ABSTRACT

The Bluetooth® Mesh Software Product Brief describes the overall Bluetooth Mesh software content provided by Texas Instruments (TI). This document provides a high-level overview of the features and capabilities of our Bluetooth Mesh stack without specific details. The Bluetooth Mesh stack is provided inside TI’s Bluetooth 5 Stack’s Software Development Kit (SDK). For further information on APIs and implementation details, see the “BLE5-Stack User’s Guide” [9] inside the SIMPLELINK-CC13XX-CC26XX-SDK.

Table of Contents

1 Introduction ............................................................................................................................................................................ 2
2 Reference Examples .................................................................................................................................................................. 2
3 Software Block Diagram ........................................................................................................................................................ 3
4 Network Topology and Features ........................................................................................................................................... 4
5 Security .................................................................................................................................................................................... 5
6 Performance and Test Data......................................................................................................................................................... 5
   6.1 Hardware Setup .................................................................................................................................................................. 5
   6.2 Software Setup .................................................................................................................................................................. 5
7 Power Consumption of the Low Power Node (LPN) .................................................................................................................. 7
8 Out-of-Box Experience ............................................................................................................................................................ 7
9 Tools ........................................................................................................................................................................................................ 7
10 Known Limitations ................................................................................................................................................................... 8
11 References ................................................................................................................................................................................ 8

List of Figures

Figure 3-1. TI Bluetooth Mesh Software Architecture Block Diagram ...................................................................................... 3
Figure 6-1. Mesh Latency Setup .................................................................................................................................................. 5
Figure 6-2. Message Latency (ms) Across Multiple Hops (3 Retransmissions) ....................................................................... 6

List of Tables

Table 1-1. Software Overview .................................................................................................................................................. 2
Table 2-1. Flash Memory Consumed by Reference Examples Running on CC2652 .................................................................. 2
Table 4-1. TI Bluetooth Mesh Model Support Summary ............................................................................................................. 4
Table 6-1. Mesh Network Node Configuration .......................................................................................................................... 6
Table 6-2. Mesh Network Performance Across Multiple Hops ..................................................................................................... 6
Table 7-1. Measured Power Consumption for a LPN .................................................................................................................... 7

Trademarks

Bluetooth® is a registered trademark of Bluetooth SIG, Inc.
Zephyr® is a registered trademark of The Linux Foundation.
All trademarks are the property of their respective owners.
1 Introduction

Bluetooth Mesh [2] is standardized by the Bluetooth Special Interest Group (SIG) [1] in the Bluetooth Mesh Specification [3] and defines fundamental requirements to enable an interoperable mesh networking solution for Bluetooth Low Energy (LE) wireless technology. Built upon existing Bluetooth LE technology, Bluetooth mesh extends the range of wireless connectivity, supports large device networks, and offers mandatory security to offer a proven and scalable software solution.

TI’s Bluetooth Mesh solution [14] is based on the open source Zephyr® Project’s Bluetooth Mesh library. A translation layer is provided that bridges TI’s Bluetooth 5 stack [9] with the Zephyr Project solution [4], allowing for additional proprietary feature support through TI’s Bluetooth 5 stack implementation.

TI’s Bluetooth Mesh solution is designed to support most Bluetooth Mesh specified profiles [3], including both foundation models as well as custom vendor commands.

Table 1-1. Software Overview

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>TI’s Bluetooth Mesh solution</td>
<td>2.02.xx.xx</td>
</tr>
<tr>
<td>Distribution</td>
<td>Included in simplelink_cc13x2_26x2_sdk_5_10_00_xx and later <a href="https://www.ti.com/tool/SIMPLELINK-CC13XX-CC26XX-SDK">https://www.ti.com/tool/SIMPLELINK-CC13XX-CC26XX-SDK</a></td>
</tr>
<tr>
<td>IDE Support</td>
<td>CCS 10.3+ for Windows® 10, Linux®, and macOS® IAR EWARM-8.50.9 for Windows® 10</td>
</tr>
<tr>
<td>Compiler support</td>
<td>TI Arm 20.2.4.LTS+ TI Clang 1.2.1+</td>
</tr>
<tr>
<td>Recommended development kits</td>
<td>PC (host) + CC26x2R1 Launchpad CC1352P Launchpad Launchpad™ SensorTag LPSTK-CC1352R</td>
</tr>
<tr>
<td>Certifications</td>
<td>Bluetooth mesh profile specification v1.0.1 - QDID 162204</td>
</tr>
</tbody>
</table>

