WireLost: Ad-Hoc Wireless Tracking System

Nick Orr, Brendan Sileo, and Mira Yun
Department of Computer Science and Networking
Wentworth Institute of Technology
Boston, MA 02115, USA
{orrn, sileob, yunm}@wit.edu

***Abstract*— With the ubiquity of smart devices, people need a system to interact with and track those devices and objects. Homes, parking lots, retirement facilities, and augmented reality applications have various benefits from utilization of a wireless tracking system. In this paper, we propose a Wi-Fi based tracking system, *WireLost,* which provides an easy and simple triangulation solution with off-the-shelf and low-cost hardware. *WireLost* provides a webpage showing a map of all devices.**

Keywords—Object Tracking; Wi-Fi Positioning, Ad-hoc Wireless Mesh Network;

I. INTRODUCTION

People have frequent interactions with several different objects and items at home or in the office, but can easily and frequently misplace them. As Internet of Things (IoT) brings more opportunities to work with smart devices and objects, we clearly see the needs of tracking all those objects [1-3]. Homes, parking lots, retirement facilities, and augmented reality applications have various benefits from utilization of a wireless tracking system. For example, people always look for keys, wallets, and cell phones whenever they need to change their locations, or one of several remotes for an entertainment system. People can easily misplace these small items. To avoid this frustration, we propose *WireLost* to make the locating of these small objects far simpler and painless by being able to locate them through a webpage with a map of all their devices.

Having this framework in mind, *WireLost* is designed to provide an easy and simple triangulation solution with off-the-shelf and low cost hardware. In addition, *WireLost* is easy to set up for a user. In Section II, we provide our choices for wireless method, hardware, software, and tracking methods. Section III present the system architecture and implementation details of *WireLost.* Finally, Section IV summarizes the paper and outlines ideas for future work.

II. TECHNOLOGY CHOICES

In this section, we provide our choices of wireless technologies, hardware, software, and methods of tracking for *WireLost*.

A. Wireless Technology

Alongside a wide variety of wireless technologies to choose from, Bluetooth, RFID, and Wi-Fi technologies could be considered as possible wireless methods for tracking the objects. Since RFID devices do not require an additional power source, we considered RFID as the most practical and efficient method. However, RFID long-range readers are costly making it a poor choice for a low-cost system. Bluetooth was considered as the next choice, allowing for a similar setup to the eventual Wi-Fi ad hoc choice that was later decided on. It allowed for data transfers between devices, was easy to setup up initially for a couple devices but was found to be very inaccurate for distances and hard to set up for large scale systems [4].

Wi-Fi was the next choice, leading us specifically to an ad hoc network, and was found to work very well. A peer-to-peer network allows all devices to communicate with each other, without relying on an outside access point or switch. This communication is what allows the nodes to check transfer times. In addition, the network was very easy to set up, allowing for largescale deployment, limiting the overhead required to get the system running.

B. Hardware Options

WireLost nodes will be a semi-permanent system, similar to a smart speaker style of device, where it will likely remain in one location, and always remain plugged in. With the choice of ad hoc in mind, we looked to *Arduino*s and *Raspberry Pi*s. We found that the wireless chip in the *Raspberry Pi 3* (*Cypress CYW43438*) is capable of ad hoc and would allow for the desired network setup [5]. In initial testing, two

*Raspberry Pi 3*s were used for checking practicality of the setup and transferring speeds. However, it could be too expensive for the ideal cheap solution. The *Pi Zero* was next looked into as it contains the same wireless chip, allowing it to also work in ad hoc, has a smaller form factor making it less noticeable in a living space, and requires less power which is optimal if it would be plugged in long-term.

C. Software Options

When the hardware and wireless option choices were determined, there was little option for what software could be used on the *Raspberry Pi Zero*. The *Raspberry Pi Zero* can use *Raspbian*, and the version *'Jessie'* was chosen because of its compatibility with the *Raspberry Pi* ad hoc tool, named *'RPiAdHocWiFi'*, created by Simon Levy. The tool is publicly available on *GitHub* and forces the *Raspberry Pi* into an ad hoc mode on boot, allowing us to simplify the setup experience for the user. *Python 3* would be our language of choice for all code created for its ease of use, adaptability if users want to make changes, and also because of its compatibility with a long list of libraries.

