24-1 IoT실습 Lab8

22100579 이진주

export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH

실습#0

Broker, subscriber, publisher에서 기본 명령 실행

```
leeejjju@DESKTOP-3DANK54:~/IoT/lab8/broker$ mosquitto -v
1717084800: mosquitto version 2.0.18 starting
1717084800: Using default config.
1717084800: Starting in local only mode. Connections will only be possible from cli
ents running on this machine.
1717084800: Create a configuration file which defines a listener to allow remote ac
cess.
1717084800: For more details see https://mosquitto.org/documentation/authentication
-methods/
1717084800: Opening ipv4 listen socket on port 1883.
1717084800: Opening ipv6 listen socket on port 1883.
1717084800: mosquitto version 2.0.18 running
1717085458: New connection from 127.0.0.1:55834 on port 1883.
1717085458: New client connected from 127.0.0.1:55834 as auto-C53CAC48-4712-65FF-12
6F-41DF1FF0935B (p2, c1, k60).
1717085458: No will message specified.
1717085458: Sending CONNACK to auto-C53CAC48-4712-65FF-126F-41DF1FF0935B (0, 0)
1717085458: Received SUBSCRIBE from auto-C53CAC48-4712-65FF-126F-41DF1FF0935B
1717085458:     handong (QoS 0)
1717085458: auto-C53CAC48-4712-65FF-126F-41DF1FF0935B 0 handong
1717085458: Sending SUBACK to auto-C53CAC48-4712-65FF-126F-41DF1FF0935B
1717085518: Received PINGREQ from auto-C53CAC48-4712-65FF-126F-41DF1FF0935B
1717085518: Sending PINGRESP to auto-C53CAC48-4712-65FF-126F-41DF1FF0935B
1717085533: New connection from 127.0.0.1:44182 on port 1883.
1717085533: New client connected from 127.0.0.1:44182 as auto-3C666FAA-19C4-E996-20
85-20CA340DEAE3 (p2, c1, k60).
1717085533: No will message specified.
1717085533: Sending CONNACK to auto-3C666FAA-19C4-E996-2085-20CA340DEAE3 (0, 0)
1717085533: Received PUBLISH from auto-3C666FAA-19C4-E996-2085-20CA340DEAE3 (d0, q0
, r0, m0, 'handong', ... (5 bytes))
1717085533: Sending PUBLISH to auto-C53CAC48-4712-65FF-126F-41DF1FF0935B (d0, q0, r
0, m0, 'handong', ... (5 bytes))
```

```
leeejjju@DESKTOP-3DANK54:~/IoT/lab8/subscriber$ mosquitto_sub -v -t handong
handong hello
```

```
leeejjju@DESKTOP-3DANK54:~/IoT/lab8/publishser$ mosquitto_pub -t handong -m hello
```

실습#1

Subscriber #1: $ mosquitto_sub -v -t handong/csee/# 실행 결과

```
leeejjju@DESKTOP-3DANK54:~/IoT/lab8/subscriber$ mosquitto_sub -v -t handong/csee/#
handong/csee/a 2222
handong/csee/b 3333
handong/csee/c 4444
handong/csee 55555
```

위 명령어에 사용된 와일드카드는 #으로, handong/csee/#으로 시작하는 모든 주제를 구독하고자 함을 의미한다. 때문에 handong/csee/아래의 토픽 아래로 들어온 2,3,4,5를 수신하고, handing/mec 등 다른 토픽 아래의 메시지는 수신하지 않았다.

Subscriber #2: $ mosquitto_sub -v -t handong/+/b 실행 결과

```
leeejjju@DESKTOP-3DANK54:~/IoT/lab8/subscriber$ mosquitto_sub -v -t handong/+/b
handong/csee/b 3333
handong/mec/b 7777
```

위 명령어에 사용된 와일드카드는 +으로, handing 하위의 어떤 토픽이던, 그 아래 b의 토픽으로 들어온 메시지를 구독하고자 함을 의미한다. 때문에 handing/(csee, mec 등 아무거나)/b 토픽으로 들어온 3, 7의 메시지를 수신하였다. 유의할 점은, + 와일드카드는 단일 레벨의 토픽만 커버하므로 handing/여기에/하나/이상/껴있으면/b 토픽은 구독하지 않는다.

실습#2

Wireshark) QoS Level 1과 QoS Level 2에 대한 MQTT 패킷 분석

1. 각 QoS Level에 대해 Publisher Client의 CONNECT~DISCONNECT 과정과 Broker의 메시지 Publish 과정을 MQTT 레벨에서 설명
   - 두 레벨 모두 기본적인 절차는 CONNECT-PUB-DISCONNECT, CONNECT-SUB-DISCONNECT의 흐름을 따르나, PUB 절차에서 차이가 난다. 아래는 QoS레벨 1에서 subscribe하고 두 번 publish한 뒤 연결을 끊은 과정의 패킷 캡쳐 화면이다.

