Bluetooth Mesh Networking

An Introduction for Developers.

Mesh is a new network topology option available for *Bluetooth*® Low Energy (LE) adopted in the summer of 2017. It represents a major advance which positions Bluetooth to be the dominant low power wireless communications technology in a wide variety of new sectors and use cases, including Smart Buildings and Industrial IoT.

Author: Martin Woolley

Version: 1.0.1

Revision Date: 2 December 2020

Bluetooth®

Revision History

Version	Date	Author	Changes
1.0	9 October 2020	Martin Woolley	Initial Version
1.0.1	2 December 2020	Martin Woolley	Language Updates

Table of Contents

1.0 Introduction 6

2.0 Taking Control 7

Smart Buildings Get Truly Smart 7

3.0 Bluetooth Mesh — The Basics 8

Concepts and Terminology 8

Mesh vs. Point-to-Point 8

Devices and Nodes 8

Elements 9

Messages 9

Addresses 10

Publish/Subscribe 11

States and Properties 11

Messages, States and Properties 12

State Transitions 13

Bound States 13

Models 13

Generics 14

Scenes 14

Provisioning 15

- Step 1. Beaconing 15
- Step 2. Invitation 15
- Step 3. Exchanging Public Keys 15
- Step 4. Authentication 15
- Step 5. Distribution of the Provisioning Data 15

Table of Contents

Features 16

Relay Nodes 16

Low Power Nodes and Friend Nodes 16

Proxy Nodes 17

Node Configuration 18

4.0 Mesh System Architecture 19

Overview 19

Bearer Layer 19

Network Layer 20

Lower Transport Layer 20

Upper Transport Layer 20

Access Layer 20

Foundation Models Layer 21

Models Layer 21

5.0 Security 22

Mesh Security is Mandatory 22

Mesh Security Fundamentals 22

Separation of Concerns and Mesh Security Keys 23

Node Removal, Key Refresh and Trashcan Attacks 24

Privacy 24

Replay Attacks 24

Table of Contents

6.0 Bluetooth Mesh in Action 25

Message Publication and Delivery 25

Multipath Delivery 25

Managed Flooding 25

- Heartbeats 25
- TTL 25
- Message Cache 26
- Friendship 26

Traversing the Stack 26

7.0 Bluetooth Mesh — New Frontiers 27

References 27

1.0 Introduction

Bluetooth has been actively developed since its initial release in 2000, when it was originally intended to act as a cable replacement technology. It soon came to dominate wireless audio products and computer peripherals such as wireless mice and keyboards.

In 2010, Bluetooth LE provided the next, major step forward. Its impact has been substantial and widely felt, most notably in smartphones and tablets, as well as in Health and Fitness, Smart Home and Wearables categories.

Wireless communications systems based around mesh network topologies have proved themselves to offer an effective approach to providing coverage of large areas, extending range and providing resilience. However, until now they have been based upon niche technologies, incompatible with most computer, smartphone and accessory devices owned by consumers or used in the enterprise.

120 Bluetooth SIG member companies participated in the work required to bring mesh networking support to Bluetooth. This is significantly more than is typically the case, and is representative of the demand for a global, industry standard for a Bluetooth mesh networking capability.

The addition of mesh networking support represents a change of a type, and of such magnitude that it warrants being described as a paradigm shift for Bluetooth technology.

2.0 Taking Control

Smart Buildings Get Truly Smart

Imagine arriving at the office in your car, early one dark, winter morning. The security system lets you in and a parking bay is automatically allocated to you. The bay number over the parking space lights up so you can drive easily to it. The parking bay allocation system is updated to note that this space is now occupied.

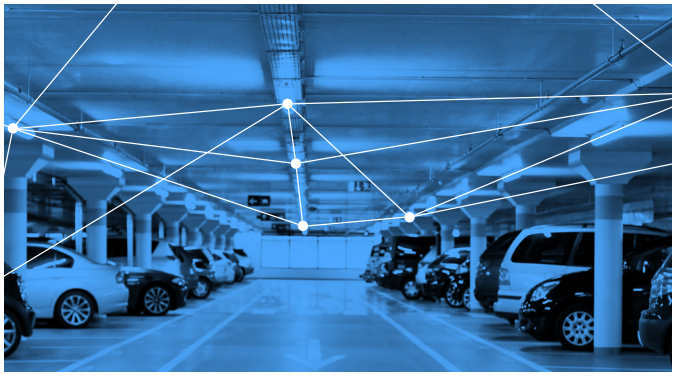

Figure 1 - A Bluetooth mesh network could span the office and car park

Entering the building, occupancy sensors note your arrival and identify you from the wearable technology about your person. You take the elevator to the 2nd floor and exit. You're the first to arrive, as usual. As the lift doors open, the lights from the elevator to your office and the kitchen come on. Coffee is deemed of strategic significance in your company! Other areas are left in darkness to save power.

You walk to your office and enter, closing the door behind you. The LED downlights and your desk lamp are already on and at exactly the level you prefer. You notice the temperature is a little warmer than the main office space, reflecting your personal preference. Proximity with your computer automatically logs you in.

Your day started well, with the building responding to your needs, taking into account your preferences. It's clear that systems are being used efficiently. What made this possible?

