Multilateration-based localization of devices within a LoRaWAN network

Daniel Saul
University College London

Supervisor: Dr Miguel Rio

March 23, 2017

Abstract

The Internet of Things is growing at a tremendous rate and with it the need for location estimates of connected devices, particularly in Low Power Wide Area Networks which provide long range communications to low cost and low power devices. Ensuring low power consumption of such devices necessitates alternatives to using a GNSS system such as GPS. This report therefore looks at positioning techniques that might be used with one such network, LoRaWAN. Hardware is built that includes a GPS receiver and LoRa transceiver to gather measurements and ultimately used to evaluate the feasibility of using multilateration to find position estimates of nodes from the Time Difference of Arrival of signals at multiple gateways. Results suggest that with current network infrastructure, reliable and accurate positions cannot be obtained but coarse estimates at the kilometre level can, with higher accuracies achieved by oversampling data from a stationary node. A number of suggestions are made for the increase in accuracy of such results in future studies, with the main being the improvement of the network gateway hardware.

Contents

Chapter 1

Introduction

The Internet of Things (IoT) is currently one of the most hyped of technology areas but the term was coined as long ago as 1999 by Kevin Ashton. Back then, all data on the internet was generated or uploaded by humans, whether text, images or information. But Ashton envisioned a future of computer systems generating and collating data on their own, with little to no input from us [1]. Today we are in the situation where 6.4 billion IoT devices were in use globally during 2016 [2] with estimates of up to 30 billion devices by 2020, compared to world population estimates of just 7.7 billion people [3]. IoT is likely to impact up to 6% of the global economy, driving down costs, increasing productivity and generating revenues. It has been claimed that the Internet of Things is 'critical for human progression' and providing higher quality of life around the world [4].

Consumers may find themselves with internet connected cars and appliances, app controlled security systems and smart metering on their energy supplies - but the Internet of Things is far broader than just the home. In agriculture, irrigation systems, environmental sensors and even animals will all become interconnected. In smart cities, parking, traffic and refuse will all be monitored. In the environment, sensors will be networked to provide warning for forest fires, earthquakes and floods. In industry, equipment will have connected sensors and workers will be tracked [5].

Many technologies comprise the Internet of Things, encompassing a huge range of use cases, and include RFID, Bluetooth, Zigbee, WiFi and cellular networks, amongst others [6]. But the only one of these to provide wide area coverage over long ranges are the cellular networks which were not designed for connecting such massive numbers of devices. Humans use networks to transfer large amounts of data in a continuous fashion but many IoT devices are wildly different with sporadic transmissions of short pieces of data. Low Power Wide Area Networks (LPWAN) have begun to exploit this space to provide a long range yet cost and energy efficient bridge to the internet [7].

LPWANs typically consist of star topology networks utilizing unlicensed radio bands, with single gateways supporting large numbers of end devices at ranges up to 15km in rural settings and 5km in urban ones. Unique modulation techniques ensure they can function in harsh radio environments and they have the capacity to provide wide area coverage, even being rolled out across entire countries. Examples of such networks include Sigfox, Ingenu, NB-IoT and LoRaWAN, with LoRaWAN being chosen as the focus of this project due to its open nature and increasing popularity.

As the Internet of Things continues to grow, the demand for geolocation grows in tandem [8]. It isn't hard to see the applications of positioning to IoT devices, ranging from industry to smart cities. Whilst in the past, GPS has been the go-to, it is no longer suitable for many devices due to its cost,

power consumption and lack of indoor penetration. Workers and assets can be tracked across potentially huge areas, positioning can inform logistic and manufacturing flows, incidents responded to in realtime and safety of people enhanced. It is envisioned that geolocation will ultimately become a standard for IoT, driving location-aware services and business processes, with the known position of every connected device being the norm. Up to a third of devices are critically dependent on location data [9], yet often remains an afterthought in the development of networks.

1.1 Project

1.1.1 Objectives

The objective of the project is therefore to explore the feasibility of locating devices over long distances using existing LoRaWAN infrastructure in a complex urban environment. No modification of the gateway hardware is possible and devices should remain low cost and low power, as described in later chapters. A device will be built and used to collect measurements from the network. These measurements will then be used to evaluate localization techniques using LoRaWAN.

1.1.2 Outline

The remainder of the report is split into chapters, with each chapter providing its own short introduction in addition to the outline below.
Chapter 2 will focus on LoRaWAN, introducing it and explaining how it works before also introducing Things Connected, the specific network used in this project.
Chapter 3 explores various positioning techniques starting with the Global Positioning System and arriving at Time Difference of Arrival, which is described in greater detail.
Chapter 4 goes on to explain what was actually done in the project with regards to hardware and software, interfacing both with the LoRaWAN network.
Chapter 5 then goes into detail about the investigations and data analysis that was carried out, presenting both results and discussions.
Finally, Chapter 6 provides conclusions on all the above and ponders future improvements to attain greater location accuracy.

Chapter 2

LoRaWAN

Whilst many LPWANs are currently being introduced, LoRaWAN is probably one of the most well discussed and explored - likely due to the openness of the standard developed by the LoRa Alliance. This chapter will briefly introduce LoRaWAN and the particular network being used in this report, Things Connected.

2.1 Overview

LoRaWAN is a member of a new and upcoming group of technologies, LPWANs, aimed at enabling a large proportion of the Internet of Things. It is particularly designed to connect low power, low cost devices known as nodes with the internet via gateways over long range wireless connections. Low power, low cost and long range are key tenets of the LoRaWAN philosophy, as are secure bi-directional communications, ease of use and mobility. Additionally, a fantastic feature of LoRaWAN is how well it functions with Non-Line-of-Sight, achieving long range even in complex urban environments and having deep indoor penetration too.

LoRaWAN itself is a media access control (MAC) layer protocol developed by a non-profit organization, the LoRa Alliance, as an open standard based on a proprietary technology by Semtech [10]. This proprietary technology, LoRa, which stands for 'Long Range', provides the physical layer of the system.

2.2 LoRa Physical Layer

The LoRa technology, patented by Semtech [11][12], uses a Chirp Spread Spectrum (CSS) modulation technique and can operate in the 434MHz, 868MHz or 915MHz ISM bands, dependent on a particular region's regulations [6]. CSS, which uses frequency chirps that vary linearly in frequency over time, means that any frequency offsets between the transmitter and receiver can be easily eliminated [10]. Offsets can be up to 20% of the bandwidth without affecting performance which allows for cheaper, less accurate clock crystals to be used, reducing the overall cost. This is also reduces effects of doppler shift in moving devices.

The bit rate achieved with LoRa varies with the bandwidth, spreading factor and code rate of the forward error correction (FEC). The bandwidth is typically 125kHz or 250kHz and bit rate typically

ranging from 0.3kbps to 11kbps [6].

The use of Chirp Spread Spectrum modulation should also aid the accuracy of timestamping the receive time in gateways for use in localization [13].

2.3 LoRaWAN MAC Layer

The first version of the LoRaWAN specification was released in 2015 [14]. A network consists of end-devices (nodes), gateways and a network server in a star topology. End-devices communicate with the gateways using the LoRa physical layer and the gateways act as middle-men, forwarding data to the network server via Internet Protocol (IP) backhaul [10]. LoRaWAN does not allow inter-device communication, data can only flow from an end-device to a server and vice versa.

LoRaWAN defines three classes of device. Class A devices are the most common and allows bi-directional communications by having two windows for receiving after every transmission. Therefore any downlink communication from the server must wait until the next uplink from the device. This means devices can sleep for extended periods of time between transmissions without worrying about listening for downlink communications, thus keeping power consumption low. Class B devices are similar, but also have scheduled times for additional receive windows. Class C devices have continuous receive windows between transmissions, thus have high power consumption.

Another key feature of LoRaWAN is the adaptive data rate, whereby depending on the quality of the link, the spreading factor will be altered to gain the best ratio of energy efficiency to robustness. As described previously, changing the spreading factor, along with the bandwidth and FEC code rate, will change the effective data rate [6].

LoRaWAN devices must be registered on the network before they can be used, either by Over-the-Air Activation (OTAA) or by Activation by Personalization (ABP). OTAA requires a device to request to join the network every session and the network must accept, providing a session key. ABP requires a device to have session keys stored in memory which have been setup by the network in advance.

2.4 Things Connected

LoRaWAN networks can be setup and operated by anyone, with both companies and individuals able to purchase hardware. This is in contrast to an LPWAN such as Sigfox who control the entire infrastructure themselves, allowing third parties to connect devices to their network using their hardware.

A number of LoRaWAN networks already exist globally, with The Things Network (TTN) being a notable crowdsourced example where individuals can deploy their own gateways using the TTN network server and anyone can freely use the network with any LoRa device. Other commercial networks include those by Everynet and SK Telecom, who claim to have provided LoRaWAN coverage to 99% of the population of South Korea [15].

