Original scientific paper

Informacije MIDEM
Journal of Microelectronics,
Electronic Components and Materials
Vol. 48, No. 2(2018), 85 – 95

Multi-hop communication in Bluetooth Low Energy ad-hoc wireless sensor network

Branko Skočir[1], Gregor Papa[1,2], Anton Biasizzo[1,2]

[1]Jožef Stefan International Postgraduate School, Ljubljana, Slovenia
[2]Jožef Stefan Institute, Ljubljana, Slovenia

Abstract: This paper presents a multi-hop mechanism for Bluetooth Low Energy (BLE) 4.0 ad-hoc wireless sensor network (WSN). The BLE 4.0 protocol supports only the piconet topology and does not support data transfer over multiple nodes. To overcome this limitation a mechanism to relay sensor data over multiple BLE 4.0 nodes using Master/Slave switching was developed. The mechanism dynamically creates communication paths within the BLE ad-hoc sensor network where all BLE nodes are identical. The sensor data query is initiated from a designated node and forwarded through the dynamically build network. The sensor data are collected over the same path, back to the designated node. The mechanism does not perform route discovery therefore no routing tables are needed. Using this mechanism, the range of the sensor data acquisition can be extended from BLE range to whole BLE sensor network.

Keywords: Bluetooth Low Energy (BLE); wireless sensor network; multi-hop mechanism

Multi-hop komunikacija v Bluetooth Low Energy ad-hoc brezžičnem senzorskem omrežju

Izvleček: Članek predstavlja multi-hop mehanizem za Bluetooth Low Energy (BLE) 4.0 ad-hoc brezžično senzorsko omrežje. BLE 4.0 protokol omogoča kreiranje samo piconet topologije in ne podpira prenosa podatkov preko več vozlišč. Da bi obšli to omejitev smo razvili mehanizem za posredovanje senzorskih podatkov preko več BLE 4.0 vozlišč s preklapljanjem med Master in Slave načinom delovanja. Mehanizem v BLE ad-hoc brezžičnem senzorskem omrežju, kjer so vsa vozlišča enaka, dinamično ustvarja komunikacijske poti. Poizvedba o senzorskih podatkih se sproži iz točno določenega vozlišča in se posreduje prek dinamično sestavljenega omrežja. Podatki posameznih senzorjev se posredujejo nazaj po isti poti do vozlišča, ki je poizvedbo sprožil. Glavna prednost tega mehanizma je vzpostavitev senzorskega omrežja brez usmerjevalnih tabel. Z uporabo tega mehanizma se pokritost z BLE lahko razširi na celotno senzorsko omrežje.

Ključne besede: Bluetooth Low Energy (BLE); brezžično senzorsko omrežje; multi-hop mehanizem

* *Corresponding Author's e-mail: b.skocir@gmail.com*

1 Introduction

Bluetooth Low Energy (BLE) is a communication technology developed for low power consumption and low cost applications. It was introduced in 2006 as "Wibree" and in 2010 merged into the main Bluetooth standard with the adoption of the Bluetooth Core Specification Version 4.0 [1].

Due to the low energy consumption and ease of use the BLE technology is used in various devices, ranging from health care, fitness, home automation, toys, transportation and industry. Because of its low power consumption design, it is especially suitable for coin cell battery communication devices like wireless sensors. Several wireless sensors can form a Wireless Sensor Network (WSN) [2] that can acquire environmental data and send it through the network to the main device or gateway. From here data can be sent over existing network infrastructure to the remote client thus making a WSN and associated sensors a part of Internet of Things (IoT) [3].

In contrast to other wireless technologies, that are also used for wireless sensor networks like Zigbee and

ANT+, the BLE 4.0 only supports Peer-To-Peer (P2P) and star topology with no multi-hop capabilities. Although the BLE 4.0 protocol is already offering a wide spread of possible usage the data relaying over multiple nodes within the network would highly increase the number of potential uses.

While BLE lacks the multi-hop capabilities, there have been some attempts to implement multi-hop functionality using the Bluetooth application layer. Some implementations of multi-hop data transfer with BLE 4.0 are presented by Mikhaylov and Tervonen [4] as well as by Maharjan, Witkowski and Zandian [5]. In [4] a mechanism that relays data to a node with known address over intermediate nodes was suggested. The proposed mechanism was implemented on only four nodes carrying out two to three hops. The emphasized problem in the article is the high route time discovery, which takes about 25 seconds on average and is due to the service discovery time and data exchange for route establishment. In [5] the multi-hop mechanism is based on a tree network with unique node identity number addressing. The presented implementation of this approach limits the number of nodes that can be connected to each central device to at most three. The selected range of node addresses enables the forming of tree topology network up to level 5.

In this paper a mechanism that enables multi-hop data transfer to a central node over multiple BLE 4.0 nodes is presented. A wireless sensor network is constructed using tree topology. All wireless sensor nodes are identical and can transfer acquired data and/or relay data from the connected nodes. An evaluation wireless sensor network with 7 nodes was built to validate our multi-hop mechanism. Each node is composed of Bluegiga BLE113 [6] communication module mounted on breakout board [7], temperature sensors [8] with additional peripheral hardware, and power supply. BLE node was implemented on our developed demonstration evaluation board.

