Technische Hochschule Rosenheim

Faculty of Engineering Sciences

Course <Master's in Automation Technology>

Geolocation of wireless IoT devices in LoRaWAN Infrastructure using Multilateration technique

komro
Mehr Freiraum. Mehr Leben.

Master Thesis

by

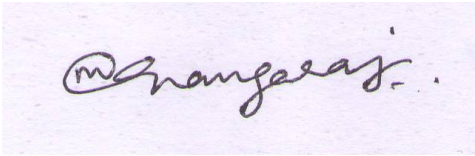
Thangaraj Mukara Dhakshinamoorthy (897150)

Date of Submission:	30.09.2020
First Supervisor:	Prof. Dr. Holger Stahl
Second Supervisor:	Prof. Dr. Markus Stichler
External Supervisors:	Dipl.- Met. Markus Heigl (Komro GmbH)
	Dipl.- Ing. Martin Landinger (Komro GmbH)

Declaration

I affirm that I have produced this work independently, have not submitted it elsewhere for examination purposes, have not used any sources or aids other than those indicated, and have not marked literal and analogous quotations as such.

Rosenheim, 30.09.2020



Thangaraj Mukara Dhakshinamoorthy

Abstract

The Geolocation is becoming a more crucial part of wireless sensor applications since the Internet of Things (IoT) deployment is growing very fast. The geolocation of wireless sensors gives a challenge of effective power consumption and long battery life. The IoT network standard LoRa/LoRaWAN enables the low power wide area network (LPWAN) which consumes low power at the same time communication over long range is possible. But building a communicating GPS receiver on the LoRa sensors for geolocation functionality consumes high power. This contradicts its inherent principle of low power consumption. Therefore, many LoRa gateway manufacturers enable geolocation feature which uses TDoA ranging technique that requires synchronisation only between the gateways (which act as reference points) and not necessary between the end nodes (whose position need to be resolved). That is the devices are running on their own time (unsynchronised or non-cooperative) and just sending normal uplink messages to multiple gateways which are time-synchronised. These received uplink frames are finetimestamped in nanosecond resolution at each gateways. The finetimestamps of single uplink frame which is received at multiple gateways gives rise to calculation of the Time Difference of Arrival (TDoA) which gives the loci of hyperbola corresponding to the possible device locations. With increasing number of gateways the location of device is narrowed to a point of intersection of hyperbolae. Thus, the geolocation of a wireless end-device is estimated without a GPS receiver, also with very low power consumption. However, the number of geolocation gateways needed per end-device would be equal to/more than three, i.e., the network operator should densify their gateways in order to have good coverage for multilateration algorithm. These geolocation gateways should have several components such as GPS receiver, special board support packages for generating finetimestamps and synchronised GPS clock leading to high capital cost. In this thesis, the various geolocation solutions in LoRaWAN network are implemented and its associated challenges, performance evaluation & optimization has been studied. This work was done for and supported by Komro GmbH, an internet service provider and daughter company of Stadtwerke Rosenheim, Bavaria, Germany.

Keywords: IoT, LoRa, LoRaWAN, Geolocation, Multilateration, TDoA, Network server, Geo-Resolver Server, Gateway(receiver or reference point), Device (target).

List of Abbreviations

IoT- Internet of Things
LoRa- Long Range
LoRaWAN- Long Range Wide Area Network
LPWAN- Low Power Wide Area Network
MQTT- Message Queue Telemetry Transport
HTTP- Hyper Text Transfer Protocol
TOA- Time of Arrival
TDOA- Time Difference of Arrival
RSSI- Received Signal Strength Indicator
SNR- Signal-to-Noise Ratio
DB- Database
NS- Network Server
GRS- Geo-Resolver Server
GRB- Geo-Resolver Backend
GW- Gateway
BSP- Board Support Package
SF- Spreadfactor
DR- Data Rate
NLOS- Non Line Of Sight
LOS- Line Of Sight
UL- Uplink
DL- Downlink
Tx- Transmission
Rx- Reception
M- Meters
RMS- Root Mean Square
DF- Dataframe
Geoloc- Geolocation related
KF- Kalman Filter

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

This chapter shows overview of Internet of Things (IoT), IoT standards and protocols, LoRa/LoRaWAN, Hardware requirement for geolocation capability, different types of Geolocation architectures in a LoRaWAN infrastructure.

