BlueSkyNet: BLE Multi-Hop Network Management Architecture

Makito Kano
Intel Corporation
makito.kano@intel.com

ABSTRACT

Bluetooth Low Energy (BLE) is aimed for Internet of Things (IoT) devices with limited battery capacity and small bandwidth. Despite the increasing number of IoT devices and the improving performance of BLE, the usage model is still limited to point-to-point with no mobility. This limitation prevents the devices to spread across a large field or a home. While multi-hop technologies had been studied and commercialized, they do not fully take advantage of the BLE's low-power feature; and their configurations cannot easily be changed once the network is deployed. We have designed an architecture called BlueSkyNet. It allows network administrators to form and manage a BLE multi-hop network that allows nodes to be mobile. It takes advantage of the improving performance of BLE, and allows the network configurations to be modified in a software-defined way.

CCS CONCEPTS

• **Networks** → **Network architectures**; **Programmable networks**;

KEYWORDS

Bluetooth Low Energy, Multi-Hop Network

ACM Reference format:
Makito Kano. 2017. BlueSkyNet: BLE Multi-Hop Network Management Architecture. In *Proceedings of MobiArch '17, Los Angeles, CA, USA, August 25, 2017,* 6 pages.
https://doi.org/10.1145/3097620.3097621

1 INTRODUCTION

Internet of Things (IoT) is the trend that connects various types of devices such as home electronics, sensors, actuators, and vehicles to the Internet. Among different wireless protocols, Bluetooth Low Energy (BLE) is considered to be one of the strongest candidates in many IoT network settings. The annual Bluetooth device shipments is expected to reach 5 billion by 2021, where BLE devices account for 27%[12].

BLE is adopted to the Bluetooth protocol from version 4.0. While the classic Bluetooth is typically used with mobile devices, BLE is aimed for IoT applications because of its low power feature.

Bluetooth has consistently been upgraded, and the latest version 5 has four times range, twice data rate, and eight times broadcasting bandwidth, than version 4.2, while inheriting the low energy feature of BLE[13].

MobiArch '17, August 25, 2017, Los Angeles, CA, USA

ACM ISBN 978-1-4503-5059-4/17/08.
https://doi.org/10.1145/3097620.3097621

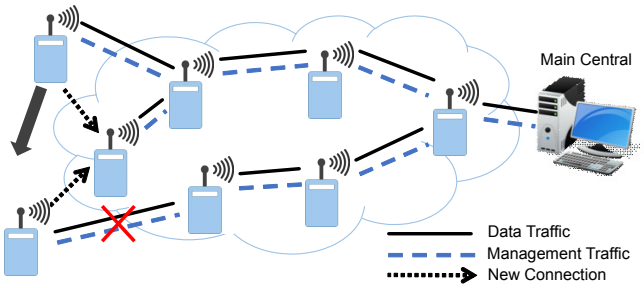

Figure 1: Example use case of BlueSkyNet.

While various performance improvements have been made, BLE is still expected to be used for a point-to-point connection. However, this is slowly changing; BLE multi-hop network architectures and routing protocols have been studied in the past few years[3, 6, 8]. In addition, a few BLE multi-hop network products have been commercialized[4, 11]. However, while the studies and the products propose effective solutions, they do not take advantage of the BLE's low-power feature and/or their network configurations cannot easily be changed once they are deployed, which limits their use cases.

To support wider range of IoT device use cases with BLE, we have designed, implemented, and evaluated an architecture called *BlueSkyNet* (BSN). Initially, it allows the BLE devices to report Received Signal Strength Indication (RSSI) of surrounding devices to the main central device to form a multi-hop network of connected devices. Then it allows network administrators to programmatically manage the devices in the network from the main central device.

We introduce two use cases of BlueSkyNet. The first is a multi-hop sensor network in an outdoor field shown in Figure 1. The network administrator uses the Main Central device to receive the sensor data and send the management traffic to the network. BlueSkyNet allows the Main Central to adjust the topology if a device malfunctions or gets disconnected as shown with the bottom left device. BlueSkyNet also allows the devices to be connected when they have to be moved around as shown with the top left device.

