Developer Study Guide: An introduction to Bluetooth Mesh Networking

Basic Theory

Release	:	2.1.0
Document Version:		2.1.0
Last updated	:	29th June 2021

Contents

Revision History

Version	Date	Author	Changes
1.0.0	15th June 2018	Martin Woolley Bluetooth SIG	Initial version.
1.0.4	14th December 2018	Martin Woolley Bluetooth SIG	Minor errata: #11 - Exercising the generic level state would work visibly even if the onoff state was OFF. This was not correct. Imagine a dimmer control which when pressed acts as an on/off switch. Rotating the knob will have no effect if the lights are switched off. That's how the generic onoff server and generic level server should work when incorporated together in a device. The two states are now handled completely independently. Code and documentation adjusted accordingly. #12 - Light node should have subscribed to the group address once for each model. A bug in Zephyr 1.12 allowed a subscription to only one model to be sufficient for all models to have messages published to that address routed to them so there was no user discernible impact of this issue at Zephyr 1.12. Code has been adjusted.
2.0.0	16th December 2019	Martin Woolley Bluetooth SIG	Study Guide release 2.0. No significant changes to this document.
2.0.1	21st December 2020	Martin Woolley Bluetooth SIG	Language changes
2.1.0	29th June 2021	Martin Woolley Bluetooth SIG	**Release:** Zephyr code now based on version 2.6.0 of the Zephyr SDK **Document**: No changes

Introduction

This document contains a study exercise. Read exercise 1 and make sure you're happy with the concepts explained and the terminology introduced. It is assumed that you have no prior knowledge of Bluetooth mesh.

It is not the intention that all aspects of Bluetooth mesh be covered in this document nor that we go into detail on every point. The theory covered here should give you enough knowledge to proceed to the hands-on exercises, starting in part 3.

The aim is to give you enough of a theoretical grounding to be able to understand and complete the hands-on exercises, with achieving practical results more important than having a comprehensive but purely theoretical knowledge of the subject.

Those readers wishing to read more about Bluetooth mesh are directed to the Bluetooth mesh technology overview, the Bluetooth mesh models technical overview and to the mesh profile and mesh models specifications.

Bluetooth mesh contains a fairly significant amount of terminology, much of which may be unfamiliar to you. It will be necessary to use this terminology as we progress through the various parts of this resource. If you start to lose track of what the various terms mean you may find it quickest and easiest to consult the Bluetooth Mesh Glossary of Terms.

Exercise 1

Bluetooth Mesh Essentials

Fundamental Concepts

A Bluetooth mesh network consists of a collection of specially prepared Bluetooth devices which we refer to as *nodes*. There can be many thousands of nodes in a mesh network and they communicate with each other by sending messages.

Communication between nodes can take place substantially beyond the direct range of the underlying Bluetooth radio because messages may hop from node to node through a process known as *relaying* until they reach their eventual destination. This is known as *multi-hop* message delivery.

Copies of messages typically travel to their destination via multiple paths through the mesh. This is a strategy designed to make Bluetooth mesh networks very reliable, in the face of the various challenges to radio communication that might exist in an environment, including physical barriers like walls. This is known as *multi-path* message propagation.

In principle, a node can communicate with many other nodes in a mesh network and may itself be communicated with by many other nodes. The topology of a mesh network is, therefore, said to be *many to many*.

