2020-06

Performance Evaluation of Bluetooth in a Wireless Body Area Network for Practical Applications

Reich, O

http://hdl.handle.net/10026.1/17132

10.1109/sam48682.2020.9104399
2020 IEEE 11th Sensor Array and Multichannel Signal Processing Workshop (SAM)
IEEE

Performance Evaluation of Bluetooth in a Wireless Body Area Network for Practical Applications

1st Olaf Reich, 2nd Erik Hübner, 3rd Bogdan Ghita, 4th Matthias Wagner, 5th Jörg Schäfer
Department Computer and Engineering Sciences
Frankfurt University of Applied Sciences
Frankfurt am Main, Germany
olaf.reich@fb2.fra-uas.de, ehuebner@stud.fra-uas.de, mfwagner@fb2.fra-uas.de, jschaefer@fb2.fra-uas.de
University of Plymouth
Plymouth, United Kingdom
bogdan.ghita@plymouth.ac.uk

***Abstract*—This work focuses on practical research in the field of Bluetooth 5.0 and the maximum number of Bluetooth sensors nodes in a wireless body area network (WBAN) for human activity recognition (HAR). As network topology M:1 multi-pairing is used. For the best of our knowledge no direct related work in this field can be found. We investigated how many Bluetooth connections can be simultaneously connected with different smartphones (iOS) with real sensor data and in which time intervals the data can still be processed and stored. Our results show that a maximum of 14 Bluetooth devices can be connected under practical conditions. A stable transmission with 48 bytes per package are found with most reliability in an minimal interval of 40 ms per node.**

***Index Terms*—Bluetooth, Internet of Things (IoT), performance-evaluation, wireless body area network (WBAN), Special Interest Group (SIG), NINA, sensor networks**

I. Introduction

In the Internet of Things (IoT) and Industry 4.0 the Bluetooth standard is a widely used communication protocol [1]. Private concerns like headsets, smartphones or smart homes are the most commonly usage for it. Additionally it is more and more applied in professional settings like smart factory or eHealth. Bluetooth is still actively being researched and developed. The current clusters of research are real-world implementations, lab implementations and theoretical concepts [2]. Since 1999 Bluetooth standards are managed and released by the Special Interest Group (SIG). The current Bluetooth specification has the version 5.1 [3]. Version 5.0 and later versions contain more features like the Bluetooth Low Energy (BLE) technology which was formerly introduced in Bluetooth version 4.0 [4]. In July 13 2017 mesh network capability was adopted. The introduction of these features yields new applications, possibilities and research fields for IoT [5]. An overview of the changes between the different Bluetooth versions can be found in the work of Yin et al. [2].

The Bluetooth standard supports different topologies as shown in figure 1. The early specifications version 1.0 up to 4.2 focused on point-to-point (Pairing) and point-to-multipoint (Broadcasting) connections between devices. The mesh topology has been part of the specification since version 5.0. Often many sensors are used to detect human motions [6] [7]. All distributed sensors send their data to a central device, like a smartphone or tablet. So our focus lies on the M:1 topology (Multi-Pairing).

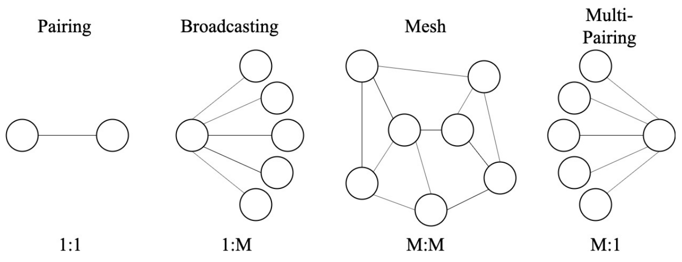

Fig. 1. Topologies: *There are various possible Bluetooth topologies. The simplest is 1:1 or 1:M (M = Many). An M:M topology creates a mesh network. Using Bluetooth in a WBAN the M:1 topology is interesting.*

Furthermore an important point for the transmission of many data with very short measuring cycles is the choice of the right connection mode. The Bluetooth version 5.0 allows to configure various types of procedures in order to retrieve values from connected devices [8]. One possibility is to perform a simple read. This procedure involves two communication steps. First, the client desiring a characteristic value of a server has to send a read request to the server. Afterwards the client will get a read response from the server in case of success or in case of failure an error response. Another possibility is to configure indications which also require two steps. By using the procedure, the server inhabiting the desired characteristic value, sends an indication including the value to the client. It sends the indication to the client whenever its value changes. The client responses to the server with a confirmation. The configuration of notifications as procedure is a third type. It is basically the same as described for indications but the server does not expect any confirmation from the client. Following, this procedure just involves one communication step from the server to the client. This is very interesting for applications focusing on high performance.

