BLE Beacon with Data Accumulation Functionality

KOBAYASHI Kakeru and FUJITA Satoshi

Department of Information Science, Graduate School of Advanced Science and Engineering, Hiroshima University, Kagamiyama 1-4-1, Higashi-Hiroshima, 739-8527, Japan

Abstract. This research looks into the implementation and utilization of BLE beacons with data accumulation functionality. BLE applications are typically divided into two categories: one that broadcasts a fixed value for each device, and the other that identifies contact with a particular user by scanning the data broadcasted in its surroundings. The proposed framework, however, follows a different approach where each beacon scans for data broadcasted in the area, then changes state based on the scan result, and finally, broadcasts data based on the new state. This can be regarded as an extension of the BLE mesh standard, which enables data flooding through repeated scanning and broadcasting. The paper also explores two practical use cases of the technology: detecting and announcing congestion around devices, and a stamp rally application.

Keywords: Bluetooth Low Energy, advertisement, data accumulation, congestion detection.

1 Introduction

Recently Bluetooth Low Energy (BLE) is widely used for periodically broadcasting binary data from a beacon installed at a location to its surroundings. The data transmitted by a beacon can be easily picked up by nearby devices such as smartphones equipped with BLE capabilities and the advertisement can contain a fixed ID (in the case of iBeacon [12]) or a shortened URL of a website related to the installation location (in the case of Physical Web [17]), depending on the application. The main advantage of advertisement used in BLE protocol is that there is no need to pre-establish a connection between the sender and receiver, while in addition to the advertisement-based communication, BLE supports connection-based protocol to provide secure communication encrypted with 128-bit AES in CCM mode.

BLE communication has a wide range of applications, such as providing directions in a museum or promoting products in a retail store. In these scenarios, it is important to accurately identify the location of the beacon from the received data, which is typically done by sending a fixed value depending on the location. Another emerging use case of BLE communication is the contact trace of smartphone users to suppress the spread of COVID-19. In such an application, a unique

⋆ A part of this paper was presented at CANDAR 2021 as "An Enhancement of Physical Web with Stateful BLE Beacons" by the same authors.

DOI: 10.5121/ijcnc.2023.15501

random token is assigned to each user and broadcasted to the surrounding area, allowing proximity between users to be detected by scanning the advertisement. To maintain user privacy, the relationship between the token and the user must be securely managed on the server and to identify the date and time of contacts in detail, the transmitted data is encoded with both the user's identifier and the time of transmission. The BLE mesh standard released in 2017 shows another direction for the future research in which the notion of data routing is supported. In this protocol, data routing occurs in a way that a device can both receive data from a nearby device and advertise it to its surroundings, where such a dynamic role switching in a single node is a new addition to BLE 5.0.

This paper proposes a new framework that utilizes BLE devices as an *accumulator* of peripheral information. In this framework, state transitions are triggered by input in BLE beacons, and a cycle of *information collection, state transition, and dissemination* is repeated by the beacon. Information collection can be achieved through ordinary scanning, and dissemination can be achieved through ordinary advertisement. Thus the challenge in this framework lies in efficiently implementing state transitions triggered by input, and it is also important to draw concrete use cases for the framework. It is worth noting that the simple cycle of *collection and dissemination* without state transitions is already in practice, such as in the previously mentioned BLE mesh standard. In this sense, our proposal could be regarded as an extension of the BLE mesh standard.

In this paper, we summarize findings related to these issues. At first, we study three different ways of implementing state transitions, and show that the storing of the state inside the beacon and updating it with test-and-set operation is promising (this part is an extension of the paper we presented at [14]). Next, we consider two applications of the proposed framework: congestion detection/announcement around beacons and stamp rally application. The former uses a mechanism similar to the contact tracing to determine the number of smartphone users around a beacon, and advertises the estimated congestion to the surrounding users. There are existing applications that upload the contact information detected by BLE devices to a server and announce the predicted congestion status calculated on the server through the Web, but a big difference from these methods is that the beacon itself *advertises* the results it has collected and calculated. In fact, there is great potential for a method that quickly notifies (only) those who are currently at the location of the beacon of the processing results. For example, a smartwatch that detects arrhythmia, such as ventricular fibrillation, can notify an alert to the surrounding beacons, and a beacon that detects "a user in need of rescue is nearby" can announce the location of the nearest AED to the users at that location. In the stamp rally application, on the other hand, each beacon can give individual stamp to the users who pass near the beacon using connection-based communication. Each stamp can contain hints for solving the quests presented in the stamp rally, and

the beacons can have states that allow for more flexible and precise hints based on the distribution of the number of people who have passed through the beacon, the time of day, and other factors. These two applications were implemented on a firmware-less IoT system called Obniz.

