**Implementing device tracking using BLE protocol stack**

Dhruv Gupta

Department of Computer Science

Georgia State University

Atlanta, Georgia

E-mail: dgupta3@student.gsu.edu

*Abstract*— Checking the presence of individuals, specifically department faculty, in a vicinity can render specific advantages to people and companies. This paper presents an implementation of checking device presence using the Bluetooth protocol stack in an effort to take advantage of high proliferation of Bluetooth enabled mobile devices.

***Keywords - Bluetooth; Raspberry pi; device tracking;***

1. **INTRODUCTION**

This project arose out of the need for location awareness of certain individuals. The scope was limited to checking if the specified individuals were in their office or not. An automated system would help reduce overhead for people to come and check the offices only to find an empty office space.

With indoor localization becoming a crucial component in many applications, there has been a lot of research in this field, though no standardized method exists. The selection of the method of localization is thus application dependent. An exhaustive list of some of these methods can be found in [1]. This paper will discuss why Bluetooth was chosen for this application, analyze the technology stack, present an implementation and compare this implementation with some other methods.

1. **PROBLEM STATEMENT**

The design constraints for the system are:

* Accurately update the system if there is change in device presence
* Changes should be tracked fast enough to keep information close to real time
* Track presence in a given office space with low impact of environmental factors such as doors being closed
* Should work for all the users defined in the system
* Passive system: should not require individuals to carry special sensing hardware
* Low cost

For this application, since mobile phones enjoy a high penetration rate in society today, and everyone carries their phone with them, it is ideal to rely on the phone to provide tracking information. This will circumvent the need for deploying of a complete system including transmitters and receivers ensuring a low cost system with passive sensing.

1. **BACKGROUND**

**Selecting a technology stack**

Relying on the mobile phone provides access to the technology stacks of the cell phone provider, Wi-Fi and Bluetooth. The location information provided by the cell phone towers is accurate for localization in the city however indoor location information cannot be gathered accurately with this method. It will require a phone app that relays information to a server all the time relying on the user’s data plan which cannot be relied upon for this system and raises privacy concerns since the location data will always be shared irrespective of the current location of the user.

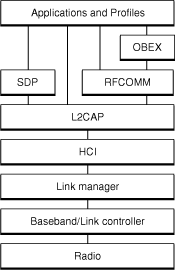
Another possible method of location determination is using the Wi-Fi. However, this would require the mobile device to be connected to the wireless network which cannot always be guaranteed. In addition, this requires access to the wireless network infrastructure in the university to run a localization algorithm as proposed in [2]. Using the Wi-Fi network is a drain on the battery even though there are best practices in software engineering that minimize the impact on battery life. The effects of constant Wi-Fi usage on battery life are detailed further in [3].

Bluetooth Low Energy (BLE) devices are proliferating even faster than Wi-Fi devices today with the rapid adoption of wearable devices. Personal tracking devices like Fitbit, and smart wearables like Apple Watch / Google Glass are on the uptick and support the Bluetooth Low Energy stack (Bluetooth 4.0). BLE devices offer a substantial advantage in terms of power consumption even in an always on state. [4] presents its findings of 18.5% battery life reduction when a mobile device’s BLE module is constantly used in a Denial of Service (DoS) attack. When tracking a phone and/or wearable for the same person the probability that someone left a device at home is lower and thus the system is more reliable. The architecture of BLE personal area networks (PANs) is dependent on sub-PANs or Piconets, this allows for the deployment of several host devices that can check for the presence of a client device ensuring that the entire target area is covered. Since high precision of location information is not needed, we only need to confirm presence near a specified location, these host devices can be used to update a list of devices in their vicinity.

