Designing Bluetooth Low Energy Smart Applications with Bluetooth Mesh - Part 1

Contributed By Digi-Key's North American Editors
2018-03-21

Editor's Note: Part 1 of this two-part series details the architecture and capabilities of the Bluetooth mesh 1.0 protocol. The protocol has been introduced as a complementary stack to Bluetooth low energy firmware and brings open-standard mesh networking to Bluetooth low energy for the first time. In addition, this article details the strengths and weaknesses of Bluetooth mesh, allowing designers to compare it with other low power wireless technology alternatives to determine if it is a good match for their wireless applications. Part 2 will describe how to integrate Bluetooth mesh into Bluetooth low energy designs using ICs and development kits available from Digi-Key suppliers.

Mesh networking is a key requirement for low-power wireless technologies targeting smart home and smart industry applications because it overcomes range limitations, eases scalability and builds in robustness. However, designers looking to use Bluetooth low energy have until recently been frustrated by its lack of mesh networking support.

This lack of support has put them in the position of having to opt for other technologies such as Zigbee and Thread for smart home applications, even though Bluetooth low energy may otherwise be perfectly suitable and widely supported. (See Digi-Key article "Comparing Low-Power Wireless Technologies".)

The Bluetooth SIG has now addressed this weakness with the introduction of a supplementary specification, Bluetooth mesh 1.0. The specification requires no additional hardware and can run on all Bluetooth low energy chips (v4.0, 4.1, 4.2, and 5). Some vendors are already supporting Bluetooth mesh 1.0 with their own firmware implementations of the specification and associated development tools.

However, before embarking on mesh network designs adopting the new specification, designers should become familiar with how the Bluetooth SIG's implementation of mesh differs from alternative technologies. For example, the trade-off concerning simplicity, power consumption and flexibility because the differences influence the design choices and process.

This article takes the specification and explains the architecture of Bluetooth mesh for designers. It describes its operational characteristics and describes how it supports smart home and applications such as smart lighting. The article finishes by introducing some suitable Bluetooth mesh design tools and supporting hardware and software solutions.

The advantages of mesh networking

Bluetooth low energy was initially designed to complement "classic" Bluetooth by extending the wireless technology to peripheral devices with modest battery capacity. Examples of peripherals are sports sensors such as heart rate belts or wirelessly controlled toys. Each peripheral communicates across an independent channel with a central supervisory device, such as a smartphone, forming a star network topology.

Partly due to its interoperability with smartphones, Bluetooth low energy quickly expanded into other sectors including the smart home for applications such as lighting control. In these types of applications, the drawbacks of star networks quickly became apparent.

For example, Bluetooth low energy solutions can only cope with a limited number of simultaneous connections (typically eight). Lighting installations with greater than that number of bulbs can't be controlled with a single command, introducing latency. Second, in a large house, bulbs in distant locations might be out of range of the central controller.

In a mesh network, instead of a central device communicating with individual peripheral devices, a message is relayed from one point in the network to any other by hopping across bi-directional channels connecting multiple nodes. In this way, mesh networking brings notable advantages because it allows simultaneous control over dozens of connected devices, overcomes range limitations, and builds in redundancy. (Figure 1.)

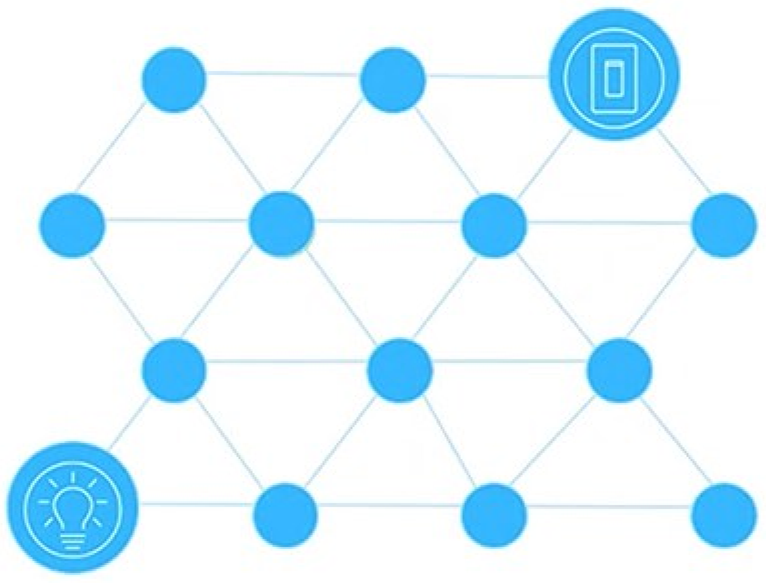

Figure 1: Mesh network topology. A message can be relayed from one point in the network to any other by hopping across bi-directional channels connecting multiple nodes. (Image source: [Silicon Labs](#))