In the present paper we evaluate the maximum number in multi-pairing topology and the performance of several parameters such as number of nodes, different smartphones and smallest stable measurement interval. Section II discusses the related work. An overview of the used hardware, the configuration and the setup of the experiment will be shown in Section III. Section IV concludes the paper.

II. Related Work

There is mainly research working with the theoretical specifications, features and architecture of Bluetooth [2] [9] [10]. Since the introduction of mesh topology in Bluetooth version 5.0, studies have focused on mesh networks and their performance optimization. For example, there is research on investigating real-time communication in a mesh network [11]. Additionally, there is some extensive research about tuning power consumption in order to increase lifetime of a battery from 9.55 days to 2.32 year [12]. Furthermore, as seen in the work of Dian et al. [13] the important part of time synchronization in a mesh network between BLE devices also gets addressed. They figured out that poor environmental conditions can lead to divergence of more than 17 µs in time synchronication. Nevertheless there has been research regarding the number of simultaneous Bluetooth connections and performance analysis. As stated by Gomez et. al. [14], they found out that the theoretical maximum number of simultaneous connected Bluetooth devices lies between 2 and 5917. Moreover, they detected that there is at minimum 676 µs latency to receive sensor data.

III. Method and Environment

A. Software

We used Python as programming language for the experiment. Scripting languages like Python are popular for data analysis, data acquisition and simple data processing. The simple scripting language allows for easy and fast programming. The test setup can be modified quickly. In addition, python does work without a graphical user interface and can be executed directly on the smartphone by using the *Pythonista*[1] application. This means that the development of a complete mobile application is not necessary. For a part of the experiment we needed to use a lightweight database. We chose *sqlite* because it does not need a separate server process.

B. Hardware

For the tests we used different hardware. As hardware on the receiver side different *iPhone* types are tested and on the transmitter side several *Arduino Nano 33 BLE Sense* were used.

1) iPhone: Since the technical data is not fully published on the *Apple* website, we have used the freeware application *x-CPU* to gain hardware information. Data sheets of the Bluetooth chips were not available. For our experiment we had several smartphones at our disposal: *iPhone 6s*, *iPhone X*, *iPhone XR*, *iPhone 11*, and the *iPhone 11 Pro Max*. Our script to perform the experiments is executable on all devices. Since the *iPhone X* and the *iPhone XR* as well as the *iPhone 11* and the *iPhone 11 Pro Max* use the same Bluetooth chip, the experiments were only performed on one type each. Table I gives an overview of the used devices. Section V explains the performance analysis of the proposed method through extensive testing and presents our experimental results.

[1]Pythonista 3 is a full python IDE for iOS

TABLE I
Overview of used smartphones and identified hardware

	iPhone 6s	***iPhone XR***[a]	***iPhone 11***[b]
Bluetooth	version 4.2	version 5.0	version 5.1
Controller	Broadcom BCM4350	USI339S00397	USI339S00648
BLE	yes	yes	yes
iOS	13.2.3	13.2.3	13.2.3
CPU Type	Apple A9	Apple A12	Apple A13
Core Count	2	6	6
Core Speed	1848	2x2490 4x1590	2x2660 4x1730
Memory	2 GB	3 GB	4 GB

[a] Same as iPhone X (different CPU A11) [b] Same as iPhone 11 Pro Max

2) Arduino Nano 33 BLE Sense: For our tests we used 20 *Arduino Nano 33 BLE Sense* as sensor nodes. The devices are equipped with an ARM CPU, which operates with a frequency of 64 MHz and has a working memory of 256KB. There are several sensors on the boards which can be used for data generation. These include a 9 axis inertial sensor [15], barometric pressure [16] and temperature sensor [17]. The exact specifications of the sensors can be found in the respective data sheet. Although the *Ardunio Nano 33 BLE Sense* has additional sensors like microphone, gesture light and proximity we did not make use of them with regard to the HAR. The focus was on the Inertial Measurement Unit (IMU) data. A NINA-B306 chip is installed for Bluetooth communication. The chip uses version 5.0 and supports BLE. The NINA-B306 has an internal antenna and supports a maximum of 20 Bluetooth connections [18].