Your company installed a Bluetooth mesh network some months ago, starting with a mesh lighting system. Later the mesh was added to with occupancy sensors, environmental sensors, a wireless heating control system, and a mesh-based car park management system. The company is saving money on electricity and heating, and work environments have become personalized, boosting personal productivity. Maintenance costs are going down since adding items like additional light switches no longer requires expensive and disruptive physical wiring. Data is allowing the building management team to learn about the building, its services and how people act within it and are using this data to make optimizations.

The Bluetooth mesh network has made it easier and cheaper to be in control of building services, to wirelessly interact with them and to automate their behaviors. You wonder how you ever lived without such advanced building technology in the past!

3.0 Bluetooth Mesh — The Basics

Concepts and Terminology

Understanding Bluetooth mesh networking topology requires the reader to learn about a series of new technical terms and concepts, not found in the world of Bluetooth LE. In this section, we'll explore the most fundamental of these terms and concepts.

Mesh vs. Point-to-Point

Most Bluetooth LE devices communicate with each other using a simple point-to-point network topology enabling one-to-one device communications. In the Bluetooth core specification, this is called a 'piconet.'

Figure 2 - A many to many topology with message relaying

Imagine a smartphone that has established a point-to-point connection to a heart rate monitor over which it can transfer data. One nice aspect of Bluetooth is that it enables devices to set up multiple connections. That same smartphone can also establish a point-to-point connection with an activity tracker. In this case, the smartphone can communicate directly with each of the other devices, but the other devices cannot communicate directly with each other.

In contrast, a mesh network has a many-to-many topology, with each device able to communicate with every other device in the mesh (we'll examine that statement more closely later on in the section entitled "Bluetooth mesh in action"). Communication is achieved using messages, and devices are able to relay messages to other devices so that the end-to-end communication range is extended far beyond the radio range of each individual node.

Devices and Nodes

Devices which are part of a mesh network are called nodes and those which are not are called "unprovisioned devices".

The process which transforms an unprovisioned device into a node is called "provisioning". Consider purchasing a new Bluetooth light with mesh support, bringing it home and setting it up. To make it part of your mesh network, so that it can be controlled by your existing Bluetooth light switches and dimmers, you would need to provision it.

Provisioning is a secure procedure which results in an unprovisioned device possessing a series of encryption keys and being known to the Provisioner device,

typically a tablet or smartphone. One of these keys is called the network key or NetKey for short. You can read more about mesh security in the Security section, below.

All nodes in a mesh network possess at least one NetKey and it is possession of this key which makes a device a member of the corresponding network and as such, a node. There are other requirements that must be satisfied before a node can become a fundamental first step. We'll review the provisioning process in more detail in a later section of this paper.

Elements

Some nodes have multiple, constituent parts, each of which can be independently controlled. In Bluetooth mesh terminology, these parts are called elements. Figure 3 shows an LED lighting product which if added to a Bluetooth mesh network, would form a single node with three elements, one for each of the individual LED lights.

Figure 3 - Lighting node consisting of three elements

Messages

When a node needs to query the status of other nodes or needs to control other nodes in some way, it sends a message of a suitable type. If a node needs to report its status to other nodes, it sends a message. All communication in the mesh network is "message-oriented" and many message types are defined, each with its own, unique opcode.

Messages fall within one of two broad categories; acknowledged or unacknowledged.

Acknowledged messages require a response from nodes that receive them. The response serves two purposes: it confirms that the message it relates to was received, and it returns data relating to the message recipient to the message sender.

The sender of an acknowledged message may resend the message if it does not receive the expected response(s) and therefore, acknowledged messages must be idempotent. This means that the effect of a given acknowledged message, arriving at a node multiple times, will be the same as it had only been received once.

Unacknowledged messages do not require a response.

Addresses

Messages must be sent from and to an address. Bluetooth mesh defines three types of address.

A unicast address uniquely identifies a single element. Unicast addresses are assigned to devices during the provisioning process.

A group address is a multicast address which represents one or more elements. Group addresses are either defined by the Bluetooth Special Interest Group (SIG) and are known as SIG Fixed Group Addresses or are assigned dynamically. 4 SIG Fixed Group Addresses have been defined. These are named All-proxies, All-friends, All-relays and All-nodes. The terms Proxy, Friend, and Relay will be explained later in this paper.

It is expected that dynamic group addresses will be established by the user via a configuration application and that they will reflect the physical configuration of a building, such as defining group addresses which correspond to each room in the building.

