THE BACH KHOA YOUTHS SCIENCE AND TECHNOLOGY CONFERENCE

Design and Development of a Dynamic Multiprotocol Application for IoT systems

Son Q.Vo, Thu T. M. Nguyen and Thong Q. Phan

***Abstract*— Thread and Bluetooth Low Energy (BLE) are two low-power protocols developed for interconnection among wireless devices within the Internet of Things (IoT) systems. While BLE was born in early 2010, Thread was released by Thread Group Inc. in 2015. This paper provides an overview of Thread protocol with an adequate Thread network topology analysis and a description of BLE and BLE's upper layers. Both protocols have their own advantages, therefore, this paper also discusses a method to combine their benefits of them on a single System-on-Chip (SoC) using a technique called dynamic multiprotocol. Dynamic multiprotocol technique allows a network node to run on BLE and Thread simultaneously. Additionally, by building a service called GATT BLE/Thread forwarding service on top of the BLE and Thread dynamic multiprotocol firmware provided by Nordic Semiconductor, a connection between a Thread-only node and an Android smartphone BLE application can be established with a BLE/Thread-multiprotocol node acting as a forwarder. Remote control and observation of this network system from the Internet or cloud are enabled by using the Message Queueing Telemetry Transport for Sensor Networks (MQTT-SN) protocol. Our work has been successfully implemented and tested on Nordic nRF52840 hardware platform, giving an average delay time of less than 250ms.**

Keywords: Internet of things, IoT, CoAP, IEEE 802.15.4, Thread, BLE, dynamic multiprotocol, MQTT-SN

1 INTRODUCTION

IoT (Internet of Things) has become a highly popular topic. One of the core technologies in IoT is the protocols used to connect devices in the system. BLE and Thread are two protocols defined to solve this problem. They both provide low energy consumption and secure communication among devices.

1.1 Thread overview

Recently, there have been a lot of end devices for customers based on Thread in the smart home market. Thread is an IPv6-based protocol for IoT devices in IEEE 802.15.4 standards [1]. Thread operates in the 2.4GHz band and uses O-QPSK modulation. The default hop limit is 36 hops, while its radio signal range is 30 meters per hop.

Thread devices can be a router-capable device (Full Thread Device - FTD) or an end device (Minimal Thread Device - MTD) regardless of their functions in a network. FTDs are always on to maintain the network performance, while MTDs usually work as sleepy end devices.

A Thread network includes four roles [2]: Thread Border Router, Thread Leader, Router, and End device (Fig. 1)

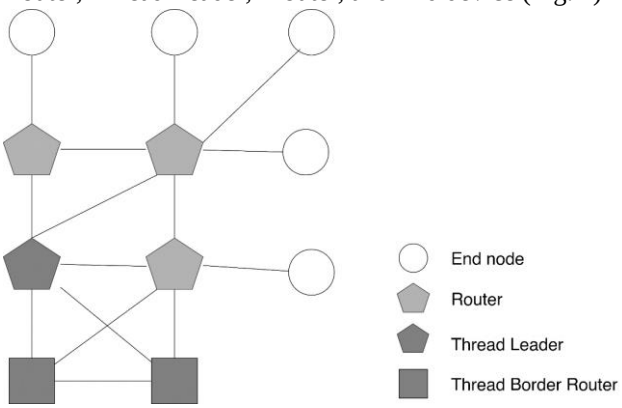

Fig. 1. Thread network topology

A Thread device implements an IPv6 addressing mechanism in RFC 4291 [3]. One Thread node can have multiple IPv6 addresses: Link-Local, Mesh-Local and Global. The IPv6 Global Unicast Address has two parts: prefix (64 bits) and ID (64 bits). This is formed by Thread Border Router and allows a Thread node to communicate with external network through Border Router.

Thread can implement both Constrained Application Protocol (CoAP) and Message Queuing Telemetry Transport (MQTT) at the application level due to low memory and low processing requirement.

In this paper, we use CoAP and MQTT protocols on the application layer of Thread. CoAP is used during local control mode and MQTT is used during remote control mode. Fig. 2 compares how MQTT and CoAP exchange data.