C. Configuration and Setup

The configuration and the construction are divided into two steps. In the first step, called pretest, the maximum number of Bluetooth nodes is evaluated. In the second step, called main-test, we determined the performance.

1) Pretest: For this purpose 20 Bluetooth sensor nodes were placed next to the smartphone as shown in figure 2. Using a Python script, one Bluetooth sensor node after the other was connected to the smartphone via Bluetooth. It turned out that it was important to disable the nodes sending data packets to the smartphone during this process in order to connect the maximum number of nodes. After no additional Bluetooth sensor node was able to be connected, an already connected Bluetooth sensor node was disconnected to determine if a not yet connected Bluetooth sensor node connects automatically because of the released connection slot. Due to the arbitrary arrangement there were always battery blocks between a Bluetooth sensor node and the smartphone. However, the result remained unaffected by this.

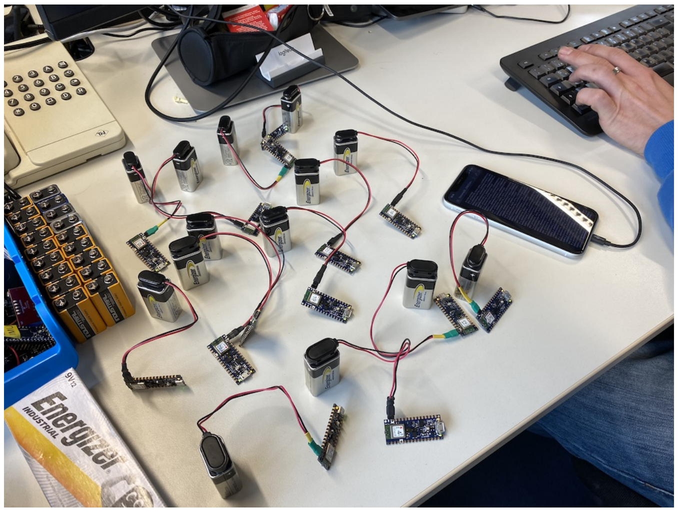

Fig. 2. Test setup for pretest: *Bluetooth sensor nodes are simply placed next to the smartphone.*

2) Main-test: All smartphones have been set to flight mode during the experiment. Only the Bluetooth interface was switched on. All other connections and applications have been disabled via the operating system. In addition, the smartphones were connected to a charger via a cable to ensure maximum performance. The sensor nodes were operated, as in real use, with 9 volt block batteries. The Bluetooth sensor nodes were positioned in a row opposite to the smartphone as shown in figure 3. There are no barriers between the smartphone and the Bluetooth sensor nodes. This means that the Line of Sight (LoS) is free.

The main-test was divided in two parts. In the first part we identified the optimal time at which different smartphones can handle receiving Bluetooth packages from 14 Bluetooth sensor nodes. For identifying the optimal transmission time we executed several trials where we changed the transmission time of every node from 25 ms to 50 ms. Every single trial took 5 min in total. This was repeated for all smartphones in table I. Each packet which the nodes sent to the smartphone had a size of 48 bytes for simulating realistic sensor measurements. The second part evaluates the optimal transmission time at which a complete data handling (inclusive persistent saving) could be practicably used. To achieve this, we used the smartphone with the best performance from the first part. For the complete data handling we used several methods:

- writing the data into a file without threading
- passing the data into a separate thread where it was written to a file
- passing the data into a separate thread where it was written to a database

For identifying the optimal method we performed several trials with transmission time ranging from 25 ms to 50 ms. Again, each trial took 5 min. Our experience showed that it is not possible to implement multiprocessing on an iOS device. Following it is only possible to use multithreading.