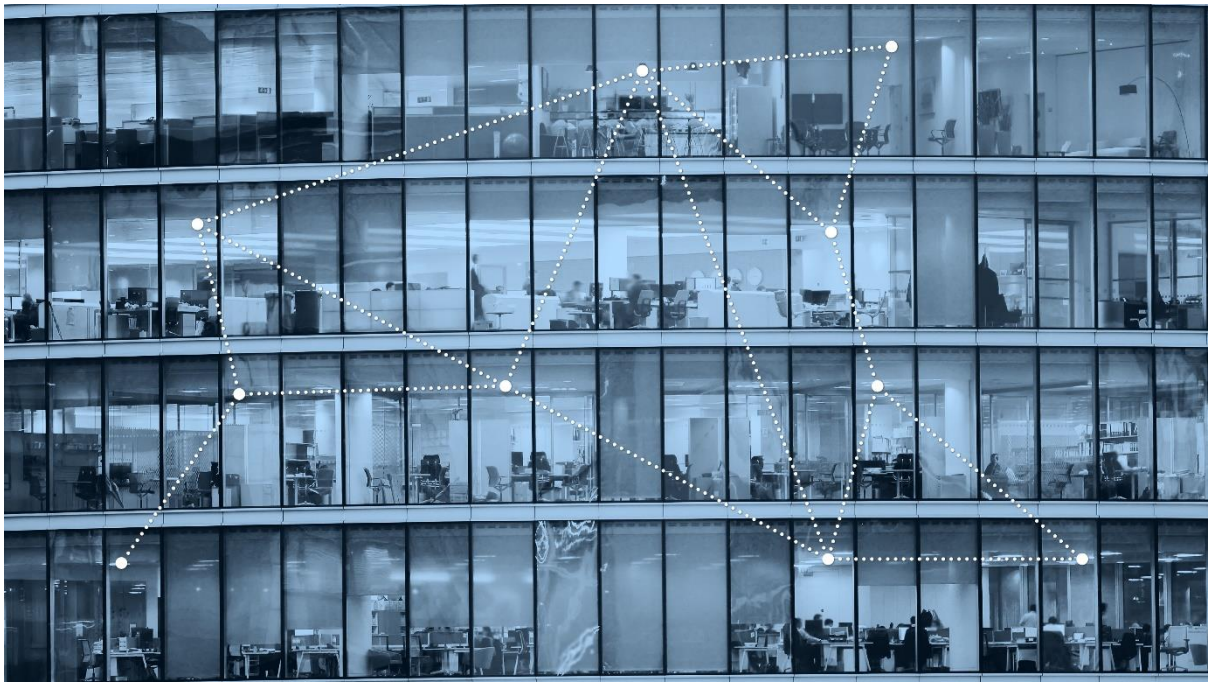


Figure 1 - mesh networks form many to many topologies

Flavours of Bluetooth

There are two distinct types of Bluetooth radio.

Bluetooth BR/EDR is the original version of Bluetooth and is designed for device to device data streaming applications like wireless audio.

Bluetooth LE is designed for very low power operation involving small amounts of data and is ideal for use when controlling other devices, monitoring the state of devices and for broadcasting limited quantities of information.

You can read more about Bluetooth radio versions at the bluetooth.com web site.

Bluetooth mesh is a networking stack not a radio technology. The Bluetooth mesh stack sits on top of Bluetooth LE. Bluetooth LE provides the underlying radio communications capability used by Bluetooth mesh.

Between them, the various Bluetooth technologies can support different communication topologies.

Bluetooth BR/EDR supports one or more point-to-point (1:1) connections between pairs of devices and can be thought of as a wireless replacement for physical connection cables.

Bluetooth LE also supports point-to-point communication but in addition, has the ability to broadcast data. This is connectionless communication and any suitable LE receiver within range can receive and process data broadcast in this way. This is a one-to-many (1:m) topology.

Bluetooth mesh allows many-to-many (m:n) communication.

You can read more about Bluetooth topology options at the bluetooth.com web site.

