A Summary of Bluetooth Low Energy

John Harris, Logan Small,
Matt Hopkins, Nathaniel de Lautour
Defence Technology Agency (DTA),
New Zealand Defence Force (NZDF),
April 2020.

***Abstract*—Bluetooth Low Energy (BLE) is a low power evolution of the Bluetooth personal area network standard capable of power efficient short-range wireless communication. BLE chipsets are now commonplace in consumer electronic devices facilitating connectivity to a multitude of wireless sensors and wearables. A key driver in this market is the BLE Advertisement which allows small packets of data to be passed easily between devices. This paper explores the technology behind Bluetooth, BLE and the concept of an Advertisement.**

I. Introduction

Bluetooth is a wireless personal area network designed for short range, low data rate applications such as handsfree phones, wireless keyboards and audio streaming applications. With an effective range of less than 100 metres, Bluetooth is intended to compliment other wireless networking standards such as IEEE 802.11 (Wi-Fi) which can offer significantly increased data throughput at the cost of high system resource usage and power consumption. With the initial specification released in May 1998, Bluetooth has seen significant market growth, totalling four billion shipments in 2019 [1]. This is primarily due to the smartphone revolution in the early 2000's and now the potential of the Internet of Things (IoT).

The Bluetooth technical standard is maintained by the Bluetooth Special Interest Group (SIG) which consists of over 35,000 companies and organisations worldwide [1]. The SIG is responsible for defining how Bluetooth devices should operate, ensuring interoperability amongst manufacturers and adding enhancements to the standard to track the latest technological developments. Over the last 20 years since Bluetooth 1.0 there have been over eleven amendments to the standard culminating in the latest release of Bluetooth 5.2 in January 2020. Possibly the most important improvement, and certainly the most relevant to this discussion, is the addition of Bluetooth Low Energy (BLE) in release 4.0.

Since release 4.0 in June 2010, the Bluetooth standard comes in two forms: Basic Rate/Enhanced Data Rate (BR/EDR) and Low Energy (LE). Of these two forms,

- BR/EDR encompasses all versions of *classic* Bluetooth. i.e. Bluetooth between versions 1.0 and 4.0. This form offers a physical layer data rate of between 1Mbps for BR and 3Mbps for EDR.
- LE uses a new physical layer which is not backwards compatible with classic Bluetooth products. Designed for ultra-low power IoT applications, BLE reduces the data rate to 1Mbps, and the maximum transmit (TX) power to 10mW from 100mW in classic Bluetooth.

A Bluetooth chipset can either be *single-mode*, whereby only the LE stack is implemented; or *dual-mode*, where both the BR/EDR and the LE stacks are implemented. Typically, a smart phone or PC would use a dual-mode device, whereas a low power wearable would implement a single-mode device for power preservation purposes.

Although much of this discussion is applicable to both forms of Bluetooth, the remainder of this paper will consider only Bluetooth Low Energy. An excellent resource for highly detailed information on BR/EDR and LE is the Bluetooth standard available at [2].

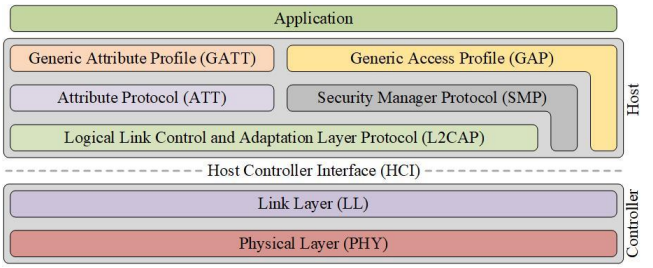

Figure 1: Bluetooth Low Energy software stack splitting controller and host functionality

An important concept which should be considered when reading this paper is the BLE stack, as depicted in Figure 1. The stack represents the hierarchical position of the key software components that make up Bluetooth Low Energy. This paper is loosely structured around each component of the stack, starting from the bottom and working up.

II. Physical Layer

The Bluetooth Low Energy physical layer (PHY) operates in the unlicensed industrial, scientific and medical (ISM) band at 2.4GHz. BLE defines 40, 1MHz channels spaced evenly between 2400-2483.5MHz with centre frequencies at $f = 2402 + (k * 2)MHz,\ \ k = \{0, 1 \ldots 39\}$. Figure 2 depicts the BLE channels with the logical channel number assignments listed below. The BLE spectrum is shared with other 2.4GHz ISM transmitters such as Wi-Fi which uses three non-overlapping channels in this range. BLE channels 37, 38 and 39 are assigned as *advertising channels* (coloured purple in Figure 2) with the remaining 37 used as *data channels* (coloured blue). The advertising channels are distributed across the available spectrum to avoid the busiest Wi-Fi channels which will interfere with BLE transmissions.