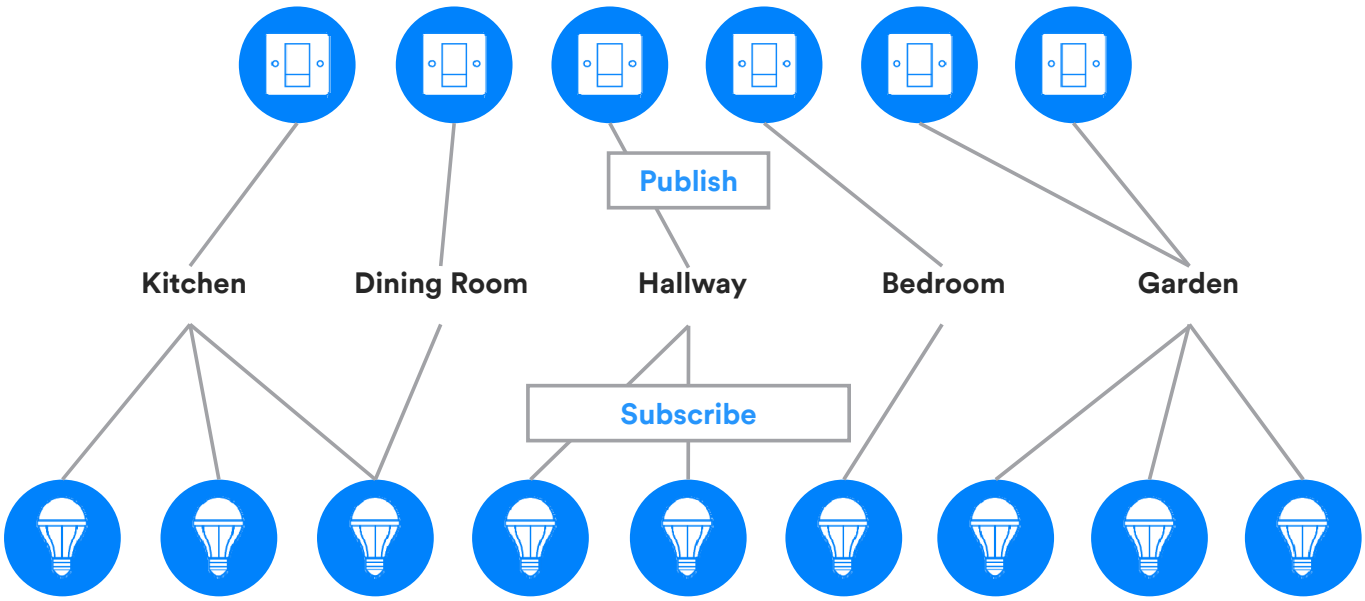

Figure 4 - Publish/Subscribe

A virtual address is an address which may be assigned to one or more elements, spanning one or more nodes. It takes the form of a 128-bit UUID value with which any element can be associated and is much like a label.

Virtual addresses will likely be preconfigured at the point of manufacture and be used for scenarios such as allowing the easy addressing of all meeting room projectors made by this manufacturer.

Publish/Subscribe

The act of sending a message is known as publishing. Nodes are configured to select messages sent to specific addresses for processing, and this is known as subscribing.

Typically, messages are addressed to group or virtual addresses. Group and virtual address names will have readily understood meaning to the end user, making them easy and intuitive to use.

In Figure 4, above, we can see that the node “Switch 1” is publishing to group address Kitchen. Nodes Light 1, Light 2, and Light 3 each subscribe to the Kitchen address and therefore receive and process messages published to this address. In other words, Light 1, Light 2, and Light 3 can be switched on or off using Switch 1.

Switch 2 publishes to the group address Dining Room. Light 3 alone subscribed to this address and so is the only light controlled by Switch 2. Note that this example also illustrates the fact that nodes may subscribe to messages addressed to more than one distinct address. This is both powerful and flexible.

Similarly, notice how both nodes Switch 5 and Switch 6 publish to the same Garden address.

The use of group and virtual addresses with the publish/subscribe communication model has an additional, substantial benefit in that removing, replacing or adding new nodes to the network does not require reconfiguration of other nodes. Consider what would be involved in installing an additional light in the dining room. The new device would be added to the network using the provisioning process and configured to subscribe to the Dining Room address. No other nodes would be affected by this change to the network. Switch 2 would continue to publish messages to Dining Room as before but now, both Light 3 and the new light would respond.

States and Properties

Elements can be in various conditions and this is represented in Bluetooth mesh by the concept of state values.

A state is a value of a certain type, contained within an element (within a server model - see below). As well as values, States also have associated behaviors and may not be reused in other contexts.

As an example, consider a simple light which may either be on or off. Bluetooth mesh defines a state called Generic OnOff. The light would possess this state item and a value of On would correspond to and cause the light to be illuminated whereas a Generic OnOff state value of Off would reflect and cause the light to be switched off.

The significance of the term Generic will be discussed later.

Properties are similar to states in that they contain values relating to an element. But they are significantly different to states in other ways. Readers who are familiar with Bluetooth LE will be aware of characteristics and recall that they are data types with no defined behaviors associated with them, making them reusable across different contexts. A property provides the context for interpreting a characteristic.

To appreciate the significance and use of contexts as they relate to properties, consider for example, the characteristic Temperature 8, an 8-bit temperature state type which has a number of associated properties, including Present Indoor Ambient Temperature and Present Outdoor Ambient Temperature. These two properties allow a sensor to publish sensor readings in a way that allows a receiving client to determine the context the temperature value has, making better sense of its true meaning.

Properties are organized into two categories: Manufacturer, which is a read-only category and Admin which allows read-write access.