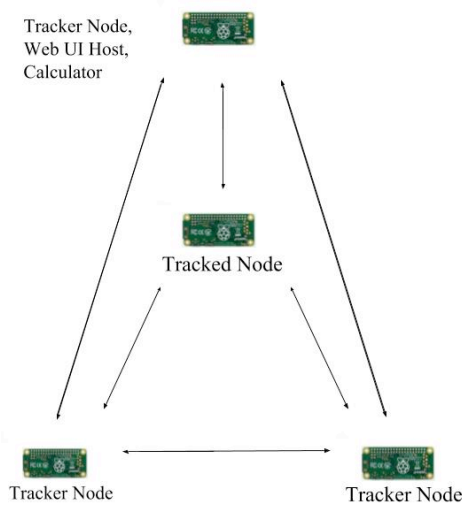

Fig. 1. Wireless Tracking System Example

D. Similar Solutions

While looking for ideas on how to implement the method of tracking, the initial ideas came to solutions similar to GPS tracking, cellular towers triangulation, and systems similar to the 'Tile' [6]. With GPS tracking being somewhat impractical because of the desire to use the system mostly indoors, cellular tower triangulation was looked to next. This option seemed the most practical, as there are methods of tracking with only two or three towers, adding precision with additional towers. The Tile Tracker was looked at as well, but relied on using semi-precise Bluetooth predictions, and then an auditory signal to the user to precisely locate it. The cellular tower option seemed the most practical and was looked at more in depth.

E. Method of Tracking

With the wireless technology, hardware, software, and a general idea of the method chosen, the next step was to create an effective way to track. The initial desired setup had three tracker nodes all talking to each other, and then one object being tracked as shown in Figure 1. With ease of use in mind, we wanted everything to set up for the user as easily and automatically as possible, but unless the position of the nodes were known on an X, Y axis, the system would not be able to start on its own. This led us to look for alternative methods that would still be easy to setup but would not require any loss in functionality.

Our final method of tracking took advantage of the idea that if two nodes were considered to be on the X-axis, (one at (0,0) and one at (distance from (0,0),0), the angles and position of the third triangle could be found. As shown in Figure 2, this would be accomplishing the same goal as before, but with only two trackers, and one tracked object, requiring three total objects.

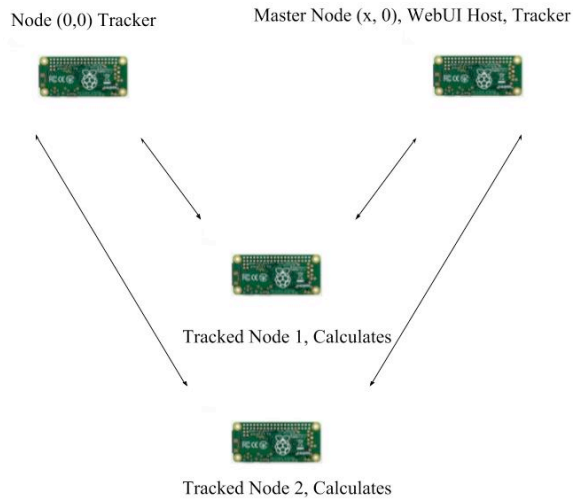

Fig. 2. *WireLost* System Diagram

III. Design and Implementation

With the mechanics of the system determined, the tracking system needed to be designed, the math needed to be put into a reusable formula and implemented into code.

A. *Tracking Technology*

The project has two main components, the first being the tracking technology. The initial step in tracking our target objects is to find their distance from the tracker nodes. This is done by sending a set amount of data from the object to each tracker. Based on the transfer speed it is possible to calculate the rough distance from the object to the tracker. With several devices checking for distance in an area, they will be able to calculate the location of tracked objects.

B. *Method of Displaying Info*

With a system in place for tracking, an easy to use solution for visually seeing the information would need to be created. Initially the first node would be static to the display each time the system was used, but the remaining nodes and tracked objects would be dynamic. As shown in Figure 3, the user interface (UI) now displays the tracking nodes at the top, on the same x-axis, and the tracked node visible at the bottom with distances. This UI system would be contained on one of the nodes, perpetuating the ease of use of the entire system.

