

IoT 특론

9차시

AI첨단기술학과

이의혁

3. 사물 인터넷 통신

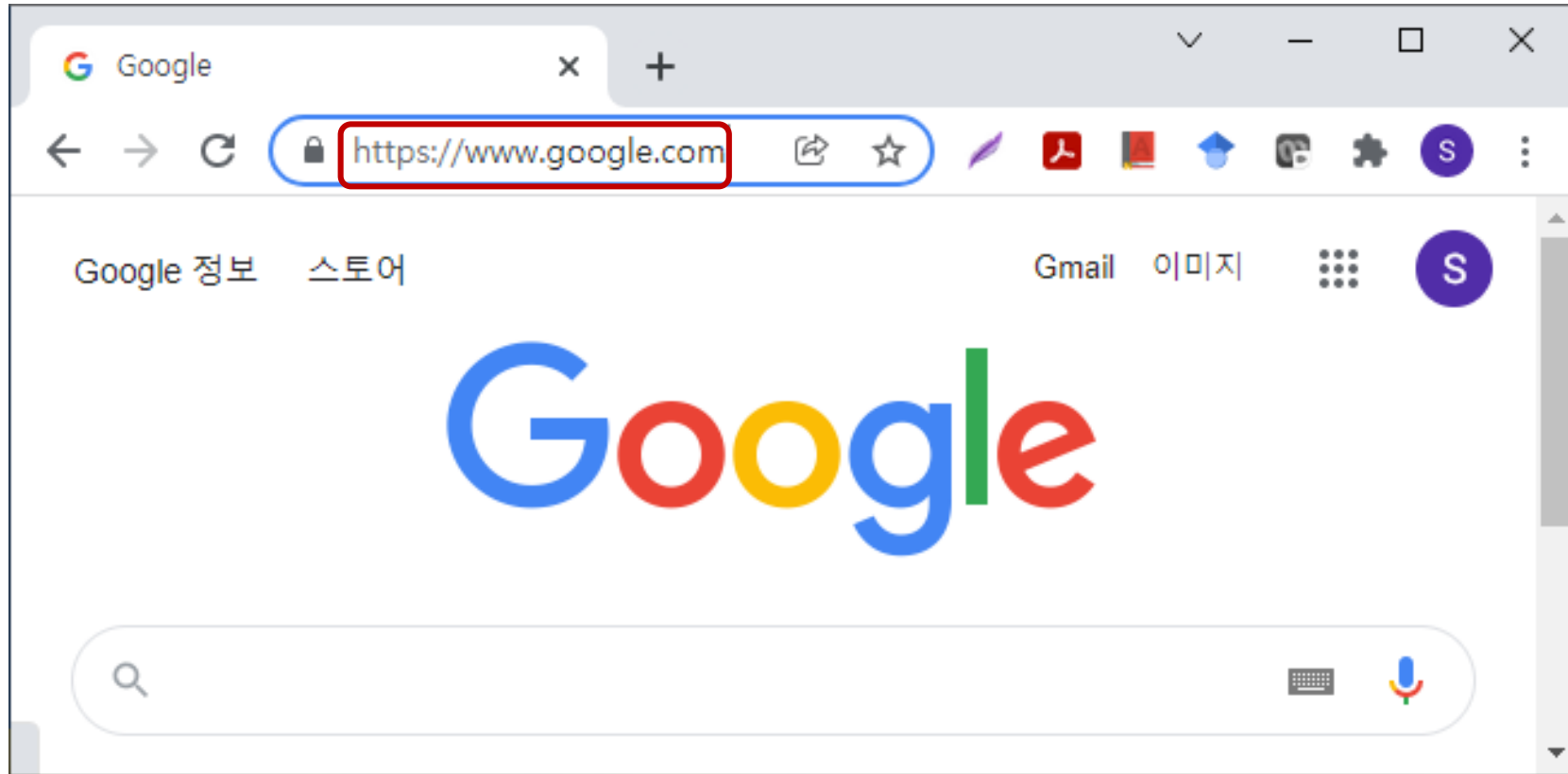
3-1. IoT 통신 프로토콜

프로토콜?

- 인터넷에서 사용되는 프로토콜에는 어떤 것들이 있을까요
 - ??

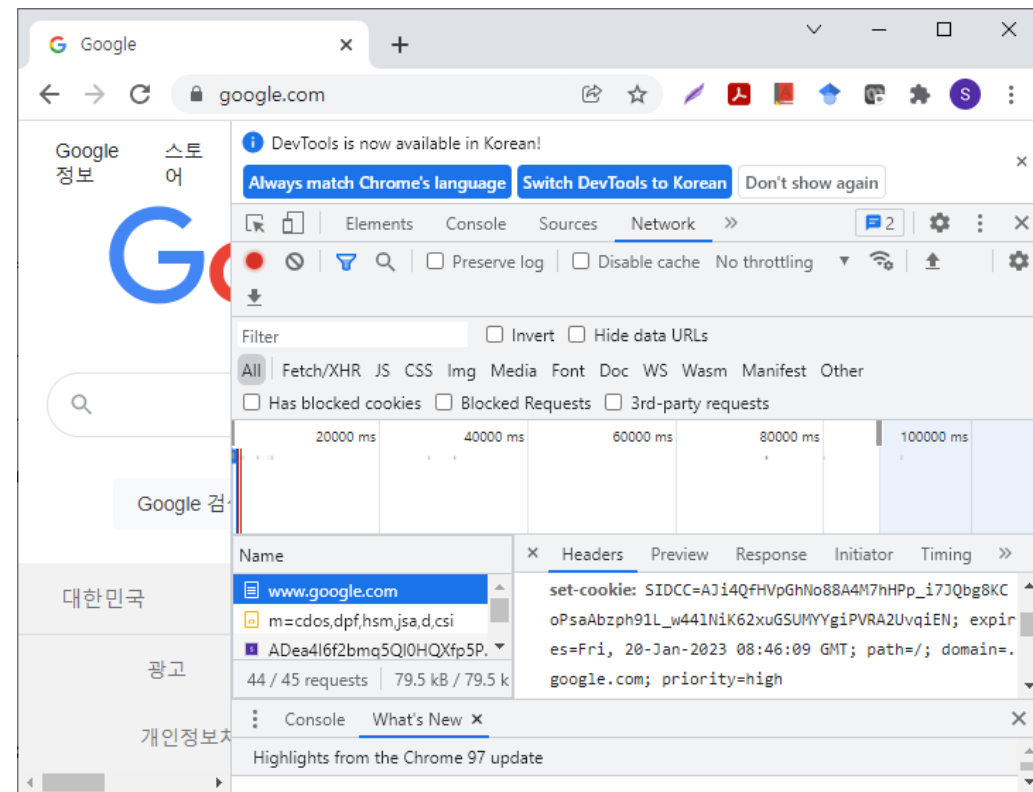
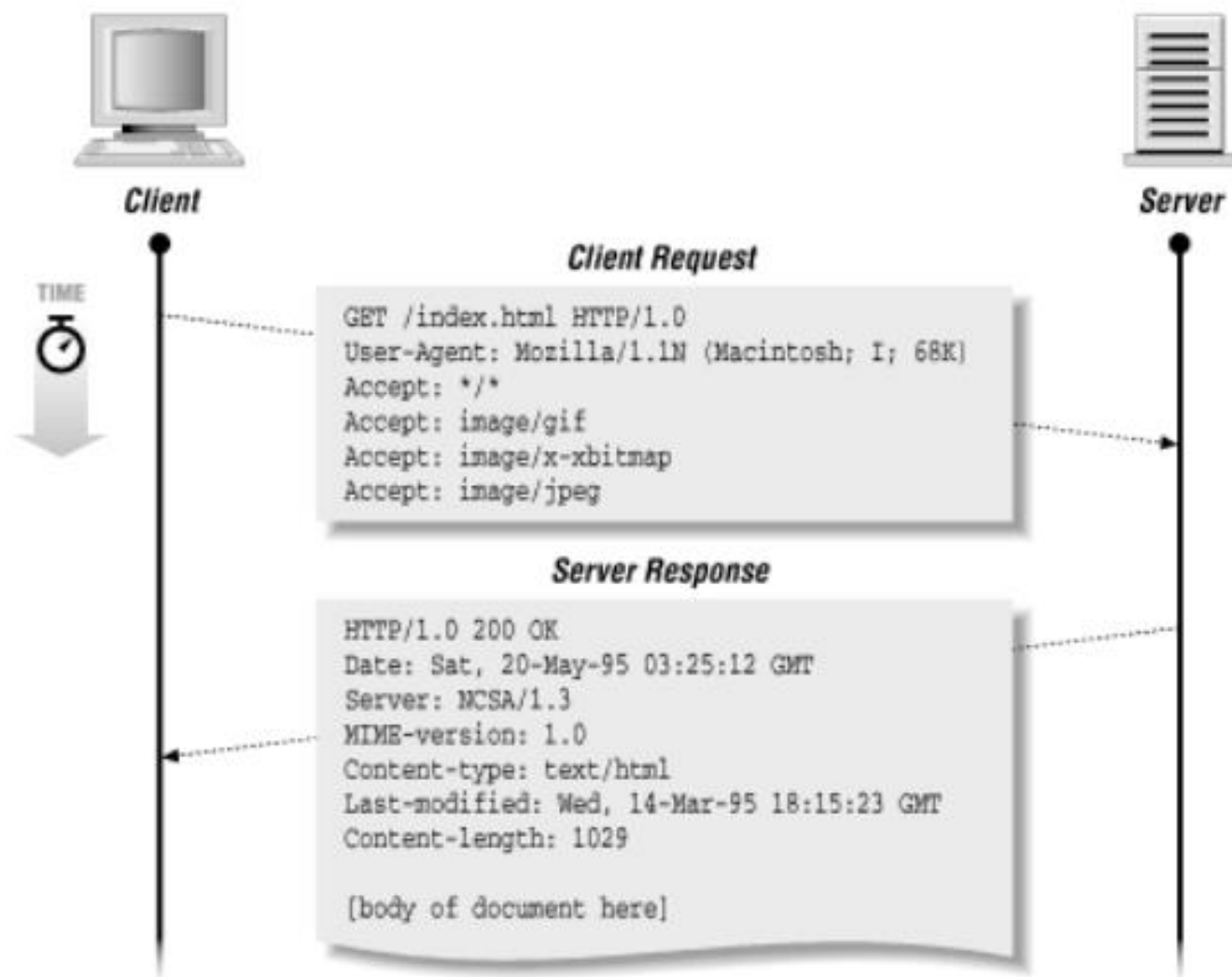
네트워크 프로토콜

- 컴퓨터 네트워크 환경에서 네트워크 연결된 컴퓨터 사이에 서로 데이터를 주고 받기 위해 컴퓨터와 네트워크 장비들이 지켜야 하는 사전 정의된 규약/규칙
 - 규칙에 맞게 데이터를 작성해서 전송해야 그것을 수신하는 측에서 데이터를 받고 이해할 수 있음
- 웹 브라우저에서 웹 사이트에 접속하여 해당 웹 페이지를 볼 수 있는 것은 웹 브라우저와 웹 서버 사이에 HTTP/HTTPS 프로토콜에 따라서 데이터가 송수신 되기 때문



- 브라우저에 `www.google.com` 을 입력하면 앞에 `https://` 이 추가되어 있는 것을 볼 수 있음
- 여기서 `https`가 사용하는 프로토콜을 나타냄