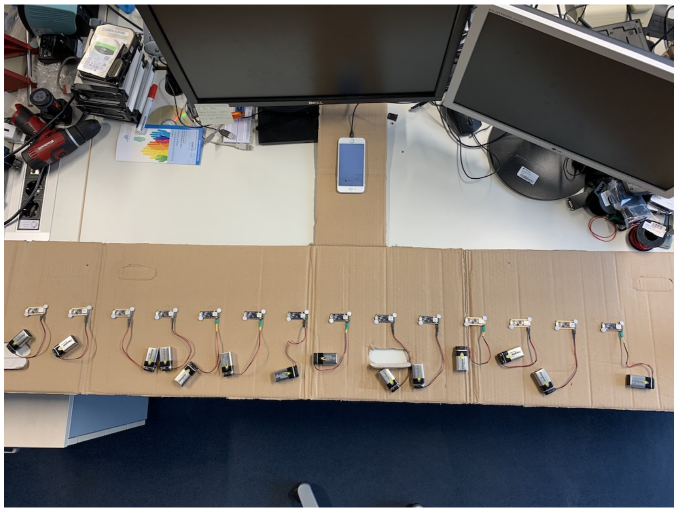

Fig. 3. Test setup for the main-test: *In order to always have the same test conditions, the devices were positioned on a template (light brown). The smartphone can be seen in the top part of the figure in front of the monitor. In the bottom part the 14 battery powered Bluetooth sensor nodes are equally positioned in a row. To avoid slipping between the tests, the nodes were attached to the template. Each space between two nodes was approximately 6 cm and the smartphone was located about 29 cm from the nearest node. The space between the smartphone and the nodes was kept free.*

IV. Result

The results are divided in saving or not saving the received packets using the *iPhone 6s*, *iPhone XR* and *iPhone 11*.

A. Without saving packets

The measurements in figure 4-6 show that the minimal transmission time receiving packets from 14 Bluetooth sensor nodes without saving them differs with the chosen smartphone. It appears that the *iPhone XR* can handle 14 nodes when their transmission time is set to send packets every 0.03 s. In that case, the packets of the nodes could be received with a mean time of 0.031 s and 0.032 s. Setting up the 14 nodes to send packets every 0.025 s resulted in the *iPhone XR* that it could not keep the mean transmission time receiving packets of the nodes equally. It ended up in a broader range from 0.025 s to 0.031 s.

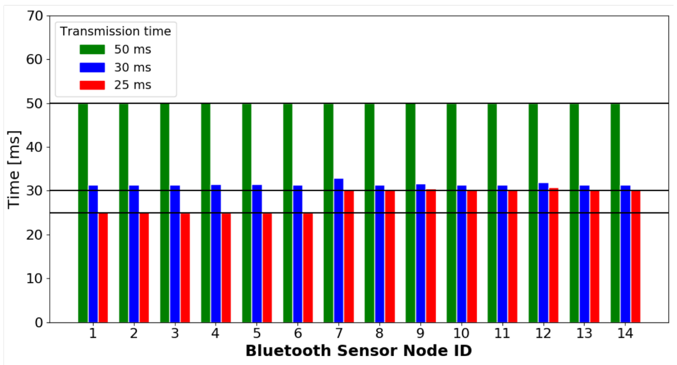

Fig. 4. *iPhone XR* evaluation: The green colour shows the trial where the setup of the nodes was 50 ms, the blue was set to 30 ms and the red was set to 25 ms.

As shown in figure 5 the *iPhone 11* could handle a data rate of 0.05 s by equally receiving all packets from the nodes with a mean of 0.05 s. By turning the transmission time down to 0.04 s it treated the nodes unequally with a range of 0.046 s to 0.051 s.
Figure 6 also reveals that the *iPhone 6s* could not handle all nodes equally at the lowest tested transmission time of 0.05 s. The packets were received with a mean ranging from 0.05 s to 0.064 s.