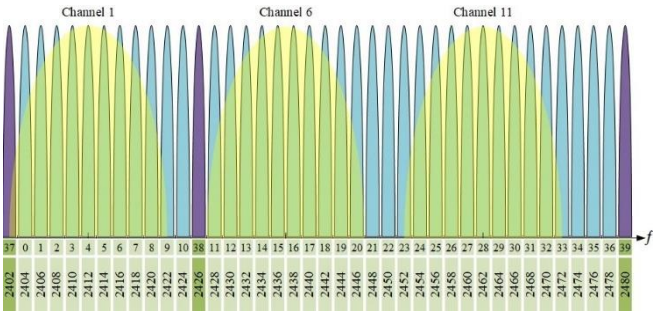

Figure 2: BLE channel assignments in the 2.4GHz ISM band. Also shown are Wi-Fi channels 1, 6 and 11 for reference.

To minimise complexity, BLE uses Gaussian frequency shift keying (GFSK), modulating binary symbols at a rate of 1Msym/s hence achieving a maximum bit rate of 1Mbps.

Interference and fading are overcome with adaptive frequency hopping spread spectrum (FHSS) over the 37 data channels and three advertisement channels. BLE employs both frequency division multiple access (FDMA) and time division multiple access (TDMA) by assigning specific hop patterns and time slots in which devices can transmit a packet. BLE specifies the minimum TX power as 0.01mW (-20dBm) and the maximum as 10mW (10dBm) further enhancing the power efficiency of devices.

III. Link Layer

The Bluetooth Low Energy link layer (LL) is responsible for controlling the PHY and ensuring multiple devices interact using known formats and protocols. To achieve this, the Bluetooth SIG have defined multiple *states* and *roles* for BLE devices at the link layer. The most relevant of the states are:

- *Advertising* – A device is actively transmitting advertising packets on channels 37, 38 and 39.
- *Scanning* – A device is listening for advertising packets on channels 37, 38 or 39.
- *Connection* – A device becomes either a *master* or *slave*, exchanging data packets between two devices.

Two devices wishing to connect first enter a discovery mode whereby one is in the advertising state (the *advertiser* role) and the other is in the scanning state (the *scanner* role). The advertiser sends advertising packets containing basic information about the device. All scanners receive these packets. At some point, the scanner enters the connection state becoming an *initiator* by sending a specific data packet to the advertiser. Should the advertiser accept this connection, the scanner now becomes the master and the advertiser, the slave. At this point the master is also known as a *central* and the slave a *peripheral*. An advertiser and scanner that never initiate a connection may be known as a *broadcaster* and *observer* respectively. These terms are discussed further in Section V.

To ensure interoperability, the Bluetooth standard defines the specific format and sequence of packets that must be exchanged to achieve the process described above.

A. Air Interface Packets

Figure 3a shows the fundamental format of all BLE packets. The basic structure is thus: The Preamble is used to synchronise clocks in the transmitter and receiver. The Access Address is fixed for advertisement packets but varies for data packets and is used to identify connections where the device address is not present. The protocol data unit (PDU) contains header information and the device's payload it wishes to transmit. A cyclic redundancy check (CRC) completes the packet to add limited error detection capability.

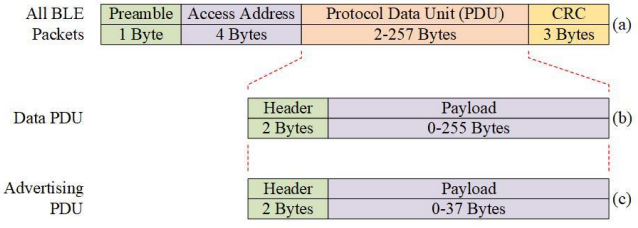

Figure 3: Basic format for BLE advertisement and data packets

Although the basic structure depicted in Figure 3a is identical for both advertisement and data packets, the PDU varies depending on the state and role of the BLE device. An overview of each PDU type is given in the following sections, however the reader should study [2] (Vol. 6, Part B) for full details.