The Bluetooth mesh stack

Since its introduction as part of the Bluetooth Core Specification version 4.0, Bluetooth low energy has been revised through versions 4.1, 4.2 and 5. Bluetooth 5 introduced range, throughput, broadcast, and coexistence improvements. (See Digi-Key article "[Bluetooth 4.1, 4.2 and 5 Compatible Bluetooth Low Energy SoCs and Tools Meet IoT Challenges](#)".)

As the latest production introduction, it could be assumed that Bluetooth mesh 1.0 forms an upgrade to Bluetooth 5 only, but this is not the case. Any legacy (4.0, 4.1, 4.2, 5) Bluetooth low energy chip can be modified to run Bluetooth mesh with just a firmware upgrade, allowing field installations to take advantage of the new technology.

The key to this backwards compatibility comes from the fact that Bluetooth mesh is not an integral part of the Bluetooth low energy stack - rather it is a separate new entity comprising seven layers (Figure 2.)

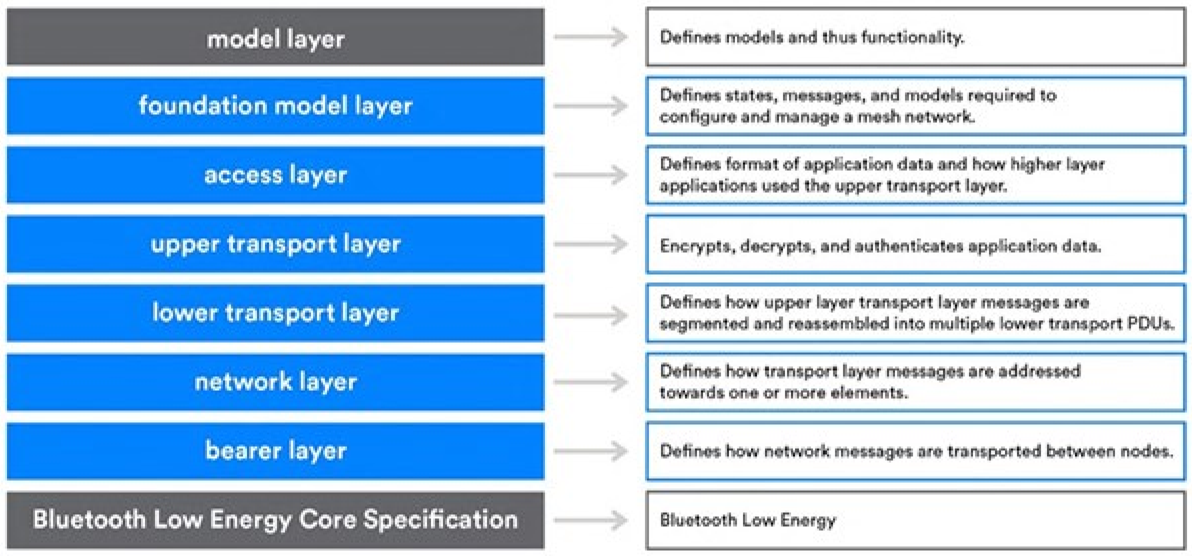
model layer
Defines models and thus functionality.
foundation model layer
Defines states, messages, and models required to configure and manage a mesh network.
access layer
Defines format of application data and how higher layer applications used the upper transport layer.
upper transport layer
Encrypts, decrypts, and authenticates application data.
lower transport layer
Defines how upper layer transport layer messages are segmented and reassembled into multiple lower transport PDUs.
network layer
Defines how transport layer messages are addressed towards one or more elements.
bearer layer
Defines how network messages are transported between nodes.
Bluetooth Low Energy Core Specification
Bluetooth Low Energy

Figure 2: Bluetooth mesh comprises a seven layer stack complementing the Bluetooth low energy protocol. (Image source: Bluetooth SIG)

When the Bluetooth mesh node receives a message, it passes the message up the layers from the underlying Bluetooth low energy stack via the bearer layer to the network layer. The network layer applies various checks to decide whether to pass the message to the transport layers or discard it.