2 Reference Examples

<table>
<thead>
<tr>
<th>Application</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>simple_mesh_node</td>
<td>Implements a basic Bluetooth Mesh node with Relay, Proxy, Friend, and Low Power Node capabilities using GATT/ADV bearer. Concurrent operation of a Bluetooth Mesh node and a Bluetooth LE peripheral is also supported.</td>
</tr>
<tr>
<td>simple_mesh_node_oad_offchip</td>
<td>Same as simple_mesh_node above with off-chip OAD functionality to enable wireless firmware updates over the Bluetooth LE connection.</td>
</tr>
<tr>
<td>simple_mesh_node_oad_onchip</td>
<td>Same as simple_mesh_node above with on-chip OAD functionality to enable wireless firmware updates over the Bluetooth LE connection.</td>
</tr>
<tr>
<td>mesh_app_python</td>
<td>A Python based network processor project that utilizes eRPC to interface with a simple_mesh_node.</td>
</tr>
</tbody>
</table>

Table 2-1. Flash Memory Consumed by Reference Examples Running on CC2652

<table>
<thead>
<tr>
<th>Code Example and Wireless MCU</th>
<th>Consumed FLASH (in bytes)</th>
<th>Consumed RAM (in bytes)</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>simple_mesh_node</td>
<td>178929</td>
<td>27175</td>
<td>Basic node</td>
</tr>
<tr>
<td>simple_mesh_node</td>
<td>200277</td>
<td>25594</td>
<td>Basic node + ERPC (2-chip)</td>
</tr>
<tr>
<td>simple_mesh_node</td>
<td>182665</td>
<td>27351</td>
<td>Low Power Node</td>
</tr>
<tr>
<td>simple_mesh_node</td>
<td>204690</td>
<td>25770</td>
<td>Low Power Node + ERPC (2-chip)</td>
</tr>
<tr>
<td>simple_mesh_node</td>
<td>179095</td>
<td>27175</td>
<td>Relay</td>
</tr>
<tr>
<td>simple_mesh_node</td>
<td>200440</td>
<td>25594</td>
<td>Relay + ERPC (2-chip)</td>
</tr>
</tbody>
</table>
The above data is taken based off the simple_mesh_node example with select configurations. Additionally, devices that were compiled with ERPC support consequently have UART enabled, as ERPC enables a 2-chip configuration over serial port (i.e. UART). This leads to a slightly higher memory footprint with the ERPC configured projects. For more information about ERPC and our host tool, refer to the following section of our User's Guide: TI Bluetooth Mesh Host PC Tool. Note: the links provided in this document pertain to SDK version 5.30.01.01. For the most accurate information, be sure to leverage the documentation dedicated to the SDK version you are using. Actual results may vary slightly based on the SDK you are leveraging, device type, and other factors like IDE and compiler. The results shown above were taken using a CC2652R1 Launchpad example on the following SDK: simplelink_cc13xx_cc26xx_sdk_5_30_01_01 using CCS v11.0 and the TI Clang compiler v1.3.0 LTS. The data provided above should be used for reference purposes only.

3 Software Block Diagram

TI’s Bluetooth mesh stack is a certified mesh stack, implemented per the specification [3] provided by the Bluetooth SIG [1]. TI’s Bluetooth Mesh solution is based on the open source Zephyr Project’s Bluetooth Mesh library [4]. A translation layer is provided that bridges TI’s Bluetooth 5 stack with the Zephyr Project solution. Shown below is the software flow diagram with regards to Zephyr Mesh and TI’s BLE5 stack.

Figure 3-1. TI Bluetooth Mesh Software Architecture Block Diagram

For more details on the software layers of the Zephyr Mesh stack, please refer to the Bluetooth Mesh Basics section of the BLE5-Stack’s User Guide. In addition to the User’s Guide section mentioned above, users can refer directly to documentation provided on the Zephyr Project page.
4 Network Topology and Features

TI’s Bluetooth mesh solution supports all node features defined by the specification, including:

<table>
<thead>
<tr>
<th>Node Features</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relay</td>
<td>Yes</td>
</tr>
<tr>
<td>Proxy</td>
<td>Yes</td>
</tr>
<tr>
<td>Friend</td>
<td>Yes</td>
</tr>
<tr>
<td>Low Power</td>
<td>Yes</td>
</tr>
</tbody>
</table>

As part of the specification [3], nodes can be configured to support multiple node features at one time. For example, a node can be configured to support Proxy features and Relay Features. However, a node configured as a Low Power Node may not be configured for any additional role.