For the purposes of testing in this report, the Things Connected network [16], primarily situated in London, will be used. It is a LoRaWAN network setup in late 2016 by Digital Catapult, whose remit is to work with UK firms on digital innovation to help strengthen the UK economy. Things Connected is free to use for the development and exploration of LPWAN products and services and has multiple partners and users across academia and business.

Chapter 3

Localization

In this chapter, various localization techniques will be examined and considered for use in an existing LoRaWAN network. Popular methods used with other radio systems include using the Received Signal Strength Indicator (RSSI), Angle of Arrival (AoA), Time of Arrival (TOA) and Time Difference of Arrival (TDOA) of signals between multiple gateways. Ultimately the decision is made that Time Difference of Arrival, also known as multilateration, is the most viable option for a number of reasons. However even this will present some hurdles due to limitations of the existing infrastructure with regards to gateway hardware, timing and the urban environment.

It is important to take a look at the scenario before exploring options. The network is operating in a complex urban environment resulting in multipath effects and Non-Line-of-Sight (NLOS) transmissions. Gateways are already in place but not evenly spread. This hardware cannot be altered but the exact locations of the gateways are known and a timestamp is provided for each packet received at each gateway. With regards to nodes, they should also remain low cost and low power, since these are key reasons for using an LPWAN, and ideally should require no additional hardware beyond the LoRa transceiver. As discussed in the sections below, these factors all limit the viable options greatly.

3.1 Global Positioning System

The simplest method of adding geolocation capabilities to any device is to utilize a Global Navigation Satellite System (GNSS) such as the commonly known United States Global Positioning System (GPS). Also available and supported by most GPS receivers are the European Galileo system and the Russian GLONASS system. GNSS systems can provide a typical accuracy of 10 metres, down to just 1 metre in some cases.

However to use GPS, additional hardware is required on a node which is undesirable. Adding a GPS receiver will add weight and bulk to an otherwise small, lightweight device, requiring another antenna in addition to that for the LoRa transceiver. More importantly, GPS would substantially increase the cost of a node and the power consumption.

To achieve a positional fix and time, a minimum of four satellites must be visible and this can be limited both outdoors in urban environments and severely limited in any indoor environments. With one of the selling points of LoRaWAN being the deep indoor penetration, this is another disadvantage of a GNSS system.

Since for most IoT and LPWAN use cases, location does not need to be known by the device itself but rather by the network, it makes sense to explore localization techniques based on the network infrastructure with no additional hardware rather than an external solution like GPS.

3.2 Received Signal Strength

The signal strength at a receiver is based on the amplitude of the incoming signal. As an electromagnetic signal travels through air or any other material, it is subject to attenuation - simply put, the further the distance travelled, the greater the attenuation. The free space path loss model can be used to describe the relation between transmission power and receive power in a line-of-sight (LOS) scenario.

Theoretically then, the distance from a transmitter to a receiver can be calculated from the RSSI. With multiple receivers, or gateways, a system of equations can be setup and solved to find the location of the transmitter - effectively the point at which circles extended at a radius of the calculated distance from the receivers intersect one another. This is known as trilateration. However, the received power is also related to other factors, such as antenna gains and the signal wavelength, and affected by many more including NLOS signals, interference, weather, and multipath effects. This results in low accuracy over long distances [17].

With LoRaWAN in particular, RSSI based positioning has been investigated in an indoor environment and found to give unsatisfactory and noisy results [18], which would only be amplified in an urban environment. However, encouraging results using RSSI fingerprinting and machine learning techniques for long distance location with Sigfox have recently emerged at the time of writing [19].

3.3 Angle of Arrival

The Angle of Arrival of a signal can be used to estimate the bearing of the transmitter from the receiver. With AoAs from multiple receivers, the location of the transmitter can be estimated as the point at which these bearings intersect. This is also known as triangulation. As few as two receivers are needed to estimate a position in this way and no time synchronization is required between either receivers or transmitter [20].

However, AoA techniques require sophisticated antenna arrays on a gateway and still often only provide bearings at a low resolution depending on the size of the array. The Angle of Arrival can also not be relied upon at all in heavily NLOS and multipath environments such as an urban city, where signals are reflected in reaching the gateway and could arrive from any direction [21]. LoRaWAN gateways only use simple, single antennas to keep them relatively cheap and so this technique is impossible with the current gateway hardware.

3.4 Time of Arrival

Using the Time of Arrival of a signal at a receiver is another distance-based trilateration technique like using RSSI. If the time of transmission and time of reception are known, then the propagation time of the signal can be calculated. This is then related to the propagation distance from the transmitter to the receiver by the speed of the signal (approx. $3 * 10^8 \mathrm{m\,s^{-1}}$). Once again, distances from multiple gateways can be formed into a system of equations and solved to find the position of the transmitter.

However, the important thing to note is the requirement for an accurate timestamps both at transmission and reception. With any timing based technique, high precision measurements are required due to the high propagation speed of the signals - just 1 µs difference is 300 metres. Transmitting an accurate timestamp from a node would require an extremely accurate clock source with little drift, which increases both the cost and power consumption. The simplest way to gain an accurate timestamp is from a GPS unit, which if available would clearly negate the requirement for a timestamp entirely.

3.5 Time Difference of Arrival

To avoid the need for particular hardware which would compromise the low cost and low power of nodes, the Time Difference of Arrival can be used. TDOA is known as a multilateration technique, whereby the time of transmission is unknown but the time of reception at multiple receivers is known. The difference between the times of arrival at pairs of receivers can be calculated or found from correlation techniques and hyperbolic equations formed as detailed below [20]. The transmitting node is then found to be located at the intersection of the hyperbolic curves. For a 2-dimensional location estimate, times from at least three receivers are required. The accuracy of the position estimate is dependent on the accuracy and precision of the timestamps at the receivers, on which clocks should be synchronized [22].

3.5.1 TDOA Equations

The Time Difference of Arrival of a pair of gateways forms a non-linear hyperbolic equation with two unknowns in the 2-dimensional case: the x and y coordinates of the transmitting node [23][24]. To solve for x and y, at least two equations are required meaning a minimum of three receiving gateways.

Letting the unknown position of the node $p_{node} = (x, y)$ and the known position of the ith receiving gateway $p_i = (x_i, y_i)$, the euclidean distance between these is $D_i = \sqrt{(x - x_i)^2 + (y - y_i)^2}$.

The difference between the distance to the ith gateway, D_i, and the distance to the jth gateway, D_j, is equal to the difference in arrival times multiplied by the speed of light, $(D_i - D_j) = C \cdot (t_i - t_j)$.

Therefore the full hyperbolic equation for a pair of gateways, i and j, is as defined in equation 3.1.

$$\sqrt{(x - x_i)^2 + (y - y_i)^2} - \sqrt{(x - x_j)^2 + (y - y_j)^2} = C \cdot (t_i - t_j) \tag{3.1}$$

Normally the first gateway to receive the transmission will be used as the jth gateway to keep the numbers positive and will remain the same in every formed equation, whilst the ith gateway will change for each subsequent gateway to receive the transmission. This will result in $n - 1$ equations where n gateways receive the transmission.

Figure 3.1 plots the three hyperbolic equations formed from the times of arrivals from four gateways, with the transmission location found at the intersection of the three hyperbola. Having more equations than needed due to additional receiving gateways, such as in this case, is an over-determined situation and can enhance the positional accuracy in the likely event that the timings are inexact.

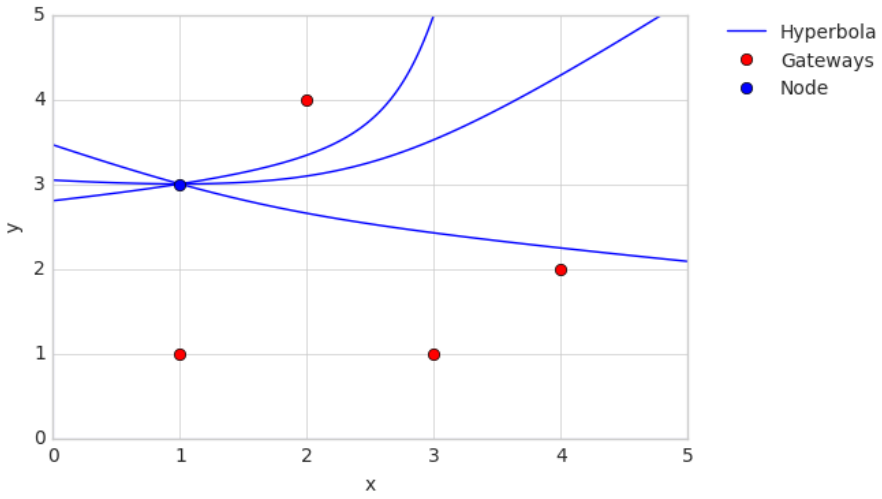


Figure 3.1: TDOA Hyperbola with 4 gateways

3.5.2 Hyperbolic Solver Algorithms

The non-linear nature of the set of hyperbolic TDOA equations 3.1 means it is more difficult to solve for the node position than in trilateration techniques. A number of efficient methods exist that include both iterative and closed form algorithms [23]. Closed form algorithms are more computationally efficient and directly calculate exact solutions but cannot easily support over-determined situations [25]. Two such algorithms are Fang's [26] and Chan's [27]. Iterative algorithms are popular and can take additional data in over-determined situations, with most based on the theory of least squares [28]. They begin with an initial guess and improve the guess on each iteration. Gauss Newton is one such algorithm. Kalman filters are also popular methods for solving the set of equations.

