Broker MQTT

MAC0352 EP1

Théo Borém Fabris

USP

26 de Abril de 2022

Introdução

- O broker MQTT implementa os principais comandos do MQTT v3.1.1 e suporta apenas comandos com qualidade de serviço (QoS) igual a zero;
- Esta apresentação seguirá o seguinte cronograma:
  1. Implementação dos comandos do MQTT;
  2. Estruturas de dados utilizadas;
  3. Fluxograma de controle do Broker;
  4. Análise do consumo de CPU e rede.

Comandos MQTT

Os comandos MQTT implementados e os detalhes importantes foram:

- CONNECT: comando enviado do cliente ao servidor:
  - pela especificação do MQTT, este deve ser o primeiro comando enviado para o servidor;
  - um segundo envio do comando CONNECT causará finalização da conexão, conforme especificado na especificação do MQTT v3.1.1;
- CONNACK: comando enviado do servidor ao cliente como resposta a uma conexão bem sucedida realizada pelo comando CONNECT:
  - por ser uma mensagem fixa, é implementado como o write de um array constante;
- PINGREQ: comando enviado do cliente ao servidor como forma de verificar se as extremidades da conexão continuam ativas;
- PINGRESP: comando enviado do servidor ao cliente como resposta ao PINGREQ:
  - por ser uma mensagem fixa, é implementado como o write de um array constante;

Comandos MQTT

- SUBSCRIBE: comando enviado do cliente ao servidor requisitando sua inscrição em um ou mais tópicos:
  - utiliza-se uma fila (queue.h) para implementar conseguir armazenar pedidos com vários tópicos;
- SUBACK: comando enviado do servidor ao cliente como resposta a uma inscrição bem sucedida do cliente em um tópico;
  - os elementos da fila são utilizados para gerar uma confirmação na ordem que cada inscrição foi requisitada;
- PUBLISH: comando enviado do cliente ao servidor ou do servidor ao cliente como forma de publicar uma mensagem recebida:
  - ao receber uma publicação em um determinado tópico de um cliente, o servidor passará a mensagem aos clientes inscritos;
- DISCONNECT: comando enviado do cliente ao servidor requisitando a desconexão do cliente, sendo suficiente fechar a conexão com o cliente.

Estrutura de Dados

Baseado nas necessidades requeridas pelos comandos MQTT implementados, foram necessárias as seguintes estruturas de dados:

- Fila é implementada no arquivo queue.c sendo utilizada apenas no processo de enviar um reconhecimento de inscrição (SUBACK);
- Árvore de busca binária é implementada no arquivo search.c. Ela foi utilizada nas seguintes situações:
  - Para manter o registro dos clientes conectados, utilizando como chave de busca o identificador único da thread do atribuída ao cliente;
  - Para manter o registro, privado de cada thread dos clientes, dos tópicos nos quais o cliente foi inscrito.

Fluxograma do Broker

A organização geral do broker é realizado pelas funções abaixo:

- a função brokerController gera as threads que interagem com os clientes após o accept da conexão pelo broker. Além disso, finaliza o broker se um EOF for lido da entrada padrão;
- a função clientmain (código das threads dos clientes) gerencia todo o protocolo da camada de aplicação, chamando as funções apropriadas, implementadas em mqtt.c, para cada comando MQTT, alocando recursos para manter uma tabela com o registro de tópicos nos quais o cliente está inscrito. Junto com a leitura do socket do cliente, também realiza a leitura do pipe criado para o cliente receber as mensagens publicadas, sendo tarefa do cliente filtrar aquelas que ele não se inscreveu.
- a biblioteca pthread fornece funções para permitir a resolução de problemas associados ao acessos simultâneos de clientes;

Configurações do Broker

Vale notar as seguintes propriedades sujeitas à configuração baseada na demanda esperada de clientes:

- o número de arquivos abertos pelo broker é aproximadamente igual a soma dos número de clientes publicadores com o produto do número de clientes inscritos por três, devido ao uso de um pipe para cada cliente inscrito em um tópico. Assim, dependendo no número de clientes esperados pode ser necessário aumentar o limite do sistema operacional para o arquivos abertos por processo, obtido por meio do comando ulimit -n;
- o tamanho da fila de conexões definido na chamada da função 'listen' tem valor igual a constante 'LQUEUELEN', 2000 por padrão.