The rest of the paper is organized as follows: in the following section an overview of BLE 4.0 technology is presented. The third section is focused on the related work in BLE 4.0 multi-hop data transfer. In the fourth section our multi-hop data transfer mechanism is described. In the fifth section the implementation of the mechanism is described and the results are given. The paper concludes with section six.

2 BLE 4.0 technology

Bluetooth Low Energy (BLE) 4.0 is a wireless communication technology that allows data to be transmitted over the air between two BLE devices. It uses the 2.400 GHz - 2.4835 GHz Industrial, Scientific and Medical (ISM) band. BLE uses frequency hopping spread spectrum modulation technique called Direct-Sequence Spread Spectrum (DSSS) for interference minimization and security maximization [9]. In contrast to the classic Bluetooth that uses 79 1MHz channels the BLE frequency hopping spans over 40 2MHz channels. 37 channels are used for data transfer, whereas the remaining three channels are reserved for advertising [1].

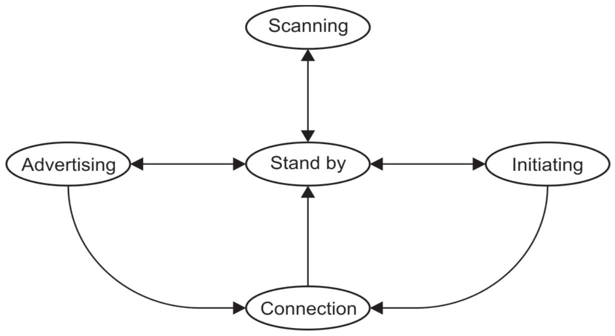

Figure 1: BLE 4.0 link layer state diagram

The connection and data transfer is achieved through the link layer state machine as shown in Figure 1. In Advertising state, the device has an advertiser role and transmits the advertisement packet on three advertising channels. The advertisement packet can contain from 2 to 31 bytes of advertisement data. Advertisement data can be intercepted by the devices that are in Scanning state. Depending on the gathered advertisement data the scanner can demand additional data, initiate the connection with the advertiser, or ignore the advertiser. After the connection is established both devices are in Connection state. Typically, the device that was initially scanning assumes the role of a master device whereas the advertising device becomes a slave device. While connected the devices cannot change their role.

When two devices are connected they form a master/slave pair as shown in Figure 2a. According to Bluetooth 4.0 standard a device cannot simultaneously assume master and slave role. Furthermore, while a master device can connect to several slave devices a slave device can connect only to one master device. This limits the possible network configurations to a P2P pair and to a simple star topology known as piconet topology, as depicted in Figure 2b.

Once the connection is established the devices can transfer data over 37 data communication channels. The data can be sent in unencrypted or encrypted packets that can have 0 to 31 bytes of Protocol Data Unit (PDU) payload [1]. Depending on the devices link layer state the PDU payload can contain the advertis-

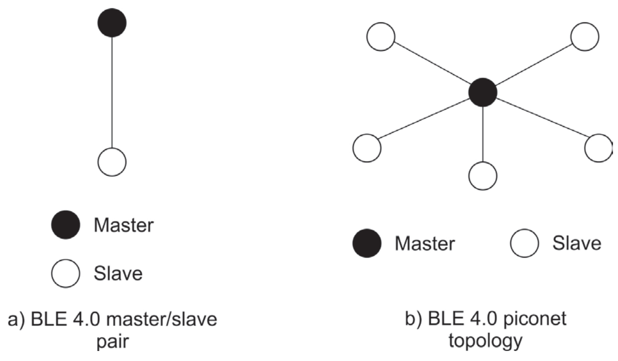

Figure 2: BLE Master/Slave connection options

ing or scanning device addresses and advertise or scanning response data. The encryption is achieved with security manager protocol (SMP) key distribution between master and slave during the pairing process.

The BLE 4.0 stack is composed of several protocol layers which can be divided into two logical entities: Controller and Host, as shown in Figure 3.

The Controller is composed of physical layer (PHY) and link layer (LL). Hardware implementation of Physical Layer is composed of a balun and an antenna which forms a radio frequency (RF) transceiver while the layers from Link Layer and above reside in a SoC.

The RF transceiver as a part of PHY is responsible for data transfer between devices.

The link layer from the Controller manages the transmission and reception of data packets with respect to the flow control and connection parameters established with other nodes. If device is in scanning, advertising or connection mode the link layer manages the data receiving and transmitting. It also provides first line of security by allowing the data exchange only from selected nodes. The connectivity between Controller and Host stack is managed by host controller interface (HCI).

Following the HCI layer are the Host layers. The logical link control and adaptation protocol (L2CAP) manages the data multiplexing for the attribute protocol (ATT) and security manager protocol (SMP). The SMP offers different security modes, data encryption and authentication services. The ATT manages the discovery of server-client attributes and enables the attributes reading and writing functionalities. The discovery and characteristics exchange of the BLE 4.0 protocol services are defined by the generic attribute profile (GATT) framework built on top of the ATT layer. The uppermost layers of BLE 4.0 stack are the generic access profile (GAP) and the applications layers. The GAP manages the communication between lower layers and applications and is responsible for establishing different modes of operation such as scanning and advertising [10].