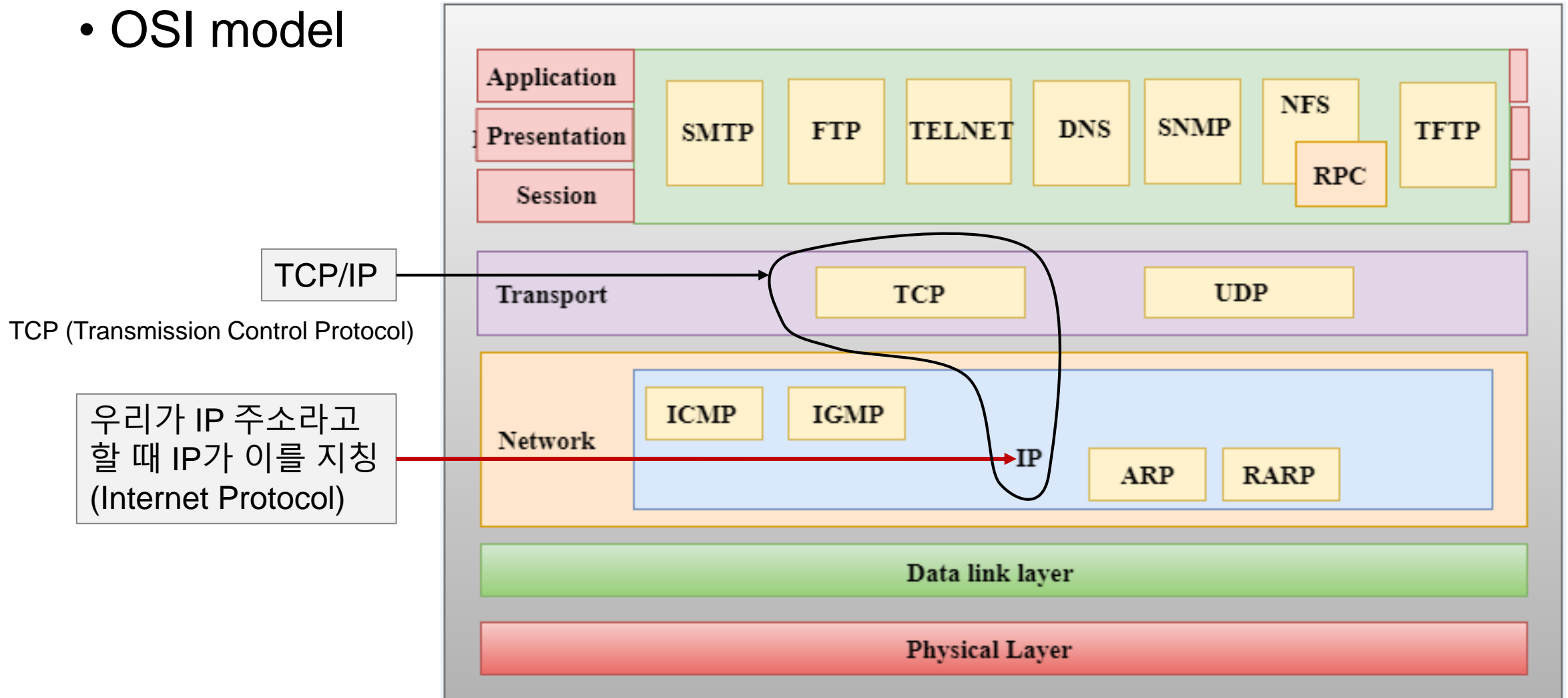
HTTP request/response



크롬 브라우저에서 확인하는 방법
F12 키 → Network 탭 → Headers

네트워크 계층 모델

- OSI model



대표적인 인터넷 응용 프로토콜

- HTTP/HTTPS
- SMTP
- POP3
- FTP
- DHCP
- SNMP
- Telnet
- SSH
- RFB

대표적인 인터넷 응용 프로토콜

- HTTP/HTTPS (Hyper Text Transfer Protocol (Secure))
- SMTP (Simple Mail Transfer Protocol)
 - 이메일을 메일 서버로 전송하는 데 사용되는 프로토콜
- POP3 (Post Office Protocol 3)
 - 메일 서버에 전송된 이메일을 수신하는 데 사용되는 프로토콜
- FTP (File Transfer Protocol)
 - 파일 전송 프로토콜
- DHCP (Dynamic Host Configuration Protocol)
 - 동적으로 IP 주소를 할당하는 데 사용되는 프로토콜
- SNMP (Simple Network Management Protocol)
 - 네트워크 관리와 모니터링을 위한 프로토콜
- Telnet
 - 원격 서버에 접속을 위한 프로토콜
- SSH (Secure Shell)
- RFB (Remote Frame Buffer)
 - VNC에서 사용

사물 인터넷 프로토콜

- 왜 사물 인터넷 통신 프로토콜이 새롭게 필요할까?
- 사물 인터넷 통신을 위해서는 어떤 점을 고려해야 할까?

IoT 응용 계층 프로토콜

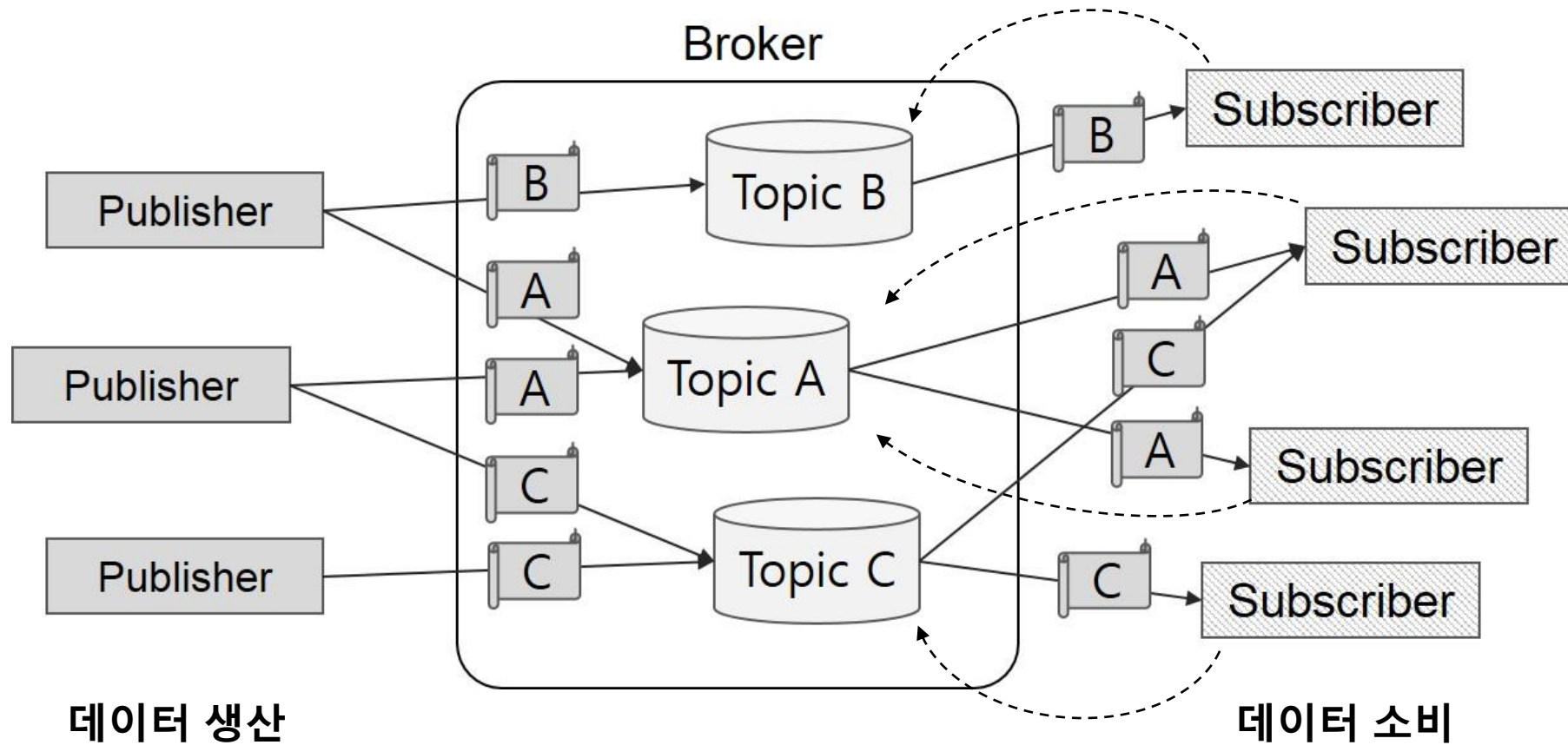
- MQTT
 - Message Queuing Telemetry Transport
- CoAP
 - Constrained Application Protocol
- AMQP
 - Advanced Message Queuing Protocol
- XMPP
 - Extensible Message and Presence Protocol

...

MQTT (Message Queuing Telemetry Transport)

- 경량의 브로커 기반 발행/구독(Publish/Subscribe) 메시징 프로토콜
 - 사물 지능 통신(M2M), 사물 인터넷을 위한 연결(connectivity) 프로토콜로 주목
 - 1999년에 IBM과 Arcom(현 Eurotech)에서 원격 검침(telemetry) 분야에서 사용하기 위해 만듦
 - 통신 대역폭이 제한적이고, 불안정하고 지연시간이 긴 네트워크 상황, 클라이언트에서 사용 가능한 자원의 제약이 큰 상황에서도 동작할 수 있도록 단순화, 자원 사용의 최소화, 빈번한 네트워크 장애 대비, 서비스 품질 제공 등을 주요 고려 사항으로 하여 설계됨
 - 가장 작은 메시지 사이즈가 2바이트 밖에 되지 않아 센서 장치나 원격 검침 장치 또는 모바일 기기 간의 연결 및 메시지 교환을 위한 용도로 사용하는데 장점