1.1 Internet of Things (IoT)

This section shows the overview to Internet of Things (IoT) technology with the help of block diagrams and texts in brief.

IoT is a system of interconnected devices with Unique ID which have ability to transfer data to a network architecture through some dedicated protocols which demands no Human Intervention. This technology is grown between 2008 to 2010 in multiple sectors. Nowadays, there are several IoT applications such as commercial application, consumer application, Industrial application, Infrastructure application. The most uses cases can be seen in consumer applications since everyone is fascinated to modern technologies like controlling everything over voice for instance Home automation gadgets like Wearable, Voice Controlled Speakers etc. which are not so expensive. The second most use cases can be seen in Industrial applications since the term Industry 4.0 has been evolved which focuses on real-time like predictive maintenance of machine elements and condition monitoring. The IoT technology adapted for industrial applications are termed as Industrial Internet of Things (IIoT). This drives towards increased production with effective monitoring of machinery, thereby avoiding unexpected down-times. The IoT Infrastructure management is also developing nowadays since it reduces cost and time for maintaining the construction structures in good condition. For example, condition of Bridges, Roads etc. Many Internet Service Providers or Network Operators are focusing on adapting IoT network standards in order to deploy Smart city concepts which involves connecting different applications such Home Automation, Building Automation, GPS Asset Tracking applications for Ships, Waste Removal trucks etc., Various condition monitoring applications like Bee Hive Weight Monitoring, Intelligent Parking systems, Intelligent Waste Management, Smart meters for Water Consumption, Heat Consumption, Power Consumption which requires no Human Intervention (to take readings) rather it transmits the meter readings according to the user defined period to the central network server (typically owned by the State Government) over the air (using wireless IoT standards). Moreover the data transmission is done with less power consumption. These IoT devices also have high battery life typically lasts for years. IoT is not only all about efficient gathering of data to central server but also involves controlling or accessing the data from the central server via the smart phones through an IoT application such as MQTT client which subscribes to a specific uplink data sent of a particular application (device). However,

the privacy and security is a great concern for IoT technologies ever since it is evolved. There are many encryption techniques developed for secure data transmission such as AES, Certificate based SSL/TLS, VPN Tunneling etc. Cisco IBSG predicts that, by the year 2020 the number of IoT connected devices would reach 50 billion. The combination of machine learning algorithms with IoT technology leads to more intelligent applications in multi-disciplinary fields. Especially in the field of agriculture, Microsoft invests a lot of time and money to have a precision yield.

1.1.1 Types of IoT Standards & Protocols

This section shows the overview to various IoT standards & protocols with the help of block diagrams and texts in brief.

Table 1.1 IoT Wireless Standards [14]

Wireless Technologies Used in IoT							
Technologies	Modulation	Frequency	Coverage	Bandwidth	Maximum Data Rate	Private Deployments	Power Consumption
SIGFOX	BPSK	433MHz, 868 MHz, 915 MHz	10–40 km	100 Hz	100 bps	No	Low
NB-IoT	QPSK	Licensed under LTE	2–20 km	200 kHz	200 kbps	No	Low
LoRa	CSS	433 MHz, 868 MHz, 915 MHz	1–10 km	125 kHz, 250 kHz	50 kbps	**Yes**	**Low**
WiFi	DSSS,OFDM	2.4 GHz, 5 GHz	10–100 m	20 MHz, 40 MHz, 80 MHz, 160 MHz	Gbps	Yes	High
ZigBee	DSSS,QPSK	868 MHz, 2.4 GHz	10–100 m	2 MHz	250 kbs at 2.4 GHz	Yes	Low
Bluetooth	GFSK	2.4 GHz	10–100 m	1 MHz	2 Mbps	Yes	Low

1.1.2 LoRa/LoRaWAN

This section shows the overview to LoRa/LoRaWAN in IoT-Smart city infrastructure with the help of block diagrams and texts in brief.