BLE 4.0 protocol stack provides the device with all the support to operate in master or slave mode individually. Unfortunately, no support is provided for the device to operate in master and slave mode simultaneously thus BLE 4.0 protocol does not support multi-hop data transfer.

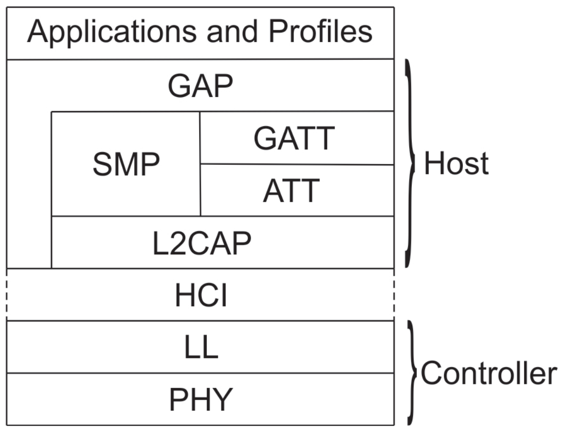

Figure 3: BLE 4.0 protocol stack

3 Current ble 4.0 multi-hop solutions

Although the BLE 4.0 emerged in 2010 there are currently just a few multi-hop implementations. In the first multi-hop implementation [4] the data transfer is achieved by implementing a new multi-hop GATT layer service. A service is divided into two parts: route discovery and data transfer. A transmitting node has a known end-node address so at first a route over intermediate nodes has to be established. At each route generation a gateway node to the target node, the target node, and number of required hops are stored in each node. After route discovery is completed the data is sent through the known route to the target node. The prime challenge at service implementation, as authors emphasized, was advertiser/scanner operational state switching and memory availability. The working multi-hop service was implemented in four nodes and results were presented. In another implementation of multi-hop data transfer [5] a tree network topology is used.

This implementation distinguishes three different node types: root node as main or central device, intermediary node that act as a peripheral or central device depending on connection initiation, and leaf nodes as the peripheral devices in the tree structure. Intermediary node is constructed from two devices: a central and peripheral device, which are interconnected with I^{2}C bus [11]. In their implementation the central device can be connected to at most 3 peripheral devices and with 2 byte addressing the tree network can be extended up to level 5. The tree network topology was successfully implemented and the results were presented. For future work authors emphasize on reducing the negative effects of node failures thus hopefully increasing the robustness of the network.

4 Multi-hop data acquisition mechanism

Data from distant nodes cannot be transferred using a direct connection, for instance if we have three nodes A, B, and C, as shown in Figure 4, where node C is not visible from node A, node A cannot directly acquire data from node C. In order for node A to acquire data from C, node B must first acquire data from C and then relay them to node A. To do this, node B must first become a master node, detect node C using scanning, make a connection, and retrieve data from node C. To relay data it must switch back to slave mode and advertise that the gathered data are available. The switch from slave to master operation in node B is initiated by node A inquiry.

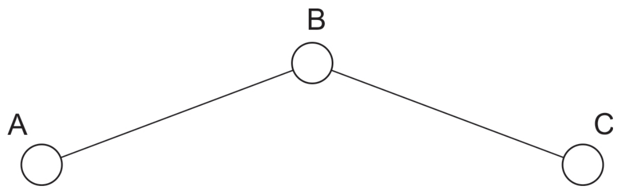

Figure 4: Distant BLE nodes

The basis of our multi-hop communication mechanism is the ability of each node to temporary assume the master role and to acquire the data from not already processed nodes, and relay acquired data to its master node along with its sensor data. Since this procedure is recursively repeated on each node a communication tree is dynamically generated on each data inquiry and can change with each query depending on the nodes availability and their latency.

The multi-hop communication mechanism can be viewed as a two phase protocol:

1. In first phase the communication tree is build. This is started when root node begins to collect available idle nodes. A node in master mode connects to the available idle nodes and sends COLLECT command to claim exclusive access to child node. Child node records connected master node as its parent node in order to prevent connections from other master nodes. This procedure is recursively repeated on each child node: it switches to master mode and start collecting available nodes. First phase is finished when there are no available nodes.
2. In second phase, every non-root node has a parent node which was determined during the first phase. Sensor data are reported back from leaf nodes towards the root node. The report is made in similar fashion as collecting: the node with acquired data from all its children switches back to slave mode and advertises that it has required data. When the parent detects that its child has available data, it connects to the child and initiates data transfer by sending READ command. When the data transfer from all children nodes is finished, the children reinitialize as idle nodes.

In the presented mechanism each master node connects twice to each of its children: first to collect available idle nodes, and then to request acquired data from child node. Since actions of the slave nodes occur when they are connected to a parent node, they are triggered by commands issued by parent node using established data connection. The multi-hop communication mechanism can be viewed as building the communication tree by sending commands towards leaf nodes and gathering sensor data in the opposite direction. The global root node initiates the data acquisition and triggers the generation of the communication network as depicted in Figure 5. Global root node operates as a master node only, thus its functionality can be interpreted as a subset of the sensor node functionality.

