IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 23, Issue 6, Ser. III (Nov. –Dec. 2021), PP 26-32
www.iosrjournals.org

Implementing Associated Routing Protocol for Bluetooth Low Energy Devices

Ahmed Said[1], Atef Galwish[2], Mohamed Assal[1], Mohamed Elgazar[3]
[1]*Faculty of Computers & Artificial Intelligent, MTI University, Egypt*
[2]*Faculty of Computing & Artificial Intelligent, Helwan University, Egypt*
[3]*Vodafone, Egypt*

Abstract:
Bluetooth Low Energy (BLE) is an extensively used technology for short range communications and Internet of Things (IoT). BLEs are currently vastly gaining popularity, and which might become an enabler for implementing in practice the Internet of Things (IoT) concept. In this paper an associated routing protocol for BLE devices is proposed. Furthermore, the protocol implemented and tested using real Bluetooth SoCs including the Broadcom BCM2837 chipset which is used in the Raspberry Pi 3. The Experiments demonstrate the routing delay and throughput using networks containing different numbers of nodes to demonstrate the impact of network size on performance.

Keywords: *Bluetooth Low Energy, BLE, IoT, Routing, Connectivity, Mobility*

Date of Submission: 05-12-2021 Date of Acceptance: 19-12-2021

I. Introduction

Bluetooth Low Energy (BLE) is a short-range wireless transmission technology targeting low-cost, low complexity communication. This technology was released in the beginning under the name "Wibree" and was later joined into the main Bluetooth standard in 2010 as an enhanced feature of Bluetooth when Bluetooth Core Specification version 4.0 was adopted [1][2]. BLE technology opens entirely new markets for devices featuring low cost and low power wireless connectivity. This technology was created to extend the life of battery-powered gadgets by allowing short bursts of data transmissions rather than continuous streams, which deplete the battery. BLE is a vital technology to support the anticipated market growth in wearable devices. Progressively, BLE devices can be found now in many application domains such as fitness, home automation, etc. and it gradually making their way into people's daily lives. Due to power and battery constraints, this class of devices can only communicate with each other at a short range. An important research topic is how to use those devices to form a network and share information between those devices [3].

The previous studies have proved that the BLE technology is potentially capable of providing higher throughput i.e. in [4][5] and is more energy efficient in [6][7] than such state-of-art technologies as ZigBee. Another important advantage of the protocol is that already today there are plenty commercial smart phones, tablets, and single board computers such as Raspberry Pi & Dragon Board which integrate the dual-mode radio transceivers supporting both the communication over classical Bluetooth and BLE. This enables easy connectivity of the BLE-based sensors and the Bluetooth Smart Ready devices and might enable several innovative exciting applications thus playing crucial role in bringing the Internet of Things (IoT) concept to life.

In many previous works, the authors were investigating the peer-to-peer (P2P) BLE links, leaving aside the multi-node scenarios. Among the major reasons for this arethe complexity of the BLE communication, which impedes its analysis and the absence of the tools capable of simulating the BLE networks [8].

In our previous work[9], a novel simulation was presented to simulate the proposed associated routing protocol in BLE networks. This paper focus on implementing the routing protocol using real BLEs devices such as Raspberry Pi 3, dragon board, smart phone & beacons.

II. BLE Technology

The BLE protocol stack consists of two main components, BLE Controller and BLE Host (see Figure 1(a)). These two either reside on the same physical device or might be implemented by different devices. The Controller is a logical entity which is responsible for Physical (PHY) layer and LL. The Host implements

functionalities of the upper layers. They include: L2CAP, GAP, ATT, GATT and SM. The Logical Link Control and Adaptation Protocol (L2CAP) multiplexes packets of the upper layers, manages connection establishment, configuration and destruction and might support packet segmentation. The Security Manager (SM) handles pairing, authentication, bounding and encryption of BLE communication. The Attribute protocol (ATT) specifies mechanisms for discovering, reading, and writing attributes on a peer device, and the Generic Attribute profile (GATT) provides framework for discovering services and for reading and writing characteristic values. In the end, the Generic Access Profile (GAP) interfaces application and BLE stack. Host and controller are connected through the standardized Host Controller Interface (HCI) [1].

