6.5 Esercizi MQTT

6.5.1 Guida all'installazione di Mosquitto su Linux

Per installare un broker MQTT (Mosquitto) è necessario scrivere la seguente linea di comando:

```
sudo apt install mosquitto
```

Per avviare il broker si utilizza il comando:

```
sudo systemctl start mosquitto
```

Per verificare che il broker funzioni si utilizza il comando:

```
sudo systemctl status mosquitto
```

Per fermare il broker si utilizza il comando:

```
sudo systemctl stop mosquitto
```

L'installazione di un pub/sub MQTT è eseguito con il comando:

```
sudo apt install mosquitto-clients
```

Comandi utili:

Descrizione	Comando
Indirizzo IP broker	-h oppure --host
Topic messaggio	-t oppure --topic
Contenuto messaggio	-m oppure --message

6.5.2 Domande sull'esempio Pub/Sub

Dato l'esempio:

- mosquitto_sub -t "temperatura"
- mosquitto_pub -m "22" -t "temperatura"

Si risponda alle seguenti domande.

Partendo dall'esercizio mostrato nelle istruzioni, cosa succede se pubblico una temperatura prima di aver lanciato il subscriber?

Il publisher invia il valore 22 con il topic temperatura al broker attivo, ma la richiesta non viene consegnata a nessun subscriber poiché non esistono. Quindi, viene pubblicato il messaggio, ma nessuno lo riceve (eccetto il broker).

Partendo dall'esercizio mostrato nelle istruzioni, creare un'applicazione pub/sub con 2 sensori di temperatura relativi a 2 stanze diverse. Quante finestre di terminale devo aprire?

Dipende da quanto si vuole isolare ogni componente. Nel caso in cui si utilizzino 3 terminali:

- Subscribers:
  1. Terminale "archivio" della prima stanza:
     ```
     mosquitto_sub -t "temperatura/stanza1"
     ```
  2. Terminale "archivio" della seconda stanza
     ```
     mosquitto_sub -t "temperatura/stanza2"
     ```
- Publisher:
  1. Terminale "sensore di temperatura" delle due stanze. Quindi verrà eseguita una pubblicazione prima di una stanza:
     ```
     mosquitto_pub -m "22" -t "temperatura/stanza1"
     ```
     E poi dell'altra:
     ```
     mosquitto_pub -m "22" -t "temperatura/stanza2"
     ```

Altrimenti, nel caso in cui si voglia distinguere ogni termostato che pubblica la temperatura, allora:

- Subscribers:
  1. Terminale "archivio" della prima stanza:
     ```
     mosquitto_sub -t "temperatura/stanza1"
     ```
  2. Terminale "archivio" della seconda stanza
     ```
     mosquitto_sub -t "temperatura/stanza2"
     ```
- Publishers:
  1. Terminale "sensore di temperatura" della prima stanza. Quindi verrà eseguita una pubblicazione della stanza numero 1:
     ```
     mosquitto_pub -m "22" -t "temperatura/stanza1"
     ```
  2. Terminale "sensore di temperatura" della seconda stanza. Quindi verrà eseguita una pubblicazione della stanza numero 2:
     ```
     mosquitto_pub -m "22" -t "temperatura/stanza2"
     ```

Partendo dall'esercizio precedente, come fare per avere un unico subscriber per entrambe le temperature? Come si fa a distinguere da quale stanza proviene la temperatura?

Per avere un unico subscriber è necessario chiudere un terminale dei due subscriber attivi e l'unico che rimane, modificarlo con gli wildcards.

- Subscriber:
  1. Terminale "archivio unico" di entrambe le stanze:

```
mosquitto_sub -t "temperatura/#"
```

- Publishers:
  1. Terminale "sensore di temperatura" della prima stanza. Quindi verrà eseguita una pubblicazione della stanza numero 1:

```
mosquitto_pub -m "22-stanza1" -t "temperatura/stanza1"
```

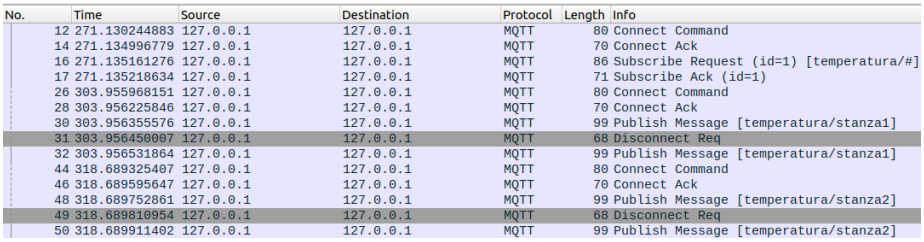
  2. Terminale "sensore di temperatura" della seconda stanza. Quindi verrà eseguita una pubblicazione della stanza numero 2:

```
mosquitto_pub -m "22-stanza2" -t "temperatura/stanza2"
```

Come si vede dal messaggio inviato, per capire la stanza dalla quale proviene, una soluzione possibile è modificare il messaggio aggiungendo la stanza di provenienza accanto al valore.

