Bluetooth Triangulator

Varun Almaula
valmaula@cs.ucsd.edu

Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093

David Cheng
ddcheng@cs.ucsd.edu

Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093

Abstract*—The Bluetooth wireless RF standard has been established as one of the most prevalent protocols implemented for wireless personal area networks. Its relatively low power consumption compared to implementations such as 802.11 make it suited for a variety of short range applications suitable for small embedded devices. The cell phone market has been a huge catalyst in making the Bluetooth protocol mainstream, as the majority of mobile phones now feature Bluetooth connectivity.*

As Bluetooth enabled devices become a part of our wireless world, there is increased interest in locating and communicating with other Bluetooth devices within a local network. While basic discovery of surrounding Bluetooth devices is already possible, rich software applications could better use a Bluetooth service that was able to triangulate the location of surrounding Bluetooth devices.

This paper will describe the Bluetooth Triangulator, an experiment combining hardware and software design to allow Bluetooth devices to communicate with each other and determine the position of other Bluetooth devices using signal strength readings. This type of service is useful not only as a device location utility, but can also be the foundation for rich software applications such as security or social networking applications.

Index Terms*— Bluetooth, BlueZ, cross-compile, dbus, HCI Inquiry, Intel XScale PXA27x DVK, Linux, Bluetooth Stack, Triangulate, location, locator, Triangulator, USB Bluetooth Dongle.*

I. INTRODUCTION

Bluetooth is an industrial standard for wireless personal area networks. It is primarily designed for low power consumption and short range operations among several mobile and embedded devices. Since it is an RF based communication protocol, line of sight is not required between the communicating devices. Bluetooth provides connection management and data exchange among devices that are within close proximity and do not require high bandwidth data links. Unlike Ethernet based WiFi networks, discovery of surrounding Bluetooth devices and available services is simplified, making the process of establishing connections a streamlined paired device setup.

In recent years, the mobile phone market has increasingly used Bluetooth as the preferred method of device communication, data exchange, and accessory pairing. Many PC accessories including mice, keyboards, headsets, and printers also employ the Bluetooth standard for wireless communication. A wide array of Bluetooth profiles provides the development framework for these rich hardware and software applications. With an increasing number of Bluetooth enabled devices becoming common place in wireless networks, device discovery and connection is extremely important.

The existing Bluetooth stack provides a simple interface for users to discover surrounding Bluetooth devices and their available services. However, the interface does not provide any relevant distance information to these surrounding devices. Many rich hardware and software applications would be able to utilize this extra distance information. For example, social networking applications and other ubiquitous computing applications could use a locator utility to provide spatial maps of other users within a given network for social communication. In security conscious networks, a spatial device mapping can be used to locate intruders or unidentified devices in a corporate wireless network. The Bluetooth Triangulator utility is designed provide location information so that these high level applications can be implemented.

II. Background And Current State of The Art

A. Background

As in other wireless locator schemes, the transmission radio signal strength can be used to infer the location of a mobile node. Using multiple readings of this signal strength across three or more different points in space can be used to predict the exact location of the mobile node. For the Bluetooth protocol, the Received Signal Strength Indication (RSSI) is a measurement of the received radio signal strength. Normally, for a wireless networking device, RSSI is used as a reference to determine the most optimal radio energy for certain link. This value (in dB) can be obtained when inquiring a surrounding device or communicating with an already connected device.

B. Current State of the Art

Location tracking has been attempted by several researchers to date. Huang's Bluetooth locator system deployed simple USB Bluetooth dongles on PC's throughout a building [1]. In this implementation mobile devices scan the USB Bluetooth beacons to get distance information. Through experimentation, Huang realized RSSI alone is not an accurate candidate for distance measurement. Huang found that the link quality was more apt to make distance calculations with. Nevertheless, Huang found that location tracking was best suited for small rooms with minimal obstructions and could provide room-level granularity.

Feldman *et. al* created a very precise mapping of the test room to a spatial grid [2]. Using a comprehensive collection of RSSI data across the room, a simple distance-RSSI mathematical mapping was created. Three Bluetooth access points were used and a precision of about two meters was possible when the access points and mobile devices were separated by no more than 8 meters. This study showed that perhaps RSSI can be used to indicated distance if the room is well known and empirical data is used to create a precise mapping.

Halberg *et. al* also found that directly using RSSI for distance calculation was problematic and extensive research about the room and environment was very necessary [3]. The results showed that some hard-coding for the room geometry and Bluetooth access point locations was beneficial to attain accuracy up to 10m distances.