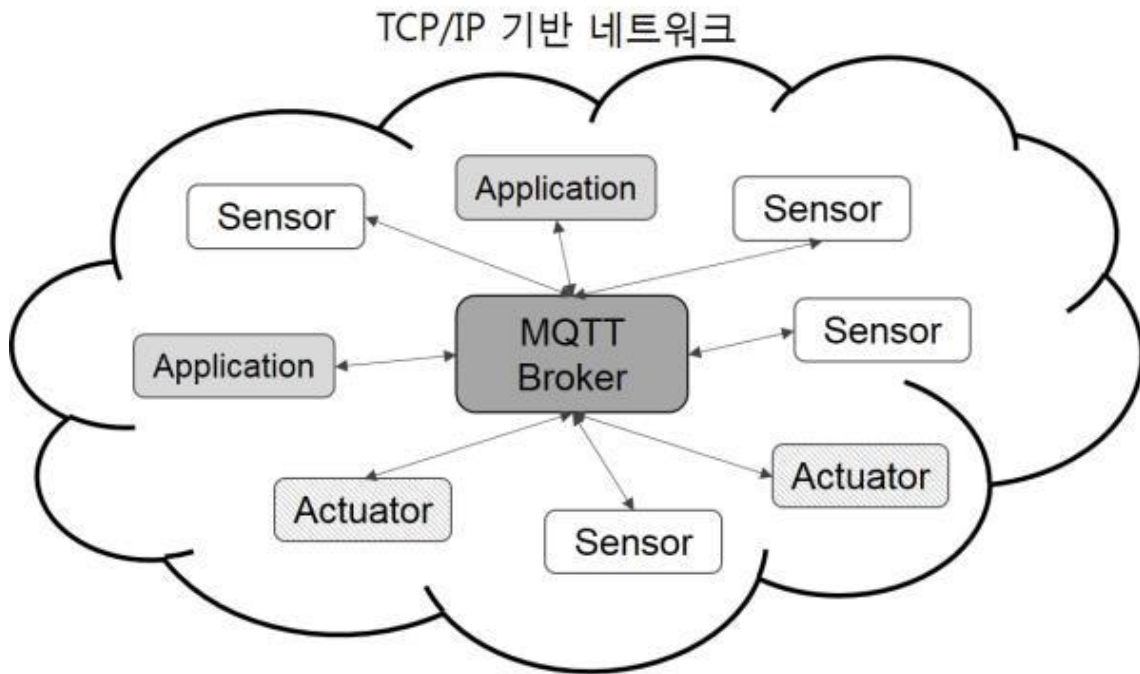
발행/구독(Publish/Subscribe) 모델



MQTT 특징

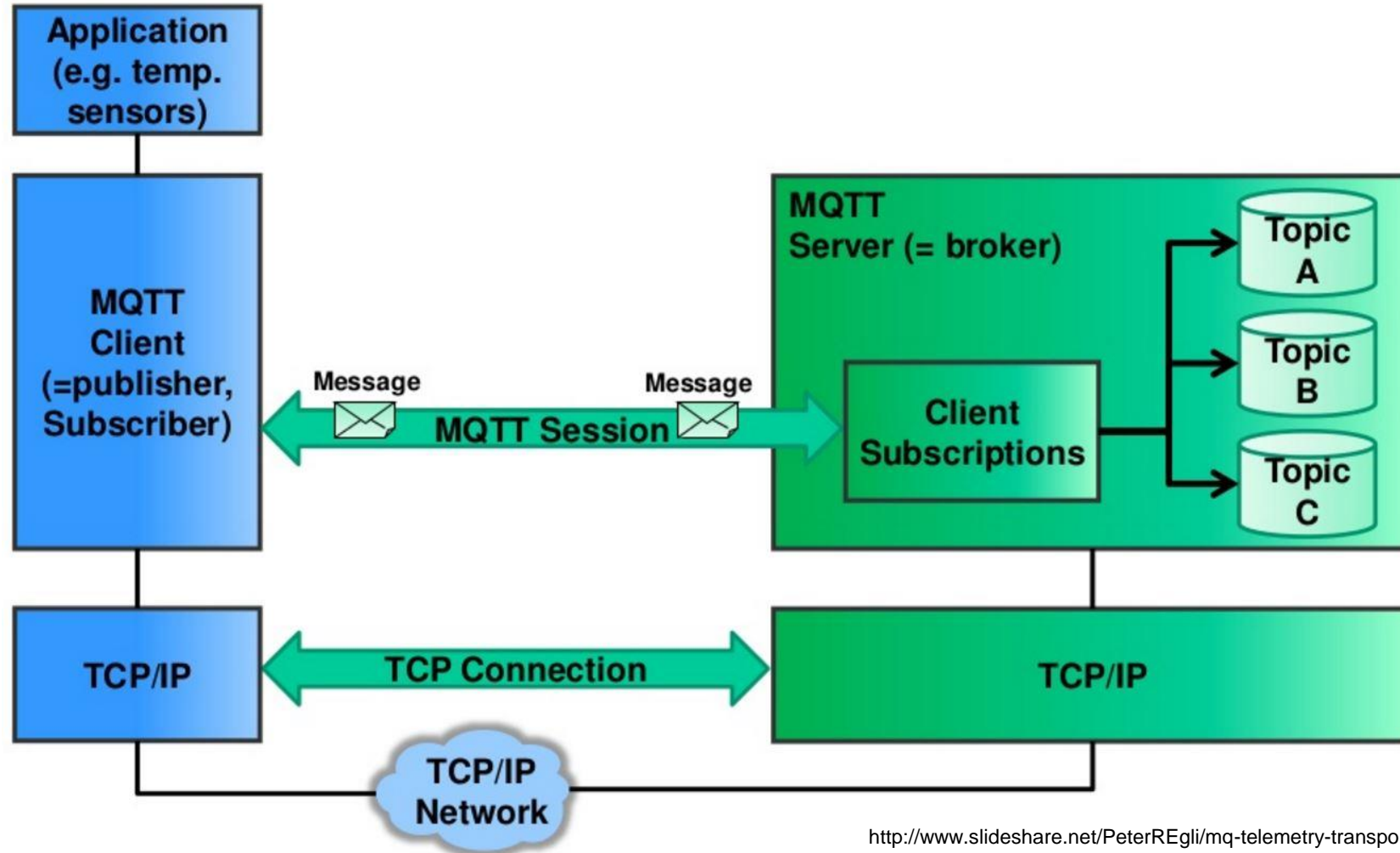
- 공개 표준
- 메시지 기반 비동기식 커뮤니케이션(Asynchronous communication) 모델
- 발행/구독(Publish/Subscribe) 모델
- 토픽(Topic)을 통한 데이터 생산자(data producer, publisher)와 데이터 소비자(data consumer, subscriber)의 분리
- 경량 프로토콜
 - 낮은 오버헤드 (2바이트 헤더), 낮은 구현 복잡도
- 메시지 전달의 신뢰성을 위해 QoS(Quality of Service) 옵션 제공
- TCP/IP 기반

브로커 기반의 MQTT 구조



- 예제 시나리오
 - 빌딩의 각 방의 온도를 모니터링하고 에어컨을 자동 제어하는 빌딩 관리 애플리케이션이 MQTT를 기반으로 동작한다고 가정해보자
- 애플리케이션(Application)
 - 브로커에게 구독 요청 - “방의 온도 데이터를 전달 받고 싶다”
 - 온도 데이터 분석
 - 에어컨 제어 여부를 결정하고 제어 메시지를 브로커에 전송
- 각 방의 에어컨 제어기(Actuator)
 - 브로커에게 구독 요청 - “에어컨 제어 메시지를 받겠다”
 - 애플리케이션이 생성한 제어 메시지에 따라 제어 동작을 수행
- 각 방의 온도 센서(Sensor)
 - 온도를 지속적으로 측정하여 측정된 온도 데이터를 브로커에 전송
- 브로커
 - 온도 데이터를 구독하는 빌딩 관리 애플리케이션에 온도 데이터 메시지를 전달
 - 애플리케이션이 생성한 제어 메시지를 구독 요청한 에어컨 제어기에 전달

MQTT 서버-클라이언트 모델



- MQTT 클라이언트
 - 클라이언트는 데이터를 만들어내는 데이터 발행자(publisher)와 생성된 데이터를 사용하는 데이터 구독자(subscriber)를 포함
- MQTT 서버
 - MQTT 브로커가 서버로 동작
 - 서버는 여러 가지 토픽을 관리하며 각 토픽 별로 발행자가 생성하여 전송한 데이터 메시지를 받고 이 토픽을 구독하는 구독자에게 메시지를 전달
 - 이를 위해 구독자에게 특정 토픽에 대한 구독 요청(subscription)을 받아 관리
- 메시지(Message)
 - 메시지는 MQTT 클라이언트들, 즉 특정 토픽의 데이터를 구독하는 구독자와 그 토픽에 대한 데이터를 발행하는 발행자 사이에 데이터를 교환하는 단위
 - MQTT는 메시지의 내부 구조에 상관없이 동작
- 세션(Session)
 - 클라이언트와 서버 사이에 통신을 위한 일시적인 연결을 나타냄
 - 서버와 클라이언트 사이에 메시지를 주고받는 것은 모두 한 세션의 일부로 이루어지게 됨

- 토픽(Topic)

- 데이터 발행자와 데이터 구독자 간에 데이터 교환이 이루어질 수 있도록 지정된 데이터의 주제
 - 예를 들어, 빌딩 관리 애플리케이션이라면 빌딩의 각 방의 온도, 미세먼지 농도, 조도 등이 각각 토픽이 될 수 있음
 - 온도 센서나 먼지 농도 센서는 자신이 생성하는 센서 데이터를 해당 토픽으로 발행을 하면 브로커에 전달되어 해당 토픽의 메시지 큐에서 관리가 되고, 브로커는 다시 이 토픽을 구독하는 애플리케이션에 해당 데이터를 전송하게 됨
 - 슬래시(/)를 구분자로 하여 계층적으로 표현
 - 예를 들어, 빌딩의 각 방에 있는 여러 온도 센서에서 온도 데이터를 생성한다고 가정 하면, 아래와 같은 구조로 토픽을 표현할 수 있음
 - /ROOM_NUMBER/SENSOR_ID/temperature
- 예: /125/sensor1/temperature

- 구독 요청(Subscription)

- 구독자(Subscriber)가 특정 토픽에 대한 메시지를 받고자 할 때 브로커에 등록

- 구독 요청 종류

- 일시적(transient) 요청: 세션이 생성된 기간 동안만 유지가 되기 때문에 그 중간에 발생한 데이터 메시지는 받을 수가 없음
 - 지속적(durable) 요청: 세션의 생성 여부와 상관없이 유지가 되고 그렇기 때문에 세션이 열리지 않은 기간에 발생한 메시지라고 하더라도 나중에 전달받을 수 있음
 - 이것은 연결(CONNECT) 메시지의 clean 세션 플래그에 따라서 결정

- 구독 요청 방식

- 구체적인 특정한 토픽을 지정하여 그 토픽에 대한 발행 메시지만 받는 방식
 - 여러 토픽을 동시에 구독할 수 있도록 와일드카드(wildcard)를 이용하여 그에 매칭이 되는 복수 개의 토픽에 대해서 구독 요청을 하는 방식
 - 와일드카드: +, #
 - +: 계층 구조의 한 단계만을 위해 사용
 - #: 계층 구조에서 # 이후의 모든 단계를 표현하기 위해 사용

- Publish되는 토픽이 “a/b/c/d”라고 가정하면

- 매칭되는 구독(Subscribe) 요청

- a/b/c/d
 - +/b/c/d
 - a+/c/d
 - a/+/+d
 - +/+/+
 - #
 - a/#
 - a/b/#
 - a/b/c/#
 - +/b/c/#

