 4096 bytes.
- Durante uma execução do *broker* (antes de se utilizar o comando *make clean*), é possível a criação de até 65534 (*unsigned short max - 1*) tópicos e clientes diferentes;

Estrutura do Broker

- Internamente, cada tópico e cada *client_id* é associado a um *unsigned short* não nulo, que chamamos de *item_index*.
- Vale notar que, em uma implementação real, seria mais seguro utilizar uma função de *hashing* que associe cada *client_id* e *topic_name* a um *hash*, mas essa escolha foi omitida nesta implementação.
- Também é relevante que, como o *broker* foi projetado para funcionar com tráfego leve, alguns problemas e inconsistências surgem caso mais de 20 clientes tentam se conectar em um intervalo de 10^{-5} segundos, por surgir um problema de **concorrência** de acesso aos arquivos vitais para o *broker*.

Estrutura do Broker - Arquivos Criados durante Execução

*/tmp/mac352ep1AP/ **topics.txt***

Guarda as relações *topico - item_index*

clients.txt

Guarda as relações *client_id - item_index*

topics/<index>

Guarda os *indexes* dos clientes inscritos no tópico de *index* *<index>* e **que estejam online**, bem como o QoS que aceitam.

clients/<index>

Guarda os *indexes* dos tópicos em que o cliente de *index* *<index>* é inscrito, bem como o QoS aceito por ele em cada tópico.

pipes/<index>

Endereço do *pipe / fifo* aberto para o cliente *<index>*.

Estrutura do Broker - Código

De maneira análoga ao contido no código exemplo dado no enunciado, o programa realiza uma operação *fork()* para criar um processo filho responsável por cada conexão com um cliente.

Após a conexão ser efetivada utilizando o comando *CONNECT*, é aberto um *pipe* para cada cliente conectado. Ao ser enviada uma mensagem *PUBLISH* em um dos tópicos nos quais um cliente é inscrito, o broker encaminha essa publicação para ele copiando o pacote *PUBLISH* no seu *pipe*.

Para concretizar a troca de mensagens entre clientes, cada processo responsável por um cliente invoca o *fork()* novamente e se duplica, dedicando um processo a ler o *pipe* do cliente e o repassar ao *socket* da conexão, e dedicando o outro processo a ler os pacotes enviados pelo cliente no *socket*.

Estrutura do Broker - Código

No laço principal do processo filho responsável por ler o *socket* da conexão com o cliente, sempre é testado qual o comando de controle MQTT da mensagem recebida até que seja recebido um pacote que não seja *PUBLISH*, *SUBSCRIBE*, ou *PINGREQ*, caso no qual o cliente é desconectado.

Caso o pacote enviado pelo cliente deixe de respeitar alguma das convenções do protocolo MQTT 3.1.1, como um conjunto de *bits* reservados, a conexão também é encerrada.

Toda vez que o servidor receber algum pacote válido do cliente, o seu *keep alive time* é restaurado.

Organização do Código

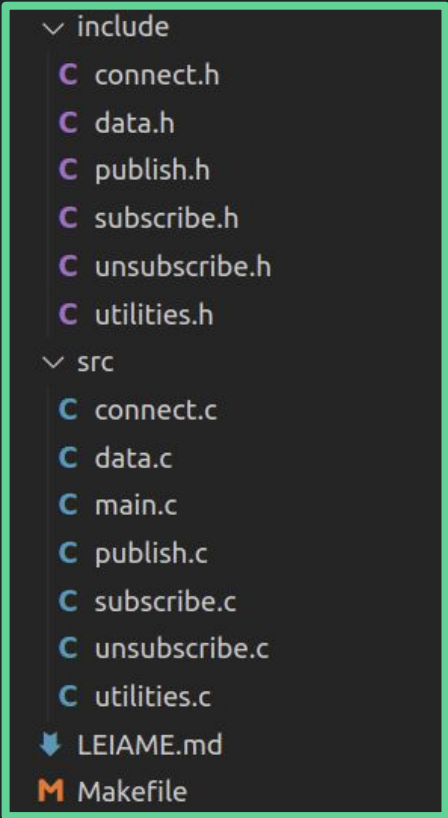
LEIAME.md : contém informações importantes sobre a execução e compilação do código.

main.c : contém código principal do broker.

data.c + .h : contém métodos usados para a manipulação dos dados do broker nos seus arquivos.