The remainder of this paper is organized as follows. After reviewing related work in Section 2, Section 3 describes an overview of the BLE protocol. Section 4 discusses the way of implementing stateful BLE beacons. Sections 5 and 6 describe the details of the implementation of the congestion detection system and stamp rally application. Section 7 presents the results of experiments conducted on the implemented system. Finally, Section 8 concludes the paper with future issues.

2 Related Work

BLE (Bluetooth Low Energy) is widely used for its ability to broadcast an identifier through a BLE beacon [7]. This makes it possible for nearby devices, such as smartphones and tablets, to take specific actions when they are in proximity to the beacon. BLE is also used for proximity detection solutions like indoor positioning [15, 6], action recognition [5, 26], and vehicle wake-up systems [20], as well as for mobile crowd sensing and contact tracing [11] (A review of context-aware mobile crowdsensing systems can be found in [22]). Additionally, BLE can be used to connect wireless sensors and receivers in healthcare and smart home applications [25]. However, its use in new, non-beacon applications such as Vehicular Ad Hoc Network (VANET) [2, 24], smart infrastructure [4, 13, 18, 23], multimedia streaming devices [8, 16, 19], and mobile payment systems [1, 21] is limited by its lack of support for high-capacity data transmission, mesh networking, and other factors.

In Building Management Systems (BMS), it is crucial to keep track of both individual activities, like walking and moving, and group activities, such as meetings. Smartphones offer promising possibilities for such Human Activity Recognition (HAR) research. Chena *et al.* [3] focused specifically on Group Activity Recognition (GAR) and proposed a GADAR framework using smartphones and beacons in a smart building environment to recognize a wide range of human activities. The paper delves into the following topics: (1) what human behavior features should be considered in GAR, (2) how smartphones sensors and Bluetooth beacons can be integrated into group activity recognition, and (3) how effective the framework is in addressing GAR compared to other studies.

In [9], a framework named SocializeME was introduced to detect social interactions among users using commercially available devices. This framework collects and analyzes data from BLE signals emitted from wearable devices. The paper describes an experiment with high school students to gather wireless data and correctly label data related to various interaction phases (non-interaction, proximity, and interaction). The experiment was designed to mimic real-life human interactions, and the analysis focuses particularly on the differences in signals emitted, such as frequency

and RSS, from different mobile device configurations and uses (e.g. standing with a smartphone in hand, standing with a smartphone in a pocket, sitting, etc.). The results of the study offer important insights into the practical usage of commercial mobile devices for social interaction monitoring and the characteristics of BLE signals in real-world scenarios. The dataset used in the experiment is available in [10].

3 Overview of BLE Protocol

The communication protocol that utilizes BLE technology is governed by two rules: GAP (Generic Access Profile) which outlines communication procedures, and GATT (Generic Attribute Profile) which determines how data is accessed.

3.1 GAP

In BLE communication, each device (or node) is assigned one of four roles: broadcaster, observer, central, or peripheral. These roles determine the node's function and responsibilities within the communication. It is worth noting that one node can play multiple roles in the same application, depending on the requirements of the communication. A broadcaster, for example, periodically sends out advertising packets to nearby nodes, while an observer listens for these packets that are transmitted by a broadcaster.

The **broadcaster** is responsible for advertising its presence and the data it wants to share, while the **observer** listens and receives the data. This allows the observer to discover and connect to the broadcaster, enabling communication between the two devices[1]. Each advertising packet can contain data of up to 31 bytes in length. iBeacon is a typical implementation of BLE technology that uses a transmitter and receiver to play the roles of broadcaster and observer respectively. The iBeacon transmits advertising packets at a regular interval, and nearby devices that have the iBeacon app installed can receive and interpret the data contained in the packets.

In BLE, the central and peripheral roles are used to create a star-shaped network with a central node at its center. In this network, the **central** can communicate directly with every **peripheral** device in both directions after explicitly establishing a connection. To do this, the central detects advertising packets transmitted by surrounding peripherals, decides whether to add them to the network, and then initiates the pairing process with the selected peripherals. Figure 1 shows such a procedure. This relationship between the central and peripherals is asymmetric,

[1] As per the BLE standard, the transmission interval, which is the time between sending out advertising packets, should be set within a range of 20 milliseconds to 10.24 seconds. A shorter transmission interval results in higher responsiveness, but also leads to an increase in power consumption.

