

Bluetooth® Low Energy Tree Structure Network

Stanford Li, Yan Zhang, and Marie Hernes

ABSTRACT

This application report presents the concept of the wireless tree structure using Bluetooth Low Energy technology. The important steps when designing a Bluetooth Low Energy tree structure are elaborated on a detailed level throughout the document. With the use of the TI SimpleLink™ Bluetooth low energy software Stack, the tree structure can be done in a simple and intuitive way.

The accompanying software example can be found on [github](#).

Contents

1	Introduction	2
2	Bluetooth Low Energy Basic Knowledge	2
3	Three Kinds of Bluetooth Low Energy Network Structure	2
4	Bluetooth Low Energy Tree Structure Network Analysis	4
5	Bluetooth Low Energy Tree Structure Network Realization	6
6	Bluetooth Low Energy tree Structure Network Test	10
7	References	11

List of Figures

1	Star Network.....	2
2	Mesh Network	3
3	Tree Structure Network	4
4	GATT Write + GATT Notify Accomplish Bluetooth Low Energy Tree Structure Read Function	5
5	With Bottom-Up Network Establishment, There is a Risk of a Wrong Connection Between Father Devices ...	5
6	Top-Down Establishment	6
7	Different Index in Grandpa Node and Father Node	6
8	Different Index in Father Node and Children Node	7
9	Index in Tree Structure Networks.....	7
10	Top-Down Write.....	8
11	Down-Top Notify	9
12	Overall Connection Diagram.....	10
13	Tree Structure Write and Notify Test	11

Trademarks

SimpleLink is a trademark of Texas Instruments.
 Bluetooth is a registered trademark of Bluetooth SIG, Inc.
 Zigbee is a registered trademark of ZigBee Alliance.
 All other trademarks are the property of their respective owners.

1 Introduction

Bluetooth Low Energy is a personal local network technology that is designed and sold by Bluetooth Special Interest Group (Bluetooth SIG). It is aimed at applications in the fields of medical and health care, sports and fitness, beacon, security, family entertainment and so on. Especially in the fields of industry and building automation, Bluetooth Low Energy has very low energy consumption and extensive wireless networking features, so Bluetooth Low Energy has a vast market. The Bluetooth Low Energy tree structure network is an extended topology based on star structure networks. Using a tree structure allows the network to include more nodes and a wider range, which makes the Bluetooth Low Energy suitable for more wireless control and sensor applications.

TI SimpleLink Bluetooth Low Energy CC26x2 wireless microcontroller as a Bluetooth central or peripheral has the ability to connect to a maximum of 32 other Bluetooth Low Energy devices simultaneously, that also makes it possible to use the TI Bluetooth Low Energy solution to build a larger and extendable Bluetooth Low Energy network. A tree structure network based on SimpleLink CC26x2 wireless MCU is proposed and analyzed in this application report.

2 Bluetooth Low Energy Basic Knowledge

According to the Bluetooth Low Energy Core Specification, there are two roles defined (GAPRoles) when the Bluetooth Low Energy connection is established. The node that initiates the connection defined as the Central device and the node that is connected to by the Central is defined as the Peripheral device.

There are several parameters that are important for the Bluetooth Low Energy connection, two of them are connection interval and slave latency. The connection interval determines how often the central asks for data from the peripheral. The slave latency represents the number of times the peripheral can choose not to answer when the central asks for data. The connection interval and slave latency typically affect the performance of a Bluetooth Low Energy link the most. For example, the lower the slave latency and faster the connection interval, the faster the effective data transfer rate between the peripheral and central. On the other hand, this also leads to a higher average current consumption since the devices use their radio more often than at a longer connection interval.

3 Three Kinds of Bluetooth Low Energy Network Structure

3.1 Star Network

Star topology is the easiest topology in the three kinds of network structures. It consists of one central node and several peripheral nodes. Each peripheral node can only communicate with the central node and cannot communicate directly with the other peripheral nodes. Peripheral nodes can communicate indirectly through forwarding by the central node.

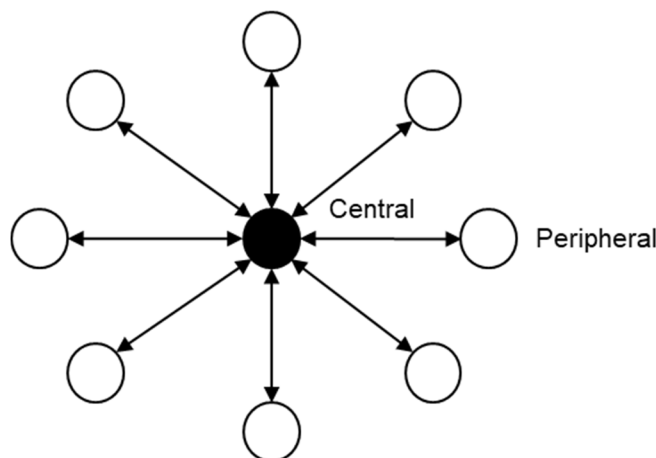


Figure 1. Star Network

The star topology has the advantage of simplicity and directness, thus it has the lowest latency of the networks presented here. However, this network coverage is the smallest because of the limited number of peripheral nodes that can be connected with the central node.