TI’s Bluetooth mesh solution also supports the following Bluetooth SIG [1] defined models. At this time, the below models are supported by TI but not qualified by the Bluetooth SIG (with the exception of Foundation models) [1]. In addition to the below list, TI supports the creation of custom Vendor models as well.

Table 4-1. TI Bluetooth Mesh Model Support Summary

<table>
<thead>
<tr>
<th>Model Group</th>
<th>Mesh Models</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foundation</td>
<td>Foundation Models (configuration &amp; health server/client)</td>
<td>Yes</td>
</tr>
<tr>
<td>Vendor</td>
<td>Custom Vendor Model</td>
<td>Yes</td>
</tr>
<tr>
<td>Generic</td>
<td>Generic OnOff Server</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Generic OnOff Client</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Generic Level Server</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Generic Level Client</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Generic Default Transition Time Server</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Generic Default Transition Time Client</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Generic Power OnOff Server</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Generic Power OnOff Setup Server</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Generic Power OnOff Client</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Generic Power Level Server</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Generic Power Level Setup Server</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Generic Power Level Client</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Generic Battery Server</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Generic Battery Client</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Generic Location Server</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Generic Location Setup Server</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Generic Location Client</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Generic Admin Property Server</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Generic Manufacturer Property Server</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Generic User Property Server</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Generic Client Property Server</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Generic Property Client</td>
<td>No</td>
</tr>
<tr>
<td>Sensor</td>
<td>Sensor Server</td>
<td>Partial</td>
</tr>
<tr>
<td></td>
<td>Sensor Client</td>
<td>Partial</td>
</tr>
<tr>
<td></td>
<td>Sensor Setup Server</td>
<td>No</td>
</tr>
<tr>
<td>Time and Scenes</td>
<td>All</td>
<td>No</td>
</tr>
<tr>
<td>Lighting</td>
<td>All</td>
<td>No</td>
</tr>
</tbody>
</table>
Our solution supports provisioning over both GATT (PB-GATT) and Advertisement (PB-ADV) Bearer’s, as defined in the Bluetooth Mesh Profile Specification [3].

5 Security

TI’s Bluetooth Mesh solution, based on the Bluetooth Mesh Profile specification [3], supports mandatory key security features to protect against privacy concerns, replay attacks, trashcan attacks, and more. Features like encryption, authentication, the use of separate application and network keys, key refresh, sequence numbering and IV indexes, along with AES-CMAC and AES-CCM algorithms are all used to protect a Bluetooth mesh network from malicious attacks.

All messages in the mesh network are encrypted and authenticated using two types of keys. One key type is for the network layer communication. This ensures that all communication within a mesh network uses the same network key. The other key type is for application data. Separating the keys for networking and applications allows sensitive access messages (e.g. for access control to a building) to be separated from non-sensitive access messages (e.g. for lighting).

In addition, the network security model utilizes a privacy mechanism called obfuscation which utilizes the Advanced Encryption Standard (AES) to encrypt the source address, sequence numbers, and other header information using a private key. The intent for obfuscation is to make tracking nodes more difficult.

For more information regarding Bluetooth mesh security, refer to the SDK User’s Guide and navigate to Bluetooth Mesh Overview [9].

6 Performance and Test Data

TI’s Bluetooth Mesh solution undergoes robust quarterly testing that includes a variety of stability, performance, and large network tests. This section summarizes a performance study performed on our solution to evaluate end to end latency alongside packet error rate (PER) across multiple hops with variable sized message payloads. In mesh networking, messages are propagated from a source node along a path to a target destination node. A path consists of mesh nodes, and a message “hops” from node to node until it reaches its destination.

6.1 Hardware Setup

The test involves 7 CC26x2R1 Launchpad Kits configured as mesh nodes: 1 source node, 1 destination node, and 5 relay nodes.