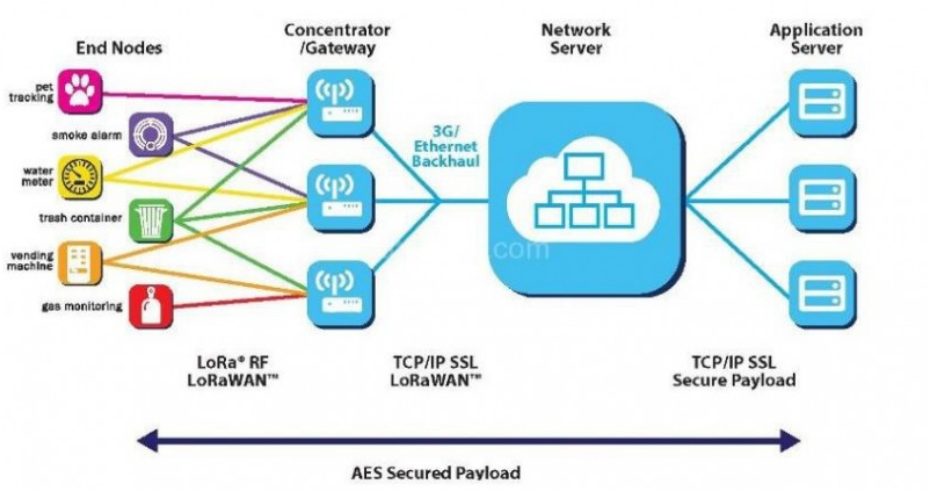


Figure 1.1 Classic_LoRaWAN_solution_Architecture [20]

The LoRa (a modulation technique or a physical layer protocol) and LoRaWAN (a MAC or network layer protocol) has been designed and developed by Semtech Corporation. In the figure 1.1 shown the classic architecture of LoRaWAN network. The End-nodes transmits the sensor data called Uplink data transmission happens periodically or event-driven based on user defined device configuration. The Uplink data reaches the gateway which acts as base station. The Gateway with the Unique Device EUI and Application key from the Network Server classifies the uplink messages to be published on particular topic to the Network server (under which the gateway is configured) through Packet Forwarder component over UDP/IP protocol (port 1700) and MQTT Bridge over MQTT Protocol (port 1883). After reaching to particular application tab in Network Server, it could be accessed from several third party Application Servers (from smart phones). The entire network is managed by LoRaWAN (Long Range wide Area Network), an IoT Network layer Protocol and the Modulation technique used between End-node and Gateway is called LoRa (Long Range, derived from chirp stack spread spectrum-CSS). The communication between Gateways and Network servers/ Application Servers are managed by TCP/IP Protocol, which is achieved by connecting the network cable to the gateway. It also consumes high power since the number of devices connected to a gateway creates multiple channels concurrently. Therefore, a power supply is needed for gateways in general. From the End-nodes till it reaches the application server the payloads are secured with AES encryption. Due to its property of Low power consumption, the data rates are slower but with longer coverage ranging in six levels of Data Rate (DR) starting from **DR0...DR5** which are corresponding to different Spreading Factors (SF) **SF12...SF7** respectively. With **DR5/SF7** being the best signal for faster transmission with less energy consumption among others. However the coverage range between End-node and Gateway is less in SF7 mode, therefore it is used to transmit over short distances. The Adaptive Data Rate feature (ADR) is implemented on Network Server to choose/ shift from one

DR/SF value to another automatically [8] [7].