Ambiente computacional e rede

Realizou-se as análise de uso de CPU e rede no seguinte ambiente computacional:

- Máquina onde executou-se os clientes tem as seguintes especificações:
  - Processador Qualcomm Technologies, Inc SDM845 8 CPUs
  - Sistema Operacional Android 10, Linux 4.9.186
  - mosquitto versão 2.0.11-6 (pacote termux)
- Máquina onde executou-se o broker tem as seguintes especificações:
  - Processador Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz (with SSE4.2)
  - Sistema Operacional Debian 10, Linux 4.19.0-20-amd64
  - Network controller: Intel Corporation Wireless 3165 (rev 81)
  - Conectado por Wi-Fi ao hotspot criado pelo smartphone acima
  - Dumpcap (Wireshark) 2.6.20 (Git v2.6.20 packaged as 2.6.20-0+deb10u3)

Metodologia

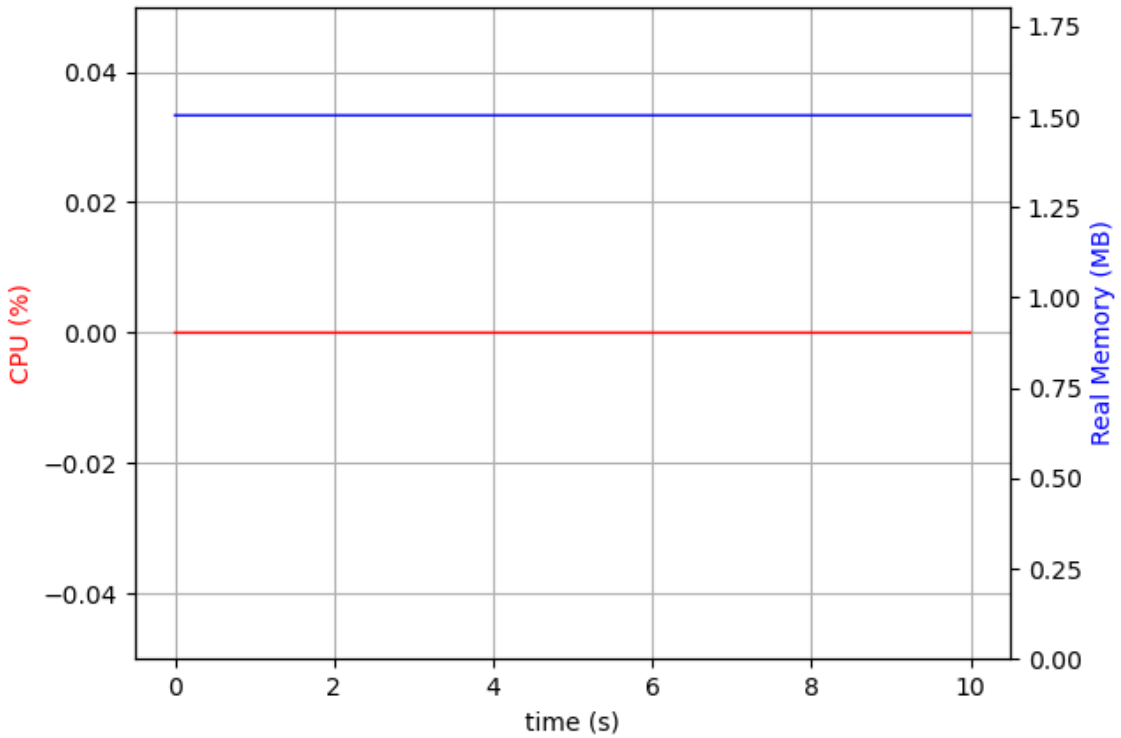
As métricas dos experimentos foram obtidas por meio dos seguintes métodos:

- o consumo de CPU do broker foi medido por meio do utilitário psrecord;
- o consumo de Rede do broker foi medido por meio do utilitário wireshark filtrando os pacotes TCP pela suas portas e obtendo os dados estatísticos fornecidos.

Em ambas as métricas foram realizados dez experimentos, com duração de tempo específica para cada experimento, e calculou-se o consumo médio e máximo de CPU neste período e o número médio de bytes transmitidos por segundo para cada experimento. Por fim, calculou-se a média e o desvio padrão das médias.

Experimento: zero clientes conectados

Cada experimento apenas iniciou o broker e o deixou rodar por 10 segundos. O uso de CPU e rede ficaram em 0% durante todo o intervalo em todos os experimentos. A figura abaixo ilustra o comportamento do servidor no experimento 1.



Experimento: cem clientes conectados

Cada experimento conectou 50 subscribers (em 10 tópicos e o timeout de 2 segundos), em seguida conectou 50 publisher publicando em um tópico, e finalizou com a desconexão dos 50 subscribers. O intervalo de duração do experimento é o início do broker até o tempo de desconexão do último subscriber.

