Parameter Optimization for BLE Mesh Sensor Networks Using an MQTT Gateway

Philipp Bolte
Dept. of Electronics and Circuit Technology
South Westphalia University of Applied Sciences
Soest, Germany
bolte.philipp@fh-swf.de

Ulf Witkowski
Dept. of Electronics and Circuit Technology
South Westphalia University of Applied Sciences
Soest, Germany
witkowski.ulf@fh-swf.de

***Abstract*— The BLE Mesh standard is well suited for use in IoT applications because of its low energy consumption. The Mesh extension enables communication beyond directly connected devices. The provisioning of the network can be performed using a smartphone utilizing a compatibility layer for traditional BLE. Connectivity to the Internet is required for many Internet of Things applications, which is not supported by BLE Mesh by default. We propose a bidirectional BLE Mesh to MQTT Gateway (GW) architecture to overcome this limitation. We have evaluated the effect of several BLE Mesh parameters on packet loss using the proposed GW architecture.**

Keywords—Mesh Networks, Internet of Things, Sensor Node, BLE Mesh, MQTT

I. INTRODUCTION

The smart home industry is growing at a rapid pace. In Europe, the number of smart homes increased from 83.9 million to 111.9 million from end of 2018 to end of 2019 while an annual increase of 20.2% is excepted until 2024 [1]. Similar growth is also predicted for the smart building sector. For Europe, the market share of smart building sector is estimated to grow at an annual rate of 17.6% from 2021 to 2028 [2]. Smart home and smart building are a segment of the Internet of Things (IoT) with special characteristics and requirements. Protocols traditionally used for smart building, e.g., BACnet, DALI or KNX, are using wired connections [3]. Those wired networks are inflexible and their expansion is often not easy to realize. Wireless protocols targeted for smart home applications, e.g., ZigBee, Z-Wave or Wi-Fi, are widely used [4]. Those systems can be retrofitted comparatively easily to existing buildings. However, these IoT protocols, except for Wi-Fi, require a sophisticated configuration. Furthermore, most of the wireless protocols are not designed for battery operated devices in terms of power consumption.

We propose a Bluetooth Low Energy (BLE) Mesh based sensor network for the use with smart building and smart home that solves the previous mentioned issues. This mesh extension of the BLE standard implements a flooding algorithm to forward messages across the network, so participants who are not in direct reach can communicate with each other. Low-power nodes with a very low current consumption enable the integration into battery powered devices. One key advantage of using BLE Mesh is the ability to configure the network using a smartphone or laptop without the need of additional accessory. The BLE Mesh standard further defines the application layer, providing interoperability between devices of different manufactures. We have designed a Gateway (GW) that forwards messages between the BLE Mesh network and a Message Queuing Telemetry Transport (MQTT) broker, e.g., for integration into cloud-based services or for bridging multiple spatial separated networks.

The second Section of the paper gives an introduction of the BLE Mesh standard with focus on IoT relevant features. The third Section introduces the proposed BLE Mesh to MQTT GW. In Section 4, the used set-up for the experiments is described. The results of the experiments are evaluated in Section 5. Section 6 concludes the paper.

II. BLE MESH

The BLE Mesh standard is based on the Bluetooth low-energy part of the Bluetooth 4.0 specification and shares the lower protocol layers [5]. Bluetooth devices usually implement a point-to-point connection between one central and multiple peripherals. In contrast, in a BLE Mesh network each device is able to communicate with all reachable devices of the network. The protocol stack of BLE Mesh is shown in Figure 1.

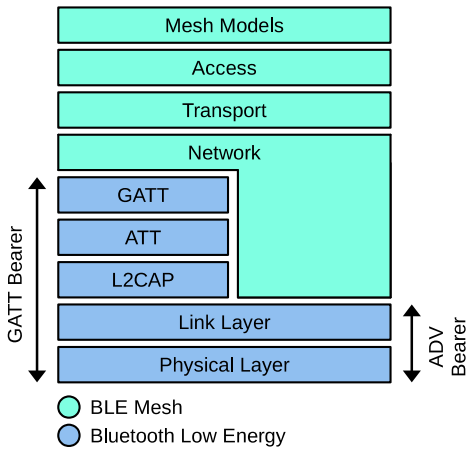

Figure 1. BLE Mesh protocol stack [5]

The lower two levels of BLE Mesh are shared with traditional BLE. The bearer level implements the transport of Protocol Data Units (PDU) using either advertisement packets of BLE at fixed time intervals or Generic Attribute (GATT) profiles for communication with legacy BLE devices. Packets are encrypted on network layer with a network key. The decrypted packets are retransmitted by the relay nodes while the Time-To-Live (TTL) field is decremented until it reaches zero.