1) Advertisement PDU

As shown in Figure 4a, the advertising header is comprised of multiple fields. The two most important of these are PDU Type and Length. PDU Type is a four bit enumerated field with possible values listed in [2]. The two most common Types are ADV_IND (0x00) and ADV_NONCONN_IND (0x02). The specifics of these types can be found in [2] (Vol. 6, Part B, Section 2.3.1) however, in summary, ADV_IND defines a connectable undirected advertising event, i.e. Any scanning device can request a connection to the advertiser; and ADV_NONCONN_IND defines an undirected non-connectable advertising event i.e. scanning devices can observe advertising packets but cannot request a connection to the advertiser. In this sense, a device using Type ADV_NONCONN_IND is a broadcaster. The Length field of the advertising header defines the number of bytes in the payload excluding the MAC address.

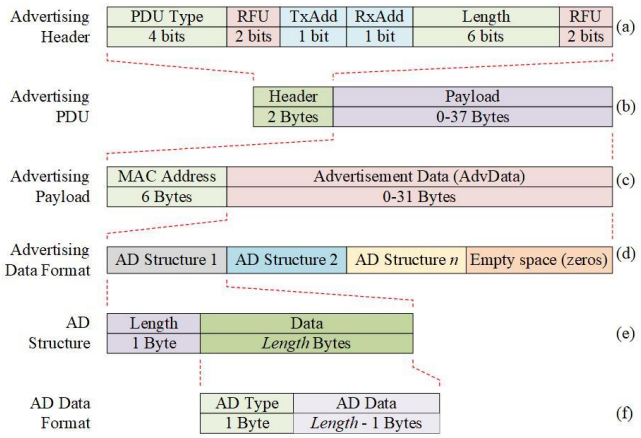

Figure 4: Advertisement packet format

As shown in Figure 4c, an advertisement payload follows the header field. This comprises a six byte MAC address (unique to the device) and multiple advertising data (AD) structures (d). Any spare payload not filled with AD structure is zero padded. Each AD structure consists of a one byte Length field noting the number of bytes in the following Data structure (e). This data structure is further subdivided into a one byte AD Type field and multiple AD Data fields (f). This complex subdivision creates a standard format of advertisement packets which can be consistently interpreted by listening observers. The AD Data field will contain data passed through the host controller interface (HCI) from a host profile and may be used by listening scanners to discover the properties of an advertiser (more information in Section V).

2) Data PDU

At the link layer, the PDU for a BLE data packet is simpler than the advertisement packet since much of the complexity is hidden in higher layer framing. As shown in Figure 5b, the data payload is prefixed by a two byte header, containing the fields listed in (a). Full details of the meaning of these fields is given in [2] (Vol. 6, Part B, Section 2.4). The LLID is an enumerated indicator identifying the data payload as either a link layer data PDU (Figure 5d) or link layer control PDU (c). If the data payload is a LL data PDU, then CtrData is simply data passed from a higher layer, usually L2CAP (Figure 1).

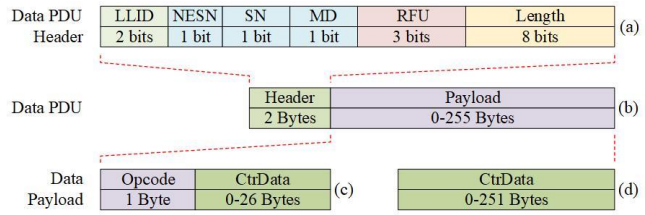

Figure 5: BLE data PDU header and payload

The length of this payload is up to 251 bytes, defined by the Length header field. If the data payload is a LL control PDU, then the format in Figure 5c is used. The Opcode is one of an enumerated list of instructions, contained within the CtrData field with a size defined by the header Length field up to a maximum of 26 bytes. Depending on the Opcode instruction number, the CtrData field in (c) will be further subdivided and formatted to convey the instructions between devices. A full description of these messages is outside the scope of this discussion, however they include instructions to commence and terminate connections, pass connection parameters and change channel mappings.

B. Air Interface Protocol

A device in the advertising state must actively transmit advertisement packets on each of channels 37, 38 and 39. The advertiser must hop channel for each transmission with no more than 10ms between subsequent transmission. One sequence of three advertisement packet transmissions is called an Advertisement Event. This is depicted in Figure 6.

Each advertisement event is repeated with a period of T_advEvent, where T_advEvent = advInterval + advDelay. The value of advInterval is chosen by the advertising node and should be an integer multiple of 0.625ms in the range 20ms to 10.24 seconds. The value of advDelay is a random value in the range 0-10ms, used to add limited collision avoidance when multiple devices are present in the same environment. To enter the connected state, a scanner may transmit a connection request on the same channel within the advertising event. The advertising device can then choose to accept the connection event and data transfer can commence.