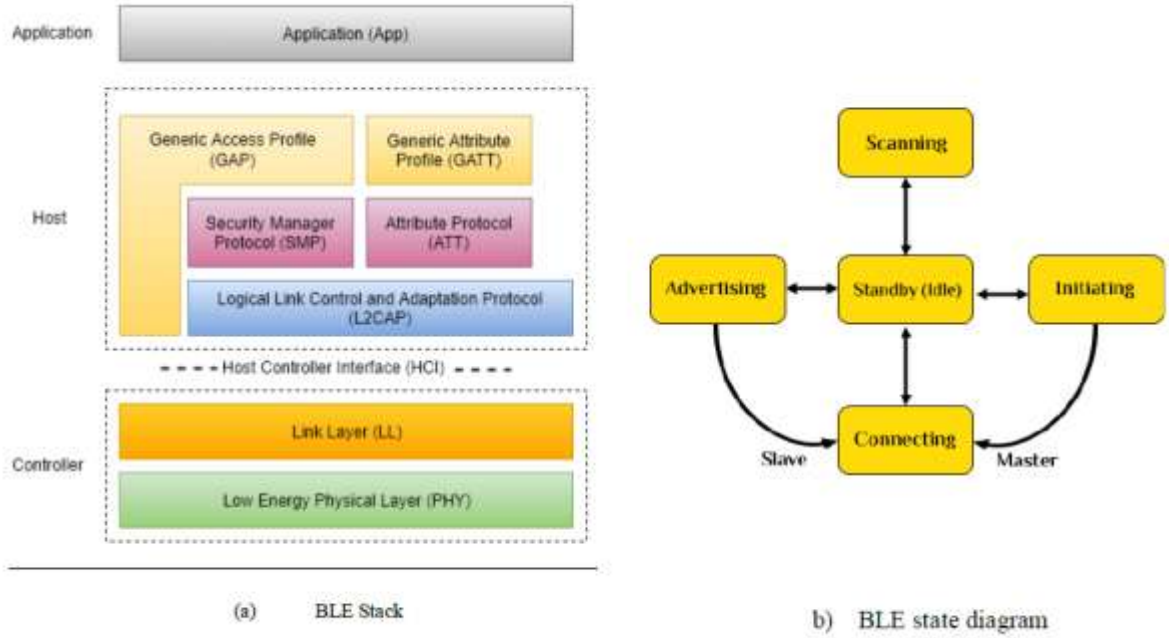

Figure 1. BLE Technology Fundementals

In order to simplify the design and to minimize the cost of transceivers, BLE PHY is based on Gaussian Frequency Shift Keying (GFSK) modulation. To simplify the implementation even further, BLE communication relies on very short data packets with the maximum packet size of just 47 bytes. The data transmission between BLE devices is bound to time units known as advertising and connection events.

The advertising events are used by BLE devices to broadcast small blocks of data or to agree on parameters of a connection to be established in data channels. In the beginning of an advertising event an advertiser, i.e., a device which either has some data to transmit or which expects to get some data, sends an advertising frame. Latter instance, after transmitting a frame, the advertiser switches to receive and waits for possible connection establishment requests. If the connection request from a device (which is referred to as an originator) is received, the two devices start peer-to-peer connection in data channels .

Once a connection in data channels has been created, the initiator becomes master, and the advertiser becomes slave (see Figure 1(b)). The communication in each connection event is started by a master, which sends a frame to a slave. The master and the slave alternating sending data frames on the same data channel while at least one of the devices has data to transmit or until the current connection event ends. In the case of either of the devices receives two consecutive frames with Cyclic Redundancy Check (CRC) errors, the connection event is closed. Once a connection event is closed, both master and slave might switch to low-power sleep mode until the start of the next connection event. BLE always assumes that a master is typically more complex and richer on resources than a slave.A crucial part of BLE stack is the client-server mechanisms enabling the discovery of available services, applications and providing the devices with the universal mechanism for data exchange. Those are implemented by ATT, GATT and GAP layers. more details on the practical issues can be found e.g., in [10].