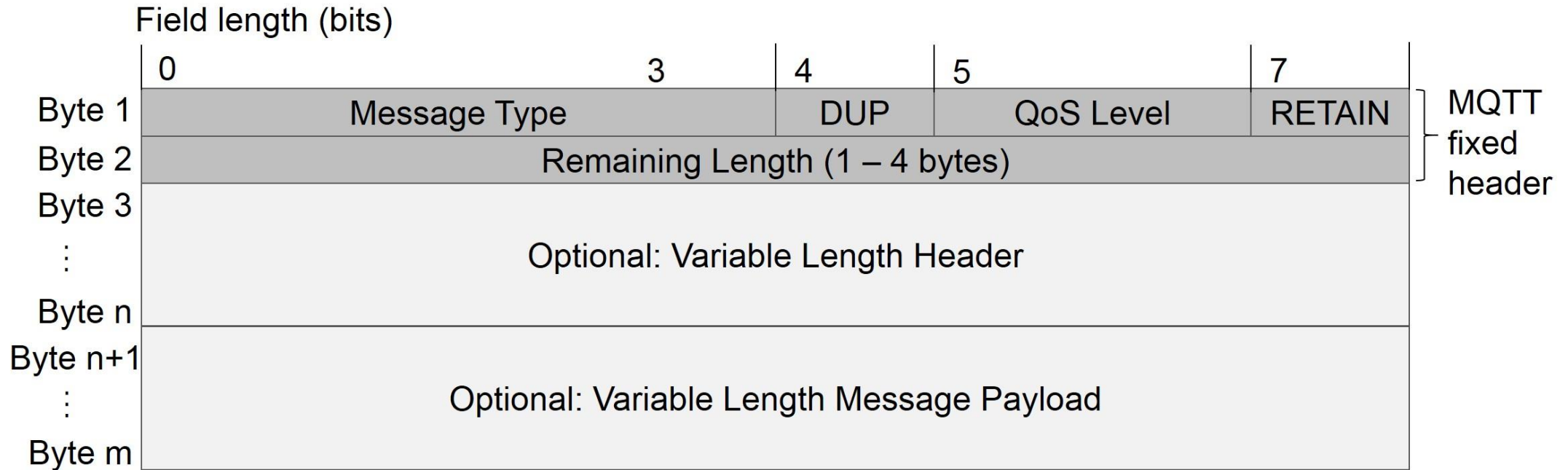
- 아래는 매칭이 안 되는 구독(Subscribe) 요청

- a/b/c
 - b+/c/d
 - +/+

+: 계층 구조의 한 단계만을 위해 사용

#: 계층 구조에서 # 이후의 모든 단계를 표현하기 위해 사용

MQTT 메시지 포맷



- 헤더 + 메시지 페이로드 (실제 전송할 데이터)
 - 헤더: 최소 2바이트 고정 헤더 + 가변 길이 헤더(옵션)
 - 페이로드: 가변 길이

MQTT 메시지 헤더

- Message Type

| | | | |
|-------------|--------------------------------|-----------------|-------------|
| 0: Reserved | | 8: SUBSCRIBE | 클라이언트 구독 요청 |
| 1: CONNECT | 클라이언트에서 서버로 연결 요청 | 9: SUBACK | 구독 요청 ACK |
| 2: CONNACK | 연결 요청 ACK | 10: UNSUBSCRIBE | 클라이언트 구독 해제 |
| 3: PUBLISH | 데이터 발행 | 11: UNSUBACK | 구독 해제 ACK |
| 4: PUBACK | 데이터 발행 ACK | 12: PINGREQ | Ping 요청 |
| 5: PUBREC | Publish received (QoS 레벨 2 관련) | 13: PINGRESP | Ping 응답 |
| 6: PUBREL | Publish release (QoS 레벨 2 관련) | 14: DISCONNECT | 클라이언트 연결 해제 |
| 7: PUBCOMP | Publish complete (QoS 레벨 2 관련) | 15: Reserved | |

- DUP

- 중복 메시지 플래그
- 메시지 수신자에게 이 메시지는 이미 수신한 메시지일 수도 있다는 것을 알려줌

- QoS Level

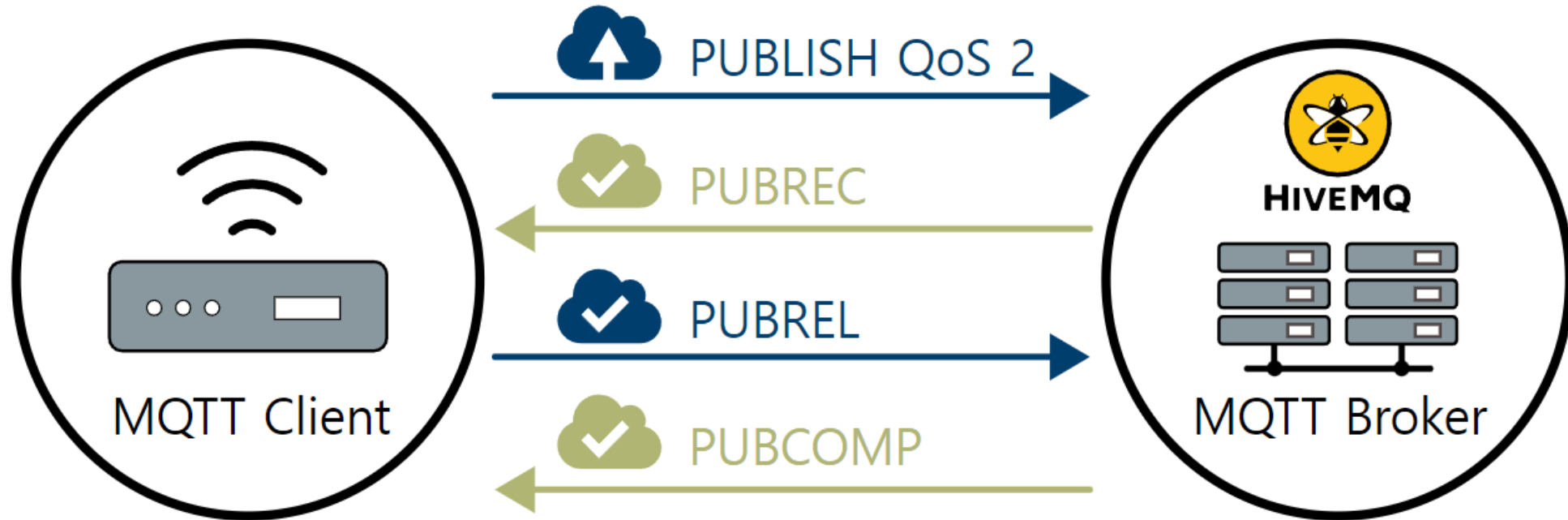
- PUBLISH 메시지의 메시지 전달을 보장하는 수준을 나타냄
- 0, 1, 2 세 가지 수준의 QoS를 제공
- 0(At-most-once delivery)
 - 메시지를 한번만 전달하며, 전달 성공 여부를 확인하지 않는다.
- 1(At-least-once delivery)
 - 메시지는 반드시 한번 이상 전달된다. 중복된 메시지를 받을 수 있다.
- 2(Exactly-once delivery)
 - 4단계 핸드셰이크 과정을 거쳐서 메시지가 정확히 한번 성공적으로 전달될 수 있도록 함
 - 높은 메시지 전송 품질을 보장하지만 부가적인 핸드셰이크 메시지를 주고받아야 하므로 오버헤드가 있음

QoS 1



Quality of Service level 1: delivery at least once

QoS 2



Quality of Service level 2: delivery exactly once

- RETAIN

- 이 필드의 값이 1이면, 서버는 마지막에 수신한 PUBLISH 메시지를 보관하고 있다가 새로운 구독 요청이 있을 때 이에 대한 첫 메시지로 전송하게 됨

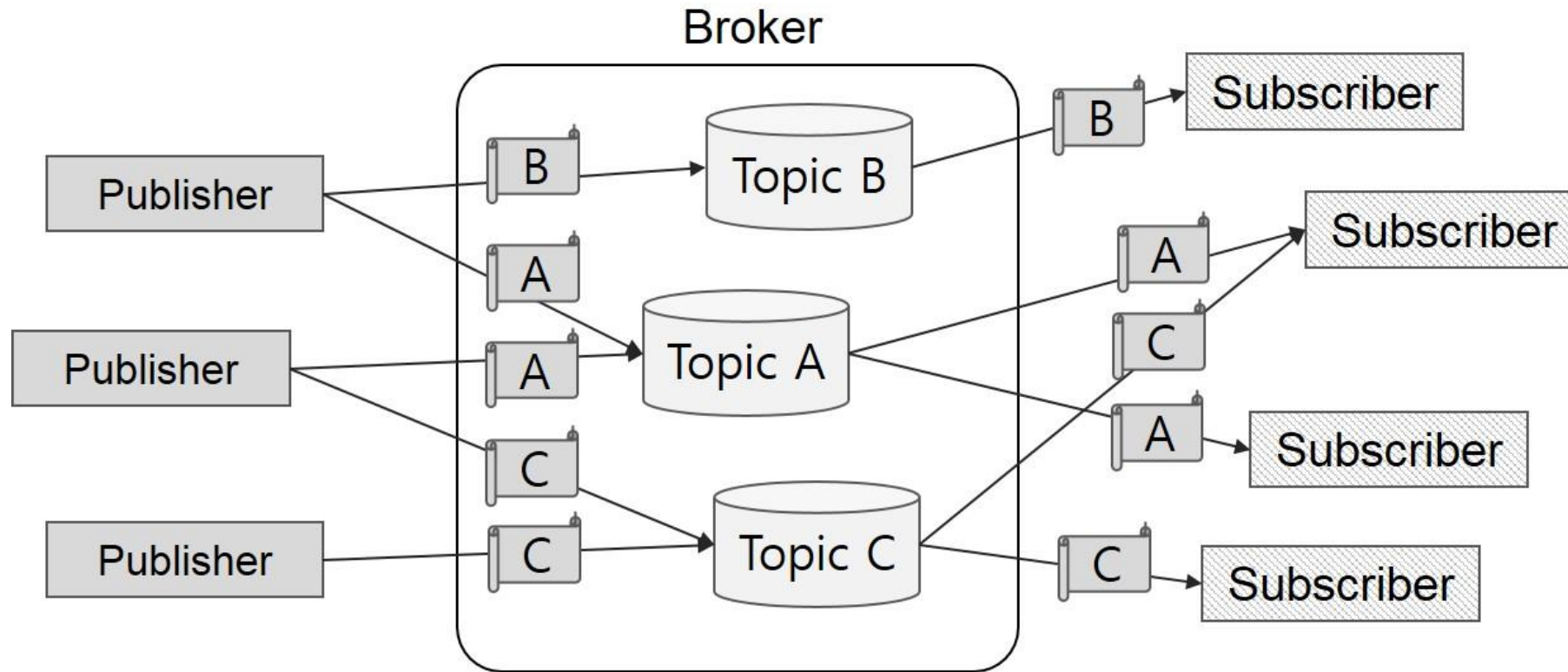
- Remaining Length

- 고정 헤더를 제외한 메시지의 나머지 길이를 바이트로 표현
- 즉, 선택적인 가변 길이 헤더와 페이로드의 길이
- 메시지의 길이에 따라서 1에서 4 바이트로 표현될 수 있음