utilities.c + .h : contém funções auxiliares diversas.

connect, publish, subscribe e unsubscribe .c + .h : contém funções para processar os comandos **CONNECT**, **PUBLISH**, **SUBSCRIBE** e **UNSUBSCRIBE**.



```

  include
  C connect.h
  C data.h
  C publish.h
  C subscribe.h
  C unsubscribe.h
  C utilities.h
  src
  C connect.c
  C data.c
  C main.c
  C publish.c
  C subscribe.c
  C unsubscribe.c
  C utilities.c
  LEIAME.md
  Makefile

```

A screenshot of a file explorer window with a dark background and a green border. It shows a project structure with two main folders: 'include' and 'src'. The 'include' folder contains six header files: connect.h, data.h, publish.h, subscribe.h, unsubscribe.h, and utilities.h. The 'src' folder contains six corresponding source files: connect.c, data.c, main.c, publish.c, subscribe.c, and unsubscribe.c. Below these folders, there is a file named 'LEIAME.md' with a blue arrow icon, and a file named 'Makefile' with an orange 'M' icon.

Comandos MQTT Implementados

CONNECT + CONNACK

O comando *CONNECT* é utilizado para solicitar a conexão de um cliente, e o pacote *CONNACK* é a resposta do *broker* ao cliente, informando sobre a efetivação (ou não) da dada conexão.

Se pelo cabeçalho do pacote *CONNECT* for possível identificar o desrespeito a uma das limitações impostas, o pacote *CONNACK* de resposta informará ao cliente o porquê da conexão estar sendo rejeitada.

Internamente, cria ou recupera o *item_index* do cliente, cria o seu *pipe* e re-insere o cliente nas inscrições de cada um dos tópicos em que era inscrito, caso exista informação sobre uma sessão anterior do mesmo cliente.

O primeiro pacote enviado por um cliente **deve ser** um comando *CONNECT*

Comandos MQTT Implementados

PINGREQ + PINGRESP

O comando *PINGREQ* é enviado do cliente ao servidor quando ele quiser se assegurar de que a conexão ainda é saudável ou quando ele quiser “relembrar” o servidor de que ele ainda está conectado, a fim de não ser desconectado em decorrência da tolerância do *keep alive time*.

O servidor sempre responde com um simples PINGRESP.

Esses dois tipos de comandos são extremamente simples e, portanto, são tratados diretamente no código principal (***main.c***) .

Comandos MQTT Implementados

SUBSCRIBE + SUBACK

O comando *SUBSCRIBE* enviado por um cliente ao *broker* sinaliza o interesse desse cliente em se inscrever a 1 ou mais tópicos (passados na carga útil do pacote), bem como o valor de QoS máximo que ele aceita para receber mensagens vindas de cada um dos tópicos passados.

Caso o cliente já esteja inscrito em algum dos tópicos passados, ele permanece inscrito com o QoS mínimo entre o antigo e o recentemente passado.

O *broker* responde o pedido do cliente com um pacote do comando *SUBACK*, que informa, para cada tópico no qual o cliente pediu para ser inscrito, qual o QoS máximo garantido a ele.

Comandos MQTT Implementados

UNSUBSCRIBE + UNSUBACK

O comando *UNSUBSCRIBE* é utilizado por um cliente que deseja remover sua inscrição de um ou mais tópicos. Após remover as inscrições solicitadas pelo cliente, o *broker* responde com um pacote do comando *UNSUBACK*.

Internamente, são removidas as entradas do tipo (*item_index*, QoS) nos devidos arquivos nos diretórios *.../topics/* e *.../clients*.

Comandos MQTT Implementados

PUBLISH

Como o nosso *broker* implementa apenas a troca de mensagens com $QoS = 0$, as mensagens publicadas são sempre transmitidas em pacotes do comando *PUBLISH*.

Para publicar uma mensagem a um tópico, um cliente envia a mensagem em um pacote *PUBLISH* ao *broker* que, para cada cliente ativo inscrito no tópico, copia o pacote de *PUBLISH* nos seus respectivos *pipes*, que são lidos pelos processos filhos correspondentes e, então, encaminhados para os sockets dos clientes inscritos.