III. BLE Multi-Criteria Routing Protocol

In this section, a fully distributed routing protocol based on an associative routing framework presented in [11]. Associative routing replaces the semantics-free destination addresses with semantics-rich *destination*

descriptors that dynamically bind to nodes at **routing time**. Destination descriptors provide qualitative descriptions of the destination nodes where a packet should be delivered.

Associative routing does not require nodes to have unique identifiers, instead nodes identify themselves by the services they are willing to provide, resources they are willing to share, data they store, or any other attributes of relevance to applications. there is no direct mapping between the data packet destination specified by the sender and the physical nodes the packet will be delivered to. Multiple different destination descriptors may lead to the same node (or same group of nodes). Also, at different times, the same destination descriptor may lead to different nodes (or groups) as their attributes change. For example, the battery energy level of a node could be part of its identification profiles, so if the energy level dropped, the node disqualifies for a descriptor that it used to qualify for in the past. Such highly dynamic nature of addressing in associative routing makes it an attractive solution to routing in large scale networks of dynamic topologies, like sensor networks. Figure 2 illustrates identification and addressing in associative routing.

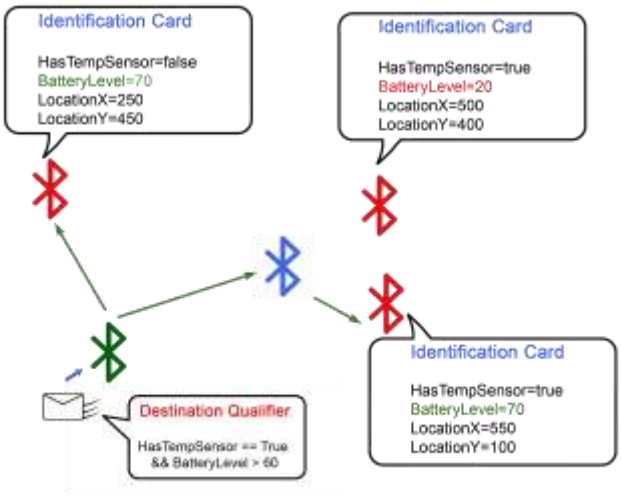

Figure 2. Associative Routing Identification and addressing

The routing protocol adopts the targeted messaging model, which allows the exchange of messages between a source node and a targeted group of nodes. Each node defines its identification profile. When a source node needs to reach a targeted group of nodes, its application agent defines a destination descriptor predicate called Target Predicate (TP) specifying its targeting criteria such as $(hasTemperatureSensor == true \&\& BatteryLevel > 60)$. The application agent combined the defined destination descriptor along with any application data (payload) required in a Targeted Message (TM) then passed it to the transport agent.

The routing protocol uses criteria indexes to speed up routing of messages to frequently targeted groups. An Index Working Set (IWS) by router nodes containing actively used indexes. The routing protocol relies on six routing information tables for its operations.

Table 1: Protocol routing information tables

Name	Fields
Identification Profile Table	Criterion, Value, LastUpdated
Index Definition Table	Index, Properties, Segmented, SegmentSize, SeqNo, Time
Index Routing Table	Index, Criterion, Value, Time, Gateway
Index Update Table	Index, SeqNo, Source, Cost, Gateway, Time
Response Routing Table	Source, Cost, SeqNo, Gateway, Time
Message Routing Table	Source, PredID, Cost, SeqNo, ResHash, Gateway, Time

Device Identification Card

A dictionary or collection of key/value pairs representing the device profile including all its attributes, capabilities and all its information. The Collection must implement the comparable interface so it can be used later to compare with the Destination Qualifiers.

