Application Note

TI Bluetooth® Mesh Software Product Brief

Ammar Nek

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

List of Figures

List of Tables

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

Component	Version
TI's Bluetooth Mesh solution	2.02.xx.xx
Distribution	Included in **simplelink_cc13x2_26x2_sdk_5_10_00_xx** and later https://www.ti.com/tool/SIMPLELINK-CC13XX-CC26XX-SDK
IDE Support	CCS 10.3+ for Windows® 10, Linux®, and macOS® IAR EWARM-8.50.9 for Windows® 10
Compiler support	TI Arm 20.2.4.LTS+ TI Clang 1.2.1+
Supported devices	CC2642R, CC2652R, CC2652RSIP, CC1352R, CC1352P, CC2652R7
Recommended development kits	PC (host) + CC26x2R1 Launchpad CC1352P Launchpad Launchpad™ SensorTag LPSTK-CC1352R
Certifications	Bluetooth mesh profile specification v1.0.1 - QDID 162204

2 Reference Examples

Application	Usage
simple_mesh_node	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.
simple_mesh_node_oad_offchip	Same as simple_mesh_node above with off-chip OAD functionality to enable wireless firmware updates over the Bluetooth LE connection.
simple_mesh_node_oad_onchip	Same as simple_mesh_node above with on-chip OAD functionality to enable wireless firmware updates over the Bluetooth LE connection.
mesh_app_python	A Python based network processor project that utilizes eRPC to interface with a simple_mesh_node.

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

Code Example and Wireless MCU	Consumed FLASH (in bytes)	Consumed RAM (in bytes)	Configuration
simple_mesh_node	178929	27175	Basic node
simple_mesh_node	200277	25594	Basic node + ERPC (2-chip)
simple_mesh_node	182665	27351	Low Power Node
simple_mesh_node	204690	25770	Low Power Node + ERPC (2-chip)
simple_mesh_node	179095	27175	Relay
simple_mesh_node	200440	25594	Relay + ERPC (2-chip)

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.