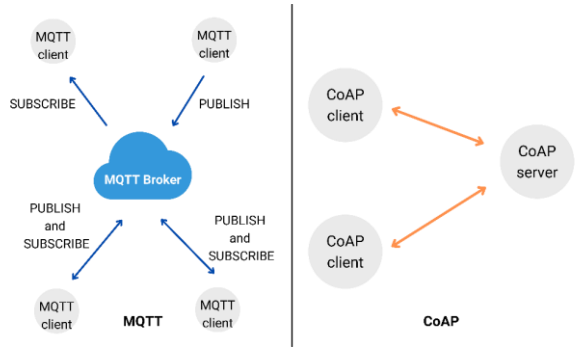

Fig. 2. MQTT and CoAP protocol

1.2 BLE overview

BLE operates in the 2.4 GHz band, utilizing frequencies between 2402 and 2480 MHz. The used spectrum is divided into 40 channels, each employing a space of 2 MHz. These channels are divided into 3 primary advertisement channels and 37 connection-oriented channels. [6]

A BLE device contains three main blocks in its architecture, including the **application**, the **host** and the **controller**. Fig. 3 shows the architecture of BLE along with the layers in each block. Since we mainly focus on exploiting the upper layers of BLE, only the **application** layer, the **Generic Access Profile (GAP),** and the **Generic Attribute Profile (GATT)** are discussed in this section.

Application This is a use-case dependent layer which is built on the top of the BLE architecture. The application handles data received from and sent to other devices and the logic behind it. [8]

Generic Access Profile (GAP). A mandatory framework that defines how BLE devices interact, communicate and exchange data with each other.[8]

Generic Attribute Profile (GATT). This defines the way that two Bluetooth Low Energy devices transfer data back and forth using concepts called Services and Characteristics. Service is a collection of information exchanged in a BLE connection, while Characteristic is a value that defines how exchanged information is presented. [8]

In this paper, we built a BLE/Thread forwarding service on the GATT and application layers of BLE.

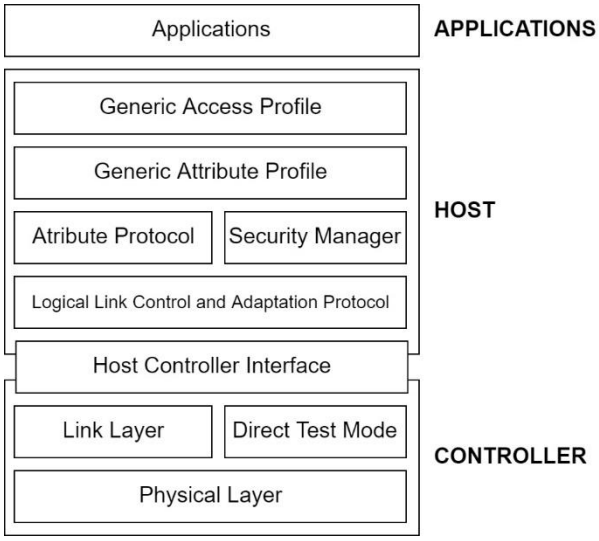

Fig. 3. Architecture of BLE [8]

2 DYNAMIC MULTIPROTOCOL TECHNIQUE: BLE AND THREAD

BLE and Thread have their own advantages. In particular, BLE has its existence in most smartphones in the market but it is not an IP-based protocol and requires complex configuration to enable Internet remote control, while Thread is an IPv6-based protocol that allows a node to easily communicate with outside IP-based networks. As a result, combining these two protocols on one RF System-on-Chip (SoC) can make a node possess both of their advantageous features.

The nRF SoCs of Nordic Semiconductor supports multiple radio protocols. They have protocol support for Bluetooth LE, Bluetooth mesh, Thread, Zigbee, 802.15.4, ANT, and 2.4 GHz proprietary stacks.

There are two methods to implement radio protocols concurrency: switched multiprotocol and dynamic multiprotocol. However, switched multiprotocol technique requires a switching impact between two protocols, while no switching is required in the dynamic multiprotocol technique. Dynamic multiprotocol technique allows a node to establish and communicate on two different radio connections simultaneously. Therefore, in this paper, only dynamic multiprotocol technique is discussed.

In dynamic multiprotocol, radio hardware is time-sliced between all protocols. Each radio protocol requests a timeslot prior to any radio operation. Fig. 4 shows how BLE and Thread are time-sliced in this technique. Dynamic multiprotocol requires concurrent (time-multiplex) radio access.