 ISBN: 978-1-61208-987-4

This flooding approach does not require knowledge of the network structure. However, this approach is resulting in poor performance issues that can lead to an increased packet loss for large networks with many participants [7]. The simplicity, on the other hand, is especially beneficial for resource constraint devices typically used in IoT. The transport layer implements message segmentation, buffering of messages for coupled Low-Power Nodes (LPNs) that are currently not reachable, and encryption of the data from upper layers with application specific keys. The application specific encryption enables multiple isolated applications within a single network. Messages are retransmitted by the relay nodes even if the corresponding application key is not known. A BLE Mesh device can integrate multiple elements that are identified by a unique address. An element is an addressable unit, e.g., a switch panel with multiple buttons, in a BLE Mesh network. Each element includes one or multiple models. The models exchange data by a publish and subscribe pattern. BLE defines various mesh models with standardized attributes. Data producing devices usually implement some sort of server model. Data consuming devices, on the other hand, typically implement a client model. There are many predefined models, ranging from generic on/off or level models to very specific models, e.g., for lighting applications [8]. For the proposed application, we are using a sensor server for the temperature sensor nodes and sensor clients for the display nodes. The GW implements both, a sensor client to fetch sensor data and a sensor server to transmit requests from MQTT to a BLE Mesh device.

A. *Message Transport*

The maximum size of access PDUs is 380 bytes, while packets above 11 bytes are segmented by the network layer [9]. The performance of segmented messages is poor compared to unsegmented messages [10]. The transport PDUs of segmented messages are not transmitted instantly but rather in successive BLE advertise packets. Therefore, it is reasonable to send the data in several smaller access layer messages, as the overhead is negligible. In this case, a lost transport PDU affects only a single measurement and not a complete data series.

Figure 2 shows the transfer of BLE Mesh transport PDUs using the Advertising (ADV) bearer. The advertiser, i.e., sender, transmits the transport PDU in three identical *ADV_NONCONN_IND* packets on BLE channels 37, 38 and 39, that are primary used for BLE advertisement. The sequence of packets is determined by the fixed *advInterval*, that is between 20 ms and 10.24 s, and a random delay *advDelay* between 0 and 10 ms [11]. The scanner, i.e., receiver, listens successively on the advertisement channels for the time *scanWindow* with an interval defined by *scanInterval*.

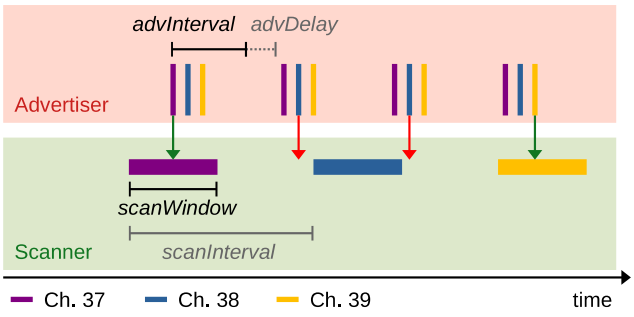

Figure 2. BLE Mesh data transfer using ADV bearer

Since the receiver does not listen permanently, the data must be sent multiple times to ensure reception. BLE Mesh implements three types of retransmissions [12]. The retransmission of network PDUs sent by the note is controlled by the *Network Transmission Count* (NTC), which allows 1 to 8 transmissions, and *Network Transmit Interval* (NTI), in a range between 10 to 320 ms. The retransmission of received network PDUs from other nodes of the network is controlled by the parameters *Relay Retransmit Count* (RRC) and *Relay Retransmit Interval* (RRI), having the same range as for network scope. On model layer, the retransmission of published messages is controlled by the *Publish Retransmit Count* (PRC) and *Publish Retransmit Interval* (PRI), ranging from 50 ms to 1.6 s. A message can be sent up to 64 times if the network and publish retransmit count parameters are set to maximum.