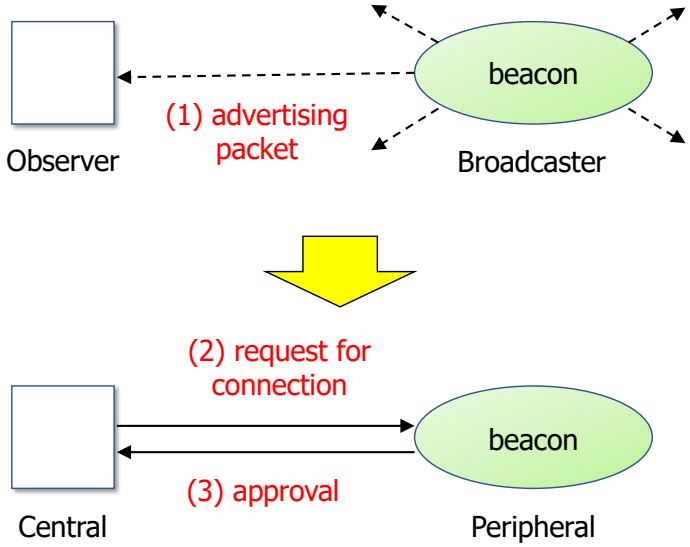


Fig. 1. The BLE protocol, as specified in GAP, defines four roles for nodes to take on: broadcaster, observer, central, and peripheral. Beacon devices transmit radio waves in an omni-directional manner, and their transmission range is approximately 10 meters for Class 2 beacons.

with the central able to connect to any number of peripherals, while each peripheral can connect to only one central. There are several proposals to extend this procedure to create a mesh-structured network, such as the BLE mesh standard supported in BLE 5.0. In this case, a message issued by a node is propagated over the network by using both the broadcaster and observer roles. A node, after receiving an advertising packet from a neighbor, it forwards the received packet to its own neighbors by switching its role from observer to broadcaster. This allows for a more efficient and robust communication between all the devices in the network.

3.2 GATT

Data access in the BLE protocol is based on two entities called service and characteristic, and three types of access patterns, known as Read, Write, and Notify. A **service** is a bundle of one or more characteristics and a **characteristic** is a tuple of value, property, and descriptor. For example, a profile for a multifunctional scale service could have several characteristics such as weight, body fat percentage, and BMI. For each characteristic, the value corresponds to binary data carried by BLE packets, properties are attributes that define the access rights to the values, and descriptors provide additional information. Services and characteristics have a unique identifier of 16-byte length called UUID. In addition, we can use an abbreviated form of UUID (2-byte length) for several services defined in the Bluetooth SIG standard.

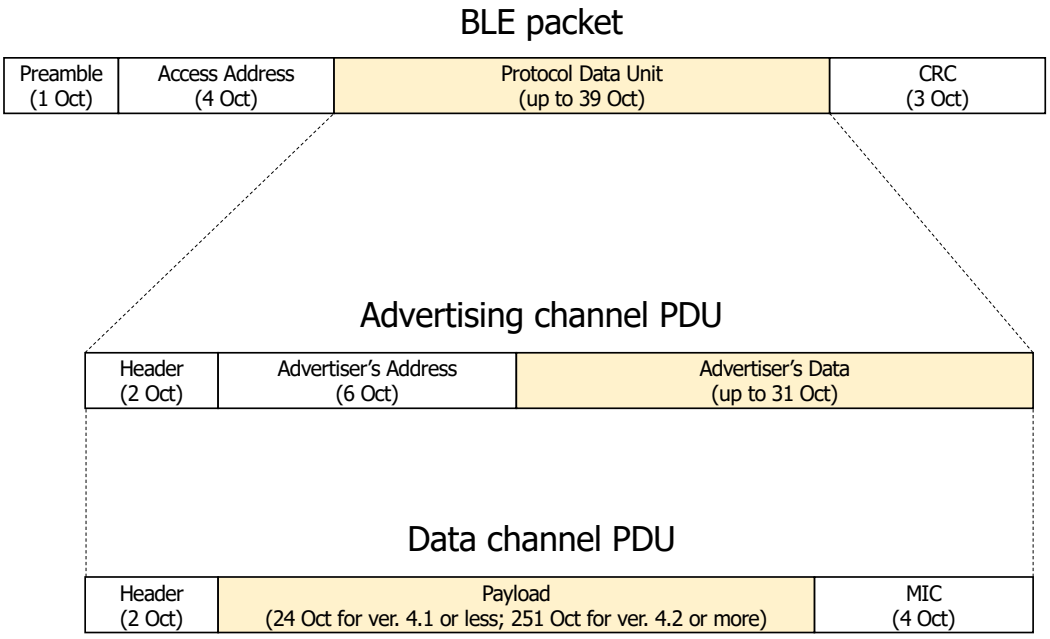


Fig. 2. This figure displays the format of BLE packets, including advertising packets transmitted uni-directionally by a broadcaster and data packets exchanged bidirectionally between central and peripheral devices. It is significant to note that the format of payload in iBeacon differs from that in Eddystone.

In the BLE protocol, specific UUIDs must be designated to access the desired value of a characteristic. If the central device connected to a peripheral device is unaware of the services and characteristics available on that peripheral, it can discover the necessary UUIDs by sending requests for service and characteristic discovery to the peripheral. Among the three access patterns (Read, Write, and Notify), Write is used by the central to write a value to the peripheral, while Read and Notify are used to read a value from the peripheral. The main difference between Read and Notify is that Read retrieves the value from the peripheral only once, while Notify retrieves the value repeatedly at intervals determined by the peripheral (a common use case for Notify is continuous reading of a value from a temperature sensor). The value size is limited to 20 bytes for Write and 22 bytes for Read, as determined by the maximum payload of a BLE packet. See Figure 2 for illustration.