No.	Time	Source	Destin	Protocc	Length	Info	src, dest, type
4	0.…	::1	::1	MQTT	102	Connect Command	42898,1883,1
6	0.…	::1	::1	MQTT	92	Connect Ack	1883,42898,2
8	0.…	::1	::1	MQTT	99	Subscribe Request (id=1) [test]	42898,1883,8
9	0.…	::1	::1	MQTT	93	Subscribe Ack (id=1)	1883,42898,9
16	4.…	::1	::1	MQTT	102	Connect Command	36076,1883,1
18	4.…	::1	::1	MQTT	92	Connect Ack	1883,36076,2
20	4.…	::1	::1	MQTT	109	Publish Message (id=1) [test]	36076,1883,3
21	4.…	::1	::1	MQTT	109	Publish Message (id=1) [test]	1883,42898,3
23	4.…	::1	::1	MQTT	92	Publish Ack (id=1)	1883,36076,4
24	4.…	::1	::1	MQTT	92	Publish Ack (id=1)	42898,1883,4
25	4.…	::1	::1	MQTT	90	Disconnect Req	36076,1883,14
37	11…	::1	::1	MQTT	102	Connect Command	36092,1883,1
39	11…	::1	::1	MQTT	92	Connect Ack	1883,36092,2
41	11…	::1	::1	MQTT	109	Publish Message (id=1) [test]	36092,1883,3
42	11…	::1	::1	MQTT	109	Publish Message (id=2) [test]	1883,42898,3
43	11…	::1	::1	MQTT	92	Publish Ack (id=1)	1883,36092,4
44	11…	::1	::1	MQTT	92	Publish Ack (id=2)	42898,1883,4
46	11…	::1	::1	MQTT	90	Disconnect Req	36092,1883,14
50	14…	::1	::1	MQTT	90	Disconnect Req	42898,1883,14

   - 첫 Connect 요청의 방향을 보아, broker의 port는 1883, subscriber의 port는 42898로 보인다. 다음 connect를 보니 Publisher의 port는 36076일 것이다.
   - Subscriber와 broker의 connect, connect ack 패킷이 오고 갔고, 이어서 subscribe 요청과 ack이 오고 갔다.

- publish하는 과정에서, publisher(36076)가 broker(1883)에게 publish message를 보낸 후 broker는 즉시 subscriber(42898)에게 message를 전달하는 모습을 볼 수 있다. 이후 broker->publisher로의 ack과 subscriber->broker로의 ack이 응답된다.
- QoS Level 1의 원칙에 따라서, Publish 과정에서 message가 적어도 한 번 전달됨을 보장하기 위하여 publish ack을 두는 모습이다.
- 아래는 QoS Level 2에서 같은 과정을 수행했을 때의 패킷 캡쳐 화면이다.

No.	Time	Source	Destin	Protocc	Length	Info	src, dest, type
4	0.…	::1	::1	MQTT	102	Connect Command	56718,1883,1
6	0.…	::1	::1	MQTT	92	Connect Ack	1883,56718,2
8	0.…	::1	::1	MQTT	99	Subscribe Request (id=1) [test]	56718,1883,8
9	0.…	::1	::1	MQTT	93	Subscribe Ack (id=1)	1883,56718,9
14	2.…	::1	::1	MQTT	102	Connect Command	46596,1883,1
16	2.…	::1	::1	MQTT	92	Connect Ack	1883,46596,2
18	2.…	::1	::1	MQTT	109	Publish Message (id=1) [test]	46596,1883,3
19	2.…	::1	::1	MQTT	92	Publish Received (id=1)	1883,46596,5
20	2.…	::1	::1	MQTT	92	Publish Release (id=1)	46596,1883,6
21	2.…	::1	::1	MQTT	109	Publish Message (id=1) [test]	1883,56718,3
23	2.…	::1	::1	MQTT	92	Publish Received (id=1)	56718,1883,5
24	2.…	::1	::1	MQTT	92	Publish Complete (id=1)	1883,46596,7
25	2.…	::1	::1	MQTT	90	Disconnect Req	46596,1883,14
27	2.…	::1	::1	MQTT	92	Publish Release (id=1)	1883,56718,6
30	2.…	::1	::1	MQTT	92	Publish Complete (id=1)	56718,1883,7
35	8.…	::1	::1	MQTT	102	Connect Command	46606,1883,1
37	8.…	::1	::1	MQTT	92	Connect Ack	1883,46606,2
39	8.…	::1	::1	MQTT	110	Publish Message (id=1) [test]	46606,1883,3
40	8.…	::1	::1	MQTT	92	Publish Received (id=1)	1883,46606,5
41	8.…	::1	::1	MQTT	92	Publish Release (id=1)	46606,1883,6
42	8.…	::1	::1	MQTT	110	Publish Message (id=2) [test]	1883,56718,3
43	8.…	::1	::1	MQTT	92	Publish Complete (id=1)	1883,46606,7
44	8.…	::1	::1	MQTT	92	Publish Received (id=2)	56718,1883,5
46	8.…	::1	::1	MQTT	92	Publish Release (id=2)	1883,56718,6
47	8.…	::1	::1	MQTT	90	Disconnect Req	46606,1883,14
51	8.…	::1	::1	MQTT	92	Publish Complete (id=2)	56718,1883,7
55	11…	::1	::1	MQTT	90	Disconnect Req	56718,1883,14

