

© 2016 by Kirill Varshavskiy. All rights reserved.

DEVICE MANAGEMENT OF HETEROGENEOUS BLUETOOTH LOW ENERGY  
DEVICES

BY

KIRILL VARSHAVSKIY

THESIS

Submitted in partial fulfillment of the requirements  
for the degree of Master of Science in Computer Science  
in the Graduate College of the  
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Adviser:

Professor Robin Hillary Kravets

# Abstract

With the ever-growing adoption of smart, peripheral consumer devices, users enter a world where they can monitor their health and every day activities. As such, the advent of smart homes allows for the interconnectedness of all personal devices to be available on a unified platform. The efforts to unite all device management under a single banner has proven to be a difficult task, both in academia, and in consumer technologies. As mobility and limited power became the core of everyday computing, previous device management architectures have shown to be resource intensive and inapplicable to today's usage scenarios. This Master's Thesis presents a novel device architecture which enables efficient communication between devices, optimizes for system health, and utilizes all computing ability within its reach. Additionally, to aid in device intercommunication, the Master's Thesis also outlines a novel passive synchronization technique for peripheral devices which reduces overall energy consumption spent on scanning for other nodes.

*To Father, Mother, and Brother, the amazing trifecta.*

# Acknowledgments

This project and the entire masters journey would not be possible without the help of my adviser, Professor Robin Kravets. Her guidance and insight along the way has ignited the researcher passion in me. I would also like to thank Dr. Albert Fred Harris III for motivating me and my work and pushing me to greater heights despite the difficulties of life.

This research is funded in part by a Focused Research Grant from Google Inc.

# Table of Contents

<b>List of Figures . . . . .</b>	<b>vii</b>
<b>Chapter 1 Introduction . . . . .</b>	<b>1</b>
<b>Chapter 2 Bluetooth Low Energy Environment . . . . .</b>	<b>3</b>
2.1 Working out in the gym . . . . .	4
<b>Chapter 3 Past Work and Newfound Challenges . . . . .</b>	<b>5</b>
3.1 Device Management . . . . .	5
3.1.1 Personal Server . . . . .	5
3.1.2 Smartphone-centered Hubs . . . . .	5
3.2 Clock Synchronization with Wireless Nodes . . . . .	6
3.2.1 NTP . . . . .	7
3.2.2 RBS . . . . .	7
<b>Chapter 4 Device Management with MiHub . . . . .</b>	<b>9</b>
4.1 Components . . . . .	9
4.1.1 Device . . . . .	9
4.1.2 Sensors and Services . . . . .	10
4.1.3 Proximity Group . . . . .	10
4.1.4 MiHub . . . . .	11
4.1.5 Personal Cloud . . . . .	11
4.2 Features . . . . .	11
4.2.1 Leader Election . . . . .	11
4.2.2 Replication . . . . .	12
4.2.3 Migration . . . . .	13
4.2.4 Contextual Awareness . . . . .	13
<b>Chapter 5 Passive Synchronization . . . . .</b>	<b>14</b>
5.1 Selective Scanning . . . . .	14
5.2 Architecture . . . . .	15
5.2.1 Master . . . . .	15
5.2.2 Node . . . . .	15
5.2.3 Tier Interval Beaconing . . . . .	16
5.2.4 Configuration Period . . . . .	16
5.2.5 Selectivity . . . . .	16
<b>Chapter 6 Evaluation of System Configurations . . . . .</b>	<b>18</b>
6.1 Connected vs Passive . . . . .	18
6.2 Scanning Interval Window . . . . .	19
6.3 Sensor Replication and Migration . . . . .	20
6.4 Noisy Environment Baselines . . . . .	21

<b>Chapter 7</b>	<b>Prototypes and Trials . . . . .</b>	<b>23</b>
7.1	Smart Gym Demonstration . . . . .	23
7.2	MiHub TinyBLE Simulation . . . . .	24
<b>Chapter 8</b>	<b>Conclusion and Future Work . . . . .</b>	<b>25</b>
<b>Figures</b>	<b>. . . . .</b>	<b>26</b>
<b>References</b>	<b>. . . . .</b>	<b>40</b>

# List of Figures

1	RBS in action . . . . .	27
2	MiHub System Architecture . . . . .	28
3	Several beacons with various BIs . . . . .	29
4	Alignment of broadcast windows . . . . .	30
5	Energy consumption of various BI settings . . . . .	31
6	Aggregate current over one minute in passive and connected (active) mode . . . . .	32
7	Master's potential broadcasting window and computation for node's scanning interval . . . . .	33
8	Number of beacons received based and reconfiguration periods due to Scanning Intervals . . . . .	34
9	Energy draw based on various scanning Intervals . . . . .	35
10	Aggregate current over one minute . . . . .	36
11	Advertisement loss percentage with 1000ms and 500ms beacon period and 100, 50, 25, and 1 beacon . . . . .	37
12	Advertisement loss percentage with 1000ms and 500ms beacon period and 100, 50, 25, and 1 beacon . . . . .	38
13	Mini lat pull machine, free weight, Nexus 5 smartphone, LG G smartwatch, and two Estimote beacons, demo configuration at HotMobile 2015 . . . . .	39



# Chapter 1

## Introduction

Over the past several decades, mobile devices have taken a foothold in the consumer market. From more specialized devices such as Walkmans [31], gaming handhelds, and pedometers, the industry gave way to computing behemoths such as laptops and smartphones. However, as the number of specialized devices users carry around has decreased, the number of devices still used by an average US online adult has approached 4 in 2015 [12]. The mobility of devices has also paved the way for energy-efficient computing, allowing smaller form factors to produce self-proclaimed "smart" devices that function as connected peripherals, giving users more insight into their every day lives. As a result, about one in five US online adults use a fitness band and smart wearable device [12], digitizing users' activities, and making it more accessible to consume activity behavior, among other valuable data.

As mobile computing makes this transition into smaller form factors that are able to produce valuable data, there needs to be a way to efficiently collect and manage this information. Currently, the smartphone acts as a mobile hub for most peripherals, however, as these peripherals gain computing power, they have proven to function as stand-alone devices, such as the intention of some Android Wear smartwatches [18]. The smartphone, however, remains the anchor for all peripherals, even when it is not within reach. Take, for example, a scenario in which a user goes to the gym to exercise, where a fitness band, along with other peripheral sensors, may collect lots of valuable information while the smartphone rests in the user's locker. As more functionality is added to smaller devices, new device organizational hierarchies have to be created. Smaller devices, capable of processing complex information, could take on hub responsibilities and work in conjunction with smartphones to produce a dynamic resource management system.

Furthermore, modern consumer devices often come with the new Bluetooth Low Energy (BLE) [1] protocol, which prioritizes a lower cost of communication over robust communicational guarantees. As such, communication in a BLE network is a very costly commodity. Most inter-device communicational paradigms, especially ones that attempt to synchronize various devices, rely on robust communication. With BLE, there are new challenges that need to be overcome in order to create a communicational protocol that is both functional and energy efficient.

This Master's Thesis will outline a novel architecture that extends knowledge of previous Hub systems and applies them to the modern devices. Additionally, this system, known as a MiHub architecture, allows for intelligent device management, reduction of redundancy, and prolonging the overall health of the system. In order to aid in the lower cost of communication, this Thesis also outlines a technique to reduce energy consumption in the synchronization of BLE nodes.

## Chapter 2

# Bluetooth Low Energy Environment

Bluetooth Low Energy (BLE) [14] is abundantly used in modern Internet of Things (IoT) systems, alongside WiFi, Bluetooth, and Zigbee. BLE has proven to be more power efficient than the others, and is now the de-facto industry standard for peripherals. Even with the low energy protocol, however, there are certain modes that can drain more battery in the peripheral device. A BLE device can take on one of four roles: a peripheral, a server, a client, or a central. Typically, a central is the device that scans and initiates connections, often a phone or a sniffer of some kind. A peripheral is a device that constantly advertises its presence and can respond to connection requests. This Peripheral mode is the most power-efficient (depending on the beacon interval) and is the core functionality that can be used. Upon establishing a connection, the devices can communicate in a client-server way where one device reads off another device's information. However, the connected mode is known to take up more energy when communicating in the client/server paradigm. This behavior is evaluated in Chapter 6.

Additionally, smaller, battery powered BLE devices typically contain only one radio, which limits its functionality. This restriction forces the device to decide whether to broadcast or scan for others' beacons, but not both simultaneously. This puts a sizable strain on the effectiveness of various synchronization algorithms which depend on concurrent broadcasts such as Reference-Broadcast Synchronization (RBS). Furthermore, beaconing at a constant interval is much more power efficient than scanning for other devices. This is also shown in the evaluation in section 6.1 where various scanning windows are assessed for accuracy and analyzed in their power tradeoffs.

Finally, the BLE protocol includes a random 10ms jitter as part of every broadcast. As the protocol does not follow standard CDMA/CS practices to avoid broadcasting in a congested environment, it lessens a possibility of collision by adding a random offset of 0-10ms to the advertising time. Unfortunately, this jitter makes it harder to know exactly when a beacon is advertising and also introduces a relative drift due to random jitter accumulation throughout time. This also means that instead of a single point at which a beacon is expected to be heard, there is an interval of at least 10ms. The paper takes this into consideration when researching the scanning interval tradeoffs.

As such, BLE introduces new challenges to inter-device communication. As scanning for beacons is an energy-intensive task, reducing the amount of scanning time required prolongs the battery life of the device. To reduce the scanning, however, devices need to know when to scan so as to maximize the chance of hearing certain beacons. This requires beacon synchronization, and as outlined in chapter 3 is not an easy task in a BLE environment.

## 2.1 Working out in the gym

To aid in showing the advantages of novel inter-device communication paradigms, I present a scenario in which a user goes to a gym to work out. The user carries with them a collection of "smart" devices (e.g. smartwatch, smart shoe, music player, fitness band) that regularly offload data to the user's smartphone. The gym itself is equipped with weights and machines that allow authorized devices to connect to them and read off their data in real time. Furthermore, the collection of the devices and equipment used by the user changed as the workout progresses (e.g. different machines use different sets of devices for data collection). The goal of the paper is to ensure that the user's data is collected in an organized manner with minimal impact on the health of devices.

# Chapter 3

## Past Work and Newfound Challenges

### 3.1 Device Management

Academic circles have approached the idea of managing collections of devices several times in the past, however, a lot of the papers were written at the time where the modern abundance of devices was merely a vision. Since then, the way devices interact has changed dramatically, and the consumer market has produced new standards for computations. Prior papers, however, give lots of helpful insight into how researchers wanted to organize peripherals and aggregate all of their data.

#### 3.1.1 Personal Server

The mobile computing revolution produced some ideas of a local anchor. This anchor, the personal server, was the main point of contact and storage in the multi-device world. This idea that was at the cornerstone of Ubiquitous computing, had a trustworthy central node that would be in charge of the devices around it. However, these personal-server type solutions [4, 34] were often not constrained by resources, having a wired connection to their peripherals. Furthermore, this architecture created a single point of failure whereas if the personal server was downed, the system links were broken and peripheral devices were unable to communicate.