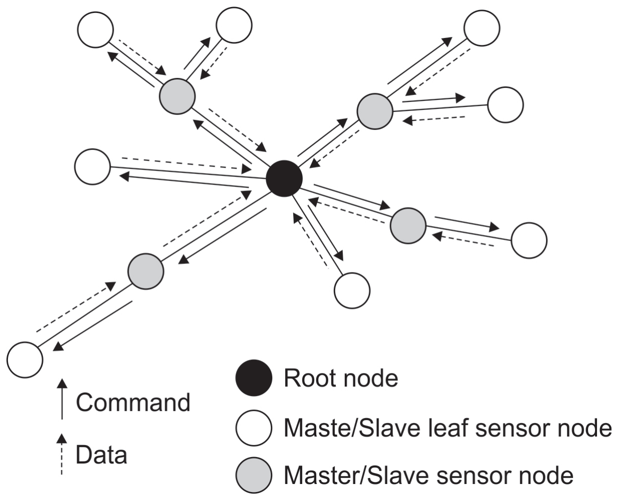

Figure 5: Command and data path in tree topology network

Let us present example with only three nodes to illustrate the concept of our multi-hop mechanism. Node A and node C cannot be directly connected due to long distance or some obstacle, while node B is reachable by both nodes. Let us assume that the data inquiry is started from node A. The behaviour of the multi-hop mechanism is depicted in Figure 6 and Figure 7 where master nodes are denoted by black shading and slave nodes are white.

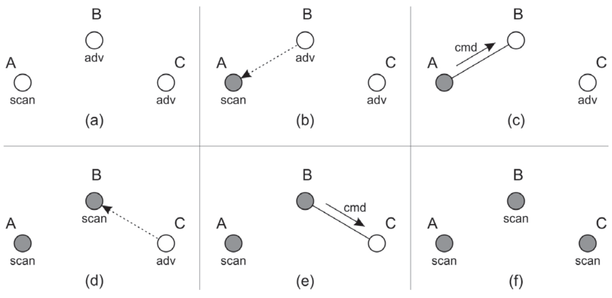

Figure 6: Communication tree formation

Figure 6 describes the communication tree formation of multi-hop communication mechanism:

- Initially all nodes are idle and are advertising their presence (Figure 6a).
- Node A is selected as a root node (inquiry node), node A became a master node and is scanning for neighbour nodes. It detects node B as an idle node (Figure 6b).
- Since node B is idle, node A connects to node B and issues a command to node B to start its own data inquiry (Figure 6c).
- Both node A and node B are scanning, however node C is reachable only by node B. Node B detects node C as an idle node (Figure 6d).
- While node A is still scanning nodes (for the data inquiry results) the node B connects to node C and issues a command to node C to start its own data inquiry (Figure 6e).
- All three nodes are scanning; node A and node B are scanning for data results, while node C is scanning for an idle node (Figure 6f).

The first phase terminates when there is no available idle node. The scanning of the node, which is searching for an idle node, is terminated by a predetermined timeout. The timeout must be long enough to guarantee the detection of an idle node in the vicinity (we used a value of 1 second).

Figure 7 describes the data acquisition of multi-hop communication mechanism:

- After the timeout in node C (Figure 7a), node C reads its sensor data and switches from scanning mode to advertising mode indicating that it has data available.
- Node B is scanning for its child nodes with sensor data and detects that node C have data ready (Figure 7b). Note that while node A is also scanning for nodes with data ready the node C is not its child and would be ignored even if it was detected.
- Node B connects to node C and node C sends its sensor data to node B. Node B adds its sensor readings to the received data from node C (Figure 7c).
- Node B finishes its scanning since it has processed all its children. Then it switches to advertising mode indicating that it has sensor data available. Node A which is scanning for its child nodes with sensor data ready, detects that node B has data ready (Figure 7d).
- Node A connects to node B and node B sends its sensor data to node A. Node A adds its sensor readings to the received data from node B (Figure 7e).
- Since Node A is the root node it report all gathered sensor data and switch from master mode to advertising mode (Figure 7f).

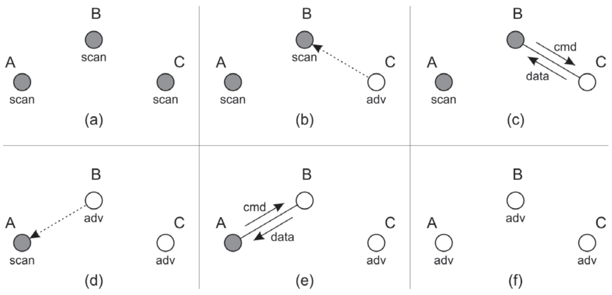

Figure 7: Data acquisition

From the single node point of view the multi-hop data acquisition mechanism is composed of:

- Master mode operation, which is further divided to operations:
  - Collecting all available neighbouring slave nodes and claiming exclusive access to the collected slave nodes. This is achieved by sending COLLECT command to all free slave nodes. Since the master node doesn't know how many available slave nodes are in its vicinity this operation has to be stopped using timeout.
  - Requesting sensor data from child node when the child node has acquired all available sensor data from its descendant nodes. This is achieved by sending READ command.
- Slave mode operation, which consists of following operation:
  - Idle operation: the node advertises that it is available and has no parent node. The node waits for the connection from any master node. After the node is connected to a master node, it accepts only COLLECT command. On

the receipt of this command it records parent node address and advertises parent node address along with its current state
- Serving data operation: the node has all available data from its descendant nodes. It advertises its parent node address and its current state. After the node is connected from the parent node it transfers the acquired data to the parent node. When the sensor data is transferred the node erases parent node address and restarts with the idle operation.