4 How to Make BLE Beacons Stateful

Building a location-aware service with BLE beacons that support iBeacon/Eddystone protocols is cost-effective. However, these services are limited because they only

Table 1. Specification of peripheral devices.

	Pixel 3 XL	Raspberry Pi 4
CPU	Snapdragon 845, Kryo 385 Gold x4 & Kryo 385 Silver x4 8-core, 2.8 GHz	BCM2711, 4-core Cortex-A72 (ARM v8) 64-bit SoC
Memory	4GB	4GB
Bluetooth	5.0	5.0

transmit advertising packets with pre-stored values. A more convenient solution would be to transmit values that change depending on the environment, such as the number of visitors and congestion levels. To do this, we need to make the protocols *stateful*.

4.1 Three Typical Implementations

This section considers the following three methods for creating stateful functionality. The first method involves preparing a variable for the state in the beacon and updating it using Read and Write commands from an external device connected to the beacon. This method follows the standard BLE protocol and allows for inexpensive devices to be used as beacons, but it has several drawbacks such as long latency, instability as the number of user devices increases, and increased costs for guaranteed state updates.

The second method involves each beacon updating its internal state based on specific external events. Examples of such events include connections from user devices and signals from nearby sensors. This method requires a single board computer or microcontroller as the beacon device and a program to handle state updates such as Test-and-Set. Although this method is highly flexible, it loses compatibility with the BLE standard and may require lightweight protocols such as MQTT to communicate with sensors and servers.

The third method involves using an external server to store the environment's state. In the case of Physical Web [17], a relay server is placed between user devices and web servers to relay messages from user devices to the corresponding web servers. The status of the environment is recorded and updated on the relay server.

4.2 Comparison of the Performance

This subsection compares the response time of three implementations of stateful BLE beacon. More specifically, we measure the time it took to connect to the peripheral beacon (Experiment 1), access the state of the beacon from the connected central device (Experiment 2), and communicate with the external server (Experiment 3). The total response time experienced by the users of the stateful BLE

Table 2. Result of Experiment 1 (over 100 trials).

		Android	Raspberry Pi
state change	avg.	979.05 ms	631.84 ms
	max	1513 ms	1133 ms
	min	737 ms	391 ms
discover service	avg.	1747.63 ms	1350.65 ms
	max	2283 ms	1839 ms
	min	1445 ms	1073 ms

Table 3. Result of Experiment 2 (over 100 trials).

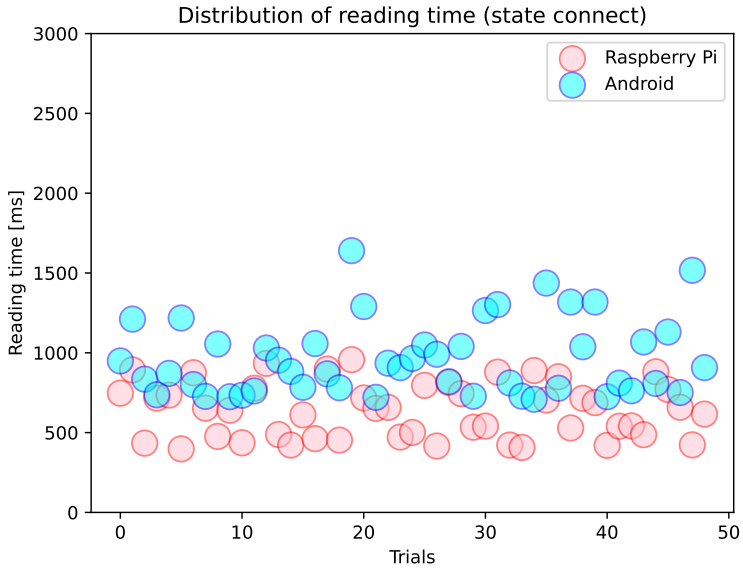
	Android	Raspberry Pi
avg.	114.31 ms	109.2 ms
max	284 ms	184 ms
min	51 ms	50 ms

service is the sum of these values. In the experiments, we used an Android smartphone (Pixel 3 XL) as the central device, a Raspberry Pi 4 and another Pixel 3 XL as peripheral beacons, and Heroku as the external server. Table 1 summarizes the specifications of each device, which are all compatible with BLE 5.0.

