BLE Mesh Introduction

Application Note *v2.0*

1 Introduction

In July of 2017, the Bluetooth SIG released *Mesh Profile Specification v1.0* which describes a mesh profile running on top of any device version 4.0 or newer.

This application note provides an introductory overview of BLE mesh by introducing you to some basic concepts regarding how a BLE mesh works.

2 Overview

A BLE mesh is a collection of up to 32,767 physical BLE devices which send and receive messages to trigger predefined behaviors in the participating nodes.

BLE mesh is a managed flood network which broadcasts (BLE advertisement) a message to other nodes that are in range. The receiving nodes re-broadcast (or relay) the message to other nodes that are also in range. This continues until the message time to live (TTL) expires; once expired, nodes that receive the message no longer re-broadcast it. If a node receives a message that it previously received, it can immediately discard it by matching it against entries in the message cache.

BLE mesh networks use relay nodes rather than nodes that function as routers. Routers can be a single point of failure that can then result in a full network failure. By not relying on routers, these mesh networks that use flooding are far more reliable. BLE mesh messages can be received from any node within range, even if an intermediary node fails or is removed from the network.

Heartbeat messages are sent periodically to indicate a node is still alive and how many hops it may be from another node. This information can be used to optimise the TTL value.

BLE mesh does not make use of connections but rather uses the BLE broadcasts (advertisements) introduced with BT4.0. This does not mean that all BT4.0 devices automatically support mesh (the OS likely needs to expose a new API) but devices that are BT4.0 devices and newer have the *potential* to support BLE mesh.

3 Topology

All nodes in a BLE mesh can transmit and receive messages but some nodes may have one or more specific features (see Figure 1).

- Relay node – Can receive and rebroadcast a message
- Low power node – Spends most of its time in a low power state with its radio turned off
- Friend node – Works alongside a low power node by storing and forwarding messages intended for a low power node when polled by the low power node
- Proxy node – Devices without a mesh stack can interact with devices in a BLE mesh using a GATT bearer to a proxy node.