Destination Qualifier

A predicate that describes one of more criteria that can match one or more device depending on the device identification cards and the values in the predicate. This predicate is serialized into a binary format attached to the BLE Packet header during the packet construction and will represent the device associative address. Once a device receives a message it isolates the predicate and validate it against it identification card.

Device Index Broadcast

Indices are stored by devices to optimize or speedup the routing process, these indices is checked by the devices a fixed amount of time looking for aging indices (didn't update for an amount of time). Any matching index must be sent to the resource device gateway to keep the gateway updated with its attributes. The BLE packet sometime can't hold all the data representing the device profile. So, the protocol must support the index segmentation to allow the sending of the index in a suitable size chunk (or segments).

Each segment is given a number which is recorded along with its size. All this information must be recorded and maintained by the routing device once it receives an update from a neighbor according to the following.

1. The device checks its table if it contains the index, the value will be updated. Also, it must update the Last Update field.
2. If the device can find the index, it will insert new records to its table.
3. The routing device must update path cost accordingly.
4. Every fixed amount of time, each device will be required to asynchronously broadcast this information to its neighbors.

Message Path Discovery

Whenever a device wants to send a target message. The device must send a Multicast Path Discovery Packet to all its gateway. The packet must include a sequence number generated by the routing controller and the targeting predicate. When the path discovery packet gets received by a device, the receiving device extract the destination qualifier and construct the predicate evaluation tree and then it compares it with its identification card.

If the comparison result is true, a unicast path setup packet must be sent back to the sender device by the receiving device. The receiving device will become a sending device, it must copy the received predicated into its response packet and set the source device to itself. Also, the response message must contain the device identification card, Resource hash and rank.The Resource had must be unique for each device and/or resource, this can be achieved by including the device spatial position in the Hash calculation function. Also, the hashing function must maintain a resource ranking mechanism.

Lastly, the device will reverse the path constructed during the discovery to avoid the rediscover of the response path whenever possible unless one or more devices left the network during this time. At this time, the response path will be discovering the same way the message (request) path discovers done the response sending process is done. The uniqueness of the *ResourceHash* is achieved by incorporating information like resource location. The reverse path is constructed using the following algorithm:

```
if d is a resource device then
        if predicate(d) == true then
            p ← new response packet
            ResouseHash ← calculate unique resource hash
            send p to s through d
        end if
else if d is a routing device then
        p.Cost ← p.Cost + link cost(d, s)
        for all x ∈ neighbors(d) and x != d do
            if predicate(x) == true {Using IndexRoutingTable of d} then
                {update reverse path}
                for all path ∈ ResponseRoutingTable and path.Source == p. Source do
                    if path.Cost ≥ p.Cost or path.Time ≤ (now() − e)) then
                        delete path from ResponseRoutingTable invalidate existing record
                    else
```

```
                        continue to next neighbor
                    end if
                end for
                path ← new record in ResponseRoutingTable
                path.Source ← p. Source
                path.Cost ← Cost
                path.Time ← now()
                forward p packet to x
            end if
        end for
end if
```

On the other hand, when a unicast response packet received by a device, it creates a forward path as follows

```
p.Cost ← p.Cost + link cost(s, d)
record found ← false
for allpath ∈MessageRoutingTable and path.PredID == p.PredID
        and path.ResHash == p.ResHash
do
        ifp.Cost <path.Cost then
            path.Gateway ← d;
            path.Time ← now();
            path.Cost ← p.Cost
        end if
        record found ← true
end for
if record found = false then
        path ← new record in MessageRoutingTable
        path.Source ← p Source;
        path.PredID ← p.PredID;
        path.ResHash ← p.ResHash;
        path.Cost ← p.Cost
        path.Gateway ← d
        path.Time ← now()
end if
ifd != p.DestNode then
        for allx ∈neighbors(s) and x !=d do
            if∃ t ∈ResponseRoutingTable where x = t.Gateway and t.Source = p.DestNode then
                forward p to d
            end if
        end for
elseif predicate limit exists then
        LimitCutoffValue ← calculate cutoff value
end if
```

Message Routing

One the message path is discovered the message can be send to the destination devices by the sender device through its gateway via the discovered path. When a message received by a routing device, the device checks its Message Routing Table for any matching predicates and higher Resource Rank. If the device found any matching devices, its forwards the message to all devices specified as its gateway.