3. 사물 인터넷 통신

3-2. MQTT 브로커/클라이언트를 이용한 MQTT 통신

MQTT 브로커 및 발행/구독 클라이언트



오픈 소스 MQTT 브로커 및 클라이언트 활용

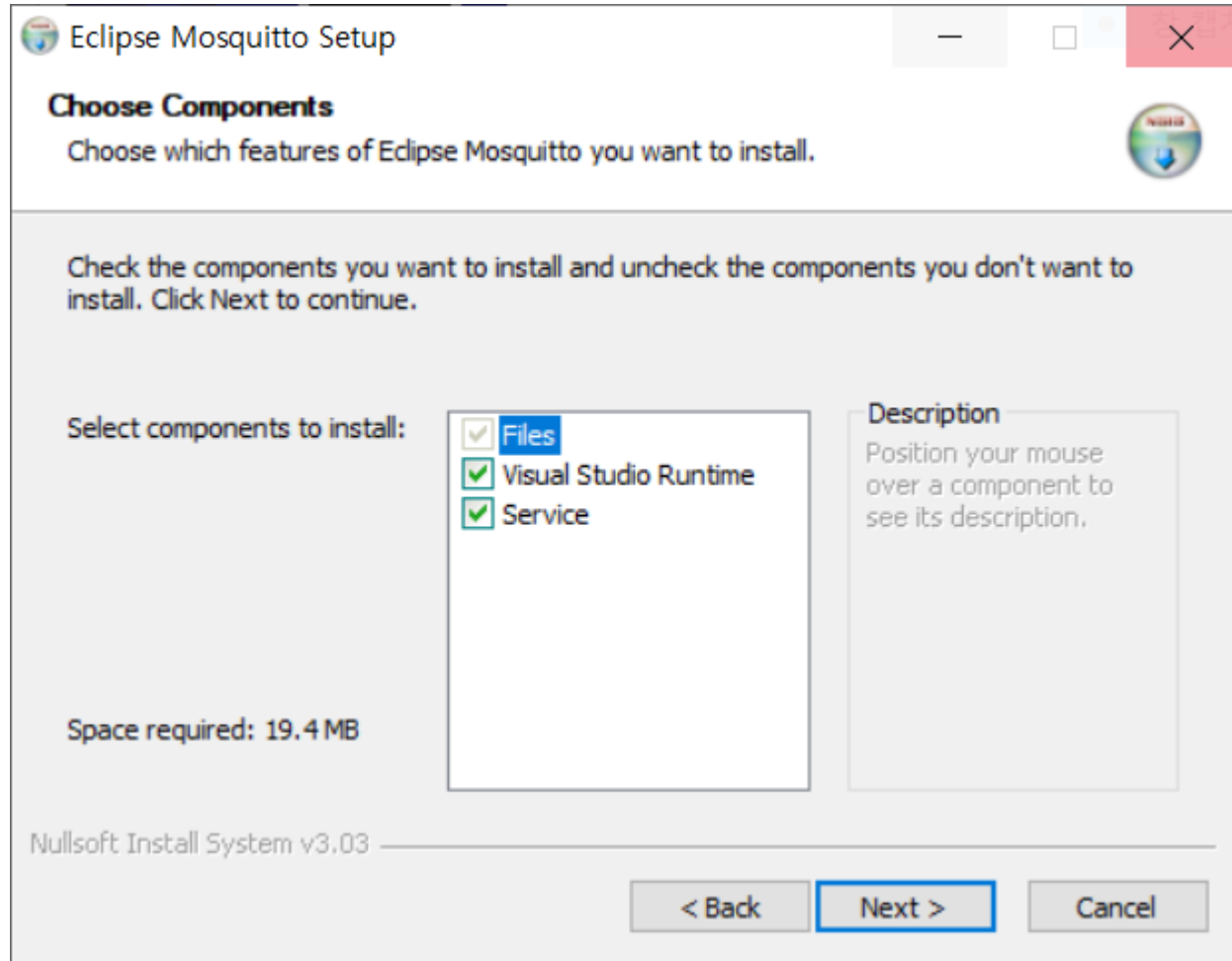
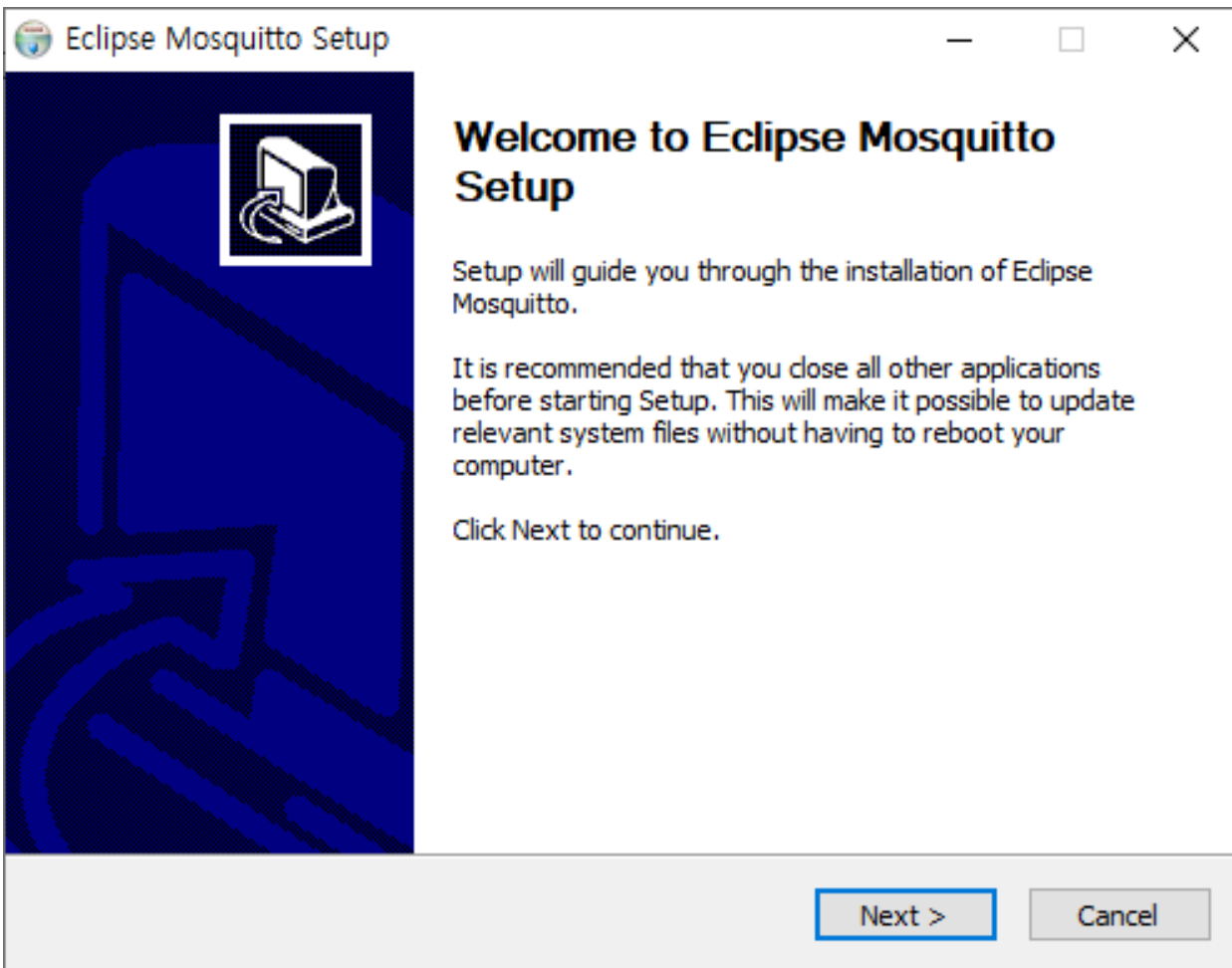
- MQTT 브로커
 - Mosquitto 브로커 설치 및 이용
 - Mosquitto 브로커 설치 시 같이 제공되는 Pub/sub 클라이언트 프로그램 이용 가능
- Paho MQTT 클라이언트
 - Paho 라이브러리를 기반으로 파이썬 클라이언트 프로그램 구현
 - Mosquitto 브로커를 이용하여 클라이언트 프로그램 실행 확인

Mosquitto MQTT 브로커

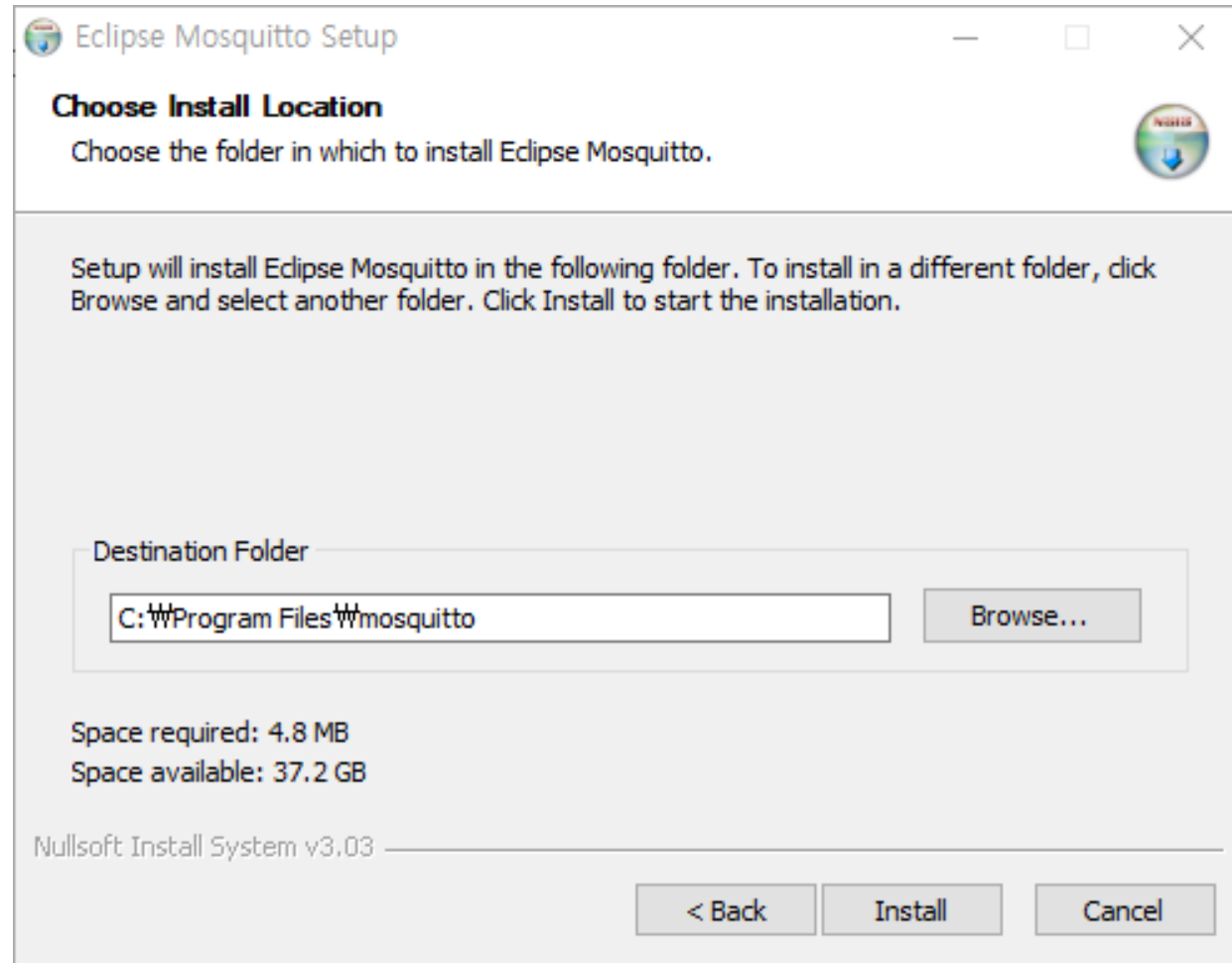
- Mosquitto
 - MQTT 프로토콜 버전 3.1, 3.1.1, 5.0을 구현한 오픈 소스 MQTT 브로커
 - MQTT 발행자와 구독자가 서로 메시지를 주고 받을 수 있도록 중간에서 중개하는 역할을 수행한다.
 - Mosquitto 홈페이지(<http://mosquitto.org/>)
 - 각 버전 별 정보, 다운로드, 문서 등 Mosquitto에 관한 정보 확인 가능
 - 지원 플랫폼
 - Windows, Mac, Debian, Ubuntu, Raspberry Pi 등 다양한 플랫폼을 위한 바이너리를 지원
 - 설치 파일 다운로드: <http://mosquitto.org/download/>

Mosquitto 설치 및 이용하기

- Windows PC
 - 윈도우용 설치 파일을 다운받아 설치
 - 여기서 제공되는 브로커, publish/subscribe 클라이언트 프로그램을 이용해서 데이터를 publish/subscribe 해 볼 수 있음
- Raspberry Pi
 - 윈도우 PC와 비슷하게 설치하고 이용할 수 있음