Advanced work has shown that multiple high-gain antennas on a Bluetooth access point with variable attenuators can produce more accurate results than simple RSSI measurements [4]. These systems take several varied and redundant measurements in attempts to provide more precise results. Bandara *et al* used this advanced hardware in a test room (a “smart space”) where the antennas were placed strategically and the access point recorded readings from Bluetooth enabled devices. The access point would connect to a device and, for each of its antennas, record RSSI readings while varying the attenuation levels on the antenna. The results are several redundant readings over multiple transmit power levels for increased accuracy. The advanced Bluetooth access point in this setup was well suited to overcome the often non-linear correlation between RSSI values and distance.

With limited time and device resources, our implementation will use elementary USB Bluetooth dongles to experiment with Bluetooth Triangulation. USB Bluetooth dongles are easy to obtain and install, cost effective, and can be repositioned quickly.

Also, it is clear that RSSI measurements are not perfect and many previous projects either mask these inaccuracies with redundancy and/or tailor a system to the specific environment in which it will be deployed.

C. Known Challenges and Limitations

The Bluetooth discovery process does not guarantee to find all the surrounding Bluetooth devices even if the device is in range. According to the published Bluetooth specification, Bluetooth uses frequency hopping and switches channels every frame [5-8]. To discover surrounding Bluetooth devices, the discovering device spreads out a discover message, on multiple channels over the discovery period. While the discovery period normally lasts around 10 seconds by default, it is possible for a nearby device to miss the discovering message either by listening on a different channel or by simply not listening at the correct time. As a result, the nearby device would not respond to the discovering message. To tackle this issue, we can perform the discovery multiple times or increase the discovery period to reduce the possibility that any Bluetooth device in range is left undiscovered.

Another challenge is rooted in the inaccuracies associated with RSSI readings, RSSI measurements can fluctuate greatly for a given mobile device even when it is stationary. Environmental conditions and physical objects (both temporary and permanent) can introduce interference for the RF based protocol and can lead to misleading RSSI results. To complicate matters, HW devices may vary on the RSSI inquiry mode implementation and the accuracy of RSSI readings. The RSSI-distance mapping is not necessarily linear and, for a variety of reasons, may be almost non-continuous, similar to a step function.

III. Implementation

A. Project Goal

The Bluetooth Triangulator can dynamically measure and report the distances to surrounding Bluetooth devices.

B. Project Description

The physical setup will consist of 3 master nodes, each collecting RSSI measurements from a slave mobile device. One of the 3 master nodes acts as a server that will collect measurements from the two client master nodes. The server master node will be running on an Intel XScale PXA27x developer board, while the two client master nodes will both run on a common x86 Dell desktop PC. All machines use the Linux 2.6.9 kernel which has been compiled with the optional Bluetooth networking stack. In addition, all master nodes rely on the BlueZ package for Bluetooth libraries and utilities.

Before sending a message packet to the server master node, a client master node will collect the RSSI readings four times for each discovered Bluetooth device. Then, the slave will pack the message packet as shown below and send it to the host for further processing. The reason to have multiple RSSI readings is two fold. First of all, RSSI reading tends to fluctuate greatly. Multiple sampling points can be used to normalize the fluctuation. Secondly, instead of sending a message packet each time a RSSI read is executed, one packet containing multiple RSSI readings for all known devices greatly reduces network overhead.

The server master node will also perform the RSSI readings for all devices it can discover while also waiting for incoming message packets from the two client master nodes. Upon receiving incoming packets, the server node will process the information and look for RSSI readings for matching devices it already knows about. For devices with RSSI readings by all 3 master nodes, the server will proceed to calculate the approximate distance to the mobile device and display the results.

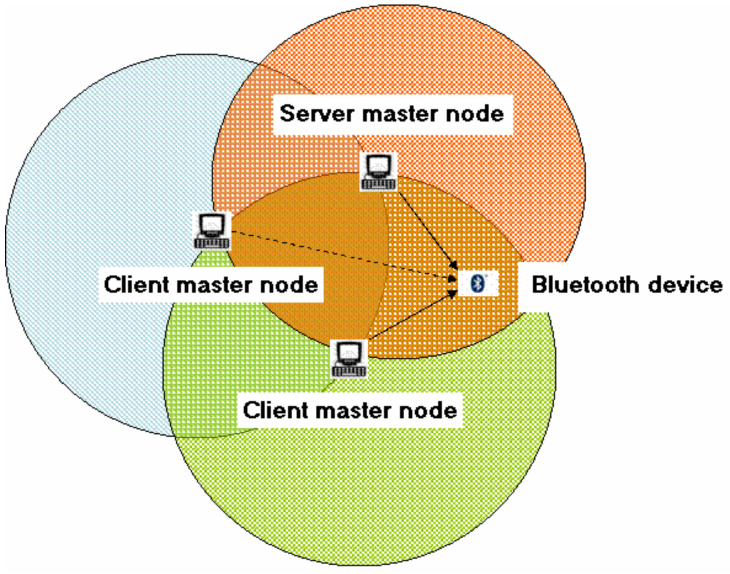
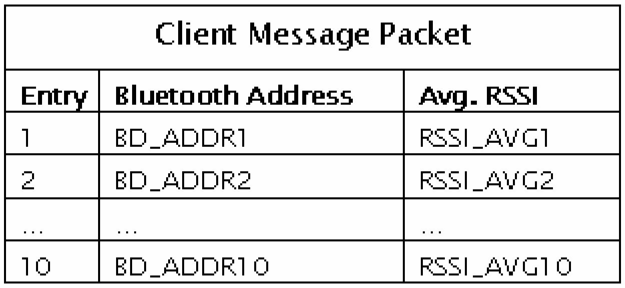


