Bluetooth Low Energy (BLE) 개념 정리

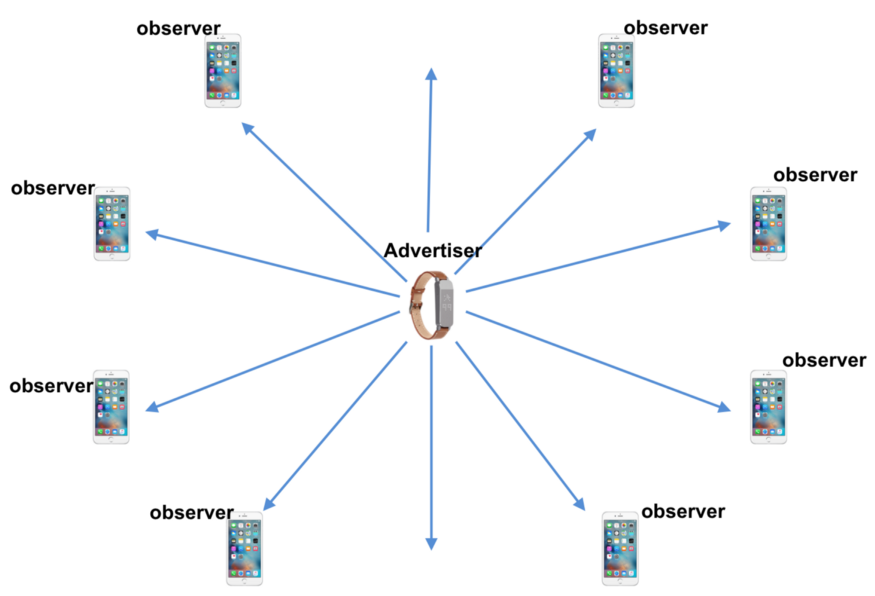
### BLE란?

블루투스라고 하면 Bluetooth Classic부터 Bluetooth Low Energy (BLE) 까지를 모두 포함한다.

그중에 BLE 는 **Bluetooth Low Energy**의 줄임말로 저전력 형태의 블루투스 통신이라고 보면 된다.

#### 연결방법 ( Advertising Mode, Connection Mode )

**1. [Advertising Mode]**



**1) 동작방식**

특정 디바이스를 지정하지 않고

**주변의 모든 디바이스에게 Signal을 보낸다.**

주변에 디바이스가 있건 없건 다른 디바이스가 Signal을 듣는 상태이건

아니건 일방적으로 Signal을 보낸다.

=> 비콘 마냥 주기적으로 계속 신호를 보낸다.

**2) 특징**

**- 한 번에 한 개 이상의 디바이스에 데이터를 전송할 수 있다.  ( 다수의 디바이스에 전송가능 )**

**- 보안에 취약하다.**

**- 보낼 수 있는 데이터의 크기가 한정적이다.**

\* 보내야 하는 데이터의 크기가 작고, 다수의 디바이스에 전송을 해야 하는 경우에 Advertising Mode를 활용

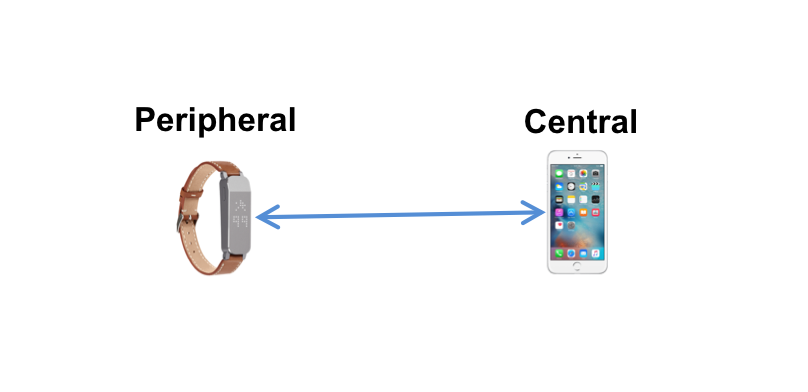
**3) 역할**

Advertiser / Observer

Advertiser : Signal을 주기적으로 보내는 디바이스

Observer : Signal을 주기적으로 **Scan** 하는 디바이스 (Signal에 귀 기울이고 있는 디바이스)

**2. [Connection Mode]**



**1) 동작방식**

보통 "블루투스 연결"하면 생각하는 바로 그  형태로,

디바이스가 **일대일로 연결을 하여서** 디바이스 간에 Channel Hopping 규칙을 정해 놓고 데이터를 주고받는 형태이다.

**양방향**으로 데이터를 주고받아야 하거나, **많은 양의 데이터**를 주고받는 경우에 Connecting Mode를 사용한다.

**2) 역할**

**Central (Master): (ex 스마트폰 )**

다른 디바이스와 Connection을 맺기 위해, Connectable Advertising Signal을 **주기적으로 Scan** 하다가,

적절한 디바이스에 연결 요청을 한다.

연결이 되고 나면 Central 디바이스는 \*timing을 설정하고 주기적인 데이터 교환을 주도한다.