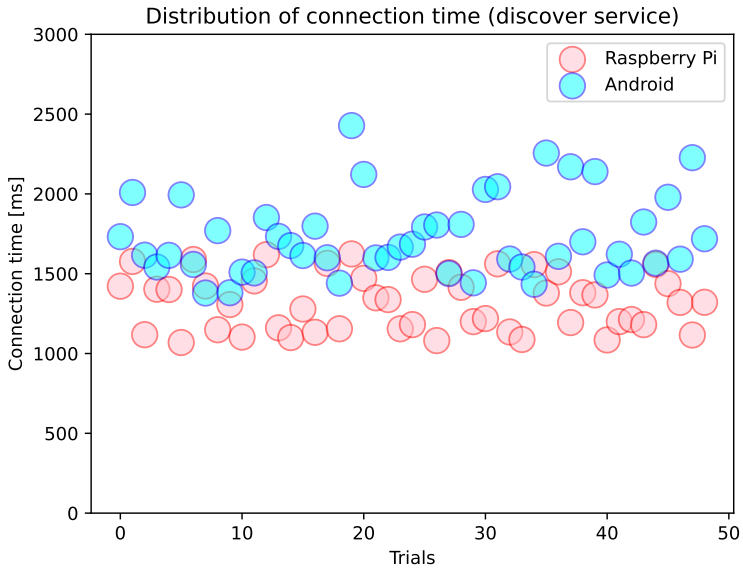
Experiment 1: Connection time At first, we evaluate the connection time to a peripheral beacon that is placed 1 meter away from the central device. The advertising interval of the peripheral is fixed at 130 ms, and we measure the connection time by calling the currentTimeMillis() method 100 times on the central device. We define two different end times for the connection process: the "state change," which is the time when the connection state becomes connected and the "discover service," which is the time when the central device identifies a service in the beacon with a designated UUID. The peripheral beacon alternates between advertising and scanning for connection requests from surrounding central devices. As the advertising interval is fixed at 130 ms, the average time taken to establish a connection is 65 ms. The results, shown in Table 2 and Figure 3, indicate that the Raspberry Pi is a more suitable device for a peripheral beacon, as it have a quicker and more stable connection time compared to the Android smartphone[2].

Experiment 2: Time for accessing the state We then evaluate the time it takes for the central to access the state of a connected peripheral beacon in the same environment as Experiment 1. The results, shown in Table 3, indicate that

[2] It is important to note that the connection time can be affected by both the scan window size of the central device and the advertising interval of the peripheral. To further reduce the connection time, we need to consider the trade-off between power consumption and scan window size, and aim to increase the scan window size.



(a) state change.



(b) discover service.

Fig. 3. The distribution of connection times over 50 trials.

Table 4. Communication time with the external server (over 50 trials).

	Method 2	Method 3
avg.	875.28 ms	886.33 ms
max	1110 ms	1326 ms
min	772 ms	774 ms

while the minimum access time is not significantly affected by the type of peripheral device, there is a difference of 1.5 times in the maximum access time, the range between the first and third quartiles, and the average access time. This difference can be attributed to the performance of the devices, as both support BLE 5.0. However, the actual data transmission rate can vary depending on the data size being transmitted. This means that we could potentially reduce the access time by optimizing the data format and packet length.

In conclusion, on average, the central can connect to a peripheral within 1.5 seconds and once connected, access the state in 100 ms. The use of single-board computers like Raspberry Pi is a better choice for peripherals over Android smartphones in terms of response time. Note that the first method requires accessing the state at least twice, as it issues a Write instruction after reading the state.

Experiment 3 Finally, we evaluate the communication time between the central and an external server. To do this, we measure the time it takes for the central to send a JSON file to the external server and receive the returned JSON file. We use the currentTimeMillis() method on the central to get these times. We also separately measure the database access time on the external server using the microtime() function. The results are summarized in Table 4. As expected, the third method takes longer to communicate with the external server compared to the second method, with an average time of 900 ms and an additional 20 ms for database access. This is still a bottleneck for the third method, with a communication time of 770 ms even in the best case scenario. In contrast, the first and second methods do not require communication with an external server as the packet returned from the beacon can include a concrete state.

5 First Use Case: Sharing Store Congestion

5.1 Assumed Scenario

Suppose that in a shopping mall, beacons are installed in each store to monitor store congestion, and beacons at the entrance provide visitors with the information. The beacons are split into two functions, one to gather information and the other to transmit it, and these functions are run in a single program file. The Obniz Board is used in this implementation, so the data collected by the beacons is shared in

the cloud. However, an edge server, which covers a smaller area, could also be used in the future to reduce costs.

The flow of information in this system is as follows: User devices send out packets as a peripheral, and the sensor beacons (A) scan the number of nearby user devices. Beacon A then shares the results with the broadcaster beacon (B), which broadcasts the information to its surroundings. Users in the vicinity receive the information from Beacon B. The details of how beacons function as sensors and broadcasters are described in the following subsections.

5.2 Sensing Functionality

To ensure accurate detection, only devices related to the service have a uniform device name, such as "iPhone_XXX". Beacons can distinguish between registered and unregistered devices by checking the names in the advertisement packets they scan. The sensor beacon retrieves the names of the sending devices from the scanned packets and temporarily stores them. After the scan is completed, it removes duplicates and calculates the estimated number of people. This information is stored in an array and shared with the broadcaster beacon (B). The scan is then repeated.