Messages, States and Properties

Messages are the mechanism by which operations on the mesh are invoked. Formally, a given message type represents an operation on a state or collection of multiple state values. All messages are of three broad types, reflecting the types of operation which Bluetooth mesh supports. The shorthand for the three types is GET, SET and STATUS.

GET messages request the value of a given state from one or more nodes. A STATUS message is sent in response to a GET and contains the relevant state value.

SET messages change the value of a given state. An acknowledged SET message will result in a STATUS message being returned in response to the SET message whereas an unacknowledged SET message requires no response.

STATUS messages are sent in response to GET messages, acknowledged SET messages or independently of other messages, perhaps driven by a timer running on the element sending the message, for example.

Specific states referenced by messages are inferred from the message opcode. Properties on the other hand, are referenced explicitly in generic property related messages using a 16-bit property ID.

State Transitions

Changes from one state to another are called state transitions. Transitions may be instantaneous or execute over a period of time called the transition time. A state transition is likely to have an effect on the application layer behavior of a node.

Bound States

Relationships may exist between states whereby a change in one triggers a change in the other. Such a relationship is called a state binding. One state may be bound to multiple other states.

For example, consider a light controlled by a dimmer switch. The light would possess the two states, Generic OnOff and Generic Level with each bound to the other. Reducing the brightness of the light until Generic Level has a value of zero (fully dimmed) results in Generic OnOff transitioning from On to Off.

Models

Models pull the preceding concepts together and define some or all of the functionality of an element as it relates to the mesh network. Three categories of model are recognized.

A server model defines a collection of states, state transitions, state bindings and messages which the element containing the model may send or receive. It also defines behaviors relating to messages, states and state transitions.

A client model does not define any states. Instead, it defines the messages which it may send or receive in order to GET, SET or acquire the STATUS of states defined in the corresponding server model.

Control models contain both a server model, allowing communication with other client models and a client model which allows communication with server models.

Models may be created by extending other models. A model which is not extended is called a root model.

Models are immutable, meaning that they may not be changed by adding or removing behaviors. The correct and only permissible approach to implementing new model requirements is to extend the existing model.

Generics

It is recognized that many different types of device, often have semantically equivalent states, as exemplified by the simple idea of ON vs OFF. Consider lights, fans and power sockets, all of which can be switched on or turned off.

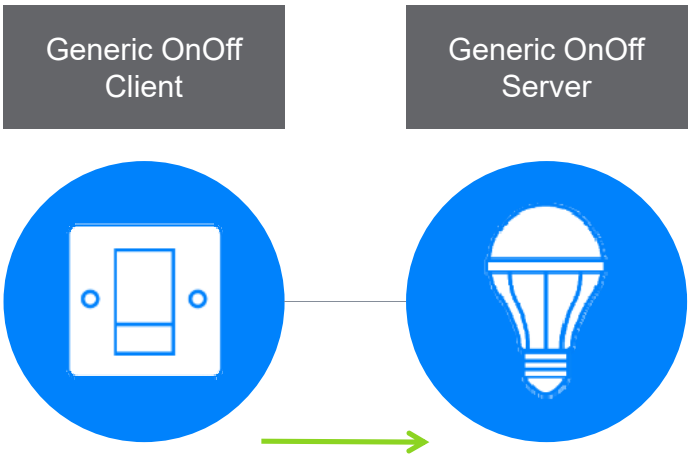

Figure 5 - Generic Models

Consequently, the Bluetooth mesh model specification, defines a series of reusable, generic states such as, for example, Generic OnOff and Generic Level.

Similarly, a series of generic messages that operate on the generic states are defined. Examples include Generic OnOff Get and Generic Level Set.

Generic states and generic messages are used in generalized models, both generic server models such as the Generic OnOff Server and Generic Client Models such as the Generic Level Client.

Generics allow a wide range of device type to support Bluetooth mesh without the need to create new models. Remember that models may be created by extending other models too. As such, generic models may form the basis for quickly creating models for new types of devices.

Scenes

A scene is a stored collection of states which may be recalled and made current by the receipt of a special type of message or at a specified time. Scenes are identified by a 16-bit Scene Number, which is unique within the mesh network.

Scenes allow a series of nodes to be set to a given set of previously stored, complimentary states in one coordinated action.

Imagine that in the evening, you like the temperature in your main family room to be 20 degrees Celsius, the six LED downlights to be at a certain brightness level and the lamp in the corner of the room on the table, set to a nice warm yellow hue. Having manually set the various nodes in this example scenario to these states, you can store them as a scene using a configuration application, and recall the scene later on, either on demand by sending an appropriate, scene-related mesh message or automatically at a scheduled time.

Provisioning

Provisioning is the process by which a device joins the mesh network and becomes a node. It involves several stages, results in various security keys being generated and is itself a secure process.

Provisioning is accomplished using an application on a device such as a tablet. In this capacity, the device used to drive the provisioning process is referred to as the Provisioner. The provisioning process progresses through five steps and these are described next.

Step 1. Beaconing
In support of various different Bluetooth mesh features, including but not limited to provisioning, new GAP AD types (ref: Bluetooth Core Specification Supplement) have been introduced, including the <<Mesh Beacon>> AD type.