Note that the Bluetooth mesh specification defines a completely new host layer that shares some concepts with the Bluetooth low energy host layer, but is not compatible with it. This is somewhat different to competing technologies such as Zigbee and Thread which were designed from the outset to include mesh networking (Figure 3).

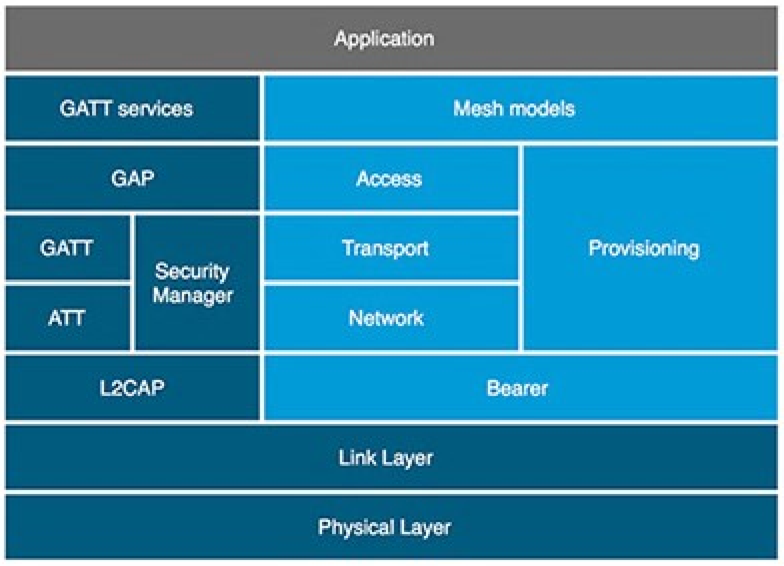

Figure 3: Arrangement of the Bluetooth mesh stack (light blue) within the Bluetooth low energy protocol (dark blue). The Bluetooth mesh specification defines a completely new host layer that shares some concepts with the Bluetooth low energy host layer, but is not compatible with it. (Image source: Nordic Semiconductor)

Bluetooth mesh nodes

Bluetooth mesh uses four types of network nodes:

Relay nodes receive and forward packets across the network. A downside of relay nodes is that they must remain constantly alert, which significantly increases power consumption. This is not a disadvantage for mains powered applications such as smart lighting, but is a problem for non-mains powered nodes such as switches that are incorporated into the network.

Low Power Nodes (LPNs) adopt the standard power-saving characteristics of Bluetooth low energy (i.e.: remaining in sleep states for long periods) and so can operate for long periods from batteries or energy harvesting. Each LPN is connected to a mains-powered Friend Node, which remains awake and caches any messages directed to the LPN. When the LPN enters a receive mode (according to a predetermined schedule) it accepts the cached messages, operates as instructed, and returns to a power-saving sleep mode.

Proxy Nodes allow devices that don't include the Bluetooth mesh stack to connect to a Bluetooth mesh network. This is useful, for example, when a consumer wishes to use a legacy smartphone to control a smart lighting network. Interaction is achieved via both the node and the device's Generic Attribute Profile (GATT) interface (Figure 4).