The second use case is the smart home IoT network. BlueSkyNet allows the users to control the devices from the main central device, such as their smartphone or PC, for purposes such as turning off all the lights in certain room. In addition, BlueSkyNet adds a mobility to the devices. The user can move the devices around the home while maintaining the connection without manually reconnecting them.

Our contributions are four-fold. First, we have designed an algorithm for all BlueSkyNet-enabled devices in the network to notify

the RSSI of devices around to the main central device. The main central can collect this information to form a topology.

Second, we have designed an API that allows the network administrators to manage the multi-hop network in a software-defined way. For example, add or delete devices and reroute traffic based on the condition such as battery level or disruption in the network.

Third, we have created a feature that allows devices to be mobile as long as they are within the range of any of the connected device.

Fourth, the architecture will benefit from the improved performance of Bluetooth 5. The increased range will allow devices to spread apart longer distance; and the increased data rate will support more bandwidth hungry applications while mitigating potential bottlenecks in a multi-hop network.

The rest of this paper is organized as follows. Section 2 describes the BLE components related to our work. Section 3 introduces studies and products about BLE multi-hop network. Section 4 describes the BlueSkyNet architecture and Section 5 describes its implementation. In Section 6, we evaluate our multi-hop connection and we conclude the paper in Section 7.

2 BACKGROUND

We describe the components of BLE in Bluetooth 4.2 that are relevant to our work.

2.1 BLE Advertisement

Generic Access Profile (GAP) is the framework that allows BLE nodes to scan, advertise, and connect. Advertising packets are sent in three channels that do not overlap with the Wi-Fi channels to minimize the device interferences. Advertising interval can be set between 10 ms to 10.24 sec. The shorter the interval, the more battery consumption to incur, but higher chance of the packets to be scanned. Similarly, scan interval and scan window define how often a scanner scans, and how long each scanning lasts, respectively.

The length of the advertising packet payload is 31 bytes. However, it can be doubled if the advertiser uses scan response, in which case the scanner needs to send a scan request. An advertising packet typically contains service and characteristic information, which define how data are organized between the master-slave pair. Each packet includes some basic header information such as device address.

Another responsibility of GAP is to establish a connection. The scanner, or central, sends a connection request to the advertiser, or peripheral. The request contains frequency hop increment for the peers to switch the frequency during the connection to avoid collision. The connection is established if the peripheral sends a connection response.

2.2 GATT Connection

Generic Attribute Profile (GATT) defines a way for data to be organized with services and characteristics. One connection contains one or more services. One service contains one or more characteristics, and each characteristic defines the type of data to be transmitted and how they can be accessed through the characteristic properties. The properties define several access types such as read, write, and notify. Read and write allow the central to read from or write to the characteristic once per operation. Notify allows the central to listen to the characteristic while the peripheral pushes the data to the characteristic.

Communications over GATT connection allows devices to use less energy compared to communications over advertisement. Peers with a GATT connection follow connection interval, which can be set between 7.5 ms and 4 sec. The interval allows the devices to turn off the radio interface when they are not transmitting data. Thus, the interval can be configured based on the type of application and the battery capacity of the device.

3 RELATED WORK

BLE multi-hop network is an ongoing research topic. ALBER[6] is the architecture that implements RPL (IPv6 routing protocol for low power and lossy network) on top of BLE. It takes advantage of the low-power and the connection oriented nature of BLE when forming a multi-hop network with RPL. There has also been a study about the BLE multi-hop routing protocol for MANET (mobile ad-hoc network)[3]. It takes advantage of the BLE's feature that allows nodes to be both master and slave. The study focused on the protocol that allows any two nodes in the network to communicate. While these routing protocols can be valuable in specific settings, our work provides a framework to implement any routing protocol; realizing solutions with BLE that can be applied to wider range of networking problems.