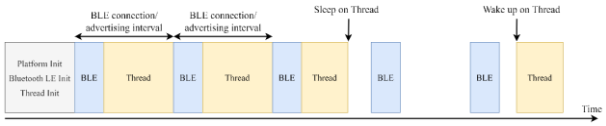

Fig. 4. BLE/Thread Dynamic Multiprotocol [9]

Nordic Semiconductor has provided a dynamic multiprotocol firmware template. In this template, a user writes the Bluetooth part of a multiprotocol application as it was a Bluetooth-only application, and a Thread part of the multiprotocol application as it was a Thread-only application [9]. BLE was configured to have higher priority than Thread in order to maintain an error rate of 0% for BLE. As a result, there are certain Thread packets lost on account of BLE activity, those lost packets are treated as dropped packets. Dropped packets are not uncommon in wireless networks and Thread provides resilience to that [8]. A solution to decreasing the number of Thread packets loss is to prolong BLE timing parameters, e.g. advertising interval or connection interval.

3 IMPLEMENTATIONS

3.1 Hardware description

The system includes three network protocols: Thread, BLE, and the Internet. The topology of the system is shown in Fig. 5. The node roles and the hardware used for each node type in this topology are discussed below.

End node. A Thread-only node, using nRF52840 SoC (Fig. 6a and Fig. 6c).

Network Co-Processor (NCP). A Thread device that is attached to a Border Router to help process OpenThread network data. It can be a nRF52840 Development Kit or nRF52840 Dongle.

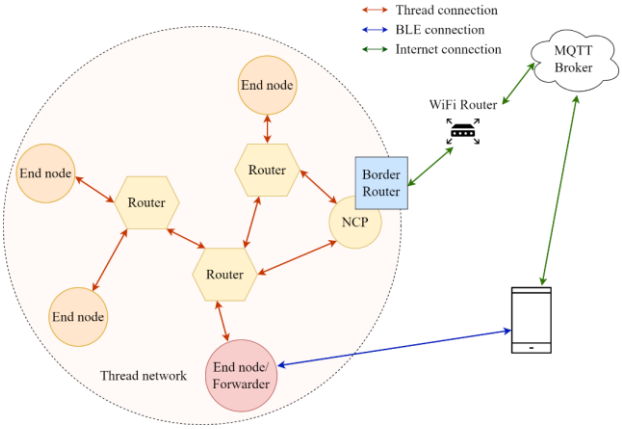

Fig. 5. Network topology

End node/Forwarder. A multi-role node operates as a BLE End node when it receives genuine BLE commands from a BLE connection or operatates as a Forwarder when it receives BLE forwarding packets. A Forwarder is a nRF52840 SoC.

BLE device. A smartphone supports BLE and functions as a BLE central to control a Forwarder. It runs a mobile application to establish BLE connection.

Android mobile application. An application that we built to provide a user interface for controlling the network system. The main features of the Android mobile application are:

- Set up a BLE connection with a BLE end node.
- Perform BLE/Thread forwarding service
- Control the End nodes over BLE.

Border Router. A Router connects a Thread network to other IP-based networks (Wi-Fi or Ethernet) and functions as an MQTT-SN Gateway to *CloudMQTT*. This router is a Raspberry PI 3B running Linux-based OS (Fig. 6b). To fully function as a Border Router, it requires a Network Co-Processor (NCP).

Cloud: CloudMQTT supports Websocket clients to view messages on the web which are pushed from both devices and browsers. Users can control the End nodes from CloudMQTT dashboard [10].

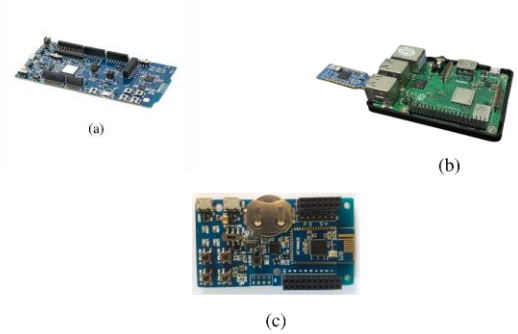

Fig. 6. nRF52840 Development Kit (a), Border Router (b), and nRF52840 Customized Development Kit (c)