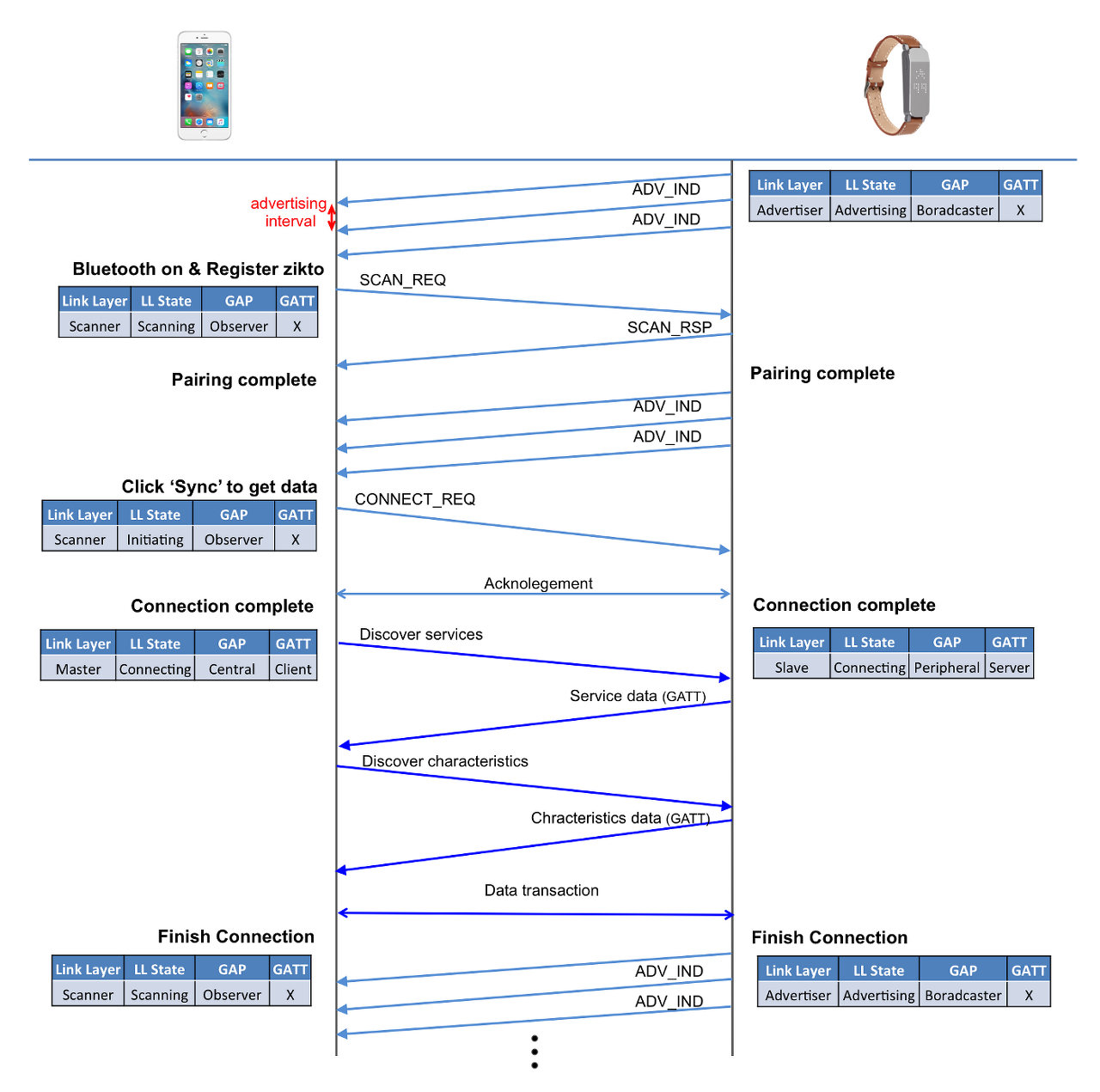
*\*timing :  두 디바이스가 매번 같은 Channel에서 데이터를 주고받기 위해 정하는 hopping 규칙*

**PeriPheral (Slave)** :   : **(ex 주변기기 - 스마트워치, 이어폰 등.)**

다른 디바이스와 Connection을 맺기 위해, Connectable Advertising Signal을 주기적으로 보낸다.

이를 수신한 Central디바이스가 Connection Request를 보내면, 이를 수락하여 Connection을 맺는다.

**3) 연결과정**



1. 디바이스가 Advertising Channeal을 Hopping 하며 Advertising Packet을 보낸다.

2. 핸드폰은 Advertising Channel을 Hopping 하며 Advertising Packet을 Scan 한다.

     디바이스를 찾은 경우에 그 디바이스에 대한 추가 정보를 얻기 위해 Scan Request를 보낸다.

3. Scan Request을 받은 디바이스가 Scan Response를 보낸다.

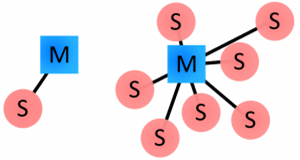
4. 핸드폰은 그 Scan Response를 파싱 하고 그 데이터를 봤을 때 연결을 하고자 하면 Connect Request를 보낸다.

5. 서로 Acknowledging을 시작하여 서 timing 정보 등을 동기화한다.

6. Connect 완료

7. 데이터 교환 시작

**4) 다중연결**



**Master는 여러 대의 Slave와 연결이 가능하다.**

\*피코넷에 연결되는 **슬레이브 장치는 오직 하나의 마스터 장치에만 연결이 가능**하고

마스터 장치는 피코넷을 통해 통신을 조율한다.

마스터 장치는 연결된 슬레이브 장치에 데이터를 전송할 수 있고 요청을 보낼 수 도 있다.

슬레이브 장치는 마스터와 데이터를 교환할 수 있지만 슬레이브 간에 통신은 할 수 없다.

이전의 블루투스에서는 하나의 피코넷에 7개의 슬레이브의 연결이 가능했지만

BLE에서는 마스터 장치와 연결될 수 있는 장치의 수가 제한이 없다고 하는 데 이 부분은 불확실해서 더 알아봐야 함!

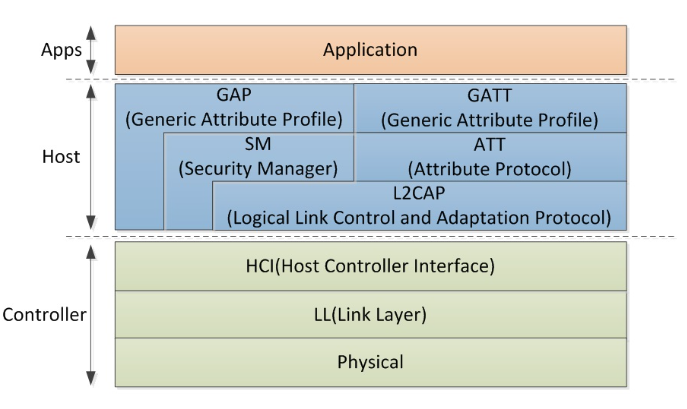
**\* 마스터이면서  다른 피코넷의 슬레이브가  될 수 있다.**

**( 하지만 동시에 2개 이상의 피코넷의 마스터가 될 순 없다. )**

*\*피코넷 : 블루투스 네트워크를 피코넷 또는 작은 망이라고 부른다. 블루투스 네트워크의 최소 단위.*

*마스터는 피코넷의 데이터량을 조절하고 피코넷에 접근하는 슬레이브를 통제하여 데이터 충돌을 막는다.*

### **BLE 통신 프로토콜 ( 프로토콜 스텍 )**



#### 1. Stack

Application Layer를 제외하면 총 8개의 Stack이 존재함을 알 수 있다.