B. *Device Addressing*

Messages, respectively transport PDUs, have a source and a destination address, which are both 16 bit wide. Each node of a network is assigned a unique unicast address during provisioning in the address range from 0x0001 to 0x7FFF, which is half of the available address space. Furthermore, various multicast address spaces are available. The address space between 0x8000 and 0xBFFF is reserved for virtual addresses. Those addresses represent a 128 bit Universally Unique Identifier (UUID) that can be used by manufactures to uniquely address elements of their products. However, this function has been barely used so far. Addresses starting from 0xC000 are used as group addresses. This address range is further subdivided to assignable addresses up to 0xFEFF and special addresses, e.g., 0xFFFE for all relays and 0xFFFF for all nodes [14]. E.g., the assignable group address could be used for assigning all devices of a certain room.

Incoming messages with the unicast address of the receiving node are always processed by the device. The models of a node can additionally subscribe to virtual addresses and group addresses. For outgoing messages, either a unicast or a multicast address can be used as destination.

C. *Low-Power Features*

The BLE standard by itself is already optimized for low energy consumption. The BLE Mesh specification introduces additional low-power features. E.g., a node can operate as LPN to further increase battery runtime. This type of node couples with a friend node and suspends the scanning for advertisement packets. The LPN periodically polls the permanent listening friend node for messages that were received since the last polling. The polling interval can be set between 10 s and several hours. The LPN enters the lowest possible power state between polling and is not reachable by other nodes.

D. *Network Provisioning*

A provisioner node is used to set-up the network. The provisioning process involves five stages. First, the unprovisioned devices announce their presence by sending mesh beacons. The provisioner sends an invitation which is replied by a capabilities report from the device. In the key exchange stage, both devices exchange their public keys. It is recommended to perform an authentication, e.g., by a pin code afterwards. This step can be skipped if the devices are not able to interact with the user. The actual provisioning is performed by the provisioner

 ISBN: 978-1-61208-987-4

by setting the network and application keys and a unicast address and optionally configure the models. Afterwards the device has joined the network and is able to send and receive messages.

E. BLE Connectivity

By default, the BLE Mesh nodes are communicating by embedding the network PDUs inside of *ADV_NONCONN_IND* advertisement packets. The hardware of legacy BLE devices, e.g., Smartphones is, in theory, capable of processing those packets but the software stack does not support the BLE Mesh protocol. To enable communication with those legacy devices, the BLE Mesh standards provides a GATT proxy feature. The network PDUs are then transmitted using the Mesh Proxy GATT service (0x1828) [15]. Typically, the provisioning of new nodes is performed using a smartphone utilizing the GATT proxy feature for easy management of the network. The GATT proxy feature is optionally and does not need to be implemented by the nodes.

III. MQTT GATEWAY

BLE Mesh devices are only capable of sending messages to other nodes of a network. For typical IoT applications, an integration of cloud services is at least desired, if not mandatory. The integration of cloud services, e.g., could enable a deeper analysis of sensor data for process optimization and remote control of actuators [16]. Our proposed GW enables the integration of low-power BLE Mesh nodes into online cloud services using the widely used MQTT protocol.

Figure 3 shows the components of our BLE Mesh to MQTT GW solution. The hardware of the system is composed of a Raspberry Pi miniature computer and a Nordic nRF52840 BLE development board. The Raspberry Pi itself embeds a Bluetooth radio which cannot be used as the software stack since is not Mesh compatible.