3.2 Mesh

In mesh networks, each device is connected to one or more of the other devices. There is no clear role definition that parallels central/peripheral. A typical real mesh topology (such as Zigbee® or Thread) consists of one coordinator, several routers and several end devices. Routers can communicate with other nodes because the Mesh protocol defines the routing rules. Mesh is considered the most flexible network and it can provide a larger network coverage area. At the same time, Mesh has a strong fault-tolerant ability. If a router crashes in the network, information can still be automatically transmitted along other routing path. On the other hand, mesh networks use complex network protocols that require a lot from the hardware and software that is used. Also, the Mesh networks typically consume more power than other networks, and the data latency is both higher and more unpredictable since the number of jumps between peer devices is not fixed.

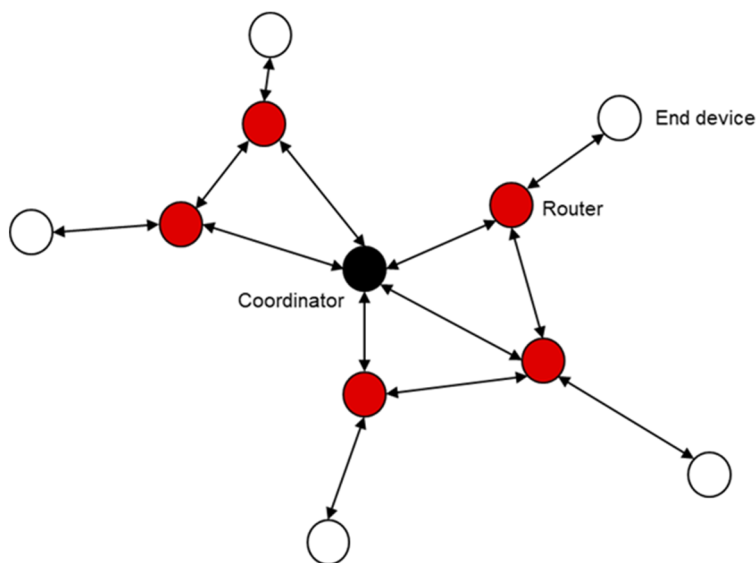


Figure 2. Mesh Network

Bluetooth Mesh is a mesh network protocol based on “message flooding” using the Bluetooth Low Energy Broadcaster and Observer GAP roles. This protocol is quite complicated and is not considered power and latency efficient compared to star networks. Bluetooth Low Energy manufacturers are still researching and developing their Bluetooth Low Energy mesh solutions at this time.

3.3 Tree Structure Network

The tree structure network consists of one grandpa node, several father nodes and several children nodes. Same as star networks, the grandpa node as root can communicate with the father nodes, and the father nodes can communicate with children nodes. Nodes on the same level (nodes of the same type) cannot directly communicate with each other and can only communicate by message forwarding.

Compared to mesh routing rules, the tree structure network routing rules are much simpler. This means the hardware and software requirements are lower than for a mesh network. This makes the tree structure network easier to achieve. Compared to the star network, the tree structure network can connect more nodes. It has more than one level of connection distance, so it has larger network coverage than the star network. The downside of the tree network is a higher latency when there are too many layers, and the network might be vulnerable if one of the father nodes is disconnected. For a real application the number of required nodes is given. Thus latency is predictable and acceptable. In this document a three layer networks is described.

