Managing IoT at the Edge: The Case for BLE Beacons

Michael Haus
Technical University of Munich

Aaron Yi Ding
Technical University of Munich

Jörg Ott
Technical University of Munich

ABSTRACT

Managing IoT devices in urban areas is becoming crucial because the majority of people living in cities and the number of deployed IoT devices are steadily increasing. In this paper we present iConfig, an edge-driven platform dedicated to manage IoT devices in smart cities. The goal is to address three major issues in current IoT management: registration, configuration, and maintenance. The core of iConfig is its programmable edge module, which can be deployed across smartphones, wearables, and smart boards to configure and interact with physically proximate IoT devices. Through testbed experiments and usability studies, we reveal the hardship and hidden pitfalls in managing IoT devices, especially for low budget Bluetooth Low Energy (BLE) beacons. Our system evaluation shows that iConfig can effectively address the aforementioned IoT management challenges by harnessing the mobile and edge cooperation. To inspire community contributions, we further present concrete use cases to illustrate how iConfig can reduce operational cost and facilitate future edge-centric IoT research.

1 INTRODUCTION

54 % of the world's population lives in urban areas, and by 2050, it could be 66 % [1]. Moreover, the number of deployed Internet of Things (IoT) devices is steadily increasing and projected to reach approximately 50 billions in 2020 [2]. Managing urban areas and its applications is hence becoming important. Urban IoTs support the smart city concept [3] which integrates traditional and modern information and communication technology (ICT) for a unified and simple access to services for the city administration and the residents [4, 5]. The aim is an enhanced use of public resources, improving quality of services for citizens while reducing operational costs of public administration [4]. The smart city environments are based on a multitude of devices, such as smartphones, sensors, embedded systems, smart meters [3]. Each of these devices have their own purpose, only together they are able to satisfy all service requirements of smart cities. For instance, IoT boards serve as local gateways, collecting sensor data, and provide backend connectivity, whereas standalone Bluetooth Low Energy (BLE) beacons are cheap and simple to attach to many objects serving mainly for indoor localization and proximity detection of devices.

In spite of a growing demand for IoT management, there is still a lack of tools to seamlessly manage large scale IoT deployments in which ad hoc management is becoming untenable. We need an up-to-date overview of all distributed devices during different phases of their life cycle, including installation, registration, user customization, and device control. Moreover, the IoT configuration framework should be vendor independent and support a diverse range of devices [6]. We identified three main challenges for IoT management. First, IoT devices are installed at various locations, which requires a unified management process and is difficult to handle in large scale deployments. Second, there is no well-defined IoT management procedure that covers all necessary operations corresponding to each phase of the life cycle for IoT devices. This causes management overhead and information fragmentation. Third, manual efforts needed to configure those distributed IoT infrastructures are time consuming and error prone, which further increase operational cost [8].

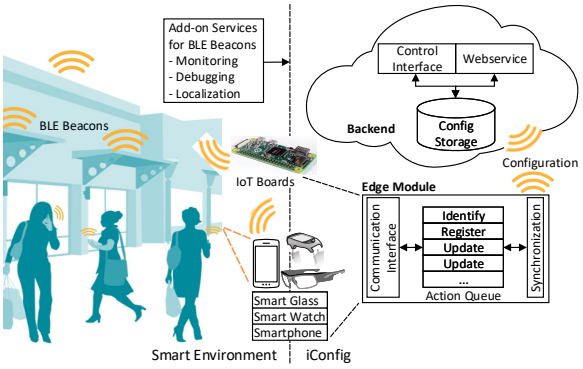


Figure 1: iConfig in the context of smart environments

To tackle these challenges, we propose iConfig, an edge-driven platform that takes care of all installed IoT devices. The framework covers the entire device life cycle: registration, configuration management, monitoring, and debugging of IoT devices. Moreover, our design aims to control the full spectrum of IoT devices, from high end IoT boards to low budget BLE beacons. iConfig enables automated edge device management and hence minimizes operational cost. Fig. 1 shows iConfig in the context of smart environments taking advantage of programmable devices to run edge modules on smart glasses or IoT boards. This allows iConfig to connect various IoT devices to its management backend, which enforces a unified configuration procedure. As an example, we successfully tested iConfig on Android smartphone and

smart glass [7]. In particular, the iConfig edge modules are dedicated for users interested in managing and interacting with IoT environments.