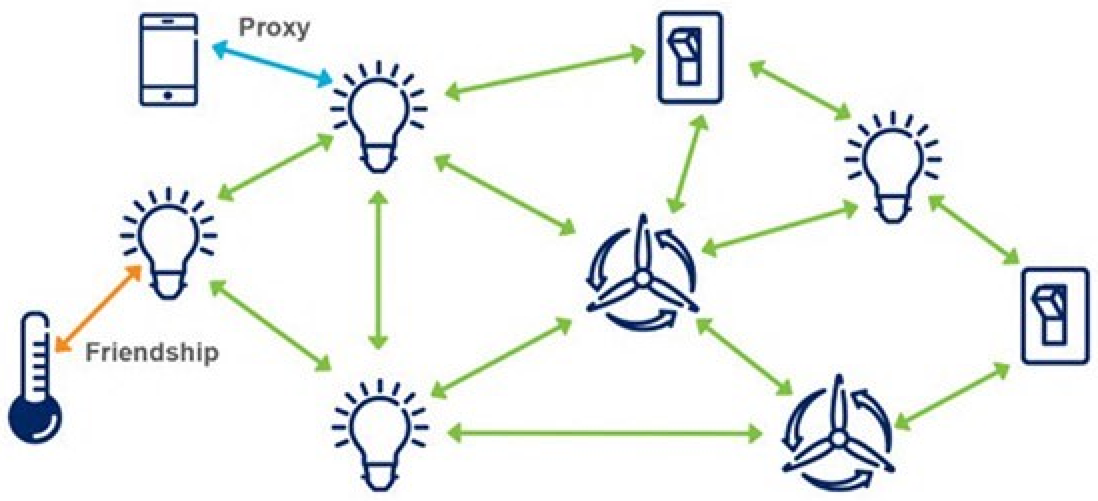

Figure 4: Bluetooth mesh uses four node types. In this image, all bulbs and switches, except the bulb to the far left, are mains powered Relay Nodes. (Image source: Ericsson)

The temperature sensor is a battery-powered LPN and periodically receives messages from the mains-powered Friendship Node formed by the bulb to the far left. The smartphone forms a Proxy Node by using the Bluetooth low energy stack's GATT interface rather than the Bluetooth mesh stack.

Before a new node can take part in routine mesh operation, it must be provisioned by a Provisioner. This is a trusted device with access to all the nodes in a network. The new node is assigned an address, along with network and device keys. After provisioning, the device key is used to establish a secure channel to configure the new node. Bluetooth mesh can support up to 32,000 nodes.

Bluetooth mesh architecture

Bluetooth mesh uses a ‘flooding’ technique to send messages across the network. Every packet is broadcast to every node in the network until it reaches the target(s). A message can be targeted at a single node, a group of nodes, or all nodes.

Groups of nodes are targeted using a group address which defines an element of the network, for example, the lights in a single room. In addition, the specification defines four Fixed Group Addresses: All-proxies, All-friends, All-relays and All-nodes to specifically target the node types. (Note that LPNs can’t be specifically addressed because of their reliance on Friend Nodes.)

The choice of a flooding mesh architecture and group addressing underpins Bluetooth mesh’s suitability for smart home applications. For example, a flooding mesh allows an “ON” command from a switch to rapidly propagate through a smart light network with every node receiving the command and acting accordingly. The lights in the target group illuminate almost instantaneously. The minimal latency in the network is much lower than, for example, that of a star network, where the central device is required to transmit an individual command to each connected light.

In typical operation, Bluetooth low energy’s advertising channels are used to advertise the presence of a Bluetooth device and scan for other devices wishing to communicate. Once the devices have paired, communication moves to one of the 37 full bandwidth channels, accelerating throughput.

In contrast, Bluetooth mesh keeps things simple and lowers latency by not moving to full bandwidth channels once nodes are linked, rather it continues to employ the advertising channels to transfer information.

The downside of this arrangement is reduced network bandwidth and the risk of congestion as just three throughput-restricted frequencies handle all traffic. Two mechanisms come into play to handle the congestion. The first is a “Time-To-Live” (TTL) counter that defines how many times a specific packet can be relayed (a typical value is three steps). The second is a packet cache which captures packets that have circulated completely around the mesh, at which point it’s assumed that further transmission is unnecessary.

The developer can also employ an optional bandwidth preserving relaying feature whereby nodes can receive packets, but not pass them on. The trade-off is a loss of flexibility.

Bluetooth mesh: models instead of profiles

Bluetooth mesh follows Bluetooth technology’s architecture by using GATT Profiles which allow many use cases to share a common information structure. But in the Bluetooth mesh stack these Profiles are called Models.

A Model represents a specific behavior or service and defines a set of states and messages that act on these states. Standard models cover typical usage scenarios like device configuration, sensor readings, and light control. Vendors can also create custom models.

The Models in a node are arranged in elements; each element acts as a virtual entity in the mesh with a unique address. Each incoming message is handled by a Model in an element (Figure 5).