- Physical: 2.4 GHz ISM 대역에서 1 Mbps의 속도로 패킷 송수신 역할 (실제 Bluetooth Analog Signal과 통신할 수 있는 회로가 구성되어 있음)

- LL (Link Layer): 5가지의 RF 상태 제어 (standby, advertising, scanning, initiating, connected) 및 디바이스의 Role 정의

- HCI (Host Controller Interface): Host 영역과 Controller 영역의 Interface 역할

- L2CAP (Logical Link Control and Adaptation Protocol): 데이터 encapulation service 제공

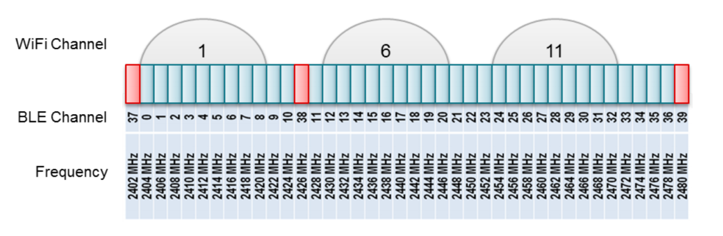
- SM (Security Manager): paring and key distributuiion 방법 정의 및 인증과 보안에 사용

- ATT (Attribute Protocol): 다른 기기로 '**attribute**'라는 **데이터 노출 및 데이터 교환**을 위한 **클라이언트**, **서버 프로토콜 정의**

- GAP (Generic Access Protocl): 장치 간의**paring과 bonding** 사용을 통해 장치 간 인터페이스 역할

- GATT (Generic Attribute Profile): ATT를 이용하는 sub-procedure를 정의하는 프레임워크, ATT 영역에서 읽어 들인 서비스의 기능 수행

#### 2. Physical Layer



- 2.4 GHz 영역에서 ISM Band 사용하며, 2 MHz 간격으로 총 40개의 채널을 사용

- 3개의 (37, 38, 39) Advertising 채널과 나머지 37개의 Data 채널을 사용

- 같은 ISM Band를 사용하는 wifi와의 간섭을 피하기 위해 주파수 호핑 방식 사용

#### 3. Link Layer

하드웨어와 소프트웨어 조합으로 구성되어 있으며,

하드웨어 단에서는 Preamble, Access Address, CRC, AES 등이 처리되고

소프트웨어 단에서는 디바이스의 연결 상태를 관리한다.

가장 중요한 건 **디바이스의 Role을 정의**하고 이에 따라 변경되는 State를 가지고 있다

Connection 전의 역할 (advertiser, scanner)과

Connection 후의 역할인 (Master, Slave)로 분류된다

**Role**

**Advertiser** : advertising packet 을 전송하는 장치  Advertising Packet을 보내는 역할.

**Scanner** : advertising packet 을 수신하는 장치  Advertising Packet을 Scanning 하는 역할.

Scanner는 아래와 같은 2가지 Scanning 모드가 있다.

**Master** : 연결을 시도하고, 연결 후에 전체 connection을 관리하는 역할. (연결을 시작하고 관리하는 장치)

**Slave** : Master의 연결 요청을 받고, Master가 보내준 timing(호핑 규칙)을 따르는 장치

Advertiser 와 Scanner 는 두 장치가 연결되기 전에 서로 담당해야 할 역할을 말합니다.

Master 와 Slave 는 연결이 되고 난 이후 각자가 맡을 역할입니다.

### **Advertising and Scanning**

BLE 는 송출하는 packet의 형식(format)이 하나로 정해져 있습니다.

그리고 advertising, data 두 종류의 패킷을 사용합니다.

먼저 advertising packet 과 이를 송출, 수신하는 Advertiser, Scanner 에 대해 다뤄보겠습니다.

Advertising packet 은 BLE 장치들은**서로를 인식하**는 과정에 사용됩니다.

* Advertiser 장치가 목적지 없이 주변에 advertising packet을 송출하면, Scanner 장치는 이걸 수신해서 주변에 있는 advertiser 장치들을 인지합니다.
* Advertising packet 은 header data(장치 정보)에 31 byte data payload (사용자 지정 데이터)를 합친겁니다. 헤더 데이터에는 6 byte MAC address (Bluetooth address) 도 포함하고 있습니다. **6 Byte MAC address** (Media Access Control)는 블루투스 장치를 구분하는 고유한 주소값입니다.
* Packet 을 송출하는 간격(advertising interval)은 20ms ~ 10.24s 사이에서 조절할 수 있습니다. 간격이 짧을수록 빠르게 인식되지만 소비 전류가 많아집니다.
* Advertiser와 Scanner는 세 개의 advertising channel 을 임의로 호핑하면서 데이터를 송출-스캔하기 때문에, 한 채널에 둘이 오버랩 되는 경우에만 데이터가 전달됩니다. 이 물론 호핑하는 속도가 빠르기 때문에 시간이 걸리더라도 오버랩은 됩니다.

Advertising packet을 수신하는 Scanner 입장에서는 advertising packet 을 수신한 이후의 동작에 따라 두 가지로 분류할 수 있습니다.

**Passive Scanning** : Scanner는 Advertising Packet을 받고 이에 대해 따로 응답을 보내지 않는다. 따라서 해당 Packet을 보낸 Advertiser는 Scanner가 Packet을 수신했는지에 대해서 알지 못한다.

**Active Scanning** : Advertising Packet을 받은 Scanner는 Advertiser에게 추가적인 데이터를 요구하기 위해 \*Scan Request라는 것을 보낸다. 이를 받은 Advertiser는 \*Scan Response로 응답한다.

                                         ------------------------------------------------------------------------------------

Advertiser 는 세 개의 상태값을 가질 수 있으며, 이 값이 Advertising packet 에 포함되어 전송됩니다.

* Connectable / Non-connectable
  + Scanner 가 advertising packet 을 받은 이후 연결을 시도할 수 있는지를 나타냅니다.
  + Non-connectable 이면 오직 advertising packet 만을 송출하는 장치라는 뜻입니다. (예 – 비컨 전용 기기)
* Scannable / Non-scannable
  + Scanner 가 advertising packet 을 받은 이후 Scan request 를 보낼 수 있는지를 나타냅니다.
* Directed / Undirected
  + Directed 상태값을 가진다면 advertising packet 에는 송출기기와 수신기기의 Bluetooth address 만 포함하고 그 밖의 사용자 데이터를 가지지 않습니다. 이 속성은 연결을 원하는 기기가 명확히 정해져 있다는 것을 의미합니다.