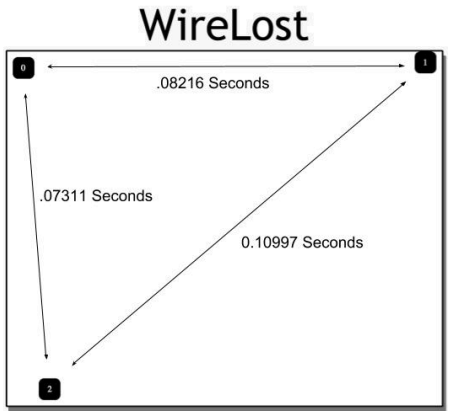

Fig. 3. *WireLost* WebUI

C. *Trackers and Triangulation*

The only information the system needs in order to calculate the location of an object is the location of the two trackers (which was stated before to be (0,0), and (distance to (0,0), 0)) and the distance between each tracker and the target object [7]. This allows the system to create a triangle between the three nodes. Knowing all three sides of that triangle allows the use of the Law of Cosines to calculate each of the three angles in the triangle as well, as seen on lines 15 through 17 of Figure 4. The next step is to calculate the slope of each of the three lines of the triangle. It is known that the line between the two trackers will always have a slope of 0, because they both lie on the X-axis. The other two slopes can be calculated by using the tangent of their respective angle. Finally, knowing the slopes of each line intersecting with the tracked object, as well as a point in each of those lines allows the formation of an equation using point-slope form for each line. Solving this equation for X reveals the X coordinate of the target object and plugging that back in allows one to find the Y coordinate as well, as shown on lines 26 through 29 of Figure 4.

```
14  def triangulate(AB, BC, AC):
15          A = (math.acos(((AC**2 + AB**2) - BC**2)/(2*AC*AB)))
16          B = (math.acos(((AB**2 + BC**2) - AC**2)/(2*AB*BC)))
17          C = math.radians(180)-(A+B)
18
19          a_loc = (0,0)
20          c_loc = (AC,0)
21
22          ac_slope = 0
23          bc_slope = math.tan(C)
24          ab_slope = -1 * math.tan(A)
25
26          x = sympy.Symbol('x')
27          Bx = sympy.solve(bc_slope*(x-AC) - ab_slope*(x))
28          Bx = Bx[0]
29          By = ab_slope * Bx
30
31          return (Bx, By)
32
```

Fig. 4. Triangulation code for calculating locations

D. *Testing*

In initial testing of the data speeds, it was found that there is a relatively consistent data transfer speed across distances, meaning that there would not need to be any sort of scaling-to-distance factor applied. While there is not a 1:1 ratio of distance to time seen in Figure 5, it can be assumed that this is within the expected range of error with low end wireless chips in the hardware, and the ability for Wi-Fi to not have significant interference from other nearby devices.

Fig. 5. Average Transfer Time and Distance

When displaying information to the WebUI, one significant method of averaging was taken where from a sample of 25 time calculations collected, the values from the lower to upper quartile were taken, and averaged to find an accurate value. This method can be seen in Figure 6.

```
32   def combine(lov):
33           lov.sort()
34           z = int((round(len(lov)*.25)) - 1)
35
36           for i in range (0,(z)):
37               del lov[i]
38               del lov[(len(lov)-1)-i]
39
40           average = float(sum(lov)/len(lov))
41           return average
```

Fig. 6. Averaging Code for Collected Values

This would serve as a safeguard against outliers which could significantly impact the averaging of the data. Additionally, some protections were added such as not allowing values to undercut or surpass the boundaries of the two nodes on the x-axis, and with these two methods implemented, accuracy was very consistent.

IV. Future Improvements

While for the price and with consideration of the hardware and algorithms implemented, *WireLost* provides a simple and easy solution to tracking objects. The biggest benefit to be gained from further work on *WireLost* is improved accuracy. In this paper, we only focused on developing the initial prototype. By adding accuracy analysis and developing new algorithms based on that, *WireLost* can be more practical.

References

[1] Locke, P. (2012). *Cell Tower Triangulation – How it Works*. [online] Wrongful Convictions Blog. Available at: https://wrongfulconvictionsblog.org/2012/06/01/cell-tower-triangulation-how-it-works/ [Accessed 3 Aug. 2018].

[2] Mathematics Stack Exchange. (2013). *Determine third point of triangle when two points and all sides are known?*. [online] Available at: https://math.stackexchange.com/questions/543961/determine-third-point-of-triangle-when-two-points-and-all-sides-are-known [Accessed 3 Aug. 2018].

[3] Mathsisfun.com. (2017). *Solving SSS Triangles*. [online] Available at: https://www.mathsisfun.com/algebra/trig-solving-sss-triangles.html [Accessed 3 Aug. 2018].

[4] Sirton, G. (2012). *Options for short range distance determination between two objects*. [online] Electrical Engineering Stack Exchange. Available at: https://electronics.stackexchange.com/questions/33110/options-for-short-range-distance-determination-between-two-objects [Accessed 3 Aug. 2018].

[5] Raspberry Pi Stack Exchange. (2018). *Ad Hoc setup in RPi 3*. [online] Available at: https://raspberrypi.stackexchange.com/questions/49660/ad-hoc-setup-in-rpi-3 [Accessed 3 Aug. 2018].

[6] Tile: https://www.thetileapp.com/en-us/

[7] Misra, S. (2015). Mathematics: How do I find coordinates of the third point in ABC triangle, knowing coordinates of A, B and angles at them? - Quora. [online] Quora.com. Available at: https://www.quora.com/Mathematics-How-do-I-find-coordinates-of-the-third-point-in-ABC-triangle-knowing-coordinates-of-A-B-and-angles-at-them [Accessed 3 Aug. 2018].