- 설치할 mosquitto 컴포넌트
- Service 선택하면 윈도우 실행 시 자동으로 실행되므로 선택 해제



- 설치할 폴더 위치 지정

- mosquitto 설치 폴더 내에 아래 실행 파일 확인
 - mosquitto.exe
 - MQTT 브로커 프로그램
 - mosquitto_sub.exe
 - subscribe 클라이언트 실행 프로그램
 - 여러 가지 파라미터를 주고 클라이언트 프로그램을 실행할 수 있음
 - mosquitto_pub.exe
 - publish 클라이언트 실행 프로그램
 - 여러 가지 파라미터를 주고 클라이언트 프로그램을 실행할 수 있음
 - Uninstall.exe
 - mosquitto 제거 파일

mosquitto 구독(subscribe) 클라이언트 실행

- 실행을 위한 명령어 입력 예

```
mosquitto_sub -h test.mosquitto.org -p 1883 -t # -v
```

- -h: 서버 호스트 네임 (-h 옵션을 주지 않을 경우 기본값은 localhost)
- -p: 포트 번호 (-p 옵션을 주지 않을 경우 기본값 1883)
- -t: 토픽
- -v: verbose 모드 출력 (로그 메시지를 자세하게 출력하라는 의미)

- mosquitto_sub man page

https://mosquitto.org/man/mosquitto_sub-1.html

mosquitto 발행(publish) 클라이언트 실행

- 실행을 위한 명령어 입력 예

`mosquitto_pub -h test.mosquitto.org -t temp/random -m 23.0`

- `-h`: 서버 호스트 네임 (`-h` 옵션을 주지 않을 경우 기본값은 localhost)
- `-p`: 포트 번호 (`-p` 옵션을 주지 않을 경우 기본값 1883)
- `-t`: 토픽
- `-m`: 발행 메시지

- 토픽과 메시지는 반드시 있어야 함

- mosquitto_pub man page

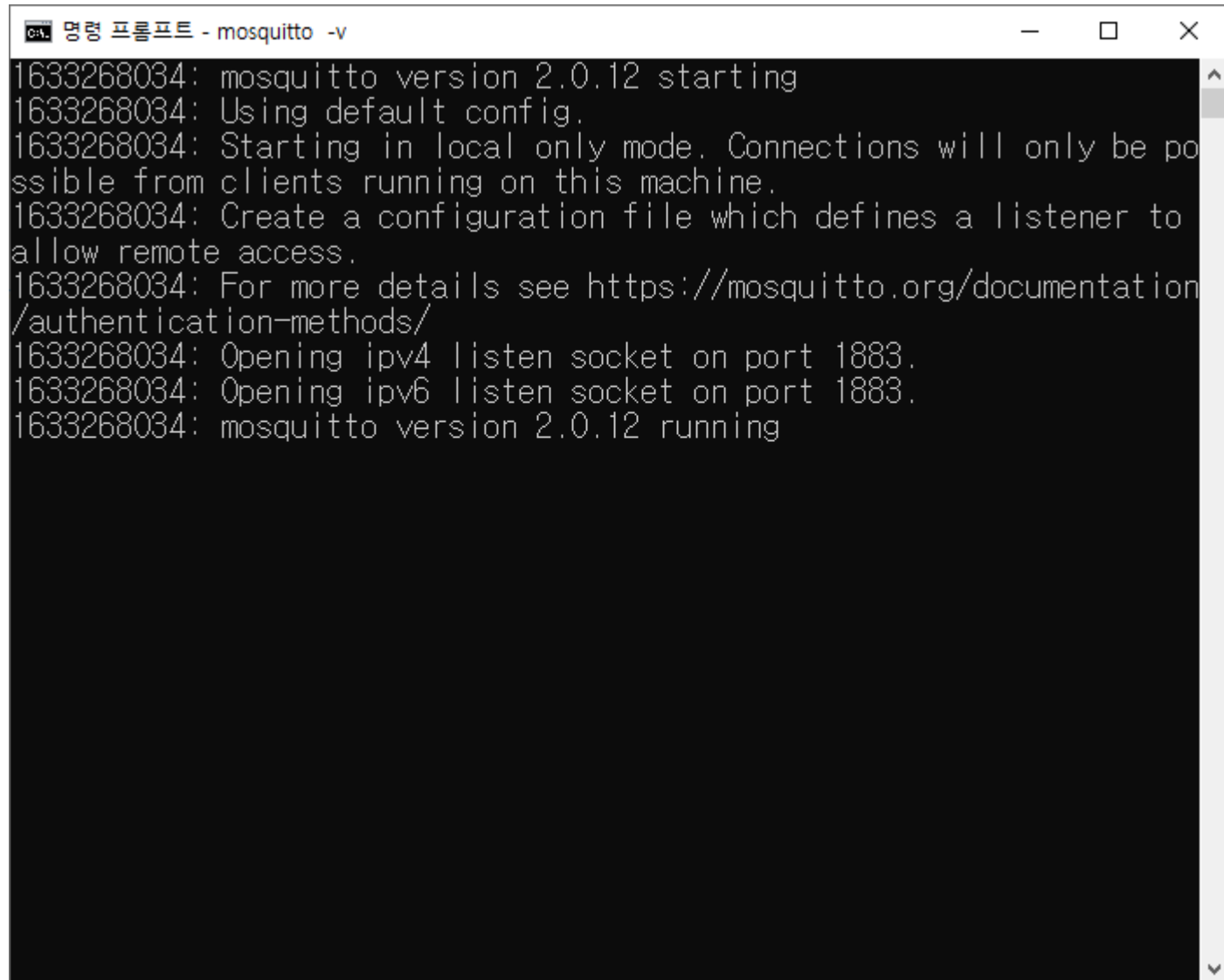
https://mosquitto.org/man/mosquitto_pub-1.html

Mosquitto 브로커 실행 및 테스트

- 로컬 Mosquitto 브로커 실행
 - Mosquitto 브로커는 mosquitto.exe 실행 파일을 통해 실행 가능
 - 윈도우즈 커맨드 창에서 mosquitto 설치 폴더로 이동 후 아래의 커맨드를 실행

mosquitto -v

- Mosquitto man page
<https://mosquitto.org/man/mosquitto-8.html>



```
명령 프롬프트 - mosquitto -v
1633268034: mosquitto version 2.0.12 starting
1633268034: Using default config.
1633268034: Starting in local only mode. Connections will only be possible from clients running on this machine.
1633268034: Create a configuration file which defines a listener to allow remote access.
1633268034: For more details see https://mosquitto.org/documentation/authentication-methods/
1633268034: Opening ipv4 listen socket on port 1883.
1633268034: Opening ipv6 listen socket on port 1883.
1633268034: mosquitto version 2.0.12 running
```

Mosquitto 브로커 실행 옵션

- Mosquitto 브로커 실행 시 사용할 수 있는 4가지 옵션
 - 모든 옵션은 선택적으로 지정하지 않을 경우 기본값으로 실행
 - 실행파일(mosquitto.exe) 실행 시 옵션 지정
 - -c, --config-file 설정 파일 경로
 - 지정된 설정 파일로부터 브로커 환경을 불러온다
 - 옵션이 설정되지 않은 경우 기본 값인 mosquitto 설치 폴더 내에 있는 "mosquitto.conf"을 사용
 - -d, --daemon
 - 지정한 경우 mosquitto 브로커를 백그라운드에서 데몬으로 실행
 - 브로커의 모든 동작은 똑같이 수행
 - -p, --port 포트 번호
 - 브로커 수신 포트 번호를 설정
 - 지정하지 않을 경우 기본 값은 1883
 - -v, --verbose
 - 최대한 자세한 브로커 로그 메시지를 출력
 - 이는 설정 파일에서 log_type을 all로 설정하는 것과 동일

- 구독 클라이언트 구독 요청
 - 새로운 윈도우즈 커맨드 창 실행
 - 여기서 mosquitto 설치 폴더로 이동
 - 아래 명령어 실행

`mosquitto_sub -t # -v`

- mosquitto -v 실행한 윈도우즈 창을 종료하면 mosquitto 브로커의 실행이 중단되니 주의

```
명령 프롬프트 - mosquitto_sub -t # -v
C:\Program Files\Mosquitto>mosquitto_sub -t # -v
```

- 구독 클라이언트 실행 창

```
명령 프롬프트 - mosquitto -v
1633268034: mosquitto version 2.0.12 starting
1633268034: Using default config
1633268034: St
ssible from cl
1633268034: Cr
allow remote a
1633268034: Fo
/auth
1633268034: Op
1633268034: Op
1633268034: mosquitto version 2.0.12 running
1633268260: New connection from ::1:62363 on port 1883.
1633268260: New client connected from ::1:62363 as auto-BE282718-E57
3-F91E-E015-CD82A2A65C63 (p2, c1, k60).
1633268260: No will message specified.
1633268260: Sending CONNACK to auto-BE282718-E573-F91E-E015-CD82A2A6
5C63 (0, 0)
1633268260: Received SUBSCRIBE from auto-BE282718-E573-F91E-E015-CD8
2A2A65C63
1633268260:      # (QoS 0)
1633268260: auto-BE282718-E573-F91E-E015-CD82A2A65C63 0 #
1633268260: Sending SUBACK to auto-BE282718-E573-F91E-E015-CD82A2A65
C63
```

- CONNECT 요청 메시지와 그에 따른 CONNACK 메시지
- SUBSCRIBE 요청 메시지와 그에 따른 SUBACK 메시지를 주고 받음을 알 수 있음

- mosquitto 브로커 실행 창
- 구독 클라이언트를 실행하면 새로운 클라이언트로 부터 구독 요청이 들어왔다는 내용을 출력하는 것 을 볼 수 있음

- 발행 클라이언트 메시지 발행

- 새로운 윈도우즈 커맨드 창 실행
- 여기서 mosquitto 설치 폴더로 이동
- 아래 명령어 실행

`mosquitto_pub -t test -m "hello world"`

- 브로커 및 구독 클라이언트를 실행한 윈도우즈 창을 종료하지 않도록 주의

```
Command Prompt
c:\Program Files (x86)\mosquitto>mosquitto_pub -t test -m "hello world"
c:\Program Files (x86)\mosquitto>
```

- 발행 클라이언트 실행 창

```
명령 프롬프트 - mosquitto_sub -t # -v
C:\Program Files\Mosquitto>mosquitto_sub -t # -v
test hello world
```

- 구독 클라이언트 실행 창
- 발행 클라이언트를 실행하면 구독 클라이언트에서 수신된 메시지를 출력하는 것을 볼 수 있음

mosquitto 브로커 실행창에서는 어떤 내용이 출력되는지 확인해보자

- 발행 클라이언트 혹은 구독 클라이언트의 QoS 설정
 - `-q` 옵션 이용
 - 0, 1, 2 값 중 하나 (기본값은 0, `-q` 옵션을 주지 않았을 경우 0으로 동작)
- 예
 - `mosquitto_sub -t # -v -q 1`
 - `mosquitto_pub -t test -m 33 -q 2`