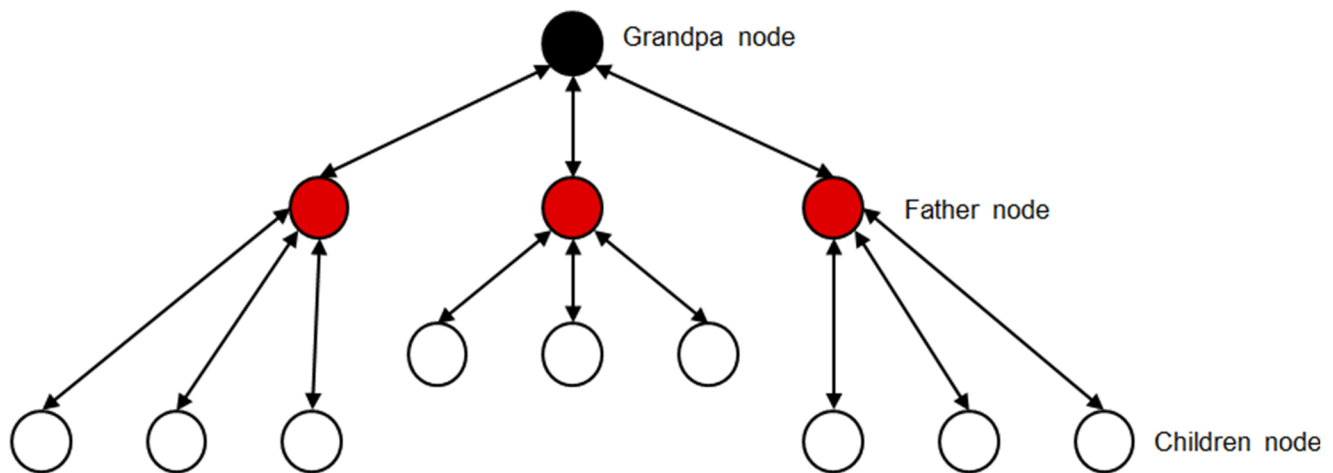


Figure 3. Tree Structure Network

4 Bluetooth Low Energy Tree Structure Network Analysis

This section discusses implementation of the basic Bluetooth Low Energy three level tree structure networks. A new level can be added into it to build a multi-level tree structure network.

4.1 Role Analysis in Bluetooth Low Energy Tree Structure

There are three basic roles in the Bluetooth Low Energy tree structure: grandpa node, father node and child node. The purpose of the Bluetooth Low Energy tree structure is that the grandpa node (could be a gateway to cloud) can exchange messages with all other nodes.

In the [Bluetooth Core Specification](#), Bluetooth Low Energy devices can support write, read and notify functions through the Generic Attribute Layer (GATT layer). To implement bi-directional communication between the grandpa node to all its children nodes, use GATT write and GATT notify. For more information, see the [BLE5-Stack User's Guide](#)). In this example:

- The grandpa node sends data to its children: GATT write
- Child nodes send data to their ancestors: GATT notify
- The grandpa node receives data from the child nodes: GATT write + GATT notify

To reduce network complexity, the only APIs used are GATT write and GATT notify. When the grandpa node has any data to send to the child nodes, it uses the GATT write command to send the data to the relevant father node. The father node uses the GATT write command to forward the data to one of the child nodes. When child or father nodes need to send data to the grandpa node, they use GATT notify to send the data. If the grandpa node needs to get or poll some information from a father or child node, it also uses the GATT write command to request the node to send the data. The child nodes will use GATT notify to reply to the grandpa node. For more details, see [Figure 4](#).

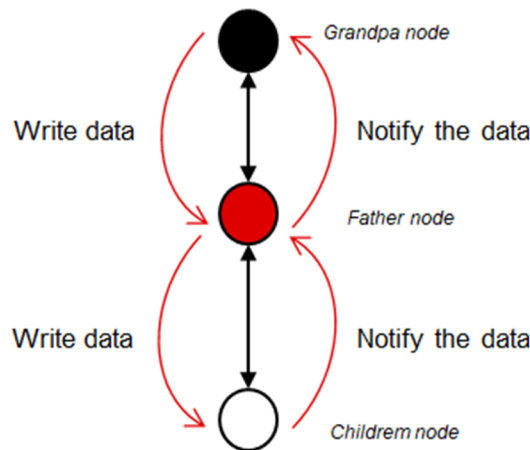


Figure 4. GATT Write + GATT Notify Accomplish Bluetooth Low Energy Tree Structure Read Function

4.2 Tree Structure Network Establishment Analysis

In theory, there are two ways to establish the tree structure network:

- **Top-down:** first the grandpa node connects to all of the father nodes. Then, each father node connects to its own child nodes.
- **Bottom-up:** first each father node connects to its own child nodes. Then, the grandpa node connects to each father node.

In the bottom-up method, after the first father node connects with its own children nodes, the father node is advertising. When the second father node starts to connect with its own child nodes, the first father node can incorrectly be connected to by the second father node.

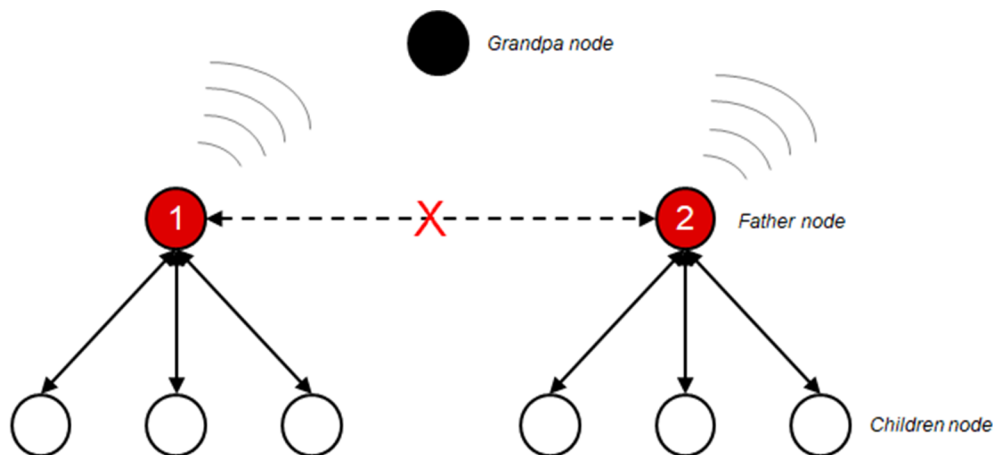


Figure 5. With Bottom-Up Network Establishment, There is a Risk of a Wrong Connection Between Father Devices

In order to avoid this, a feature that keeps track of what devices connect to what devices such as directed advertising or a white list is needed. However, this complicated the code since it means that the network must be formed with the same device in the same position each time.