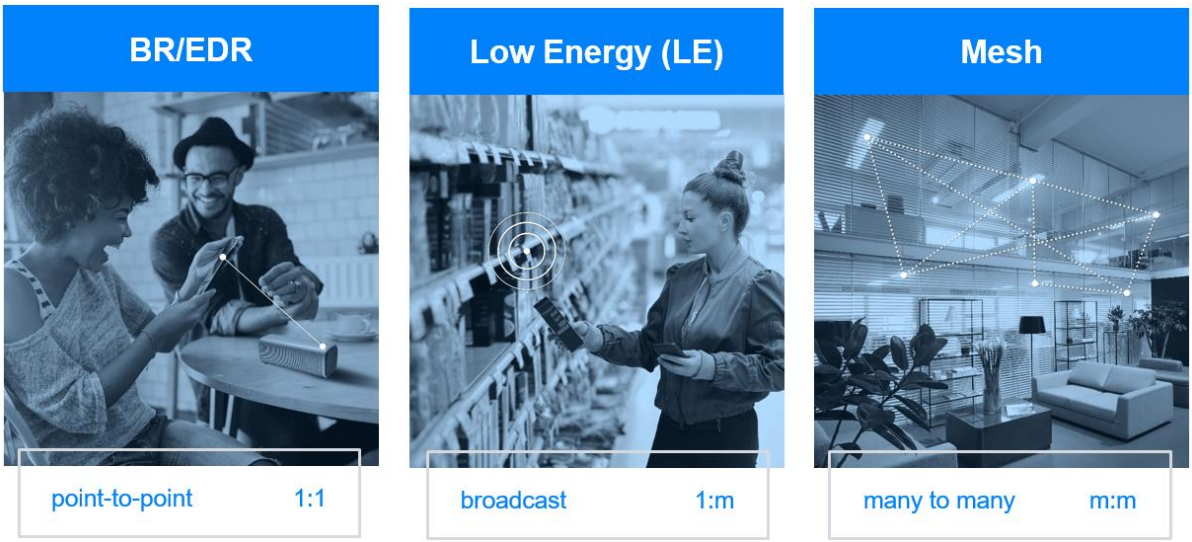


Figure 2 - Bluetooth technologies and topologies

Provisioning and Configuration

In the Fundamental Concepts section, nodes were described as *specially prepared* Bluetooth devices. This was referring to the fact that for a device to have the ability and the right to communicate in a particular mesh network, it must first go through a preparatory security procedure known as *provisioning*. A new device which is intended to be incorporated in a mesh network is initially referred to as an *unprovisioned device*. The provisioning procedure is carried out manually by a network administrator (which in the case of a domestic smart home would just be a householder) using a smartphone or tablet and it equips the unprovisioned device with a series of security keys. Having been provisioned, the device is now referred to as a *node* and it has the ability to communicate with other nodes in the network. Provisioning has made the device a *member* of this particular mesh network.

After provisioning has completed, typically the new node would also be configured in various ways appropriate to the type of product that it is. A Bluetooth light switch would be configured so that it controlled the required set of Bluetooth lights in the building, for example. Configuration is often accomplished using the same smartphone application with which provisioning was performed.

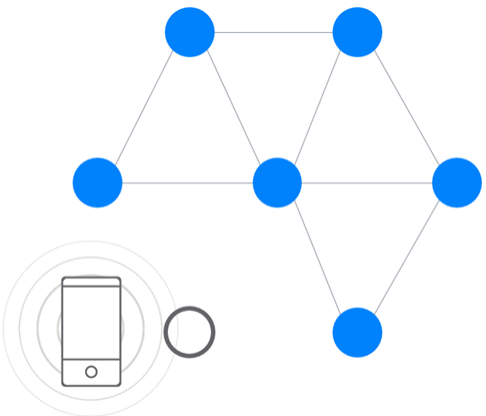


Figure 3 - provisioning a new node with a smartphone

Nodes and Elements

Nodes are devices which have been provisioned and are members of a specific mesh network identified by one of the keys provided to it by the provisioning process. Some device types are more complicated than others and may consist of multiple parts, each individually addressable but typically housed inside the same physical unit. Some LED lighting units are like this, with several individual LED lamps, each of which can be independently controlled. Bluetooth mesh defines the concept of an *element* to correspond to the individual, addressable parts of an overall node. So in the example, the LED lighting unit would be the node and each individual lamp would be an element. All nodes contain at least one element. One of the elements is designated the *primary element* and contains configuration data which is applicable to the whole node.

The terms *node* and *element* will be used interchangaebly unless it's important to be clear that we're talking about a specific element within a node.

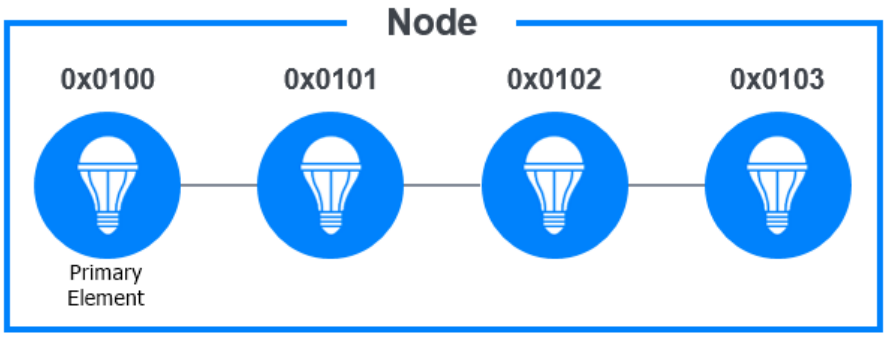


Figure 4 - A node consisting of 4 elements

Addresses

Bluetooth mesh defines an addressing scheme which allows both individual nodes to be identified and addressed and collections of nodes. There are three types of address:

Unicast Addresses uniquely identify individual nodes (strictly speaking they identify specific elements). There are 32,767 16-bit available unicast addresses in any network and therefore this is the maximum number of elements that a network may contain.