#### 3.1.2 Smartphone-centered Hubs

With the advent of the smartphone, more papers started focusing on the new mobile powerhouse [3, 26]. These papers explored the possibility of making the smartphone the hub of peripheral devices, which would aggregate all sensor data as well as provide a connection to a long term storage entity. This architecture is very representative of the current state of affairs, as more and more peripheral devices connect directly to the user's smartphone and offload information onto it, without contacting any other devices. In these architectures, the absence of the smartphone will cause local data aggregation, and in the case of limited memory on peripheral devices, data loss. Furthermore, in a scenario such as a gym, the smartphone may be

left in a locker, or the smartphone might run out of battery, reverting to the feared single point of failure that was a flaw in a personal-server type architecture. As modern devices become more powerful, they should be able to take on various hub responsibilities without having to always contact a smartphone.

## 3.2 Clock Synchronization with Wireless Nodes

With BLE, there are several challenges to align scanning intervals of listening nodes and advertising intervals of broadcasting nodes, as nodes cannot do both at once. Although there has been lots of work done concerning both synchronization and sensor networks, these approaches are not very applicable to BLE paradigms. Wireless Sensor Networks (WSN) has been a hot topic when focusing on communication between wireless devices. These networks have a more constrained communication schematic than regular sensor networks. However, there have been plenty of papers that organized these widely distributed systems with consensus-like algorithms.

Clock synchronization in a distributed, possibly wireless, system has been a topic of many papers [6, 15, 16, 22]. These papers, written at different times, were crucial in developing the idea of consensus among distributed nodes, however, they are not very applicable to low-powered systems. These synchronization protocols are mostly concerned with minimizing drift between nodes, and often use the master-slave hierarchy with virtually unbounded bandwidth between the two to ensure low latency in communication. Although the master/slave concept is abundant in synchronization paradigms, the assumption of boundless communication is one that cannot be made in low-energy systems as communication is a costly commodity. Similarly, sensor network papers that attempt to provide a consensus algorithm solution such as [24] do some under a connected sensor network hierarchy. The sensor networks mentioned in the relevant works are mostly homogeneous sensor networks with pre-set configurations where each node in the network is a fairly autonomous data publisher. WSN consensus papers [8, 10, 27, 30] tackle power efficiency and communication in a potentially lossy environment. These papers, however, assume that a node may listen and communicate at the same time, something the BLE [14, 28] cannot do at the scale of small battery-powered devices.

Surveys on clock synchronization with or without sensor networks [29, 35] provide an enticing overview of the existing solutions already implemented. NTP and RBS emerge as the better contenders for synchronization in the BLE communication paradigm.

### 3.2.1 NTP

The Network Time Protocol (NTP) [25] was introduced to provide a way to synchronize clocks among "packet switched, variable latency" machines in a network. The protocol constructs a hierarchical structure with masters at the higher tier which produce a high-precision time reading. These top tier time sources are one hop away from next tiered machines which are considered to be slightly less precise, but frequently updating their clocks to line up to the top tier masters. This hierarchical structure passes down the correct time and occasionally refreshes it due to drift and skew.

The protocol employs computation of latencies to figure out its offset from the "ground truth" provided by the top tiered master. However, as described in [29], NTP was not designed for networks power consumption is of a greater concern than shaving milliseconds off of a synchronization scheme. NTP does make the communication between peers efficient, but the protocol necessitates constant scanning which definitely drains the smaller batteries in a wireless sensor network. The protocol, along with many other synchronization protocols, assume nodes are always listening, always processing, and consider occasional messages to be lightweight. None of those three things are true in our low energy environment, as such, NTP becomes incredibly inefficient way to sync time. Furthermore, we have no need to sync nodes to an absolute time, but rather, we can use relative time intervals and causal relationships to determine the node's timely presence in the system.

### 3.2.2 RBS

Reference-Broadcast Synchronization (RBS) [9], shown in figure 1, is a novel technique used in sensor network synchronization. The algorithm is concerned not with absolute time precision, but rather relies on relative synchronization to a master node. In essence, it is a modification of the master/slave synchronization for node meshes that are concerned with aligning their clocks to the main node with minimal energy drain. A "master" broadcasts a message, nodes accrue the latency of the broadcast, and contact each other to figure out the relative latency between them and the master.

Although RBS is a very good option for wireless sensor networks, it requires all reference nodes to be listening and broadcasting at the same time. As the protocol states, once the broadcast is received by several nodes, they broadcast to each other. Unfortunately, in the case of BLE, both of the nodes will miss each other's broadcasts, or in the best case, only one node will receive it. In order to mitigate this occurrence, special communication paradigms for the protocol would have to be used where there is a backoff scenario if a node does not receive a ping from a nearby node. These approaches will mean more time scanning and potentially, more vicious scan/broadcast cycles which will quickly drain the BLE device's battery. As such,

RBS, albeit modified, will not be the most efficient way to align sleep cycles in a BLE mesh. Additionally, the BLE connected mode could be used, however, as the evaluations show in section 6.1, it is not a very power-efficient option.

Furthermore, most of the existing protocols are made for highly precise applications that require minimal latencies, which is not the primary objective for this project. As power efficiency is prioritized over anything else in the wireless system, it does not require high precision and can afford to work on human time, that is, incur some latency in order to preserve battery. The tradeoff is a lower response rate in favor of power efficiency.



## Chapter 4

# Device Management with MiHub

Along with Dr. Albert Harris and Professor Robin Kravets, I have worked on an architecture called MiHub which aims to combine all previous knowledge of inter-device communication with the mobility and resource limitations of modern devices. The architecture is designed as an overlay to existing networking structures containing peripheral devices with sensing capabilities. The MiHub architecture dictates the unified communication protocol and hierarchy of a collection of personal devices, and maintains a smooth stream of sensing information while reducing sensing redundancy in the system.

### 4.1 Components

The MiHub architecture shown in figure 2 establishes the hierarchy of devices and how they communicate.

#### 4.1.1 Device

As the foundational building block of the architecture, each device brings a collection of capabilities to the system. Every device varies in intended usage, computational ability, battery capacity, communicational bandwidth, and sensing capabilities, contributing its unique set of skills to the collection of user devices. As such, each device which intends to be part of the architecture has to be able to produce a capability list that lists all of its potential functions to the group. In addition to the system parameters, the devices also contains various sensors and consequentially, various services that it may offer for the group.

A device is also responsible for establishing its presence in the system, emitting a heartbeat advertisement with its device ID every Beacon Interval as set by the MiHub of the proximity group. If the MiHub misses three heartbeats from the device, it is marked as failed, and the service migration algorithm is started. Additionally, every device is also asked to produced a Bully score to be used in leader election. Leader election uses the Bully Algorithm [13] with the Bully Score as the heuristic function. The Bully Score is computed by the following formula:

$$M = (0.60 \times P) + (0.25 \times D) + (0.15 \times B)$$

where  $P$  is the speed of the processor in MHz (e.g. 1,200 MHz),  $D$  is the throughput of the data connection to the cloud in Mbps (e.g. 7 Mbps), and  $B$  is the remaining battery capacity in mAh (e.g. 2100 mAh). Although the current formula prioritizes computational ability and battery data bandwidth, this formula can be altered to optimize for different parameters. A device that is not the MiHub is designated as a peripheral device in the group.

#### 4.1.2 Sensors and Services

A device with specific sensors can offer various services to the system. The sensors, (e.g. accelerometer, gyroscope, temperature, heart rate) offer a glimpse into the system configuration of the device, and additional queries can be done to determine the specific capabilities of the sensors themselves. For instance, the accelerometer in a smart shoe might be capable of reading off data at a rate of 100 Hz while a smartphone can do so at 100 MHz. Depending on the importance of specific sensor information, some sensors might be more informative to the users than others.

The device's collection of hardware sensing capabilities also combine to offer data services. Service streams pre-process raw hardware data to provide a more readable version of sensing information to the environment. Additionally, sensor data from multiple sources may be combined to offer usable statistics such as motion data (e.g. combining accelerometer, magnetometer, and gyroscope) and location awareness (e.g. combining GPS, WiFi localization, and IoT beacons). These services are configurable to be sampled at variable rates to align with the configuration from the group leader. Although services from different devices may be based on similar sensors (e.g. motion sensor utilizes accelerometer and gyroscope for motion sensing on smartwatch and on smart shoe), they do not necessarily provide similar information, as such, services from similar sensors may be used for a different purpose (e.g. motion sensing on shoe sensor provides running/foot activity while smartwatch motion sensing provides more insight into arm motion activity).

#### 4.1.3 Proximity Group

Proximity groups are devices located within communication reach of each other that can be considered a computational unit. These devices are governed by a single MiHub of the group which acts as the data sink and configuration manager of the group. Proximity groups are formed once a leader is elected and designates tasks to each device, more on leader election and configuration in section 4.2.1.

#### 4.1.4 MiHub

The MiHub is the elected leader of a proximity group which acts as the hub of the group. It is responsible for making decisions on service selection as well as maintaining the overall health of the proximity group. MiHub aggregates all sensor information from the proximity group member nodes and attempts to offload all information to the personal cloud. MiHub monitors the membership of the proximity group by listening to device heartbeats and reconfigures the sensory output of the group upon any change of membership (e.g. device fails, new capable device enters). Not all devices can become a MiHub due to their resource limitations, more on MiHub election in section 4.2.1

#### 4.1.5 Personal Cloud

The personal cloud is the main aggregator of personal information. The cloud provides long-term storage of personal information visible only to its owner and may be accessible from any internet-capable device. If a user's personal devices form several proximity groups, the MiHubs of the groups are responsible for communicating with each other via the personal cloud in order to make more contextually-aware decisions, more in section 4.2.4

### 4.2 Features

In order for the aforementioned components to efficiently work together, there are certain procedures that are necessitated for smooth transition between system states. Additionally, with the architecture's stability come various features that were not possible with simple client-server communication.

#### 4.2.1 Leader Election

Whenever there is a change in the proximity group leadership, a leader election is triggered based on the following algorithm.

```
Proximity Group membership changes
if MiHub exists with Bully Score  $BS$  then
  if New Device is detected then
    MiHub requests  $BS_{new}$  Bully Score from New Device
    if  $BS_{new} > BS$  then
      Leader Election is triggered
```

```

    else
        Configure New Device
    end if
else
    Reconfigure member devices based on device departure (Migration)
end if
else
    Leader Election is triggered
end if

```

Whether a device is added to the group or removed, a leader election is triggered only when the current MiHub finds itself less capable or is not found in the group. This prevents MiHub thrashing as lower-powered devices do not trigger a new leader election.

Leader election follows the Bully Algorithm [13] where all devices advertise their Bully Score and only re-broadcast if they hear a score lower than theirs. If a device broadcasts a score and does not hear a response within two Broadcast Intervals, the device becomes the MiHub and informs all other members of the group of its newfound role. Once peripheral devices are informed of their leader, they advertise their capability lists to the MiHub. The MiHub processes all capabilities under its leadership and configures each device to provide certain services that reduce redundancy in the system yet still provide the necessary information.