일반적으로 사용되는 ADV\_IND 값은 해당 BLE 기기가 advertising 할 때 31 byte 의 사용자 데이터를 포함해서 전송하고, Scan resquest 에 응답을 하며, 연결 요청을 받아들인다는 의미입니다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Advertising    PDU | Description | Max  len | Allow scan req | Allow connect |
| ADV\_IND | Used to send connectable undirected advertisement | 31 bytes | Yes | Yes |
| ADV\_DIRECT\_IND | Used to send connectable directed advertisement | N/A | No | Yes |
| ADV\_SCAN\_IND | Used to send scannable undirected advertisement | 31 bytes | Yes | No |
| ADV\_NONCONN\_IND | Used to send non-connectable undirected advertisement | 31 bytes | No | No |
| **ADV\_EXT\_IND** | Used to indicate that an advertisement will be sent on a secondary advertisement channel. | No adv data allowed. | No | No |
| **AUX\_ADV\_IND** | Used to send connectable directed advertisement on a secondary advertisement channel. | 254 bytes | Yes | Yes |
| **AUX\_SYNC\_IND** | Used for periodic advertisements on secondary advertisement channels. | 254 bytes | No | No |
| **AUX\_CHAIN\_IND** | Used to chain advertisement packets, allowing the advertisement data to extend beyond one packet. | 254 bytes | N/A | N/A |

**State**

 Link Layer는 5가지의 RF 상태 제어가 가능하다고 말했었다.

각 디바이스는 서로 연결이 되는 과정에서 이 State를 변화시키며,

각 State의 내용은 아래와 같다.

* **Standby State** : Signal Packet을 보내지도, 받지도 않는 상태.
* **Advertising State** : Advertising Packet을 보내고, 해당 Advertising Packet에 대한 상대 디바이스의 Response를 받을 수 있고 이에 응답할 수 있는 상태.
* **Scanning State** : Advertising Channel에서 Scaning 하고 있는 상태.
* **Initiating State** : Advertiser의 Connectable Advertising Packet을 받고 난 후 Connetion Request를 보내는 상태.
* **Connection State** : Connection 이후의 상태.

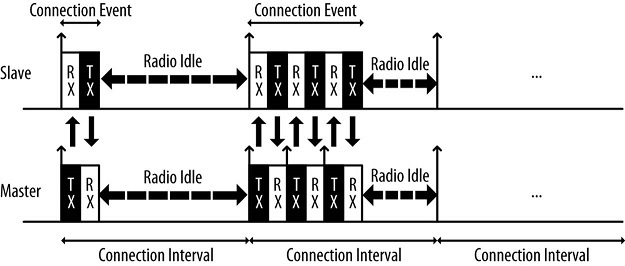
### **Connections**

두 블루투스 장치가 연결을 맺으려면 Master 장치는 스캔을 하고 connection request packet 을 보내야 합니다. 여기에 Slave 장치가 응답하면 연결이 맺어집니다. Connection request packet 은 채널 호핑을 위한 정보를 담고 있습니다.

Link Layer 는 BLE 는 블루투스 기기가 미리 정해진 몇 개의 장치만 연결될 수 있도록 해주는 White list 기능을 제공합니다. 이 기능을 사용할 경우 미리 정해진 Bluetooth address 가 아니면 연결 요청을 무시합니다.

Connection 자체는 단순히 Slave 와 Master 장치가 정해진 시간 간격에 따라 (connection interval) 데이터를 주고 받는 것입니다. 따라서 connection 을 맺을 때 마스터는 연결 설정과 관계된 connection parameter들을 전달해서 공유해야 합니다. 다음 세 항목은 그 중 주요한 파라미터들 입니다.

* Connection interval
  + 두 장치 사이의 데이터 통신(connection event)은 주기적으로 이루어집니다. 각 통신의 시작 시점 사이의 간격이 connection interval 입니다. (7.5 ms  ~ 4 s) 이 시간 간격은 배터리 소모와도 연관이 있습니다.
* Slave latency
  + Slave 장치가 주기적으로 이루어지는 connection event 에 응답하지 않아도 연결해제 되지 않는 횟수입니다.
* Connection supervision timeout
  + 지정된 시간동안 유효한 데이터 송수신이 없는 경우 연결이 해제된 것으로 간주됩니다.

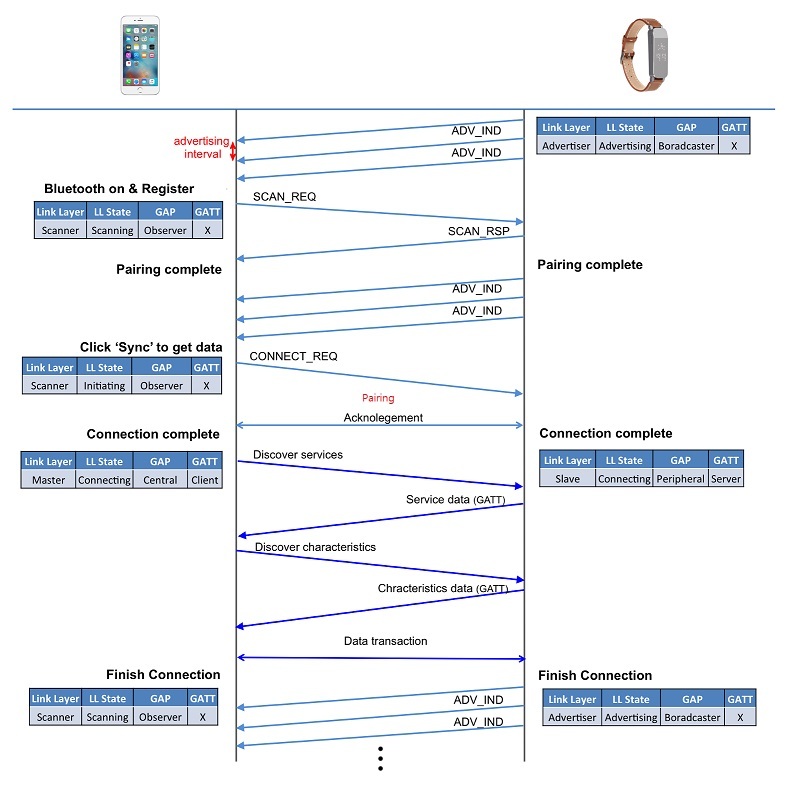


연결 후 데이터를 실어나르는 data packet 은 사용자가 사용할 수 있는 27 byte의 공간(payload)을 제공합니다. 이 공간은 프로토콜이 사용되는 방식에 따라 변경될 수 있기 때문에 일반적으로는 20byte로 제한되어 사용됩니다.