Group Addresses - identify collections or sets of devices and often correspond to certain types of device in a specific part of a building, such as all of the lights in a particular room. A group address is one of two types of *multicast address* supported by Bluetooth mesh. 16,384 16-bit group addresses are available per network.

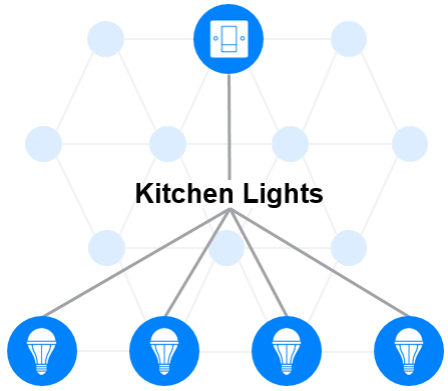


Figure 5 - example group address

Virtual Addresses - also identify collections of devices and is the other type of multicast address. Virtual addresses work differently to group addresses. A virtual address is a 16-bit value which maps to a 128-bit UUID. Its first two bits identify it as a virtual address and the other 14 bits contain a hash of the 128-bit UUID which the virtual address represents. The full 128-bit UUID is used in the calculation of a message integrity check field which all messages include but the 128-bit value itself is not transmitted, only its 16-bit virtual address. On receiving a message containing a virtual address, the stack will identify those UUIDs it knows about whose hash matches that found in the virtual address of the received message. There could be several matches and so the final match is established by calculating and comparing with the message integrity code in the message.

Virtual addresses are more complicated and less efficient to use in terms of run-time processing than group addresses but the 128-bit UUID which virtual addresses map to means that there are trillions of virtual addresses available to a network.

Messages and State

Nodes in a mesh network communicate using messages. The mesh specifications define two high level classes of message.

Control messages are concerned with the operation of the network itself and include messages such as heartbeats which indicate that a particular node is present, alive and a certain number of hops away.

Access messages relate to application functionality and a series of standard access message types are defined, each with an identifying opcode. Access messages can modify the state of nodes using *SET* messages, often resulting in a change which users can perceive. A mesh light switch might send a message to a group of lights and on receiving it, each of those lights might switch on, for example. Access messages can also request and retrieve the current state of a node using a GET message to which the remote node will reply with a *STATUS* message. Nodes can themselves announce their state by sending *STATUS* messages.

There are two variants of the SET message. *Acknowledged set messages* are responded to with a *STATUS* message containing the new state value whereas *unacknowledged set messages* are not replied to.

Messages are the fundamental mechanism by which things are made to happen in a mesh network and a mesh product developer will find message generation and handling central to their software development work.

State is a formal concept in Bluetooth mesh and nodes will often contain one or more items of state data (known as a *state*), linked in some way to a condition or aspect of the node. Specific states are defined in the Mesh Models specification.

Messages always have both a source address and a destination address. Source addresses are always unicast addresses. Destination addresses can be of any of the three address types. In practice though, unicast destination addresses are only used when configuring a node. Group addresses are most commonly used, even for cases where the set of nodes being addressed only has one member.

When a node sends a message, it is said to *publish* that message to an address. When nodes are initially configured, the group and virtual addresses in messages that they should listen for are specified. This is known as *subscribing*. Subscribing a node to a given group address is like making it a member of that group. Note that strictly speaking, it is a model (see below) on a node which subscribes to an address.

The publish/subscribe approach to messaging used by Bluetooth mesh decouples devices which send messages from those which receive them. A node publishing a message say, to a group address does not know the identities of those nodes which have subscribed to that address and will therefore receive them. Similarly, a node which has subscribed to a particular group address does not know anything about the nodes which might publish messages to that address. The benefit of this approach is that changes like adding new lights or switches to the network can be made without impacting the configuration of existing devices, such as requiring that switches are reconfigured just because an extra light has been added to a room.