An unprovisioned device indicates its availability to be provisioned by using the <<Mesh Beacon>> AD type in advertising packets. The user might need to start a new device advertising in this way by, for example, pressing a combination of buttons or holding down a button for a certain length of time.

Step 2. Invitation
In this step, the Provisioner sends an invitation to the device to be provisioned, in the form of a Provisioning Invite PDU. The Beaconing device responds with information about itself in a Provisioning Capabilities PDU.

Step 3. Exchanging Public Keys
The Provisioner and the device to be provisioned, exchange their public keys, which may be static or ephemeral, either directly or using an out-of-band (OOB) method.

Step 4. Authentication
During the authentication step, the device to be provisioned outputs a random, single or multi-digit number to the user in some form, using an action appropriate to its capabilities. For example, it might flash an LED several times. The user enters the digit(s) output by the new device into the Provisioner and a cryptographic exchange takes place between the two devices, involving the random number, to complete the authentication of each of the two devices to the other.

Step 5. Distribution of the Provisioning Data
After authentication has successfully completed, a session key is derived by each of the two devices from their private keys and the exchanged, peer public keys. The session key is then used to secure the subsequent distribution of the data required to complete the provisioning process, including a security key known as the network key (NetKey).

After provisioning has completed, the provisioned device possesses the network's NetKey, a mesh security parameter known as the IV Index and a Unicast Address, allocated by the Provisioner. It is now known as a node.

Features

All nodes can transmit and receive mesh messages but there are a number of optional features which a node may possess, giving it additional, special capabilities. There are four such optional features: the Relay, Proxy, Friend, and the Low Power features. A node may support zero or more of these optional features and any supported feature may, at a point in time, be enabled or disabled.

Relay Nodes

Nodes which support the Relay feature, known as Relay nodes, are able to retransmit received messages. Relaying is the mechanism by which a message can traverse the entire mesh network, making multiple "hops" between devices by being relayed.

Mesh network PDUs include a field called TTL (Time To Live). It takes an integer value and is used to limit the number of hops a message will make across the network. Setting TTL to 3, for example, will result in the message being relayed, a maximum number of three hops away from the originating node. Setting it to 0 will result in it not being relayed at all and only traveling a single hop. Armed with some basic knowledge of the topology and membership of the mesh, nodes can use the TTL field to make more efficient use of the mesh network.

Low Power Nodes and Friend Nodes

Some types of node have a limited power source and need to conserve energy as much as possible. Furthermore, devices of this type may be predominantly concerned with sending messages but still have a need to occasionally receive messages.

Consider a temperature sensor which is powered by a small coin cell battery. It sends a temperature reading once per minute whenever the temperate is above or below configured upper and lower thresholds. If the temperature stays within those thresholds it sends no messages. These behaviors are easily implemented with no particular issues relating to power consumption arising.

However, the user is also able to send messages to the sensor which change the temperature threshold state values. This is a relatively rare event but the sensor must support it. The need to receive messages has implications for duty cycle and as such power consumption. A 100% duty cycle would ensure that the sensor did not miss any temperature threshold configuration messages but use a prohibitive amount of power. A low duty cycle would conserve energy but risk the sensor missing configuration messages.

The answer to this apparent conundrum is the Friend node and the concept of friendship.

Nodes like the temperature sensor in the example may be designated Low Power nodes (LPNs) and a feature flag in the sensor's configuration data will designate the node as such.

LPNs work in tandem with another node, one which is not power-constrained (e.g. it has a permanent AC power source). This device is termed a Friend node. The Friend stores messages addressed to the LPN and delivers them to the LPN whenever the LPN polls the Friend node for "waiting messages". The LPN may poll the Friend relatively infrequently so that it can balance its need to conserve power with the timeliness with which it needs to receive and process configuration messages. When it does poll, all messages stored by the Friend are forwarded to the LPN, one after another, with a flag known as MD (More Data) indicating to the LPN whether there are further messages to be sent from the Friend node.

The relationship between the LPN and the Friend node is known as friendship. Friendship is key to allowing very power constrained nodes which need to receive messages, to function in a Bluetooth mesh network whilst continuing to operate in a power-efficient way.

Proxy Nodes

There are an enormous number of devices in the world that support Bluetooth LE, most smartphones and tablets being amongst them. In-market Bluetooth devices, at the time Bluetooth mesh was adopted, do not possess a Bluetooth mesh networking stack. They do possess a Bluetooth LE stack however and therefore have the ability to connect to other devices and interact with them using GATT, the Generic Attribute Profile.

Proxy nodes expose a GATT interface which Bluetooth LE devices may use to interact with a mesh network. A protocol called the Proxy Protocol, intended to be used with a connection-oriented bearer, such as GATT is defined. GATT devices read and write Proxy Protocol PDUs from within GATT characteristics implemented by the Proxy node. The Proxy node transforms these PDUs to/from mesh PDUs.