Advertise-based multi-hop network has been studied[8] and has also been commercialized. Qualcomm is the manufacturer of CSRmesh technology[11], which allows IoT devices to form a mesh network. CSRmesh uses the BLE advertisement to exchange data among the devices. It uses the grouping technique to send data to one or a group of devices.

Ilumi develops technology called MeshTek[4]. It uses both advertising and GATT connections to exchange data in a mesh network. However, both MeshTek and CSRmesh use the proprietary architecture and protocols that make it difficult to analyze.

While using advertisement packets simplifies the protocol, it incurs some drawbacks. First, a communication relying on advertising and scanning uses more energy because the central periodically needs to scan for advertisement packets without knowing whether the packets are actually advertised. Since there is no predefined path, all participating devices have to do this, which leads to large number of unnecessary advertisement and scanning. Second, the advertisement data are more limited (31 bytes without scan response in version 4.2) than the GATT connection data. Third, securing broadcasting data are more difficult than securing a dedicated connection. BLE advertising is known to allow an eavesdropper to identify and track devices using the BLE addresses in the advertisement packet[1].

4 BLUESKYNET ARCHITECTURE

BlueSkyNet architecture realizes the initial topology setup and the network management of the created topology with the management API.

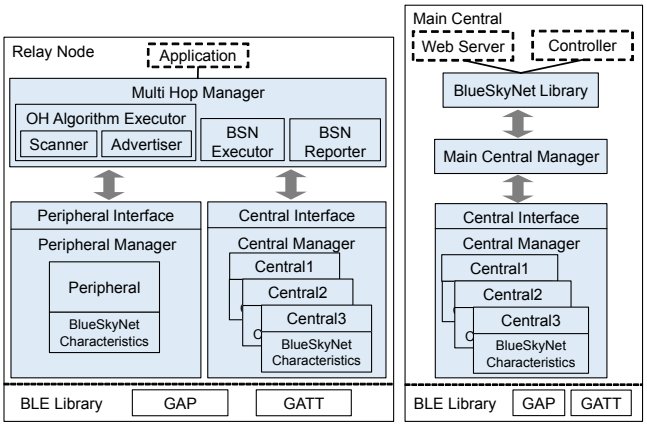

Figure 2: BlueSkyNet architecture.

4.1 Architecture Overview

Figure 2 shows the BlueSkyNet architecture. In the Relay Node, Multi Hop Manager works at the highest level of network management abstraction. During the initial topology setup phase, OverHeard (OH) Algorithm Executor uses Scanner and Advertiser to scan, advertise, and relay the packets that can be used to form a topology by the Main Central.

The first task of BSN Executor is to send application data from the device's application, such as sensor, and the performance data of the device, which are battery level and connected peripheral's RSSI, to the central. Before sending the data, it adds the "header" to identify the source. It uses the Central Interface to send these data. The second task is to act on the management commands. When the management command is received from the Peripheral Interface, it handles tasks such as start a connection based on the topology data structure, connect/disconnect/scan based on the command, and reconnect if the auto reconnection is configured. These tasks are executed through the Central Interface. In addition, it relays management commands if they are destined to the downstream device. In that case, it parses the topology structure received with the management command and determines which central to relay it.

BSN Reporter is responsible for relaying the application data and the performance data to the central device (i.e., master device) through the Peripheral Interface. It is also responsible for sending the "results" of the network management tasks such as scanned device information and connected/disconnected device information to the central. It sends the disconnected device information if the disconnection is forced (i.e., not from the management command) as well. These data are sent through the Central Interface.

Peripheral Manager is responsible for providing the Peripheral Interface. When the data are received from the characteristics, it notifies the BSN Executor. On the other hand, it provides upper layer executors a series of methods to push data into the characteristics.

Central Manager's role is similar to that of the Peripheral Manager. However, because there can be multiple centrals, it manages the centrals and their state (connected/will try to connect).