이 밖에도 Link Layer 에서는 몇 가지 제어 기능을 추가로 담당하는데 그 중에서도 다음 두 가지 기능이 중요합니다.

* Changing the connection parameters
  + 연결을 시작하기 위해 Master 장치가 연결 요청을 보낼 때 connection parameter 를 전달해서 공유한다고 했습니다. 하지만 처음에 설정된 이 파라미터들을 연결 된 상태에서 바꿔줘야 할 경우가 있습니다. 예를 들어 상당량의 데이터 전송을 위해 connection interval 을 짧게 바꾸는 대신 배터리 소모를 희생하는 상황이 이에 해당됩니다. Link Layer 는 Master/Slave 장치가 connection parameter 를 연결 도중에 변경할 수 있도록 처리해 줍니다.
* Encryption
  + 암호화된 데이터 통신을 사용하는 경우 데이터 암복호화 작업을 담당합니다.

BLE 장치가 서로를 발견한 뒤(advertising/scanning) 연결을 맺고(connection) 데이터 통신을 하기까지 서로 주고 받는 무선 신호를 도식화 하면 아래와 같습니다.



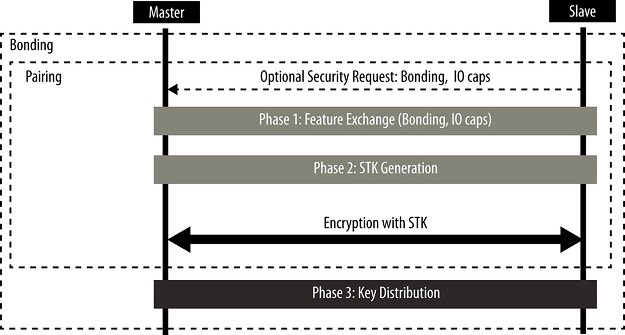
#### 4. Security Manager (SM)

SM은 두 장치가 암호화 된 보안 통신을 할 때 필요한 보안키를 생성-교환할 수 있도록 보안 알고리즘과 프로토콜을 제공합니다. SM은 BLE 장치가 담당할 두 가지 역할을 정의합니다.

* Initiator
  + Link Layer 의 Master (GAP 에서의 Central)
* Responder
  + Link Layer 의 Slave (GAP 에서의 Peripheral)

SM 은 다음의 세가지 procedure 를 제공합니다.

* Pairing
  + 보안 링크를 구축하기 위해 temporary security encryption key 를 생성합니다. Encryption key는 연결이 지송되는 동안 유지되지만 저장되지 않기 때문에 다른 연결에 재사용될 수 없습니다.
* Bonding
  + 일단 Pairing 과정이 끝나면 shared security key 를 저장해서 다음번 연결해도 재사용할 수 있는 보안 관계를 구축할 수 있습니다. 이 상태를 Bonding 또는 Pairing with Bonding 이라고 합니다.
* Encryption Re-establishment
  + Bonding 과정이 끝나면 키가 양쪽 장치에 저장됩니다. Encryption re-establishment 는 저장된 키를 이용해서 다음번에 다시 연결될 때 Pairing-Bonding 없이 보안 연결을 재설정하는 방법을 제공합니다.



SM이 동작하는 구조는 위 이미지와 같습니다. Phase 1 에서 temporary key 생성을 위해 필요한 정보들을 교환합니다. Phase 2 에서는 Temporary encryption key(STK, Short Term Key) 를 양쪽 장치에 생성하고 연결을 암호화 하는데 사용합니다.  이후  Bonding을 실행한다면 Key distribution 과정을 통해 permanent key 를 받아 저장하고 다음번 접속 때 재사용합니다.

Pairing procedure는 temporary security encryption key(STK) 를 양쪽 장치에 생성하기 위해 Security Manager Protocol(SMP) packet 을 교환합니다. 이 과정이 끝나면 최종적으로 STK 를 이용한 암호화 통신이 가능합니다.

#### 

#### 5. GAP (Generic Access Profile)

**Role**

* **Broadcaster** : Link Layer에서 Advertiser 역할에 상응한다. 주기적으로 Advertising Packet을 보낸다. 예를 들면, 온도센서는 온도 데이터를 자신과 연결된 디바이스에게 일정 주기로 보낸다.
* **Observer** : Link Layer에서 Scanner 역할에 상응한다. Broadcaster가 뿌리는 Advertising Packet에서 data를 얻는다. 온도센서로부터 온도 데이터를 받아서 디스플레이에 나타내는 태블릿 컴퓨터의 역할이다.
* **Central** : Link Layer에서 Master 역할과 상응한다. Central 역할은 다른 디바이스의 Advertising Packet을 듣고 Connection을 시작할 때 시작된다. 좋은 성능의 CPU를 가지고 있는 스마트폰이나, 태블릿 컴퓨터들의 역할이다.
* **Peripheral** : Link Layer에서 Slave 역할과 상응한다. Advertising Packet을 보내서 Central 역할의 디바이스가 Connection을 시작할 수 있도록 하게끔 유도한다. 센서 기능이 달린 디바이스들의 역할이다.