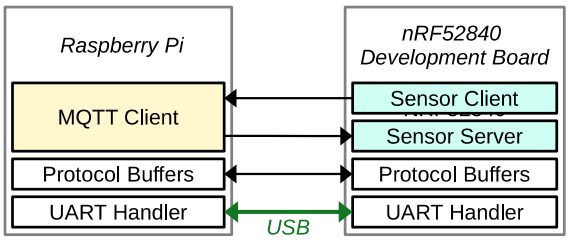

Figure 3. BLE Mesh to MQTT GW components

A. BLE Mesh Endpoint

The nRF52840 BLE development kit (DK) acts as the BLE Mesh Endpoint (EP) of our proposed GW. The nRF52840 System-on-Chip (SoC) of the DK embeds a 2.4 GHz radio and an ARM Cortex-M processing core. The developed BLE Mesh EP was derived from application examples of the Zephyr framework. We have implemented a generic sensor client model. The non-uniform data representation for different sensor types does not allow a generic conversion between BLE sensor values and standard floating point format [17]. We have implemented hooks for sensor types *present ambient humidity*, *precise present ambient temperature* and *present input voltage*. The integration of other sensor types can be achieved by implementing according hooks with data conversion to a generic float value. The sensor client is able to receive messages from multiple devices by subscribing to a group address (e.g., 0xC000). All received sensor readings are redirected to the MQTT EP after adding meta information, e.g., the sender address and Received Signal Strength Indicator (RSSI). Incoming messages from the MQTT EP are redirected to other BLE Mesh nodes by the implemented sensor server. Currently, only the above listed sensor types are supported for sending messages from MQTT to a BLE Mesh node. The integration of other sensor types requires the extension of the sensor server model. A final version of the GW would implement all defined sensor types. The destination address of the redirected sensor values is fetched from the included meta data included in the MQTT message. The configuration of the model parameters is performed by the provisioner of the network.

B. MQTT Endpoint

The MQTT protocol is based on TCP/IP stack. The used nRF52840 SoC does not support Ethernet or Wi-Fi interface. Therefore, the MQTT EP is executed on a Raspberry Pi miniature computer that integrates an Ethernet interface. Newer models also integrate a Wi-Fi module. The software of the MQTT EP is written in Python and utilizes the threading library to prevent I/O blocking. The Python program reads the required connections parameters from a configuration file. The configuration file includes the MQTT broker address, user credentials and a topic prefix to enable the use of multiple GWs on a single MQTT broker. Incoming JSON formatted messages from the MQTT broker are parsed to a Protobuf object that is then transferred to the BLE EP using the serial handler. The MQTT protocol was designed for ASCII-formatted payload, while Protoc serializes to a binary representation. Furthermore, Protoc is not supported by all cloud stacks. Therefore, the MQTT payload is JSON formatted, to enhance compatibility. Since the message objects are no longer specified by a hard-coded protocol definition, it is now the responsibility of the developer to correctly format the JSON message. Incorrectly formatted messages are discarded by the broker. All received sensor values from the BLE Mesh EP are parsed to a JSON object that is published by the MQTT client. The topic of the message includes the source address of the BLE Mesh message. Additional meta data, e.g., RSSI, is included in the payload, as shown in Figure 4.

```
{
     "seq": 236,
     "timestamp": 29470814,
     "addr": 31,
     "recvRssi": -86,
     "recvTtl": 6,
     "ambientHumidity": {
       "ambientHumidity": 33.91
     }
}
```

Figure 4. MQTT message payload example

C. Internal Message Coding

The messages transfer between the BLE Mesh EP and the MQTT EP is realized using Google Protocol Buffers (Protobuf) format. With Protobuf, structured data objects, that are specified in a platform independent ".proto"-file, can be translated to different programming languages using the Protocol Buffers compiler (protoc) [18]. The same protocol definition can be used in both, the C program on the BLE SoC and the Python program on the Raspberry Pi. In the C program, the data objects are represented by structures. In Python each defined data object is represented by a class that can be instantiated. The Protobuf library serializes those structured data objects to a binary stream that can be deserialized by the receiver. The Protoc data format is substantially more efficient compared to the common used JSON protocol and usually only produces negligible overhead [19]. The capacity of the UART connection between the Raspberry Pi and the nRF52840 board is limited to 115200 bauds, therefore an efficient data transfer minimizes packet delay and ensures good utilization of the limited UART link.

IV. Reference Set-up

A reference BLE Mesh network was set-up to evaluate the use of BLE Mesh for indoor IoT applications. We have deployed a typical smart home environment with multiple low-power and relay nodes, as shown in Figure 5. The proposed GW architecture was used to track messages inside the mesh network and to collect meta data.