In the Main Central node, Main Central Manager works the similar way as BSN Reporter, except it works together with the BlueSkyNet Library. Instead of relaying the received data to the Peripheral Interface, it sends them to the library. The library exposes API to the network management controller. The API calls are converted into the lower level method call(s) to the Main Central Manager.

There are five characteristics in BlueSkyNet architecture and they are used to organize the data transferred for both application and network management purpose.

Application characteristic(s): This is the characteristic for application data. If the application uses multiple characteristics, an application ID associated to each characteristic would be used to identify the type of data by the main central. If the devices use different number of characteristics, the largest number will be used and configured during the initial connection process.

Topology characteristic: This is the characteristic for propagating topology data during the connection process.

Performance characteristic: This is the characteristic for sending battery level of its own and RSSI of connected devices to the main central.

Connection characteristic: This characteristic is used for connect, disconnect, and auto-reconnect actions.

Scan characteristic: This characteristic is used for the scan actions by a central.

4.2 Initial Topology Setup Phase

The goal of the initial topology setup phase is to allow the main central to know the RSSI of surrounding devices for each device expected to form a network. This information helps the main central to form more robust network than from other simple approaches such as measuring the distances among the devices. In addition, the multi-hop nature of BlueSkyNet requires links to be as robust as possible. To do so, each device periodically measures RSSI of surrounding devices and advertises that information while advertising, or relaying, the data from surrounding devices.

This process continues for some predefined period of time (e.g., one minute). After the main central collected the RSSI data, it constructs a tree data structure representing the topology and pushes it to the first-hop device(s).

4.2.1 OverHeard Algorithm Overview. To relay the measured data with BLE advertisement, we have created an algorithm, called *OverHeard (OH) algorithm*. OH algorithm uses the flooding algorithm, where every node advertises their own packet while relaying received packets, as the base. Our modification is that each node rotates between advertising its own packet and relaying received packets with a fixed interval. This rotation is necessary because BLE device can only advertise one advertising data at a time. Since the scanner continuously scans, the received packets may accumulate. However, only the most recent packet is advertised and all old scanned packets are dropped during the interval.

4.2.2 OverHeard Algorithm Details. We have designed an advertising packet structure, which we call OH packet for convenience. Figure 3 shows the structure of the OH packet. We use manufacturer specific data format to send our OH Packet Payload. The payload

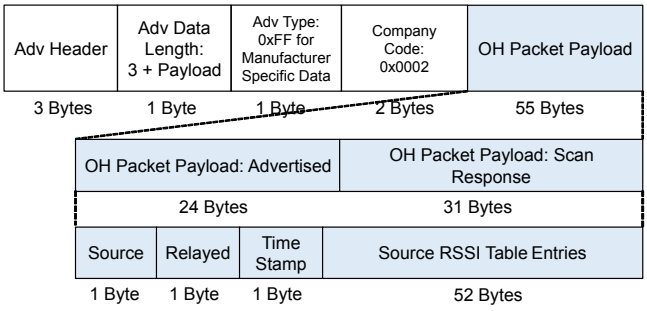

Figure 3: OH packet structure.

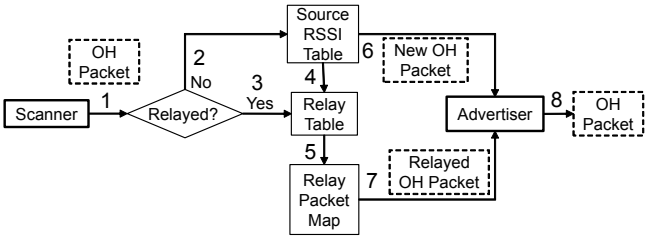

Figure 4: OverHeard algorithm steps.

has three fields: Source(1 byte), Relayed(1 byte), Time Stamp(1 byte), and Source RSSI Table Entries. Source indicates the address of a device that generates this packet. Since the 6-byte BLE address occupies a significant space in the 62-byte advertising packet, we assign one-byte alias to each device in the network and use them during the advertisement. Relayed indicates whether the packet was relayed from another device (1) or the packet was generated and sent first time out to the air (0). Time Stamp indicates the time that this packet is generated. Source RSSI Table stores the Source Address - RSSI pairs of the neighbouring devices.

