



MQTT v3.1.1 server for QoS = 0 (at most once delivery)

Source code at:

<https://github.com/giovannioliveira/MQTT-broker>

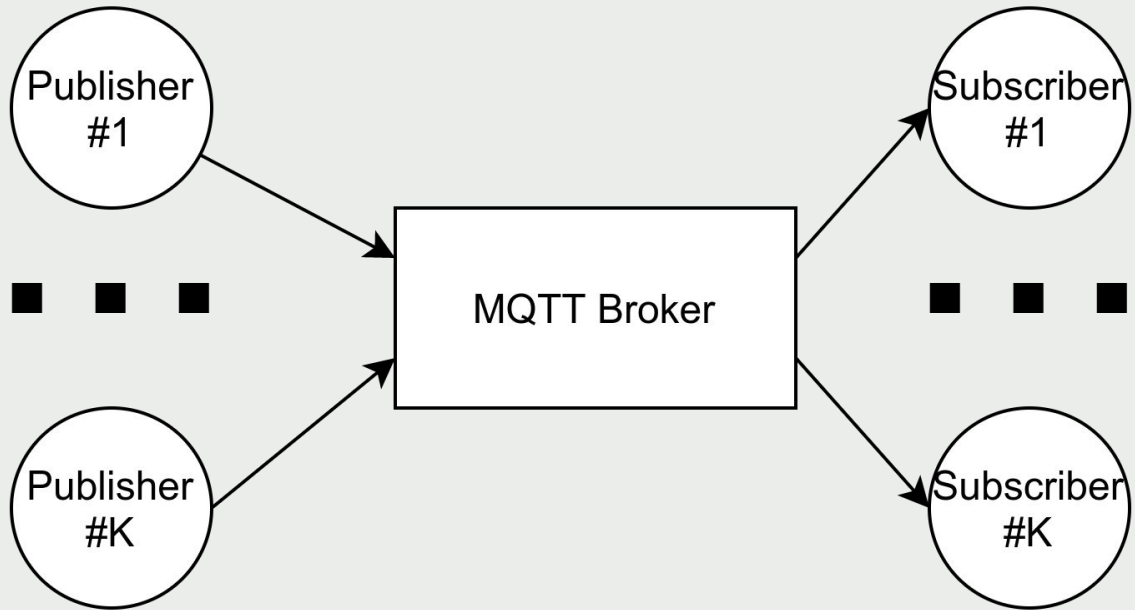
Giovani Oliveira
giovanni@ime.usp.br



IME

INSTITUTO DE MATEMÁTICA
E ESTATÍSTICA
UNIVERSIDADE DE SÃO PAULO

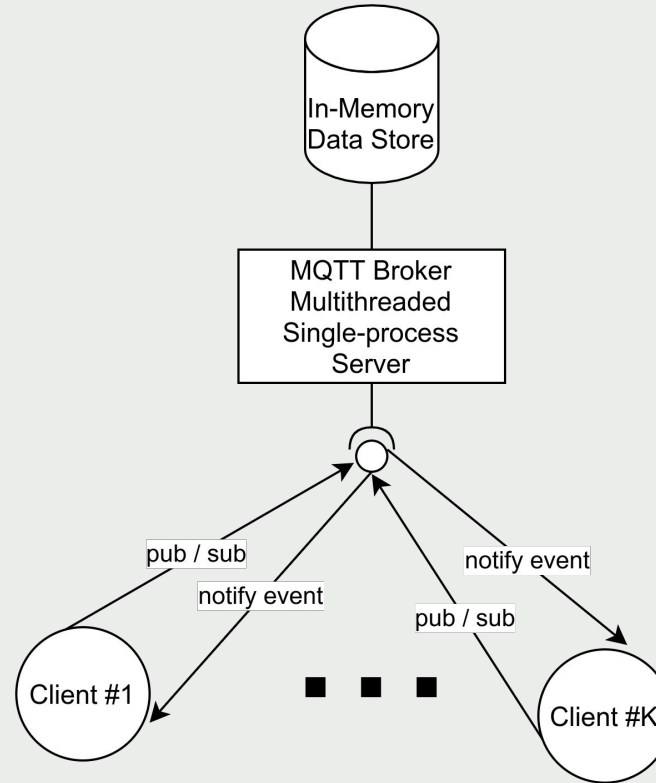
MQTT Protocol Generic View



- 1) Subscriber #i connects to MQTT broker;
- 2) Subscriber #i subscribes to topic "foo-bar";
- 3) Publisher #j connects to MQTT broker;
- 4) Publisher #j publishes message "lorem ipsum" in topic "foo-bar";
- 5) For each node subscribed to topic "foo-bar", MQTT broker sends message "lorem ipsum" (which includes Subscriber #i).

Proposed Solution Architecture

- Programming Language:
C (std. C99);
- Tested Compiler/Platform:
GNU GCC / Linux x64
- Build automation tool:
Cmake 3.16.3
- Concurrency Technology:
Multi-threading, Single-Process;
- Data store:
In-memory (volatile);
- Service Port:
Configurable (executable arg.);
- Maximum clients:
In-source (define MAXCLIENTS).