### 4.2.2 Replication

As some sensory capabilities come standard in modern device (e.g. accelerometer, gyroscope, heart rate), there may be several devices that offer similar services. Although some similar services may offer different metrics as established in section 4.1.2, redundant services may need to be either turned off or scaled down.

#### Primary and Secondary Services

For every service group, the MiHub designates a primary service node and a secondary service node. The primary service node is responsible for sampling at the full rate and providing data in an orderly manner, while the secondary service node is required to sample its service at a lower rate and provide data for data loss prevention and data verification. As such, by reducing the sampling of redundant services, the system prolongs the overall health of the proximity group. As devices can provide various services, a device might be designated as a primary service provider for specific data and a secondary for others, as such, the device's departure from the group will cause a migration delay in data. As such, in the gym scenario, the smartwatch

can sample the heart rate of the user at half the rate of the body sensor, however, if the body sensor’s battery dies or it leaves the proximity group, the secondary heart rate data provided by the smartwatch will alleviate the data lost in the reconfiguration of the primary heart rate service.

### **4.2.3 Migration**

Upon the departure of a previously utilized device, the MiHub is responsible for reconfiguration of services within the Proximity Group. If a peripheral leaves the group, its primary service responsibilities have to be assigned to a node that was the secondary service provider. As this reconfiguration period takes some time, there is some data loss in the transition, however, as shown in section 6.3, the tradeoff of the data loss to system health supports the use of secondary service providers.

If a MiHub leaves the proximity group, an election period is triggered, which will introduce a new MiHub that will reconfigure nodes to adjust their data collection protocols. For instance, if the MiHub fails or leaves the group, and a MiHub with less data bandwidth is elected, the MiHub will be able to reconfigure incoming services to be pre-processed or data rates reduced to adjust to the newfound resource limitations.

### **4.2.4 Contextual Awareness**

As the user’s personal devices may form several proximity groups, and thus, several MiHubs communicating to the Personal Cloud, various application service decisions can be made utilizing system information. Based on the stream of incoming service information from multiple MiHubs, the Personal Cloud determines the Proximity Group that currently contains the user, and thus, can make contextually aware decision on how to inform the user of various service updates. This user-aware knowledge can be used to intelligently route notifications to the device that the user is using rather than ping all user devices as is currently the standard. Additionally, the Personal Cloud can also detect the type of activity that the user is doing, and thus, prevent application service notifications from distracting the user from their task. For example, if the Personal Cloud knows the user is lifting weights based on the heart rate and motion sensing information from the smartwatch, the Cloud might hold off from sending notifications to the user until they are done. Consequentially, the Cloud can send an SMS notification straight to the smartwatch if it determines that the user’s smartphone is currently not in the User’s Proximity Group (i.e. smartphone is left in the locker as the user works out).

## Chapter 5

# Passive Synchronization

In addition to an device management hierarchy, nodes need to synchronize in order to know when to broadcast and when to listen. This will reduce the cost of communication while extending the battery life of the node. When designing a synchronization system on top of the BLE protocol, several ordinary practices have to be re-assessed. In particular, limitations of the BLE stack, especially on smaller devices, render existing protocols significantly less effective. Various architectural aspects are introduced, including a Selective Scanning idea and the structure of a deployable setup.

### 5.1 Selective Scanning

Considering various BLE modes which can aid us in the configuration frequency of our beacons. This data determines how often to reconfigure the in order to synchronize them. This reason for this motivation is to conserve power for all nodes involved by creating a more compact, efficient heartbeating technique. In an ordinary BLE environment, every beacon is configured to advertise at a certain Beacon Interval (BI). The node that scans the environment for changes has a configured Scanning Interval (SI) which dictates how long it will listen for packets and a Scan Window (SW) which will tell the node how often to restart the scanning process. Scanning is a power intensive operation, while increasing the Scan Window can reduce the time and thus power allotted to scanning. As such, it is advantageous to sync up these intervals in a way that can save power for all nodes involved.

For example, consider a beaconing schedule for six beacons with varying BIs in figure 3. As the intervals increase, it becomes more challenging to catch the advertisement at the right time. As such, the scanning node would have to increase its Scanning Interval in order to maximize the possibility of capturing the long interval beacon. In the example shown, if the Scanning Interval (SI) is 500ms and the Scanning Window (SW) is 700ms, so that the scanning node can save power by not powering the radio for 200ms, then the scan will miss the green beacon's heartbeat and the yellow beacon's heartbeat. This also means that the Scanning Interval and Scanning Window of the scanning node will have to be close together, leaving almost

no sleep time, draining the battery of the scanning node and leaving minimal room for processing data.

If the beacons are synchronized in a systematic way, the scanning load and power consumption of the MiHub can be reduced. With synchronization, the intervals line up in such a way that it provides a guarantee that the MiHub will hear certain beacons within its SI. If the broadcasting periods are aligned, the SI of the MiHub, and possibly other devices, could be massively reduced.

## 5.2 Architecture

For the synchronization technique, a master-slave approach is employed whereby a master is supposedly within reach and is a reliable node. This technique, however, is not entirely reliant on the master's existence, but rather it is largely simplified. System without designated masters will be considered later in the section. The master in this hierarchy is the MiHub.

### 5.2.1 Master

A master node is a beacon that is configured to scan at a certain BI. What makes it a master, however, is its location in the system. The master is the anchor for the hierarchical organization of nodes, that is, the relative importance of nodes and their flexibility in BI choice is measured by their distance from said anchor. For example, if a node is close to the master, it should be considered to have a higher importance to the overall health of the system as opposed to a node far away from the master, which has the liberty to extend its BI to a longer time. The master is considered a reliable node which does not change its configuration throughout its lifespan. There could be several masters in a single grouping.

### 5.2.2 Node

A node is any BLE device introduced to the system. This node will broadcast relevant information (such as color of the bottle in the case of the vending machine) or RSSI of its nearest master. The BI and SI of the node will be determined by the introduction into the system and various configuration intervals it will encounter in its lifetime. A node is assigned to a certain "tier". A tier determines the node's importance in the system, and is based on the average RSSI reading from the nearest master. This will dictate the window in which the node will have to scan in order to find nodes of a certain tier.

### 5.2.3 Tier Interval Beacons

Each node will beacon as part of the tier it belongs to. The tiers are calculated based on the node's RSSI to the master over its several readings in the Configuration Period. The tier thresholds can be altered depending on the situation. Once the node knows its tier  $n$  and the Tier Interval (TI), the node will be able to broadcast at  $n \times TI$  after the master's beacon. This way, all nodes in a single tier will broadcast at the same time (with a random jitter as dictated by the BLE spec), and create an interval in which all nodes of a certain tier can be heard. Figure 4 shows a master beacons at BI and the green interval for every tier shows when that tier is allowed to broadcast. Although nodes of the same tiers may align to different master beacon broadcasts, the amount of time passing between a master's broadcast and the tier's broadcast will be predictable. In order for the master and tier beacons to not collide, BI and TI should take up values that do not align over the course of seconds/minutes to prevent loss due to collision.

### 5.2.4 Configuration Period

Once a node is booted up and/or introduced into a group of BLE devices, it will enter its Configuration Period. In this period, the node will scan for all of its surrounding nodes and configure itself accordingly. The node will listen for two broadcasts from the nearest master. The period elapsed between the two broadcasts will be used to calculate the master's BI. Additionally, the node will listen to see if other nodes around it are beacons, and attempt to estimate their TI's as well. This second calculation, albeit less precise due to potential inconsistencies, will select the lowest TI heard and will dictate the node's assumed TI. Once this is established, the node is synced up to the system and can selectively broadcast and listen to other nodes.

### 5.2.5 Selectivity

Once a node knows the master's BI and the presumed TI of the system, it can choose to selectively listen for other beacons and broadcast in the correct tier level. The node knows that it will hear the master every BI and all members of tier  $n$  precisely  $n \times TI$  time after the master. Once that is known, the node can scan for an interval of  $SI = 20ms$  at a desired time to hear a certain node with high accuracy (Scanning Interval considerations can be found in the evaluation portion in section 6.2). Nodes in higher tiers have the flexibility to decrease their beacons frequency and still broadcast at the right tier time to be heard in the system. For example, Tier 4 nodes can scan/broadcast every 4<sup>th</sup> master's beacons however, they would have to extend their SI as there will be more potential for clock drift in the extended period of time. The master can now listen with a fairly small SI to hear all nodes in the group. Finally, if a node misses three broadcasts



from the master, it knows it has drifted, and thus, enters the Configuration Period to re-configure its BI and TI.

## Chapter 6

# Evaluation of System Configurations

In order to motivate the choice of passive peripheral communications over its alternatives, as well as the tradeoff of energy consumption compared to data loss in a MiHub migration scenario, experiments were set up with TinyBLE [32] development boards and nRF51822 BLE dongles [2]. To motivate the selection of passive peripheral mode as opposed to other alternatives, passive mode is tested against connected mode in terms of energy consumption. Next, the importance of the Beacon Interval is assessed with regards to energy consumption, and finally, the impact of Scanning Intervals is assessed with regards to energy consumption.

### 6.1 Connected vs Passive

In passive mode, the BLE device advertises its presence based on a certain beacon interval and is capable of including some data in its advertisements. Connected mode is when a listening device, often the MiHub, sends a connection request to the peripheral and they enter a more data-intensive connected mode. In this mode, the two nodes may exchange much more information, throughput of which depends largely on the connection interval which is decided by the nodes.

Passive mode can be used for heartbeating and determining the distance of the beacon, while connected mode can be occasionally used to efficiently reconfigure various parameters of the beacon. These power evaluations will give a good estimate of the relative cost of connection so that it could be determined when a connection is a more feasible approach, and possibly offer an alternative to passive beaconing.

For the preliminary evaluation of the power consumption, I instrumented a TinyBLE [32] board which contains an nRF51822 BLE beacon [2] to beacon at different beacon intervals, namely, at 100ms, 250ms, 500ms, and 1000ms as one can see in figure 5. The TinyBLE board shares the same BLE chipset as the Nordic nRF51822 Bluetooth Tag, however, it provides better means for power assessment. I measured the aggregate power consumption over the course of one minute. Similarly, I measured the power consumption of the aforementioned beacon in connected mode, in which a connection request to the TinyBLE board was sent and exchanged dummy information at a maximum rate.

I assessed the amount of time necessary to transmit 15,600 B in passive mode (just enclosing data into passive beacons) and connected mode (actively transmitting data). The TinyBLE can transmit 26 B of raw data every  $BI$  passively, taking 60 s to transmit the 15,600 B. In pull mode, the TinyBLE can transmit six 20 B packets every 250 ms, taking 32.5 s to transmit the 15,600 B. Even though it takes roughly twice the time to transmit the same amount of data in push mode as pull, our energy results show that push mode is still more efficient in energy-constrained environments. Figure 6 shows the energy efficiency of both scenarios transferring the same amount of data over the course of a minute.

As you can see, the power consumption of connected mode is significantly more expensive as passive mode, albeit faster, making it an unfavorable option for communication. Similarly, one can see that increasing the beacon interval also decreases the power consumption of the beacon. As such, the task becomes to maximize the beacon interval at which a single node advertises and minimize the number of re-configurations that a node requires so as to avoid needless connections.