**Bluetooth Stack**

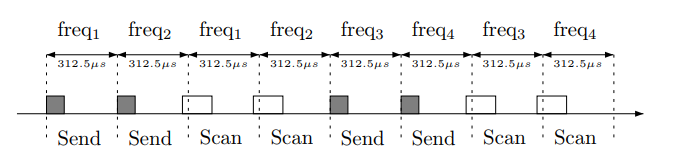
**Introduction to Bluetooth Protocol Stack**

Figure : Bluetooth Protocol Stack



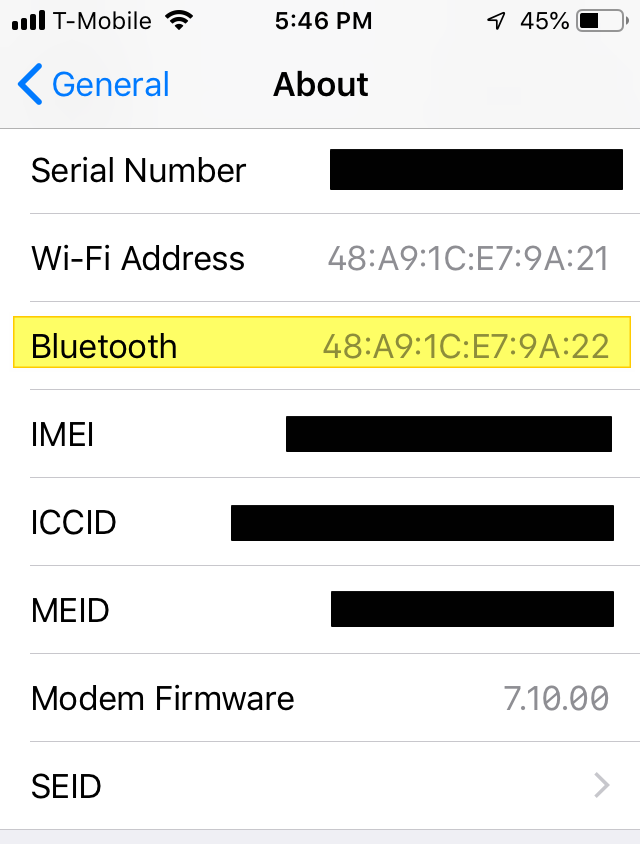
**Link Layer – Bluetooth Address**

Figure 3: Frequency division in the inquiry process



Every Bluetooth device has a unique identifier similar to the Media Access Control (MAC) address in the TCP/IP paradigm known as the device address or Bluetooth Address. This is a globally unique 48-bit address and the address spaces are managed by the IEEE Registration Authority.

Figure 2: Bluetooth Address of an Apple iPhone and Watch

This Bluetooth address is used by all the layers of the Bluetooth communication process as opposed to each layer having a different identifier. The device can also have a human readable name associated to it that is user defined, however that name is only used in the device discovery phase of the communication process. Once the devices have been paired, all communication between devices uses the Bluetooth address.

**Device Discovery**

Bluetooth devices organize into small PANs called Piconets. Each Piconet consists of a master and 7 slave devices where all devices follow a synchronized frequency hopping pattern. The Piconet formation consists of two steps: inquiry process and page process.

The inquiry process can be executed by an inquiring device (a host device looking for clients) or a scanning device (client device looking to be discovered). In the discovery mode, the inquiry device will broadcast an advertising packet on each of 32 radio channels. This advertising packet contains information about the peripheral’s name, Bluetooth address and primary function [5]. The entire list of functions can only be communicated after the pairing is complete and will be discussed under the Service Discovery Protocol section. The device sends the inquiry message on a channel followed by a second channel, it scans the channels in the same order for a response before moving to the next pair. The inquiry procedure does this for two sets of 16 channels each and sleeps between the two. The process continues until a specified number of devices are discovered or a timer runs out. [6]

Scanning devices that want to be discovered are in the inquiry scan sub-state and scan for the aforementioned inquiry packets. The channel hopping rate of these devices is much lower than that of the inquiry device in order to ensure that there is an overlap between the inquiry device and scanning device channels. On successfully receiving the inquiry, the scanning device will move to the inquiry response sub-state, wait for two time slots before sending an inquiry response. The response follows a random delay to minimize the chance of response collisions when multiple devices respond. The result of this protocol is that an inquiry must run for 10.24s to reliably detect all devices in range.

**Transport Layer Protocols**

RFCOMM

The RFCOMM protocol has the same service and reliability guarantees as TCP. It is used in the scenario where devices need a reliable way to transfer data point to point way for Bluetooth devices. The RFCOMM protocol emulates the serial cable line settings and status of an RS-232 serial port. RFCOMM connects to the lower layers of the Bluetooth protocol stack through the L2CAP layer and offers only 30 ports.