The Application Controller in called once a device receive a message that requires a response, after processing the message it sends the result back to the sender using a unicast packet. If the receiving device is a routing device it will check its Response Routing Table for any matching devices and will forward the message to them. The previous process is repeated until the response message reach to the sender as shown in figure 2.

IV. Experiments and Discussions

This section discusses the experimental results. Instead of simulating the reults on platform such as MiXiM framework [12] which is based on the popular OMNeT++ engine. Our prottocol is implemented on real hardware using number of different devices which all has BLE chipset supported the Bluetooth Standard 4.1 or later. The expirement used 2 Raspberry Pi 3 boards, 1 Dragonboard 410C, Beacons & an android smart phone.

The use of real hardware produces more accurate results when compared to the use of simulation which don't fully support some critical features of Bluetooth specification and cannot mimic real complexities such as clock synchronization. Raspberry Pi 3 [13] (Figure 3 (a)) and DragonBoard 410c [14] (Figure 3 (b)) used as routing nodes, Beacons are used as resource (targeting) nodes, while the android smart phone is used as a routing node and to collect the results.

(a) Raspberry Pi 3 Board (b) Dragonboard 410c Board

Figure 3. Development Boards

Location-aware affinity propagation (LAP) algorithm [15] is used to build a fully connected BLE network. A set of experiments have been performed in order to evaluate the quailty of our network formation and routing approach. In each experiement a number of BLE nodes are placed randomly in an area of 4x10 meters. On all experiments the number of BLE nodes diverge from 3 to 8. In each experiment, the android smart phone act as the source device. Each experiment is repeated 10 times to calculate the average message delay and average network throughput metrices are measured.

In all experiments, the network achieves full connectivity between all the BLE nodes. While the delay in the network vary depend on the network of BLE nodes placed in the experiment. The delay in a network is the time taken for a data packet to traverse through the network and reach the destination. The average delay for a source node to send a targeted message to a destination node is shown in Figure 4. Figure 5 shows the average network throughput achieved using the porposed routing protocol. Figures 6 & 7 ssshow the difference between the real devices experiments & simulated experiments for both Delay and Network Throughput.