- 여기서 broker는 1883, subscriber는 56718, publisher는 46596의 포트로 보인다.
- Connect, disconnect, subscribe 과정은 QoS level 1과 동일하나, publish 과정에서 차이를 보인다. Publish message를 받는 즉시 subscriber에게 보냈던 이전과 달리, publish received를 답변하고 publish release를 받은 후에야 subscriber에게 publish message를 전달한다. 그리고 publish complete를 publisher에게 답해줌으로서 publish 절차를 마무리한다.
- QoS Level 2의 경우 publish할 메시지에 대해 엄밀히 한 번의 전송을 보장하기 위해서 위와 같은 더 복잡한 절차를 걸치는 것이다.

2. Mosquitto에서 사용된 Clean session flag의 default값은 무엇이고, 무엇을 의미하는가?

```
▾ Connect Flags: 0x02, QoS Level: At most once delivery (Fire and Forget), Clean Ses
    0... .... = User Name Flag: Not set
    .0.. .... = Password Flag: Not set
    ..0. .... = Will Retain: Not set
    ...0 0... = QoS Level: At most once delivery (Fire and Forget) (0)
    .... .0.. = Will Flag: Not set
    .... ..1. = Clean Session Flag: Set
    .... ...0 = (Reserved): Not set
```

- Clean session bit 0은 기존 세션이 있다면 유지하겠음을, 1은 기존 세션이 있더라도 초기화하고 새로운 세션을 시작하겠음을 의미한다. 위 패킷들의 connect 부분 flag를 확인한 결과 mosquitto의 디폴트 clean session은 1로 보인다.

실습#3

Wireshark) Last Will and Testament 기능과 관련된 MQTT 패킷 분석.

- 기재된 시나리오의 실행 결과 패킷 캡쳐는 다음과 같다.

No.	Time	Source	Destin	Protocc	Length	Info	src, dest, type
4	0.…	::1	::1	MQTT	148	Connect Command	42380,1883,1
6	0.…	::1	::1	MQTT	92	Connect Ack	1883,42380,2
8	0.…	::1	::1	MQTT	109	Subscribe Request (id=1) [hando…	42380,1883,8
9	0.…	::1	::1	MQTT	93	Subscribe Ack (id=1)	1883,42380,9
14	4.…	::1	::1	MQTT	102	Connect Command	49190,1883,1
16	4.…	::1	::1	MQTT	92	Connect Ack	1883,49190,2
18	4.…	::1	::1	MQTT	109	Subscribe Request (id=1) [hando…	49190,1883,8
19	4.…	::1	::1	MQTT	93	Subscribe Ack (id=1)	1883,49190,9
24	7.…	::1	::1	MQTT	102	Connect Command	49200,1883,1
26	7.…	::1	::1	MQTT	92	Connect Ack	1883,49200,2
28	7.…	::1	::1	MQTT	111	Publish Message [handong/csee/c]	49200,1883,3
29	7.…	::1	::1	MQTT	90	Disconnect Req	49200,1883,14
30	7.…	::1	::1	MQTT	111	Publish Message [handong/csee/c]	1883,42380,3
36	10…	::1	::1	MQTT	134	Publish Message [handong/status]	1883,49190,3

- Broker는 1883, will-message를 설정한 sub1는 42380, will-message가 보내질 topic을 구독할 sub2는 49190, publisher는 49200의 포트를 가지고 있다.
- 이전과 마찬가지로 connection, subscribe, publish의 요청이 잘 이루어졌다. (QoS 레벨은 0으로 보인다)
- 회색으로 표시된 비정상 종료(subscriber1이 실행되던 터미널 창을 닫음) 패킷이 broker에 전달된 뒤, broker가 handing/status를 구독중인 sub2에게 last-will-message를 전송하는 모습을 볼 수 있다.

```
Msg Len: 44
Topic Length: 14
Topic: handong/status
Message: Subscrber #1 is disconnected
```

- Sub1이 위 케이스와 같이 비정상 종료 되자 sub1이 subscribe할 때 설정된 --will-payload 옵션에서 지정된 메시지로, --will-topic 옵션을 통해 지정된 토픽을 향해 publish되어 위와 같은 결과가 나타난 것이다.

