

# Assignment 3

Data communication (CSC0056)

National Taiwan Normal University

Gábor Szolnok

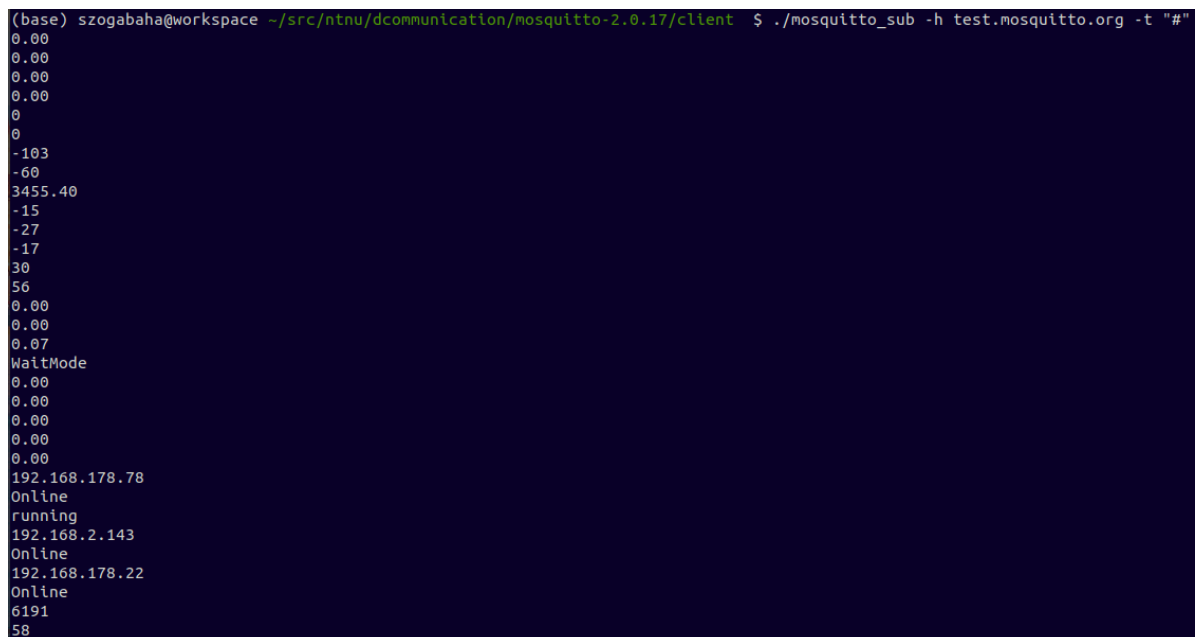
2023.11.13.

## **Task 1 (40 points). Using the public broker.**

### **Take a screenshot as an evidence that you did the peek**

As opposed to the description on the broker's website<sup>1</sup>, I couldn't use the -u and -v switches for the subscriber without being authenticated. I followed the suggestions of the public forum and subscribed to the "#" topic without specifying these parameters:

```
cd client //from the root of mosquitto
./mosquitto_sub -h test.mosquitto.org -t "#"
```



```
(base) szogabaha@workspace ~/src/ntnu/dcommunication/mosquitto-2.0.17/client $ ./mosquitto_sub -h test.mosquitto.org -t "#"
0.00
0.00
0.00
0.00
0
0
-103
-60
3455.40
-15
-27
-17
30
56
0.00
0.00
0.07
WaitMode
0.00
0.00
0.00
0.00
0.00
192.168.178.78
Online
running
192.168.2.143
Online
192.168.178.22
Online
6191
58
```

I could observe different kinds of data, doubles, strings, ip addresses as well as json files flowing through the broker.