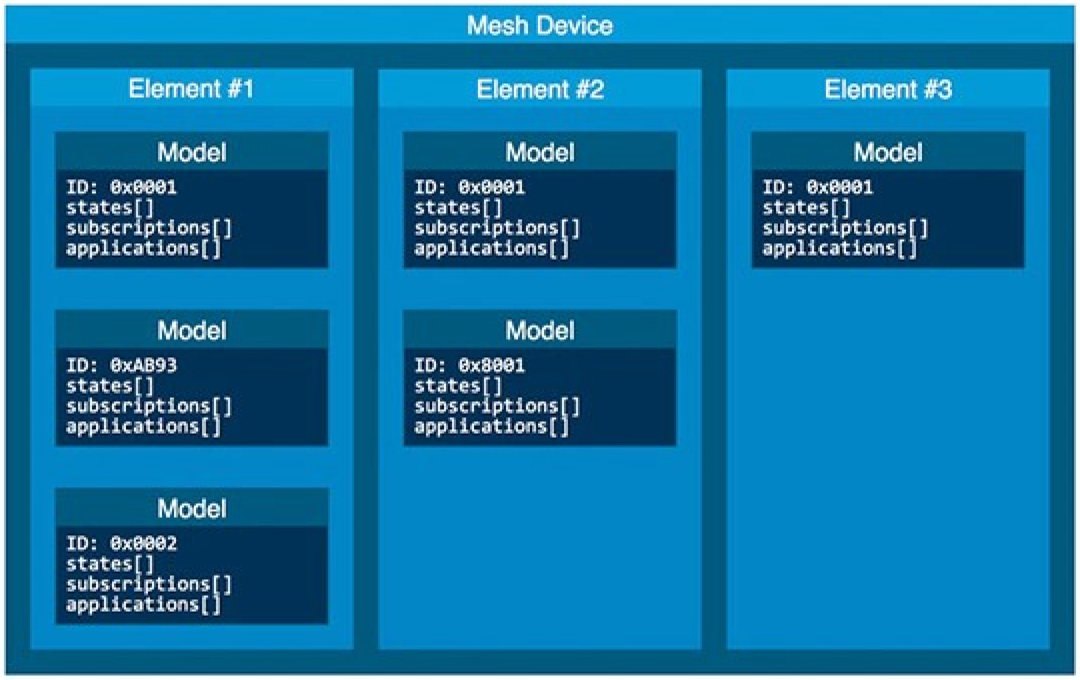

Figure 5: Each network node (Mesh Device) incorporates models grouped into elements. Each element has a unique address and the Models in an element handle incoming messages. (Image source: Nordic Semiconductor)

Models talk to each other through a “publish and subscribe” system. Publishing sends a message and nodes are configured to subscribe to messages sent to specific addresses for processing.

In Figure 6, the light switch to the far left (Switch 1) publishes to the Kitchen group address. Nodes Light 1, Light 2, and Light 3 subscribe to the Kitchen address and therefore receive, process, and act upon messages (such as “on” and “off” commands) published to this address. Note that Light 3 also subscribes to the Dining Room address and can therefore be operated from Switch 2 as well as Switch 1.

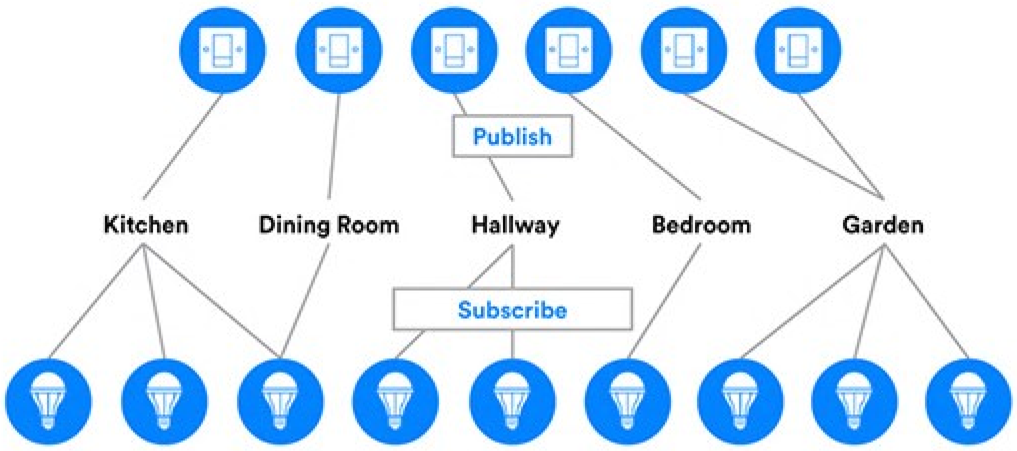

Figure 6: Models talk to each other through a publish and subscribe system. A Model can subscribe to more than one publisher. (Image source: Bluetooth SIG)