QoS 값을 1 혹은 2로 하였을 때 브로커와 클라이언트 사이에 주고 받는 메시지가 어떻게 달라지는지 확인해보자

Raspberry Pi에 mosquitto 설치 및 실행

- 설치를 위해 터미널에서 아래 명령어 실행

pi@raspberrypi:~ \$ **sudo apt-get install mosquitto**

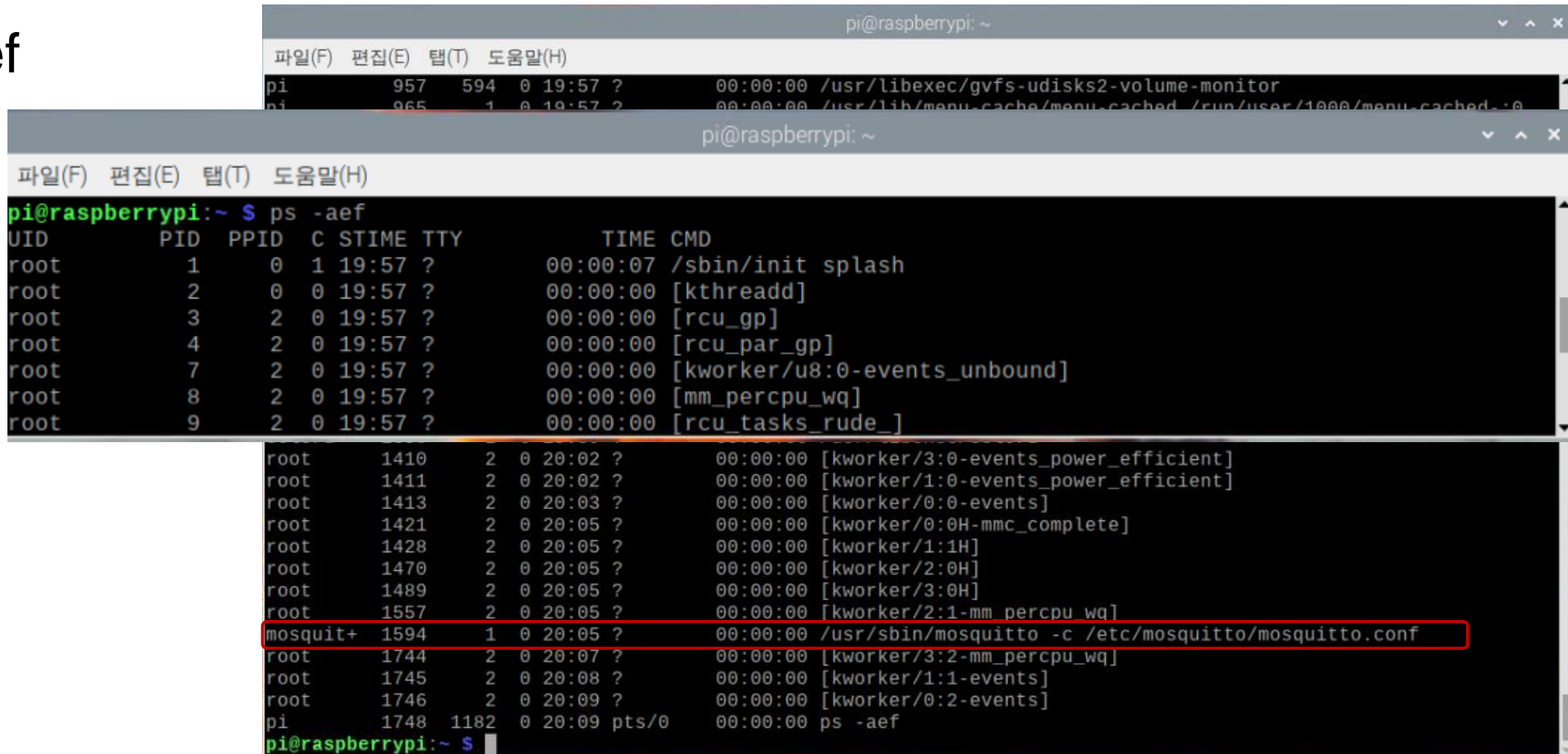
```
pi@raspberrypi: ~
파일(F) 편집(E) 탭(T) 도움말(H)
pi@raspberrypi:~ $ sudo apt-get install mosquitto
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다... 완료
상태 정보를 읽는 중입니다... 완료
다음 패키지가 자동으로 설치되었지만 더 이상 필요하지 않습니다:
  libfuse2
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  libcjson1 libdlt2 libev4 libmosquitto1 libwebsockets16
제안하는 패키지:
  apparmor
다음 새 패키지를 설치할 것입니다:
  libcjson1 libdlt2 libev4 libmosquitto1 libwebsockets16 mosquitto
0개 업그레이드, 6개 새로 설치, 0개 제거 및 117개 업그레이드 안 함.
591 k바이트 아카이브를 받아야 합니다.
이 작업 후 1,464 k바이트의 디스크 공간을 더 사용하게 됩니다.
계속 하시겠습니까? [Y/n] y
받기:1 http://ftp.kaist.ac.kr/raspbian/raspbian bullseye/main armhf libcjson1 ar
mf 1.7.14-1 [20.8 kB]
받기:2 http://ftp.kaist.ac.kr/raspbian/raspbian bullseye/main armhf libdlt2 armh
f 2.18.6-1 [45.3 kB]
```

❖ 설치가 완료되면 자동 실행 됨

Raspberry Pi에 mosquitto 설치 및 실행

- 현재 실행 중인 프로세스 확인 방법

ps -aef

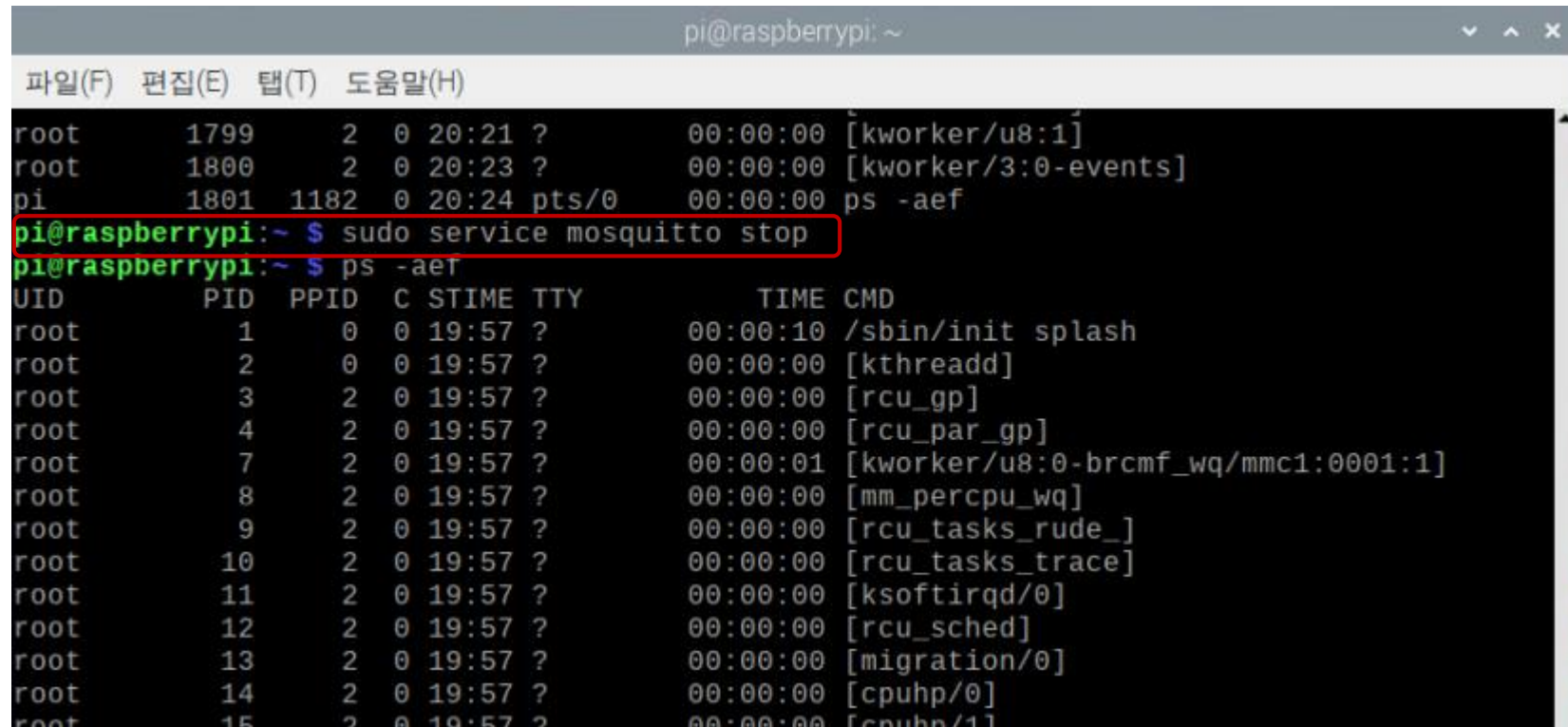


```
pi@raspberrypi: ~  
파일(F) 편집(E) 탭(T) 도움말(H)  
pi 957 594 0 19:57 ? 00:00:00 /usr/libexec/gvfs-udisks2-volume-monitor  
pi 965 1 0 19:57 ? 00:00:00 /usr/lib/menu-cache/menu-cached /run/user/1000/menu-cached-:0  
pi@raspberrypi: ~  
파일(F) 편집(E) 탭(T) 도움말(H)  
pi@raspberrypi:~$ ps -aef  
UID      PID  PPID  C  STIME TTY          TIME CMD  
root      1      0  1  19:57 ?        00:00:07 /sbin/init splash  
root      2      0  0  19:57 ?        00:00:00 [kthreadd]  
root      3      2  0  19:57 ?        00:00:00 [rcu_gp]  
root      4      2  0  19:57 ?        00:00:00 [rcu_par_gp]  
root      7      2  0  19:57 ?        00:00:00 [kworker/u8:0-events_unbound]  
root      8      2  0  19:57 ?        00:00:00 [mm_percpu_wq]  
root      9      2  0  19:57 ?        00:00:00 [rcu_tasks_rude_]  
root     1410      2  0  20:02 ?        00:00:00 [kworker/3:0-events_power_efficient]  
root     1411      2  0  20:02 ?        00:00:00 [kworker/1:0-events_power_efficient]  
root     1413      2  0  20:03 ?        00:00:00 [kworker/0:0-events]  
root     1421      2  0  20:05 ?        00:00:00 [kworker/0:0H-mmc_complete]  
root     1428      2  0  20:05 ?        00:00:00 [kworker/1:1H]  
root     1470      2  0  20:05 ?        00:00:00 [kworker/2:0H]  
root     1489      2  0  20:05 ?        00:00:00 [kworker/3:0H]  
root     1557      2  0  20:05 ?        00:00:00 [kworker/2:1-mm_percpu_wq]  
mosquit+ 1594      1  0  20:05 ?        00:00:00 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf  
root     1744      2  0  20:07 ?        00:00:00 [kworker/3:2-mm_percpu_wq]  
root     1745      2  0  20:08 ?        00:00:00 [kworker/1:1-events]  
root     1746      2  0  20:09 ?        00:00:00 [kworker/0:2-events]  
pi       1748    1182  0  20:09 pts/0    00:00:00 ps -aef  
pi@raspberrypi:~$
```