Features:

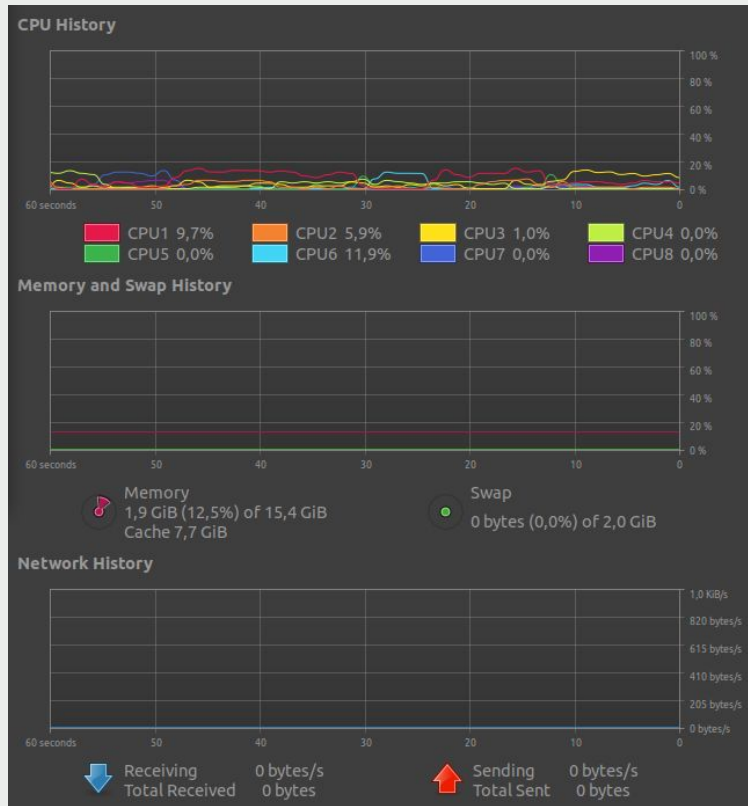
- Each client can be simultaneously listening to multiple topics and publishing to any topic;
- Connection state control and resource release on unexpected connection close;
- Packet type is checked against flag code to skip corrupted or non-MQTT packages.

Proposed Data Structure

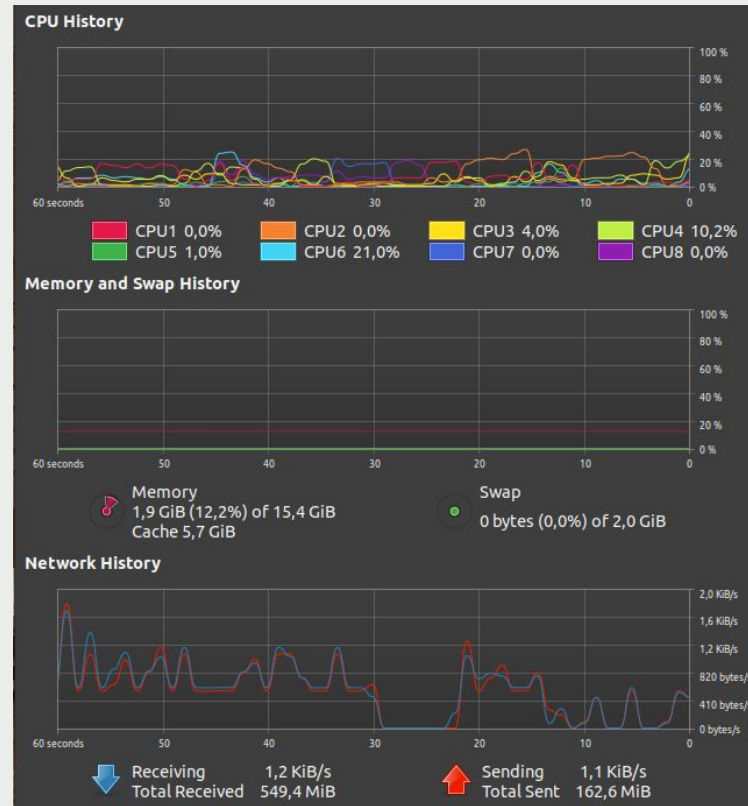
```
typedef struct {  
    int connfd;  
    pthread_t thread;  
    char *id;  
    enum connection_statuses connstat;  
    simple_set subscriptions;  
} thread_t;
```

- Each connection is stored into a reusable slot represented by a struct called “**thread_t**”, where:
 - **connfd** is the file description of the connection between client and broker;
 - **thread** is the OS-thread that is created to handle the single-client communication;
 - ***id** is a string with the identifier provided by the node at connection time;
 - **connection_statuses** is an enum to represent MQTT connection statuses (so far, CONNECTED and DISCONNECTED);
 - **subscriptions** is a set structure used to store subscriptions (topic names) of associated client. Set is imported from [4].