3.5.3 Problems

Whilst TDOA is the best suited technique for locating using LoRaWAN, it is not without its hurdles [29]. One of the biggest is the assumption that there is minimal difference between the euclidean distance between the transmitting node and receiving gateways and the length of the propagation path, which is often unlikely in an urban environment and LoRaWAN is specifically designed to not require line of sight. The consequence is that the propagation time and distance will often be longer than the euclidean distance reducing the accuracy of the estimated location. Methods of reducing the NLOS error and identifying NLOS gateways have been explored [30] and it is has been suggested that hybrid methods of TDOA with RSSI could be used to reduce the impact of both multipath effects and NLOS [31][32].

The issues of timing accuracy have already been discussed and the importance of clock synchronization between gateways is clear. With no control over the existing gateway hardware, this will likely be a limiting factor in the accuracy obtained. Ideally a high resolution timestamp would be generated at the physical layer of the receiver [25], with accurate clocks synchronized with a GPS reference time pulse. However, work has been carried out on reducing the need for synchronized clocks [33] and solving a set of unsynchronized TDOA measurements [34].

The last major issue which can limit the positional accuracy is the narrow bandwidth of the LoRa signals [29]. The ability to differentiate between two different signals depends on the signal bandwidth,

with $1/B$ giving the multipath resolution. Any signals that do not have a time difference greater than this cannot be differentiated, although there are methods of mitigating this. For example, with a typical LoRa bandwidth of 125kHz, the resolution would be 8 µs. In distance terms, multipath signals will not be resolved as being seperate unless the difference in path lengths is at least 2.4km. For this reason, many indoor positioning systems use Ultra Wideband (UWB) systems and attempts to perform wideband spreading with narrowband radios to replicate the accuracies achieved with UWB have been attempted [35]. However with such low resolution and potentially inaccurate timestamps from the gateways, this is unlikely to become relevant with the current infrastructure.

Chapter 4

Implementation

This chapter will describe the implemented test setup, ranging from the hardware through to the software and data gathering. The setup consists of a simple custom node, the existing Things Connected LoRaWAN infrastructure across London and custom server-side software. The node transmits its real location as calculated by the on-board GPS which is received by LoRaWAN gateways and uploaded to the server. These real coordinates of the transmission location can then be later compared to the estimated location by the multilateration algorithm.

4.1 Node Hardware

A simple node was designed, built and programmed in C++, consisting of an ESP8266 microcontroller, u-blox MAX8C GPS receiver and a Microchip RN2483 LoRa transceiver, as shown in figure 4.1. The RN2483 was chosen as it implements the full LoRaWAN stack on-board, reducing the complexity of the microcontroller code. The GPS receiver used a basic chip antenna whilst the LoRA transceiver used an 868MHz coiled helical antenna. The specific hardware is not particularly important however, but rather the functionality is.

The GPS receiver was configured to pedestrian mode since the primary method of testing would be a person walking with the node, as opposed to a flight or high velocity mode which add filtering to improve accuracy in these scenarios. The LoRaWAN configuration parameters (device address, network session key and application session key) were hard-coded into the microcontroller firmware.

The microcontroller follows a basic loop. In each iteration, set at roughly every 10 seconds, the microcontroller requests data from the GPS (latitude, longitude, altitude, number of satellites and time)

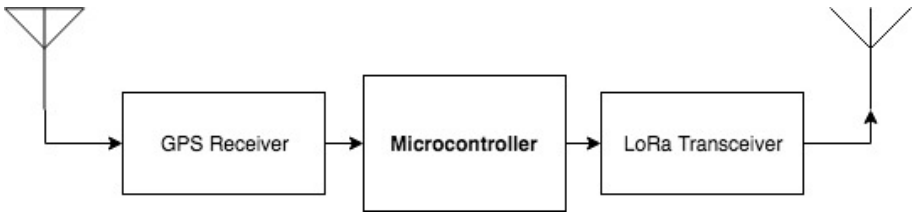


Figure 4.1: Node Diagram

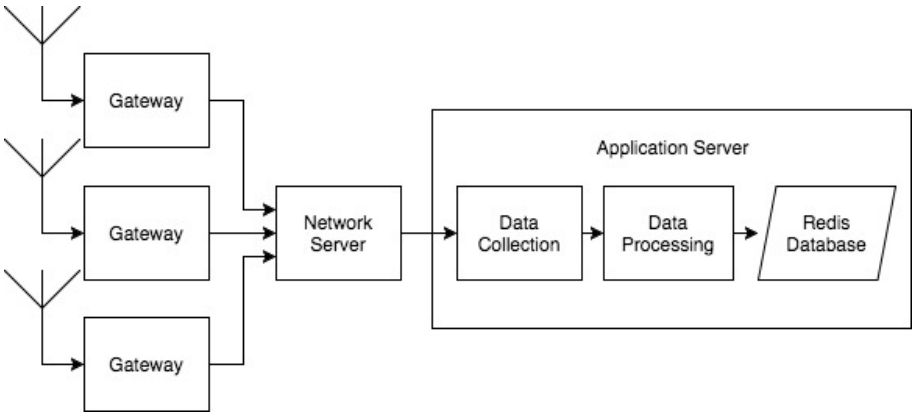


Figure 4.2: LoRaWAN to Server Diagram

and loads it into a struct datatype. A counter in the microcontroller's flash memory, such that it is persistent across power cycling, is incremented and also added to the struct. The entire 20 byte struct is then transmitted as a binary blob payload by the LoRa transceiver. Other functionality allowing transmission on a button press and short bursts of continuous data transmission was also implemented but not ultimately used.

The main file of the node firmware can be seen in appendix A. GPS interface code has been removed to a separate file and not included for brevity.

4.2 Server-side Software

When a gateway receives a packet from a node, it uploads it to the network server. The data can then be passed by the network server over an HTTP connection to the relevant application server, if the application server has implemented the necessary JSON-RPC methods [36]. A python script to be run on a public server with open port 8080 is found in appendix B.1 and implements the 'uplink' and 'post_uplink' methods. The 'uplink' method provides data as soon as one gateway has received a packet whilst the 'post_uplink' method should provide data on all receiving gateways a few seconds later. Unfortunately after testing, it was found that this latter method is not functional and that therefore obtaining details on all gateways is not possible this way. Hopefully it will be in the future.

Therefore, a workaround was found by using the unofficial API running the Things Connected dashboard. The last 20 received packets from each gateway are available from a 'messages' endpoint. By continuously polling this endpoint at a rate faster than that at which the node is transmitting, all data from all gateways can be collected. Figure 4.2 shows a simplified system diagram for the LoRaWAN and server-side systems. A python script to perform an HTTP POST request to login and continually fetch data from the unofficial API through HTTP GET requests is found in appendix B.2. The fetched data is published to a Redis Pub/Sub channel which is subscribed to by another script, found in appendix B.3, where the data is processed. The payload of each packet is decoded and the binary blob unpacked into the constituent pieces of data. If the packet payload already exists in the Redis database, the details of the new gateway receiving the packet is appended to the gateway list. If it doesn't already exist, the entire packet data is freshly inserted into the database.

It is at this point in the system that any final positioning algorithm would ultimately be included. Instead, another python script, in appendix B.4, allows all the received data to be exported from the

database to a JSON file for analysis and testing.

Chapter 5

Data Analysis & Results

In this chapter, the gathered data and the various transformations, algorithms and analytics carried out will be presented alongside the final positioning errors obtained. All investigations were programmed in Python and carried out in an iPython Jupyter notebook, with plots generated using matplotlib.

5.1 Gathered Data

The GPS node was taken for a walk through central London, through both areas with a high number and low number of gateways. Table 5.1 shows the number of gateways that each packet was received by, with 648 packets received in total by 8 different unique gateways. It can be seen that only about 25% of packets were received by 3 or more gateways, with three gateways being the minimum requirement to estimate a 2-dimensional position.

Figure 5.1 provides a geographical visualization of the data. The location of each gateway that received a packet is shown in blue, the true location of packets received by less than 3 gateways are shown in red and the true location of packets received by 3 or more gateways are shown in green. With a cursory glance, it can be seen that generally more packets are received by higher numbers of gateways in areas of high gateway density.