1.2 LoRaWAN Geolocation Architecture

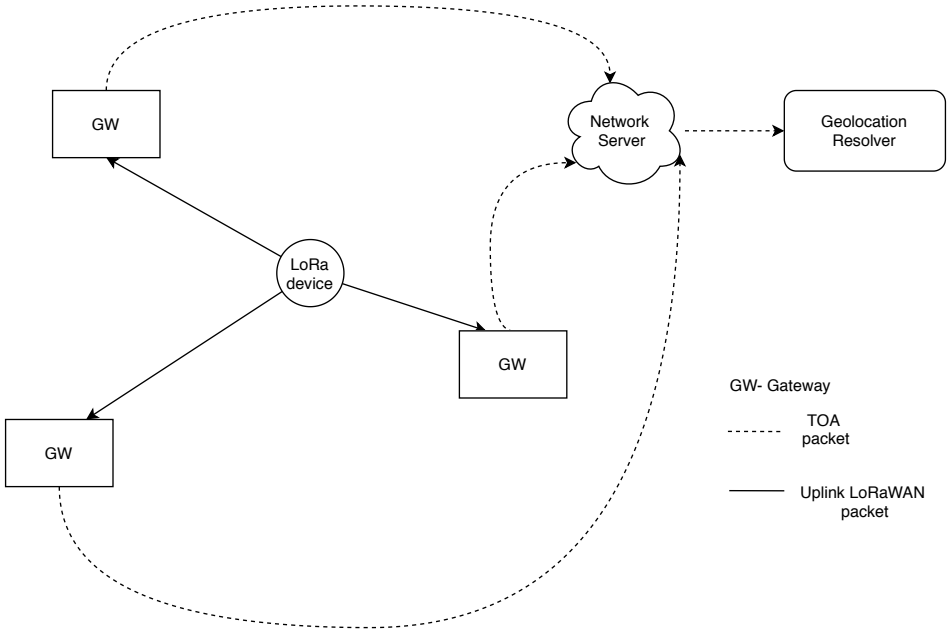


Figure 1.2 Block diagram of Geolocation architecture in LoRaWAN [12]

The LoRaWAN Infrastructure offers geolocation solution for low power wide-area networks (LPWANs), which enables to estimate the location co-ordinates for battery powered end-devices. The geolocation feature is supported by any existing LoRaWAN end-devices, without additional cost and additional processing power. However, they require "reachability of end-devices" at a minimum of three gateways and all gateways should be synchronised at nanosecond resolution [18]. Considering battery life of end-devices, LoRaWAN protocol provides two geolocation determination methods [9]:

- Received Signal Strength Indication (RSSI) ⇒ for course positioning up to 1000-2000 m accuracy
- Time Difference of Arrival (TDoA) ⇒ for finer Positioning up to 20-200 m accuracy

1.3 Hardware Requirements for Geolocation Capability

The receiver hardware (in this case, gateway) to be used for geolocation should generate fine-timestamps at a nanosecond resolution in order to have desired location accuracy in several meters[1.3.1]. In addition, the clocks of the gateways should be well synchronised with each other in order to measure the TDoA of the uplink signals, this could be either achieved by building a GPS clock on the gateways which are synchronised with GPS satellites or by using Rubidium or Cesium clocks [18]. It is important to have all mixers and oscillators of the gateway derived from exactly same timing reference. It is also

important to have the phase response of the gateway as linear as possible, which allows measurement of small phase differences. If there are any non-linear phase response, the gateway needs to be calibrated by injecting a known signal into the front end [18]. Also, the components of the gateway such as MQTT Bridge, Board support Packages need to be upgraded for geolocation functionality. For that, the gateway provider will give the necessary instructions. In general, the software stacks of gateway should have TOA estimation firmware, which timestamps the received uplink messages.

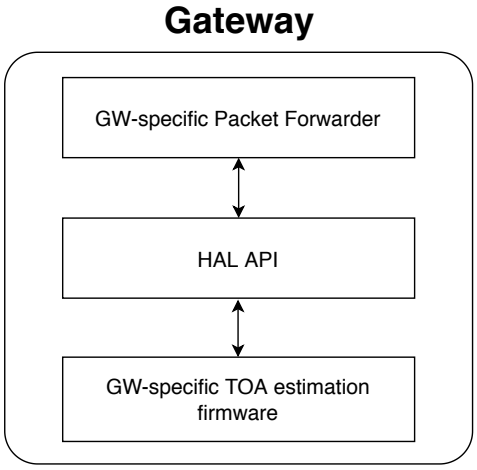


Figure 1.3 Block diagram of GW specification (typical for geolocation) [12]

1.3.1 Intuition

Assume the LoRa waves travel approximately in the speed of light

$$\begin{aligned} Speed\ of\ LoRa\ signal &= 3x10^8\ m/s \\ &= 300\ m/\mu s = 0.3\ m/nanosecond \\ &= 30\ cm/nanosecond \\ Frequency\ of\ LoRa\ f &= 868\ MHz \\ Period\ T_0 &= 1/f = 1/(868x10^6) \\ &= 1.15\ nanosecond \end{aligned}$$

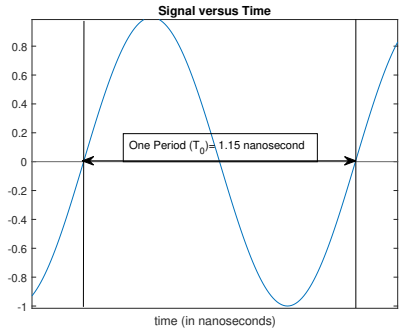


Figure 1.4 The figure shows one period of the LoRa signal which sets the minimum criteria for the gateway clock resolution to be less than or equal to 1.15 nanosecond

1.4 Types of Geolocation Architectures in LoRaWAN

This section shows the types of Geolocation Architectures in LoRaWAN infrastructure with the help of block diagrams and texts in detail.