---

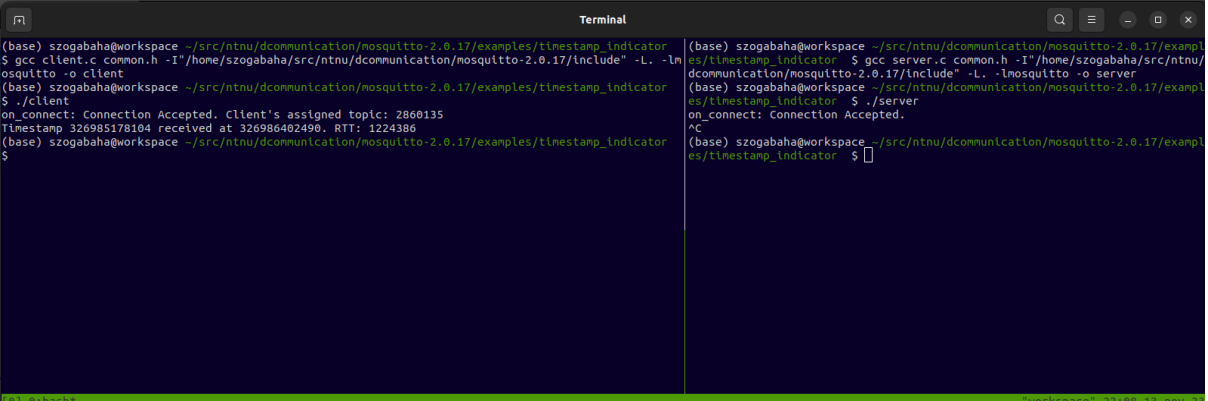
<sup>1</sup> <https://test.mosquitto.org/>

## Task 2 (60 points). Implementing a request/response.

Write your own client and server programs, both in the form of MQTT clients, and have your client send a request to the server to get the current time information there. Your code this time should not use command-line arguments, and every information should be embedded in the code itself.

I reused the public/basic-1.c subscribe/basic-1.c codes under the examples directory. The server uses the “ntnu/szogabaha” topic to receive requests from the clients.

- **Server:** it subscribes to the “ntnu/szogabaha” topic and waits for messages in an infinite loop. Upon receiving a message on the mentioned topic, it gathers the current timestamp (using CLOCK\_MONOTONIC clock\_gettime - exactly the same way as provided in the second assignment) and sends it to the topic that it received in the payload of the original request
- **Client:** the client generates a unique identifier as a topic. Then it sends a message with the “ntnu/szogabaha” topic, using the generated identifier as the payload. Then the client waits until a single message is received on the generated topic. Upon receiving the timestamp, the client prints the received timestamp, the current timestamp and the difference between the two (which is the round trip time).
  - “wait until a message arrives” - this mechanism is achieved by actively waiting in a while loop until a flag is set.
  - “generate a unique identifier” - this identifier is provided with the rnd library, the generator is salted with the PID of the client process, so that the identifier is different even if two clients are started at the same time.



```
(base) szogabaha@workspace ~/src/ntnu/dcommunication/mosquitto-2.0.17/examples/timestamp_indicator
$ gcc client.c common.h -I"/home/szogabaha/src/ntnu/dcommunication/mosquitto-2.0.17/include" -L. -ln
osquitto -o client
(base) szogabaha@workspace ~/src/ntnu/dcommunication/mosquitto-2.0.17/examples/timestamp_indicator
$ ./client
on_connect: Connection Accepted. Client's assigned topic: 2860135
Timestamp 326985178184 received at 326986402490, RTT: 1224386
(base) szogabaha@workspace ~/src/ntnu/dcommunication/mosquitto-2.0.17/examples/timestamp_indicator
$

(base) szogabaha@workspace ~/src/ntnu/dcommunication/mosquitto-2.0.17/examples/timestamp_indicator
$ gcc server.c common.h -I"/home/szogabaha/src/ntnu/dcommunication/mosquitto-2.0.17/include" -L. -ln
osquitto -o server
(base) szogabaha@workspace ~/src/ntnu/dcommunication/mosquitto-2.0.17/examples/timestamp_indicator
$ ./server
on_connect: Connection Accepted.
^C
(base) szogabaha@workspace ~/src/ntnu/dcommunication/mosquitto-2.0.17/examples/timestamp_indicator
$
```

An example of compiling and running the code can be found below. Since in the previous assignments we've learnt about how we can modify the source code, I

wanted to see how the modified and compiled code can be used here instead of installing mosquito as a package from some artifact storage.

- The {path\_to\_mosquitto} needs to be replaced with the path to the mosquito source code.
- The “libmosquitto.so” file needs to be put to the same place where client.c and server.c is put.
- “libmosquitto.so” is either a symbolic link or a copy to the compiled file “libmosquitto.so.1”, which is under the “lib” directory after compiling mosquito.

```
gcc client.c common.h -I"{path_to_mosquitto}/include" -L. -lmosquitto -o client
gcc server.c common.h -I"{path_to_mosquitto}/include" -L. -lmosquitto -o server

./client
./server
```

I also created a makefile to make the run easier. The CFLAGS needs to be adjusted (the -I flag should point to the mosquito/include directory).