The design principles of iConfig: 1) automatic configuration of IoT devices to avoid misconfigurations which become one of the dominant causes of system failures [8], 2) easy to use frontend, 3) device orchestration via global view, and 4) serving as a platform for developers and researchers to enable add-on services.

The key contributions are summarized as follows:

- We analyzed and identified key properties of IoT device management that must be attained to manage large scale deployments in smart cities.
- We designed and implemented iConfig, an edge-driven platform dedicated for IoT device management. We demonstrate the efficacy of iConfig via prototype implementations, which target at BLE beacons without backend connectivity.
- Our usability study and testbed experiments further uncover the hidden aspects in IoT management that are important and deserve future research.

The rest of the paper is structured as follows. Section 2 defines requirements for IoT device management. Section 3 introduces BLE beacons and Section 4 highlights different use cases of iConfig. In Section 5 we present the system architecture, workflow, and implementation details of iConfig. The evaluation in Section 6 consists of performance tests regarding memory usage and system scalability. In addition, we conducted a user study to show differences between manual and automatic device configuration. Section 7 presents related work and Section 8 provides a discussion about user interactions with their surrounding devices. We conclude and outline future work in Section 9.

2 REQUIREMENTS FOR IOT DEVICE MANAGEMENT

A major challenge for IoT is the management and configuration of pervasive deployments, especially for IoT devices without backend connectivity. We identified three stages of IoT device management.

The first stage covers device deployment and registration, in which the device is identified and initially configured with default settings. Afterwards, the device is registered at a backend service combined with location information. This manual first step is done only once per device.

Second, automated configuration of device parameters is the fundamental service for IoT device management. The configuration parameters depend on the specific device. Thereby, we classify IoT devices into two different types: standalone and connectivity devices. Standalone devices are only equipped with limited short-range transmission techniques, such as near field communication (NFC), ultra-wideband (UWB), ZigBee, and/or Bluetooth. These IoT devices (e.g., BLE beacons) have no backend link and require additional edge modules for device management. The edge modules can run on platforms which support short-range communication and provide a backend link. In contrast, connectivity devices (e.g., smarthome gateways) have a connection to the backend and offer easier device management.

Third and final stage of IoT device management refers to the centralized backend service which receives device registrations together with configuration data. This enables multiple add-on services: 1) monitoring of distributed IoT devices via global view, including configuration status or localization of broken devices for replacement, 2) debugging of IoT devices via collected maintenance data, such as uptime or battery voltage to identify malfunction devices, 3) software and firmware updates distributed via central backend service which ensures up-to-date software versions and improves IoT security, and 4) parameter updates for a set of devices. Thereby, the device selection can be based on different criteria, such as close proximity among IoT devices.

Our case study targets at BLE beacons, which represent one of the most challenging classes of IoT devices due to missing backend connectivity. Furthermore, the number of deployed beacons will be much higher than the number of IoT boards. Thus, it's important to have a scalable framework which covers the defined requirements for IoT device management.

3 BLE BEACONS

The battery powered BLE beacons are small-size wireless devices that transmit a short-range BLE signal to mobile computing devices (e.g., smartphones) [9]. Via BLE, users' devices are notified of the beacon proximity by receiving signals which contain contextual information, typically about indoor surroundings and its contents. Thus, the receiving end is able to perform location aware actions, such as accessing specific URLs for marketing purposes [9]. For example, the Los Angeles International Airport uses Bluetooth beacons to track and dispatch wheelchairs for passengers in need of assistance [10]. In another scenario, universities use beacons to help students navigate through the library, guiding them to resources, study spaces, and services in the library [11].

