Bluetooth Low Energy (BLE)

By Adam Lipson, ECE '21

Introduction

Bluetooth is a communication technology designed to replace wired connections between devices. It has gained widespread popularity as the default method of close-range wireless communication. With the emergence of small smart devices like phones, computer peripherals, and small smart devices, a low-power solution is necessary. Bluetooth Low Energy (BLE) was introduced in 2009 as a lightweight subset of classic Bluetooth which solves this problem. While classic Bluetooth requires devices to stay active at all times to constantly transmit large amounts of data, BLE remains in sleep mode except when sending or receiving information. This allows small devices to run for years on a small battery, opening the door to a new world of smart devices able to use wireless communication.

This tech note will describe BLE using the metaphor of a technology "stack." Aspects of the technology that are implemented with physical hardware are closer to the "bottom" of the BLE stack, while more abstract software ideas are closer to the "top."

Controller

The controller is the lowest element of the BLE stack; it is comprised of physical hardware as well as the software required to send a discrete piece of information, or packet, between devices. If we consider sending a packet over BLE like sending a letter, the hardware component of the controller is analogous to the physical envelopes, trucks, and planes required for communication: it enables a packet to physically travel between devices. The software component allows devices to request to connect to another device. In the mail metaphor, the software controller is the system of drivers and routes that knows how to connect one address to another.

ATT

The Attribute Protocol (ATT) is the next layer on the BLE stack above the controller. At this level on the stack, we break the devices in communication into two separate roles: the client and server. The server is the device which stores information, and the client is the device which can read that information from the server or write new information to the server. The server also has the option to send an unsolicited message and notify the server with information. Considering the example of a smartphone app for controlling smart lights, the phone acts as the client and the lights act as the server. The lights store their current state in information at the ATT level, while the phone can make requests read the information to display information about the lights to the user or send information to turn the lights on or off. The lights could also notify the phone to alert about a malfunction.

GATT

Moving further up the BLE stack is the Generic Attribute Protocol (GATT). The ATT level described in the previous section introduces the client/server distinction and requires that the server device stores information, but it does not describe how that information is organized; the organization of information on the server is maintained at the GATT level.

To understand how the GATT level breaks down information, consider the application of a smart watch. The top level of the GATT information hierarchy is the service. Services break information into logical chunks. A smart watch might have two services: one for biometric information and another for system diagnostics. The lower level of the information hierarchy is the characteristic. For example, the biometric information service in the

smart watch might have characteristics for heart rate, body temperature, and watch motion. Each characteristic has only one value, although that value could be multiple numbers encoded and packed together: the motion characteristic could describe both horizontal and vertical motion. Using the GATT layer, the client knows how to request and interpret the information held on the server.

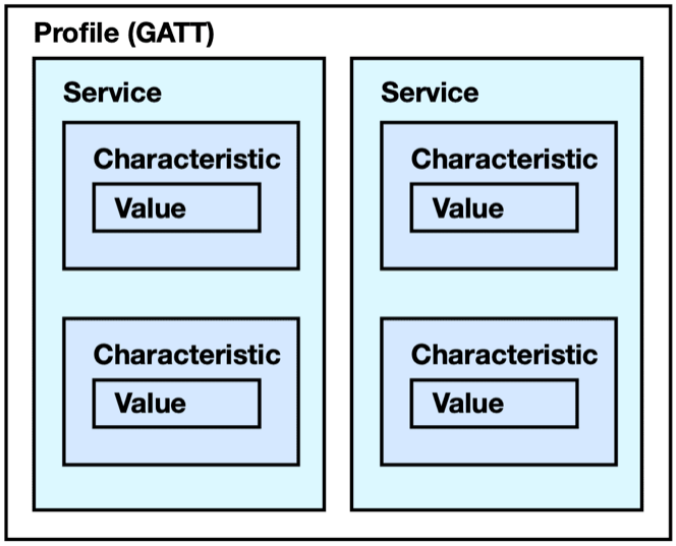

Figure 1: GATT Hierarchy [1]

GAP

The top level of the BLE stack is the Generic Access Protocol (GAP). The GAP specifies device roles as well as how devices establish connections. This is the layer that we interact with as users of BLE devices. The GAP defines two device roles: central and peripheral. These roles are completely independent of the client/server distinction at lower levels. The client/server distinction describes the storage and flow of information, while the central/peripheral distinction describes connection management.

The central device can initiate and maintain multiple connections simultaneously. Typically, the central is a more powerful device like a computer or smartphone, while a peripheral is smaller. A laptop acting as a central device can connect to many peripherals simultaneously, like a keyboard, mouse, and speaker. However, the peripherals can only maintain one connection at a time: a keyboard can only control one computer at once.

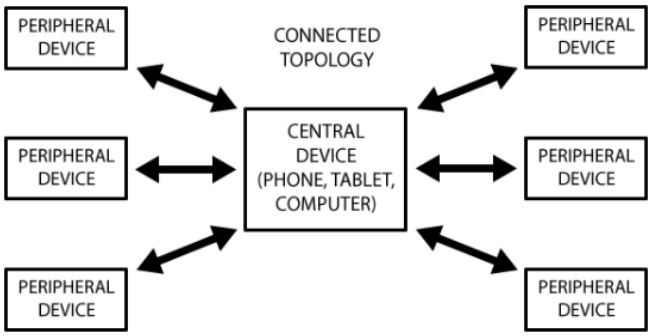

Figure 2: Simple Central/Peripheral Topology [2]

While the BLE GAP defines the behavior of a given role, it does not require that a device supports only one role. It even allows devices to support multiple roles simultaneously. Consider the case of a laptop connected to a keyboard, mouse, and smart speaker over Bluetooth. In this example, imagine that the smart speaker is connected both to the laptop and a smartphone, so the speaker is a BLE central device. In this case, the laptop is acting both as a central and peripheral. It is a central from the perspective of the keyboard and mouse, but a peripheral from the perspective of the speaker. This is possible because all levels of the BLE stack operate within one application on the device. The laptop's sound application may run as a peripheral while its handling of keyboard and mouse input may run as a central. In that case, the laptop will be unable to connect to another speaker because its sound application acts as a peripheral device.

Designing with BLE

BLE is a powerful tool for designers. It enables reliable wireless communication between devices with a low energy cost. When working with BLE, a designer will likely not need to work at the controller level. They can simply purchase a chip with an integrated radio for wireless communication and use system-level libraries for the device. The designer needs to be aware of whether the device will be a client or server, as well as a central or peripheral. Usually for small embedded-style devices, like the smart frisbee for team Razzmatazz, it makes the most sense to design a peripheral server. That way, the device can store its own data by acting as a server and act as a peripheral by connecting to a more powerful central device. The largest design task in this case is

designing the GATT profile of the device. The GATT profile depends on the way information is organized for the designer's specific application. The designer can break that information down into services and characteristics and choose communication mode(s) (read/write/notify) for each characteristic. Once the design is complete, each level of the BLE stack will function to allow the device to communicate reliably, giving the designer the power of wireless communication.

References

1. William Welbes. *What is Bluetooth Low Energy (BLE) and how does it work?,* March 2019.

2. Kevin Townsend. *Introduction to Bluetooth Low Energy*.

3. P. Bhagwat. *Bluetooth: Technology for short-range wireless apps.* IEEE Internet Computing, 5(3):96–103, June 2001.

4. Carles Gomez, Joaquim Oller, and Josep Paradells. *Overview and Evaluation of BluetoothLow Energy: An Emerging Low-Power Wireless Technology*. Sensors, 12(9):11734–11753, August 2012,