4.2 Firmware/Software Implementation

4.2.1. Forwarder

This node operates on both BLE and Thread at the same time. Nordic Semiconductor has provided a BLE/Thread dynamic multiprotocol template firmware for developers to exploit on it. However, in this template, these two protocols are functioning separately. The aim of our project is to establish a bridge between BLE and Thread so that this dynamic multiprotocol node can receive Thread commands from a smartphone via a BLE connection and then forward it to a Thread-only node via a Thread connection using CoAP protocols. To achieve this goal, we added a BLE/Thread forwarding service to the GATT layer of the BLE stack and some functions to the transport layer (CoAP) of Thread. Fig. 7 illustrates the main firmware of a forwarder node, which includes two branches: BLE main and Thread main. Figure 8 shows the forwarding message algorithm.

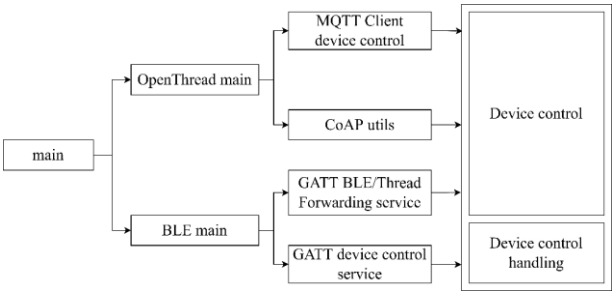

Fig. 7. Forwarder main firmware

The BLE main branch consists of two services: **GATT LEDs control service** and **GATT BLE/Thread Forwarding Service.**

GATT device control service. allows users to control a Forwarder from BLE connection.

GATT BLE/Thread Forwarding Service. forwards messages received from BLE connection to a Thread-only node in the network to CoAP utils module.

The Thread main branch consists of two blocks: **CoAP utils** and **MQTT-client device control.**

CoAP utils. continues to forward messages to Thread end nodes by generating CoAP packets.

MQTT-client device control. enables users to control Thread End nodes from the Internet via a platform called CloudMQTT dashboard.

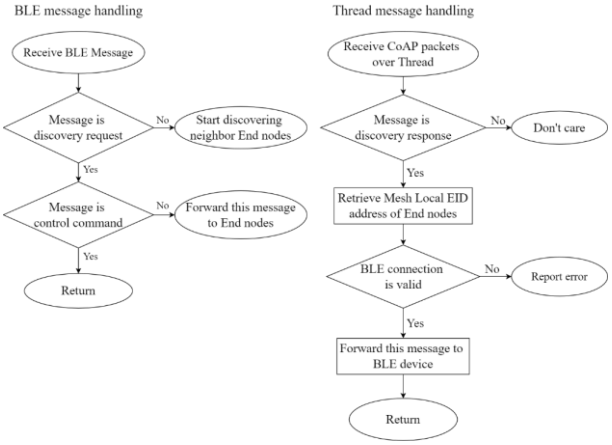

Fig. 8. Forwarding message algorithm on a Forwarder node

4.2.2 Thread End Node

This node operates only on Thread. When it receives CoAP packets from a Forwarder over a Thread connection, it processes, executes, and then sends responses to the Forwarder. This end node can also be controlled from the Internet using the MQTT-SN protocol. Fig. 9 illustrates the CoAP packets handling process of an End node.

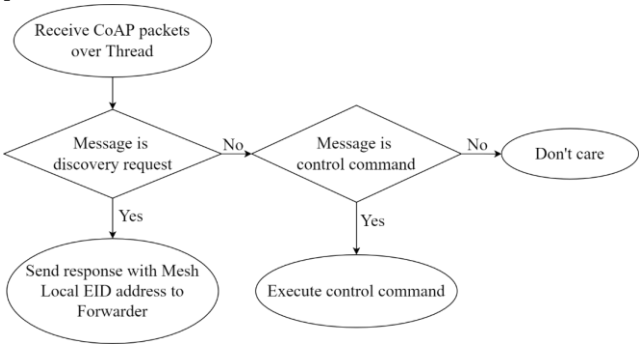

Fig. 9. Handling forwarding messages on an End node

5 EXPERIMENTAL RESULTS

5.1. Testing Environment and Procedures

Validation test. verifies all functions of the network in a practical environment. The experiment is established as described in Fig. 5.