Our beacons [12] are capable to send three different BLE message formats and can be locked via a password. The primary beacon standards are Eddystone and iBeacon. Eddystone is a protocol specification that defines a BLE message format for beacons [13]. It describes different frame types to transmit device identifier, URL, or telemetry data containing battery voltage, beacon temperature, count of advertised packets, and uptime. In contrast, the iBeacon standard

[14] transmits only a device identifier. Besides that, the beacon vendor added an own proprietary BLE message format named sBeacon to transmit another fixed device identifier imprinted on the beacon.

4 ICONFIG USE CASES

The ability of iConfig to programmatically adjust device parameters facilitates different use cases. For example, we used iConfig to set up an automated testbed for research projects to adapt BLE parameters, such as transmission power or packet advertisement rate. The goal was to measure the interference between BLE and Wi-Fi because both wireless technologies work on the same frequency of 2.4 GHz. Moreover, iConfig supports debugging and monitoring of BLE beacons by collecting maintenance data, such as uptime and battery voltage of each device. The long term benefits of iConfig come from add-on services based on the programmable interface of iConfig. In the context of a smart city management for dense, co-deployed IoT devices, the energy-aware configuration of devices is important. For instance, the developer can use iConfig to add a module to the iConfig backend to turn off devices at specific times.

5 ICONFIG DEVICE MANAGEMENT

The goal of iConfig is to manage various IoT devices by utilizing programmable edge platforms. In our case study, we focus on the management of BLE beacons as low budget IoT devices. This section presents the system architecture of iConfig and the corresponding working flow for IoT device management. In addition, we provide implementation details regarding software libraries and BLE communication.

5.1 System Architecture and Key Features

The iConfig system architecture, as illustrated in Fig. 1, consists of two major modules: mobile edge module and backend module. The framework is able to identify, register, and update IoT devices (in our case BLE beacons). Supported by our speech recognition, a user wearing an iConfig-enabled smart glass can discover, register, and configure BLE beacons while walking around. The edge module is intended to run on mobile devices (e.g., smart glasses, smartphones, tablets) and on static devices like IoT boards. The backend module of iConfig runs at a centralized infrastructure, such as a local server or in the cloud.

The iConfig edge module uses a central action queue for all beacon operations to synchronize user actions (identify and register) with automated updates of beacon settings. Thereby, user interactions are prioritized over automatic beacon updates. Each device action occurs via threads to immediately start the next action from the central operation queue. During registration and update of BLE beacons, the edge module

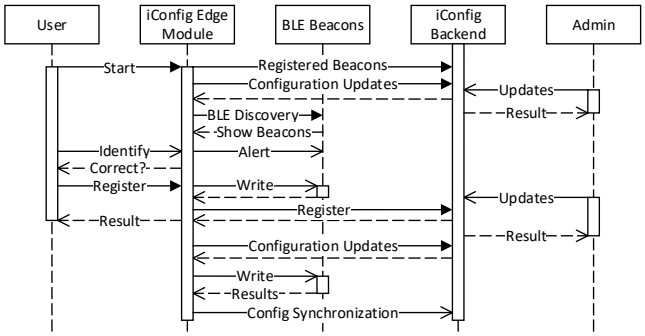


Figure 2: iConfig workflow

collects maintenance data which enables monitoring of beacon health and detection of broken beacons. To update BLE beacons, we aggregate received signal strength indication (RSSI) for each beacon and update beacons according to descending RSSI sum, which means nearest beacons first. Besides that, the iConfig edge module works in two different modes: offline and online depending on connectivity to the backend. In offline mode, beacon configurations are stored and loaded from local storage for later synchronization to the backend.

The iConfig backend stores all device data at a central storage. We ensure that only beacons are updated where BLE parameters are actually changed. Furthermore, the iConfig backend provides a control interface including status about device update, functionality (e.g., broken BLE beacon), device health, and offers localization via indoor map and place image.