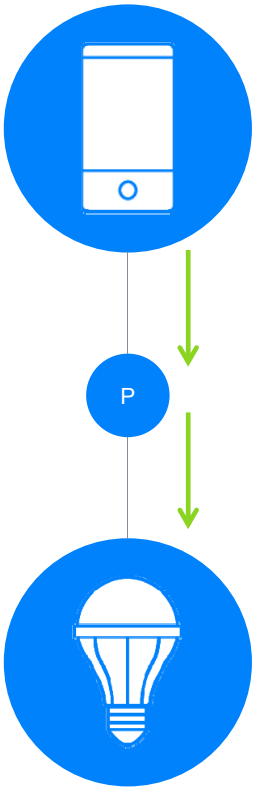

Figure 6 - Smartphone communicating via a mesh proxy node. P = proxy function on

In summary, Proxy nodes allow Bluetooth LE devices that do not possess a Bluetooth mesh stack to interact with nodes in a mesh network.

Node Configuration

Each node supports a standard set of configuration states which are implemented within the standard Configuration Server Model and accessed using the Configuration Client Model. Configuration State data is concerned with the node's capabilities and behavior within the mesh, independently of any specific application or device type behaviors.

For example, the features supported by a node, whether it is a Proxy node, a Relay node and so on, are indicated by Configuration Server states. The addresses to which a node has subscribed are stored in the Subscription List. The network and subnet keys indicating the networks the node is a member of are listed in the configuration block, as are the application keys held by the mode.

A series of configuration messages allow the Configuration Client Model and Configuration Server Model to support GET, SET and STATUS operations on the Configuration Server Model states.

4.0 Mesh System Architecture

Overview

In this section, we'll take a closer look at the Bluetooth mesh architecture, its layers and their respective responsibilities. We'll also position the mesh architecture relative to the Bluetooth LE core architecture.

At the bottom of the mesh architecture stack, we have a layer entitled Bluetooth LE. In fact, this is more than just a single layer of the mesh architecture, it's the full Bluetooth LE stack, which is required to provide fundamental wireless communications capabilities which are leveraged by the mesh architecture which sits on top of it. It should be clear that the mesh system is dependent upon the availability of a Bluetooth LE stack.

We'll now review each layer of the mesh architecture, working our way up from the bottom layer.

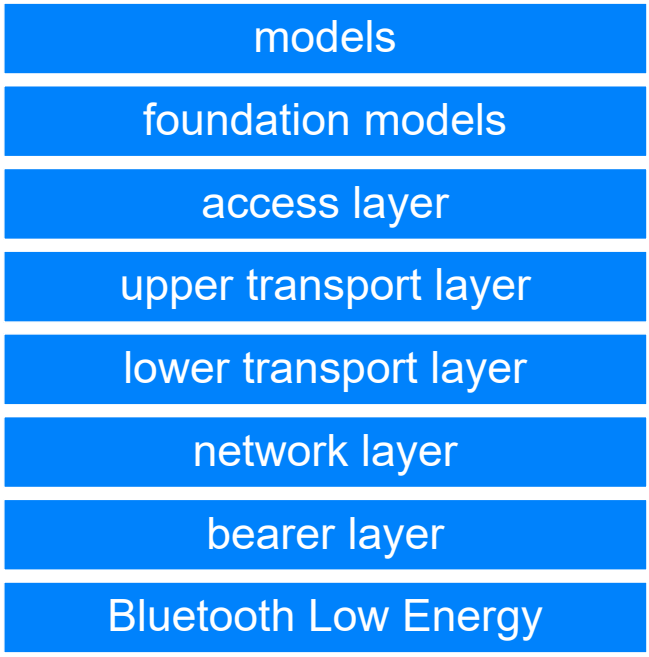

Figure 7 - The Bluetooth mesh architecture

Bearer Layer

Mesh messages require an underlying communications system for their transmission and receipt. The bearer layer defines how mesh PDUs will be handled by a given communications system. At this time, two bearers are defined and these are called the Advertising Bearer and the GATT Bearer.

The Advertising Bearer leverages Bluetooth LE's GAP advertising and scanning features to convey and receive mesh PDUs.

The GATT Bearer allows a device which does not support the Advertising Bearer to communicate indirectly with nodes of a mesh network which do, using a protocol known as the Proxy Protocol. The Proxy Protocol is encapsulated within GATT operations involving specially defined GATT characteristics. A mesh Proxy node implements these GATT characteristics and supports the GATT bearer as well as the Advertising Bearer so that it can convert and relay messages between the two types of bearer.

Network Layer

The network layer defines the various message address types and a network message format which allows transport layer PDUs to be transported by the bearer layer.

It can support multiple bearers, each of which may have multiple network interfaces, including the local interface which is used for communication between elements that are part of the same node.

The network layer determines which network interface(s) to output messages over. An input filter is applied to messages arriving from the bearer layer, to determine whether or not they should be delivered to the network layer for further processing. Output messages are subject to an output filter to control whether or not they are dropped or delivered to the bearer layer.

The Relay and Proxy features may be implemented by the network layer.

Lower Transport Layer

The lower transport layer takes PDUs from the upper transport layer and sends them to the lower transport layer on a peer device. Where required, it performs segmentation and reassembly of PDUs. For longer packets, which will not fit into a single Transport PDU, the lower transport layer will perform segmentation, splitting the PDU into multiple Transport PDUs. The receiving lower transport layer on the other device, will reassemble the segments into a single upper transport layer PDU and pass this up the stack.