In addition to the current number of people near the beacon, the array also keeps track of the number of scans and the average level of congestion, which can be used in the process of congestion prediction. This information is continually updated, with the name of the sensing beacon serving as the key.

5.3 Broadcasting Functionality

Beacon B receives the updated information from Beacon A and broadcasts it to the surrounding area. In this implementation, in addition to the congestion information for each beacon, the average number of people per location is also included in the characteristic, so that users connected to the beacon can access this detailed information individually.

In each advertisement packet, only the values in the shared array are written, and the congestion level for each beacon location is represented by a numerical value. To properly match the values of the fields with the beacons, the correspondence must be shared between the user device and the beacon beforehand. Although it is possible to add fields to the advertisement packet to represent the corresponding beacon, it would increase the payload, so the trade-off must be carefully considered based on the application.

Note that while Beacon B broadcasts at regular intervals, to reflect the updated values in the shared array from Beacon A in the advertisements, it is necessary to stop sending packets, check the contents of the shared array, and reconfigure the characteristics of the advertisement packet to be updated one.

6 Second Use Case: Stamp Rally App

6.1 Assumed Scenario

Imagine a scenario where smartphone users visit various locations equipped with BLE beacons in a specific order to collect stamps. In this system, the smartphone user writes their location and attributes to the BLE beacon and receives a hint about their next destination, which is calculated based on their input information. The process flow between the beacon and the user device is as follows:

1. The user device connects to the beacon and writes the necessary information.
2. The beacon performs a calculation based on the written information and sets the result in the characteristic.
3. The user reads the calculation result from the characteristic of the beacon.

The next subsection explains how to implement beacons with these functions.

6.2 Implementation

The payload of a packet in this implementation is a sequence of bytes that can either be variable-length data or fixed-length data. When a beacon receives a packet from a user, it checks the label at the beginning of the payload to determine the type of data. If the label indicates variable-length data, the beacon reads the next byte to find out the length of the data and then reads the value accordingly. For instance, in the stamp rally app, the packet from the user will contain labels for the user's previous destinations, and the beacon can use this information along with the list of all installed beacons to determine the user's next destination by finding a set of unvisited beacons. If the label shows that the data is fixed-length data, the beacon reads the next sequence of bytes and stores the value. In this implementation, the beacon maintains information such as the total number of visits and a breakdown by gender, so these values are updated based on the received data. However, since a single packet has a limited capacity, it might be necessary to divide the data into multiple writes if it exceeds that capacity.

7 Evaluation

In this section, we evaluate two use cases described in the previous section. Firstly, we assess the time it takes for the BLE beacon to locate the target device in a scan, by considering the distance to the target, and examine the scalability of the proposed system. Secondly, we determine the optimal advertising period for a user device to send packets by analyzing the miss rate of the central during the scan.

The Obniz Board, a microcomputer board produced by Obniz, Inc., was used as the beacon and the user device in the experiment. It can connect to Wi-Fi

Table 5. The time taken by Central to find the target device.

	avg.	max	min
1m	2156.15 ms	7457 ms	592 ms
5m	2161.89 ms	8847 ms	961 ms
10m	7700.29 ms	25617 ms	1105 ms

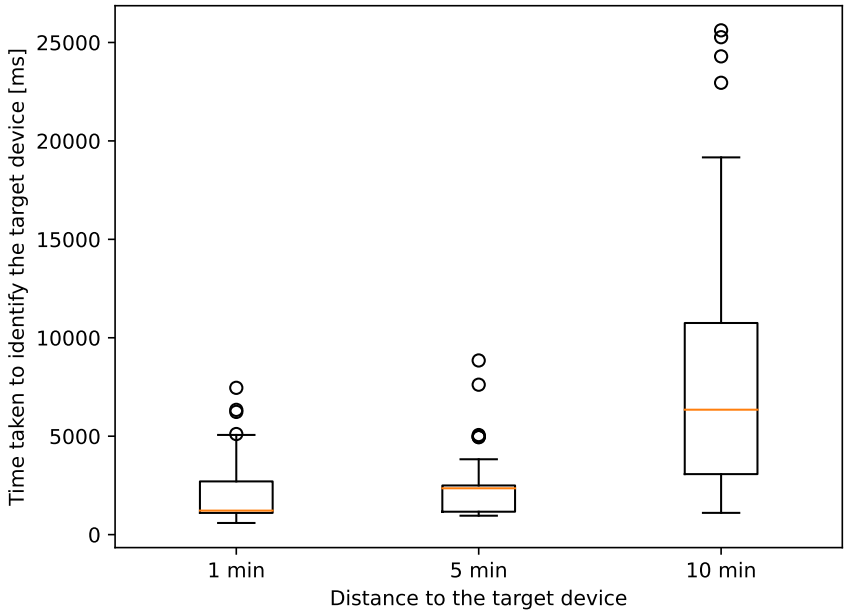


Fig. 4. The time taken by the central to find the target device.

and control devices and connected sensors through the cloud. The program for the experiments was implemented using HTML, JavaScript and the Obniz SDK (obniz.js). The Obniz Board supports Bluetooth v4.2 and the advertisement packet transmission interval was fixed at 1.28 seconds. The experiments were carried out in an indoor environment that resembles a typical apartment complex, where BLE devices and Wi-Fi signals are prevalent. All the Obniz boards were connected to the same Wi-Fi access point.