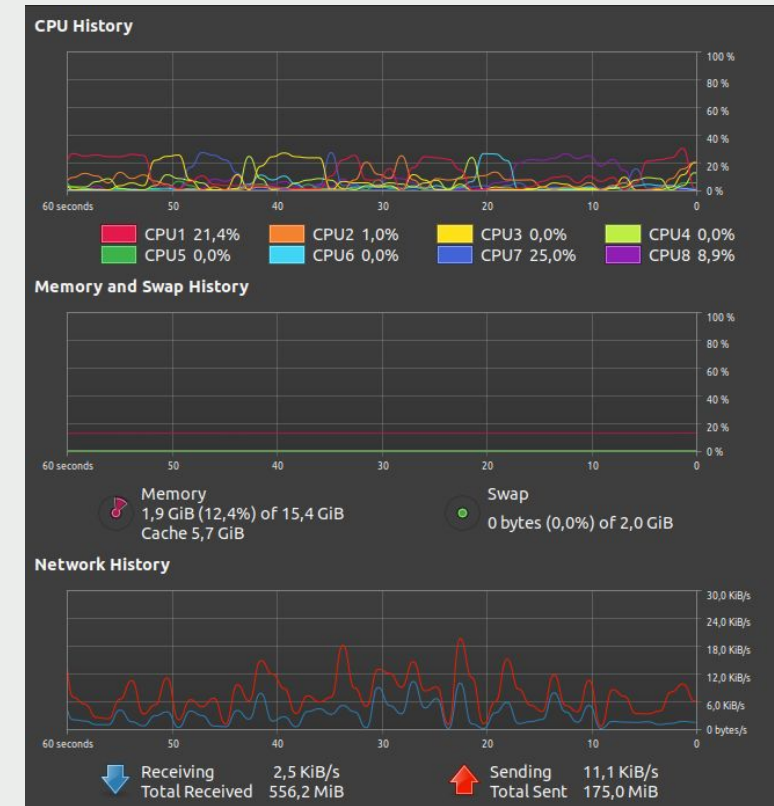
Performance Benchmarks on LAN



Case: Server-only listening to new connections



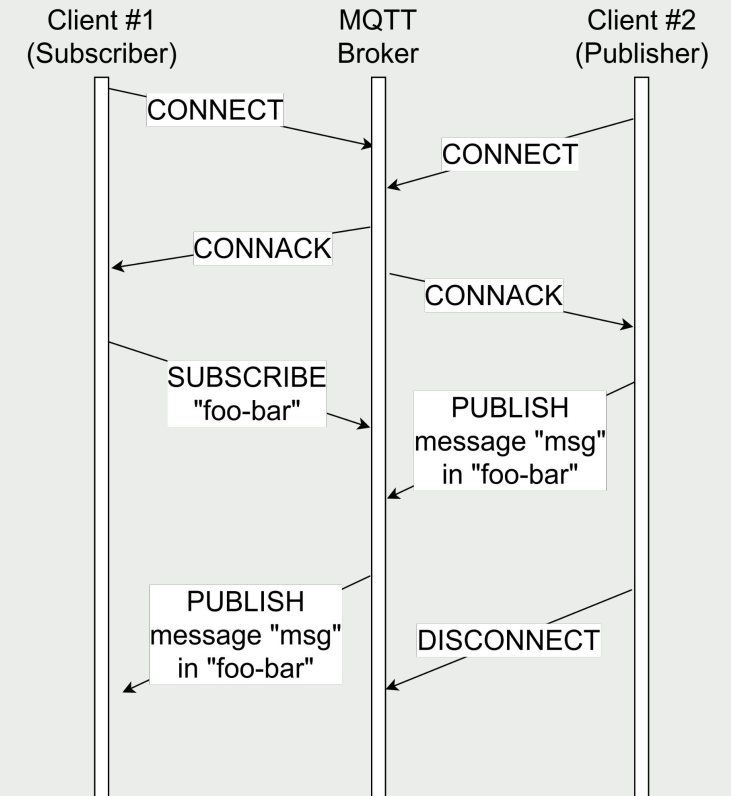
Case: Server listening to 2 clients
(1 publisher, 1 subscriber)



Case: Server listening to 100 clients
(50 publishers, 50 subscribers)

Supported Command Reference

- **CONNECT**: Client sends to Broker at connection initiation;
- **CONNACK**: Broker sends to Client at successful CONNECT request;
- **SUBSCRIBE**: Client sends to Broker to subscribe to one or more topics;
- **PUBLISH**: Client sends to Broker to publish a message into a single topic;
- **PINGREQ**: Client sends to Broker to check connection health;
- **PINGRESP**: Broker sends to Client to respond PINGREQ request;
- **DISCONNECT**: Client sends to Broker to close MQTT connection gracefully.



References

- [1] Standard, O. A. S. I. S. "MQTT version 3.1. 1." URL <http://docs.oasis-open.org/mqtt/mqtt/v3.1/> (2014).
- [2] Ritchie, Dennis M., Brian W. Kernighan, and Michael E. Lesk. The C programming language. Englewood Cliffs: Prentice Hall, 1988.
- [3] https://edisciplinas.usp.br/pluginfile.php/6485438/mod_assign/introattachment/0/mac5910-servidor-exemplo-ep1.c
- [4] <https://github.com/barrust/set>