Finally, the last option for an active request for reconfiguration is the Request/Response paradigm. In this mode, the beacon advertises its presence and a central can request more information by sending the peripheral a Scan Request packet. The peripheral can then respond with more information within a Scan Response packet and continue advertising without the need to enter connected mode. For our purposes, the central can pack reconfiguration parameters into a Scan Request packet and send it to the peripheral to make its adjustments. Unfortunately, that does come with its own cost, as the peripheral has to constantly listen for a Scan Request and as such, keep the RX radio always on. As such, for the system design, it was more logical to use a passive beaconing system for heartbeating and a short scanning window for accurate alignment as opposed to connected mode configuration.

## 6.2 Scanning Interval Window

To assess how small the Scanning Interval Window can be made, an nRF51 Dongle was instrumented to act as a MiHub which broadcasted a packet every 500ms. A TinyBLE board was used to listen for the BI of the MiHub, and log every broadcast it heard. Once the TinyBLE board heard the MiHub, it would calculate when the next broadcast should be heard, and set its new scan interval accordingly. Once the board missed three broadcasts from the MiHub, it would enter a Configuration Period in which it will scan until it heard the MiHub and continue calculating the next presumed broadcast. The time to scan for the hub node's broadcast was determined by the formula  $BI - 2$  to line up the next scanning interval with the presumed beaconing interval of the MiHub.

BI is the beacon interval of the MiHub, 2 is the milliseconds attributed to the deliver and processing latency of a broadcast. This calculation should produce the number of milliseconds the TinyBLE board had to wait until the next time it should scan for the MiHub’s beacon as per Figure 7. Due to the random 10ms jitter, when the scanning interval is less than 10ms, lots of broadcasts are missed purely because the interval does not cover all of the possible time.

In the results, averaged over five runs, the TinyBLE boards would run the routine for one minute and log all packets it received and configuration periods entered. Next, the energy consumption of each routine was recorded based on the SI of the trial, averaged over five trials. In the one minute of the trial, there were at most 120 beacons from the MiHub, however, as the timer started in the middle of the MiHub’s interval, the maximum number of logged beacons was 118. Figure 8 shows the number of beacons captured at different Scanning Intervals and the number of necessary re-configuration periods once three beacons from the MiHub have been missed. Previous consideration were made to place the scanning interval into the middle of the MiHub’s broadcast window, however, that did not prove to produce better results.

Furthermore, as you can see from figure 9, the increase of the scanning interval also increases the energy consumption of the beacon. Note, the scale of the graph starts at 2500 mA. Regardless, the difference between a 5ms scanning interval even when it causes lots of re-configurations and a 50ms interval that does not have to reconfigure is about 700mA per minute. This is fairly significant if these beacons are expected to serve months and years at a time. The inconsistencies in the trend with 9ms to 12ms energy draw readings are possibly due to the decrease of re-configuration periods. Future work includes trials to determine where these inconsistencies are rooted.

### 6.3 Sensor Replication and Migration

To test the trade off of sensor replication and energy consumption, a TinyBLE board energy draw was measured with different motion sensor sampling rates. This tradeoff shows that having a secondary sensing node sampling its sensor at a lower rate will be less costly, but at the same time, mitigate significant data loss as described in section 6.3. To determine the benefits of low-frequency secondary replication, I evaluated the energy consumption of the motion sensing component of the TinyBLE, the other main energy cost aside from transmission. The TinyBLE has an on-board MPU6050 that contains a number of motion-sensing capabilities [32]. Again using the on-board current monitor, we measured the energy consumption (via current draw) of the TinyBLE sensor with only the motion sensors enabled over the course of one minute, using the sampling rate of 100 Hz and 40 Hz. Sampling at 40 Hz consumed about 56% as much as 100 Hz

(see Figure 10).

Full sensor replication in two nodes would cause about 9.96 A of extra current draw per minute. Configuring a primary sensor node to sample at 100 Hz and a secondary at 40 Hz reduces the combined energy cost to approximately 7.7 A. Thus, by reducing the backup sensor’s sampling rate, MiHub gains the benefits of redundancy while still managing the overall energy impact on the IoT environment.

Finally, to finish off the tradeoff calculation, the amount of data lost in the migration from the primary service provider to the secondary has to be considered. This time is the same time that it takes to reconfigure the proximity group after a primary sensing node leaves. In the experimental prototype, we configured our primary sensor nodes to beacon every 100 ms. After the primary sensing node fails, the MiHub has to detect a membership change and reconfigure the secondary service provider. To determine the loss of a device, the MiHub has to miss three heartbeats from the primary device, and thus, mark it as failed. Thus, failure detection takes three *BIs*. Furthermore, to accomplish reconfiguration, MiHub must perform a short handshake with the sensor node to be promoted and then transfer 40 B of information. In our prototype, the handshake takes 250 ms and the data transfer and reconfiguration takes an additional 250 ms. Thus, with a *BI* of 100 ms, the total reconfiguration time is 800 ms. The old primary node would have transmitted 8 sensor packets in that time, which are all lost. However, during the reconfiguration process, the secondary node, which is being reconfigured to primary mode, still produces 4 secondary messages, thus the total data loss is cut in half during the migration period and only lasts 800 ms.

Based on the evaluation, secondary sensor replication provides a favorable tradeoff between replication and energy consumption wherein migration does not cause a massive disruption in sensing data, while still retaining useful data for loss mitigation and data verification.

## 6.4 Noisy Environment Baselines

In an effort to motivate the efficacy of the MiHub and Passive Synchronization design, trials were set up to determine the data loss in beacon-dense environments. If a significant amount of beacons coexist within range of each other, random advertisements without a cohesive organizational structure can cause significant loss of packets. Trials of 1, 25, 50, and 100 iBeek [5] beacons were set up to advertise Eddystone URL packets with advertisement intervals of 1000ms and 500ms. Trials with 250ms and 100ms were planned, however, the iBeek beacons entered a failed state when configured to 250ms advertising interval and thus could not be used for the remainder of the trials.

For each trial, the specified number of beacons were assembled in a room isolated from any other Blue-

tooth noise. A TinyBLE device was placed in the middle of the beacons to act as a sniffer and collect any packets it would hear from any of the beacons around it. A Texas Instruments CC2540 Sniffer [20] was placed adjacent to the set up to act as ground truth for the packets that were seen in the environment. All of the iBeek beacons were set to 1000ms advertising intervals for the Eddystone URL protocol. Additionally, the beacons advertised their own sBeacon protocol that is used for configuration every 3 seconds. For the 1000ms trial, the sniffers recorded for a period of 3 minutes and the results are averages over 5 independent trials. For 500ms, the beacons were recorded for 2 minutes over 5 separate trials.

All advertisement beacons were recorded by both devices and the average number of advertisements per beacon was calculated, which produced the Average Advertisement Count. The percentage of data loss is calculated by taking the average number of advertisements heard by the TinyBLE and dividing it by the number of advertisements per beacon heard by the TI Sniffer as the ground truth. Similarly, the theoretical upper bound produce the theoretical data loss in the environment. The theoretical upper bound is calculated as follows:

$$\frac{\text{total length of trial}}{\text{EddystoneURL Advertisement Interval}} + \frac{\text{total length of trial}}{\text{sBeacon Advertisement Interval}}$$

Which for 1000ms computes to 240 beacons and to 500ms computes to 280 beacons. The results of the trials can be seen in figures 11 and 12. Although the TI Sniffer as the ground truth created some inconclusive results, one can see that the overall trend of data loss shows an exponential increase of loss as the number of beacons in an environment increases. Additional trials with 250ms and 100ms would quite possibly support this hypothesis.

# Chapter 7

## Prototypes and Trials

### 7.1 Smart Gym Demonstration

As a demonstration of the MiHub system, Professor Kravets's research team built a prototype smart gym that monitors a user's workout. The gym consists of several lat pull machines, each equipped with a TinyBLE board to detect reps using its motion sensor. This TinyBLE board also offers provides a way for a user's devices to connect to it and read off the motion information. The gym prototype also includes free weights that are equipped with TinyBLE board that are hooked up to a portable phone battery as the power source. The demo shown at Hotmobile 2015 is depicted on figure 13.

To demonstrate the MiHub organization hierarchy, the user is equipped with a Nexus 5X smartphone (1.8 GHz processor, 16 GB of storage, 2300 mAh battery) and a Galaxy Gear Live smartwatch (1.2 GHz processor, 4 GB of storage, 400 mAh battery), both running Android Marshmallow. The user is also wearing two Estimote [11] beacons, one on their head to measure body temperature, and another in the shoe, acting as a shoe motion sensor. Both the smartphone and the smartwatch run a fitness application that employs the MiHub architecture to keep track of the devices around it.

The Nexus 5X smartphone assumes the MiHub role as the app is pulled up on the phone. The smartphone manages both Estimote beacons as well as the smartwatch once they are within BLE reach. Each peripheral device heartbeats every 100ms to maintain its presence in the proximity group. Once the smartphone leaves the area, or is put into airplane mode, the devices trigger an election period, in which the Galaxy Gear Live smartwatch emerges victorious, and reconfigures beacons to sample at a lower rate, as it cannot process data as well as the smartphone. The Galaxy Gear Live smartwatch also lets the Personal Cloud know that it is now the MiHub in the proximity group that contains the user, so any notifications that come to the server should be routed directly to the watch. To push data to the MiHub, each device creates a passive BLE advertising message by accumulating data over the 100ms interval and compresses the data into the 26 B advertising message (5 B of the maximum of 31 B for a BLE advertising message are used for configuration and data length). The prototype system constrains numerical readings (e.g., accelerometer x

axis, temperature) to a 2 B value, providing room for 13 numerical readings every beacon interval.

Once the user enters the gym and comes up to either a mini lat pull machine or a free weight, the acting MiHub uses BLE RSSI-based proximity sensing to determine the closest machine or free weight. Once it establishes a logical pairing, the MiHub connects directly to the machine’s sensors and listens for data from the device. The weight sensors act as the primary source of data for the workout, counting repetitions of each exercise. The rep counter in the software stack is powered by a machine learning model that is trained to identify biceps curls for the free weights and the correct motion for the mini lat pull down. Throughout the workout, all personal devices advertise their sensor data to MiHub, in particular, motion and heartbeat data from the watch, redundant motion data from the shoe, and temperature data from the headband. The MiHub aggregates all of the data for consumption by the user.

This MiHub prototype was shown at the MobiSys 2016 demo session as the MiHub paper [33] was accepted to WearSys, a MobiSys workshop.

## 7.2 MiHub TinyBLE Simulation

A simulation of the MiHub design was additionally created for TinyBLE development boards as well as a Python simulation. Both of the simulation have not gotten to a presentable stage, however, both show the effectiveness of both the small scanning interval window as well as a Bully Algorithm election in BLE environments.