Raspberry Pi에 mosquitto 설치 및 실행

- 실행 중인 mosquitto 서비스 종료

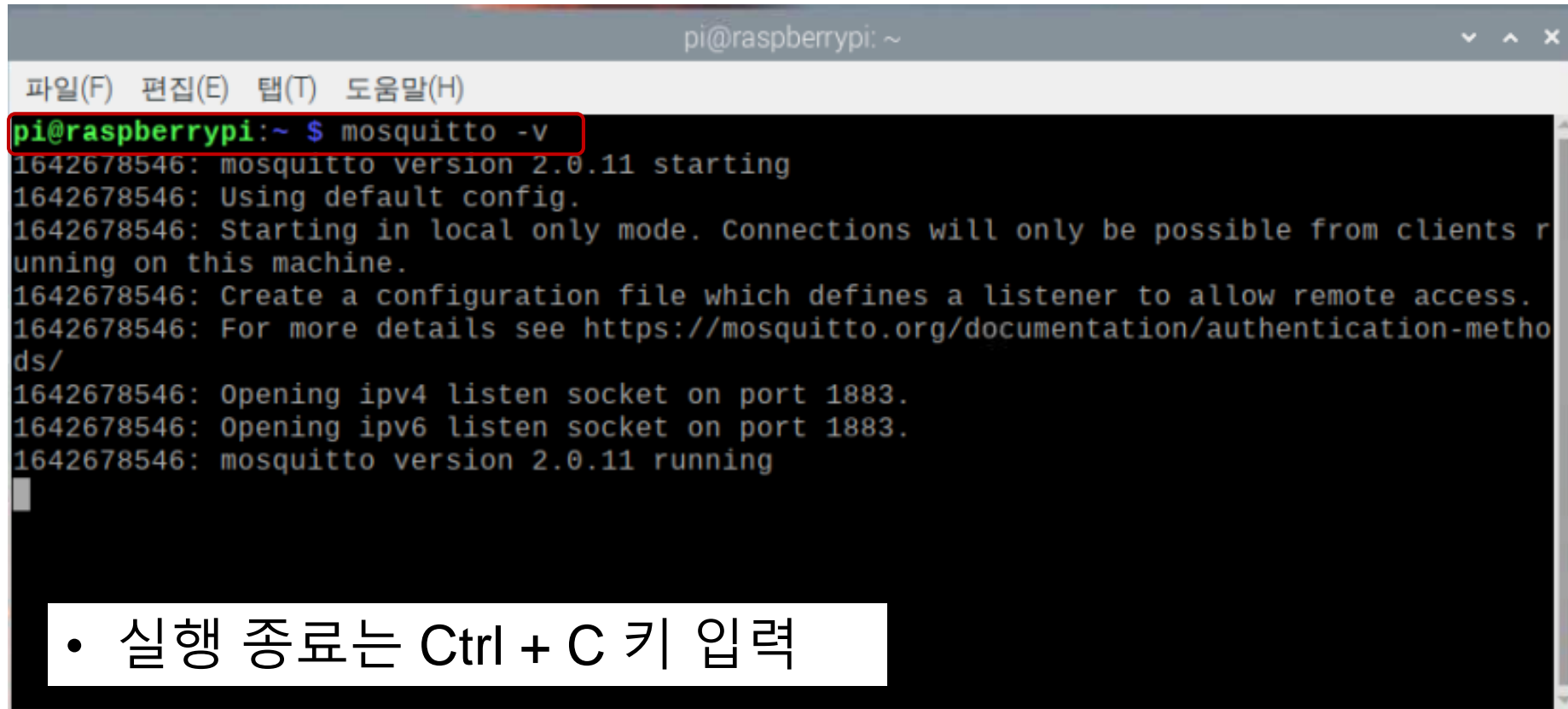
`sudo service mosquitto stop`



```
pi@raspberrypi: ~  
파일(F) 편집(E) 탭(T) 도움말(H)  
root      1799      2  0 20:21 ?        00:00:00 [kworker/u8:1]  
root      1800      2  0 20:23 ?        00:00:00 [kworker/3:0-events]  
pi        1801    1182  0 20:24 pts/0    00:00:00 ps -aef  
pi@raspberrypi:~ $ sudo service mosquitto stop  
pi@raspberrypi:~ $ ps -aef  
UID          PID    PPID  C STIME TTY          TIME CMD  
root           1        0  0 19:57 ?        00:00:10 /sbin/init splash  
root           2        0  0 19:57 ?        00:00:00 [kthreadd]  
root           3        2  0 19:57 ?        00:00:00 [rcu_gp]  
root           4        2  0 19:57 ?        00:00:00 [rcu_par_gp]  
root           7        2  0 19:57 ?        00:00:01 [kworker/u8:0-brcmf_wq/mmc1:0001:1]  
root           8        2  0 19:57 ?        00:00:00 [mm_percpu_wq]  
root           9        2  0 19:57 ?        00:00:00 [rcu_tasks_rude_]  
root          10        2  0 19:57 ?        00:00:00 [rcu_tasks_trace]  
root          11        2  0 19:57 ?        00:00:00 [ksoftirqd/0]  
root          12        2  0 19:57 ?        00:00:00 [rcu_sched]  
root          13        2  0 19:57 ?        00:00:00 [migration/0]  
root          14        2  0 19:57 ?        00:00:00 [cpuhp/0]  
root          15        2  0 19:57 ?        00:00:00 [cpuhp/1]
```

Raspberry Pi에 mosquitto 설치 및 실행

- Mosquitto 수동으로 실행하기

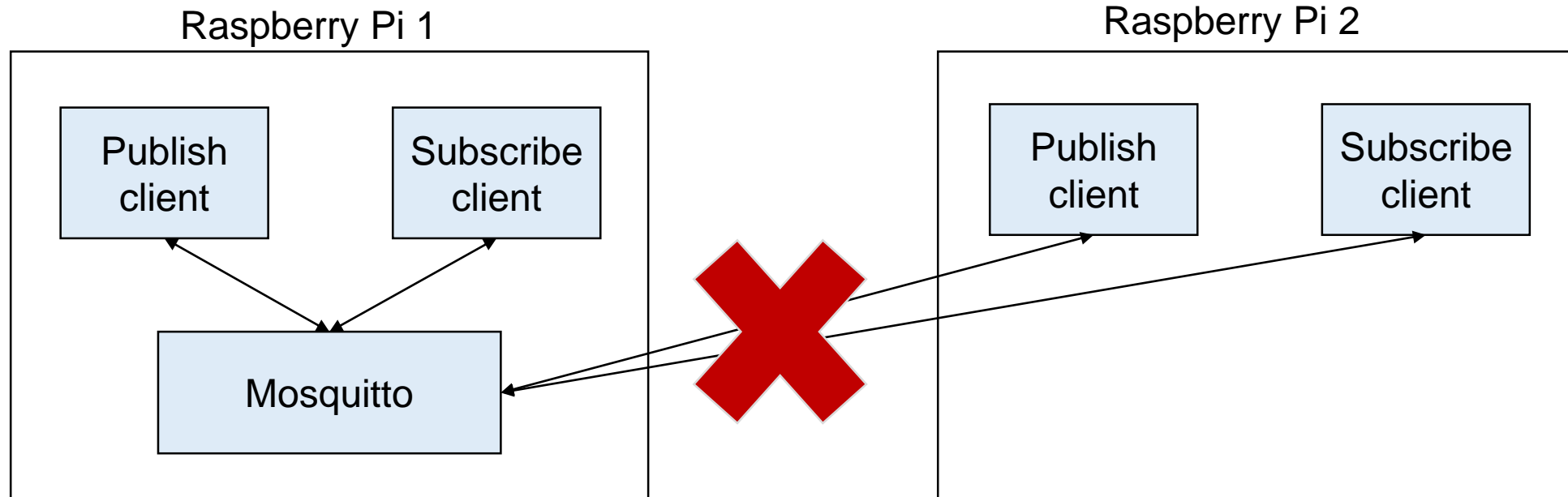


```
pi@raspberrypi: ~  
파일(F) 편집(E) 탭(T) 도움말(H)  
pi@raspberrypi:~ $ mosquitto -v  
1642678546: mosquitto version 2.0.11 starting  
1642678546: Using default config.  
1642678546: Starting in local only mode. Connections will only be possible from clients running on this machine.  
1642678546: Create a configuration file which defines a listener to allow remote access.  
1642678546: For more details see https://mosquitto.org/documentation/authentication-methods/  
1642678546: Opening ipv4 listen socket on port 1883.  
1642678546: Opening ipv6 listen socket on port 1883.  
1642678546: mosquitto version 2.0.11 running
```

- 실행 종료는 Ctrl + C 키 입력

외부 기기에서 접속 가능하도록 실행하기

- 현재 버전 (2.0.x) mosquitto 브로커를 설치하면 기본으로 local에서만 브로커에 접속이 가능하도록 실행됨
 - 예를 들어 Raspberry Pi에 mosquitto를 설치하면 이 Raspberry Pi에서 실행되는 클라이언트만 해당 mosquitto에 접속할 수 있게 됨



외부 기기에서 접속 가능하도록 실행하기

- mosquitto가 실행되는 Raspberry Pi나 컴퓨터가 아닌 다른 기기에서 실행되는 MQTT 클라이언트를 해당 mosquitto로 접속하도록 하는 경우에는 별도의 설정 필요
- 새로운 설정 파일 생성
 - 기본적으로 mosquitto를 설치하면 mosquitto.conf 라는 설정 파일 사용
 - 예를 들어 test.conf라는 파일을 만들어서 아래 내용을 작성하고 저장

listener 1883 0.0.0.0

allow_anonymous true

- 새 설정 파일을 이용해 mosquitto 실행

mosquitto -c test.conf -v

Paho MQTT Client

- Paho 프로젝트
 - 이클립스 재단의 오픈 소스 MQTT와 MQTT-SN 프로토콜 클라이언트를 개발하고 제공
 - C/C++, Java, Javascript, Python, Android 등을 포함하여 다양한 언어로 구현된 클라이언트 이용 가능
 - <http://www.eclipse.org/paho/>

Paho MQTT Python Client

- Paho Python client library
 - Python 2.7.9+ or 3.5+ 상에서 MQTT v3.1, v3.1.1, v5.0을 지원하는 클라이언트 라이브러리
 - 여기서 제공하는 여러 함수를 이용하여 클라이언트 프로그램 구현 가능
- 다운로드 및 설치
 - <https://github.com/eclipse/paho.mqtt.python>
 - pip tool을 이용한 설치
`sudo pip3 install paho-mqtt`

Paho MQTT Python Client

- 설치 명령어 실행 화면

```
pi@raspberrypi: ~  
파일(F) 편집(E) 탭(T) 도움말(H)  
1642678546: Starting in local only mode. Connections will only be possible from clients r  
unning on this machine.  
1642678546: Create a configuration file which defines a listener to allow remote access.  
1642678546: For more details see https://mosquitto.org/documentation/authentication-methods/  
1642678546: Opening ipv4 listen socket on port 1883.  
1642678546: Opening ipv6 listen socket on port 1883.  
1642678546: mosquitto version 2.0.11 running  
^C1642678651: mosquitto version 2.0.11 terminating  
pi@raspberrypi:~ $ sudo pip3 install paho-mqtt  
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple  
Collecting paho-mqtt  
  Downloading https://www.piwheels.org/simple/paho-mqtt/paho\_mqtt-1.6.1-py3-none-any.whl  
    (75 kB)  
    |████████████████████| 75 kB 119 kB/s  
Installing collected packages: paho-mqtt  
Successfully installed paho-mqtt-1.6.1  
pi@raspberrypi:~ $
```

Paho MQTT Python Client

- 정상적으로 설치된 것을 확인하는 화면

```
lee@raspberrypi:~ $ python
Python 3.9.2 (default, Mar 12 2021, 04:06:34)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import paho.mqtt.client as mqtt
>>> mqttc = mqtt.Client()
>>> 
```