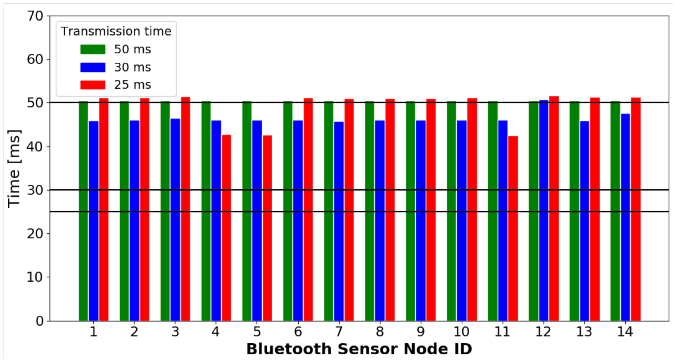

Fig. 5. *iPhone 11* evaluation: The green colour shows the trial where the setup of the nodes was 50 ms, the blue was set to 30 ms and the red was set to 25 ms.

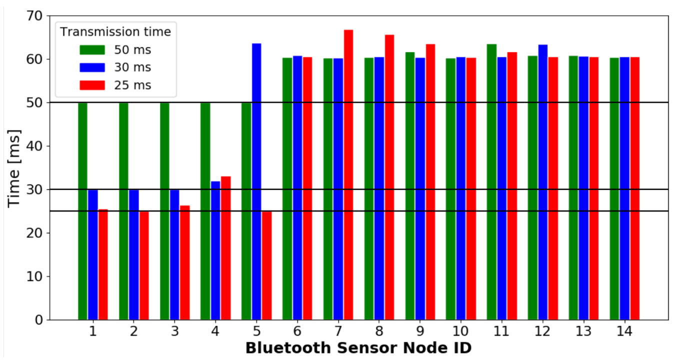

Fig. 6. *iPhone 6s* evaluation: The green colour shows the trial where the setup of the nodes was 50 ms, the blue was set to 30 ms and the red was set to 25 ms.

B. With saving packets

This experiment was conducted with the *iPhone XR* as it performed best in handling receiving packets without saving them. The results in table II reveal an effect of using different methods to save the received packets persistent in the minimal possible transmission time.
The first trial, called A, was to simply write the content of incoming packets to a file as soon as they were received. Setting the transmission time of the 14 nodes to 0.05 s resulted in handling the packets of the distinct nodes not equally, but at different mean transmission times. The packets of four nodes were written at a mean interval of 0.051 s with a deviation ranging from 0.035 s to 0.038 s. However, the packets of the other ten nodes were written at a higher mean interval of 0.092 s and 0.093 s with a deviation of 0.027 s up to 0.036 s.
The second trial, called B, implements the process to write to a file in an own thread on the *iPhone XR*, shows that the packets of all 14 nodes were written at the same mean interval and thus were treated equally. By setting the transmission time of the nodes to 0.04 s, the packets were written at a mean interval of 0.091 s on the *iPhone XR*. The deviation ranges between 0.089 s and 0.093 s.
The third trial, called C, was executed by repeating the second trial, but instead of writing to a file the content of the receiving packets was written to a database. As shown in table II by using this method it was possible to set up a transmission time of 0.035 s on the nodes while also saving the packets at a mean time of 0.035 s to 0.036 s. The deviation lies between 0.026 s to 0.048 s. In addition, trying to adjust the transmission time of the nodes to 0.025 s while using the *iPhone XR* resulted in an unequally treating of the nodes. The packets were saved at a mean time ranging from 0.027 s to 0.034 s with a deviation between 0.025 s to 0.047 s.

TABLE II
MINIMAL TRANSMISSION TIME SAVING INCOMING PACKETS BASED ON THE METHODS A (WRITING TO A FILE), B (THREADING AND WRITING TO A FILE) AND C (THREADING AND WRITING TO A DATABASE) USING AN *iPhone XR*.

Trial	Time [s]	Devices [$\sum$]	Mean [s]	Deviation [s]
A	0.050	4	0.051	0.035 - 0.038
A	0.050	6	0.092	0.030 - 0.036
A	0.050	4	0.093	0.027 - 0.035
B	0.040	14	0.091	0.089 - 0.093
C	0.035	2	0.036	0.029 - 0.039
C	0.035	12	0.035	0.026- 0.048
C	0.025	3	0.027	0.025
C	0.025	1	0.028	0.040
C	0.025	7	0.032	0.024 - 0.047
C	0.025	2	0.033	0.024
C	0.025	1	0.034	0.025