5.2 Workflow

The main task of the user is to register IoT devices (in our case BLE beacons). Therefore, the user holds a smartphone running the iConfig edge module and walks around to discover nearby BLE beacons. At the application start, the edge module automatically received registered beacons via iConfig backend and shows only unregistered beacons, when they are discovered by the user. Fig. 2 presents the iConfig workflow.

After discovering unregistered beacons, the user is able to identify one beacon at a time. The beacon shows a red light as feedback for device identification. When the beacon identification was successful, the user can register the beacon to the iConfig backend with additional information for device localization, such as nearest room number, picture of device place. Afterwards, during registration, the edge module will automatically configure the BLE beacon with a default configuration including password, iBeacon, and Eddystone identifier to ensure that the device is ready to use. Finally, the edge module will synchronize all configuration

data to the backend. The registration has to be done only once per beacon.

For advanced management, iConfig backend provides a control interface for the administrator, including a global view about installed BLE beacons. The administrator has technical knowledge and is responsible to manage the registered BLE beacons. Therefore, the administrator is able to adapt multiple device configurations at once. This is possible by linking devices to groups and groups to configurations.

The device update of a BLE beacon is independent of user actions and automatically triggered when two conditions are satisfied: 1) adapted BLE configuration from administrator via iConfig backend available, and 2) beacon is currently discovered by the user via iConfig edge module. The update process runs in the background of the iConfig edge module, only recognizable by blinking red lights as feedback of successful configuration.

5.3 Implementation Details

We implemented two prototypes of the iConfig edge module for different device types: smartphone and smart glass. We take advantage of speech recognition to enable hands-free device configuration. On the smartphone the speech recognition can be optionally activated, on the smart glass it is automatically activated to allow a convenient usage of the iConfig edge module.

For beacon configuration, the Eddystone standard provides a Bluetooth Gatt service for URL configuration. Another service allows the adaption of all Eddystone parameters. This service was not available for our beacons and does not cover all BLE parameters. We hence used the communication library provided by the beacon vendor [15], which uses a customized communication interface for beacon configuration. The communication library works asynchronously via callbacks to provide the result to set beacon parameters.

The iConfig backend is implemented in Python and provides a REST API and a control interface via cherrypy. The beacon device data can be stored in any database. In our case, we use MongoDB to store device configurations, one document per beacon. Moreover, we use an URL shortening service to overcome the limit of 17 bytes for the Eddystone URL field. For system security, the password of each beacon is stored encrypted in MongoDB via AES encryption. In addition, SSL secures the wireless network connection between iConfig edge and backend module. Thus, common attacks such as man-in-the-middle or sniffing are not possible.

6 EVALUATION

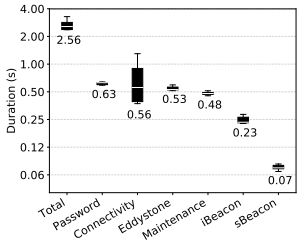
We analyzed the system performance of iConfig regarding memory usage and configuration scalability over multiple beacons. Moreover, we breakdown the configuration time for one beacon to show the duration of each configuration part. Additionally, we conducted a user study to highlight drawbacks by manually configure IoT devices.

6.1 System Performance

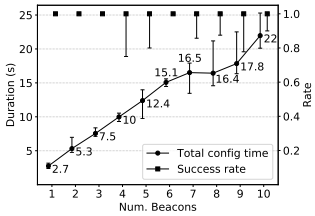
Regarding the memory usage of iConfig edge module, we measured a deployment on Android smartphone (OS 7.1.1) in offline and online mode during configuration of ten BLE beacons. In online mode, the edge module used 6.50 ± 0.98 MB similar to offline mode with a memory usage of 6.47 ± 0.96 MB. These results show that the memory footprint of the iConfig edge module is small enough to run on programmable IoT devices.