The simplified procedure of the multi-hop mechanism is:

```
while true {
  advertise idle
  while not (cmd = COLLECT) {} // wait for COLLECT command
  record connecting node
  master_mode
  advertise parent and data_ready
  while not (cmd = READ) {}    // wait for READ command
  send acquired data
  erase parent node address
}
```

The master mode operation of the multi-hop mechanism is:

```
start scanning    // switch to master mode
while not timeout {
  if node_available {
    add node to set
    connect
    send COLLECT cmd
  }
}
foreach node in set {
  if node data_ready {
    Connect node
    Send READ cmd
  }
}
```

The multi-hop mechanism was implemented using finite state machine. The states identify the current mode of operation of the node. Initially we used four states: *IDLE*, *COLLECT*, *READ*, and *READY*. States *IDLE* and *READY* states correspond to slave node operation and states *COLLECT* and *READ* correspond to master node operation. However, the resulting connection trees using these states were quite deep. This was because each slave node that entered the COLLECT state on request of a master node immediately started collecting free nodes in its vicinity and thus also competed with its master node. This is undesirable because it results in very deep connection trees with longer data acquisition time. To impose breadth-first generation of the connection tree generation an additional slave state *PREPARE* between the *IDLE* state and *COLLECT* state was introduced. This way the collected slave nodes start collecting remaining nodes after the master node finished the collect operation. Consequently, corresponding master state *TRIGGER* has to be added to control the additional slave state transition.

There are similar problems on the data acquisition part of the mechanism hence the *FREE* state was introduced between *READY* state and *IDLE* state to delay the releasing of slave nodes, and *RELEASE* state between *READ* state and *READY* state in order to control the state transition. The final finite state machine of the multi-hop node is depicted in Figure 8.

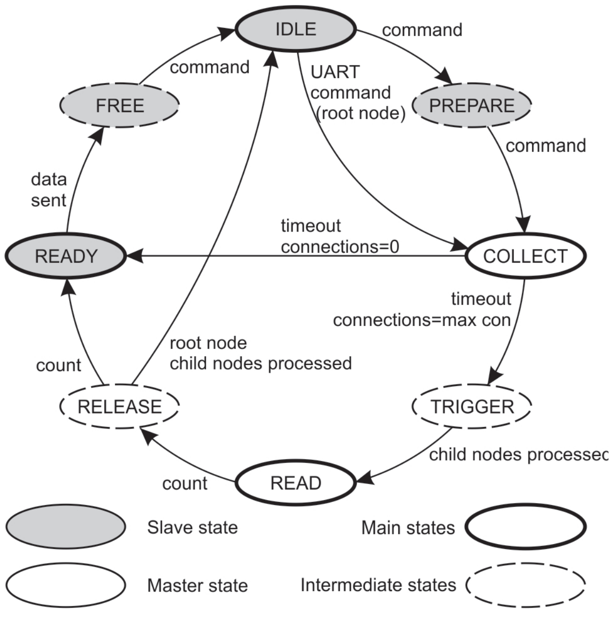

Figure 8: Master and Slave state mechanism

Initially all BLE nodes operate in slave mode and advertise their presence using unique 128bit multi-hop service UUID. Besides the multi-hop service identification master nodes require also additional data: whether the slave node is available and if the node has acquired data from its descendant nodes. While this information could be gathered by establishing the data connection such solution would cause larger delays since a slave node can connect to a single master node and while connected it is inaccessible to other master nodes. Best solution is to advertise current state of the node. Furthermore, to eliminate the unnecessary connections to the nodes, which are already collected by some node the slave node advertises the ID of its parent node. This way the master node can filter out the slave nodes which do not respond to it. Since there are no suitable predetermined advertisement packets in the GATT layer of BLE 4.0 protocol stack a custom advertising packet is used. The final advertising data are shown in Table 1.

Table 1: Advertising packet

ADV packet	Custom		128bit UUID
	Device state	Masters MAC address	
Data Length	1	6	16
Packet length	9		18

The advertising data is composed out of 27 bytes. Each advertising packet occupies one byte for packet type ID, one byte for packet length, and packet data. Device state occupies 1 byte and is a copy of the current state of the device. The master node ID is the MAC address of the master node. The master node ID of the node is initialized to NULL in the *IDLE* state. After first connection from a master node its MAC address is copied into the advertising packet and only connections from this master node is allowed until the node is released. Then the node enters the *IDLE* state and the master node ID is restored to NULL. Consequently, the master node ID of the root node is always NULL.