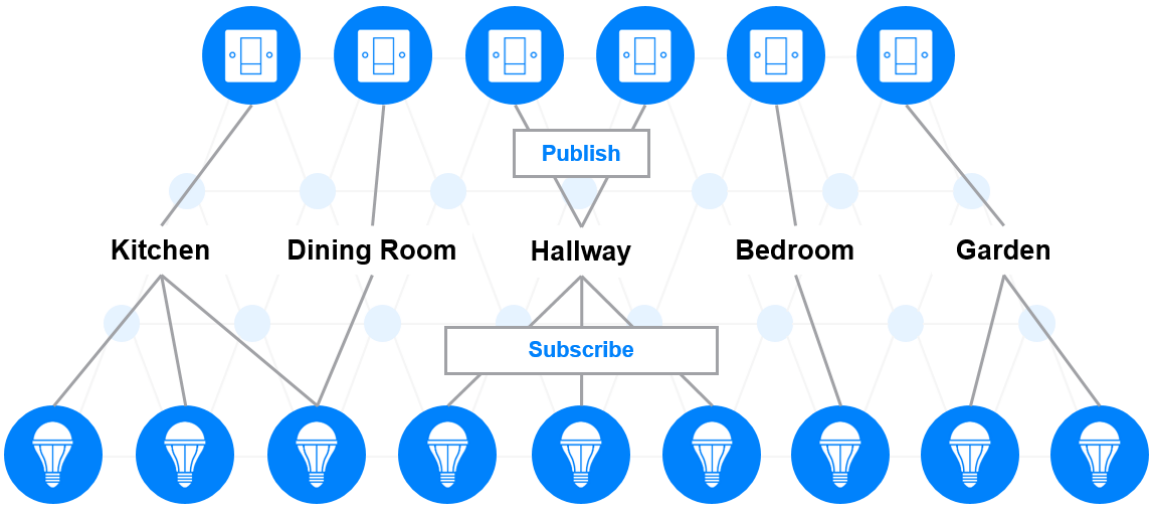


Figure 6 - publish / subscribe

Properties

States are data items which *server models* contain and which are operated on by messages. We'll learn about *models* in the next section. The particular state whose value is obtained, reported on or updated by a message is inferred from the message type. The message itself does not contain a state identifier. So for example, it is part of the specification of the *Generic OnOff Set Unacknowledged* message that its value must be used to update the Generic OnOff state in the Generic OnOff Server Model in the destination element.

Properties provide an alternative way of representing data items. They differ from states in that they include an explicit, associated 16-bit ID which indicates the property type as well as a value. Properties are associated with Bluetooth characteristics which are also used with Bluetooth GATT. The Mesh Device Properties specification lists all defined properties and indicates the associated characteristic and it is this characteristic which indicates the property value's format and representation.

Properties make it easy for models to handle a broad range of different types of data, which would otherwise require the model to include a large list of optional states. For some models, like the Sensor Server model using properties makes the most sense. Using properties with the Sensor Server means that a single model definition is applicable to any (more or less) type of sensor.

Models

Pulling together the concepts of state, message types, state changes and their effect on a node or element is the concept of the mesh *model*. Nodes contain one or more elements and each element contains one or more models.

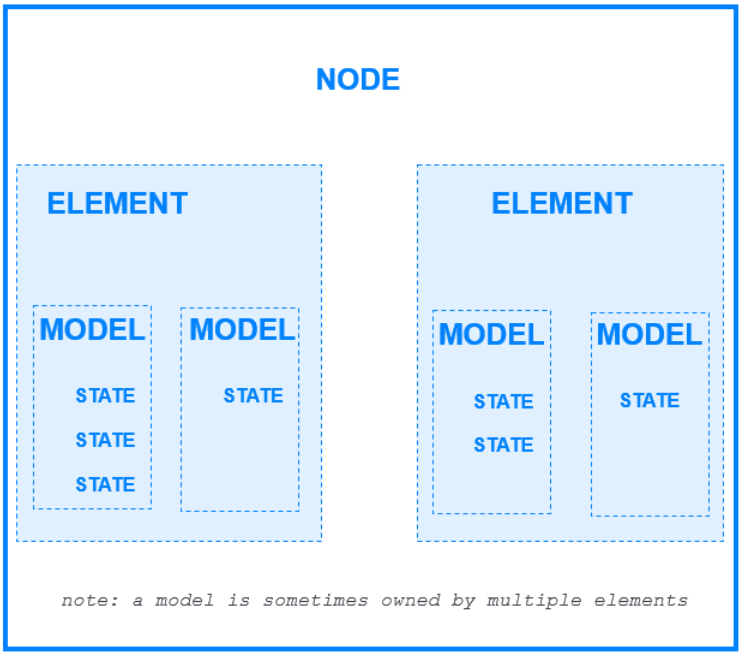


Figure 7 - Node Composition

Models can be thought of as standardised software components which make a node able to exhibit certain behaviours. The mesh profile specification and the mesh models specification define the set of standard models that have been defined, including a set of models which could be used with any type of product known as the *generics*.