Taking a closer look at the received data, each gateway provides the latitude, longitude and altitude of its position and a timestamp with microsecond precision. Since the position coordinates for each gateway vary between packets, it would seem that for every packet received, a new location is retrieved from the GPS. It is unknown whether the timestamp is also retrieved live from the GPS, or from an internal clock. Either way, microsecond precision is not ideal bearing in mind that 1 µs is equivalent to 300 metres in distance. This also means it is unlikely the timestamping occurs at the instant a packet is first received.

Table 5.1: Number of packets received by n gateways

Number of Gateways	1	2	3	4	5	Total
Number of Packets	310	168	111	46	13	648

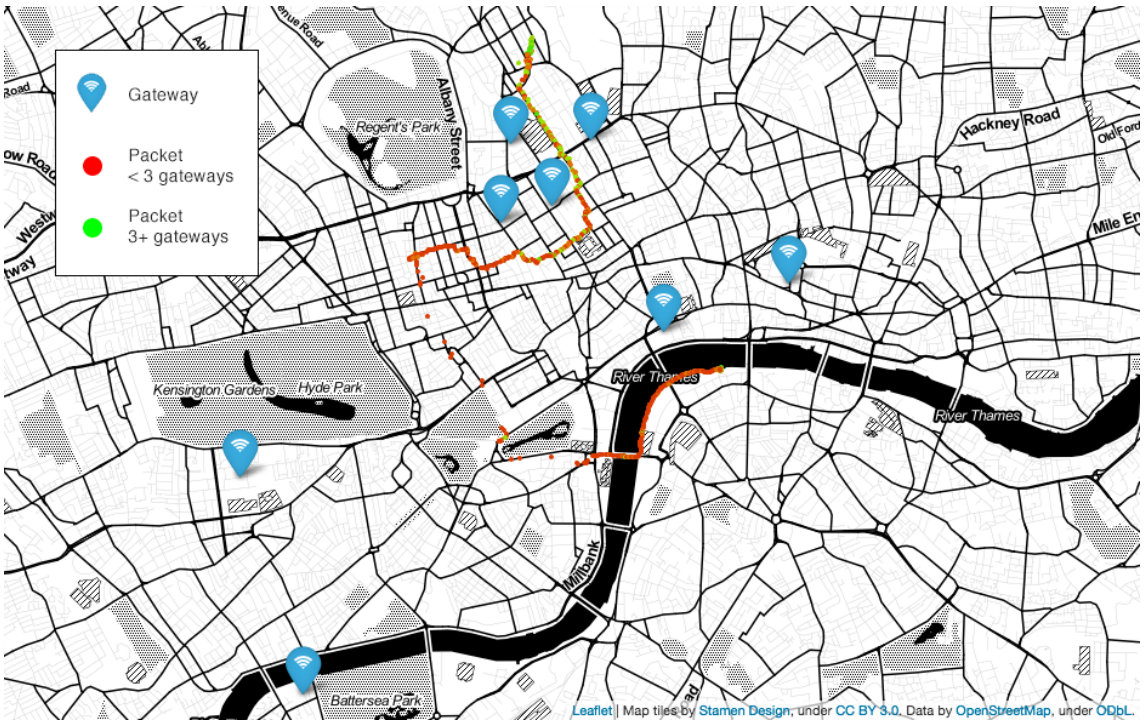


Figure 5.1: Geographical visualization of gathered data

5.2 Cartesian Coordinates

When working with the TDOA algorithms, it is important that a linear cartesian coordinate system is used, as opposed to the WGS84 latitude and longitude used by the GPS.

If 3-dimensional positioning was required, latitude, longitude and altitude (LLA) could be converted to the Earth Centred, Earth Fixed (ECEF) frame and vice versa. In this reference frame, the origin is at the centre of the Earth with the x-axis intersecting the Greenwich meridian and the equator, the z-axis being along the spin axis of the Earth and the y-axis being perpendicular to both of these. However, for 3-dimensional position estimates, a minimum of 4 gateways would be required and the effect of altitude is likely to be negligible compared to other sources of errors.

For 2-dimensional positioning, a local grid reference frame can be used. Latitude and longitude can be converted to OSGB36, the British Ordnance Survey National Grid reference system, and vice versa, through a number of calculations and adjustments detailed in OSTN02 [37]. The resulting cartesian coordinates are in metres and also allow for easy plotting of the hyperbolic equations and position estimates. When positioning errors and distances are given, it is simply the euclidean distance between two sets of (x,y) coordinates in metres.

5.3 Received Signal Strength

The Received Signal Strength Indicator (RSSI) has already been discounted as a viable method of positioning due to the broad number of factors involved of which distance is only one. Figure 5.2 plots the RSSI of each packet received by each gateway against the distance from the transmission to the

gateway. Whilst it can be seen that there is a relationship, the large variance of RSSI for any distance, particularly at the lower end, makes it unusable for positioning on its own.

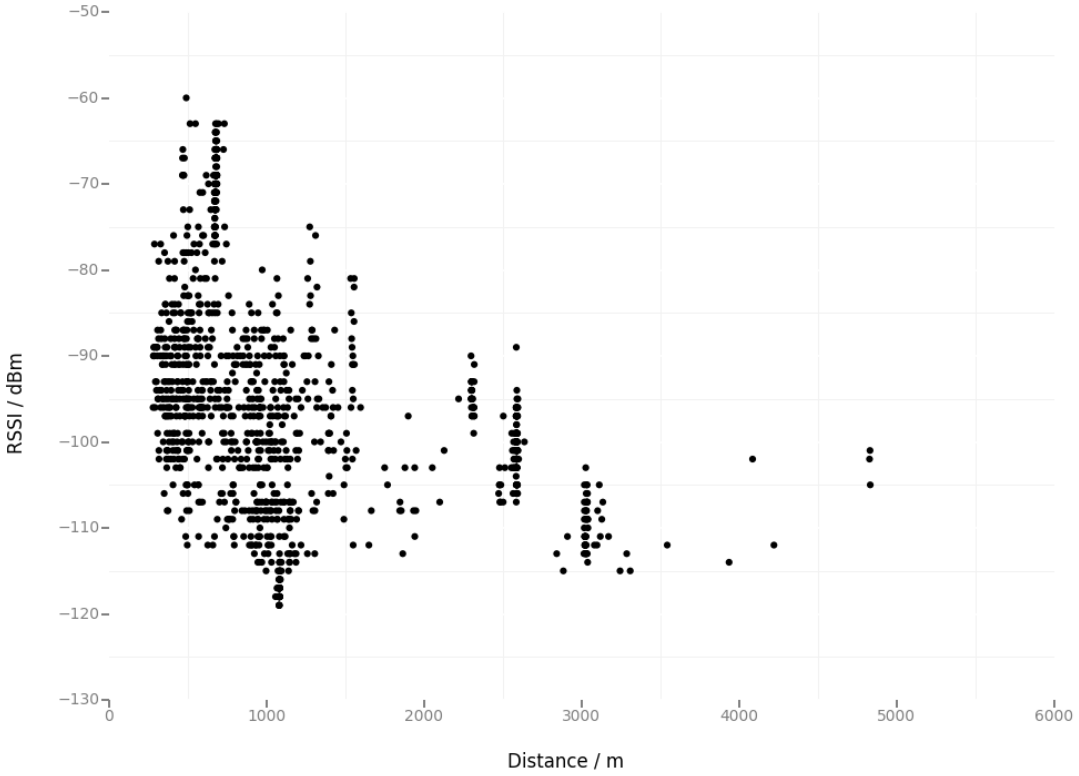


Figure 5.2: Relationship between RSSI and distance from gateway

5.4 Chan's Algorithm

Initial investigations were carried out using an implementation of Chan's algorithm [27].

The results of applying Chan's algorithm to the raw data directly, choosing the latter 3 gateways to have received each packet, were disappointing and can be seen in table 5.2. Out of 170 possible packets that were received by 3 or more gateways, the algorithm only returned position estimates for 18, or about 11%. The mean positional error of these when compared to the true locations was 1112 metres.

When the hyperbolic equations were plotted, it was found that the majority of packets did not have hyperbola that ever intersected or did not have hyperbola that were even valid due to the limited precision of the gateway timestamps. One such instance can seen in figure 5.3. A limitation of Chan's algorithm is that it calculates exact solutions to the hyperbolic equations, so in all these instances where no exact solutions exist it was unable to provide a result.