Generic multi-hop state machine implementation is:

```
while true {
  switch (state) {
    case IDLE:       // slave mode
      if (cmd=PREPARE) {
        parent = master_node
        state  = PREPARE
        advertise(parent,state)
      }
    case PREPARE:     // slave mode
      if (cmd = COLLECT && master_mode = parent) {
        state  = COLLECT
        start_scan()  // switch to master mode
      }
    case COLLECT:     // master mode
      if timeout state = TRIGGER
      else if available_idle(node) {
        connect(node)
        send_cmd(PREPARE)
        add_to_children_set(node)
      }
    case TRIGGER:     // master mode
      if children_set_processed() state = READ
      else if available_prepare(node, my_address) {
        connect(node)
        send_cmd(COLLECT)
      }
    case READ:        // master mode
      if children_set_processed() state = RELEASE
      else if available_ready(node, my_address) {
        connect(node)
        data_request_cmd(READ)
      }
    case RELEASE:     // master mode
      if children_set_processed() {
        state = READ
        advertise(parent,state)
      }
      else if available_free(node) {
        connect(node)
        send_cmd(RELEASE)
      }
    case READY:       // slave mode
      if (cmd = READ && master_mode = parent) {
        send_sensor_data()
        state = FREE
        advertise(parent,state)
      }
    case FREE:        // slave mode
      if (cmd = RELEASE && master_mode = parent) {
        clear parent
        state = IDLE
        advertise(parent,state)
      }
  }
}
```

The states of the multi-hop mechanism are:

- *IDLE* state: the node is in slave mode but it has no parent node therefore it is available. It operates as a slave and advertises its state. On connection from a parent node using the PREPARE command the state changes to *PREPARE* state.
- *PREPARE* state: the node is in a slave mode but it is reserved by the parent node. It advertises its state and its parent node. It accepts the connections only from its parent node and the only viable action is to change to *COLLECT* state by COLLECT command.
- *COLLECT* state: the node is in master mode. In the master mode the advertisement is stopped. It scans the neighbouring nodes, collects available nodes, records them and sends them TRIGGER command for an exclusive access. After reasonable timeout, the collecting process is stopped. The node then switches to *TRIGGER* state if there are some collected nodes, otherwise the node switches to *READY* state.
- *TRIGGER* state: the node is in the master mode. The collected nodes are triggered to a *COLLECT* master state in order to acquire data from distant nodes. The node scans the neighbouring nodes and when a slave node, advertising that it was collected by this node, is found, *TRIGGER* command is given to this slave node. After all collected slave nodes are processed, the node switches to *READ* state.
- *READ* state: the node is in the master mode. It scans the neighbouring nodes and when a slave node, advertising that it has data ready for this node, the read process is triggered using *READ* command. After all collected slave nodes are processed, the node switches to *RELEASE* state.
- *RELEASE* state: the node is in the master mode. It scans the neighbouring nodes and when a slave node, advertising that it could be freed is detect-

ed, *RELEASE* command is given to free the collected slave node. After all collected slave nodes are processed, the node switches to *READY* state.

- *READY* state: the node is in the slave mode. When the data connection from the parent node is established the acquired data along with the local node sensor data are sent to the parent node. After the data transfer is completed the node switches to FREE state.
- *FREE* state: the node is in slave mode. It advertises that it could be freed. On the receipt of *RELEASE* command from the parent node it re-initializes all data structures and switches to *IDLE* state.

Initially all BLE nodes operate in slave mode. To initiate the data acquisition on a selected BLE node a command is issued on its UART peripheral interface to assume the role of the root node. At this point the root node starts collecting neighbouring nodes, acquires their data, and reports the acquired data over the UART peripheral interface.

5 Implementation and results

For testing and evaluation of the multi-hop communication mechanism a custom measurement board was developed. Peripheral hardware consists of power supply, USB and UART interface, LM75B temperature sensor [8], and a Bluegiga BLE113 breakout board [7].

Table 2: BLE113 specifications

Device	Bluegiga BLE113
Supply voltage	2V – 3.6V
Peripheral interface	UART, SPI, I2C,PWM,GPIO, ADC
TX consumption	18.2mA
RX consumption	14.3mA
Sleep mode consumption	0.4uA
TX power	0dBm to – 23dBm
RX sensitivity	-93dBm
SoC	CC2541 chip: - 8051 CPU, - 32MHz clock, - 128kB or 256kB Flash - 8kB SRAM
Smart stack	GAP, GATT, L2CAP and SM Bluetooth smart profiles
Max connections in master mode	8
Throughput	100kbps +

The main specifications of BLE113 module are presented in Table 2. The module is based on a CC2541 SoC from Texas Instruments with integrated 8051 CPU, Bluetooth radio and software stack [12]. Module offers a wide range of peripheral interface on 17 configurable I/O ports, two configurable I/O ports with 20 mA driving capability and two ports that can be used as digital I/O or I^{2}C communication channel. BLE113 module configurable ports offer a broad assortment of peripheral functionality:

- UART and SPI communication,
- three timers that each can offer timer, counter or PWM functionality,
- up to 12bits of resolution ADC converter

BLE113 is capable of managing 8 slave devices in master mode with over 100kbps throughput. Module is mounted on the BLE113 breakout board which is installed on a peripheral hardware board with temperature sensor. The communication between BLE113 module and temperature sensor is implemented with I^{2}C interface.

Software was written in BGScript language [13], which is event-driven BASIC-like application scripting language. Each event handling subroutine can consist out of numerous instructions and each instruction can be executed in 1-2ms [13]. Since the BLE113 module incorporates 8051 MCU with several GPIO ports as well as some general purpose communication protocols (e.g. I^{2}C, SPIO) we were able to develop a fully standalone BLE device without the external CPU and/or additional memory. BGScript incorporates additional APIs for managing Bluetooth connections and various hardware interfaces. In this regard I^{2}C data transfers for temperature sensor readings, UART for debugging purposes, and GPIO for external triggering were used.