Comandos MQTT Implementados

Demais comandos

Caso o *broker* receba um pacote de comando *DISCONNECT* , ou de um comando reservado / que não esperava receber do cliente (como *PUBACK*, *SUBACK*, *PINGRESP* ...), o broker desconecta o cliente, respeitando a eventual escolha por ele feita ao conectar-se de que seus dados sejam apagados (*clean session = 1*).

Performance

Metodologia Utilizada nos Testes

Para os testes de rede e de uso de *cpu* , foi utilizado um *script* para a criação dos clientes. Primeiro, 70 % dos clientes são criados como sendo *mosquitto_sub*'s, sendo cada um deles **aleatoriamente inscrito em 3** dentre um total de 10 tópicos. Em seguida, os clientes restantes são criados como sendo *mosquitto_pub*'s, e cada um deles publica, **a um dos 10 tópicos (escolhido aleatoriamente) uma mensagem fixa** (“Lorem Ipsum Dolor sit Amet”). Por fim, todos os clientes *mosquitto_sub*'s são desconectados e terminados.

Entre cada par de clientes criados, há um intervalo de 5ms para que o *broker* possa processar os pedidos com mais resiliência.

Este script foi executado em um máquina virtual hospedada em meu computador, (que também executava o *broker*).

Metodologia Utilizada nos Testes

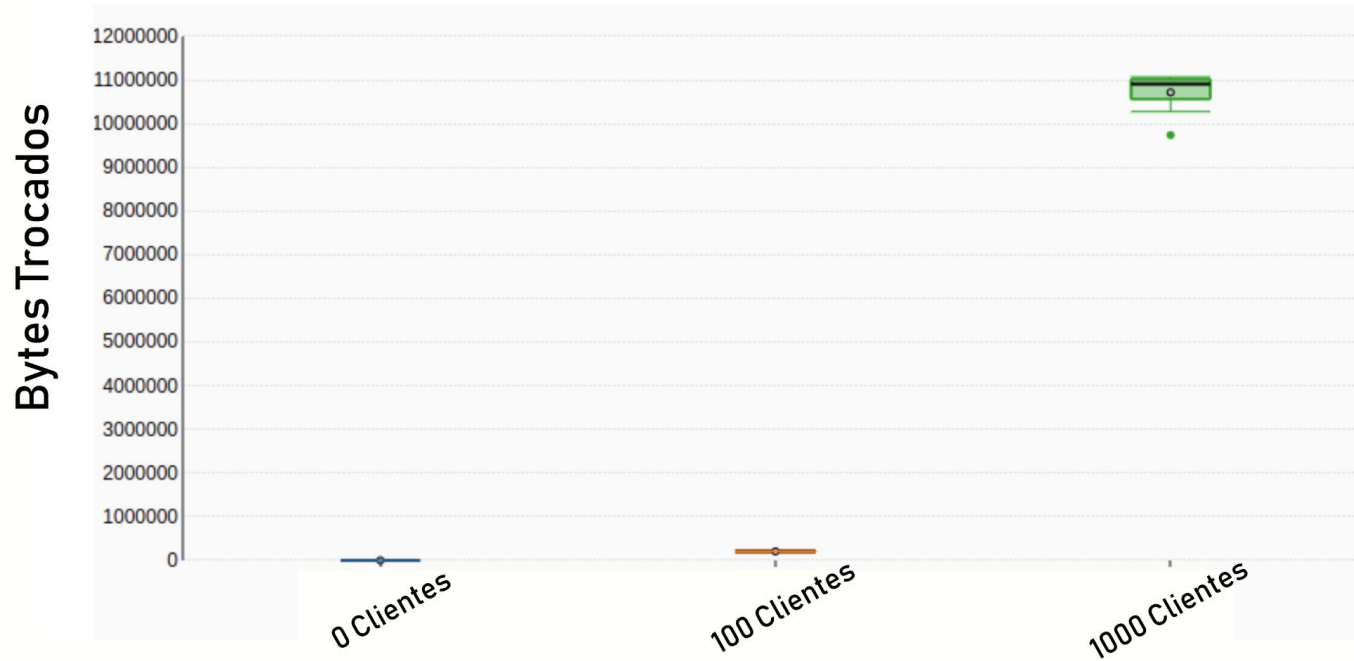
Teste de Uso de Rede

Analizou-se o número total de bytes trocados entre o *broker* e os clientes, utilizando o *Wireshark*, em 10 instâncias semelhantes (10 instâncias para cada número diferente de clientes.)

