# BLE v4.2: Creating Faster, More Secure, Power-Efficient Designs—Part 1

*Electronic Design*
(Unpublished) Sachin Gupta Rohit Kumar
Tue, 2016-08-23 15:27

Bluetooth is the protocol of choice when it comes to wireless connectivity for numerous applications like home and industrial automation, Internet of Things (IoT), wearable electronics, human interface devices (HIDs), and many more. To meet the requirements of these various applications, the Bluetooth Special Interest Group (SIG) continues to make improvements to the Bluetooth specification.

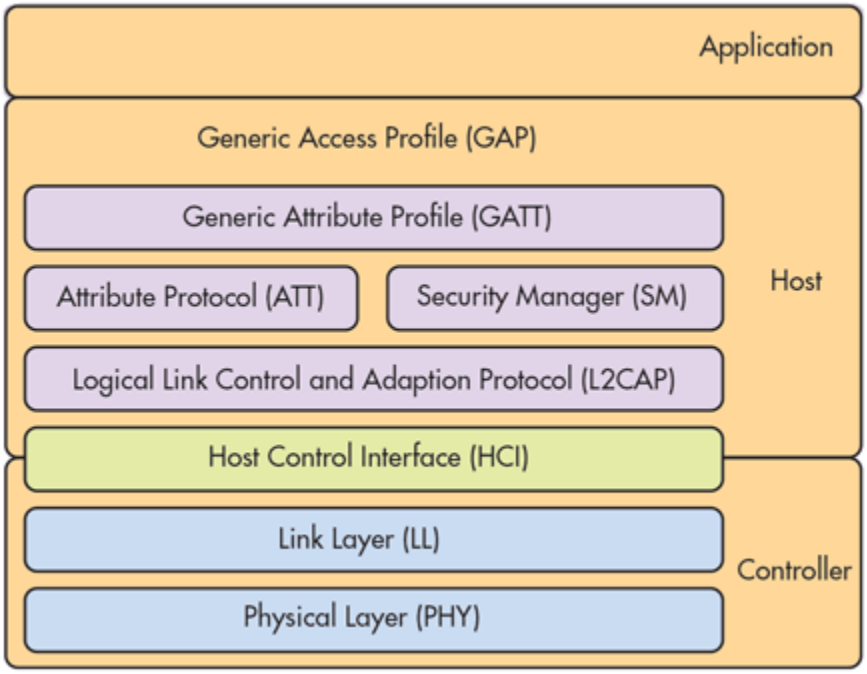
Related

Moving Bluetooth Low Energy Closer to the Mainstream

Q&A: Going Beyond Point-to-Point Wireless with Bluetooth Smart Mesh

Bluetooth Tries Working Smarter, Not Harder

In December of 2014, the Bluetooth SIG released the Bluetooth specification version 4.2, about one year after the release of version 4.1. The new 4.2 version primarily includes three updates: Low Energy (LE) Data Length Extension (DLE), privacy at Link Layer (LL), and improved security. These features increase data bandwidth, enhance privacy and security of BLE, and help reduce power consumption. In this two-part series, we will talk about each of these features in detail and how they impact system performance.



› Bluetooth Low Energy (BLE) protocol stack can be subdivided into three sections *(Fig. 1)*:

• *Controller:* This part of the protocol stack encodes the packet and transmits it as a radio signal. On reception, the controller decodes the radio signal and reconstructs the packet.

• *Host:* The host consists of various protocols and profiles (Security Manager, Attribute Protocol, and so on) that manage how two or more devices communicate with one another.

• *Application:* This is a use case that enables the host and the controller to implement a particular function.

**Link Layer (LL)**

Most of the new features in Bluetooth 4.2 center around the link layer. The link layer implements key procedures to establish a reliable physical link and features that help make the BLE protocol robust and more energy-efficient. Some link-layer functions include advertising, scanning, creating, and maintaining connections to establish a physical link. At the link layer, two roles are defined: Master and Slave.

**Data Length Extension (DLE)**

Data length extension is the feature that enables faster data transfer between two BLE devices. To understand the DLE feature, let's look at a BLE data packet at Link Layer. *Figure 2* shows the packet structure of Link Layer as per Bluetooth 4.0/4.1.

LSB MSB

| Preamble | Access address | Data Protocol Unit (PDU) | | | CRC |
|---|---|---|---|---|---|
| | | Header | Payload | MIC | |
| 1 byte | 4 bytes | 2 bytes | Up to 27 bytes | 4 bytes | 3 bytes |

If we closely observe the overhead for each data packet, there is a 1-byte preamble, 4-byte access address, 2-byte header, 3-byte cyclic redundancy check (CRC), and an optional 4-byte message integrity check (MIC). The message integrity check (MIC) is sent with the payload when encryption is used. Therefore, every link-layer packet with encryption containing 27 bytes of data has 14 bytes of overhead.