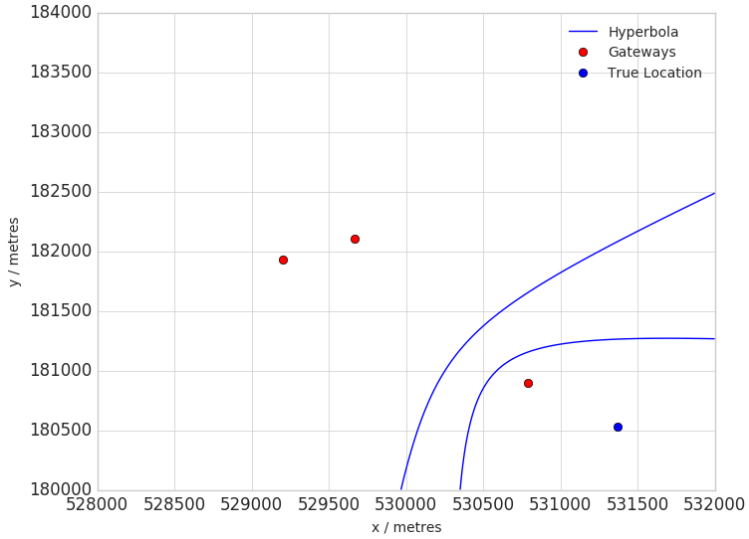


Figure 5.3: Geographical plot of packet with non-intersecting hyperbola

Table 5.2: Results of Chan's algorithm

	Resolvable Packets	Mean Error	Median Error	Standard Deviation
Raw Data	10.6%	1112m	882m	671m
Offsets	17.0%	38,810m	903m	110,200m
Offsets (outliers removed)	14.1%	866m	642m	767m

5.5 Offsets

In an attempt to improve the initial results achieved with Chan's algorithm, both in terms of the number of packets with a solution and in terms of the positional error, it was theorized that each gateway may have a timing offset associated with it which could be corrected for. This is similar to the Differential TDOA approach [33] which uses a reference node to compensate for imperfect synchronization between gateways. Offsets cannot be calculated for each gateway individually since we are using TDOA and do not have an exact transmit time, but offsets for pairs of gateways can be found.

The offset of a pair of gateways, i and j, is shown in equation 5.1. The result of the difference of the euclidean distances from the true position of the packet, $d_i - d_j$, divided by the speed of light, C, is what the time difference of arrival should be. Therefore the difference of this value and the TDOA, $t_i - t_j$, provides a time offset for that pair of gateways.

$$offset_{ij} = (\frac{d_i - d_j}{C}) - (t_i - t_j) \tag{5.1}$$

The offset is calculated for every possible pair of gateways for every packet. The probability density function (PDF) for these offsets is shown in figure 5.4 and it can be seen that they are quite varied, with the majority of values within 10 µs and the mean at 3.58 µs. In terms of distance, that is a mean offset of over 1km.

Of greater significance are the spread of values for each individual pairing, shown in figures 5.5a and

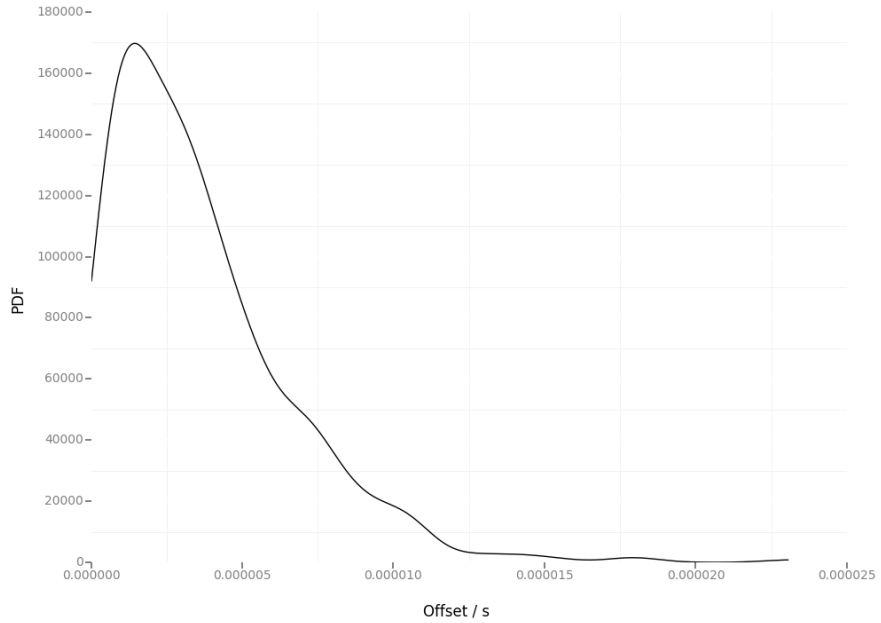


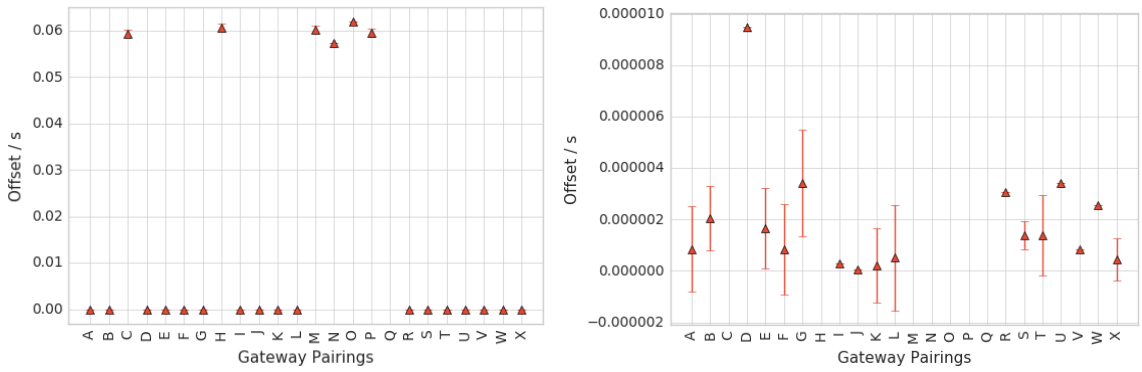
Figure 5.4: Probability Density Function for all calculated offsets

5.5b. 5.5a shows that some pairings have mean offsets of as high as 0.06s, which would cause significant errors in positioning and likely cause no solution to be found at all. 5.5b narrows down to those pairings with offsets in the microsecond range and allows the standard deviation bars to be seen. These show that the offsets are not particularly constant for most pairings, perhaps suggesting that other factors are involved such as multipath and non-line-of-sight effects or just that the gateways do not assign a timestamp in a particularly controlled way.

Applying these mean offsets to the TDOAs before using Chan's algorithm results in slight improvements with a solution for 17% of packets, shown in row 2 of table 5.2. Due to clear outliers however, resulting in solutions tens of kilometres from the gateways, the mean positional error is almost 40km but with a median of only 903m. Removing these outliers gives solutions for about 14% of packets with a mean positional error of 866m, shown in row 3 of table 5.2, down from an error of 1112m without applying offsets.

5.6 Iterative Algorithm

Even with the addition of offsets, Chan's algorithm severely limits the number of resolvable packets. Ideally an approximate solution is needed in the cases when the hyperbolic equations do not intersect and this is provided by utilizing an iterative algorithm rather than a closed. With the implementation of a least squares based iterative algorithm, an exact solution does not need to exist. In addition, all the data in over-determined situations can more easily be used, potentially improving the estimation accuracy - i.e. when there are more than 3 gateways that have received a packet and thus more TDOA equations than there are unknown coordinates. Being an iterative algorithm, an initial estimate is required and the further the estimate from the true solution, the more iterations it will take to converge on a solution. To keep processing times down, a limit on the number of iterations will normally be set and so ideally the initial estimate will be set to be as close to the solution as possible. With trial and error of several potential initial estimates, including using the origin (0,0) and the result of the previous estimate, it



(a) All gateway pairs

(b) Gateway pairs with µs offsets

Figure 5.5: Mean and Standard Deviations of Calculated Offsets

was found that using the position of the first receiving gateway minimized the number of packets which did not converge. The only other possible disadvantage of an iterative technique compared to a closed form algorithm such as Chan is the processing time - however, the processing is being performed on the server-side rather than an embedded device and so this an inconsequential issue.

When the iterative algorithm was run on the raw data, the results were immediately an improvement with a solution returned for about 44% of packets that were received by three or more gateways, compared to 14% with reasonable error for Chan's algorithm. As shown in row 1 of table 5.3, the mean positional error achieved was 972m, a similar figure to Chan's.

Since all receiving gateways are taken into account using the iterative algorithm, improvements were found by rejecting outliers in the receiving gateway timestamps. Erroneous times that would clearly put the node beyond the reach of the network were rejected, resulting in solutions for over 58% of packets with a mean positional error of just over 1km, shown in row 2 of table 5.3. By also applying the offsets for gateway pairs before using the iterative algorithm, solutions were found for about 55% of packets with a mean positional error of 930m and a lower standard deviation of 557m, shown in row 3 of table 5.3.