1.4.1 Network server dependent architecture

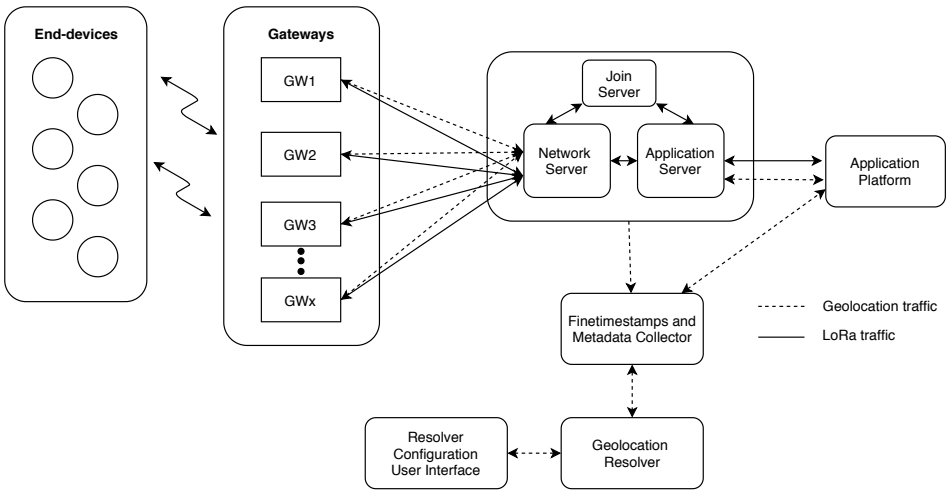


Figure 1.5 Block diagram of NS dependent architecture [12]

Two architectures are supported for geolocation in LoRaWAN architecture. First architecture is called network server dependent (NSD) architecture, the TOA packets are collected and passed to the geolocation resolving server (GRS) through the network server, as shown in figure 1.5. There are three options in this architecture as shown in figures 1.6, 1.7, 1.8

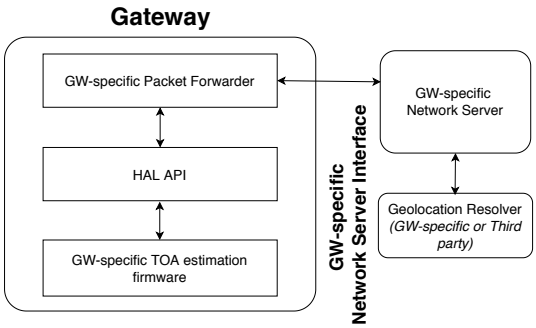


Figure 1.6 Option 1: Gateway specific- Packet forwarder & Network server [12]

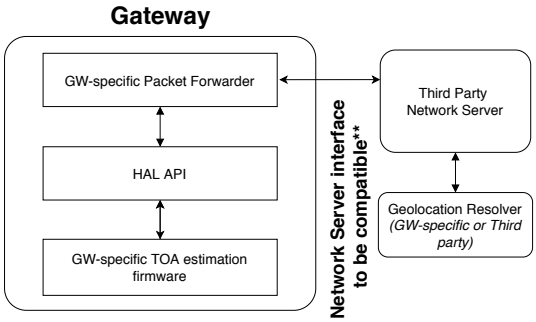


Figure 1.7 Option 2: Gateway specific Packet forwarder & Third party Network server [12]

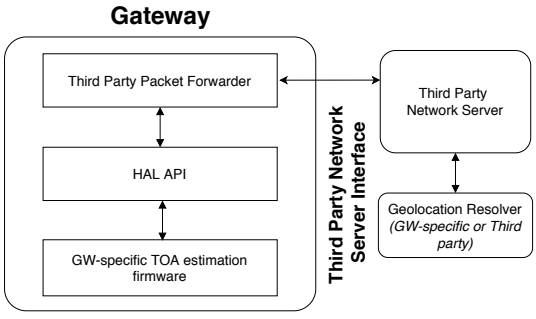


Figure 1.8 Option 3: Third party- Packet forwarder & Network server [12]