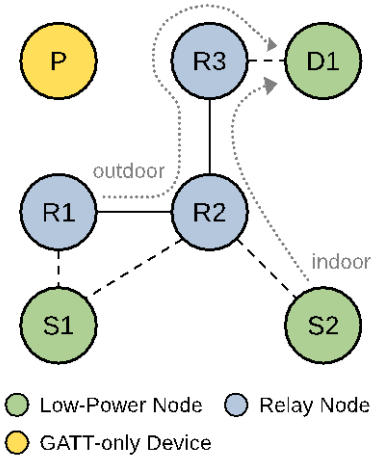

Figure 5. BLE Mesh reference set-up

The mesh network is made up of two LPN acting as temperature and humidity sensors. One LPN is equipped with an e-paper display that shows sensor readings that were transferred over the BLE Mesh network. The LPNs have no direct connection, instead three relay nodes are used to bridge all devices. Each relay acts as an MQTT GW. The LPNs establish a friendship with one of the relay nodes in reach to further reduce energy consumption. The display LPN implements a sensor client that subscribes to a group address (0xC000). The display is divided into two slots. One dataset is gathered from the mesh-local sensor *S1* (indoor) and the dataset for the second slot is injected from MQTT by router *R1* (outdoor). In this example, the second dataset is fetched from OpenWeather Application Programming Interface (API). The LPN sensors gather temperature and humidity readings each 15 minutes. The outdoor temperature and humidity is fetched at an interval of 30 minutes from the web API. The transmission of BLE Mesh messages inside the network can be tracked as each relay redirects incoming messages to the MQTT broker.

V. Experiments

We have evaluated the reliability of the data transmission based on the previously described set-up. The packet loss was used as the primary metric. The received meta data from the GWs was evaluated as well. The received RSSI values provide additional information about the link reserve. The experiments were primarily conducted to study the effect of retransmission settings. A high number of repetitions increases the probability that a message will reach the recipient. On the other hand, the medium is shared by all participants of the network. The total transmitted messages increase exponentially with the number of participants and the transmit count values, increasing the probability for collisions.

In total, three experiments were performed in an office building under real conditions. The experiments were carried out consecutively for a period of 7 days using the same hardware. Table 1 shows the used parameters for the experiments.

TABLE I. BLE Mesh Parameters

	Exp. 1	Exp. 2	Exp. 3
PTC	1	1	-
PTI [ms]	250	500	-
NTC	3	6	2
NTI [ms]	50	50	50
RRC *	3	6	2
RRI * [ms]	60	60	60

* only for relay nodes

Figure 6 shows the packet loss of the three experiments for the transmissions of LPN S1. The LPN S1 transmits three messages each 15 minutes, reporting its battery voltage, ambient temperature and humidity. The messages are first received by relay node R1 and get retransmitted until they finally reach the display node D1. The messages are tracked by the GWs.

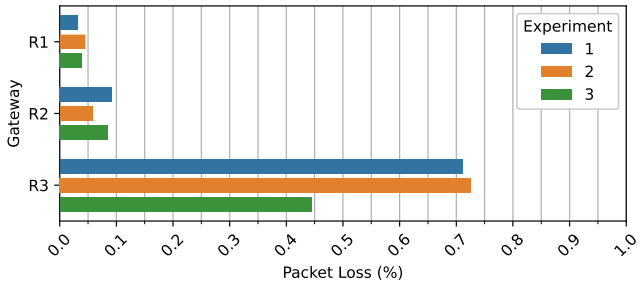

Figure 6. Packet loss for messages from LPN S1

The packet loss at the first relay node is less than 5% for all parameter sets. The loss increases to almost 10% for all experiments at node R2. Up to this point, however, no major differences are discernible between the various parameter sets. Significant differences could be observed between the transmissions from relay nodes R2 to R3. Here, the packet loss

of experiments 1 and 2 is around 70%. In the last experiment, the packet loss at R3 was 44.5% although the transmit counter was set lowest. Figure 7 shows boxplots of the RSSI parameter of the received messages sent by LPN S1.