Each device executes two processes to run the algorithm; OH Packet Scanner, which uses the Central Manager, and OH Packet Advertiser, which uses the Peripheral Manager. We use Figure 4 to describe the steps of the OH algorithm. Bold rectangle indicates a process, normal rectangle indicates a data structure, and dotted rectangle indicates a packet.

At a high level, the role of the scanner is to scan OH packets, store the Address - RSSI pair into the Source RSSI Table, and create a relay packet to be advertised by the advertiser. To describe more in detail, first, it determines if the scanned OH packet is a relayed packet from the Relayed field(1 in Figure 4). If not, it stores the Address - RSSI pair into the Source RSSI Table(2). Then it constructs a relay packet, which has the same payload as received OH packet, except the Relayed field is set to 1. The relay packet is passed into the RelayTable to check if it has never been relayed by this node to prevent the relay broadcast storm(3 and 4). If this is the first time to relay it, we store the relay packet into Relay Packet Map(5). It stores only the latest packet from each device because we are most interested in the latest Source RSSI Table entries. Thus, it is necessary to check the time stamp before storing it in Relay Packet Map.

The role of the advertiser is to periodically rotate between advertising the device's own Source RSSI Table entries(6), and Relayed OH Packets in Relay Packet Map(7) before advertising it(8).

4.2.3 OverHeard Algorithm Analysis. The usable payload size for an OH packet (using scan response) is 55 bytes where the first 3 bytes are used as header. Since each Source RSSI Table entry is 2 bytes, this algorithm can support at most 26 surrounding devices, although this value will be eight times larger with Bluetooth 5.

The speed of the Source RSSI Table data of a particular device to be relayed to the main central depends on the number of hops and the advertising rotation interval. For example, if the shortest path from a device to the main central is 5 hops and the rotation interval is 4 seconds, then it will take about 20 seconds. We believe that this delay is not significant in terms of usability because it is a one-time process.

Flooding algorithm guarantees that every node will receive the broadcasted packets, but is known to cause a broadcast storm problem[7], which has two parts. The first part is that since every packet can be sent by every node, the required bandwidth can increase exponentially. This is not the case with OH algorithm because only the most up-to-date packet from each node is sent.

The second part is that because each node receives and sends packets at the same time, collisions will occur. This is not the major concern with OH algorithm because packets are sent in a relatively long interval (a few seconds). In addition, BLE advertising packets are sent in three channels to mitigate the issue.

Although the initial advertising packets are sent in three channels, scan response packets are sent in only a single channel. This is known to cause packet drops with approximately 25% probability if large number of scanners are present within the range[5]. This is not the major concern with OH algorithm because it uses the flooding method.

4.3 BlueSkyNet Managment API

We show the ten most important methods that allow the administrators to form a multi-hop network as well as manage it in a software-defined way. We also introduce some use cases of the methods.

4.3.1 Management API Methods. Following is the list of API methods.

ohPacketListener: This method is used to collect the OH packets from all devices in the network. The collected data can be used to help construct a topology. For each predefined interval, it returns the OH packet payload that the main central scans.

formTopology(*topology*): This method is used to send the topology information into the network. The devices are connected as the topology is passed on to the next device. *topology* is the Topology tree data structure.

subscribeToApplication(*deviceID*, *appID*): This method is used to subscribe to application data such as sensor readings. *appID* indicates a particular application if there are multiple applications in one device. *deviceID* is the address alias of the device. Return value is the application data.

subscribeToPerformance(*deviceID*): All BlueSkyNet enabled devices send its battery level and RSSI of all connected devices. These