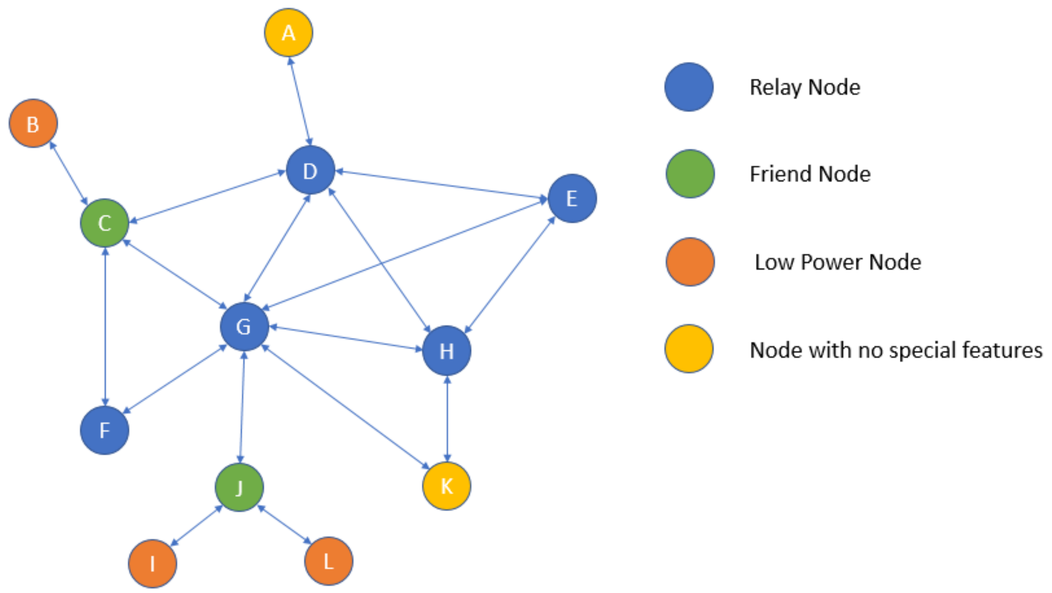

Figure 1: Specific node features

4 Publish/Subscribe

Messages are sent and received using a publish/subscribe paradigm. An outgoing message is published. The only exception is when an acknowledgment message is sent to a specific node that received the behavior-invoking message. Given the advert-based, managed-flood nature of message transmission, the following is a valid question: *How long does it take for the message to arrive at the destination?* Anecdotally, the best answer is: *messages travel at the speed of sound*.

Each message consists of an *opcode* and context *data*. The opcode dictates the behavior at the receive end; the data can be up to 380 octets. By the time the application receives a mesh message, it is completely decrypted and given to the app as a plaintext message.

5 Provisioning

When a mesh device is initially powered up, it is not provisioned. This means that it does not have a node address nor any other configuration information. In that state, when powered, it sends out 'un-provisioned' beacons upon which a provisioner acts.

To become part of a mesh network, the device must first be provisioned. To do this, a mesh provisioner sends configuration information to the un-provisioned device.

A provisioner is the only entity in a mesh that 1) is aware of all the members of the network and 2) knows how to program the publication address and the subscription lists of all the nodes (which enables the nodes to operate as a well-choreographed collective). Individual nodes are never aware of the full picture of the mesh network.

For example, in an office commissioned with 50 lights and 20 switches, only the provisioner is aware of the node addresses of all 70 devices. The individual lights and switches have no need for this information. The provisioner creates group addresses for each office region then sets the publication addresses in the switches and the subscription addresses in the lights according to the applicable group address. When a light switch's state changes, it publishes a group address and the applicable subscribed lights behave accordingly.

A provisioner is not required for the network to function. It is only required when nodes must be added or removed (blacklisted) from the network. Because of this, there is no central point of failure that could bring down the entire network.

6 Addressing

The term *group address* is simply an address value in the range 0xC000 to 0xFEFF. Consider it the *topic field* that is often mentioned in publish/subscribe communications schemes. For example, the provisioner arbitrarily decides that 0xC100 means meeting room, 0xC101 means storeroom, 0xC102 means the lobby, and so on. It configures the subscription addresses for the lights appropriately as one of those. There is no rulebook regarding how to assign those group addresses (in the range 0xC000 to 0xFEFF). The Bluetooth SIG does reserve some address values to allow messages to be sent to All-nodes (0xFFFF), All-relays (0xFFFE), All-friends (0xFFFD), and All-proxies (0xFFFC).

7 Encryption

By the time the messages are on the air, they have been **encrypted twice**. Once with an application key and the second time with a network key. Each key is 128-bit long and the encryption algorithm uses AES in CCM mode.

Because it does not have any knowledge of the keys, an outgoing message at the application layer is not encrypted. In addition, an application does not know or care about its own node address. When an application sends the message, the destination address is added and the encryption is performed by the underlying mesh stack.

Critical information such as node address, app keys, net keys, publication address, subscription list, key bindings, and other configuration information is wholly managed by the *smart*BASIC firmware. This information is only available after a provisioner provisions the device into a network.

8 References

Bluetooth Mesh – www.bluetooth.com/mesh

Bluetooth Mesh Specification v1.0 – https://www.bluetooth.com/specifications/mesh-specifications

About Bluetooth Mesh – https://blog.bluetooth.com/answers-to-your-questions-about-bluetooth-mesh

Ezurio Bluetooth Modules – https://www.ezurio .com/product-categories/connectivity-solutions/bluetooth-modules

9 Revision History

Version	Date	Notes	Contributor(s)	Approver
1.0	09 Jan 2018	Initial Release	Mark Duncombe	Jonathan Kaye
2.0	20 Mar 2025	Ezurio rebranding	Sue White	Dave Drogowski

Ezurio's products are subject to standard Terms & Conditions.