Although the BGScript provides most common programming structures and relatively rich APIs it has several limitations. The main limitations are the limited amount of available RAM used for variables, the limited amount of available non-volatile memory for program storage and most notably the limited amount of program stack. The program stack is limited to at most 100 bytes which does not allow deep function nesting nor moderate size function parameters.

For the evaluation process seven BLE devices were used from which one was used as root device and six were used as child nodes. The collected data from each child node was a temperature stored as a short integer occupying two bytes.

Total amount of program code was around 60KB which is approximately 23% of available non-volatile memory [6]. The program code was the same on all nodes in the network. To initiate the data collection through the network one node was connected to the computer. This node became a root node once it received the command to start the data collection over the USB/UART communication port. Detailed mechanism implementation in BGScript language for our measurement board is presented in technical report [14].

In the first experiment set we tested completely connected wireless sensor network where each pair of nodes are connectable. The nodes were placed in the same room at a random distance of a few meters from each other. Using such placement, we conducted two sets of tests. In the first test set the maximum number of connected slave nodes were limited to two. This way we forced the master nodes to connect to at most two child nodes which resulted in a binary tree network topology. In the second test set we released the maximum connection limitation. Since all nodes were relatively close, they formed a star topology. Both test sets consist of 2000 temperature data acquisitions and were collected automatically using python scripts on a PC computer. There was a pause of few seconds between each data acquisition to mitigate errors that might arise from environment disturbances.

In the first test set, when the number of child nodes was limited to two, the formed networks were binary tree networks with maximal depth of 3 and various fan-out shapes. In the test the isomorphic binary trees were collected together. Representations of the most frequent isomorphic network trees in the first test set are shown in Figure 9 whereas the frequency of the isomorphic network formation is presented in Table 3. The most common binary tree network was a balanced tree shown in Figure 9d. The data collecting time in balanced tree network varied from 13 to 16 seconds.

Table 3: Frequency of binary tree network formation

Binary tree network	9a	9b	9c	9d
Occurrences	33	44	413	1205

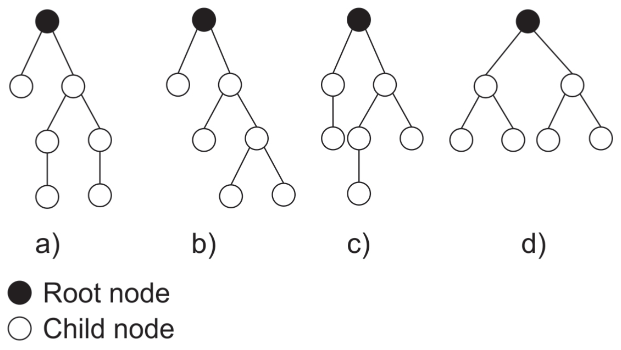

Figure 9: Binary tree networks

In the second test set performed without connection limitation, nodes formed a star topology structure where all child are nodes connected directly to root node. With this mode of operation, no multi-hop data transfer was conducted. The data collecting time varied from 20 to 38 seconds.

During 2000 data acquisition requests a total of 35 different tree network topologies were created.

In both test sets there were some cases where not all nodes were connected. This can be attributed to the environment disturbances when some nodes become inaccessible. Since we implemented only basic error handling in the multi-hop mechanism such errors might span over few consecutive tests.

Finally, we tested the mechanism in real-life environment. In the real-life environment the sensor nodes do not form a complete graph and if there is a limitation on the number of child nodes some nodes may become inaccessible. Therefore in the real-life environment the limitation on the number of the child node must be omitted. In our test the sensor nodes were placed throughout the building where they formed a sensor network shown in Figure 10.

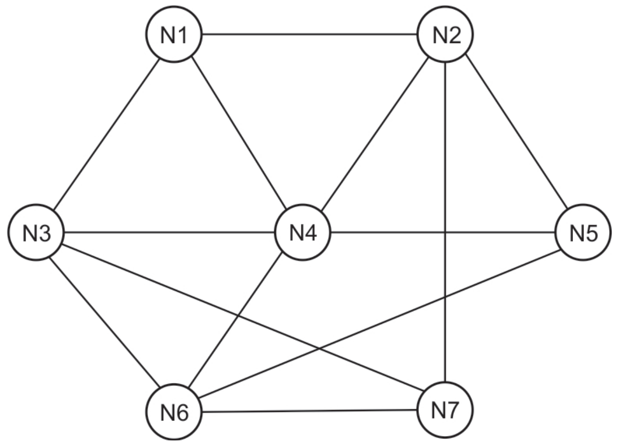

Figure 10: Sensor network connection graph

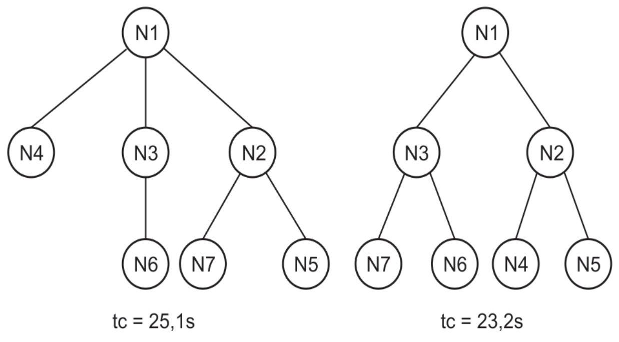

Figure 11: Tree networks with N1 as the root node

We conducted two sets of tests. In the first test set the N1 sensor node was chosen as a root node. In the Figure 11 two typical tree networks are depicted. The maximal depth of connection trees was 3. The data collecting time for these connection trees were 25,1 and 23,2 seconds respectively.