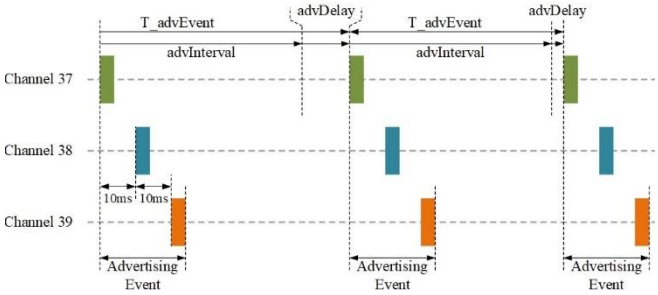

Figure 6: Advertisement transmission timing across three channels

IV. Logical Link Control and Adaptation Protocol

The logical link control and adaptation protocol (L2CAP) as depicted in Figure 1 is the most fundamental software element of the LE and BR/EDR host layer. All application layer data passes through L2CAP, through the HCI to the controller. L2CAP is responsible for abstracting the complexities of the LL and PHY layers away from the application such that more complex interactions, defined by Bluetooth *Profiles*, can be implemented. Common BLE Profiles are discussed in Section V however, the core functionality of the L2CAP layer includes:

1) Channel multiplexing

Upper layer entities are distinguished by separate logical channels connecting the L2CAP layer in each device. To an application using L2CAP, it appears that it has a direct, discrete link to a remote device. This layer multiplexes many logical links onto the same physical channel.

2) Segmentation and reassembly

Link layer data PDUs have a limited payload size of 251 bytes. The application layer should not be concerned with this limitation so L2CAP will take care of splitting the application layer PDUs to meet LL payload size, latency requirements and memory limitations.

3) Error control and retransmissions

Certain applications demand a residual error rate lower than what the link layer can natively achieve. L2CAP can add additional error checking capability and request retransmissions where necessary.

4) Support for streaming

L2CAP for BR/EDR can initiate a logical channel for streaming applications such as audio. This channel attempts to maintain a guaranteed bit rate and latency, multiplexing streaming data around other Bluetooth applications.

The full specification for the L2CAP layer can be found in [2] (Vol. 3, Part A).

V. Bluetooth Profiles

A core aspect of Bluetooth and BLE is the concept of *profiles*. A profile is a host level definition of common applications or behaviours that leverage the functionality of the link layer and L2CAP to achieve compatibility with other devices. Profiles exist for various applications such as audio streaming, human interface (mice, keyboards, etc.) and handsfree telephony. Profiles can be layered, such that one application will rely on multiple fundamental profiles.

Bluetooth Low Energy mandates the implementation of two important profiles, the Generic Access Profile (GAP) and Generic Attribute Profile (GATT). Discovery of devices is achieved through GAP, while simple inter-device communication is achieved through GATT. Since BLE is envisioned for use with low power and low data rate applications, GATT provides a lightweight data access profile used to access data on a remote device. The GAP and GATT profiles are detailed in [2] (Vol. 3, Part C) and (Vol. 3, Part G) respectively.

A. Generic Access Profile (GAP)

The generic access profile is the cornerstone in the control of a BLE host device. The GAP profile defines four roles for a BLE device:

1) Broadcaster: Periodically sends non-connectable advertising packets using the link layer advertiser role. Does not accept or initiate connections.

2) Observer: Periodically scans for advertising packets using the link layer scanner role. Does not initiate or accept connections.

3) Peripheral: Periodically sends connectable advertising packets using the link layer advertiser role and will accept connection requests from a valid central whereupon it will become the slave.

4) Central: Periodically scans for advertising packets and can initiate connections to one or multiple peripherals whreupon it will become the master.

Devices which source data, e.g. a temperature sensor, typically use the GAP broadcaster and peripheral roles. If the quantity of data is small enough to be contained within the advertisement packet payload, the sensor may choose to use an ADV_NONCONN_IND PDU type, i.e. the GAP broadcaster role. If the sensor's data is too large, or perhaps the sensor can receive calibration data, then the sensor must be a peripheral. In the peripheral role, it will initially broadcast advertisement packets of PDU type ADV_IND, allowing connections from a central, eventually transitioning to a slave device during the connection process. The key difference between broadcaster and peripheral is that a peripheral allows connections but a broadcaster does not, even though they both regularly transmit advertisement packets.