6.2 Software Setup

Source and destination nodes were configured as mesh nodes using a network processor solution, leveraging ERPC and our mesh_app_python example to control these nodes using Python [10] on a Windows PC.

The other relay nodes in the network in this test are strictly embedded nodes without network processor support. Mesh network payloads were sent from the source node to the destination node with the following configuration:
Table 6-1. Mesh Network Node Configuration

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Size</td>
<td>1, 2, 3, 4 byte(s)</td>
</tr>
<tr>
<td>Message Type</td>
<td>Unsegmented</td>
</tr>
<tr>
<td># Messages</td>
<td>100</td>
</tr>
<tr>
<td>ADV interval</td>
<td>10 ms</td>
</tr>
<tr>
<td>Scan interval</td>
<td>20 ms</td>
</tr>
<tr>
<td>Net transmit</td>
<td>1</td>
</tr>
<tr>
<td>Net transmit interval</td>
<td>10 ms</td>
</tr>
<tr>
<td>Relay retransmit</td>
<td>3</td>
</tr>
<tr>
<td>Relay retransmit interval</td>
<td>10 ms</td>
</tr>
</tbody>
</table>

The following are the latency results for the above configuration, including the time in milliseconds it took to deliver the message from source to destination node across each hop. The percentage of messages received is also recorded across each hop.

![Figure 6-2. Message Latency (ms) Across Multiple Hops (3 Retransmissions)](image)

Table 6-2 summarizes the data presented above, measured on a 6-node network with variable unsegmented payload sizes, up to 3 retransmissions allowed, and a 10ms advertisement interval.

Table 6-2. Mesh Network Performance Across Multiple Hops

<table>
<thead>
<tr>
<th>Hop</th>
<th>1 byte</th>
<th>2 bytes</th>
<th>3 bytes</th>
<th>4 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E2E delay (ms)</td>
<td>PER (%)</td>
<td>E2E delay (ms)</td>
<td>PER (%)</td>
</tr>
<tr>
<td>1</td>
<td>33.48</td>
<td>&lt;3</td>
<td>29.16</td>
<td>&lt;3</td>
</tr>
<tr>
<td>2</td>
<td>104.51</td>
<td>&lt;5</td>
<td>103.19</td>
<td>&lt;3</td>
</tr>
<tr>
<td>3</td>
<td>122.88</td>
<td>&lt;3</td>
<td>118.77</td>
<td>&lt;10</td>
</tr>
<tr>
<td>4</td>
<td>131.57</td>
<td>&lt;5</td>
<td>134.60</td>
<td>&lt;5</td>
</tr>
<tr>
<td>5</td>
<td>157.44</td>
<td>&lt;5</td>
<td>158.63</td>
<td>&lt;3</td>
</tr>
<tr>
<td>6</td>
<td>191.67</td>
<td>&lt;5</td>
<td>196.51</td>
<td>&lt;3</td>
</tr>
</tbody>
</table>

Bluetooth Mesh allows mesh nodes to retransmit messages to increase the reliability and robustness of a mesh network. The tests performed above allowed three retransmissions, resulting in a very low packet error rate. An
average PER of 3% was seen across each hop and for all payload sizes listed in the table above. However, low PER comes at the cost of additional latency to transmit the additional messages. This is an example of one of the many tradeoffs to be considered when designing a mesh network. With proper network deployment, application design, and configuration of relevant parameters of the protocol stack, Bluetooth mesh is able to support the operation of dense networks with tens to hundreds of devices.

7 Power Consumption of the Low Power Node (LPN)

One of the key features of Bluetooth mesh is the introduction of a low power node. This mesh node feature allows for a battery powered device to receive messages at configurable intervals while the device sleeps in between each wakeup period. As the specification defines, a low power node must be paired with a friend node. The role of the friend node is to receive messages for the low power node and cache them. When the receive window starts, the low power node will wake up and the friend node will transmit the cached data to its recipient. The receive window reflects the amount of time the LPN listens for a response from its friend. An LPN can reduce current consumption by spending less time listening for a response, i.e. choosing a friend with a smaller receive window.

Poll timeout is the maximum time the LPN may take before requesting cached data from its Friend. This is a critical factor for low power nodes as this time represents the maximum amount of time the node may sleep for before the next wakeup event. If poll timeout is set too small, the LPN will wakeup more often resulting in more current consumption. Setting the poll timeout too large may result in additional latency when receiving updates from the LPN, since it wakes up less frequently.