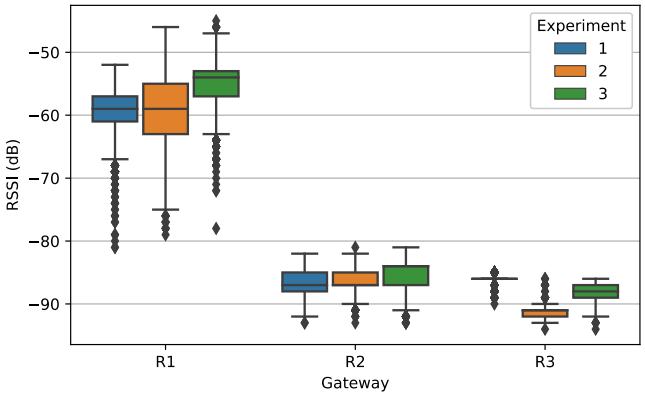

Figure 7. RSSI distribution for messages from LPN S1

The RSSI values of the received messages from S1 at relay node R1 are widely spread. The mean value ranged from -55.2 dB for Experiment 3 to -59.4 dB for Experiment 1. At stations R2 and R3 a smaller spread of the recoded RSSI values was observed. The mean values were located around -86 dB for station R2 for all experiments. For node R3 the mean values of the RSSI measurements are between -86 dB for experiment 1 and -91 dB for experiment 2. The smallest recorded RSSI value was –94 dB. The sensitivity of the used nRF52840 module is specified with -95 dB [20]. The spatial conditions used for the experiments demonstrate a reliable transmission with a link margin of 10 dB. Compared to the significantly better signal strengths at station R1, no increased packet loss could be detected. For stations that are just in reach, however, the presence of a small interferer is sufficient to interrupt the transmissions. The scatter of successive RSSI values was rather small. However, there were occasional jumps in the measured values, which can be explained by the presence of interferer. Then the signal of all messages is suppressed to the extent that they can no longer be decoded by the receiver. In such cases, the increase of transmission counters does not improve reachability. If sufficient link margin is available, the already low packet loss will not be significantly improved by the increase of transmissions.

VI. Conclusions and Future Work

The proposed GW architecture provides connectivity between a BLE Mesh network and the Internet. The provisioning of the network can be done via smartphone. Only the MQTT parameters need to be set in a configuration file on the GW. Even existing networks can be easily connected to the Internet using the proposed GW architecture. The GW includes meta data of the received BLE Mesh messages before redirection to the MQTT broker. The proposed GW architecture can be used to perform experiments on BLE Mesh networks without the need for expensive measurement equipment.

We have evaluated the effect of transmission counters and delays on packet loss when transmitting data over multiple hops. A BLE Mesh node does not listen continuously on the RF interface but instead implements a complex scheduling of scan intervals on multiple channels to reduce current consumption. A re-transmission of messages, in theory, should improve reliability. However, too many repetitions lead to a high utilization of the radio channel. Our experiments have shown that the receiver of the used nRF52840 module works very reliably with little packet loss, even at the lowest possible packet repetition rates, if sufficient link margin is given. In such cases, an increase repetition of messages does not improve the packet loss. The experiments were performed under real conditions. We could observe a significant drop of RSSI values for longer time periods on all relay nodes, caused by the presence of disturbers, e.g., Wi-Fi and other BT devices located in the lab. When a station is positioned close to the range limit, the decreased signal strength cannot be compensated by increasing the repetition counters, resulting in high packet loss during the presence of the disturber. Our recommendation is to set the corresponding parameters as small as possible, in order to reduce the RF channel utilization as low as possible.

The proposed architecture could be extended to detect packet loss between multiple GWs and to retransmit lost messages using the MQTT link. However, the flooding of the mesh network with always the same message must be prevented.

References

[1] M. Bäckman, “Smart Homes and Home Automation - IoT Research Series,” Berg Insight, Gothenburg, Sweden, 2021.

[2] “Europe Smart Building Market Size, Share & COVID-19 Impact Analysis, By Solution, By Application, and Europe Forecast, 2021-2028,” Furtune Business Insights, Apr. 2021.

[3] K. Lohin, Y. Jain, C. Patel, and N. Doshi, “Open Communication Protocols for Building Automation Systems,” in *Procedia Computer Science*, vol. 160, Nov. 2019, pp. 723-727, doi: 10.1016/j.procs.2019.11.020.

[4] S. J. Danbatta, A. Varol, “Comparison of Zigbee, Z-Wave, Wi-Fi, and Bluetooth Wireless Technologies Used in Home Automation,” in *2019 7th International Symposium on Digital Forensics and Securits (ISDF)*, Jul. 2019, pp. 1-5, doi: 10.1109/ISDFS.2019.8757472.