GAP

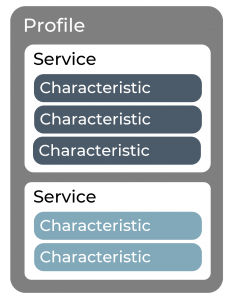
Role           Connectionless  ( 비연결 )                               Connection-oriented  ( 연결 지향 )

|  |  |  |
| --- | --- | --- |
| **Active** | **Broadcaster**   * An Advertiser * Transmit only * Periodically sends advertising data packets ( 주기적으로 advertising data packets 전송) * Does not allow connection ( 연결 허락 X ) * Example: TILE/iBeacons broadcasting ID/ location data ( 예시 - 비콘 ) | **Peripheral**   * A Follower * Advertises to establish a connection ( 연결 설정을 알림 ) * Allows single connection to central peer ( 센트럴과 단일 연결 허락 ) * Example: Bluetooth headphones in pairing mode ( 예시 - 블루투스 이어폰 ) |
| **Passive** | **Observer**   * A Scanner * Receive only * Looks for advertising data packets ( advertising data packets 을 찾는다. ) * Does not initiate connection ( 연결을 시작 하지 않음 ) * Example: Apps receiving ID/ location data for user display | **Central**   * An Initiator * Looks for advertising to initiate a connection (  연결을 하기 위해  연결 설정을 찾음  ) * Allows multiple connections to peripheral peers ( 다중 연결을 허용 ) * Example: Smartphone Bluetooth settings app searching for connectable devices |
| **Data** | * Public * Unidirectional Advertising to Scanner * 31 bytes advertising packet payload * 31 bytes scan response payload | * Before connection: Same as Broadcaster/Observer ( 연결전  브로드케스트 /  옵저버와 동일 ) * After connection: Private, Bidirectional GATT data ( 연결후  비공개 , 양방향 GATT 데이터 ) |
| **Diagram** | Figure 1. Connectionless | Figure 2. Connection-oriented  https://blog.kakaocdn.net/dn/nn7Om/btrQeT4SMQ0/7qKtKQPyQIhCbBRfRJTny0/img.png |

Note that a device is not limited to performing only one of these roles

— the specification allows for multiple communication policies simultaneously.

#### 6. GATT (General Attribute Profile)



 BLE 데이터 교환을 관리하는 GATT는 디바이스들이 데이터를 발견하고 읽고, 쓰는 것을 가능하게 하는 기조적인 데이터 모델과 절차를 정의한다

(BLE 기기간의 모든 통신은 GATT sub-procudure로 핸들링되므로 application, profiles는 GATT를 직접적으로 이용하게 된다.)

**디바이스 간의 Low-level에서의 모든 인터렉션을 정의하는 GAP와는 달리**

**GATT는 오직 데이터의 포맷 및 전달에 대해서만 처리한다.**

 GATT는 ATT를 이용하여 장치 사이에 데이터를 **전송하는 방식을 결정**하는 프레임 워이다.

ATT는 Service와 Characteristic이라는 개념을 사용하여 **어떤 데이터 'attribute'를 다른 장치에게 보여주도록 하며**,

이때 어트리뷰트를**보여주는 장치를 서버**라고 하고,**보는 장치를 클라이언트**라고 한다.

**Role**

* **Client (Master):**Central 역할을 했던 장치에서 Client역할을 하며, 연결 과정에서 연결 간격을 조정하고 Service 항목을 통해 데이터 요청의 기능을 수행   ( 스마트폰 / PC )
* **Server (Slave):**Peripheral 역할을 했던 장치에서 Server역할을 하며, 연결 과정에서 초기 연결 간격을 명시하며 Client에서 보낸 연결 간격 조정 값의 적용 여부를 판단하고 Client의 데이터 요청에 대한 응답 기능을 수행

\* 중요 Client와 Server는 일반적으로  마스터가 Client를 담당하고 슬레이브가 Server를 담당하지만.

GATT는  마스터 슬래이브의  역할에 따른것이 아니라,  데이터 흐름에 따라 다른 것이다.

즉 마스터도 - 서버가 될수있고,  슬레이브도 - 클라이언트가 될 수 있다.

**Profile**

기기에 어떻게  connect 하는지 기기가 제공하는 Service 들을 기술한다.

1개의 Profile에는 1개이상의 Service 가 정의될 수 있다.

**Service**

1개의 Service 에는 1개 이상의 Characteristic 이 정의될 수 있다.

1개의 Service  마다 고유한 UUID가 할당된다.

SIG adopted 서비스 의 UUID는 16비트로 지정되어있다.

Custom Service 의 UUID는 128비트로 지정가능하다.

**Characterisitc**

Value 와 Descriptor 를 포함한다.

Descriptor  는 Value 를 설명하는 정보.

Charaterristic 은 고유 UUID 가 할당된다.

SIG adopted Characteristic 은 지정된 16bit UUID, Custom Characteristic 은 128 bit 로 지정가능하다.

**Attribute**

ATT/GATT 계층에서 정보 표현하는 최소단위.

Handle(2bytes), Type(2bytes), Value(0~512byte), Permission(상황따라 다름) 로 구성되어있다.

Handle : Attribute address

Atrtribute Type : 16bit UUID 지정되어있고, 이는 Bluetooth SIG에 발급한것임.

Value : 데이터.

Permission : read/write, security 설정.

## 클라이언트 대 서버 - GATT 기능

**BLE 설계의 또 다른 중요한 개념은 GATT 서버** 와 **GATT 클라이언트** 의 차이점 입니다. 이러한 역할은 상호 배타적이지 않지만 일반적으로 장치는 둘 중 하나입니다. 장치가 수행하는 역할은 장치가 작동하는 데 필요한 방식에 따라 다릅니다. 다음은 각 종류의 기능에 대한 기본 요약입니다.

* + **GATT 클라이언트** - 읽기, 쓰기, 알림 또는 표시 작업을 통해 원격 GATT 서버의 데이터에 액세스하는 장치
  + **GATT 서버** - 데이터를 로컬에 저장하고 원격 GATT 클라이언트에 데이터 액세스 방법을 제공하는 장치

이전에 정의된 마스터/슬레이브 구분과 달리 애플리케이션이 연결의 각 측면에 대한 데이터 구조와 흐름을 정의하는 방법을 기반으로 한 장치가 실제로 이 두 가지 모두를 동시에 수행할 수 있음을 쉽게 알 수 있습니다. 슬레이브(주변) 장치가 GATT 서버가 되고 마스터(중앙) 장치가 GATT 클라이언트가 되는 것이 가장 일반적이지만 이것이 필수는 아닙니다. **장치의 GATT 기능은 마스터/슬레이브 역할과 논리적으로 분리됩니다.**마스터/슬레이브 역할은 BLE 무선 연결이 관리되는 방식을 제어하고 클라이언트/서버 역할은 데이터의 저장 및 흐름에 따라 결정됩니다.