To test the power consumption of an LPN, one friend node and one LPN are configured with the features listed below. All LPNs are configured with a scan delay of 0 ms and the Secure Network Beacon feature is disabled. Setting the scan delay to 0 ms ensures that the LPN wakes up in sync with its friend node to send a poll message. This saves power because the LPN maximizes the receive window and is not in receive mode (RX) for longer than necessary. During each test, 20 measurements were taken with a sample rate of 2.56 µs and a buffer size of 128k. The measured power consumption shown is the average current of an LPN taken over a period of 1 minute in the configuration detailed below.

<table>
<thead>
<tr>
<th>Friend Configuration</th>
<th>LPN Configuration</th>
<th>Legacy Advertisement</th>
</tr>
</thead>
<tbody>
<tr>
<td>No cached data</td>
<td>Poll Timeout: 5 sec</td>
<td>35.82 uA</td>
</tr>
<tr>
<td>Receive window: 50 ms</td>
<td>Poll Timeout: 1 min</td>
<td>5.31 uA</td>
</tr>
<tr>
<td>Advertise interval (poll response): 20 ms</td>
<td>Poll Timeout: 5 sec</td>
<td>51.96 uA</td>
</tr>
<tr>
<td>1 cached data packet (1B)</td>
<td>Poll Timeout: 1 min</td>
<td>14.74 uA</td>
</tr>
<tr>
<td>Receive window: 50 ms</td>
<td>Poll Timeout: 5 sec</td>
<td>35.82 uA</td>
</tr>
<tr>
<td>Advertise interval (poll response): 20 ms</td>
<td>Poll Timeout: 1 min</td>
<td>5.31 uA</td>
</tr>
</tbody>
</table>

8 Out-of-Box Experience

TI’s Bluetooth Mesh solution is a qualified Bluetooth mesh stack that offers platform software scalability alongside a scalable silicon portfolio to support different mesh nodes in a network. Our solution is market ready with the following out-of-box experience provided:

- Mesh node example showcasing Proxy functionality out of the box
- Mesh node example showcasing concurrent Bluetooth Mesh + Bluetooth Low Energy support
- Mesh node example showcasing over-the-air download (OAD) functionality for both on-chip and off-chip projects
- Mesh node support for all qualified roles (Relay, Proxy, Friend, LPN)
- Reference python scripts to control your node as a network processor
- Provisioning nodes with TI’s Simplelink Starter Mobile application [11][12]

Start by reviewing our Simplelink Academy Module for Bluetooth Mesh. [8]

9 Tools

- Code Composer Studio
For the latest supported CCS version, see the latest SDK’s Release Notes that can be found on the CC13XX-CC26XX-SDK tool page

• IAR Embedded Workbench

For the latest supported IAR version, see the latest SDK’s Release Notes that can be found on the CC13XX-CC26XX-SDK tool page

• SysConfig

The SysConfig Utility is a software tool which provides a Graphical User Interface for configuring pins, peripherals, radios, subsystems, and other components for TI devices. Results output as C header and code files that can be imported into software development kits (SDKs) or used to configure custom software.

With regards to Bluetooth Mesh, refer to the SDK’s User’s Guide for an overview of the configurable parameters that may be modified using SysConfig. [13]

• mesh_app_python host script

Paired with a simple_mesh_node example configured with ERPC support, this Python script runs on v3.7.6 [10], allowing you to control your mesh node in a network processor configuration by sending commands to the node through a serial interface

10 Known Limitations

For the latest information, including Known Issues and Limitations, see the Release Notes for the SDK version you are using. The latest SDK’s Release Notes can be found on the CC13XX-CC26XX-SDK tool page.

11 References

1. Bluetooth SIG
2. Bluetooth Mesh
3. Bluetooth Mesh Profile 1.0.1 Specification
4. Zephyr Project Mesh Documentation
5. CC13XX-CC26XX-SDK Release Notes
6. Code Composer Studio
7. IAR Embedded Workbench
8. Simplelink Academy Module for Bluetooth Mesh
10. Python 3.7.6
11. TI Simplelink Starter App on Android
12. TI Simplelink Starter App on iOS
13. SysConfig Tool
14. TI Bluetooth Mesh
IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2022, Texas Instruments Incorporated