Figure 1: Diagram illustrating master node layout and mobile devices.



Client Message Packet		
Entry	Bluetooth Address	Avg. RSSI
1	BD_ADDR1	RSSI_AVG1
2	BD_ADDR2	RSSI_AVG2
…	…	…
10	BD_ADDR10	RSSI_AVG10

Figure 2: Basic packet format sent from client master nodes to server master node.

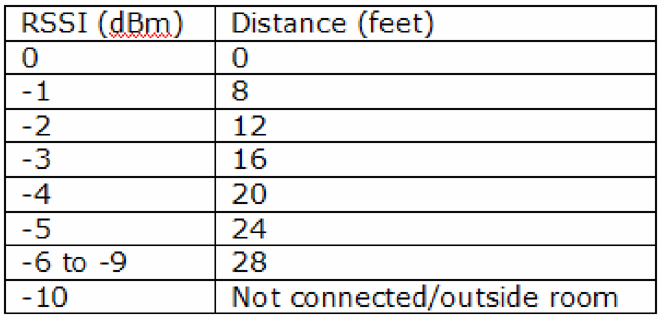
C. Results

The first step was to re-compile the Linux kernel with the Bluetooth networking stack for the server master node because its lightweight Linux OS did not have Bluetooth support. Configuration and cross-compilation for the target architecture was relatively simple using ‘menu-config’ and re-flashing the Intel PXA27x board was straightforward. The client master nodes were already running on x86 machines with a recent Red Hat Enterprise release that included Bluetooth networking support.

Cross-compiling the BlueZ Bluetooth libraries and utilities for the master server node target architecture was not as trouble free. A significant amount of time was invested in trying to port the most recent version of BlueZ (3.3.7) for our target architecture, but was ultimately unsuccessful. An earlier version of BlueZ (2.25) proved to be far more cross-compile ‘friendly’ for our target architecture (see next section for more information).

Once BlueZ was successfully configured, L2CAP was used to exchange information between master nodes. Each master node was able to collect discovery results, connect to discovered devices, and obtain redundant RSSI readings. For the client master nodes, this information was packed and sent to the server master node for processing.

Once received by the server master node, the information was processed and distances inferred for common Bluetooth devices. Using several empirical data points a simple mapping between RSSI and distance was established for the test room:



RSSI (dBm)	Distance (feet)
0	0
-1	8
-2	12
-3	16
-4	20
-5	24
-6 to -9	28
-10	Not connected/outside room

Figure 3: RSSI->Distance mapping.

By placing the dongles in an ‘L’ shaped triangle, a simple trilateration scheme was used to predict the co-ordinates and distance from the origin [9]. While RSSI did not have a linear mapping to distance, enough empirical data was collected to make a direct mapping. This resulted in accuracy within +/- 5-10 feet. RSSI can be used to indicate what ‘zone’ a device is in, rather than pinpointing a single co-ordinate. In our test setup, we found trilateration to be within +/- 5 feet of actual distance 90% of the time when measuring distances of 17 feet, and to be within +/- 5 feet of actual distance 80% of the time when measuring distances of 30 feet.

D. *Project Summary and Outlook*

There were a handful of issues we encountered in this project, described below.

Tool chain issues

While the latest version of the BlueZ (3.3.7) libraries were able to be compiled for our arm-linux target, the application utilities (the useful tools to develop higher level Bluetooth applications) had a dependency on a separate installation known as ‘dbus’. After countless hours of trial and error and reading online resources, we realized cross compiling ‘dbus’ was too unreliable. Further investigation uncovered that the BlueZ project required dbus as part of the installation since version 3.x. In an attempt to minimize risk and stick to our original schedule as much as possible, we chose to install an older version of BlueZ (2.25). This version, while not as mature or featured as the current version, implements all functionality needed for our project and, most importantly, does not require dbus allowing for simple cross-compilation.