The next evaluation part of iConfig refers to the beacon configuration. As illustrated in Fig. 3 (a), we breakdown the configuration time for one beacon over 20 rounds. Thereby, we measured a total configuration time of 2.56 s shared over six different configuration phases. The connectivity and maintenance phase takes almost half of the configuration time (41 %). The connectivity phase consists of connect and disconnect time. It shows the largest time fluctuation due to interference among BLE beacons. Regarding device configuration, to set the password takes most of the time (25 %). To configure Eddystone packets, including telemetry, URL, and identifier sum up to 21 % of the configuration time. iBeacon and vendor specific sBeacon takes less time. Our results show the worst case, in which all configurable fields are adapted. Usually, the identifiers for Eddystone and iBeacon are set only once during device registration.

For scalability testbed and to evaluate iConfig in a dense deployment we placed ten beacons in a circle with a diameter of 1 m around the smartphone running iConfig edge module. Thereby, we evaluated ten cases (from one to ten beacons) each over 20 rounds. All BLE parameters for transmission power (5 dBm - 80 m) and advertisement rate (10 Hz - 10 times a second) were set to the maximum, which reflects the worst case setting in terms of interference among BLE beacons. Our evaluation yielded 18 unauthenticated connect errors (meaning BLE configuration is not possible) out of 1100 connect attempts, i.e., rate of 1.64 %. Fig. 3 (b) illustrates scalability results when configuring ten BLE beacons against ten cases each over 20 rounds. The configuration time shows a linear increase over all beacons, on average an increase of 2.2 s per beacon. In addition, we evaluated the success rate which describes whether configuration parameters were correctly set. A success rate of 100 % means that all parameters are correctly set to the predefined value. In our evaluation, the lowest success rate was 75 % during the configuration of four beacons. In most cases, iConfig achieved a success rate of 100 %. In our testbed with a dense deployment of BLE beacons and high interference among devices, iConfig is still able to achieve a reasonable configuration time per beacon with high success rates.



(a) Breakdown of configuration time for one beacon



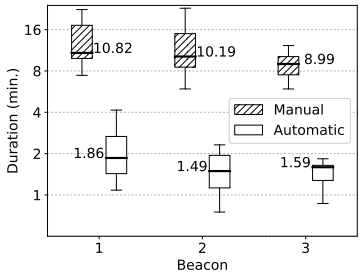
(b) Configuration time and success rate from one to ten beacons

Figure 3: iConfig system evaluation

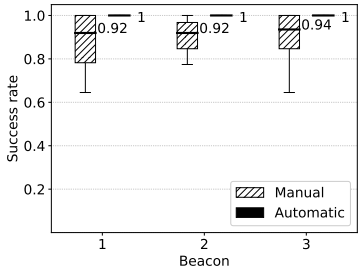
6.2 Usability Study

Our user study serves two purposes: 1) to reveal the gap between a manual and automatic configuration system, and 2) to test the application interface of the iConfig edge module. In total, ten persons participated in the study, all PhD students or Postdocs with a strong background in computer science. The user study consisted of a questionnaire and the configuration of BLE beacons. The participants manually configured three beacons with a predefined configuration using the vendor application [16]. In the next step, the same beacons were registered via iConfig, in which the default configuration was automatically written to each BLE beacon. For each beacon, we calculated the success rate by comparing predefined configuration with actual beacon settings. The configuration time for the vendor app was taken manually. Fig. 4 (a) shows the configuration time per beacon. The manual configuration took in average six times longer than the iConfig automatic configuration, which reflects a time saving of 83 %. The results show a slight decrease of configuration time when the user is more familiar with the configuration system. Fig. 4 (b) presents the success rate, how many parameters were correctly set at the beacon. In case of manual configuration, the lowest success rate over all beacons was 58 % and the median is around 92 %. In total, only 1/3 of all manual configurations were entirely correct. On the other hand, our iConfig framework achieved for all BLE configurations the success rate of 100 %. In general, this shows that the manual configuration of BLE beacons is time consuming and error prone due to type errors by users. Fig. 4 (c) presents the iConfig edge module to register BLE beacons and Fig. 4 (d) illustrates the iConfig control interface which is part of the backend module.