Maximum range test. finds the maximum line-of-sight range that a BLE-to-BLE and a Thread-to-Thread connection in our system can achieve.

Optimal range test (Packet Reception Ratio (PRR) > 95%). finds the line-of-sight range that the PRR of a BLE-to-BLE and a Thread-to-Thread connection can be in the range between 96% and 100%.

Multi-hop test. Fig. 10 illustrates the implementation of two schemes for this test with parameters in Table 1. The first scheme (Fig. 10a) is conducted outdoors, in an environment with few obstacles, while the second one (Fig. 10b) is conducted in a real household setting. The multi-hop test is designed to evaluate the PRR, Delay Time and Round-trip Time (RTT) per-hop in the environment.

TX power	0 dBm
Sensitivity	-95 dBm
BLE advertising packet cycle	100 ms

Table 1. Radio specification of a node [9]

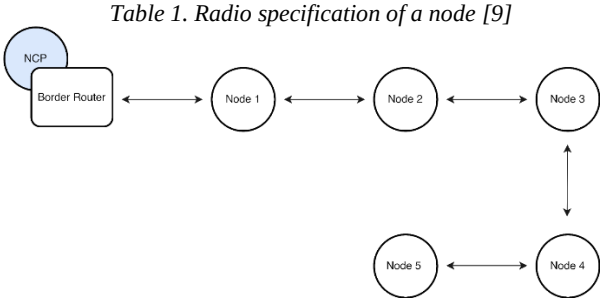

Fig. 6a. Multi-hop test – Scheme 1: outdoors, few obstacles

Fig. 10b. Multi-hop test – Scheme 2: real house

Remote control test. When users are not in the BLE connection range, they can control home devices or update information via an MQTT dashboard on their smartphones. This experiment aims to simulate practical usage of the network to evaluate the performance of MQTT.

BLE/Thread Forwarding service test. When users have no internet connection to control nodes over Cloud, they can control home devices using BLE/Thread Forwarding Service on their smartphones. This experiment aims to simulate practical usage of the network to evaluate the performance of a Forwarder node.

Power consumption evaluation. measures the total average current of an End node and a Forwarder to estimate the battery life of a network nodes as well as find way to optimize the current firmware. Fig. 11 illustrates how to measure the current of the nRF52840 DK.

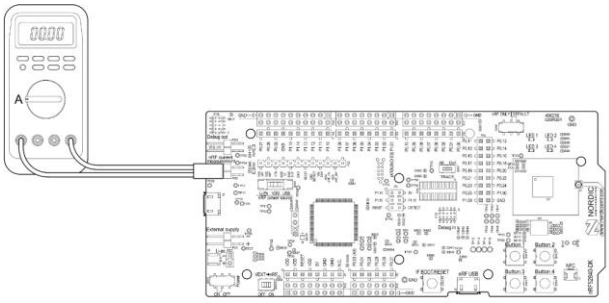

Fig. 11. Using an ampere meter for current measurement [9]

5.2 Performance and Results

Validation test result. The Android application can successfully establish a BLE connection with a Forwarder

node and perform a forwarding test to control End nodes. The CloudMQTT dashboard also allows users to control a Thread End node over the Internet.

Fig. 12 represents an example of the Thread packets pinging process from a Thread End node to a Forwarder node, in which the Forwarder node was simultaneously exchanging data with the Android app via a BLE connection. Fig. 13 and Fig. 14 respectively show the User Interface of the mobile application and the CloudMQTT web dashboard.

Maximum range test result. Table 2 shows maximum ranges of different hardware running BLE and Thread. Based on this result, it is concluded that a BLE/Thread Forwarder can scan Thread end nodes within 50m or 75m depending on Tx Power setting (in an environment with few obstacles). And Thread end nodes can exchange data packets within 37m or 60m depending on Tx Power setting.

Hardware	**Protocol**	**TX Power (dBm)**		
		0	4	8
nRF52840 Customized Kit	Thread	37m	48m	60m
nRF52840 Development Kit	Thread	40m	52m	63m
nRF52840 Customized Kit	BLE 1Mbps	51m	62m	74m
nRF52840 Development Kit	BLE 1Mbps	53m	65m	79m

Table 2. Maximum range test result

Optimal range test result (Packet Reception Ratio (PRR) > 95%). Table 3 shows the ranges that give the best PRR – within 16-19m. We used result from this test for setting multihop test.