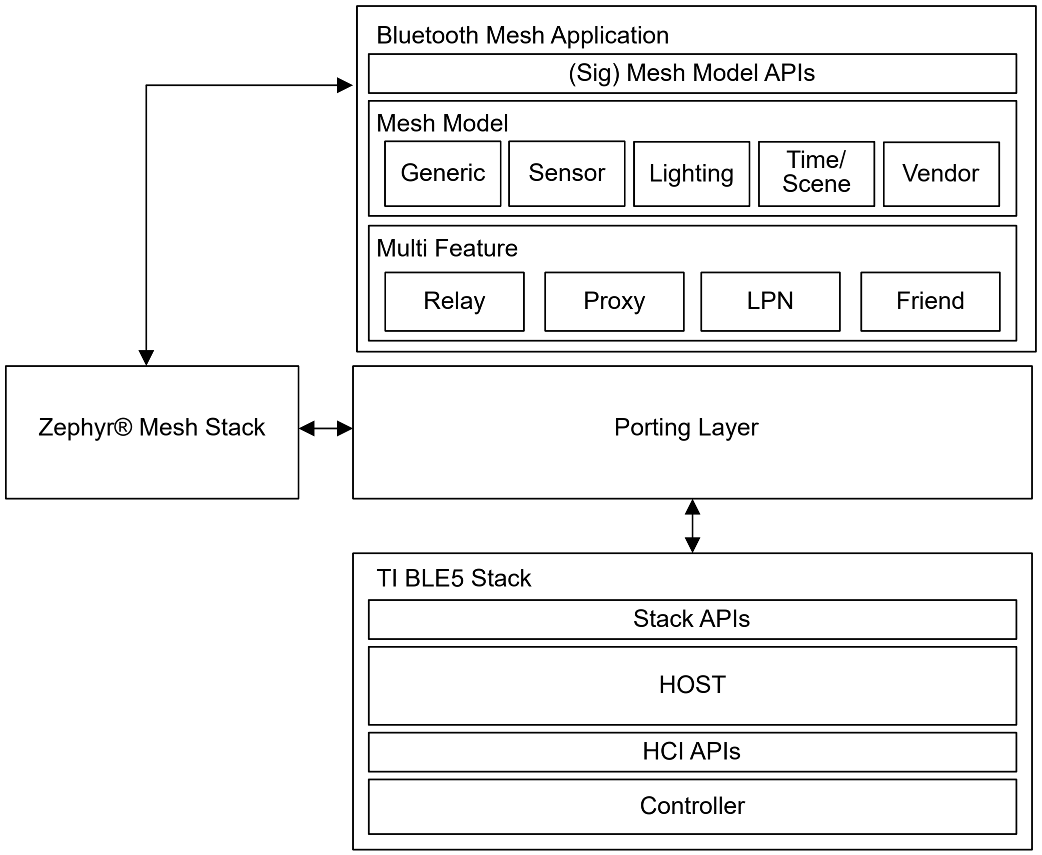

Figure 3-1. TI Bluetooth Mesh Software Architecture Block Diagram

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:

Node Features	Supported
Relay	Yes
Proxy	Yes
Friend	Yes
Low Power	Yes

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

Model Group	Mesh Models	Supported
Foundation	Foundation Models (configuration & health server/client)	Yes
Vendor	Custom Vendor Model	Yes
Generic	Generic OnOff Server	Yes
	Generic OnOff Client	Yes
	Generic Level Server	Yes
	Generic Level Client	Yes
	Generic Default Transition Time Server	Yes
	Generic Default Transition Time Client	Yes
	Generic Power OnOff Server	Yes
	Generic Power OnOff Setup Server	Yes
	Generic Power OnOff Client	Yes
	Generic Power Level Server	No
	Generic Power Level Setup Server	No
	Generic Power Level Client	No
	Generic Battery Server	Yes
	Generic Battery Client	Yes
	Generic Location Server	No
	Generic Location Setup Server	No
	Generic Location Client	No
	Generic Admin Property Server	No
	Generic Manufacturer Property Server	No
	Generic User Property Server	No
	Generic Client Property Server	No
	Generic Property Client	No
Sensor	Sensor Server	Partial
	Sensor Client	Partial
	Sensor Setup Server	No
Time and Scenes	All	No
Lighting	All	No

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.

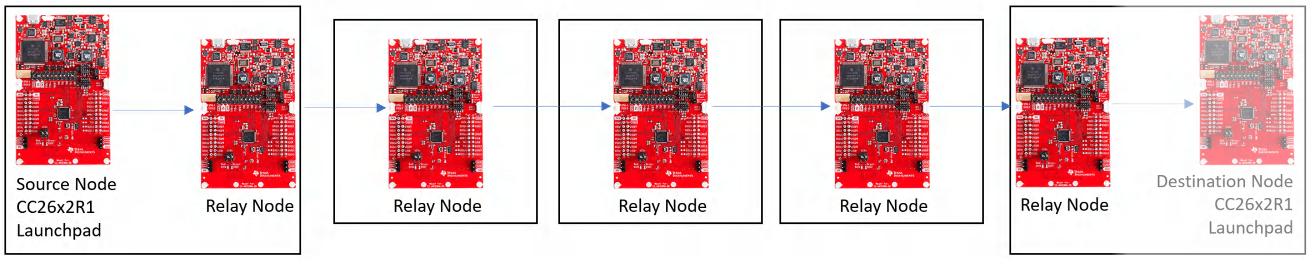

Figure 6-1. Mesh Latency Setup

Each node was placed in a clean box, connected together using coax cables/SMA connectors to ensure a noise-free environment during testing.