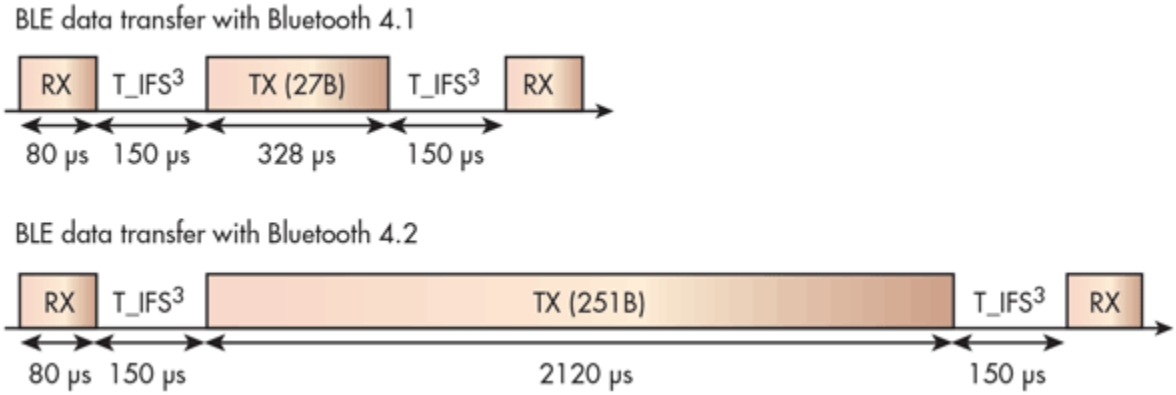
**LL Packet Structure**

Now, let's look at the link layer packet structure as defined in Bluetooth 4.2 *(Fig. 3)*. The payload size for the Bluetooth 4.2 link-layer data packet can now go up to 251 bytes, compared to 27 bytes in the older version of Bluetooth specification. Per-packet overhead remains same (i.e., 14 bytes). However, this overhead is now associated with up to 251 bytes instead of 27 bytes. This change in the maximum payload size improves throughput and reduces transaction time.

LSB MSB

| Preamble | Access address | Data Protocol Unit (PDU) | | | CRC |
|---|---|---|---|---|---|
| | | Header | Payload | MIC | |
| 1 byte | 4 bytes | 2 bytes | Up to 251 bytes | 4 bytes | 3 bytes |

*Figure 4* shows the throughput when data needs to be transmitted from one device to another using Bluetooth 4.1 and Bluetooth 4.2. In this diagram, packet time is calculated as following:




Packet Time = 8*(Number of Preamble bytes + Number of Access Address bytes + Number of Header bytes + Number of Payload bytes + Number of MIC bytes + Number of CRC bytes)/Data Rate seconds

In case of Rx packets, there will be no payload and MIC bytes. So, Rx packet time is:

Rx Packet Time = $8 \times (1 + 4 + 2 + 3)/10^6$ seconds

= 80 µs

Tx Packet time with 27-byte payload is:

Tx Packet Time = $8 \times (1 + 4 + 2 + 27 + 4 + 3)/10^6$ seconds

= 328 µs

Similarly, Tx packet time for 251 bytes payload is 2120 µs.

Also, as shown in the figure, with every Tx/Rx packet, there are two associated inter-frame spacing (T_IFS), one Tx period, and one Rx period. As the number of frames increases for a transaction, there's a proportional increase in the time taken for the transaction. Compared to Bluetooth 4.1, Bluetooth 4.2 packs more data in one frame when the data length feature is enabled, hence reducing the total time per transaction and increasing the throughput (where throughput = payload size /total time).

As shown in Fig. 4, in the case of Bluetooth 4.1 link layer, maximum payload size is 27 bytes (216 bits) and the total time taken for the transaction is 708 µs. That gives us the theoretical throughput of ~298 kb/s.

In the case of 4.2 link layer, maximum payload size is 251 bytes (2008 bits) and the total time is 2500 µs, which gives us the theoretical throughput of~ 784 kb/s. Thus, Bluetooth 4.2 provides ~2.6 times higher throughput compared to Bluetooth 4.1.

BLE 4.2 allows data length to be negotiated between the master and the slave device. It also allows RX and TX payload sizes to be asymmetric. Effective use of this feature and selection of appropriate Rx/Tx data length is important in achieving maximum throughput.