Harware	**Protocol**	**Packet Receipt Ratio - PRR (%)**				
		100	99	98	97	96
CC2538 Customized Kit	Thread	9m	12 m	13 m	15 m	16 m
nRF52840 Customized Kit	Thread	10m	11 m	13 m	15 m	17 m
nRF52840 Development Kit	Thread	11m	12 m	15 m	17 m	19 m

Table 3. Optimal range test result

Multi-hop test result.

Scheme 1: Table 4a shows that the achieved PRR is good (> 95%) and RTT is low (< 40ms).

Node	Hardware	Firmware	Distance (m)	RTT (ms)	PRR (%)
1	CC2538	Thread-only	10m – 1 hop	14.23	100
2	nRF52840	Dynamic multiprotocol	20 – 2 hops	19.43	100
3	CC2538	Thread-only	30 – 3 hops	25.12	99
4	nRF52840	Dynamic multiprotocol	40 – 4 hops	28.38	98
5	CC2538	Thread-only	50 – 5 hops	35.94	96

Table 4a . Scheme 1 test result

Scheme 2: Real House – Table 4b shows the achieved result of good PRR (> 90%) and RTT (< 35ms)

Node	Firmware	Distance (m)	RTT (ms)	PRR (%)
Router A	Thread-only	10 – 1 hop	18.73	96
Child B	Thread-only	4 – 2 hops	34.13	90
Child C	Thread-only	3 – 2 hops	32.14	92
Router D	Dynamic multiprotocol	10 – 1 hop	15.12	98
Child E	Dynamic multiprotocol	8 – 2 hops	30.63	94

Table 4b. Scheme 2 test result

Remote control test result. Table 5 describes the PRR and the average RTT from the test, proving that users can immediately see information updates when controlling home devices via MQTT protocol.

PRR	**Delay time**	**RTT**
94%	175 ms	350 ms

Table 5 . Remote control test result

BLE/Thread Forwarding service test result. Most of the packets lost were from the connection between Forwarder and Thread end node (Table 6). This is because on dynamic multiprotocol firmware, BLE has higher priority over Thread.

Route	**PRR**	**Delay time**
Smartphone -> Forwarder: 5m	100%	6.2 ms
Forwarder -> Thread: 10m	96%	15.3ms

Table 6 . BLE/Thread forwarding service test result

Power consumption evaluation. By default, nRF52840 DK using a coin cell battery with 3V/100 mAh. From Table 7, we can see that forwarder nodes have to use more power due to complex functionality.

Node	Idle State	Average Current	Estimated Working Time
End node	2.7 uA	42 uA	4600 hours ~ 190 days
Forwarder	3.1 uA	215 uA	900 hours ~ 36 days