Upper Transport Layer

The upper transport layer is responsible for the encryption, decryption and authentication of application data passing to and from the access layer. It also has responsibility for transport control messages, which are internally generated and sent between the upper transport layers on different peer nodes. These include messages related to friendship and heartbeats.

Access Layer

The access layer is responsible for defining how applications can make use of the upper transport layer. This includes:

- Defining the format of application data.
- Defining and controlling the encryption and decryption process which is performed in the upper transport layer.
    - Verifying that data received from the upper transport layer is for the right network and application, before forwarding the data up the stack.

Foundation Models Layer

The foundation model layer is responsible for the implementation of those models concerned with the configuration and management of a mesh network.

Models Layer

The model layer is concerned with the implementation of Models and as such, the implementation of behaviors, messages, states, state bindings and so on, as defined in one or more model specifications.Tuning connection parameters significantly are several steps required after waking up the Low Power Node.

5.0 Security

Mesh Security is Mandatory

Bluetooth LE allows the profile designer to exploit a range of different security mechanisms, from the various approaches to pairing that are possible, to individual security requirements associated with individual characteristics. Security is in fact, totally optional, and its permissible to have a device which is completely open, with no security protections or constraints in place. The device designer or manufacturer is responsible for analyzing threats and determining the security requirements and solutions for their product.

Figure 8 - security is central to Bluetooth mesh networking

In contrast, in Bluetooth mesh, security is mandatory. The network, individual applications and devices are all secure and this cannot be switched off or reduced in any way.

Mesh Security Fundamentals

The following fundamental security statements apply to all Bluetooth mesh networks:

1. All mesh messages are encrypted and authenticated.
2. Network security, application security and device security are addressed independently. See “Separation of Concerns” below.
3. Security keys can be changed during the life of the mesh network via a Key Refresh procedure.
4. Message obfuscation makes it difficult to track messages sent within the network providing a privacy mechanism to make it difficult to track nodes.
5. Mesh security protects the network against replay attacks.
6. The process by which devices are added to the mesh network to become nodes, is itself a secure process.
7. Nodes can be removed from the network securely, in a way which prevents trashcan attacks.

Separation of Concerns and Mesh Security Keys

At the heart of Bluetooth mesh security are three types of security keys. Between them, these keys provide security to different aspects of the mesh and achieve a critical capability in mesh security, that of “separation of concerns”.

To understand this and appreciate its significance, consider a mesh light which can act as a relay. In its capacity as a relay, it may find itself handling messages relating to the building’s Bluetooth mesh door and window security system. A light has no business being able to access and process the details of such messages but does need to relay them to other nodes.

To deal with this potential conflict of interest, the mesh uses different security keys for securing messages at the network layer from those used to secure data relating to specific applications such as lighting, physical security, heating and so on.

All nodes in a mesh network possess the **network key (NetKey)**. Indeed, it is possession of this shared key which makes a node a member of the network. A network encryption key and a Privacy Key are derived directly from the NetKey.

Being in possession of the NetKey allows a node to decrypt and authenticate up to the Network Layer so that network functions such as relaying, can be carried out. It does not allow application data to be decrypted.

The network may be sub-divided into subnets and each subnet has its own NetKey, which is possessed only by those nodes which are members of that subnet. This might be used, for example, to isolate specific, physical areas, such as each room in a hotel.

Application data for a specific application can only be decrypted by nodes which possess the right **application key (AppKey)**. Across the nodes in a mesh network, there may be many distinct AppKeys but typically, each AppKey will only be possessed by a small subset of the nodes, namely those of a type which can participate in a given application. For example, lights and light switches would possess the lighting application’s AppKey but not the AppKey for the heating system, which would only be possessed by thermostats, valves on radiators and so on.

AppKeys are used by the upper transport layer to decrypt and authenticate messages before passing them up to the access layer.

AppKeys are associated with only one NetKey. This association is termed “key binding” and means that specific applications, as defined by possession of a given AppKey, can only work on one specific network, whereas a network can host multiple, independently secure applications.

The final key type is the device key (DevKey). This is a special type of application key. Each node has a unique DevKey known to the Provisioner device and no other. The DevKey is used in the provisioning process to secure communication between the Provisioner and the node.

Node Removal, Key Refresh and Trashcan Attacks

As described above, nodes contain various mesh security keys. Should a node become faulty and need to be disposed of, or if the owner decides to sell the node to another owner, it's important that the device and the keys it contains cannot be used to mount an attack on the network the node was taken from.

A procedure for removing a node from a network is defined. The Provisioner application is used to add the node to a reject list and then a process called the Key Refresh Procedure is initiated.

The Key Refresh Procedure results in all nodes in the network, except for those which are members of the reject list from being issued with new network keys, application keys and all related, derived data. In other words, the entire set of security keys which form the basis for network and application security are replaced.

As such, the node which was removed from the network and which contains an old NetKey and an old set of AppKeys, is no longer a member of the network and poses no threat.

Privacy

A privacy key, derived from the NetKey is used to obfuscate network PDU header values, such as the source address. Obfuscation ensures that casual, passive eavesdropping cannot beused to track devices and the people that use them. It also makes attacks based upon traffic analysis difficult.