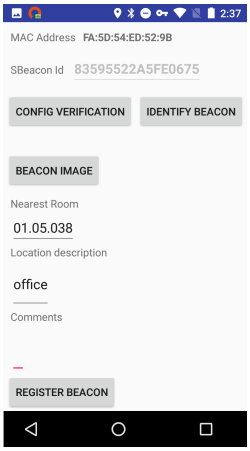
In the following questionnaire, most participants rated the manual configuration via the vendor app as difficult. On the other hand, the automatic configuration by iConfig were entirely rated as easy. The configuration process is faster and device registration requires less manual input by the user. Moreover, we asked the participants of the user study



(a) Config duration per beacon



(b) Config success rate per beacon



(c) iConfig edge module

ACTION	UPDATE STATUS	MAC ADDRESS	BEACON ID	NEAREST ROOM	MAINTENANCE
Edit	up-to-date	C9:ED:20:D2:6A:5F	8EC968106F2	01.05.038	PKT COUNT: 35155830 / VOLTAGE: 3.6000001 / UPTIME: 459853408
Edit	up-to-date	EB:26:C9:E4:DC:5B	C28C3BC4AE8	01.05.038	PKT COUNT: 24238715 / VOLTAGE: 3.6000001 / UPTIME: 459856770
Edit	up-to-date	D6:82:27:05:E4:0B	2F0446B54A6	01.05.038	PKT COUNT: 31240056 / VOLTAGE: 3.6000001 / UPTIME: 459853016

(d) iConfig control interface (reduced due to space limits)

Figure 4: Results of user study and screenshots of iConfig framework to administrate IoT devices

for useful features of the iConfig backend. The monitoring of beacon health and the localization of beacons achieved highest consent.

7 RELATED WORK

A majority of the related work are dedicated to services taking advantage of beacon deployments, while BLE beacons themselves are not considered. The work in [17] presented a system where an IoT hub is dynamically selected from a changing set of users' devices. The IoT hub is responsible for configuring a set of services running on proximate devices to ensure an efficient management of available resources. Harris *et al.* [18] used BLE-tagged products for inventory control. Their work included a detailed analysis of BLE regarding signal propagation, deployment, and protocol attributes. In our case, iConfig is dedicated for IoT device management including BLE beacons.

Indoor localization and proximity detection are major use cases for BLE beacons. Faragher *et al.* [19] found that BLE positioning systems provide a higher accuracy compared with Wi-Fi fingerprinting. The authors of [20] proved the feasibility of using BLE beacons for proximity detection in working places. The detailed analysis of BLE parameters, such as advertising interval and transmission power shows the impact of these settings on the proximity detection mechanism. Another use case is highlighted by Michalevsky *et al.* [21] using cryptographic secret handshakes over BLE protocol. Their proposal can enable private communication among nearby devices without central servers, which addresses a crucial concern for privacy in device-to-device (D2D) communication [22].

8 USER INTERACTIONS

A key observation from our user study and experiments is that the interaction among users, their smart gadgets and surrounding IoT devices via the conventional screen-keyboard setup is far from optimal. Especially for smart cities with a multitude of services empowered by IoT devices, the system interaction should be more natural and fluent. Even when users adopt iConfig on smartphones, the keyboard input is still hindering the user experience no matter how automated iConfig has made the entire configuration process. This is the main reason for our second prototype dedicated for wearables (e.g., smart glass). The combination of speech recognition and hands-free devices can enable a more integrated interaction during user movement and limit the distraction of user attention. iConfig is hence endeavored to enhance user experience and to streamline management of large scale IoT deployments, especially for low budget devices such as BLE beacons without backend connectivity.