7.1 Effect of Distance between Devices on Scan Time

The aim of the first experiment is to determine the time it takes for the central (using an Obniz board with experimental programs) to scan for and locate a peripheral, and to find out the size of the area in which the central can reliably detect the peripheral. The experiment will indicate: in the first use case, how far away the installed beacon can locate a user device with certainty; and in the second use case, how close the user needs to be to the beacon to access the service.

Table 6. The number of successful scans out of 25 trials (central and peripheral were kept 1 meter apart).

advertisement period	Number of successes
3 sec	20
5 sec	24
10 sec	25

In the experiment, the peripheral constantly emits packets and the central scans the surrounding area, knowing the name of the target peripheral. We measured the time from the start of the scan to when the central found the advertisement packet of the target device. The time was measured using the performance.now() method of the Performance API. The distance between the central and the target device was varied as 1, 5 and 10 meters.

The results are shown in Figure 6 and the statistics derived from them are listed in Table 5. In the case of 1 meter and 5 meter distances, there are differences in the minimum value and range of the first and second quartiles for the 1 meter scenario. However, there was no significant difference in the mean value. At a communication distance of 10 meters, the minimum value is not significantly different from the other scenarios, but the range and average value of each quartile increases and the time to discovery becomes unstable due to the variation in time. This indicates that the communication between two devices becomes more challenging as the distance between them increases.

In summary, the results indicate that in the first use case, peripherals can be found stably within a 5 meter radius of the central. Additionally, the more peripherals found in a short scan time, the more devices are present in the central's vicinity. However, the time to find all objects increases as the number of peripherals grows, so longer scan times are necessary for applications that require high detection accuracy. The second use case also shows stable communication when the user is within 5 meters of the beacon's location.

7.2 Impact of Advertising Period to the Success Rate of Scans

In the second experiment, we examine how the length of the advertisement packet transmission period affects the central's miss rate from a power-saving perspective. In the experiment, the central and peripheral were kept 1 meter apart and we recorded the number of successful scans out of 25 trials, where BLE scans search for peripherals in the vicinity and a successful scan occurs when an advertisement from a specific peripheral is detected. We tested three different advertisement periods of 3, 5, and 10 seconds and the results, displayed in Table 6, indicate that at a distance of 1 meter, having the peripheral transmit for more than 5 seconds greatly reduces the chance of being missed by the central.

8 Concluding Remarks

In this paper, we introduce the concept of states for BLE beacons, which were previously used for transmitting a fixed value to nearby devices through advertising packets. To demonstrate this functionality, we implement three different methods using commercially available devices and assess their performance. Next, we explore the design of stateful BLE beacons and implement two use cases using the Obniz board. Our evaluation shows that the surrounding BLE devices were accurately detected under normal usage conditions with a distance of about 3 meters, and that phone identifiers such as names were correctly obtained if the phone continued to advertise for more than 3 seconds at 1.28 second intervals.

Future concerns encompass the need for a social implementation of the proposed framework and the conduction of a large-scale simulation to enhance the reflection of real-world usage. It is also imperative to experimentally evaluate the extent of the increase in power consumption resulting from the introduction of the concept of state into the beacon. Moreover, it is essential to establish a mechanism for safeguarding the accumulated data and preventing attacks by third parties.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. D. Baldie. System and method for providing a bluetooth low energy mobile payment system. 2016, US Patent App. 14/469, 230.
2. W. Bronzi, R. Frank, G. Castignani, T. Engel. Bluetooth low energy performance and robustness analysis for inter-vehicular communications. *Ad Hoc Netw.*, 37: 76–86 (2016).
3. H. Chena, S. H. Chab, T. W. Kim. A framework for group activity detection and recognition using smartphone sensors and beacons. *Building and Environment*, 158: 205–216 (2019).
4. D. Cianciulli, G. Canfora, E. Zimeo, Beacon-based context-aware architecture for crowd sensing public transportation scheduling and user habits, in *Proc. the Int'l Workshop on Smart Cities Systems Engineering*, 2017.
5. D. De, P. Bharti, S.K. Das, S. Chellappan. Multimodal wearable sensing for fine-grained activity recognition in healthcare. *IEEE Internet Comput.*, 19 (5): 26–35 (2015).
6. R. Faragher, R. Harle. Location fingerprinting with bluetooth low energy beacons. *IEEE J. Sel. Areas Commun.*, 33 (11): 2418–2428 (2015).
7. M.S. Gast. *Building Applications with iBeacon: Proximity and Location Services with Bluetooth Low Energy*, O'Reilly Media, Inc., 2014.
8. D. Giovanelli, B. Milosevic, E. Farella. Bluetooth low energy for data streaming: Application-level analysis and recommendation in *Proc. of the 6th IEEE Int'l Workshop on Advances in Sensors and Interfaces (IWASI)*, IEEE, 2015, pp. 216–221.
9. M. Girolami, F. Mavilia, F. Delmastro. Sensing social interactions through BLE beacons and commercial mobile devices. *Pervasive and Mobile Computing*, 67: 101198 (2020).
10. M. Girolami, F. Mavilia, F. Delmastro. A bluetooth low energy dataset for the analysis of social interactions with commercial devices. *Data in Brief*, 32: 106102 (2020).