(Note that Models is an involved subject. Readers are advised to consult reference 1 for detail beyond that summarized in this article.)

Designing with Bluetooth mesh

Some Bluetooth low energy vendors have introduced standard-compliant Bluetooth mesh stacks. Because Bluetooth mesh is complementary to the established Bluetooth low energy protocol, there is no requirement for a developer to replace a mature and proven Bluetooth low energy stack with a new one to take advantage of Bluetooth mesh. And the Bluetooth mesh firmware brings all of the features of the technology to new or existing designs without the developer having to write a lot of new code.

Nordic Semiconductor, for example, has recently added the nRF5 SDK for Mesh to its development kit line up. The software development kit includes a selection of drivers, libraries and examples, and is designed to run on several integrated development environments (IDEs) and compilers including CMake and SEGGER Embedded Studio.

The compiled code runs on the company's nRF52 DK development kit which is based on the company's nRF52832 silicon.

Nordic's documentation details how to develop a mesh network with clear instructions on how to compile the Bluetooth mesh stack, how to provision the mesh, how to build a network, and how to create new Models.

In Figure 7 below, a new device (a light bulb) is provisioned and configured using the nRF5 SDK for Mesh. In this illustration, the light bulb signals to the Provisioner that it is looking for a network to join. The Provisioner validates the light bulb's beacon and invites it to join the network. If the authentication succeeds, the device is given the necessary keys and addresses to join the network and prepare for configuration. Next, the light bulb is given the "home automation" application key. The publication state of the "OnOff Server" (which controls the light bulb) is set, and finally a subscription to the "light group" is added.

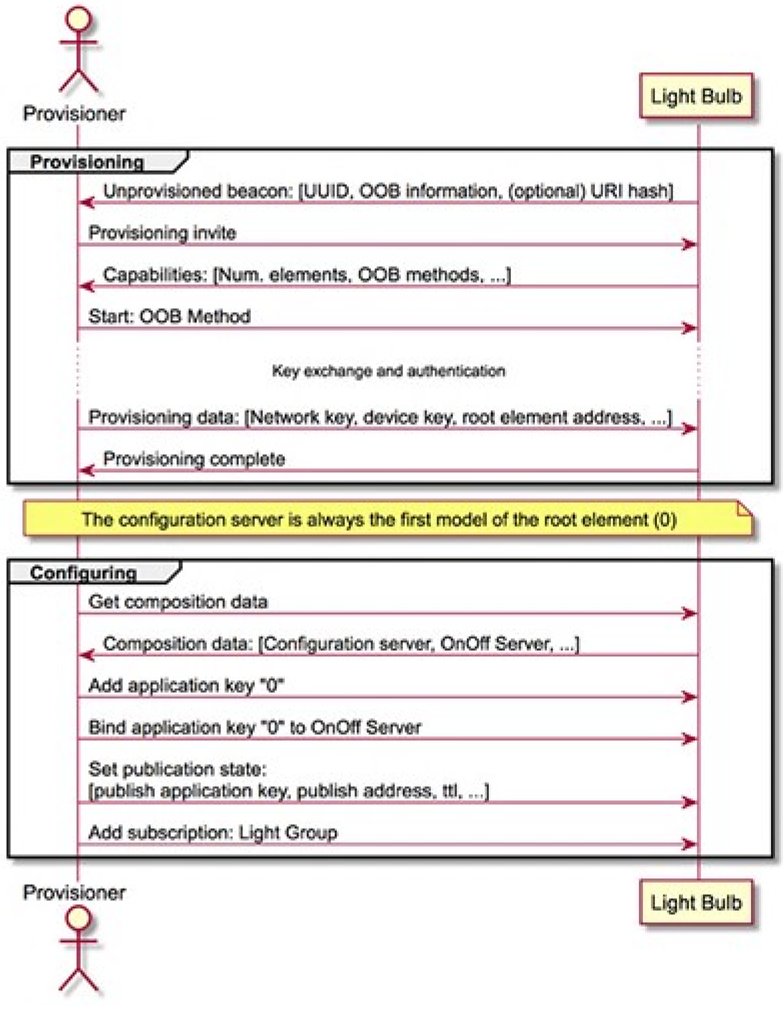
Provisioner
Light Bulb
Provisioning
Unprovisioned beacon: [UUID, OOB information, (optional) URI hash]
Provisioning invite
Capabilities: [Num. elements, OOB methods, ...]
Start: OOB Method
Key exchange and authentication
Provisioning data: [Network key, device key, root element address, ...]
Provisioning complete
The configuration server is always the first model of the root element (0)
Configuring
Get composition data
Composition data: [Configuration server, OnOff Server, ...]
Add application key "0"
Bind application key "0" to OnOff Server
Set publication state: [publish application key, publish address, ttl, ...]
Add subscription: Light Group
Provisioner
Light Bulb