L2CAP

The Logical Link Control and Adaptation Protocol (L2CAP) is responsible for:

* Establishing Asynchronous, Connectionless links
* Multiplexing between higher layer protocols
* Repackaging data from upper layers to lower layers

L2CAP employs the concept of channels to track the source and destination of data packets. The L2CAP layer is a required part of every Bluetooth system. Ports are called *Protocol Service Multiplexers* and can take on odd-numbered values between 1 and 32767 and ports from 1-1023 are reserved [8]. The connectionless nature of the L2CAP protocol will be further used in this project.

Service Discovery Protocol (SDP)

In order to overcome the limitation to the number of custom applications that can be created using Bluetooth due to the number of ports available (~15,000 for L2CAP and 30 for RFCOMM), the SDP was added. Using this protocol allows a Bluetooth device to check what services another device offers over the Bluetooth connection, in the response the device will provide a list of the services but not access to them. An SDP client communicates with an SDP server using a reserved channel on an L2CAP link to find out what services are available. When the client finds the desired service, it requests a separate connection to use the service. The reserved channel is dedicated to SDP communication so that a device always knows how to connect to the SDP service on any other device. An SDP server maintains its own SDP database, which is a set of service records that describe the services the server offers. Along with information describing how a client can connect to the service, the service record contains the service’s UUID. Universally Unique Identifiers (UUIDs) describe the services that are offered and thus circumvent the need for a universal register of services [8].



Figure 4: Comparison of the Bluetooth Stack to Internet stack

**Python - PyBluez**

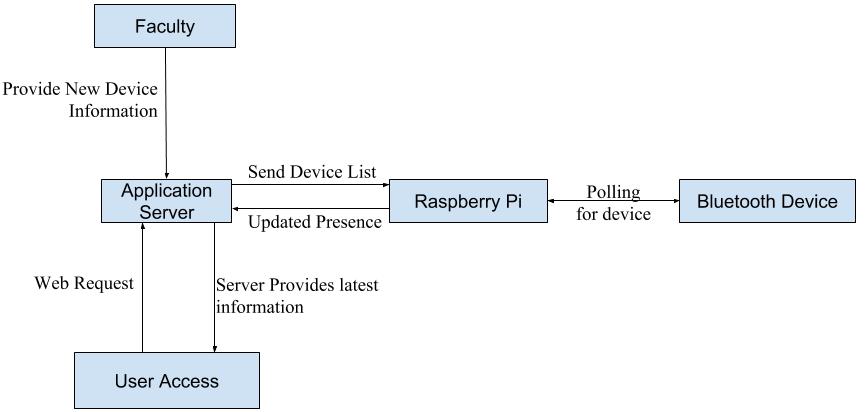
PyBluez is a Python extension module written in C that provides access to system Bluetooth resources in an object oriented, modular manner. It is written for the Windows XP (Microsoft Bluetooth stack) and GNU/Linux (BlueZ stack) [8].

1. **SOLUTION**

The proposed solution hinges on making use of the L2CAP Ping. The L2CAP layer is responsible for providing connectionless and connection oriented services to the upper layers of Bluetooth communication stack. The L2ping is similar to the ICMP ping in the internet protocol suite and can be used to check connectivity and round trip time to other Bluetooth devices.

Using the L2CAP layer, the handshaking process can be avoided, which is acceptable in this case because access to the upper layer services of the client devices is not needed, no data is being exchanged between the system and target device apart from round trip time. This also results in a better battery life expectancy for the devices being tracked as well as reduced security risk.

Figure 5: Architecture Diagram of Solution

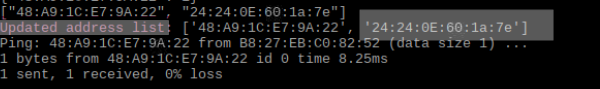


The Raspberry Pi acts as the sensor which will connect to the Bluetooth devices. Since the Raspberry Pi is small, affordable and runs on a Linux operating system distribution called Rasbian, it can be configured to provide high reliability and availability as a distributed sensor bed. The Pi runs a python script at a configured frequency to ping all the devices in its device list and updates their presence in a hash table. The ping is performed thrice every cycle, with the mode of the results being taken as the result to provide accurate results.