Usare Wireshark per capire quale protocollo viene utilizzato da tutti gli attori a livello 3 e 4. Quali sono le porte di livello trasporto coinvolte?

Come protocollo viene utilizzato MQTT:



No.	Time	Source	Destination	Protocol	Length	Info
12	271.130244883	127.0.0.1	127.0.0.1	MQTT	80	Connect Command
14	271.134996779	127.0.0.1	127.0.0.1	MQTT	70	Connect Ack
16	271.135161276	127.0.0.1	127.0.0.1	MQTT	86	Subscribe Request (id=1) [temperatura/#]
17	271.135218634	127.0.0.1	127.0.0.1	MQTT	71	Subscribe Ack (id=1)
26	303.955968151	127.0.0.1	127.0.0.1	MQTT	80	Connect Command
28	303.956225846	127.0.0.1	127.0.0.1	MQTT	70	Connect Ack
30	303.956355576	127.0.0.1	127.0.0.1	MQTT	99	Publish Message [temperatura/stanza1]
31	303.956450007	127.0.0.1	127.0.0.1	MQTT	68	Disconnect Req
32	303.956531864	127.0.0.1	127.0.0.1	MQTT	99	Publish Message [temperatura/stanza1]
44	318.689325407	127.0.0.1	127.0.0.1	MQTT	80	Connect Command
46	318.689595647	127.0.0.1	127.0.0.1	MQTT	70	Connect Ack
48	318.689752861	127.0.0.1	127.0.0.1	MQTT	99	Publish Message [temperatura/stanza2]
49	318.689810954	127.0.0.1	127.0.0.1	MQTT	68	Disconnect Req
50	318.689911402	127.0.0.1	127.0.0.1	MQTT	99	Publish Message [temperatura/stanza2]

Anche se nella figura viene visto il livello 3 e 4 poiché si ferma solo al livello 2, il protocollo è lo stesso.

Le porte utilizzate sono:

- Il subscriber utilizza la porta 38120;
- Il publisher della stanza 1 utilizza la porta 54474;
- Il publisher della stanza 2 utilizza la porta 60688.

La porta utilizzata dal broker, visibile poiché il pacchetto 16, 30 e 48 comunicano con lui e viceversa, è la numero 1883.

Prova a pubblicare un valore di umidità relativa (topic "UR"); il subscriber interessato alla temperatura lo riceve? Come si fa a creare un subscriber interessato all'umidità? Costruire un'applicazione pub/sub con 4 finestre per produrre e visualizzare sia valori di temperatura sia valori di umidità.

Il subscriber interessato alle temperature non riceve il messaggio del publisher sull'umidità, poiché il topic a cui è iscritto il subscriber è diverso.

Per creare un subscriber interessato all'umidità, è necessario aprire un terminale e scrivere il comando:

```
mosquitto_sub -t "UR"
```

Per costruire un'applicazione pub/sub con 4 finestre:

- Publishers:
  1. Terminale "sensore di temperatura". Quindi verrà eseguita una pubblicazione del tipo:
     ```
     mosquitto_pub -m "22" -t "temperatura"
     ```
  2. Terminale "sensore di umidità". Quindi verrà eseguita una pubblicazione del tipo:
     ```
     mosquitto_pub -m "5" -t "UR"
     ```
- Subscribers:
  1. Terminale "archivio temperature":
     ```
     mosquitto_sub -t "temperatura"
     ```
  2. Terminale "archivio umidità":
     ```
     mosquitto_sub -t "UR"
     ```

Si vuole costruire con MQTT un servizio di messaggistica universitaria:

- ***Il rettore può leggere tutti i messaggi***
- ***La segreteria può leggere i messaggi dai docenti e dagli studenti***
- ***I docenti possono leggere i messaggi dai docenti e dagli studenti***
- ***Gli studenti possono leggere solo i messaggi degli altri studenti***

Prima di descrivere i comandi da eseguire per creare l'applicazione, si presenta qua di seguito l'albero dei topic:

- Livello 1:
  - universita
- Livello 2:
  - universita/personale
  - universita/direzione
- Livello 3:
  - universita/personale/docenti
  - universita/personale/studenti

Adesso si impostano i 4 subscribers in 4 terminali diversi:

- Il rettore ha i permessi di leggere qualsiasi messaggio dell'università, quindi deve avere accesso a qualsiasi topic:

```
mosquitto_sub -t "universita/#"
```

- La segreteria deve avere la possibilità di leggere solo i messaggi provenienti dai docenti e dagli studenti:

```
mosquitto_sub -t "universita/personale/#"
```

- I docenti hanno gli stessi permessi della segreteria e quindi il comando sarà:

```
mosquitto_sub -t "universita/personale/#"
```

- Gli studenti hanno i permessi più restrittivi di tutti, ovvero possono solo leggere i messaggi provenienti da altri studenti:

```
mosquitto_sub -t "universita/personale/studenti"
```