A. 클라이언트의 Last Will Message는 Broker에 어떻게 등록되는가? 어떤 패킷과 어떤 Field를 사용하는가?

/*TODO 스크린 캡쳐*/

- Last will message는 connection 패킷을 통해 broker에 등록되게 된다.

```
▾ Connect Flags: 0x06, QoS Level: At most once delivery (Fire and Forget), Will Flag
    0... .... = User Name Flag: Not set
    .0.. .... = Password Flag: Not set
    ..0. .... = Will Retain: Not set
    ...0 0... = QoS Level: At most once delivery (Fire and Forget) (0)
    .... .1.. = Will Flag: Set
    .... ..1. = Clean Session Flag: Set
    .... ...0 = (Reserved): Not set
```

- 위는 Sub1의 connection 요청 패킷 속 connection flags 부분이다. Will flag가 1로 설정되었으므로 last will message를 사용할 것임을 알 수 있고, will message의 QoS는 0이며, will message를 retain하지 않을 것임을 알 수 있다.

```
Keep Alive: 60
Client ID Length: 0
Client ID:
Will Topic Length: 14
Will Topic: handong/status
Will Message Length: 28
Will Message: Subscrber #1 is disconnected
```

- 위는 connection 요청 패킷의 payload 부분이다. Will topic과 will message가 포함되어 있어, 해당 정보를 broker가 저장하고 관리할 수 있도록 해준다.

실습#4

새로이 접속하는 사용자들에게 가장 쾌적한 환경의 서버를 안내하는 publisher가 있다고 가정해보자. 해당 publisher는 서버들의 환경이 변화해 가장 쾌적한 서버 1순위가 변동될 때 마다 해당 서버의 ip를 publish한다.

subscriber들은 쾌적한 환경 정보를 받아와 새로운 통신이 필요할 때 마다 최신 정보의 쾌적한 서버와 통신을 시도할 것이다.

이 경우, 만약 서버들의 상태가 유지되어 쾌적한 순위의 변동이 없는 상황에서, 새로운 사용자가 접속하여 사용자의 subscriber가 쾌적한 서버 정보를 얻고자 하였을 때, retain message를 활용하면 서버 순위가 업데이트되고 새로운 메시지가 publish되기를 기다릴 필요 없이 subscribe를 한 즉시 현 시점에서의 최신 정보를 제공받을 수 있게 된다.

Broker가 구동된다.

```
leeejju@DESKTOP-3DANK54:~$ mosquitto -v
1717257494: mosquitto version 2.0.18 starting
1717257494: Using default config.
1717257494: Starting in local only mode. Connections will only be possible from clients running on this machine.
1717257494: Create a configuration file which defines a listener to allow remote access.
1717257494: For more details see https://mosquitto.org/documentation/authentication-methods/
1717257494: Opening ipv4 listen socket on port 1883.
1717257494: Opening ipv6 listen socket on port 1883.
1717257494: mosquitto version 2.0.18 running
```

Subscriber 1이 구독을 시작한다.

```
leeejju@DESKTOP-3DANK54:~$ mosquitto_sub -v -t info/server
```

Publisher가 세 차례 정보를 업데이트한다.

```
leeejju@DESKTOP-3DANK54:~$ mosquitto_pub -t info/server -m 192.168.120.24 -r
leeejju@DESKTOP-3DANK54:~$ mosquitto_pub -t info/server -m 192.168.120.43 -r
leeejju@DESKTOP-3DANK54:~$ mosquitto_pub -t info/server -m 192.168.120.49 -r
leeejju@DESKTOP-3DANK54:~$
```

Subscriber 1은 업데이트되는 정보를 실시간으로 받아본다.

```
leeejju@DESKTOP-3DANK54:~$ mosquitto_sub -v -t info/server
info/server 192.168.120.24
info/server 192.168.120.43
info/server 192.168.120.49
```

Publisher는 이후 정보 업데이트를 멈추었다.

Subscriber 2가 뒤늦게 들어오자마자, -r 옵션으로 지정된 마지막 토픽, 즉 최신의 retain message를 받아볼 수 있게 된다.

```
leeejju@DESKTOP-3DANK54:~$ mosquitto_sub -v -t info/server
info/server 192.168.120.49
```

만약 publisher가 -r 옵션을 지정하지 않았다면, subscriber 2는 쾌적한 서버순위 1순위에 변동이 있을 때까지 연결할 서버 정보를 받지 못한 채 기다려야만 했을 것이다.