The application server hosts a multithreaded python application which listens for connection requests from a client Raspberry Pi. The client provides an updated table of device presence information. The server in return provides the list of devices that the server is keeping track of. If a device on the Pi’s list is not on the server provided list, it is removed from the hash table that the Pi maintains. The application server updates a database that it is maintaining. In this case the application server has a mySQL server. The SQL server also connects to a web application being served to the internet by an Apache Web Server. The up to date information can then be viewed on the this website.

1. **RESULTS**

Figure 10: Raspberry Pi Reciving updated device list



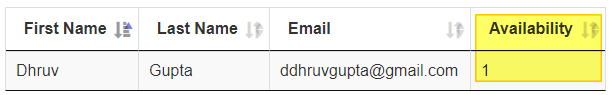
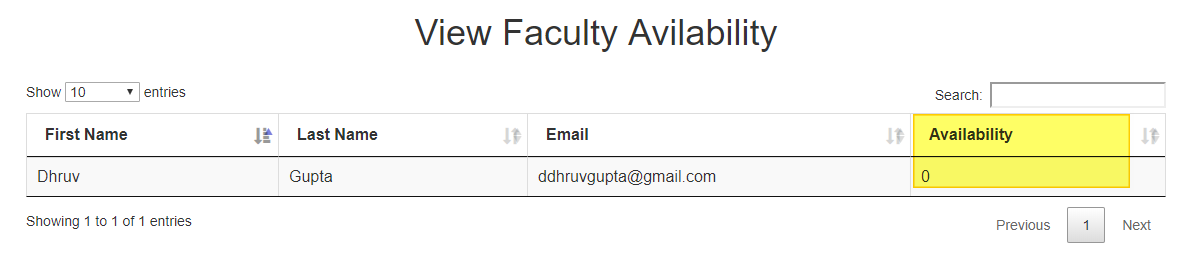


Figure : Faculty Availability Page of Web Application



The system allows users to view faculty from the web based interface described earlier. If the user is in the vicinity the Raspberry Pi can ping the user’s devices and send the updated results to the server.

Figure 11: Updated availability on the Web Application

In the scenario that was tested, there is one user with two devices. The application can update the presence of the user based on positive results from any of the tracked user’s devices, thus providing redundancy in case someone forgot a device at home.

Figure : Raspberry Pi pinging a device

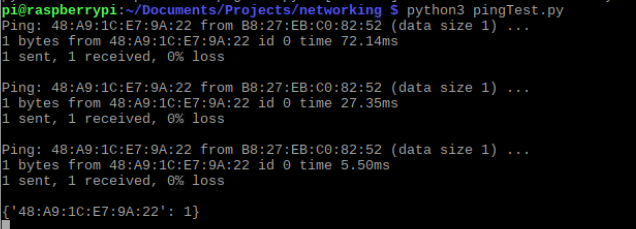


Figure 9: Server receives a connection request from a Pi and receives updated information

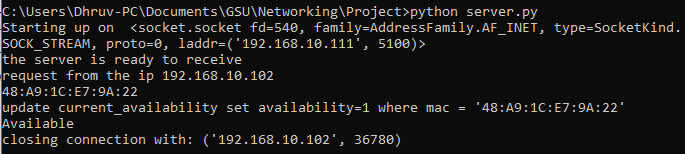
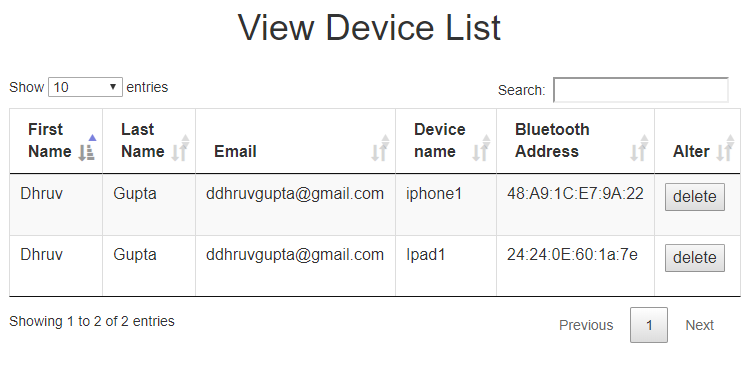


Figure 8: User with multiple devices



The Raspberry Pi receives the list of updated devices to check for before the connection is closed. The Pi will then complete the mandated time-period before continuing its search of the new devices.