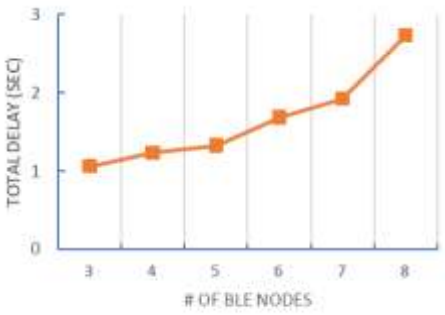

Figure 4 Average Delay in seconds using real devices

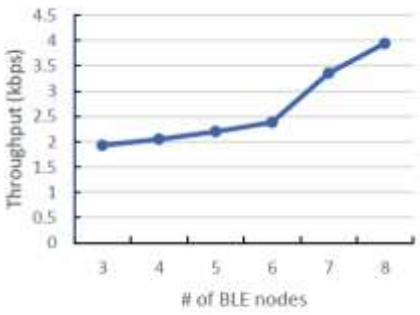

Figure 5 Network Throughput using real devices

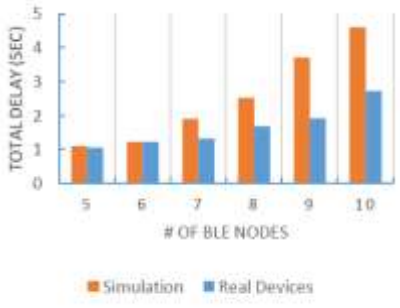

Figure 6 Average Delay real devices vs simulation

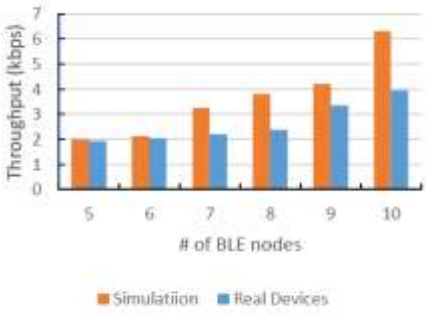

Figure 7 Network Throughput using real devices

V. Conclusions

The paper described an implementation of an Adaptive Multi Criteria Routing Protocol for BLE devices. The expirements on real hardware produces more accurate results when compared to the use of simulation which don't fully support some critical features of the Bluetooth specificition. However, the protocol successfully route the targeted message in all experiments with a great network throughput and reasonable delay. In future work, experiments may take the energy consumption and load on the devices into consideration to make the protocolo more efficient in realworld usage scenarios.

References

[1]. Bluetooth Special Interest Group, Bluetooth specification version 4.1, Kirkland, WA, USA: Bluetooth Special Interest Group, 2013.
[2]. I. G. H. L. T. a. X. C. Z. Guo, An on-demand scatternet formation and multi-hop routing protocol for BLE-based wireless sensor networks, IEEE, 2015.
[3]. T.-C. W. G. C. S. A. R. Muhammad Rizwan Ghori, Optimization of the AODV-Based Packet Forwarding Mechanism for BLE Mesh Networks, Electronics, 2021.
[4]. D. B. P. K. C. S. Philip Dost, "Multi-Purpose Communication Protocol for Wired and Wireless Sensor Networks with Actuators," in *Industrial Technology (ICIT) 2019 IEEE International Conference*, 2019.
[5]. R. V. F. John Dian, "Formulation of BLE Throughput Based on Node and Link Parameters," *Canadian Journal of Electrical and Computer Engineering* , vol. 43, no. 4, pp. 261-272, 2020.
[6]. N. P. J. T. Konstantin Mikhaylov, "Performance Analysis and Comparison of Bluetooth Low Energy with IEEE 802.15.4 and SimpliciTI," *Journal of Sensor and Actuator Networks 2*, vol. 2, no. 3, pp. 589-613, 2013.
[7]. S. H. S. T. a. J. S. A. Dementyev, "Power consumption analysis of Bluetooth Low Energy, ZigBee and ANT sensor nodes in a cyclic sleep scenario," *2013 IEEE International Wireless Symposium (IWS)*, pp. 1-4, 2013.
[8]. K. Mikhaylov, "Accelerated Connection Establishment (ACE) Mechanism for Bluetooth Low Energy," *IEEE 25th International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1264-1268, 2014.
[9]. A. Said, A. Galwish, M. Assal, M. Elgazar, "Simulating Associated Routing Protocol for Bluetooth Low Energy Devices". IOSR Journal of Computer Engineering (IOSR-JCE) Volume 23, Issue 6, Ser. II
[10]. R. Heydon, Bluetooth Low Energy: The Developer's Handbook, Pearson, 2012.
[11]. R. M. Eltarras, "BioSENSE: Biologically-inspired Secure Elastic Networked Sensor Environment," Faculty of the Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 2011.
[12]. R. K. K. a. V. K. Malothu, "MIXIM framework simulation of WSN with QoS," in *016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, 2016.
[13]. R. P. Foundation, "Raspberry Pi Documentation," [Online]. Available: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#raspberry-pi-3-model-b. [Accessed 10 November 2021].
[14]. Qualcomm, "DragonBoard 410c User Manual," [Online]. Available: https://www.96boards.org/documentation/consumer/dragonboard/dragonboard410c/hardware-docs/hardware-user-manual.md.html. [Accessed 10 November 2021].
[15]. M. Elgammal and M. Eltoweissy, "Location-aware Affinity Propagation Clustering," in *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, 2009.