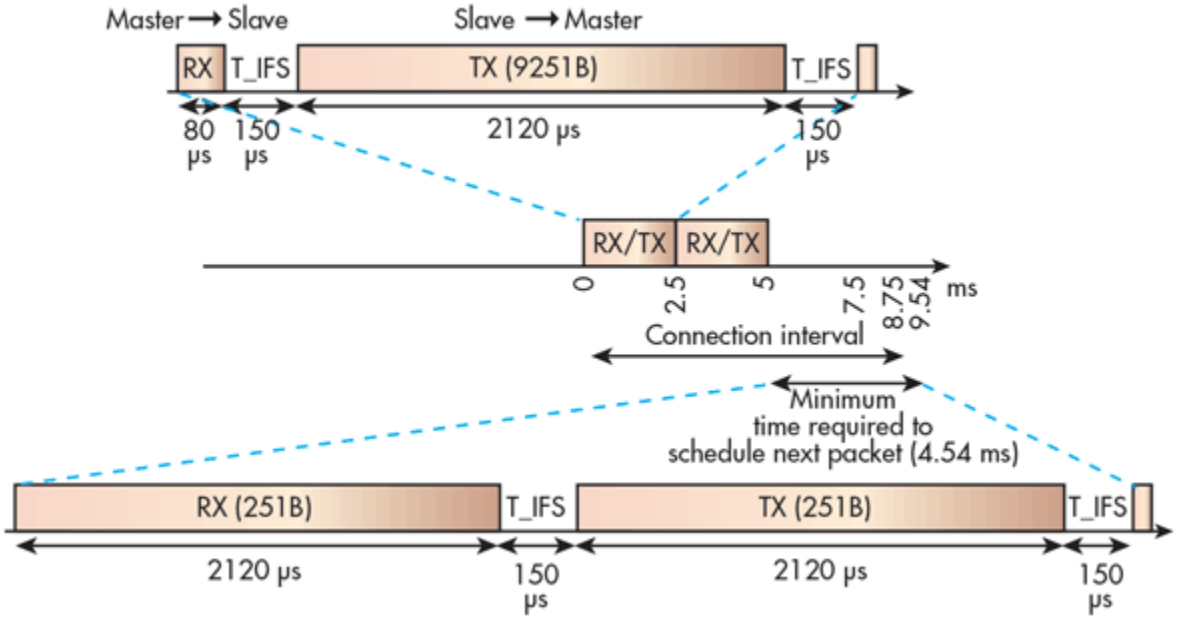
**Master/Slave Case Studies**

Let's consider an application where a BLE Slave device needs to transfer several thousand bytes to a Master device, receive empty packets from the Master device, and the connection interval is 8.75 ms. Consider one of the following settings during negotiation of data length (Slave):

Case 1: TX is 251 bytes, RX is 251 bytes

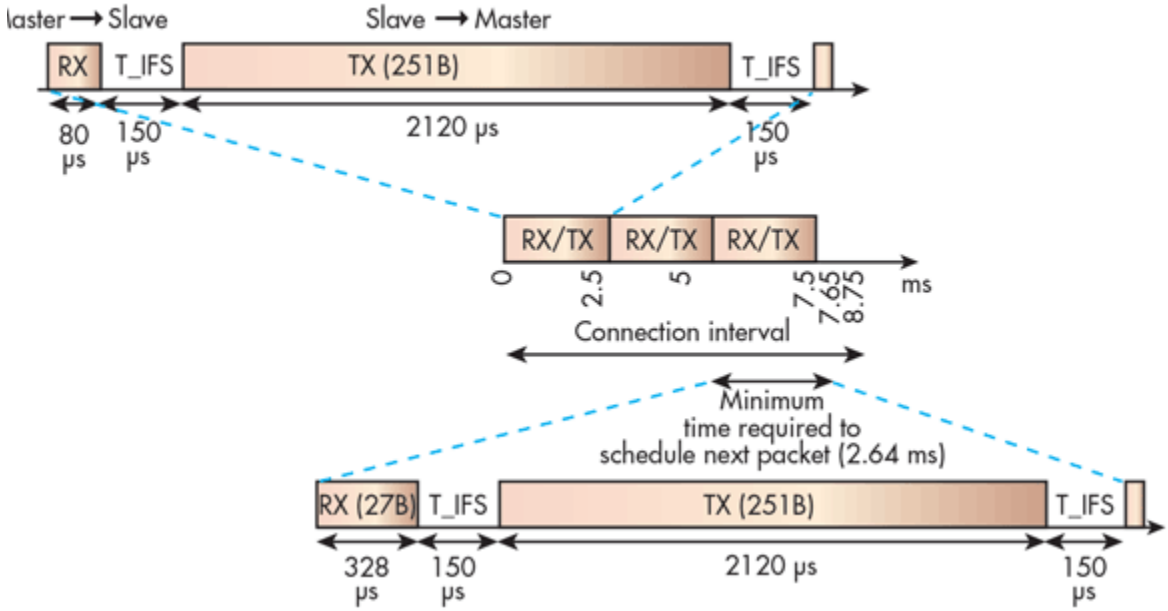
e 2: TX is 251 bytes, RX is 27 bytes

In case 1 *(Fig. 5)*, during the first Rx/Tx packet, the Rx payload size is 0 bytes and Tx payload size is 251 bytes, taking 2.5 ms including inter-frame spacing. This is the same for the second Rx/Tx packet as well. These two Rx/Tx packets take a total of 5 ms, leaving 3.85 ms available in this connection interval.