Table 5.3: Results of Iterative algorithm

	Resolvable Packets	Mean Error	Median Error	Standard Deviation
Raw Data	44.1%	972m	850m	614m
Raw Data (reject outliers)	58.2%	1080m	929m	716m
Offsets	55.3%	930m	863m	557m

To better visualize the positional error, figure 5.6 plots the cumulative density function for the iterative algorithm. Whilst the mean error for all three sets of results are around 1km, the variability of the positional error is large with errors as great as 3km occurring. It is seen again that including offsets provides slightly better results, with a 1440m accuracy at 80% confidence level. The raw data with outliers rejected is interestingly the worst, with 1720m accuracy at 80% confidence and ordinary raw data at 1510m accuracy - though it must be remembered that the ordinary raw data achieved a lower number of resolvable packets. Ultimately though, all three curves are very similar and so precise and reliable positioning would therefore appear to be beyond the scope of the current network, but a rough position over a broad city scenario can certainly be achieved.

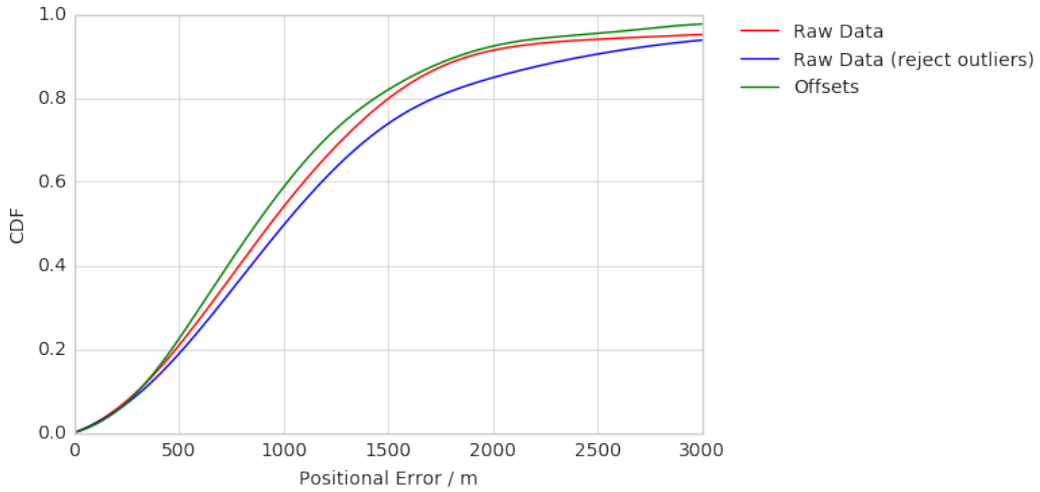


Figure 5.6: Cumulative Distribution Function for Positional Error using Iterative Algorithm

5.7 Oversampling

Lastly, oversampling was employed in an attempt to improve the positional accuracy and reduce noise. With the test node stationary at one location known to be picked up by multiple gateways, a large number of transmissions were made. Two methods were considered to test the viability of oversampling, the first being to calculate every possible TDOA and find the mean for each gateway pair to achieve a higher resolution time difference. The second was to find an estimated location for every packet and take the mean of the coordinates as the final estimate. This latter method was chosen as simpler and likely to produce similar results.

Table 5.4: Results of Oversampling with Iterative algorithm

	Resolvable Packets	Mean Error	**Final Estimate Error**
Data	29%	1196m	**467m**
Data (reject outliers)	64%	1198m	**392m**

Running the iterative algorithm on each individual packet received from the location resulted in a mean positional error of almost 1.2km with only 29% of packets achieving a solution. However, the mean location calculated from the set of estimated coordinates achieved a far lower positional error of just 467m, shown in row 1 of table 5.4.

Running the algorithm again whilst rejecting outliers resulted in a mean positional error of almost 1.2km but with 64% of packets achieving a solution. With more packets therefore being using to estimate the final location, a positional error of 392m is found, shown in row 2 of table 5.4. These results are visualized in figure 5.7, where the individual location estimates can be seen with the final estimated location in green and true location in blue.

Whilst these results are promising and more accurate than estimating locations based on an individual transmission, this method is likely only suitable for use on stationary nodes. Whilst transmitting multiple packets in bursts is possible, it is undesirable both due to power constraints and sharing use of the network amongst potentially large numbers of other nodes. In reality, any oversampling of location will happen over longer periods of time, reducing usefulness for moving nodes which are the primary use case for

positioning.

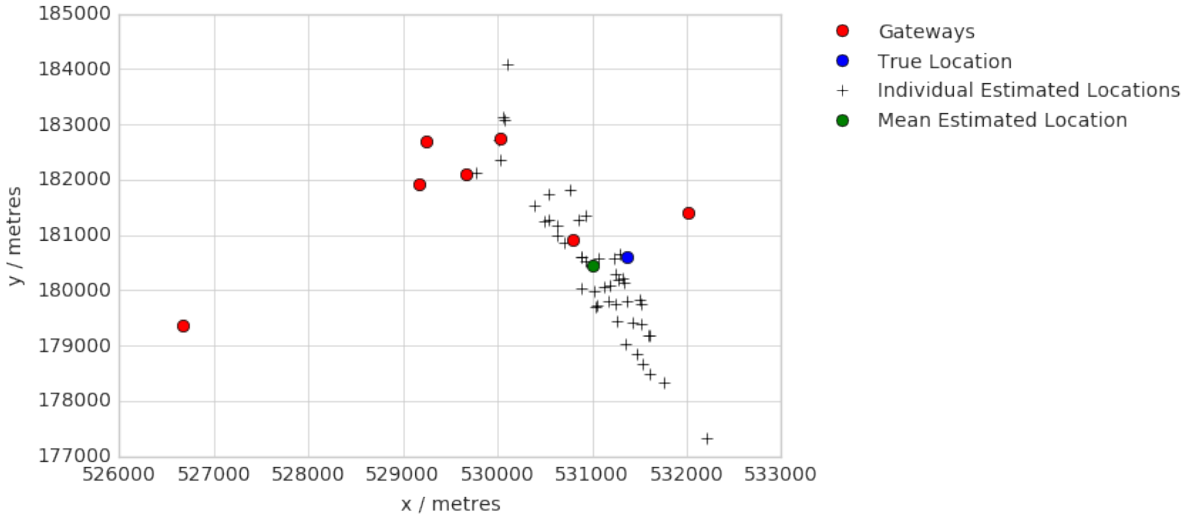


Figure 5.7: Geographical plot of oversampling with iterative algorithm

Chapter 6

Conclusion

With the huge growth rate of the Internet of Things and broad range of applications, it is clear that positioning will be a key requirement for many devices. As LoRaWAN too increases in popularity due to its long range, ease of use and low costs and low power consumption for connected devices, locating these devices without giving up any one of these becomes highly desirable.

It was decided that using the Time Difference of Arrival between pairs of gateways was the most viable approach and using an iterative algorithm was found to provide better results than a closed form solution for the collected data, but ultimately only coarse positioning estimates were able to be obtained. The smallest mean positional error was 930m, with the smallest accuracy being 1440m at an 80% confidence level. With only 25% of packets being received by 3 or more gateways and only just over half of these resulting in position estimates, it is clear that LoRaWAN as it currently stands could be suitable for general positioning over large areas but not for applications that require reliability or accuracy.

It was found that calculating offsets for pairs of gateways did improve the positional accuracy, however they had large variances likely due to both NLOS transmissions and the way in which timestamping is implemented in the gateways. Ideally offsets could be calculated on solely LOS transmissions allowing for the quantification of the effects of NLOS on accuracy. Cross validation should also be performed on a larger dataset since it is possible that the small improvements found were due to overfitting of the offsets to the data, particularly in the case of gateway pairs with only a few transmissions.

Oversampling of transmissions from a stationary node did appear to cause significant improvements in accuracy, down to a positional error of just 400m. However most applications for location estimates in the IoT will not be stationary. Bursts of transmissions in a short space of time may help for slow moving devices, but this is undesirable for low power devices and particularly in a network with many transmitting devices.

It is important to note, however, that the Things Connected network tested is relatively new and that with a broader spread of gateway locations, the numbers of receiving gateways are likely to be improved. It should also be noted that investigating the number of receiving gateways and distance from gateways was not a primary aim and ideally more data representing a greater geographical spread around the network's gateways would be collected and thus the reliability of these statistics improved.

With more gateways that are more evenly distributed, there would also be a higher proportion of transmissions that are closer to being LOS, reducing the error due to NLOS paths. A higher proportion of packets would also be picked up by 3 or more gateways, providing a better conversion rate of transmissions to location estimates. Femto LoRaWAN gateways have been suggested to provide increased coverage in

urban environments, particularly aiding positioning [5]. Combined RSSI and TDOA techniques could also be investigated, with the potential for utilizing machine learning with increased data.