For the top-down method, when the father node is connected it stops advertising, so different father nodes will never connect to each other. This network establishment method can simplify the software effort compared to implementing directed advertising or a white list.

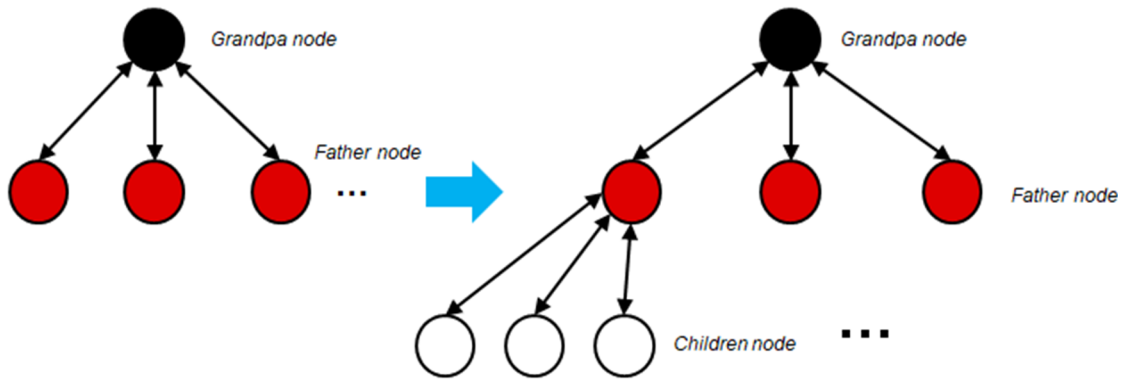


Figure 6. Top-Down Establishment

5 Bluetooth Low Energy Tree Structure Network Realization

5.1 Tree Structure Establishment

Following the discussion in [Section 4.2](#), the top-down method is selected to build the tree structure network. When the grandpa node and a father node establish a connection, the notifications are disabled per default. Notifications must be enabled by the grandpa node when the connection is established. After the connection is established, the grandpa node and father node will update their own GAP connection information list according to the connected order. For the grandpa node, the father node's indexes are 0, 1, 2, and so on. For all father nodes, the grandpa node's index is 0 in their GAP connection info list as shown in [Figure 7](#). After connection, the grandpa node can send each father node's index to them to establish the routing table.

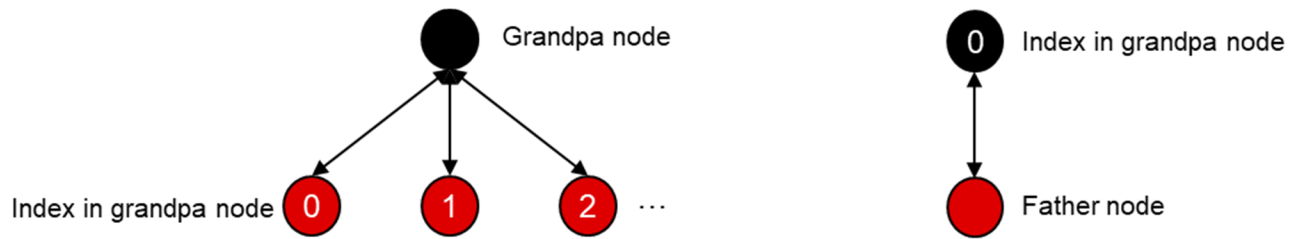


Figure 7. Different Index in Grandpa Node and Father Node

After the grandpa node connects to all of the father nodes, the father node connects the children nodes in order. The father nodes and children nodes will update their GAP connection information list. For the father node, index 0 is always grandpa's node, and the children's node indexes are 1, 2, 3, and so on. For the children node, the index of the father node is always 0, as shown in [Figure 8](#). The father node can send its index and children's node index to the children nodes. Now the whole tree structure routing table is established.