SDK 아카이브 및 [**온라인**](http://community.silabs.com/t5/Wireless-Knowledge-Base/tkb-p/4%40tkb/label-name/bluetooth%20smart?labels=bluetooth+smart) 에 있는 대부분의 예제 프로젝트는 GATT 서버로 설계된 슬레이브(주변) 장치를 구현합니다. 이는 iOS 플랫폼용으로 무료로 제공되는 많은 스캔 앱과 Android 4.3에서 공식 BLE API가 구현된 Android 플랫폼용으로 출시될 앱으로 쉽게 테스트할 수 있습니다. 그러나 GATT 클라이언트용으로 설계된 연결의 마스터(중앙) 끝을 구현하는 몇 가지 예가 있습니다. **여기에는 SDK 아카이브의 /src** 폴더에 있는 "thermometer-demo" 프로젝트 와 [**이 BGScript 기반 Health Thermometer Collector가 포함**](http://community.silabs.com/t5/Wireless-Knowledge-Base/BGScript-htm-collector-Health-Thermometer-collector-BLE-master/ta-p/147715) 됩니다.

BGAPI 프로토콜을 사용하여 외부에서 모듈을 제어하도록 설계된 [**Python**](https://github.com/jrowberg/bglib/tree/master/Python/Examples) 및 [**Visual C# .NET**](https://github.com/jrowberg/bglib/tree/master/MSVCSharp/Examples) 에 사용할 수 있는 타사 비공식 BLE 마스터 예제도 있습니다 (자세한 내용은 [**이 KB 문서**](http://community.silabs.com/t5/Wireless-Knowledge-Base/REFERENCE-What-is-the-difference-between-BGScript-BGAPI-and/ta-p/147575) 참조 ).

## 수신 vs. 전송 - 데이터 이동

이제 마스터와 슬레이브 장치, GATT 클라이언트와 GATT 서버 간의 차이점을 설정했으므로 실제로 GATT 구조를 정의하는 방법과 장치 A에서 장치 B로 데이터를 이동하는 데 사용하는 방법 서버 대 클라이언트 또는 클라이언트 대 서버? 물론 양방향 모두 쉽게 가능합니다.

SDK를 사용하여 빌드된 BLE 프로젝트에서 GATT 구조는 프로젝트 소스 파일의 일부인 " **gatt.xml " 파일에 정의됩니다.**이 파일의 구조는 기술 포럼의 BLE112 및 BLE113 페이지에서 사용할 수 있는 **Profile Development Kit 개발자 가이드 에 문서화되어 있습니다.**데이터의 구조와 흐름은 항상 GATT 서버에서 정의되며 클라이언트는 단순히 서버에서 노출되는 모든 것을 사용하게 됩니다.

<https://community.silabs.com/s/article/ble-master-slave-gatt-client-server-and-data-rx-tx-basics?language=en_US>

# Indication and Notification

indication 및 notification 은 속성(ATT) 프로토콜을 통해 보낼 수 있는 명령입니다.

따라서 ATT 계층에는 두 가지 역할이 정의되어 있습니다.

* **클라이언트** 장치는 GATT 프로토콜을 사용하여 BLE 링크를 통해 원격 리소스에 액세스합니다.  
  일반적으로 마스터는 클라이언트이기도 하지만 필수 또는 필수는 아닙니다.
* **서버** 장치에는 GATT 데이터베이스, 액세스 제어 방법이 있으며 원격 클라이언트에 리소스를 제공합니다.   
  일반적으로 슬레이브는 서버이기도 합니다.

BLE 표준 은 서버의 데이터를 클라이언트로 전송 하는 두 가지 방법 인

indication 및 notification를 정의합니다.

각 메시지의 사양에 정의된 최대 데이터 페이로드 크기는 **20바이트**입니다.

indication 및 notification는 **서버에 의해 시작**되지만 **클라이언트에 의해 활성화**됩니다.

Notification 은 승인할 필요가 없으므로 더 빠릅니다.

따라서 서버는 메시지가 클라이언트에 도달했는지 알 수 없습니다.

Indication 은 의사 소통을 위해 확인해야합니다.

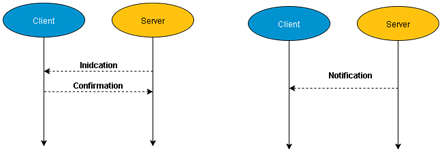
클라이언트는 확인 메시지를 서버에 다시 보냈습니다.

이렇게 하면 서버는 메시지가 클라이언트에 도달했음을 알 수 있습니다.

ATT 프로토콜에 의해 정의된 한 가지 흥미로운 점은 확인이 수신되지 않은 경우

서버가 두 개의 연속 표시를 보낼 수 없다는 것입니다.

즉, 다음 표시를 보내기 위해서는 각 indication의 확인을 기다려야 합니다.



정리

 - 서버에서 클라이언트에게 데이터를 전송하는 방법에는 Notification / Indication 두가지 방법이 있다.

 -  Notification / Indication 는 서버가 클라이언트에게 보내지만, 활성화는 클라이언트가 한다.

 -  Notification 은 서버가 클라이언트에게 데이터를 보내지만, 받았는지 확인 할 수없다 ( 응답 X )

 -  Indication 은 데이터를 받은 클라이언트가 서버에게 확인메세지를 보내야 한다. ( 응답 0 )

추가

서버는 통신 시작 시 표시 또는 알림을 보낼 수 없습니다.

먼저 클라이언트는 서버 측에서 알림 및 표시 권한을 활성화해야 서버가 표시 또는 알림을 보낼 수 있습니다.

이 절차는 클라이언트가 통지/표시될 특성의 CCCD(Client Characteristic Configuration Descriptor)를 작성하는 것을

포함합니다.

즉, 클라이언트는 서버에 특정 특성에 대한 알림을 요청할 수 있습니다.

클라이언트가 서버에서 이러한 특성에 대한 알림을 활성화하면

서버는 사용할 수 있게 될 때마다 클라이언트에 값을 보낼 수 있습니다.

예를 들어 심박수 센서 응용 프로그램을 심박수 스마트폰 응용 프로그램에 연결하는 경우를 생각해 보십시오.

 심박수 서비스는 심박수 측정 특성을 알릴 수 있습니다.

이 경우 센서는 서버이고 스마트폰은 클라이언트입니다.