1. **CONCLUSION**

The core objectives of the project were met by using the Bluetooth protocol. Specifically, by utilizing the L2CAP layer of the protocol stack. The system described in this paper is low cost, configurable with user input via the web application and can report the presence of an individual with high precision. The key features of this system are:

* Raspberry Pi updating a hash table with results of an L2Ping
* Server – Client communication between the Pi and application server
* Web application to present updated information
* Ability to remotely update the list of devices being tracked
* Ability to track person through multiple Bluetooth enabled devices
* Low battery consumption because an active paired Bluetooth connection is not used

1. **FUTURE WORK**

This project needs more features on the web application, such as ability to track multiple data sources when more than one Raspberry Pi is used, the system as it stands right now would have a conflict if two Raspberry Pis were to send the opposite result, logic needs to be added to handle this situation for scalability. The ability to update the number of times a device is polled in every round and the duration of the wait times between polling cycles should also be added.

The Raspberry Pi application should be setup using a CRON daemon so as improve the availability of the application in case the Pi looses power or is manually restarted, it will be able to automatically trigger the tracking application at a certain time.

Maintaining the privacy of the users being tracked is very important and thus a data retention policy needs to be explored. The security aspect of the Bluetooth protocol should also be studied in detail, since the standard is updated frequently it is important to get an understanding of the direction in which security is moving.

1. **REFERENCES**

[1] Gabriel Deak, Kevin Curran, Joan Condell, A survey of active and passive indoor localisation systems, Computer Communications, Volume 35, Issue 16, 2012, Pages 1939-1954, ISSN 0140-3664

[2] Youssef, M., Agrawala, A.: The Horus WLAN location determination system. In: MobiSys 2005: Proceedings of the 3rd international conference on Mobile systems, applications, and services (June 2005)

[3] Anand, A., Manikopoulos, C., Jones, Q., & Borcea, C. (2007, June). A quantitative analysis of power consumption for location-aware applications on smart phones. In *Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on* (pp. 1986-1991). IEEE.

[4] Moyers, B. R., Dunning, J. P., Marchany, R. C., & Tront, J. G. (2010, January). Effects of wi-fi and bluetooth battery exhaustion attacks on mobile devices. In *System Sciences (HICSS), 2010 43rd Hawaii International Conference on* (pp. 1-9). IEEE.

[5] “Core Bluetooth Programming Guide, Apple, 18 Sept. 2013, https://*developer.apple.com/library/archive/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth\_concepts/CoreBluetoothOverview/CoreBluetoothOverview.html*

[6] Duflot, M., Kwiatkowska, M., Norman, G., & Parker, D. (2006). A formal analysis of Bluetooth device discovery. *International journal on software tools for technology transfer*, *8*(6), 621-632.

[7] Scarfone, K. A., Padgette, J., & Chen, L. (2012). Guide to Bluetooth security (No. Special Publication (NIST SP)-800-121 Rev 1).

[8] Huang, A. S., & Rudolph, L. (2007). Bluetooth essentials for programmers. Cambridge University Press.

[9] Hay, S., & Harle, R. (2009, May). Bluetooth tracking without discoverability. In International Symposium on Location-and Context-Awareness (pp. 120-137). Springer, Berlin, Heidelberg.

[10] Caldwell, L., Ekerfelt, S., Hornung, A., & Wu, J. Y. (2006). The art of Bluedentistry: Current security and privacy issues with Bluetooth devices. unpublished, Dec, 13.

[11] Celosia, G., & Cunche, M. (2018, June). Detecting smartphone state changes through a Bluetooth based timing attack. In Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks (pp. 154-159). ACM.