The observer and central roles offer the inverse functionality to broadcaster and peripheral respectively. A central is typically a higher power dual-mode BLE device, incorporated into a host system such as a smart phone or PC. A central device will first enter the link layer scanning state to gain knowledge of BLE devices advertising in its proximity. The central may then choose to initiate a connection to a peripheral whereupon it will become the master during the connection process. A device which never initiates a connection and only scans may be described as an observer.

B. Generic Attribute Profile (GATT)

The generic attribute profile builds on the functionality of the attribute protocol (ATT) [2] (Vol. 3, Part F) to define a framework of procedures and formats in which data can be organised and presented on a BLE device. Once a connection between two BLE devices has been established, GATT uses a simple client/server model to exchange data between each device. The data source, which is typically a peripheral device, becomes the *server* (link layer slave), while the central becomes the *client* (link layer master). Data is requested by the client and issued by the server. One BLE device can take the role of both client and server in different connections if both have data to share.

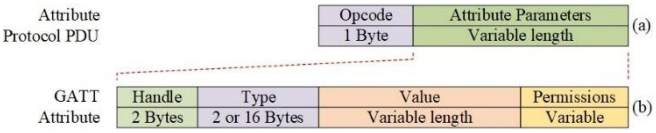

Figure 7: Packet format for ATT and GATT paylods

To ensure a common format for the organisation of data in a GATT server, the profile defines an *attribute*. Attributes are passed between BLE devices in the payload of an ATT PDU, the precise details of which are outside the scope of this paper. The format of an attribute is depicted in Figure 7b and consists of the following fields. The attribute Handle is an index assigned by the server relating to this attribute; the attribute Type is a 16 or 128 bit universally unique identifier (UUID), precisely identifying this attribute; the Value field contains the data described by the attribute Type; and the Permissions field is used by the server to determine if the client can read or write to the attribute. Although UUIDs do not need to be defined in a global repository, the Bluetooth SIG maintains a list of commonly used 16 bit UUIDs to ensure coherent discovery of data for popular applications [3]. A 128-bit UUID may be generated by the application to uniquely represent the data contained within the attribute.

The GATT profile is necessary to organise the data in a server. Using the attribute as a building block, GATT defines two final concepts, the GATT *service* and *characteristic*. A service is a collection of characteristics and other services which accomplish a specific function. A characteristic is a collection of at least two attributes which contain application data. Both the service and characteristic are attributes themselves. These concepts are conveyed in Figure 8.

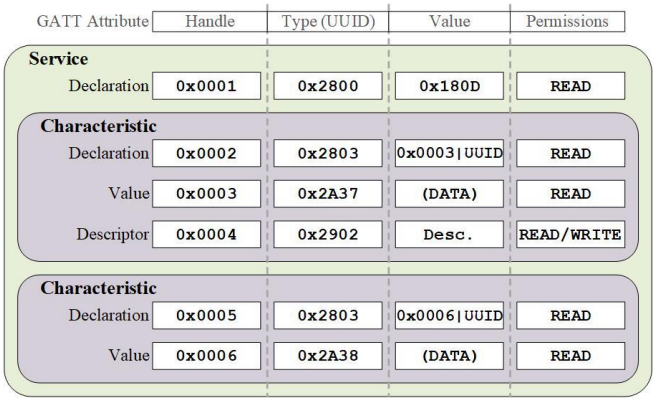

Figure 8: An example GATT service for a heart rate monitor

From Figure 8, a primary service is identified by the *service declaration* attribute with UUID 0x2800. Within the primary service, there are two characteristics. The first characteristic contains three attributes, a *characteristic declaration* attribute with UUID 0x2803 which declares the purpose (via another UUID) of this characteristic; a *value* attribute with UUID 0x2A37 which holds the application data in its value field; and the *characteristic descriptor* attribute with UUID 0x2902 which can convey a human readable description of the characteristic. Characteristics may contain any number of optional descriptors to provide information about the data.

A GATT client can interrogate a server's data via GATT transactions. The information that the server exposes can be enumerated using a request which returns all primary service UUIDs. The client may then request all characteristics contained within the service and subsequently all descriptors. Based on this information a client application can read, and when permissible write, to/from a characteristic. To prevent unnecessary polling of data, a GATT client can subscribe to *notifications* whereupon the client will be notified whenever the server has updated values.

VI. References

[1] M. Powell, "Bluetooth Market Update," 2019. [Online]. Available: https://tinyurl.com/y298qbje.

[2] "Bluetooth SIG," [Online]. Available: https://www.bluetooth.com/.

[3] B. SIG, "Bluetooth Generic Attributes," [Online]. Available: https://tinyurl.com/ydzbkour.