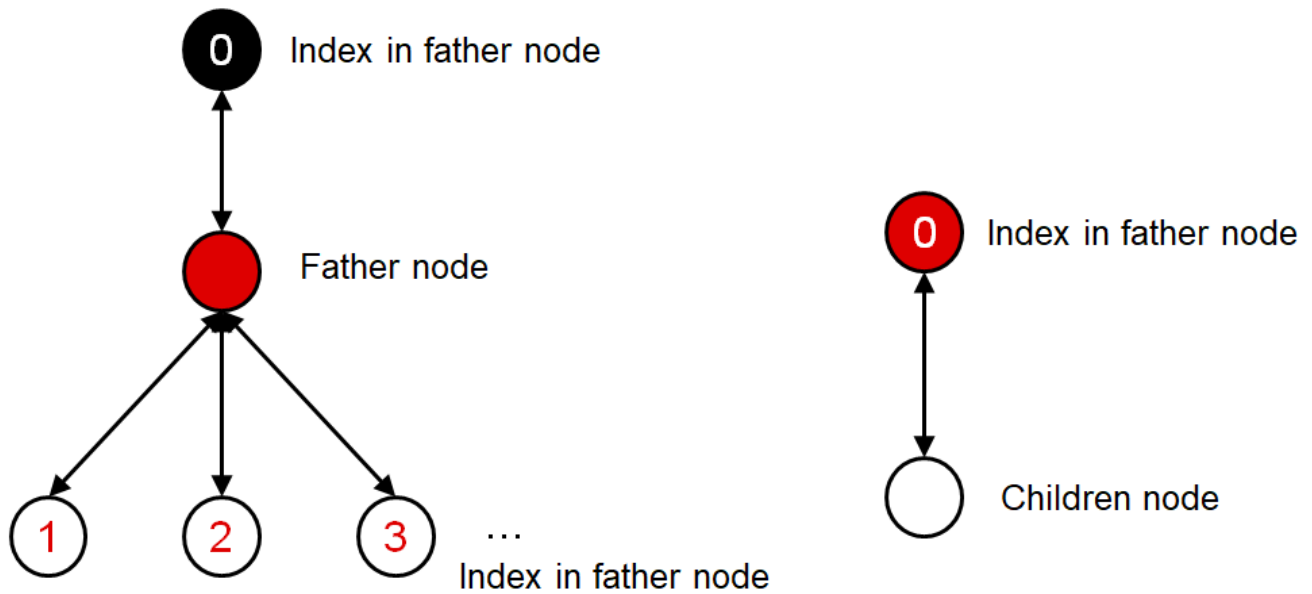


Figure 8. Different Index in Father Node and Children Node

However, it is important to note that some commands are sent directly from the grandpa node to the father node without the need for the father node to forward them to the children nodes. Because of this, there us a distinct index for each node. This document presents a simple method that uses 2 bytes to represent the index. For more details, see [Figure 9](#).

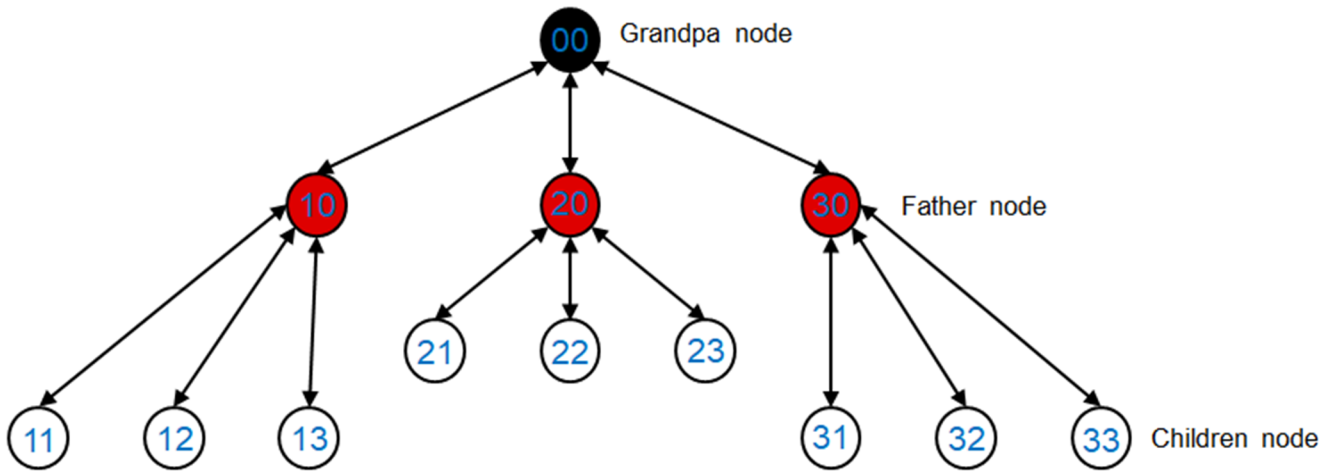


Figure 9. Index in Tree Structure Networks

Now the tree structure network and routing table are established.