9 CONCLUSION AND FUTURE WORK

We designed and implemented iConfig for IoT device management in smart cities. The goal is to streamline the management of large scale IoT deployments, especially for low budget devices such as BLE beacons. We have evaluated iConfig in two ways. In the usability study, we revealed the hardship introduced by existing mechanisms. Our results highlighted the time saving and higher success rates of iConfig owing to its automatic configuration with minimal user interaction. The system evaluation of iConfig further revealed a small memory footprint of the mobile edge module, making it suitable for various smart devices. We also evaluated the robustness and scalability of iConfig by configuring multiple BLE beacons in a testbed. The configuration time scales linearly with high success rates. Moreover, we discussed several use cases enabled by iConfig, such as dynamic adjustment of beacon parameters for IoT testbeds, and efficient detection of broken devices. For future work, we plan to enhance the mobile edge module. An immediate step is to port the current Android implementation to Linux environment, which can be deployed on multiple programmable IoT boards. This will ensure a more reliable and faster update process by using iConfig platform to unify the management of edge and IoT devices. Besides that, another extension is to improve synchronization redundancy when several iConfig edge modules are in offline mode and ready to update BLE beacons with latest device settings.

REFERENCES

[1] https://esa.un.org/unpd/wup/publications/files/wup2014-highlights.Pdf (visited on 05/29/2017)

[2] https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/ (visited on 05/29/2017)

[3] Schaffers et al. Smart Cities and the Future Internet: Towards Cooperation Frameworks for Open Innovation. In *Future Internet. Lecture Notes in Computer Science, Vol. 6656. Springer.* 2011. 431–446.

[4] Zanella et al. Internet of Things for Smart Cities. *IEEE Internet of Things Journal* 1, 1 (2014), 22–32.

[5] Zdraveski et al. ISO-Standardized Smart City Platform Architecture and Dashboard. *IEEE Pervasive Computing* 16, 2 (2017), 35—43.

[6] Dalipi et al. EC-IoT: An Easy Configuration Framework for Constrained IoT Devices. In *Proceedings of the IEEE 3rd WF-IoT.* 2016.

[7] http://madgaze.com/x5/ (visited on 05/29/2017)

[8] Xu et al. Systems Approaches to Tackling Configuration Errors. *ACM Computing Surveys* 47, 4 (2015), 1–41.

[9] Wang et al. Managing Large Scale, Ultra-Dense Beacon Deployments in Smart Campuses. In *Proceedings of IEEE INFOCOM WKSHPS* 2015.

[10] http://www.airport-world.com/news/general-news/5976-lax-uses-bluetooth-beacon-technology-to-improve-wheelchair-operations.html (visited on 05/29/2017)

[11] http://www.edtechmagazine.com/higher/article/2017/02/bluetooth-beacons-could-improve-student-experience-higher-ed-campuses (visited on 05/29/2017)

[12] http://bluvision.com/wp-content/uploads/2016/12/Specs-iBEEK1.6.pdf (visited on 05/29/2017)

[13] https://github.com/google/eddystone (visited on 05/29/2017)

[14] https://developer.apple.com/ibeacon/ (visited on 05/29/2017)

[15] http://developer.bluvision.com/developer/beeks-beacons-sdk/ (visited on 05/29/2017)

[16] https://play.google.com/store/apps/details?id=com.bluvision.beaconmaker (visited on 05/29/2017)

[17] Varshavskiy et al. MiHub: Wearable Management for IoT. In *Proceedings of the WearSys.* 2016. 1–6.

[18] Harris et al. Smart LaBLEs: Proximity, Autoconfiguration, and a Constant Supply of Gatorade(TM). In *Proceedings of the 1st IEEE/ACM SEC.* 2016. 142–154.

[19] Faragher et al. Location Fingerprinting With Bluetooth Low Energy Beacons. *IEEE Journal on Selected Areas in Communications* 33, 11 (2015), 2418–2428.

[20] Montanari et al. A Study of Bluetooth Low Energy Performance for Human Proximity Detection in the Workplace. In *Proceedings of the IEEE PerCom.* 2017. 1–10.

[21] Michalevsky et al. MASHaBLE: Mobile Applications of Secret Handshakes over Bluetooth LE. In *Proceedings of the 22nd MobiCom.* 2016.

[22] Haus et al. Security and Privacy in Device-to-Device (D2D) Communication: A Review. *IEEE Communications Surveys & Tutorials* 20, 2 (2017).