V. CONCLUSION

The experiments revealed that a maximum of 14 Bluetooth sensor nodes can be connected to a smartphone using BLE. This is in context of HAR analysis beneficial, as it can be an advantage to have the ability to equip all body extremities with a node. The experiments also revealed that the *iPhone XR* performed best compared to the *iPhone 11* and *iPhone 6s*. The minimal transmission time receiving packets of all 14 nodes at an equal interval is 0.03 s. It is crucial for later data analysis to save the data in a persistent way. Therefore, it is promising that the minimal possible transmission time at which the *iPhone XR* is able to save the incoming packets of the 14 nodes at an equal interval can be achieved at a transmission time of 0.035 s.
Repeating the experiments when the sensors are attached to a persons body in a realistic test scenario, extending the test duration and increasing the package size should be considered next.

References

[1] K.-H. Chang, ”Bluetooth: a viable solution for IoT? [industry perspectives],” IEEE Wireless Communications, vol. 21, no. 6, pp. 6–7, 2014.

[2] J. Yin, Z. Yang, H. Cao, T. Liu, Z. Zhou, and C. Wu, ”A Survey on Bluetooth 5.0 and Mesh: New Milestones of IoT,” ACM Transactions on Sensor Networks (TOSN), vol. 15, no. 3, p. 28, 2019.

[3] Special Interest Group (SIG), ”Bluetooth Core Specification v 5.1,” Bluetooth version 5.1 standard, Volume 0, January 2019.

[4] S. Raza, P. Misra, Z. He, and T. Voigt, ”Bluetooth smart: An enabling technology for the Internet of Things,” in 2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 155–162, IEEE, 2015.

[5] Special Interest Group (SIG), ”Bluetooth Technology Website.” https://www.bluetooth.com/. [Access] 2019-12-02.

[6] L. la Blunda, D. Corral-Plaza, M. Wagner, G. Ortiz and I. Medina-Bulo, ”Distributed real-time based Human activity analysis System,” Proceedings of the 16th International Conference on Applied Computing, Cagliari (Italy), November 2019.

[7] D. Corral-Plaza, O. Reich, E. Hübner, M. Wagner and I. Medina-Bulo, ”A Sensor Fusion System Identifying Complex Events for Localisation Estimation,” Proceedings of the 16th International Conference on Applied Computing, Cagliari (Italy), November 2019.

[8] Special Interest Group (SIG), ”Bluetooth Core Specification v 5.0,” Bluetooth version 5.0 standard, Volume 0, December 2016.

[9] M. Collata, G. Pau, T. Talty and O. k. Tonguz, ”Bluetooth 5: A concrete step forward toward the IoT,” IEEE Communications Magazine, vol. 56 no.7, pp 125-131, 2018.

[10] P. Ray and S. Agarwal, ”Bluetooth 5 and Internet of Things: Potential and Architecture,” International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), pp. 1461-1465, 2016.

[11] L. Leonardi, G. Patti and L. L. Bello, ”Multi-Hop Real-Time Communications Over Bluetooth Low Energy Industrial Wireless Mesh Networks,” IEEE Access journal, vol. 6, pp. 26505-26519, 2018.

[12] A. Liendo, D. Morche, R. Guizzetti and F. Rousseau, ”BLE Parameter Optimization for IoT Applications,” IEEE International Conference on Communications (ICC), pp. 1-7, 2018.

[13] F. J. Dian, A. Yousefi and K. Somaratne, ”Performance evaluation of time synchronization using current consumption pattern of BLE devices,” IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), pp. 906-910, 2018.

[14] C. Gomez, J. Oller and J. Paradells, ”Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology,” Sensors journal, no 12, pp. 11734-11753, 29 August 2012.

[15] STMicroelectronics, ”iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer,” LSM9DS1 datasheet, Revision 3, March 2015.

[16] STMicroelectronics, ”MEMS nano pressure sensor: 260-1260 hPa absolute digital output barometer,” LPS22HB datasheet, Revision 6, June 2017.

[17] STMicroelectronics, ”Capacitive digital sensor for relative humidity and temperature,” HTS221 datasheet, Revision 4, August 2016.

[18] ublox, ”NINA-B3 series: Stand-alone Bluetooth 5 low energy modules,” UBX-17052099 datasheet, Revision R06, May 2019.