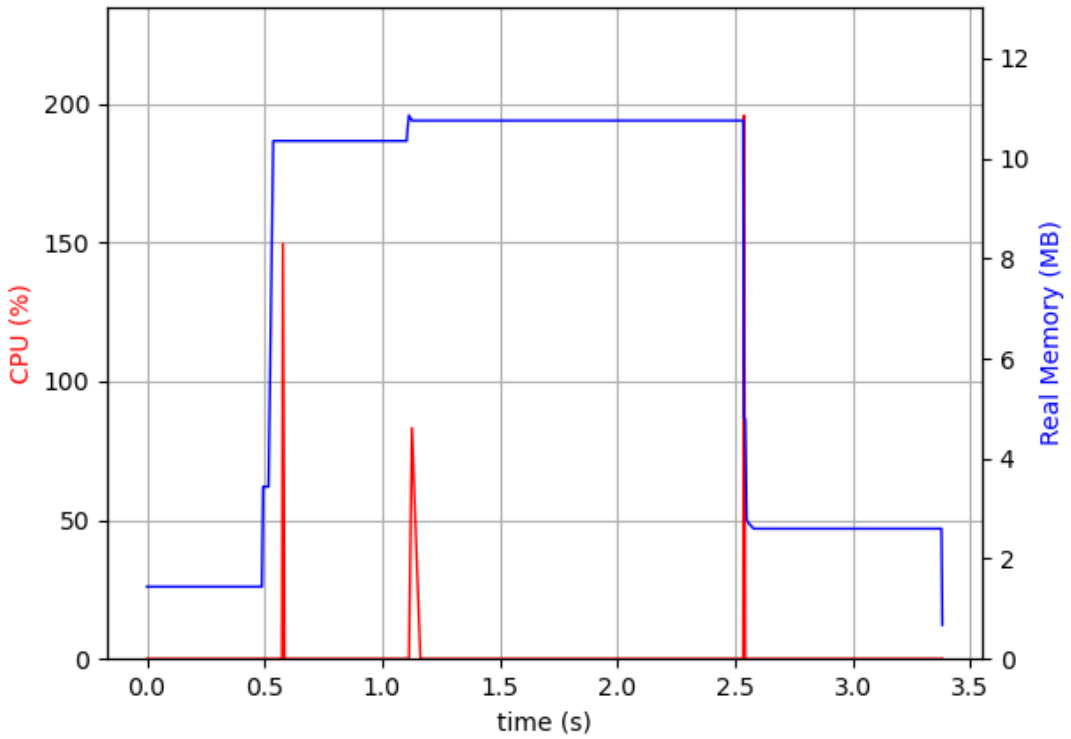
Exp.	Tempo (s)	máx. CPU (%)	média CPU (%)	média bytes/s
1	2.062	231.100006	4.952178	127k
2	2.062	398.200012	3.617847	123k
3	2.088	231.399994	1.239099	121k
4	2.079	98.500000	0.817427	122k
5	2.061	169.500000	3.303927	124k
6	2.062	196.399994	4.425210	125k
7	2.066	211.199997	4.162535	123k
8	2.097	195.800003	1.028866	122k
9	2.080	211.600006	3.970590	123k
10	2.063	448.700012	1.925473	120k

Experimento: cem clientes conectados

Logo:

- o uso médio de CPU em todo o período foi em média 2.9443 % com desvio padrão 1.5450;
- o consumo de bytes por segundo em todo o período foi em média 123 bytes por segundo com desvio padrão 2.

A figura abaixo ilustra o comportamento do servidor no experimento 8.



Experimento: mil clientes conectados

Cada experimento conectou 500 subscribers (em um tópico e o timeout de 10 segundos), em seguida conectou 500 publisher publicando em um tópico, e finalizou com a desconexão dos 500 subscribers. O intervalo de duração do experimento é o início do broker até o tempo de desconexão do último subscriber.

Exp.	Tempo (s)	máx. CPU (%)	média CPU (%)	média bytes/s
1	11.865	437.799988	31.400871	109k
2	10.795	532.299988	31.423399	120k
3	10.959	421.500000	33.553951	118k
4	10.879	625.400024	32.846790	119k
5	10.801	375.899994	30.291616	120k
6	10.772	462.500000	30.878117	120k
7	10.996	449.100006	32.590855	118k
8	10.861	530.700012	32.663525	119k
9	11.057	363.600006	30.812698	117k
10	10.997	606.400024	29.188782	118k

Experimento: mil clientes conectados

Logo:

- o uso médio de CPU em todo o período foi em média 31.5650 % com desvio padrão 1.34208;
- o consumo de bytes por segundo em todo o período foi em média 117.8 bytes por segundo com desvio padrão 3.258.

A figura abaixo ilustra o comportamento do servidor no experimento 10.