5.2 Role Differentiation in Tree Structure Network

According to Section 2, in the tree structure the grandpa node is always a central device. The child nodes are always peripheral role devices. But the father node needs to be both. When in the connection to the grandpa node, it is peripheral role, and when in the connection to the child nodes, it is a central role. This peripheral role to central role changing behavior can be used to identify the father node. If the node's role is changed after the GAP link establishment event, for example, from peripheral to the central, then the node can be identified as a father in the network. The sample code is shown in below:

```

// Update the role type when the lnk established
uint8_t role = ((gapEstLinkReqEvent_t*) pMsg)->connRole;
static uint8_t last_role;

// Judge the role change
if ( ( (role == GAP_PROFILE_CENTRAL) && (last_role == GAP_PROFILE_PERIPHERAL) )
      || ( (role == GAP_PROFILE_PERIPHERAL) && (last_role == GAP_PROFILE_CENTRAL) ) )
    my_role = GAP_PROFILE_MULTIROLE;
else
    my_role = role;

// Save the last role state
last_role = role

```

5.3 Data Transmission in Tree Structure Network

For top-down write function, the data packet includes the commands and destination node router index, so the grandpa node writes the data to the appointed father node according to the router index. The father node receives the data in GATT message process function, then it decides whether this data is destined for itself or one of its child nodes. If the data is meant for the father node itself, the father node directly executes the data. If the data is meant for a child node, the father node forwards the data to the target child node. Then, the top-down transmission is finished. For more details, see Figure 10 and .

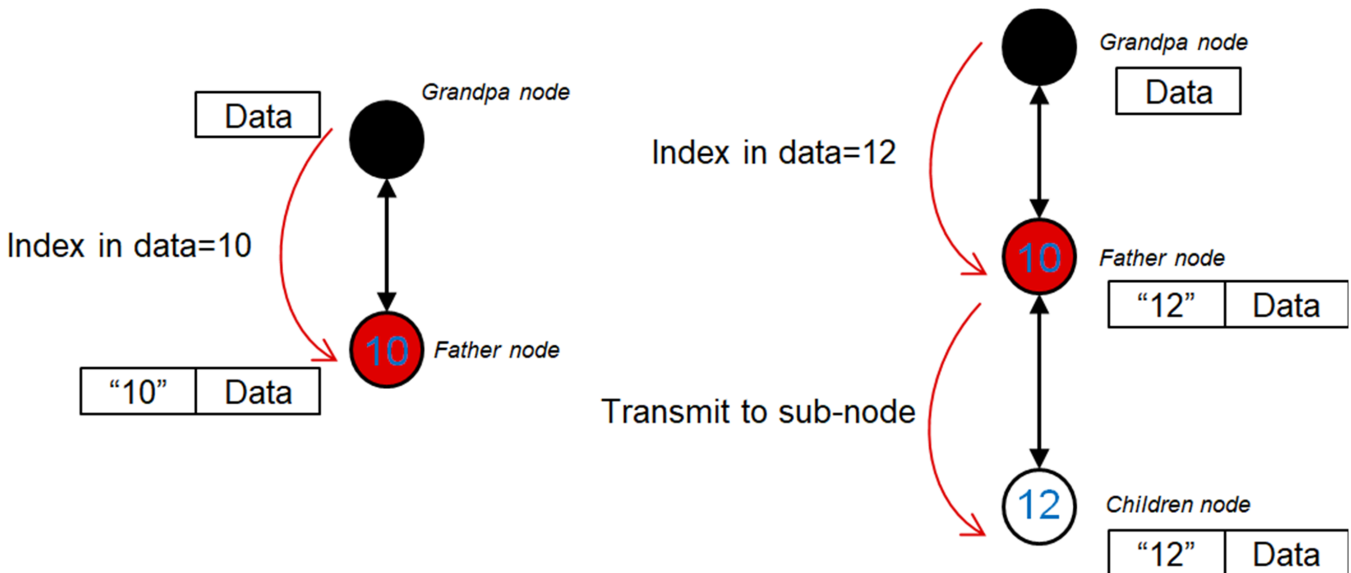


Figure 10. Top-Down Write


```

// Noted that newValue[0] is reserved in the example code
// newValue[2] = '0', message to father, no transmit
if (newValue[2] == '0')
{
    Display_printf(dispHandle, MR_ROW_my_message, 0, "Get message %s", newValue);
    PIN_setOutputValue(ledPinHandle, Board_PIN_RLED, newValue[3] & 0x01);
    // set/reset Led
}
//newValue[2] != '0', message to child, transmit to the appointed child
else if ( (newValue[2] - 48) <= (valid_table_sum - 1) )
{
    child_index = newValue[2] - 48;
    state = multi_role_doSelectConn(child_index);
    if (state == false)
        Display_printf(dispHandle, MR_ROW_doSelectConn, 0, "doSelectConn fail: %d", child_index);
    else
    {
        multi_role_doGattWriteString(connList[child_index].connHandle, newValue);
        // Transmit the data
    }
}
}
    
```

For the bottom-up notify, the father node knows the index of the child node that sent the notification, and the grandpa node knows the index of the father node that sent the notification. For the grandpa node to distinguish which child node that originated the notification, the father node adds the index information in the notification data from the child node before the notification is forwarded to the grandpa node. To distinguish whether the notification's source is the father node or a child node, adding "00" (2 bytes for write data alignment) before the payload data means the notification is from father node. Now, the grandpa node can know where the notification came from. Then, the down-top notification function is finished. For more details, see [Figure 11](#) and .