The *Generic On/Off Server* model allows a node which implements it to be switched on or off by another node which contains the *Generic On/Off Client* model by it sending a *Generic OnOff Set* message to an address which an instance of the server model in a node subscribes to. A light switch would include the Generic OnOff Client model whereas a light would contain the Generic OnOff Server model in this example.

The Generic Level Server model allows some aspect of a node to be reduced or increased as directed by messages published by a Generic Level Client. You can easily imagine this being used to turn a

thermostat up or down, for example. You might imagine it also being used to dim or brighten lights as well. These models could be used for this purpose but there are a set of models designed specifically for lighting, which has some surprisingly sophisticated requirements.

Models can extend one or more other models meaning that more specialised models can be created, based on and inheritting the basic capabilities of other models. The Light Lightness Server extends the Generic Power OnOff Server and the Generic Level Server models. It allows lights to be dimmed or brightened but takes into account the fact that human perception of brightness is non-linear. This comes from the "Weber-Fechner rule, which says the perceived lightness is proportional to a square root of actual intensity measured with an accurate non-human instrument"[i].

Server models contain state items whereas clients do not. The Generic On/Off Server model contains the *Generic OnOff* state, for example. There's a 3rd type of model defined in the mesh profile specification called the *control model* which contains both client and server model functionality plus control logic. Control models are used to implement sophisticated and highly configurable control behaviours between models. Commercial lighting is an example of a place where such requirements are encountered and a special control model called Light LC is defined for this purpose.

Mandatory Models

Special models called the Configuration Server Model and the Health Server Model must be supported by the *primary element* on a node. The configuration server model contains state values which constitute the configuration of the node and the health server model is concerned with device diagnostics.

The Relationship Between Elements and Models

An element contains one or more models and typically a model will service a single owning element.

But it's also possible for a model to service more than one element. For example, lights whose colour can be controlled may consist of three independent elements, the first controlling *hue (H)*, the second controlling *saturation (S)* and the third controlling *lightness (L)*. Element H might subscribe to a group called *Office Hue*, element S to *Office Saturation* and L to *Office Lightness*. Independent rotary controls may now use Generic Level messages sent to each of these three group addresses to control the hue, saturation and lightness of the office lights, independently.

A state called the *Subscription List* contains the list of group addresses and virtual address UUIDs which have been subscribed to. There is one instance of this state in a node, for each combination of element and model. In other words, Subscription List sits at the intersection in a many-to-many logical relationship between elements and models.

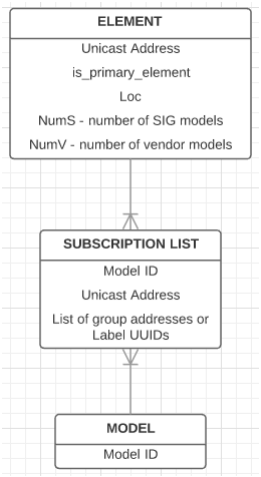


Figure 8 - the many to many relationship between Element and Model

Node Composition and Configuration

Every *primary element* in every node must include a model called the *Configuration Server Model* and this model contains states which collectively form the node's configuration. The configuration server model cannot be associated with an element that is not the primary element and it is the inclusion of the configuration server model that is one way in which the primary element is special and different to other elements that a node might have.

There are 17 different states defined for the configuration server model. Examples include states which contain lists of security keys (*NetKey List* and *AppKey List*) and states which indicate whether or not the node can take on special network roles such as acting as a relay which retransmits all the messages it receives so that they can hop across the network.

The *Composition Data* state is particularly interesting. Amongst other things, it contains fields which define the composition of the node in terms of elements it contains and, for each element, the models they each support.

Security

Bluetooth mesh has a series of security features. All messages are both encrypted and authenticated and every message includes a unique sequence number which is used to safeguard against replay attacks, for example. Security keys or material derived from them are used in many of the security features.

The provisioning process equips a new node with a network key *(NetKey)*. The configuration process which follows provisioning goes on to equip the new node with one or more application keys *(AppKey)* and its own key known as the device key *(DevKey)*.