Ideally though, the gateway hardware would be improved to provide accurate, precise timestamps at the physical layer. Synchronizing the gateway clock with a reference time pulse from GPS and assigning a time of arrival to a packet as the signal first arrives will greatly improve accuracy, rather than carrying out timestamping in the less time critical MAC layer which it is suspected is currently the case.

Bibliography

[1] K. Ashton, "That 'Internet of Things' Thing," 2009. [Online]. Available: http://www.rfidjournal.com/articles/view?4986

[2] "Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016, Up 30 Percent From 2015." [Online]. Available: http://www.gartner.com/newsroom/id/3165317

[3] A. Blanter and M. Holman, "Internet of Things 2020: A Glimpse into the Future," ATKearney, Tech. Rep. [Online]. Available: https://www.atkearney.com/documents/4634214/6398631/A.T.+Kearney{_}Internet+of+Things+2020+Presentation{_}Online.pdf/af7e6a55-cde2-4490-8066-a95664efd35a

[4] D. Evans, "The Internet of Things - How the Next Evolution of the Internet is Changing Everything," *CISCO white paper*, no. April, pp. 1–11, 2011. [Online]. Available: http://scholar.google.com/scholar?hl=en{\&}btnG=Search{\&}q=intitle:The+Internet+of+Things+-+How+the+Next+Evolution+of+the+Internet+is+Changing+Everything{\#}0

[5] T. Lestable, "The Cellular IoT Landscape: a disruptive opportunity for IoT NOW!" [Online]. Available: http://www.eucnc.eu/files/2015/presentations/workshop5/IoT-SAGEMCOM-EUCNC-29062015-v(0.3).pdf

[6] L. Vangelista, A. Zanella, and M. Zorzi, "Long-Range IoT Technologies: The Dawn of LoRa TM," 2015. [Online]. Available: http://www.patavinatech.com/en/

[7] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi, "Long-Range Communications in Unlicensed Bands: the Rising Stars in the IoT and Smart City Scenarios." [Online]. Available: https://arxiv.org/pdf/1510.00620.pdf

[8] "Converging the Internet of Things with Geolocation: The rise of the Location-Aware IoT," 2017. [Online]. Available: http://wyres.eu/company-news/converging-internet-things-geolocation-rise-location-aware-iot/

[9] "LoRa® Geolocation: Unlocking New Value for IoT Solutions." [Online]. Available: https://www.semtech.com/wireless-rf/lora-geolocation

[10] A. Augustin, J. Yi, T. Clausen, and W. Townsley, "A Study of LoRa: Long Range & Low Power Networks for the Internet of Things," *Sensors*, vol. 16, no. 9, p. 1466, sep 2016. [Online]. Available: http://www.mdpi.com/1424-8220/16/9/1466

[11] C. Hornbuckle, "Fractional-n synthesized chirp generator," Sep. 7 2010, uS Patent 7,791,415. [Online]. Available: http://www.google.com/patents/US7791415

[12] O. SELLER and N. Sornin, "Low power long range transmitter," Feb. 2 2016, uS Patent 9,252,834. [Online]. Available: https://www.google.com/patents/US9252834

[13] R.-R. Wang, X.-Q. Yu, S.-W. Zheng, and Y. Ye, "Design of a TDOA location engine and development of a location system based on chirp spread spectrum." *SpringerPlus*, vol. 5, no. 1, p. 1963, 2016. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/27900237http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5108751

[14] "LoRaWAN Specification v1," 2015. [Online]. Available: {https://www.lora-alliance.org/portals/0/specs/LoRaWAN%20Specification%201R0.pdf}

[15] S. Bicheno, "SK Telecom sets commercial IoT precedent with LoRaWAN launch," 2016. [Online]. Available: http://telecoms.com/473831/sk-telecom-sets-commercial-iot-precedent-with-lorawan-launch/

[16] "Things Connected: Digital Catapult launches IoT network," 2016. [Online]. Available: https://www.digitalcatapultcentre.org.uk/digital-catapult-things-connected/

[17] J. Yang and Y. Chen, "Indoor localization using improved rss-based lateration methods," in *Proceedings of the 28th IEEE Conference on Global Telecommunications*, ser. GLOBECOM'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 4506–4511. [Online]. Available: http://dl.acm.org/citation.cfm?id=1811982.1812130

[18] R. Henriksson, "Indoor positioning in LoRaWAN networks," Ph.D. dissertation, Chalmers University of Technology, 2016. [Online]. Available: http://publications.lib.chalmers.se/records/fulltext/241190/241190.pdf

[19] H. Sallouha, A. Chiumento, and S. Pollin, "Localization in Long-range Ultra Narrow Band IoT Networks using RSSI," *ArXiv e-prints*, Mar. 2017.

[20] K. J. Krizmant, T. E. Biedkatt, and T. S. Rappaportt, "Wireless Position Location: Fundamentals, Implementation Strategies, and Sources of Error," *IEEE Vehicular Technology Conf*, 1997. [Online]. Available: http://faculty.poly.edu/{~}tsr/wp-content/uploads/CV/ICP/1997-05-WirelessPositionLocation-Fundamentals,ImplementationStrategies,andSourcesofError.pdf

[21] L. Zimmermann, A. Goetz, G. Fischer, and R. Weigel, "Gsm mobile phone localization using time difference of arrival and angle of arrival estimation," in *International Multi-Conference on Systems, Sygnals Devices*, March 2012, pp. 1–7.

[22] Z. Li, Z. Li, D. C. Dimitrova, and T. Braun, "TDOA-Based Localization System with Narrow-band Signals." [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.415.3528

[23] N. El Gemayel, S. Koslowski, F. K. Jondral, and J. Tschan, "A low cost TDOA localization system: Setup, challenges and results," in *2013 10th Workshop on Positioning, Navigation and Communication (WPNC)*. IEEE, mar 2013, pp. 1–4. [Online]. Available: http://ieeexplore.ieee.org/document/6533293/

[24] C. Steffes, "Field experiments for TDoA-based localization of GSM base stations," in *2014 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*. IEEE, oct 2014, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/document/6954711/

[25] C. Li, C. Liu, and G. Liao, "Comparison of TDOA location algorithms with direct solution method," *Journal of Electronics (China)*, vol. 28, no. 4-6, pp. 652–657, nov 2011. [Online]. Available: http://link.springer.com/10.1007/s11767-012-0752-8

[26] M. Aatique, "Evaluation of Tdoa Techniques for Position Location in Cdma Systems," no. September, 1997.

[27] Y. Chan and K. Ho, "A simple and efficient estimator for hyperbolic location," *IEEE Transactions on Signal Processing*, vol. 42, no. 8, pp. 1905–1915, 1994. [Online]. Available: http://ieeexplore.ieee.org/document/301830/

[28] D. Koks, "Numerical Calculations for Passive Geolocation Scenarios," Defence Science and Technology Organisation, Australia, Tech. Rep., 2007.

[29] "LoRa Localization." [Online]. Available: https://www.link-labs.com/blog/lora-localization

[30] L. Cong and W. Zhuang, "Non-line-of-sight error mitigation in tdoa mobile location," in *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, vol. 1, 2001, pp. 680–684 vol.1.

[31] J. Lategahn, M. Muller, and C. Rohrig, "TDoA and RSS Based Extended Kalman Filter for Indoor Person Localization," in *2013 IEEE 78th Vehicular Technology Conference (VTC Fall)*. IEEE, sep 2013, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6692433

[32] N. El Gemayel, H. Jakel, and F. K. Jondral, "A Hybrid TDOA/RSSD Geolocation System Using the Unscented Kalman Filter," in *2013 IEEE 78th Vehicular Technology Conference (VTC Fall)*. IEEE, sep 2013, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/document/6692038/

[33] Z. Li, D. C. Dimitrova, D. H. Raluy, and T. Braun, "TDOA for narrow-band signal with low sampling rate and imperfect synchronization," in *2014 7th IFIP Wireless and Mobile Networking Conference (WMNC)*. IEEE, may 2014, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6878849

[34] S. Burgess, Y. Kuang, J. Wendeberg, K. Åström, and C. Schindelhauer, *Minimal Solvers for Unsynchronized TDOA Sensor Network Calibration*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 95–110. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-45346-5{_}8

[35] B. Kempke, P. Pannuto, and P. Dutta, "Harmonia: Wideband spreading for accurate indoor rf localization," in *Proceedings of the 1st ACM Workshop on Hot Topics in Wireless*, ser. HotWireless '14. New York, NY, USA: ACM, 2014, pp. 19–24. [Online]. Available: http://doi.acm.org/10.1145/2643614.2643616

[36] "Everynet Core API v1.0," 2017. [Online]. Available: https://everynet.atlassian.net/wiki/display/EP/Everynet+Core+API+v.1.0

[37] "OSTN02," 2002. [Online]. Available: https://www.ordnancesurvey.co.uk/business-and-government/help-and-support/navigation-technology/os-net/ostn02-ntv2-format.html