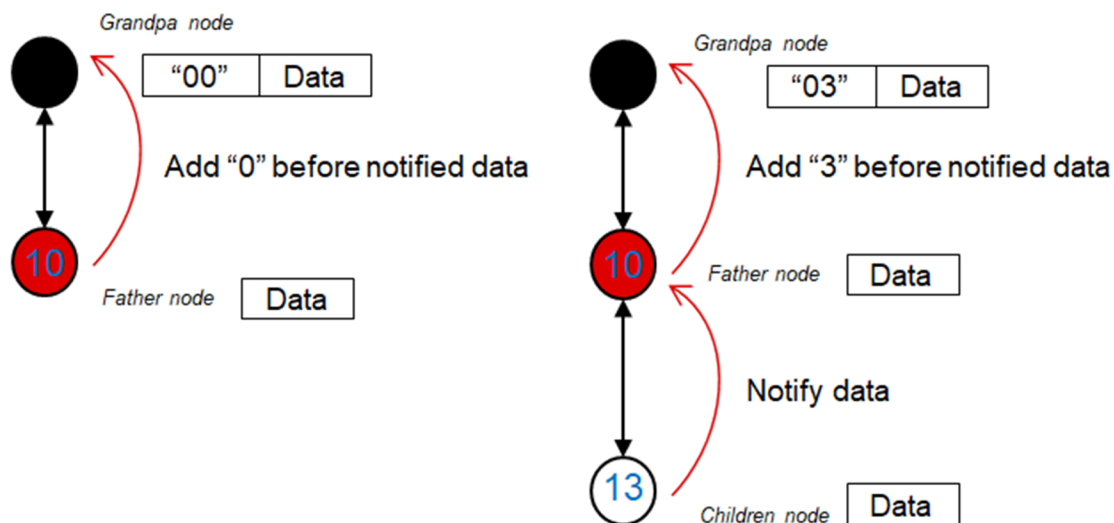


Figure 11. Down-Top Notify

```

// Noted that pMsg->data[0] is reserved in the example code
// order number in tree = connHandle
pMsg->data[2] = '0' + pMsg->connHandle;
memcpy(noti.pValue, pMsg->data, SIMPLEPROFILE_CHAR1_LEN);

//Forward the notification
status = GATT_Notification( grandfather_index, &noti, FALSE );
    
```

6 Bluetooth Low Energy tree Structure Network Test

The overall connection diagram is shown in [Figure 12](#). In this setup, there are four father nodes and each father node is connected to seven child nodes. The key parameters used in this example are shown below:

- Min connection interval = 160
- Max connection interval = 320
- Slave latency = 5