Hardware and Bluez issues

The key information required for the Bluetooth Triangulator to determine the distance between Bluetooth devices is the RSSI reading. There are two methods to collect this piece of information [5-8]. One is by creating a connection to another Bluetooth device and then using a HCI command to read the link’s RSSI. The other method uses the HCI_Inquiry command to gather RSSI information while discovering surrounding Bluetooth devices. The standard HCI_Inquiry command will not have the RSSI information returned. Instead, the inquiry mode must be changed to ‘inquiry with RSSI mode’ to have this piece of information returned during the discovery process. Ideally, the second method is more desirable as it does not require a connection to be established, incurring far less overhead in terms of communication, resource consumption, and time. However, two issues prevented us from proceeding with this route. First, the D-Link dongles used did not support the HCI command to change the inquiry mode. This was a costly discovery since we did not realize that the hardware was the culprit and we spent a significant amount of time debugging the code. We finally dug deep enough to come to the conclusion that the command was unknown to the hardware device (the command is not supported by the D-Link dongle). After borrowing another newer dongle (supporting Bluetooth version 2.0) we were able to verify that it supported the HCI command to change the inquiry mode. However, the events coming back did not reflect the changes made to the inquiry mode setting; the events were still in the standard inquiry format, which does not contain the RSSI information. Cutting our losses at this point, we went forward with the first approach which required establishing a connection to get RSSI information. One drawback of our current implementation is that not all mobile devices will grant a connection request and the system will not be able to collect RSSI readings for the device. With more advanced hardware and some more tinkering, a future implementation should be able to take advantage of the ‘inquiry with RSSI’ mode.

RSSI issues

As previously discussed, the accuracy of RSSI measurements are varied due to implementation, device, and environmental limitations. With a more advanced Bluetooth AP, such as the one implemented by *Bandara et. al*, these issues can be masked by taking redundant RSSI readings across a spectrum of transmit power levels using attenuators.

Other Issues & Outlook

For future implementations, a variety of outstanding issues will need to be addressed. The update frequency (how often the master nodes should poll for new devices and collect RSSI readings) has an acute accuracy vs. overhead tradeoff. This will also determine how well the system handles moving mobile devices and stale entries in each of the master node lists. Scalability is also an issue as environments with a large number of Bluetooth devices will require the faster method of RSSI inquiry to avoid the connection overhead.

There are some broader issues to consider as well. Bluetooth is distance limited based on class (anywhere from 1 to 100 meters). Distance accuracy should only be expected for some small fraction of this range. Creating more wireless friendly environments can help mitigate issues relating to RF interference. Standardizing RSSI hardware implementations and the expected granularity/accuracy would also benefit Bluetooth location schemes. Perhaps, the next iteration of the Bluetooth protocol could provide some basic interface for positioning information and exchange.

References

[1] Albert Huang. “The Use of Bluetooth in Linux and Location Aware Computing” Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2005.

[2] Silke Feldmann, Kyandoghere Kyamakya, Ana Zapater, Zighuo Lue. “An indoor Bluetooth-based positioning system: concept, Implementation and experimental evaluation” Institute of Communications Engineering.

[3] J. Halberg, M. Nilsson, and K. Synnes ”Positioning with Bluetooth” International Con-ference on Telecommunications, 2003.

[4] Udana Bandara, Mikio Hasegawa, Masugi Inoue, Hiroyuki Morikawa, Tomonori Aoyama. “Design and Implementation of a Bluetooth Signal Strength Based Location Sensing System” IEEE, 2004.

[5] Bluetooth SIG. “Specification of the Bluetooth System v2.0 Volume 0, Master Table of Contents and Compliance Requirements” 2004.

[6] Bluetooth SIG. “Specification of the Bluetooth System v2.0 Volume 1, Architecture and Terminology Overview” 2004.

[7] Bluetooth SIG. “Specification of the Bluetooth System v2.0 Volume 2, Core System Package (Controller Volume)” 2004.

[8] Bluetooth SIG. “Specification of the Bluetooth System v2.0 Volume 3, Core System Package (Host Volume)” 2004.

[9] William M. Spears, Jerry C. Hamann, Paul M. Maxim, Thomas Kunkel, Rodney Heil, Dimitri Zarzhitsky, Diana F. Spears, and Christer Karlsson. “Where Are You?” Computer Science Department, University of Wyoming, 2006.