## Chapter 8

# Conclusion and Future Work

More companies are trying to tie together their consumer devices to appear in a so-called "home mesh" whereby device activity is monitored by the personal cloud. Service handoff protocols have reared their heads in the Apple [17] macOS and iOS platforms as well as in the Windows 10 [19] family integrations. Now that the industry has gone from centralized, one device, computing to decentralized, peripheral, computing, the trend is to make devices work together. If the consumer market develops that idea for their own platform, device management will remain a fragmented, ecosystem-only feature. In order to be a fully heterogeneous device management solution, a protocol has to be made that will leave room for any and all devices to join the group, making only an assumption for the form of communication, but not the software that runs it.

As a new Bluetooth 5.0 standard rolls out, many of the assumptions of lower throughput and more energy consumption for connected mode that were mentioned in this Thesis will be less powerful. However, the ideas presented in this paper can still be applied to the new energy profiles provided by the new protocol, making communication between devices even more streamlined and organized.

Furthermore, batteries will also retain longer charges [23], allowing for more processing and activity on smaller devices. Although a lot of the concessions made for the sake of energy efficiency in this paper will not be as relevant, the investment in making an energy-efficient protocol will only be advantageous in the long run, allowing bigger batteries and efficient communicational protocols to extend the lives of device groups even more.

Finally, further work has to be done to address various privacy concerns that any BLE system produces. This is especially important when wearables are introduced as they are prime suspects of security side channels of personal information [7]. To aid in privacy, Professor Hari Sundaram designed a BLE privacy paradigm called Incognito [21] which could be integrated into the MiHub architecture in the future.

# Figures

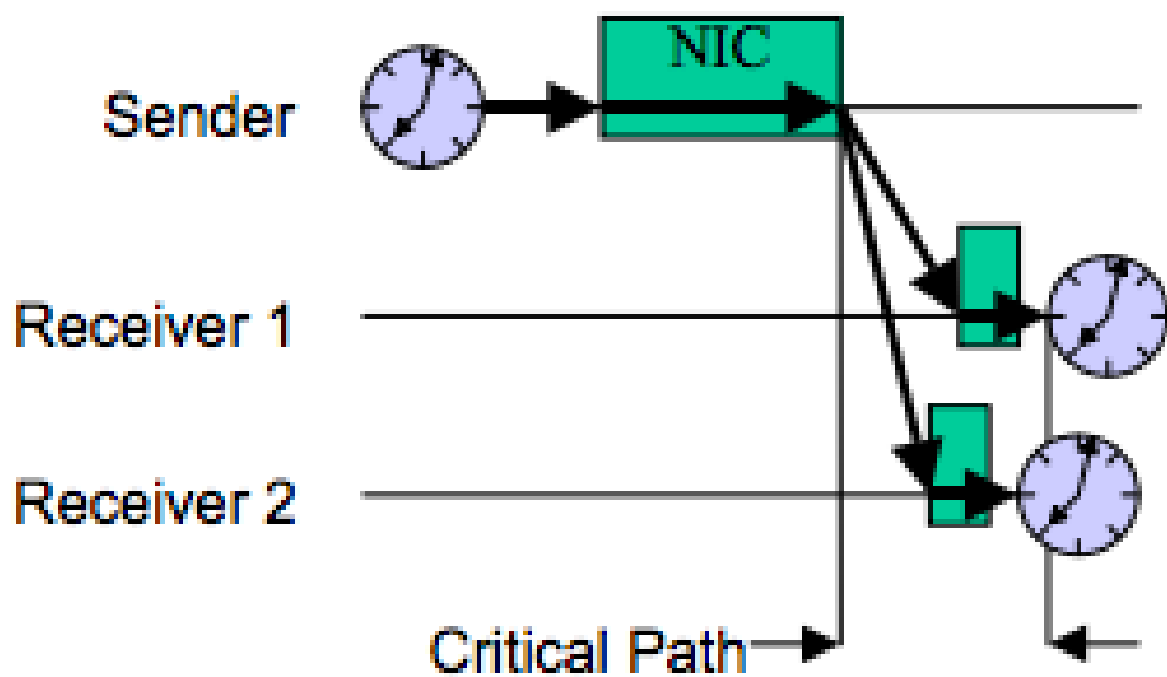
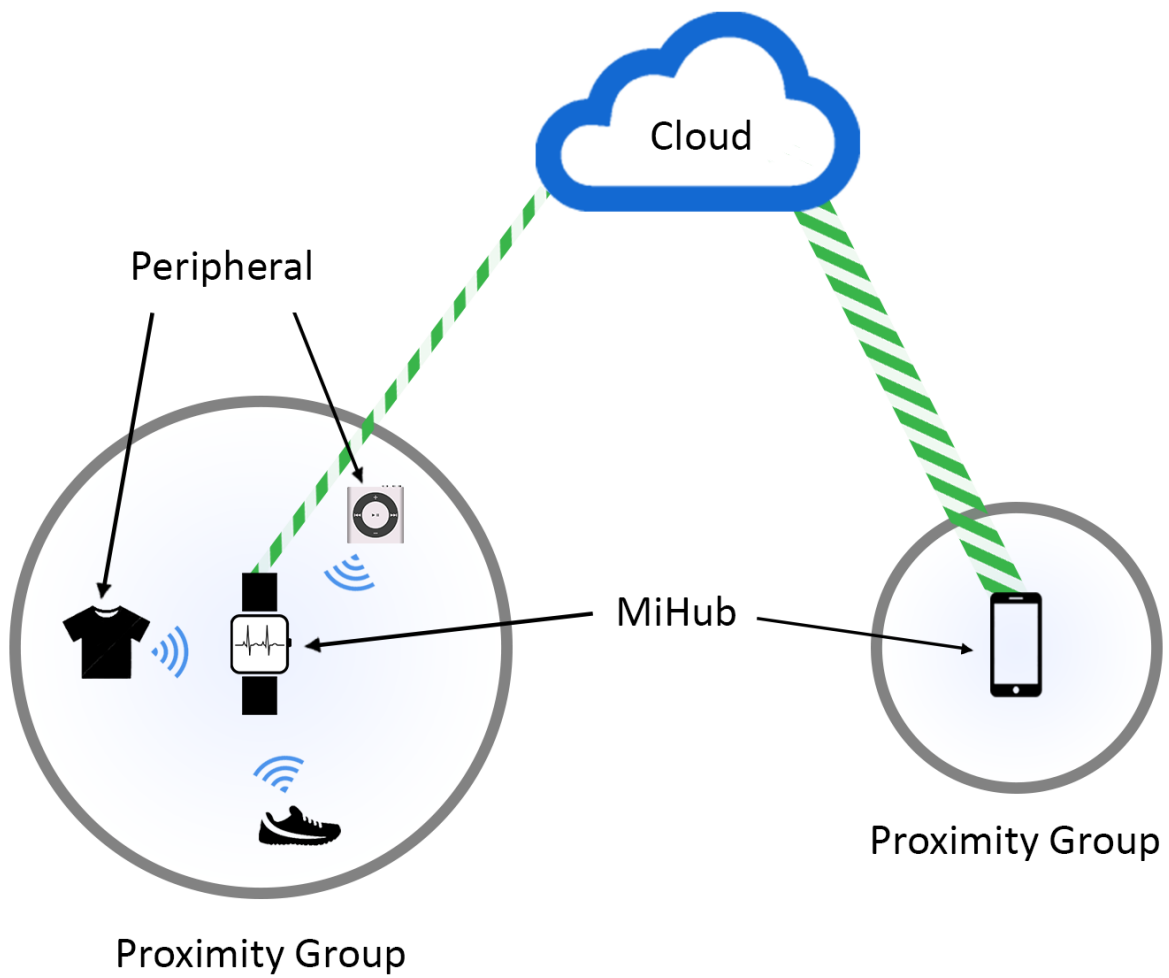
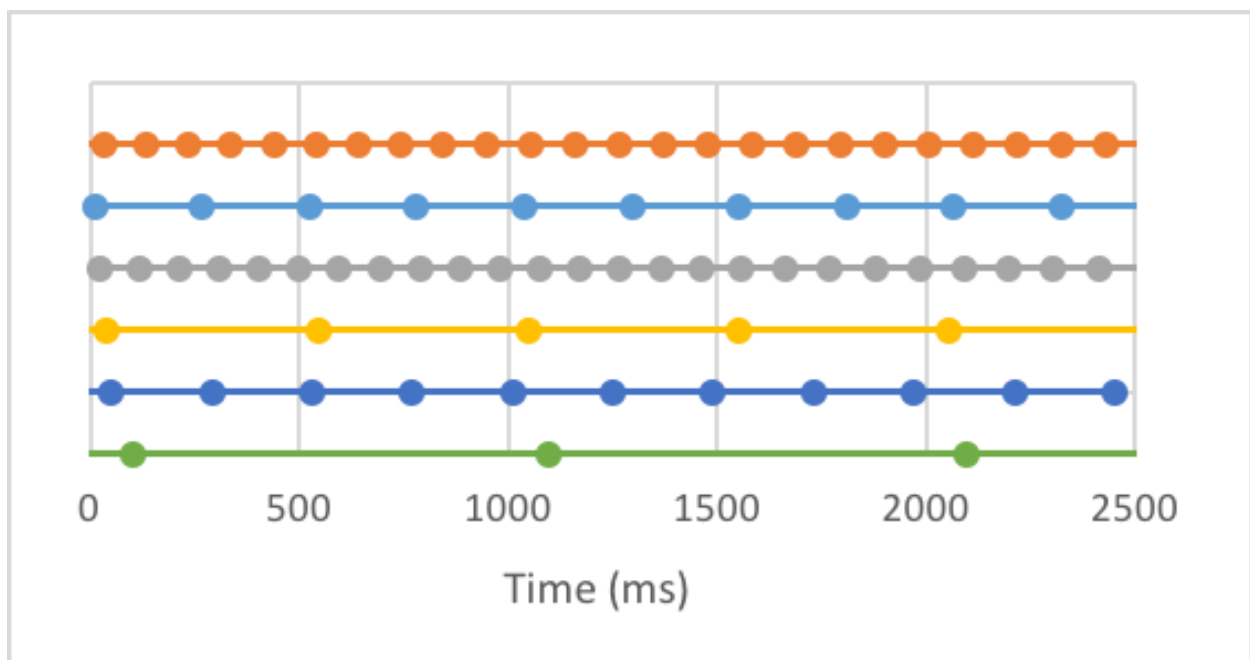