In the second test set N7 was chosen as the root node. In the Figure 12 two examples of tree networks are shown. The data collecting time for these connection trees were 27,2 and 25,5 seconds respectively.

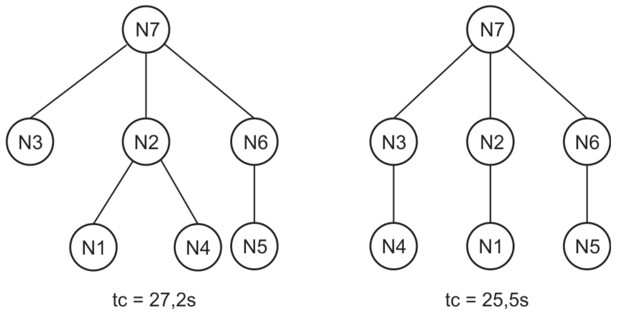

Figure 12: Tree network with N7 as the root node

6 Conclusions

As Bluetooth Low Energy technology continues to gain its use in wireless sensor network the demand for viable data transfer mechanisms over wireless sensor network also increases.

In this paper a multi-hop communication mechanism for data acquisition in Bluetooth Low Energy 4.0 ad-hoc wireless sensor network is presented. The goal was to extend the range of BLE wireless sensor network by relaying data over a series of identical sensor nodes in the wireless network.

Initially, the mechanism was coded using numerous nested functions and medium-size function parameters that increased program stack consumption. That led to program stack overflow and subsequently to unpredictable behaviour of BLE113 module such as device reboot or variable corruption. The code was later flattened to eliminate program stack overflows.

We demonstrated that our multi-hop data acquisition mechanism can be used for a sensor data collection in tree topology networks. Although the sensor data collection can take a noticeable amount of time the mechanism can be nonetheless used for relatively slow processes such as temperature measurements.

In future we will improve the reliability of the mechanism and include a time-to-live mechanism that will improve robustness and allow the use of mechanism in more dynamic environments.

7 Acknowledgments

The authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0098).

8 References

1. Bluetooth Core Specification, Version 4.0, SIG, June 2010.
2. M. Ilyas and I. Mahqoub, »Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems,« CRC press LLC, 2005.
3. G. Kortuem, F. Kawsar, D. Fitton, and V. Sundramoorthy, »Smart objects as building blocks for the internet of things,« Internet Computing, IEEE, vol. 14, 44-51, 2010.
4. K. Mikhaylov and J. Tervonen, »Multi-hop Data Transfer Service for Bluetooth Low Energy«, *2013 13th International Conference on ITS Telecommunications* (*ITST*), IEEE 2013.
5. B. K. Maharjan, U. Witkowski and R. Zandian, »Tree network based on Bluetooth 4.0 for wireless sensor network applications,« in Education and Research Conference (EDERC), 2014 6th European Embedded Design in, pp. 172-176, 2014.
6. BLE113 *Bluetooth* Smart Module. Available at: https://www.bluegiga.com/en-US/products/ble113-bluetooth-smart-module/, visited on 15.5.2016
7. BLE113 Bluetooth Low Energy Breakout, Available at: http://www.inmojo.com/store/ jeff-rowberg/item/ble113-bluetooth-low-energy-breakout/, visited on 15.5.2016
8. LM75B Digital temperature sensor and thermal watchdog, Available at: http://www.nxp.com/documents/data_sheet/LM75B.pdf, visited on 4.6.2016
9. K. Mikhaylov, N. Plevritakis, and J. Tervonen, »Performance analysis and comparison of Bluetooth Low Energy with IEEE 802.15.4 and SimpliciTT,« J. Sens. Actuator Networks, vol. 2, no. 3, pp. 589-613, Aug. 2013.
10. Carles Gomez, Joaquim Oller and Josep Paradells, »Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology«, Sensors, 29 August 2012.

11. Wikipedia – I^{2}C, Available at: http://en.wikipedia.org/wiki/I2C, visited on 22.8.2017
12. »2.4-GHz Bluetooth Low Energy and Proprietary System-on-Chip«, Texas Instruments, June 2013
13. BGScript, Available at: http://www.hmangas.com/ Electronica/Datasheets/Bluetooth%20Module/BLE112/Bluetooth+Smart+BGScript+Developer+Guide.pdf, visited on 18.6.2016
14. Branko Skočir, Gregor Papa, Anton Biasizzo. Multi-hop communication code, IJS technical report No. 12380, 2018.

Arrived: 19. 12. 2017
Accepted: 03. 05. 2018