Ideally, there should be another Rx/Tx packet in the same connection interval. However, the scheduler in the master device doesn't schedule another Rx/Tx packet in this connection interval. This is because the scheduler checks if enough time is available for Tx/Rx packet based on the negotiated data length, which is 251 for both Tx and Rx in this case. An Rx/Tx packet with both Rx and Tx payload size of 251 requires 4.54 ms, as shown in diagram. However, the time available after the first two packets is 3.85 ms, resulting in only two Tx packets in this connection interval.

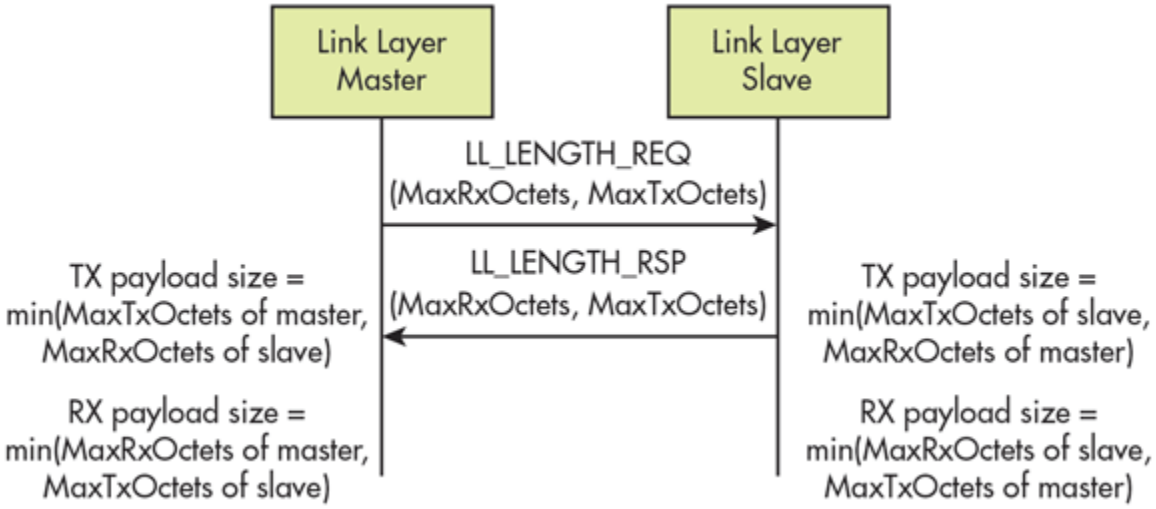
In case 2, the scheduler requires only 2.64 ms to be available in the connection interval to schedule a packet. Hence, a third packet can be accommodated in 8.75-ms connection interval *(Fig. 6)*. As shown, this case will provide 50% higher throughput compared to case 1.




While selection of the PDU size impacts throughput, there are other factors that come into play, like connection interval and maximum transmit unit (MTU) size. However, these topics are not discussed here, since they go beyond the scope of this article.

**DLE Details**

Data-length extension can be triggered by the controller for any connected device. If both devices support the data-length extension feature, then one of the devices can send a request for an updated data length and the other device responds with its own parameters. *Figure 7* shows the negotiation process.




If one device doesn't support the DLE feature and it receives a data-length update request, it returns an unknown response. This informs the device triggering the request that the other device doesn't support DLE. It will continue to transfer data with the Bluetooth 4.1 PDU size. This supports backward-compatibility.

Though the implied advantage here is improved throughput, DLE helps lower power consumption by reducing radio active time (e.g., the radio needs to be active for a shorter period of time, since fewer RX/TX packets are needed for a transaction if data size, in the case of Bluetooth 4.2, is more than 27 bytes).

example, consider a use case where 135 bytes of data need to be transmitted. In a BLE 4.1-based implementation, it takes five TX/RX packets to transmit the data while maintaining a connection. However, when the same amount of data needs to be transmitted using a BLE 4.2-based device, it takes just one TX/RX. In wireless applications, the radio consumes the majority of system power. This reduction in radio activity time can significantly improve battery life when using DLE.

Stay tuned for part two of this series, in which we'll discuss the Privacy 1.2 feature in Bluetooth 4.2.



**Source URL:** http://electronicdesign.com/communications/ble-v42-creating-faster-more-secure-power-efficient-designs-part-1