Figure 7: Provisioning and configuring a light bulb in a mesh network using Nordic Semiconductor's nRF5 SDK for Mesh. (Image source: Nordic Semiconductor)

Silicon Labs also offers Bluetooth mesh with its EFR32 Blue Gecko Bluetooth Starter Kit. The company recommends purchasing three or four of the kits to build a prototype mesh network. The kits are based on the company's EFR32MG1 Bluetooth low energy SoC. In addition to the hardware, the developer will require a Bluetooth SDK and a Bluetooth mesh SDK which are both downloadable from the company's website.

Simplicity Studio IDE is required to get up and running with Silicon Labs' Bluetooth mesh technology. The development tools include precompiled demos, application notes, and examples. An Android app is available to provision, configure and control Bluetooth mesh nodes from a smartphone application.

STMicroelectronics has adopted a similar approach to Nordic Semiconductor and Silicon Labs with an SDK that enables development of networked devices based on the company's BlueNRG-2 Bluetooth low energy SoCs.

Conclusion

Bluetooth mesh brings mesh networking to Bluetooth low energy, eliminating the need to use proprietary mesh firmware for smart home applications.

The technology trades off simplicity against slightly increased power consumption and some flexibility. A key advantage is that the technology is compatible with all Bluetooth low energy chips, not just the latest Bluetooth 5 products.

Bluetooth low energy chip vendors have started to introduce Bluetooth mesh software development kits to complement their mature hardware and protocol firmware products.

Part 2 of this article will go into detail about how to integrate Bluetooth mesh into Bluetooth low energy using readily available hardware, firmware and development kits.

Reference

1. "*Bluetooth Mesh Networking/ An Introduction for Developers*", Bluetooth SIG, August 2017.

Share This Article

Related Videos

Silicon Labs
EFR32xG12 Wireless Gecko

- Multi-protocol
  - ZigBee / Thread (Mighty Gecko)
  - Bluetooth BLE (Blue Gecko)
  - Proprietary (Flex Gecko)
- Improved 2.4 GHz sensitivity
- Improved Sub-GHz performance

SILICON LABS
EFR32xG12
Digi-Key

Silicon Labs EFR32xG12 Wireless Gecko|Digi-Key Daily

Publish Date: 2017-05-16

Related Product Training Modules

Introduction to Wireless Gecko
Introduction to a platform for IoT solutions needing Mesh, Bluetooth, or Proprietary wireless protocols

Related Product Highlight

nRF52832 Bluetooth® v4.2 and BT 5
Nordic’s nRF52832 SoC is the first in a line of powerful, highly flexible, ultra-low-power, multiprotocol SoC devices ideally suited for Bluetooth® low energy.

Wireless Gecko IoT Connectivity Portfolio
Silicon Labs offers integrated, robust, reliable, and easy-to-use wireless and RF IC solutions available, with its wireless Gecko IoT connectivity portfolio.

Nordic BLE Mesh
Nordic's software development kit for Bluetooth mesh solutions using nRF51 series and nRF52 series enables new applications for Bluetooth in smart home, lighting, beaconing, and asset tracking applications.

About this publisher

Digi-Key's North American Editors