[5] Nordic, *nRF SDK for Mesh v3.2.0*, Accessed: Nov. 19, 2021. Accessed: Mar. 02, 2022. [Online]. Available: https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.meshsdk.v3.2.0/md_doc_introduction_basic_concepts.html

[6] S. M. Darroudi, C. Gomez, and J. Crowcroft, “Bluetooth Low Energy Mesh Networks: A Standards Perspective,” in *IEEE Communications Magazine*, vol. 58, Apr. 2020, pp. 95-101, doi: 10.1109/MCOM.001.1900523.

[7] R. Rondón, A. Mahmood, S. Grimaldi, and M. Gidlund, “Understanding the Performance of Bluetooth Mesh: Reliability, Delay, and Scalability Analysis,” in *IEEE Internet of Things Journal*, vol. 7, Mar. 2020, pp. 2089-2101, doi: 10.1109/JIOT.2019.2960248.

[8] *Bluetooth Specification - Mesh Model*, rev. 1.0.1, Jan. 2021. Accessed: Mar. 02, 2022. [Online]. Available: https://www.bluetooth.com/specifications/specs/mesh-model-1-0-1.

[9] Á. Hernández-Solana, D. Pérez-Díaz-De-Cerio, M. García-Lozano, A. V. Bardají, and J. Valenzuela, "Bluetooth Mesh Analysis, Issues, and Challenges," in *IEEE Access*, *vol. 8*, pp. 53784-53800, 2020, doi: 10.1109/ACCESS.2020.2980795.

[10] A. Aijaz, A. Stanoev, D. London, and V. Marot, “Demystifying the Performance of Bluetooth Mesh: Experimental Evaluation and Optimization”, in in *IEEE Wireless Days 2021*, Preprint

[11] Rohde & Schwarz, “Bluetooth Low Energy Over-The-Air Advertiser Testing,” Application Note, Rev. 1C109, Dec. 2017.

[12] *Bluetooth Specification - Mesh Profile*, rev. 1.0.1, Jan. 2019. Accessed: Mar. 02, 2022. [Online]. Available: https://www.bluetooth.com/specifications/specs/mesh-profile- 1-0-1.

 ISBN: 978-1-61208-987-4

[13] M. Woolley, „Bluetooth Mesh Networking: An Indtoduction for Developers," Version 1.0.1, Dec 2020. Accessed: Mar. 02, 2022. [Online]. Available: https://www.bluetooth.com/wp-content/uploads/2019/03/Mesh-Technology-Overview.pdf.

[14] Kai R. and M. Woolley, „The Fundamental Concepts of Bluetooth Mesh Networking", Aug. 2017. Accessed: Mar. 02, 2022. [Online]. Available: https://www.bluetooth.com/blog/the-fundamental-concepts-of-bluetooth-mesh-networking-part-2/.

[15] *Bluetooth Specification – 16-bit UUID Numbers Document*, Jan. 2022. Accessed: Mar. 02, 2022. [Online]. Available: https://btprodspecificationrefs.blob.core.windows.net/assigned-values/16-bit%20UUID%20Numbers%20Document.pdf

[16] A. Rajith, S. Soki, and M. Hiroshi, „Real-time optimized HVAC control system on top of an IoT framework," in *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*, Barcelona, Spain, Apr. 2018, pp. 181-186, DOI: 10.1109/FMEC.2018.8364062 .

[17] Nordic, *nRF Connect SDK*, Accessd: Mar. 16, 2022. [Online]. Available: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/libraries/bluetooth_services/mesh/sensor_types.html

[18] Google, *Protocol Buffers Language Guide*, Mar. 06, 2022. [Online]. Available: https://developers.google.com/protocol-buffers/docs/proto3

[19] S. Popić, D. Pezer, B. Mrazovac, and M. Teslic, „Performance evaluation of using Protocol Buffers in the Internet of Things communication," in *2016 International Conference on Smart Systems and Technologies (SST)*, Oct. 2016, pp. 261-265.

[20] Nordic, „High-end multiprotocol Bluetooth Low Energy (LE) SoC supporting: Bluetooth 5.3 / Bluetooth mesh / Thread / Zigbee / 802.15.4 / ANT," nRF52840 Product Brief, Rev. 3.0. Accessed: Mar. 02, 2022. [Online]. Available: https://nsscprodmedia.blob.core.windows.net/prod/software-and-other-downloads/product-briefs/nrf52840-soc-v3.0.pdf

 ISBN: 978-1-61208-987-4