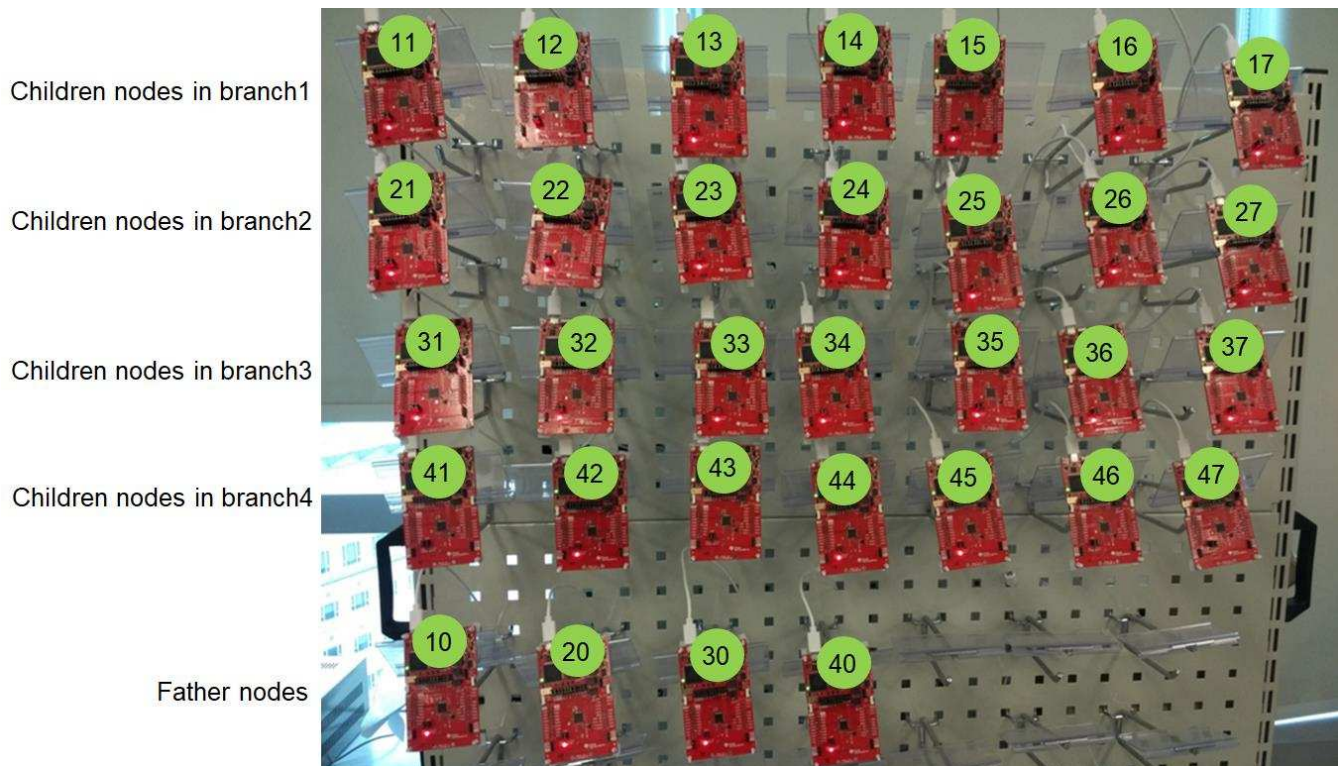


Figure 12. Overall Connection Diagram

The tree structure network write and notify function on CC2652R1F test result is shown in [Figure 13](#). From the information shown in the serial port, the grandpa node writes to each father node and each father node in turn forwards the data to its child nodes. Meanwhile, each child node notifies the data to its father node and the father node in turn forwards each notification to the grandpa node.

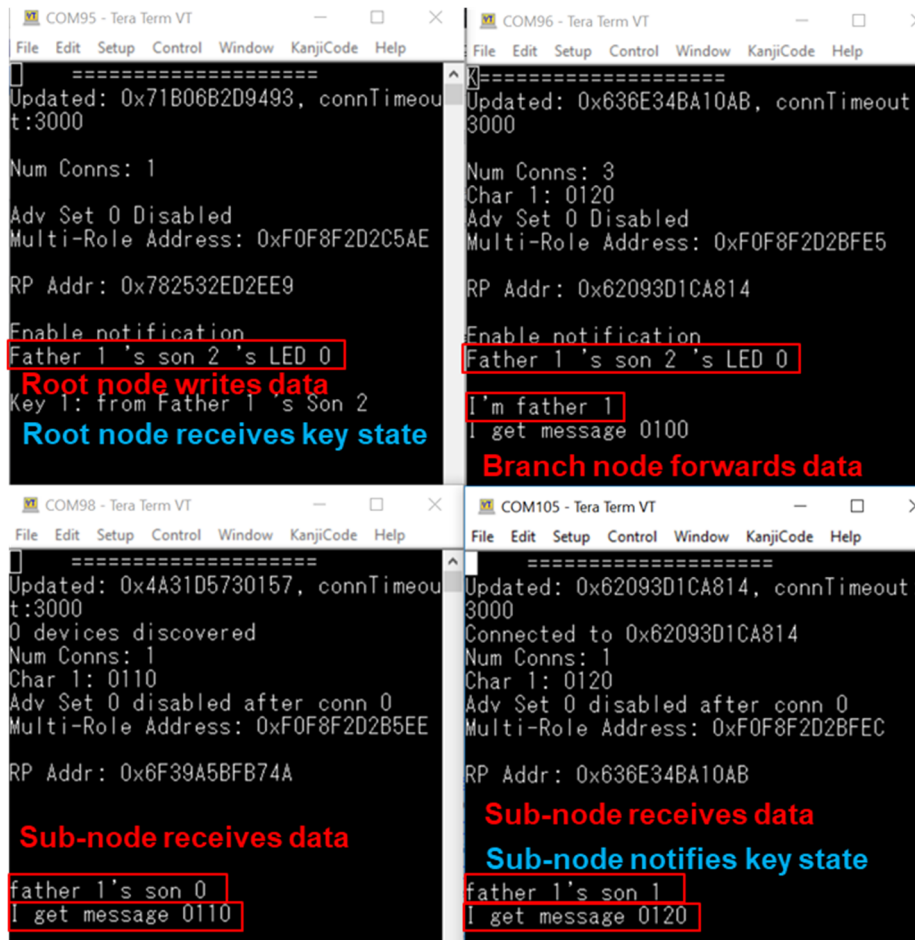


Figure 13. Tree Structure Write and Notify Test

7 References

- [Bluetooth.com, Core specification v5](https://www.bluetooth.com/specifications/core-specification)
- [Bluetooth.com, Supplements to the Core Specification \(CSS\) v5.0](https://www.bluetooth.com/specifications/core-specification-supplements)
- [TI BLE-Stack User's Guide](#)
- [SimpleLink CC13x2/CC26x2 Bluetooth Low Energy SDK Installer](#)
- [TI Packet Sniffer](#)
- [CC13xx, CC26xx SimpleLink™ wireless MCU technical reference manual](#)

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), 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, 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 (www.ti.com/legal/termsofsale.html) 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.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2019, Texas Instruments Incorporated