Os dados gerados foram organizados em um gráfico de *boxplot* com eixos em escala linear e em uma tabela.

Resultado - Teste de Rede

Bytes Trocados por Teste



Resultados - Teste de Rede

Bytes Trocados por Teste (*bytes*)

Teste	Média	Mediana	Desvio Padrão
0 Clientes	0	0	0
100 Clientes	205413.2	206605	5908.6
1000 Clientes	9645354.5	10862608	3415640.5

Metodologia Utilizada nos Testes

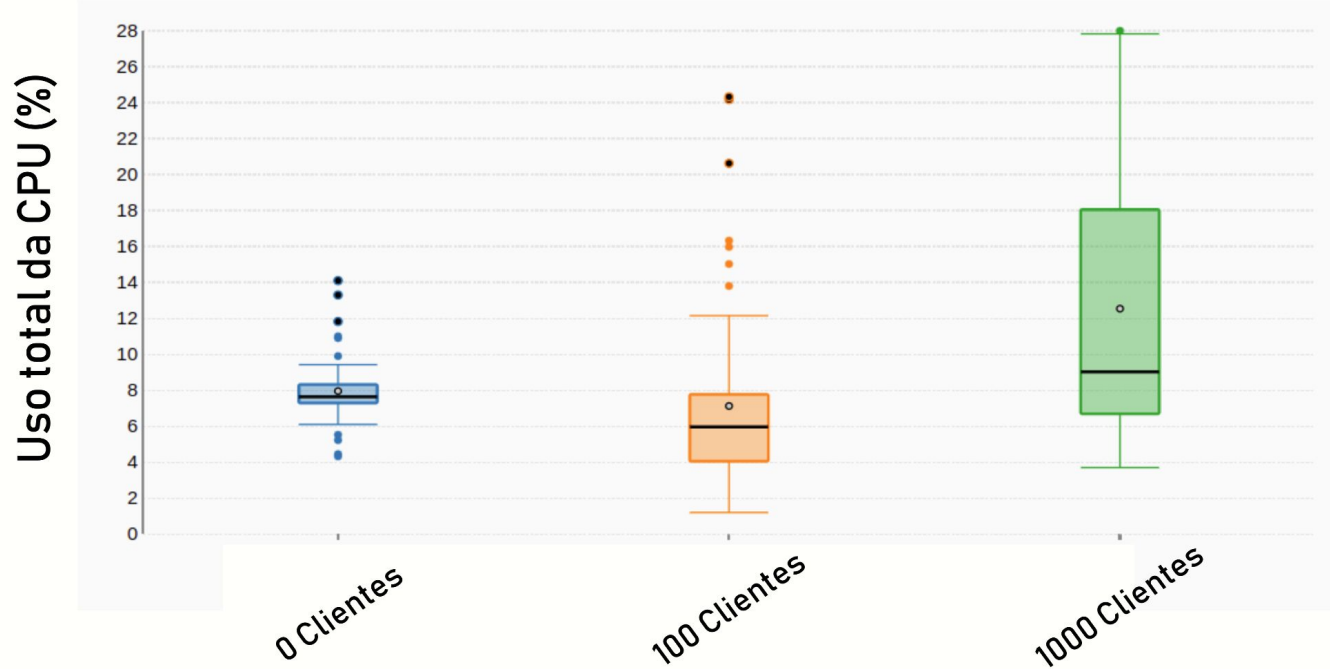
Teste de Uso da CPU

Após todos os programas (exceto o *broker*, a máquina virtual e um terminal) serem fechados, invocou-se o comando *top* com amostragem a cada 200ms e executou-se o *script* na máquina virtual. Em seguida, foi medido o **uso total da cpu quando retirado o consumo da máquina virtual**, para quantificar *aproximadamente* o custo computacional do *broker*.

Os experimentos foram repetidos diferentes vezes para cada número de clientes (0, 100 e 1000) e os resultados foram organizados na forma de um gráfico *boxplot* e uma tabela.

Resultado - Teste de CPU

Uso da CPU por teste



Resultados - Teste de CPU

Uso aproximado da CPU (%)

Teste	Média	Mediana	Desvio Padrão
0 Clientes	7.9	7.6	1.74
100 Clientes	7.1	6.0	5.0
1000 Clientes	12.5	9.0	7.2