Appendix A

Node Firmware

```
/*  LoRaWAN GPS Tracker
 *  Daniel Saul 2017
 */

#include <rn2xx3.h>
#include <SoftwareSerial.h>
#include <EEPROM.h>
#include "gps.h"

#define EEPROM_LOW_BYTE 0
#define EEPROM_HIGH_BYTE 1

#define TX_COUNT 1
#define WAIT_TIME 5000

const char DEVICE_ADDRESS[]             = "";
const char APPLICATION_SESSION_KEY[]    = "";
const char NETWORK_SESSION_KEY[]        = "";

SoftwareSerial loraSerial(5, 4);

rn2xx3 myLora(loraSerial);

struct payload {

  int32_t lat;
  int32_t lon;
  int32_t alt;
  uint16_t i;
  uint8_t hour;
  uint8_t mins;
  uint8_t secs;
  uint8_t sats;

};

void setup() {

  // Start Serial
```

```
  Serial.begin(57600);
  loraSerial.begin(57600);
  delay(1000);
  Serial.println("Startup");

  // Start EEPROM
  EEPROM.begin(4);

  // Setup RN2483 radio
  initialize_radio();

  // Setup GPS
  gps_setup();

  delay(2000);

}

void loop() {

  payload msg;

  msg.lat = 0;
  msg.lon = 0;
  msg.alt = 0;
  msg.hour = 0;
  msg.mins = 0;
  msg.secs = 0;
  msg.sats = 0;
  uint8_t fx = 0;

  uint8_t b = 0;
  while(fx != 3 && fx != 2 ){
    Serial.println("Getting GPS...");
    b = getLocation(&msg.lat, &msg.lon, &msg.alt);
    delay(1);
    b = gps_get_time(&msg.hour, &msg.mins, &msg.secs);
    delay(1);
    b = gps_check_lock(&fx, &msg.sats);
    delay(1);
    Serial.println(msg.sats);
  }

  counter_inc();
  msg.i = counter_get();

  Serial.println("TXing...");

  for(int i=0;i<TX_COUNT;i++){
    myLora.txBytes((unsigned char*) &msg, sizeof(msg));
    delay(10);
  }

  delay(WAIT_TIME);
}
```

```
void initialize_radio() {

  delay(100);
  loraSerial.flush();

  String hweui = myLora.hweui();
  while(hweui.length() != 16)
  {
    Serial.println("Communication with RN2483 unsuccessful. Power cycle the board.");
    Serial.println(hweui);
    delay(10000);
    hweui = myLora.hweui();
  }

  Serial.println(myLora.sysver());

  bool join_result = false;
  join_result = myLora.initABP(DEVICE_ADDRESS, APPLICATION_SESSION_KEY, NETWORK_SESSION_KEY);

  if(!join_result){
    Serial.println("Unable to join network with provided settings.");
  }

  Serial.println("RN2483 Initialised.");

}

uint16_t counter_get()
{
    byte lowByte = EEPROM.read(EEPROM_LOW_BYTE);
    byte highByte = EEPROM.read(EEPROM_HIGH_BYTE);
    return ((highByte << 8) | lowByte);
}

void counter_set(uint16_t new_counter)
{
    EEPROM.write(EEPROM_LOW_BYTE, (byte) (0xFF & new_counter));
    EEPROM.write(EEPROM_HIGH_BYTE, (byte) (new_counter >> 8));
}

void counter_inc()
{
    counter_set(counter_get() + 1);
}
```

Appendix B

Server-Side Software

B.1 Python Everynet API Script

```
# -*- coding: utf-8 -*-
"""
    Integrate with the EveryNet Core API for LoRaWAN.
    Daniel Saul 2017
"""

from werkzeug.wrappers import Request, Response
from werkzeug.serving import run_simple

from jsonrpc import JSONRPCResponseManager, dispatcher

import json

from base64 import b64encode

@dispatcher.add_method
def uplink(**kwargs):
    print "uplink"
    with open('uplink.json', 'a') as f:
        json.dump(kwargs, f)
        f.write("\n")
    return "ok"

@dispatcher.add_method
def post_uplink(**kwargs):
    print "post_uplink"
    with open('post_uplink.json', 'a') as f:
        json.dump(kwargs, f)
        f.write("\n")
    return "ok"

@dispatcher.add_method
def downlink(**kwargs):
    return
```

```
@Request.application
def application(request):
    response = JSONRPCResponseManager.handle(
        request.get_data(cache=False, as_text=True), dispatcher)
    return Response(response.json, mimetype='application/json')


if __name__ == '__main__':
    run_simple('0.0.0.0', 8080, application)
```

B.2 Python Fetch Script

```
"""
    Use unofficial Things Connected dashboard API to get new packets
    Daniel Saul 2017
"""

from redis import StrictRedis
from redis.exceptions import ConnectionError

import time
import requests

import settings as s

def login(session):
    resp = session.post('https://dashboard.thingsconnected.net/api/login', json={'email': s.LOG

    if resp.status_code != 200:
        print "Unable to login."
    else:
        print "Logged in."

    return

def loop(session,r):

    resp = session.get('https://dashboard.thingsconnected.net/api/devices/%s/messages/' % s.DEV
    if resp.status_code != 200:
        login(s)

    else:
        r.publish('dashboard_messages', resp.text)

    time.sleep(s.FREQUENCY)


if __name__ == '__main__':
    session = requests.Session()

    r = StrictRedis(host=s.REDIS_HOST, port=s.REDIS_PORT, db=s.REDIS_DB)
    try:
        if r.ping():
            print "Redis connected."
    except ConnectionError:
        "Error: Redis server not available."

    login(session)

    while True:
        loop(session,r)
```

B.3 Python Processing Script

```
"""
    Process new packets
    Daniel Saul
"""

from redis import StrictRedis
from redis.exceptions import ConnectionError

import settings as s

import json
import struct
import base64
import copy

r = None

def process(msg, last_msg):
    data = json.loads(msg)
    new_last_msg = None

    for item in data:

        if new_last_msg is None:
            new_last_msg = copy.deepcopy(item)

        if last_msg==item:
            break

        try:
            payload = unpack_payload( base64.b16decode(item['payload'], True) )
            gateway = item['gw_gps']
            gateway['addr'] = item['gw_addr']
            gateway['time'] = item['gateway_time']
            gateway['rssi'] = item['rssi']

            saved_item = r.get('packet_%d' % payload['i'])
            if saved_item:
                saved_item = json.loads(saved_item)
                saved_item['gateways'].append(gateway)
                r.set('packet_%d' % payload['i'], json.dumps(saved_item))
            else:
                new_item = {'payload': payload, 'gateways':[gateway]}
                r.set('packet_%d' % payload['i'], json.dumps(new_item))
                r.rpush('packet_list', payload['i'])

            print r.get('packet_%d' % payload['i'])

        except:
            continue

    msg = json.dumps(new_last_msg)
    r.set('last_msg', msg)
```

```
    return new_last_msg


def unpack_payload(packed):
    # Payload Structure
    #0  int32_t lat;
    #4  int32_t lon;
    #8  int32_t alt;
    #12 uint16_t i;
    #14 uint8_t hour;
    #15 uint8_t mins;
    #16 uint8_t secs;
    #17 uint8_t sats;
    #18 char[] padding;

    fields = ['lat','lon','alt','i','hour','mins','secs','sats']
    unpacked_payload = list(struct.unpack('<iiiHBBBB2s', packed))
    payload = dict(zip(fields, unpacked_payload))
    payload['lat'] /= 10000000.0
    payload['lon'] /= 10000000.0
    payload['alt'] /= 1000.0
    return payload


if __name__ == '__main__':

    r = StrictRedis(host=s.REDIS_HOST, port=s.REDIS_PORT, db=s.REDIS_DB)
    try:
        if r.ping():
            print "Redis connected."
    except ConnectionError:
        "Error: Redis server not available."

    msg = r.get('last_msg')
    if not msg:
        last_msg = None
    else:
        last_msg = json.loads(msg)

    p = r.pubsub()
    p.subscribe('dashboard_messages')

    for message in p.listen():
        if message['type'] == 'message':
            last_msg = process(message['data'], last_msg)
```

B.4 Python Export Script

```
"""
    Export data from redis
    Daniel Saul 2017
"""

from redis import StrictRedis
from redis.exceptions import ConnectionError

import settings as s

import json


if __name__ == '__main__':

    r = StrictRedis(host=s.REDIS_HOST, port=s.REDIS_PORT, db=s.REDIS_DB)
    try:
        if r.ping():
            print "Redis connected."
    except ConnectionError:
        "Error: Redis server not available."

    keys = r.lrange('packet_list', 0, -1)

    data = []

    for key in keys:
       value = r.get('packet_%d' % key)
       data.append(json.loads(value))

    with open('packets.json', 'w') as f:
        json.dump(data, f)
```