The network key secures network-related data and application keys secure data from higher up the stack. Networks can be divided into subnets, each with its own key and distinct applications such as heating, lighting and air conditioning can each have their own application keys.

The separation of network security from application security means that any type of node can relay any type of message safely, without being given unnecessary access to application layer data. Relaying is a network operation and the data involved is secured by the network key which will allow network data to be decrypted, leaving higher layer data still encrypted.

An obfuscation process, which encrypts message source addresses, sequence numbers and other network information provides a privacy mechanism to protect against network traffic analysis attacks.

Several industry standard cryptography algorithms are used. For key derivation purposes, AES-CMAC is used. The obfuscation process uses the AES-ECB block cipher mode and message encryption and authentication uses the AES-CCM block cipher mode.

Application Key Binding

Every model must have an application key associated with it through a process called *Application Key Binding*. This allows the model to decrypt and authenticate access messages which are routed to it. Each application key is also associated with a single network key so that it can only be used within that network.

Referencing Keys

The set of NetKey and AppKey values possessed by a node reside in states called the NetKey List and AppKey List respectively. Provisioning and configuration operations reference keys in these lists using index values called the NetKey Index and AppKey Index.

See section *4.3.1.1 Key indexes* of the mesh profile specification.

SEQ and IV Index

Bluetooth mesh security includes protection against *replay attacks*. This is accomplished through two fields, SEQ and IV Index, which taken together provide a very large set of nonce values (see https://csrc.nist.gov/glossary/term/nonce). SEQ is present in all messages. Up to two IV Index values, may be active in the network at any one time and messages reference the relevant IV Index value to use in combination with SEQ using a bit field called the IVI field. Every message sent must use a combined IV Index/SEQ value larger than previous messages from that node. Messages whose IV Index/SEQ value fails that test are ignored as suspected replay attacks.

See sections *3.8.3 Sequence number* and *3.8.4 IV Index* of the mesh profile specification.

TTL

TTL stands for *Time To Live* and limits the number of hops a message will take across the network. This can help make use of the network more efficient since there would be no point having an on/off message from a light switch make 20 hops across the network to the top floor of an office block if all the lights it controls are in direct range in the same room as the switch. All mesh access messages include a TTL value.

Transactions

Bluetooth mesh supports the concept of a *transaction*. A transaction is a logically related, series of state changes, instigated by a sequence of messages. Models which support transactions define their transactional behaviour. A message which may be part of a transaction must include an 8-bit field called the Transaction Identifier (TID) which is incremented (and wrapped around at 255) for each transaction.

Mesh Protocols and PDUs

The Bluetooth mesh profile specification defines a number of protocols and their associated Protocol Data Units (PDU)s. These include network PDUs, mesh beacons and provisioning PDUs. Network PDUs contain messages published by a node and are the most common PDU to be encountered within the mesh therefore.