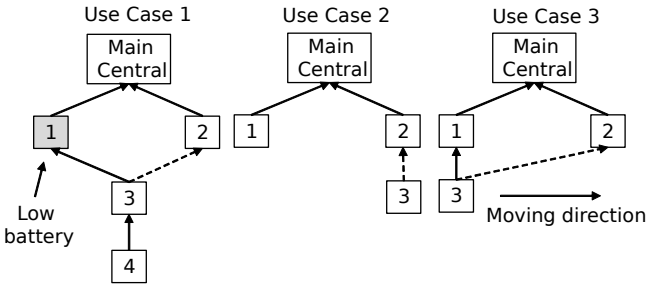

Figure 5: Use cases of the API methods.

data can be used to help take action. *deviceID* indicates the device to subscribe to these performance data.

writeToApplication(*deviceID*, *appID*, *appData*): This method is used to write the application data, *appData*, to the application characteristic indicated by *appID* of device indicated by *deviceID*, if it allows to.

connect(*deviceIDFrom*, *deviceIDTo*): This method is used to add a device to the network. *deviceIDTo* will be connected to *deviceIDFrom*.

disconnect(*deviceID*): This method is used to disconnect a device from the network. The disconnected device will automatically start to advertise while the connected devices to the disconnected device will be unaffected (i.e., stay connected).

disconnectListener(*deviceID*): This method is used to listen for a disconnection event of the device indicated by *deviceID*.

turnOnScan(*deviceID*, *scanPeriod*): This method is used to tell a device indicated by *deviceID* to scan for *scanPeriod* number of seconds. It returns the scanned device ID and its RSSI, if any.

autoReconnect(*deviceIDFrom*, *deviceIDTo*, *scanPeriod*): If a device indicated by *deviceIDTo* is disconnected from its central, the device indicated by *deviceIDFrom* scans and tries to reconnect for *scanPeriod* number of seconds.

4.3.2 Method Use Cases. Figure 5 shows three use cases of the API methods. Use case 1 shows when the main central notices the decline in the battery level for node 1 through *subscribeToPerformance*. It disconnects its peripheral with *disconnect* and tells node 2 to connect to it with *connect*. Node 4 is unaffected by this reconnection.

Use case 2 shows when node 3 is added to the network. Suppose node 1 and 2 are the reasonable candidates. Main central tells node 1 and 2 to scan and return the device ID and its RSSI with *turnOnScan*. After it decides to connect with node 2, it tells it to connect to 3 with *connect*.

Use case 3 shows when node 3 moves to right while connected to node 1. If *autoReconnect* is configured to node 2, node 2 starts to scan for node 3 and tries to connect to it when it is disconnected.

5 IMPLEMENTATION

We have implemented the BlueSkyNet architecture with Raspberry Pi 3 computers[2]. We have used bleno[9] and noble[10] BLE libraries and implemented the BlueSkyNet components with Node.js on top of them. The total number of lines of code excluding the libraries is approximately 1700, which is light enough for most of the IoT devices.

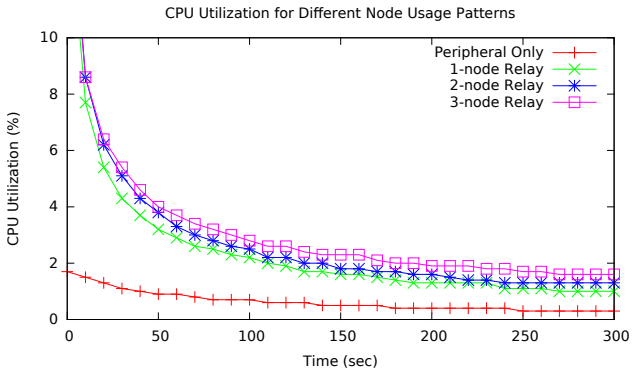

Figure 6: CPU utilization of BlueSkyNet nodes.

We have implemented the network management controller with a web server that uses the BlueSkyNet Library and the corresponding client web page with Socket.IO. Socket.IO enables real time and bi-directional communication between the client and the server. It allows the main central to receive the stream of real time data from the BLE devices and send API calls simultaneously.