11. G. Grekousisa, Y. Liu. Digital contact tracing, community uptake, and proximity awareness technology to fight COVID-19: a systematic review. *Sustainable Cities and Society*, 71: 102995 (2021).
12. iBeacon, Apple Developer. `https://developer.apple.com/ibeacon/`
13. J.-E. Kim, M. Bessho, S. Kobayashi, N. Koshizuka, K. Sakamura. Navigating visually impaired travelers in a large train station using smartphone and bluetooth low energy. in *Proc. of the 31st Annual ACM Symp. on Applied Computing*, ACM, 2016, pp. 604–611.
14. K. Kobayashi and S. Fujita. An Enhancement of Physical Web with Stateful BLE Beacons. in *Proc. of the 9th Int'l Symp. on Computing and Networking (CANDAR Workshops)*, 2021, pp. 21–27.
15. X.-Y. Lin, T.-W. Ho, C.-C. Fang, Z.-S. Yen, B.-J. Yang, F. Lai,. A mobile indoor positioning system based on ibeacon technology. in *Proc. the 37th Annual Int'l Conf. on Engineering in Medicine and Biology Society (EMBC)*, IEEE, 2015, pp. 4970–4973.
16. M. Meli, O. Rion. Streaming speech and music using bluetooth low energy. in *Proc. of the Embedded World Conference*, Nuremberg, February, WEKA Fachmedien, 2015, pp. 24–26.
17. D. Namiot and M. Sneps-Sneppe. The physical web in smart cities. In *Proc. of the 2015 Advances in Wireless and Optical Communications (RTUWO)*, IEEE, 2015, pp. 46–49.
18. G. Rajagopal, V.M. Lodd, A. Vignesh, R. Rajesh, V. Vijayaraghavan. Low cost cloud based intelligent farm automation system using bluetooth low energy. in *Proc. of the IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, IEEE, 2014, pp. 127–132.
19. C. Shao, S. Nirjon, J.-M. Frahm. Years-long binary image broadcast using bluetooth low energy beacons. in *Proc. of the Int'l Conf. on Distributed Computing in Sensor Systems (DCOSS)*, IEEE, 2016, pp. 225–232.
20. T. Takahashi, Y. Shibata, S. Imata, N. Suzuki. Evaluation of wake-on-demand accurate activation and in-vehicle wireless connection control,. in *Proc. of the10th Int'l Conf. on Broadband and Wireless Computing, Communication and Applications (BWCCA)*, IEEE, 2015, pp. 143–149.
21. M. Todasco. Systems and methods for completion of item delivery and transactions using a mobile beacon. 2015, US Patent App. 14/154,414.
22. H. Vahdat-Nejad, E. Asani, Z. Mahmoodian, M. H. Mohseni. Context-aware computing for mobile crowd sensing: A survey. *Future Generation Computer Systems*, 99: 321–332 (2019).
23. M. Vochin, A. Vulpe, G. Suciu, L. Boicescu. Intelligent displaying and alerting system based on an integrated communications infrastructure and low-power technology. in *Proc. of the World Conference on Information Systems and Technologies*, Springer, 2017, pp. 135–141.
24. J. Yang, C. Poellabauer, P. Mitra. Using bluetooth low energy for dynamic information-sharing in vehicle-to-vehicle communication. *SAE Int. J. Passeng.Cars-Electron. Electr. Syst.* 10 (2017-01-1650): 240–247 (2017).
25. J. Yang, C. Poellabauer, P. Mitra, C. Neubecker. Beyond beaconing: Emerging applications and challenges of BLE. *Ad Hoc Networks*, 97: 102015 (2020).
26. K. Židek, J. Pitel. Smart 3d pointing device based on MEMS sensor and bluetooth low energy. in *Proc. of the IEEE Symp. on Computational Intelligence in Control and Automation (CICA)*, IEEE, 2013, pp. 207–211.