Types of Node

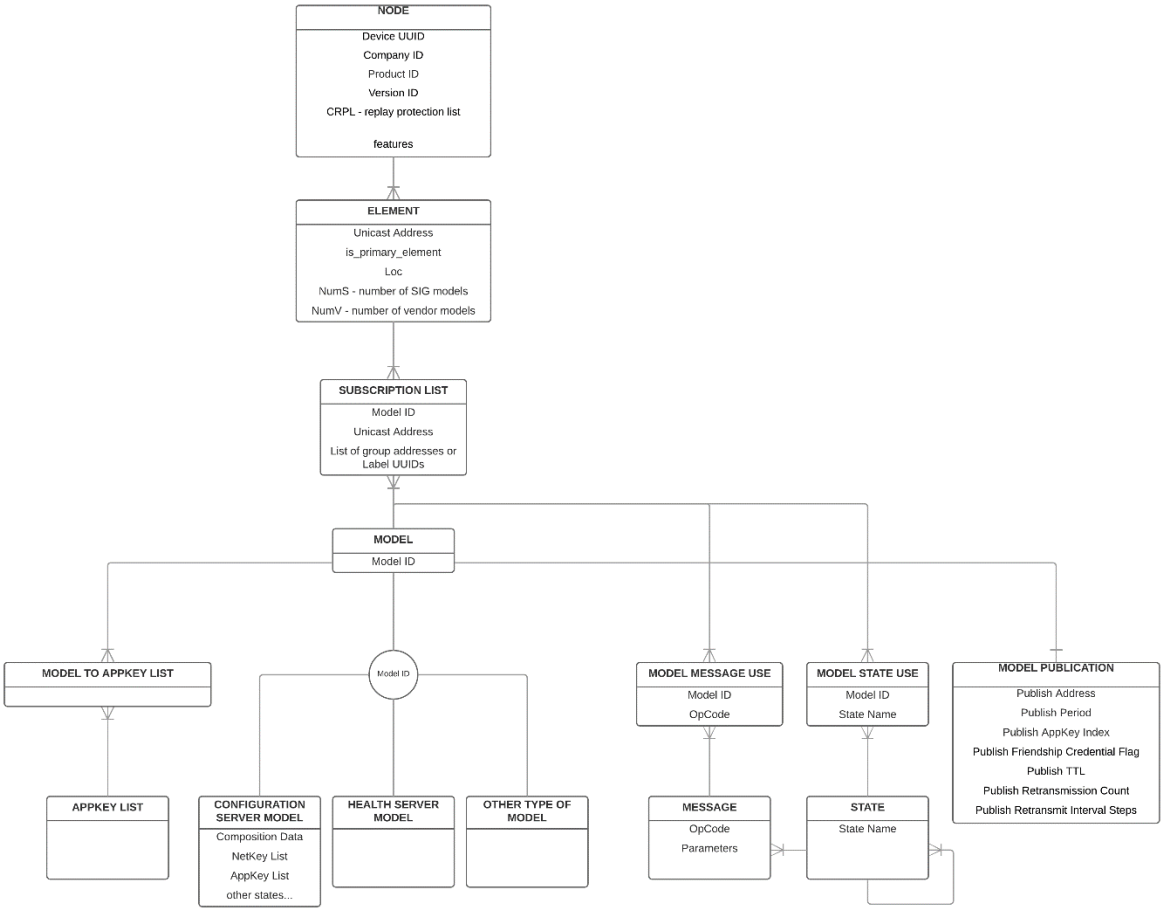
Nodes may have special software features, of which four are defined, enabling them to offer various network services. Those features are as follows:

> ***Relay***: A node with the relay feature will retransmit messages it receives, contributing to the multi-hop journey across the network a message can take.
>
> ***Friend***: This feature is typically to be found enabled in nodes which are *power rich*. A friend node provides a message store-and-forward service for associated *Low Power Nodes* which ask the friend for undelivered messages it has in its store, quite infrequently to save power.
>
> ***Low Power***: Low power nodes (LPN) usually need to only occasionally receive messages, are typically battery operated and may be installed in hard to access places. Consequently, conserving energy is essential. Rather than activate the radio receiver frequently just in case a rare message is there to be received, LPNs form a relationship called *friendship* with a nearby friend node so that the LPN need only activate its receiver very infrequently when it explicitly asks the friend node if it has any messages awaiting delivery for it.
>
> ***Proxy***: Proxy nodes make it possible for non-mesh Bluetooth LE devices like smartphones and tablets to securely interact with the mesh network using application software and the platform's existing Bluetooth LE APIs. In other words, many existing Bluetooth LE devices can become part of a mesh network without the need for their operating systems or APIs to be upgraded to support the full Bluetooth mesh protocol stack.

Concepts and their Relationships

When implementing mesh firmware, it's useful if not essential to understand the relationships between the various concepts since ultimately this will determine things like the data structures you use in your code. The API you're working with may provide much of this for you so that you can't go wrong but it's still a good idea to have a good mental image such as, for example, how a model relates to an element. To that end, the following incomplete Entity Relationship Diagram (ERD) may be useful. It doesn't show every mesh entity or relationship but it should encapsulate those concepts that are most important for the purposes of this study guide.

If you're unfamiliar with ERDs, each box is an entity or in plain language, *thing* and each line shows a relationship. The *crows feet* at one end of a line indicates that there may be more than one instance of the entity type at that end of the line for each instance of the type of entity at the other end of the line. Use your favourite search engine to seach for "introduction to entity relationship diagrams" and you'll find a number of suitable texts on the subject.



Next

Armed with the foundational knowledge that exercise 1 has given you, you're now ready to proceed to part 3, where you'll develop the firmware for a number of Bluetooth mesh nodes. You'll find part 3 in the document called Bluetooth Mesh Developer Study Guide - 3. Coding Exercises Introduction Vm.n.docx.

---

[i] Mesh Model Specification Appendix A.2 Light in Numbers