The degree of security offered in this technique is fit for purpose.

Replay Attacks

In network security, a replay attack is a technique whereby an eavesdropper intercepts and captures one or more messages and simply retransmits them later, with the goal of tricking the recipient, into carrying out something which the attacking device is not authorized to do. An example, commonly cited, is that of a car's keyless entry system being compromised by an attacker, intercepting the authentication sequence between the car's owner and the car, and later replaying those messages to gain entry to the car and steal it.

Bluetooth mesh has protection against replay attacks. The basis for this protection is the use of two network PDU fields called the Sequence Number (SEQ) and IV Index, respectively. Elements increment the SEQ value every time they publish a message. A node, receiving a message from an element which contains a SEQ value less than or equal to that which was in the last valid message, will discard it, since it is likely that it relates to a replay attack. IV Index is a separate field, considered alongside SEQ. IV Index values within messages from a given element must always be equal to or greater than the last valid message from that element.

6.0 Bluetooth Mesh in Action

Message Publication and Delivery

A network which uses Wi-Fi is based around a central network node called a router, and all network traffic passes through it. If the router is unavailable, the whole network becomes unavailable.

In contrast, Bluetooth mesh uses a technique known as managed flooding to deliver messages. Messages, when published by a node, are broadcast rather than being routed directly to one or more specific nodes. All nodes receive all messages from nodes that are in direct radio range and, if configured to do so, will then relay received messages. Relaying involves broadcasting the received message again, so that other nodes, more distant from the originating node, might receive the message broadcast.

Multipath Delivery

An important consequence of Bluetooth technology's use of managed flooding is that messages arrive at their destination via multiple paths through the network. This makes for a highly reliable network and it is the primary reason for having opted to use a flooding approach rather than routing in the design of Bluetooth mesh networking.

Managed Flooding

Bluetooth mesh networking leverages the strengths of the flooding approach and optimizes its operation such that it is both reliable and efficient. The measures which optimize the way flooding works in Bluetooth mesh networking are behind the use of the term "managed flooding". Those measures are as follows:

Heartbeats

Heartbeat messages are transmitted by nodes periodically. A heartbeat message indicates to other nodes in the network that the node sending the heartbeat is still active. In addition, heartbeat messages contain data which allows receiving nodes to determine how far away the sender is, in terms of the number of hops required to reach it. This knowledge can be exploited with the TTL field.

TTL

TTL (Time To Live) is a field which all Bluetooth mesh PDUs include. It controls the maximum number of hops, over which a message is relayed. Setting the TTL allows nodes to exercise control over relaying and conserve energy, by ensuring messages are not relayed further than is required.Heartbeat messages allow nodes to determine what the optimum TTL value should be for each message published.

Message Cache

A message cache must be implemented by all nodes. The cache contains all recently seen messages and if a message is found to be in the cache, indicating the node has seen and processed it before, it is immediately discarded.

Friendship

Probably the most significant optimization mechanism in a Bluetooth mesh network is provided by the combination of Friend nodes and Low Power nodes. As described, Friend nodes provide a message store and forward service to associated Low Power nodes. This allows Low Power nodes to operate in a highly energy-efficient manner.

Traversing the Stack

A node, receiving a message, passes it up the stack from the underlying Bluetooth LE stack, via the bearer layer to the network layer.

The network layer applies various checks to decide whether or not to pass the message higher up the stack or to discard it.

In addition, PDUs have a Network ID field, which provides a fast way to determine which NetKey the message was encrypted with. If the NetKey is not recognized by the network layer on the receiving node, this indicates it does not possess the corresponding NetKey, is not a member of that subnet and so the PDU is discarded. There's also a network message integrity check (MIC) field. If the MIC check fails, using the NetKey corresponding to the PDUs Network ID, then the message is discarded.

Messages, are received by all nodes in range of the node that sent the messages but many will be quickly discarded when it becomes apparent they are not relevant to this node due to the network or subnet(s) it belongs to.

The same principle is applied higher up the stack in the upper transport layer. Here though, the check is against the AppKey associated with the message, and identified by an application identifier (AID) field in the PDU. If the AID is unrecognized by this node, the PDU is discarded by the upper transport layer. If the transport message integrity check (TransMIC) fails, the message is discarded.

7.0 Bluetooth Mesh — New Frontiers

This paper should have provided the reader with an introduction to Bluetooth mesh, its key capabilities, concepts and terminology. It's Bluetooth but not as we know it. It's a Bluetooth technology that supports a new way for devices to communicate using a new topology.

Most of all, it's Bluetooth that makes this most pervasive of low power wireless technologies, a perfect fit for a whole new collection of use cases and industry sectors. ■

References

[1] Bluetooth SIG, Bluetooth Mesh Specification
See: www.bluetooth.com/specifications/adopted-specifications

[2] Bluetooth SIG, Bluetooth Mesh Model Specification
See: www.bluetooth.com/specifications/adopted-specifications

[3] Bluetooth SIG, Bluetooth 5 Core Specification
See: www.bluetooth.com/specifications/adopted-specifications

[4] Bluetooth SIG, Bluetooth Core Specification Supplement
See: www.bluetooth.com/specifications/adopted-specifications