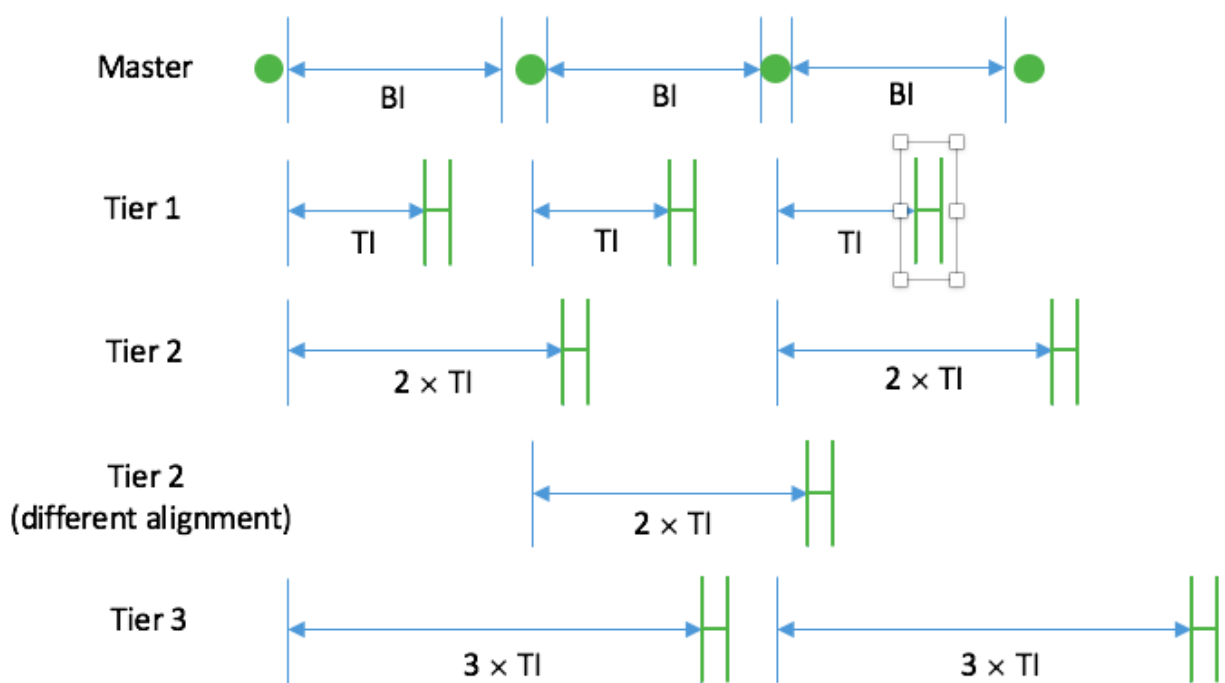
Figure 1: RBS in action



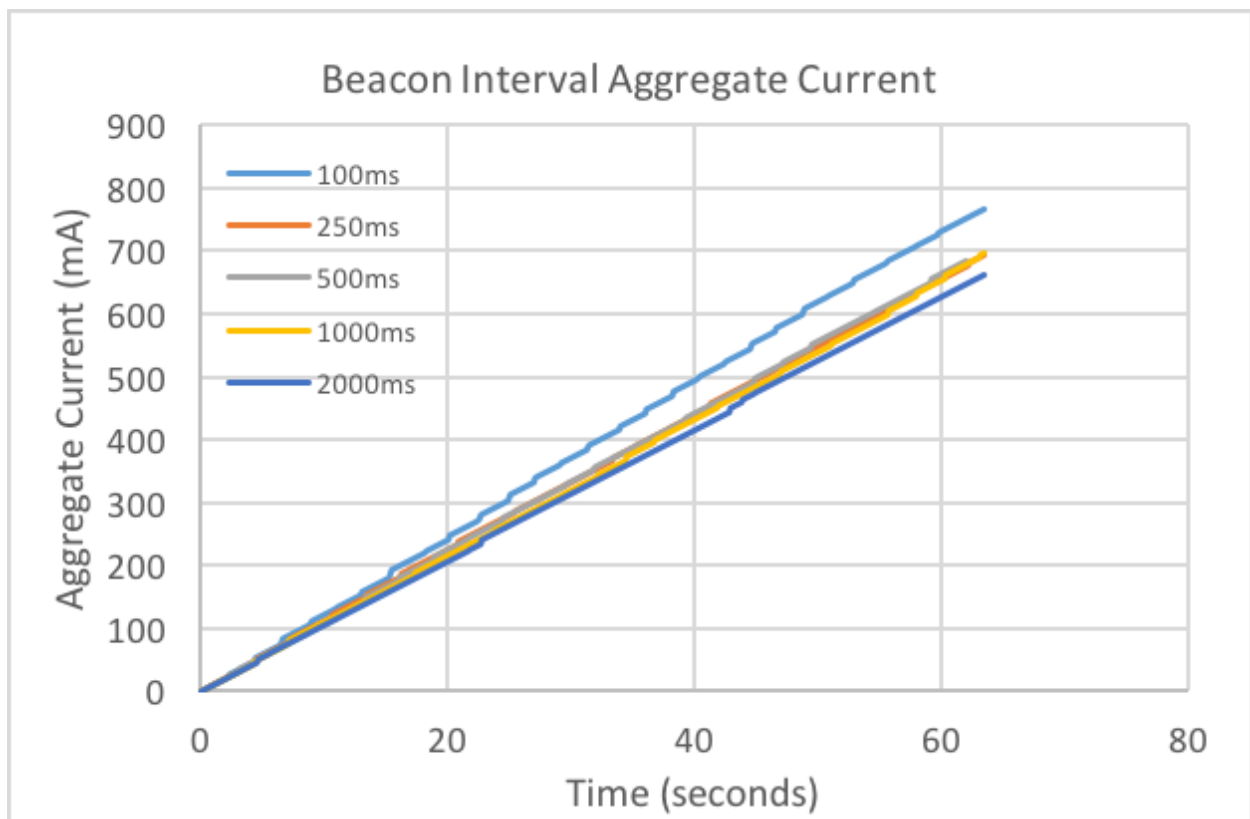
**Figure 2:** MiHub System Architecture



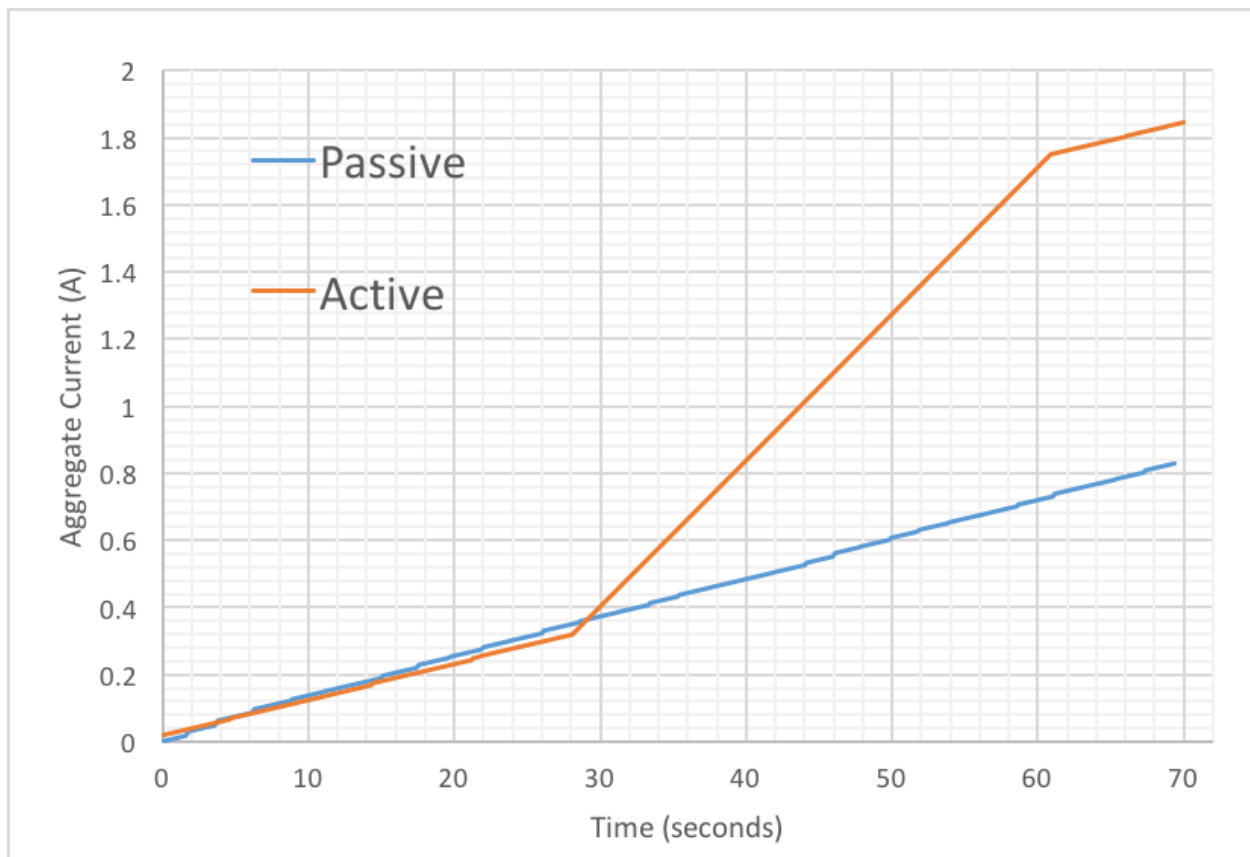
**Figure 3:** Several beacons with various BIs