6 EVALUATION

We have measured CPU utilization of BlueSkyNet-related processes in a device when it relays generated sample data every 1 second, the most common task of a device.

Figure 6 shows the evaluation results. Peripheral Only indicates that the device acts as a peripheral and sends data to its central as in the standard BLE usage. This is the base case. 1-node Relay indicates that a device is connected to one peripheral and one central device and relays the data from its peripheral to the central. 2-node and 3-node Relay are the same as 1-node Relay except the number of peripherals is different.

The graph shows that the increase from adding one node is about 0.1% in a long run. The difference between the base case and 1-node Relay is about 1%. Thus, the CPU utilization of BlueSkyNet-related processes is negligible.

7 CONCLUSION

We have shown BlueSkyNet, BLE-based multi-hop network management architecture, to take advantage of BLE's low-energy feature as well as to provide network administrators ability to manage the network in a software-defined way. We have described the algorithm to form a network of BLE devices and BlueSkyNet's management API that allows the administrators to programatically manage the network. The architecture adds mobility to the devices in the network.

REFERENCES

[1] Aveek K. Das, Parth H. Pathak, Chen-Nee Chuah, and Prasant Mohapatra. 2016. Uncovering Privacy Leakage in BLE Network Traffic of Wearable Fitness Trackers. In *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications (HotMobile '16)*. ACM, New York, NY, USA, 99–104. DOI: https://doi.org/10.1145/2873587.2873594

[2] Raspberry Pi Foundation. 2017. Raspberry Pi 3 Model B. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. (2017).

[3] Z. Guo, I. G. Harris, L. f. Tsaur, and X. Chen. 2015. An on-demand scatternet formation and multi-hop routing protocol for BLE-based wireless sensor networks. In *2015 IEEE Wireless Communications and Networking Conference (WCNC)*. 1590–1595. DOI: https://doi.org/10.1109/WCNC.2015.7127705

[4] Ilumi. 2017. The only Bluetooth Smart solution with connection & broadcast based mesh. https://ilumisolutions.com/. (2017).

[5] Robin Kravets, Albert F Harris, III, and Roy Want. 2016. Beacon Trains: Blazing a Trail Through Dense BLE Environments. In *Proceedings of the Eleventh ACM Workshop on Challenged Networks (CHANTS '16)*. ACM, New York, NY, USA, 69–74. DOI: https://doi.org/10.1145/2979683.2979687

[6] T. Lee, M. S. Lee, H. S. Kim, and S. Bahk. 2016. A Synergistic Architecture for RPL over BLE. In *2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. 1–9. DOI: https://doi.org/10.1109/SAHCN.2016.7732968

[7] F. Li and Y. Wang. 2007. Routing in vehicular ad hoc networks: A survey. *IEEE Vehicular Technology Magazine* 2, 2 (June 2007), 12–22. DOI: https://doi.org/10.1109/MVT.2007.912927

[8] K. Mikhaylov and J. Tervonen. 2013. Multihop data transfer service for Bluetooth Low Energy. In *2013 13th International Conference on ITS Telecommunications (ITST)*. 319–324. DOI: https://doi.org/10.1109/ITST.2013.6685566

[9] Sandeep Mistry. 2017. bleno. https://github.com/sandeepmistry/bleno. (2017).

[10] Sandeep Mistry. 2017. noble. https://github.com/sandeepmistry/noble. (2017).

[11] Qualcomm. 2017. CSRmesh technology for Bluetooth-based networks. https://www.qualcomm.com/products/features/csrmesh. (2017).

[12] ABI Research. 2016. Bluetooth Low Energy Devices to Account for 27% of Total Bluetooth Shipments by 2021 as New Enhancements Expand Opportunities in IoT. https://www.abiresearch.com/press/bluetooth-low-energy-devices-account-27-total-blue/. (2016).

[13] Bluetooth SIG. 2017. Bluetooth 5: What it's all about. https://www.bluetooth.com/specifications/bluetooth-core-specification/bluetooth5. (2017).