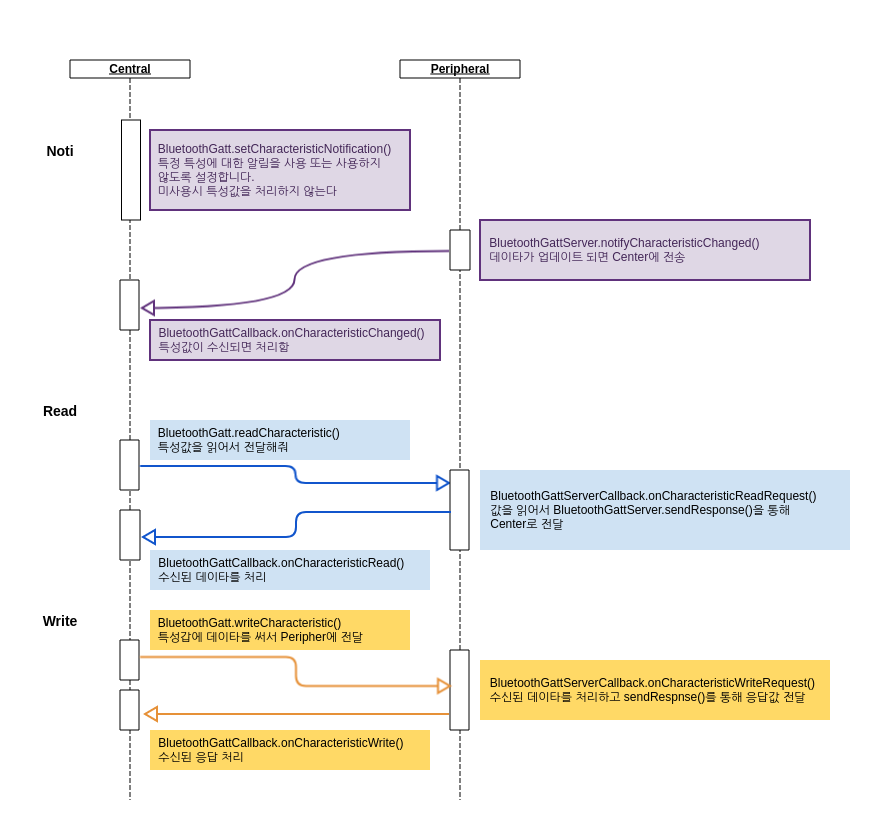
장치가 연결되면 스마트폰 애플리케이션은 CCCD를 통해 심박수 측정 특성의 알림 권한을 설정해야 합니다.

그런 다음 스마트폰 애플리케이션(클라이언트)이 알림이 활성화된 CCCD를 설정하면 심박수 센서(서버)가 심박수 측정이 가능할 때마다 알림을 보낼 수 있습니다.

특성에 표시 속성이 있는 경우에도 이와 동일한 절차가 필요합니다.

결국 클라이언트는 서버가 특성을 표시하거나 알릴 수 있도록 하는 것입니다.

Read  Write Noti   흐름



용어

* **Generic Attribute Profile (GATT)**  
  GATT 프로필은 BLE 링크를 통해 "속성"으로 알려진 짧은 데이터 조각을 보내고 받기 위한 일반 사양입니다. 현재의 모든 BLE 애플리케이션 프로필은 GATT를 기반으로 합니다. 자세한 내용은 GitHub에서 [Android BluetoothLeGatt 샘플](https://github.com/android/connectivity-samples/tree/master/BluetoothLeGatt) 을 검토하세요 .
* **Profiles  
  Bluetooth SIG** 는 BLE 장치에 대한 많은 [프로필 을 정의합니다.](https://www.bluetooth.org/en-us/specification/adopted-specifications)프로필은 특정 응용 프로그램에서 장치가 작동하는 방식에 대한 사양입니다. 장치는 둘 이상의 프로필을 구현할 수 있습니다. 예를 들어, 장치에는 심박수 모니터와 배터리 잔량 감지기가 포함될 수 있습니다.
* **Attribute Protocol (ATT)**  
  GATT는 **Attribute Protocol (ATT)** 위에 구축됩니다.   
  이를 GATT/ATT라고도 합니다. ATT는 BLE 장치에서 실행하도록 최적화되어 있습니다. 이를 위해 가능한 한 적은 바이트를 사용합니다. 각 속성은 정보를 고유하게 식별하는 데 사용되는 문자열 ID에 대한 표준화된 128비트 형식인 UUID(Universally Unique Identifier)로 고유하게 식별됩니다. ATT에 의해 전송되는 속성 은 특성 및 서비스 로 형식이 지정됩니다 .
* **Characteristic**  
  **Characteristic**에는 단일 값과 특성 값을 설명하는 0-n 설명자가 포함됩니다.   
  **Characteristic**은 클래스와 유사한 유형으로 생각할 수 있습니다.
* **Descriptor  
  Descriptor**는 특성 값을 설명하는 정의된 속성입니다.   
  예를 들어 **Descriptor**는 사람이 읽을 수 있는 설명, 특성 값에 대해 허용되는 범위 또는 특성 값에 특정한 측정 단위를 지정할 수 있습니다.
* **Service  
  Service**는 **Characteristic**의 모음입니다.   
  예를 들어 "심박수 측정"과 같은 **Characteristic**을 포함하는 "심박수 모니터"라는 서비스가 있을 수 있습니다. [bluetooth.org](https://www.bluetooth.org/en-us/specification/adopted-specifications) 에서 기존 GATT 기반 프로필 및 서비스 목록을 찾을 수 있습니다 .

**MTU  ( Maximum Transmission Unit )**

- 네트워크 인터페이스에서 세그먼트 없이 보낼수 있는 최대 데이터 그램 크기값.

만약에 MTU 값 이상의 크기라면,  여러개의 패킷으로 분할이 될 것임.( 분할을 직접 해줘야 하는지 자동인지 확인 필요 )

- 간단하게 MTU는 패킷이 한번에 보낼수 있는 최대 크기라고 볼수 있다.

MTU 최대값(ATT 메시지 전송 단위의 최대 크기)은 250바이트

최대 길이는 MTU – 3으로 제한되며 이 경우 247바이트가 됩니다. ( 알림 및 표시 3바이트 제외 )

데이터 구조 참고

<https://punchthrough.com/ble-throughput-part-4/>