**Figure 4:** Alignment of broadcast windows

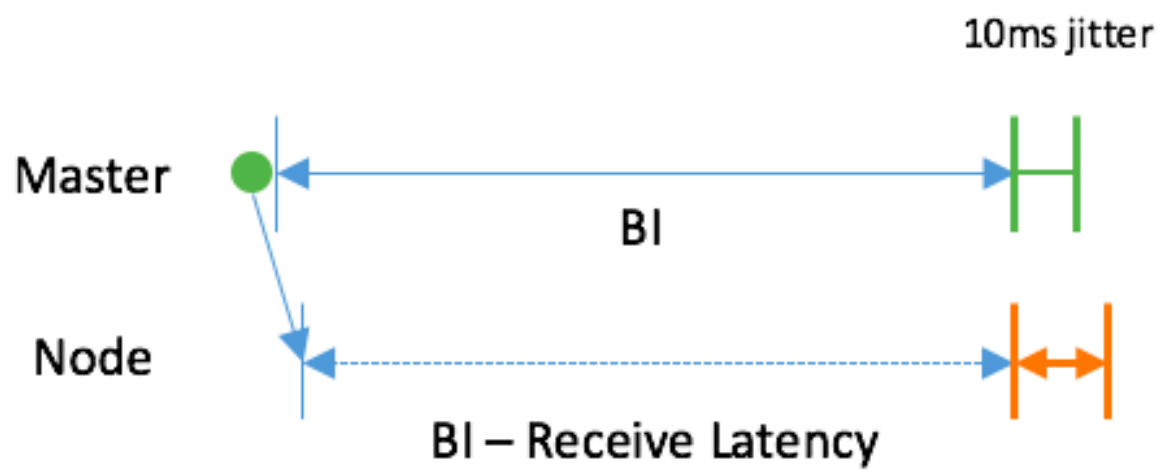


**Figure 5:** Energy consumption of various BI settings

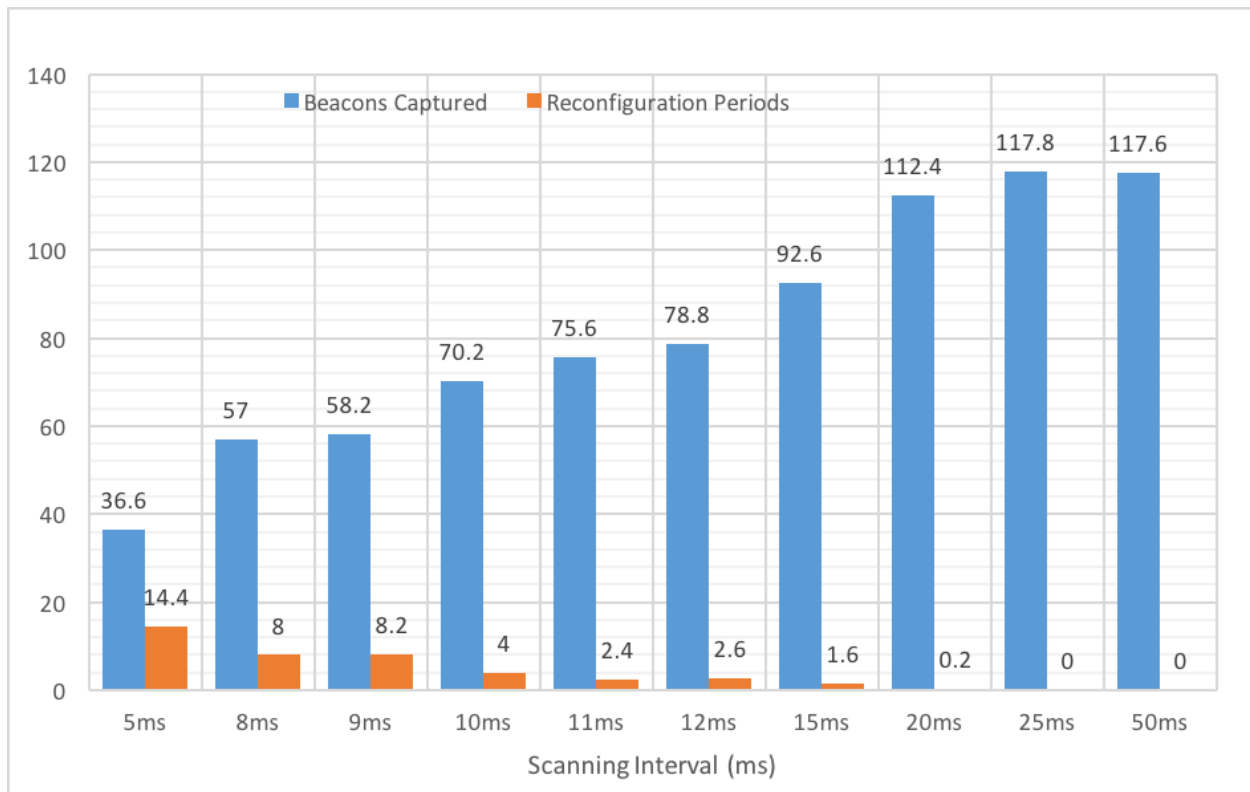


**Figure 6:** Aggregate current over one minute in passive and connected (active) mode

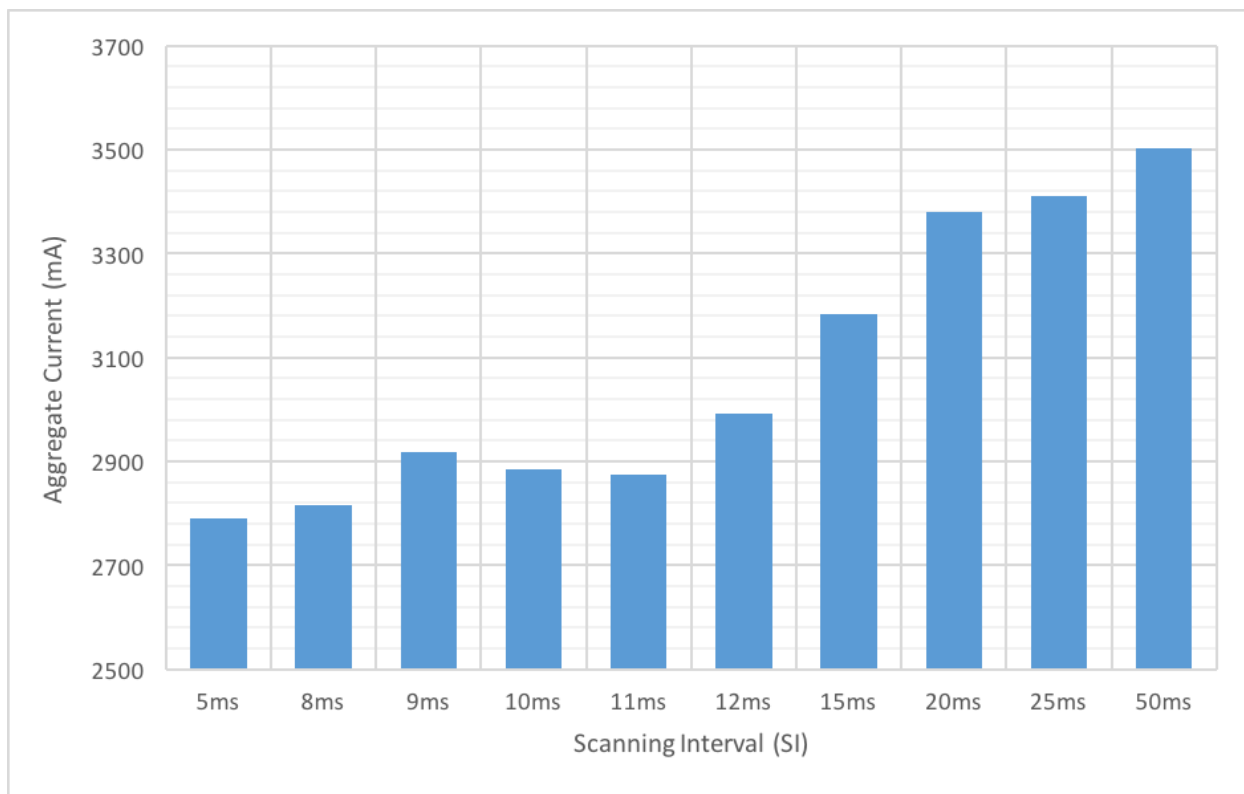




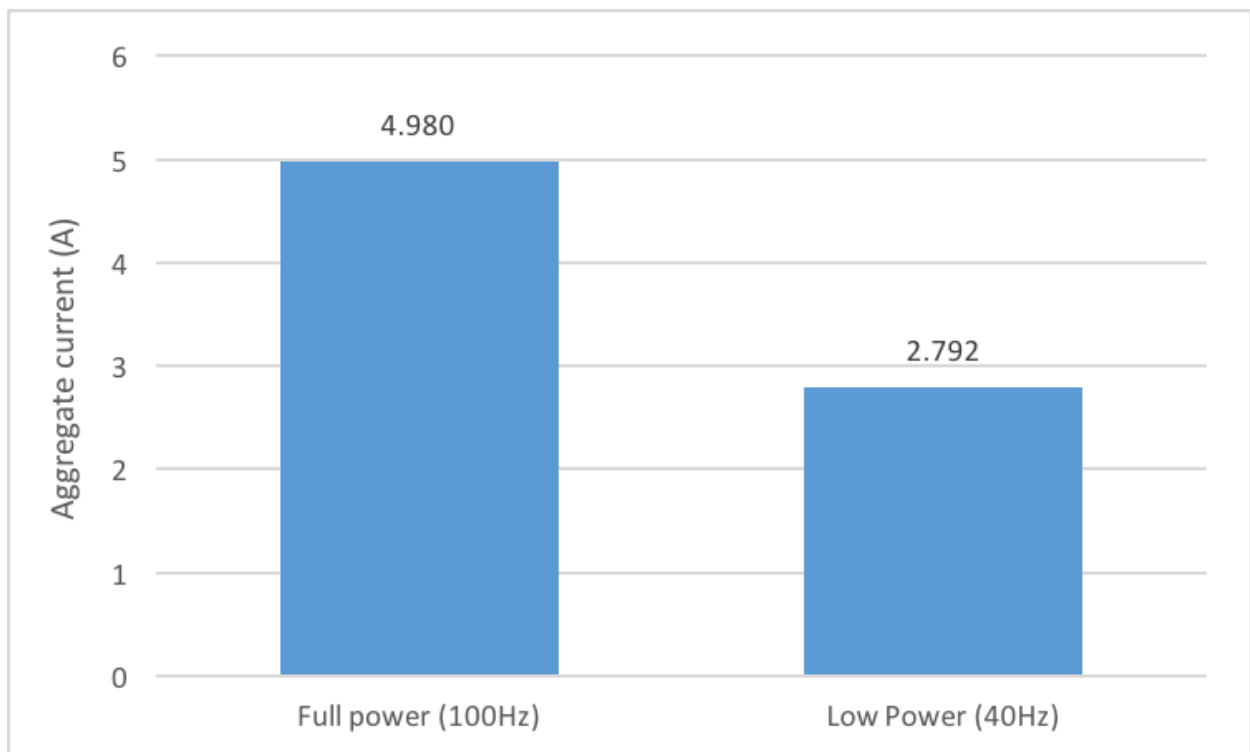
**Figure 7:** Master's potential broadcasting window and computation for node's scanning interval