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

Configuration	Value
Data Size	1, 2, 3, 4 byte(s)
Message Type	Unsegmented
# Messages	100
ADV interval	10 ms
Scan interval	20 ms
Net transmit	1
Net transmit interval	10 ms
Relay retransmit	3
Relay retransmit interval	10 ms

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.

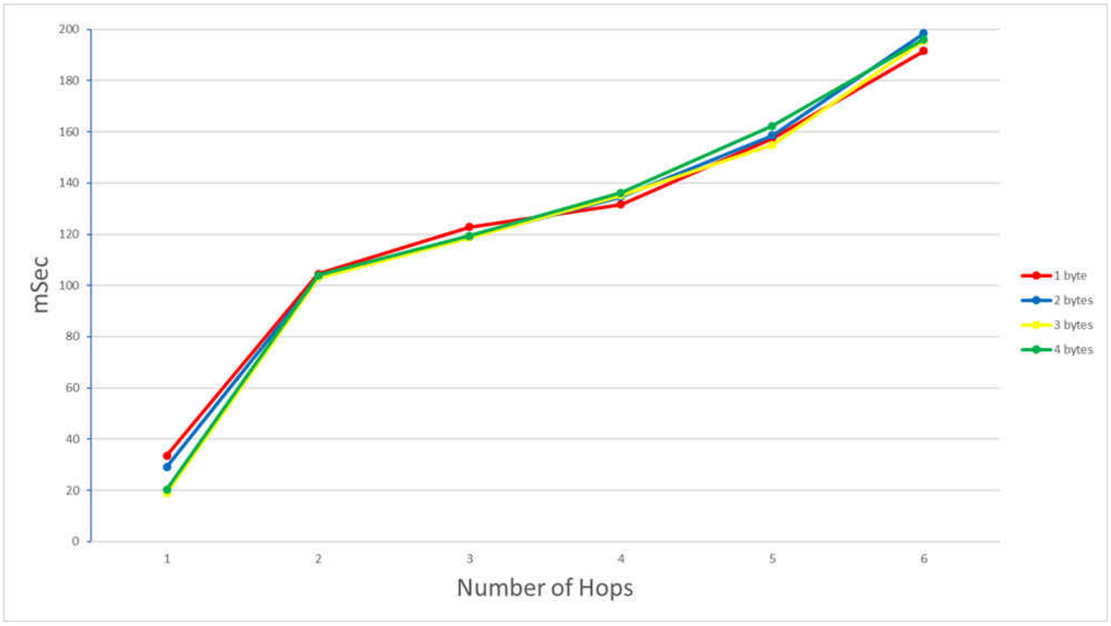

Figure 6-2. Message Latency (ms) Across Multiple Hops (3 Retransmissions)

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

Hop	1 byte		2 bytes		3 bytes		4 bytes	
	E2E delay (ms)	PER (%)	E2E delay (ms)	PER (%)	E2E delay (ms)	PER (%)	E2E delay (ms)	PER (%)
1	33.48	<3	29.16	<3	18.96	<5	20.28	<10
2	104.51	<5	103.19	<3	102.96	<3	103.94	<3
3	122.88	<3	118.77	<10	118.66	<3	119.41	<5
4	131.57	<5	134.60	<5	135.25	<3	136.13	<3
5	157.44	<5	158.63	<3	154.85	<5	162.25	<5
6	191.67	<5	198.51	<3	195.64	<3	196.11	<10

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 7-1. Measured Power Consumption for a LPN

Friend Configuration	LPN Configuration	Legacy Advertisement
• No cached data • Receive window: 50 ms • Advertise interval (poll response): 20 ms	• Poll Timeout: 5 sec	35.82 uA
	• Poll Timeout: 1 min	5.31 uA
• 1 cached data packet (1B) • Receive window: 50 ms • Advertise interval (poll response): 20 ms	• Poll Timeout: 5 sec	51.96 uA
	• Poll Timeout: 1 min	14.74 uA

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
9. Texas Instruments BLE5-Stack User's Guide
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