Table 7 . Power consumption test result

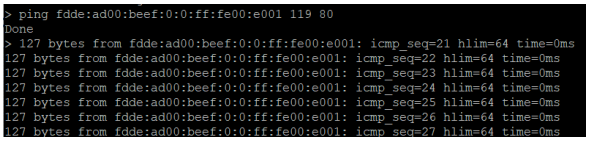

Fig. 12. Successful ping result between two nodes after establishing the network

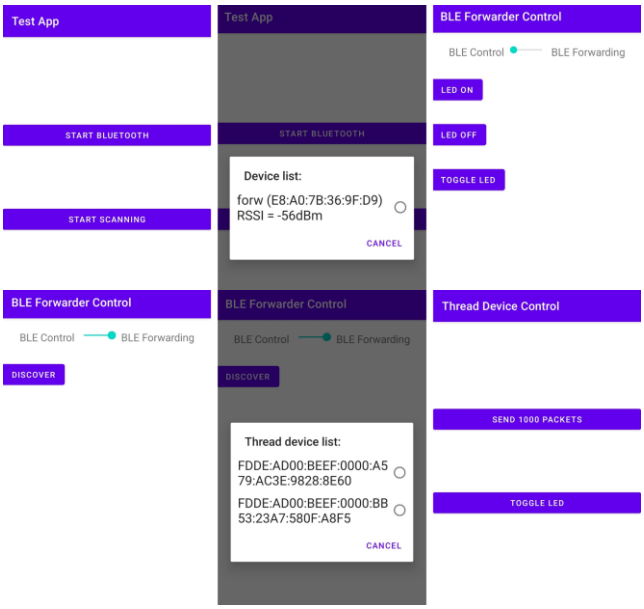

Fig. 13. Android App user interface

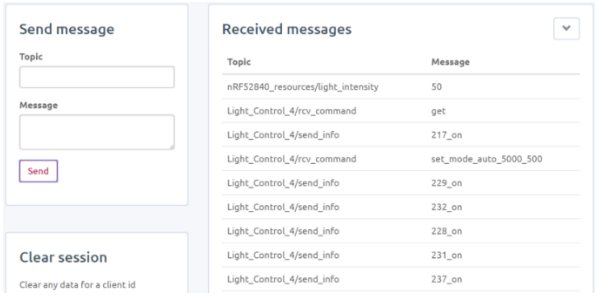

Fig. 14. Websocket UI displays CoAP controlling messages for LEDs controller.

6 CONCLUSION AND FUTURE WORK

In this paper, we discussed Thread, BLE, and a method to combine these two protocols on one single Radio Frequency (RF) SoC: dynamic multiprotocol. With dynamic multiprotocol, a SoC can simultaneously form a BLE and a Thread connection. Multiprotocol takes advantage of the Thread network (based on IPv6) for a robust routing mechanism. From the test results, the PRR and RTT are efficient enough for application in a radius of 20 to 30 meters. This explains why Thread is designed for smart home IoT applications.

In Nordic Semiconductor dynamic multiprotocol firmware template, BLE application and Thread application function separately from each other. As a result, this paper also provides an additional BLE Service: GATT/BLE Thread forwarding service to the available dynamic multiprotocol firmware so that a mobile application can interact with a Thread end node with a multiprotocol node acting as a forwarder. We implemented our work on real hardware and created some test cases to verify our system's quality.

Currently, we are working on adding sensors and actuators to the network to enable more automatic features in our current system.

REFERENCE

[1] I. Unwala, Z. Taqvi and J. Lu, "Thread: An IoT Protocol," 2018 IEEE Green Technologies Conference (GreenTech), 2018, pp. 161-167, doi: 10.1109/GreenTech.2018.00037.

[2] *OpenThread*. [Online]. Available: https://openthread.io/. [Accessed: 15-Feb-2022].

[3] RFC 4291, “IP Version 6 Addressing Architecture”, https://datatracker.ietf.org/doc/html/rfc4291

[4] RFC7252, “The Constrained Application Protocol (CoAP),” https://datatracker.ietf.org/doc /html/rfc7252

[5] M. U. H. Al Rasyid, F. Astika and F. Fikri, "Implementation MQTT-SN Protocol on Smart City Application based Wireless Sensor Network," 2019 5th International Conference on Science in Information Technology (ICSITech), 2019, pp. 7-12, doi: 10.1109/ICSITech46713.2019.8987546.

[6] M. Baert, P. Camerlynck, P. Crombez and J. Hoebeke, "A BLE-Based Multi-Gateway Network Infrastructure with Handover Support for Mobile BLE Peripherals," 2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), 2019, pp. 91-99, doi: 10.1109/MASS.2019.00020.

[7] Mohammad Afaneh, *Intro to Bluetooth Low Energy*, 2018. [Online] Available: Novel Bits e-book

[8] Bluetooth Core Specification: 5.1, Bluetooth Special Interest Group, Kirkland, Washington, U.S., 2019.

[9] Nordic Semiconductor, "Multiprotocol support with BLE/Bluetooth," *nRF5 SDK for Thread and Zigbee*, 10.07.2020. [Online]. Available: https://infocenter.nordicsemi.com/topic/sdk_tz_v4.1.0/ble_154_multiprotocol.html. [Accessed: 15-Feb-2022].

[10] *CloudMQTT*. [Online]. Available: https://www.cloudmqtt.com/. [Accessed: 15-Feb-2022].

Authors' information:

Ph.D.Vo Que Son
Ho Chi Minh University of Technology
Email: sonvq@hcmut.edu.vn
Mobile phone: (+84) 908 259 522

Nguyen Thi My Thu
Ho Chi Minh University of Technology
Email: thu.nguyen2007@hcmut.edu.vn

Phan Quang Thong
Ho Chi Minh University of Technology
Email: thong.phandn@hcmut.edu.vn