**Figure 8:** Number of beacons received based and reconfiguration periods due to Scanning Intervals



**Figure 9:** Energy draw based on various scanning Intervals



**Figure 10:** Aggregate current over one minute

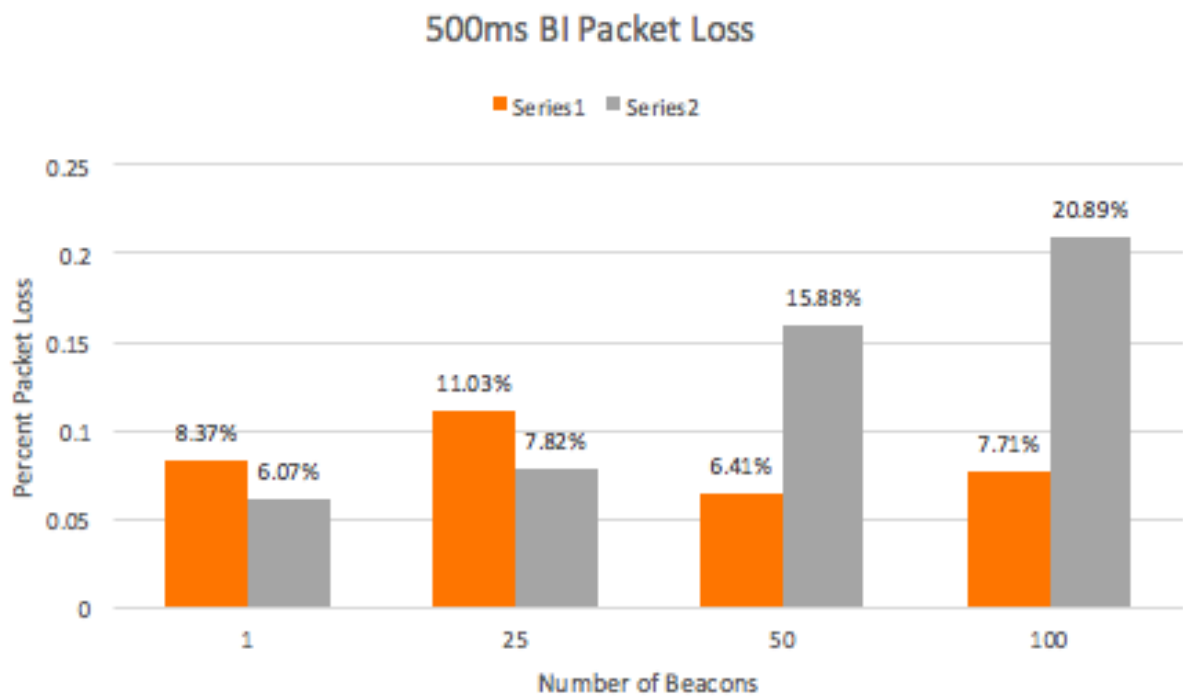
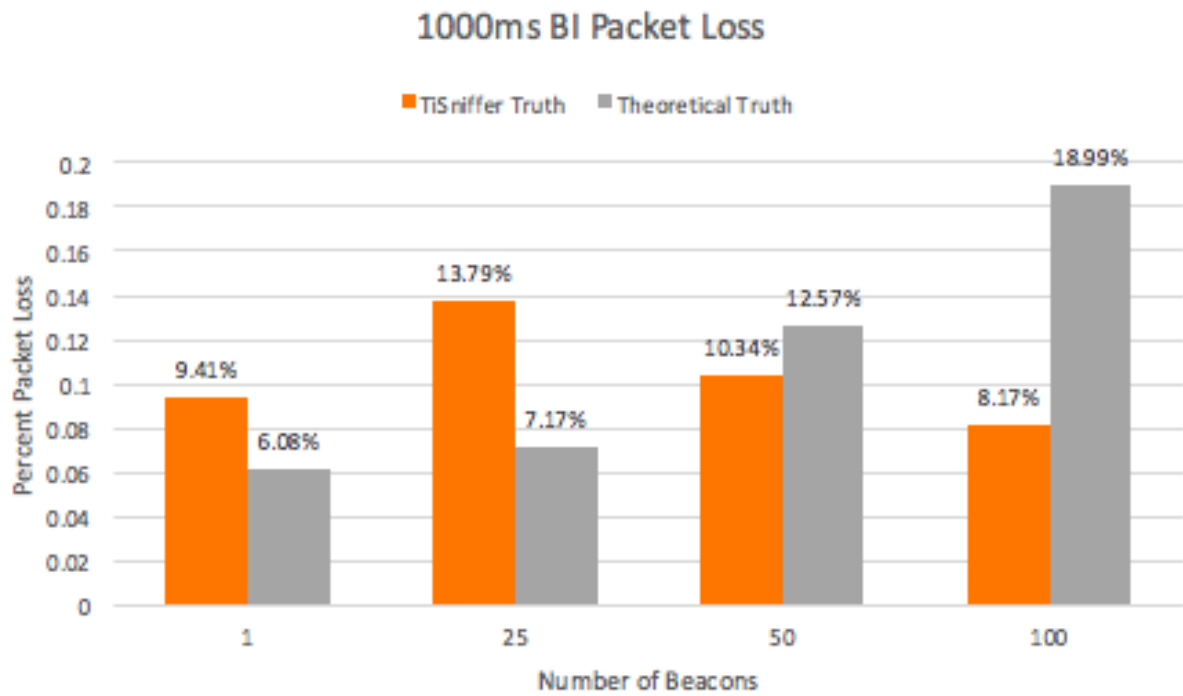
### 1000ms Beacon Interval

Beacon Count	Avg Advertisement Count	tiSniffer	% tiSniffer truth	% theoretical truth
100	178.536	194.42	0.918300586	0.810083333
50	188.136	209.8222	0.896644874	0.874259167
25	192.06	222.79	0.862067418	0.928291667
1	204.2	225.42	0.905864608	0.93925

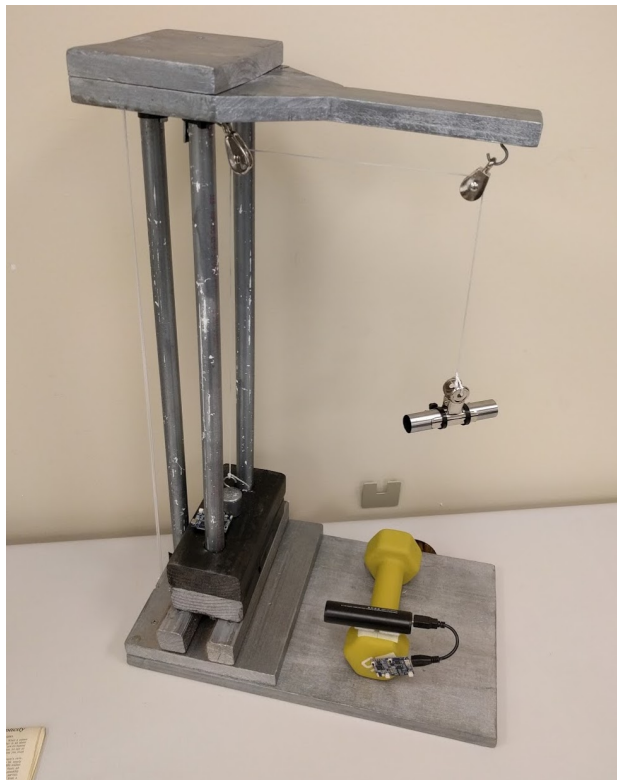
### 500ms Beacon Interval

Beacon Count	Avg Advertisement Count	tiSniffer	% tiSniffer truth	% theoretical truth
100	204.43	221.52	0.92285121	0.791142857
50	220.43	235.532	0.935881324	0.841185714
25	229.62	258.096	0.88966896	0.921771429
1	241	263	0.91634981	0.939285714

**Figure 11:** Advertisement loss percentage with 1000ms and 500ms beacon period and 100, 50, 25, and 1 beacon



**Figure 12:** Advertisement loss percentage with 1000ms and 500ms beacon period and 100, 50, 25, and 1 beacon



**Figure 13:** Mini lat pull machine, free weight, Nexus 5 smartphone, LG G smartwatch, and two Estimote beacons, demo configuration at HotMobile 2015

# References

- [1] Bluetooth smart technology: Powering the internet of things <http://www.bluetooth.com/pages/bluetooth-smart.aspx>.
- [2] nrf51822 bluetooth smart beacon kit. <http://bit.ly/1L34QCz>.
- [3] G. Aloï, G. Caliciuri, G. Fortino, and P. Pace. A smartphone-centric approach for integrating heterogeneous sensor networks. In *Proceedings of the 9th International Conference on Body Area Networks, BodyNets '14*, pages 247–252, ICST, Brussels, Belgium, Belgium, 2014. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
- [4] M. Blackstock and R. Lea. Iot interoperability: A hub-based approach. In *Internet of Things (IOT), 2014 International Conference on the*, pages 79–84, Oct 2014.
- [5] Bluvision. ibeek specification sheet.
- [6] F. Cristian. Probabilistic clock synchronization. *Distributed Computing*, 3(3):146–158.
- [7] A. K. Das, P. H. Pathak, C.-N. Chuah, and P. Mohapatra. Uncovering privacy leakage in ble network traffic of wearable fitness trackers. In *ACM HotMobile (the 17th International Workshop on Mobile Computing Systems and applications)*, 2016.
- [8] J. Elson and D. Estrin. Time synchronization for wireless sensor networks. In *IPDPS*, volume 1, page 186, 2001.
- [9] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 36(SI):147–163, Dec. 2002.
- [10] J. Elson and K. Römer. Wireless sensor networks: A new regime for time synchronization. *ACM SIGCOMM Computer Communication Review*, 33(1):149–154, 2003.
- [11] Estimote. Estimote.
- [12] G. Fleming. Data digest: Announcing our annual benchmark on the state of us consumers and technology in 2015, sep 2015.
- [13] H. Garcia-Molina. Elections in a distributed computing system. *IEEE Transactions on Computers*, C-31(1):48–59, Jan 1982.
- [14] C. Gomez, J. Oller, and J. Paradells. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, 12(9):11734–11753, 2012.
- [15] R. Gusella and S. Zatt. The accuracy of the clock synchronization achieved by tempo in berkeley unix 4.3 bsd. *Software Engineering, IEEE Transactions on*, 15(7):847–853, 1989.
- [16] IEEE. Ieee standard for a precision clock synchronization protocol for networked measurement and control systems. *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pages 1–269, July 2008.
- [17] A. Inc. Mac and ios. they really click.



- [18] G. Inc. Android wear.
- [19] M. Inc. Windows 10 family.
- [20] T. Instruments. Cc2540 usb evaluation module kit.
- [21] R. Kravets, G. S. Tuncay, and H. Sundaram. For your eyes only. In *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services*, MCS '15, pages 28–35, New York, NY, USA, 2015. ACM.
- [22] P. Krzyzanowski. Clock synchronization. *Lectures on distributed systems*, 2002, 2000.
- [23] L. Li, J. Zhang, Z. Peng, Y. Li, C. Gao, Y. Ji, R. Ye, N. D. Kim, Q. Zhong, Y. Yang, H. Fei, G. Ruan, and J. M. Tour. High-performance pseudocapacitive microsupercapacitors from laser-induced graphene. *Advanced Materials*, 28(5):838–845, 2016.
- [24] Q. Li and D. Rus. Global clock synchronization in sensor networks. *Computers, IEEE Transactions on*, 55(2):214–226, 2006.
- [25] D. L. Mills. Internet time synchronization: the network time protocol. *Communications, IEEE Transactions on*, 39(10):1482–1493, 1991.
- [26] C. Perera, P. Jayaraman, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Dynamic configuration of sensors using mobile sensor hub in internet of things paradigm. In *Intelligent Sensors, Sensor Networks and Information Processing, 2013 IEEE Eighth International Conference on*, pages 473–478, April 2013.
- [27] L. Schenato and G. Gamba. A distributed consensus protocol for clock synchronization in wireless sensor network. In *Decision and Control, 2007 46th IEEE Conference on*, pages 2289–2294. IEEE, 2007.
- [28] B. SIG. Bluetooth low energy standard.
- [29] F. Sivrikaya and B. Yener. Time synchronization in sensor networks: a survey. *Network, IEEE*, 18(4):45–50, 2004.
- [30] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie. Protocols for self-organization of a wireless sensor network. *IEEE personal communications*, 7(5):16–27, 2000.
- [31] Sony. Walkman.
- [32] S. Studio. Tiny ble, jun 2015.
- [33] K. Varshavskiy, A. F. Harris, III, and R. Kravets. Mihub: Wearable management for iot. In *Proceedings of the 2016 Workshop on Wearable Systems and Applications*, WearSys '16, pages 1–6, New York, NY, USA, 2016. ACM.
- [34] R. Want, T. Pering, G. Danneels, M. Kumar, M. Sundar, and J. Light. *UbiComp 2002: Ubiquitous Computing: 4th International Conference Göteborg, Sweden, September 29 – October 1, 2002 Proceedings*, chapter The Personal Server: Changing the Way We Think about Ubiquitous Computing, pages 194–209. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.
- [35] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, 2008.