1.4.2 Network server independent architecture

In this architecture, the gateway directly sends the TOA packets to the GW-specific geolocation resolver component i.e. Fine Timestamp and Metadata Collector (FTMC)

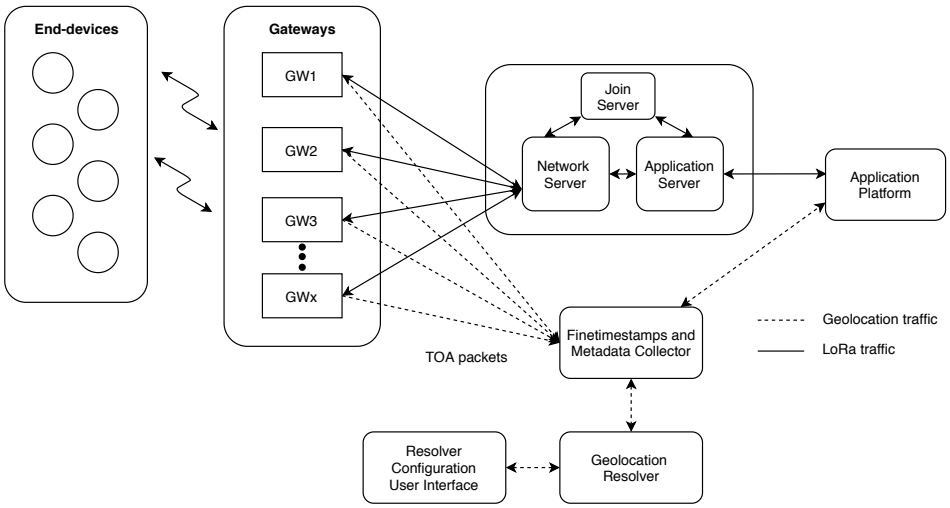


Figure 1.9 Block diagram of NS independent architecture [12]

2 Literature Review

This chapter shows some similar literature works (related to Geolocation in LoRaWAN) published previously [17].

2.1 Multilateration using TDoA Measurement Technique

In this section 2.1, a Github Project of Michael Jurasovic has been shown. It illustrates the method of drawing loci of hyperbolae using mathematical expressions which has errors associated with the input arguments like **TDoAs aka** $'t'$. It shows an set of mathematical equations which has to be satisfied/solved in order to calculate the target device position with uncertain time stamps t. However, the optimization techniques to improve accuracy of the target device position has not been shown in this project. In general, these errors follow a distribution pattern, for example Linear, Non-Linear, Gaussian etc.

Assume in Euclidean $\mathbb{R}^2$ space, the target device (transmitter) positioned at $\vec{x}$ whose transmission at the time t_0 propagates at the speed of light ($v \ \ ms^{-}1$) and the transmitted uplink signal is received by a set of n gateways. Let the closest gateway (receiver) that receives the transmission first be at co-ordinates $\vec{p_c}$ with the receive time t_c. Let the remaining towers (n-1) be located at $\vec{p_i}$ with transmission receive times t_i.

For the first gateway $\vec{p_c}$ one can say that the distance between the target device $\vec{x}$ and the tower is equal to the transmission propagation speed multiplied by the Time of Flight (ToF).

$$\|\vec{x} - \vec{p_c}\| = v(t_c - t_0) \tag{2.1}$$

Similarly, for each of the other gateways.

$$\begin{aligned} \|\vec{x} - \vec{p_i}\| &= v(t_i - t_0) \\ &= v(t_i - t_c + t_c - t_0) \\ &= v(t_i - t_c) + v(t_c - t_0) \\ \|\vec{x} - \vec{p_i}\| - v(t_i - t_c) &= v(t_c - t_0) \end{aligned} \tag{2.2}$$

Combine equations 2.1 & 2.2 to obtain the following, where the only unknown is $\vec{x}$.

$$\|\vec{x} - \vec{p_c}\| = \|\vec{x} - \vec{p_i}\| - v(t_i - t_c) \tag{2.3}$$

Expand equation 2.3 in order to draw loci (hyperbola). In TDoA measurement, with $'n'$ gateways, $'n-1'$ set of hyperbolic expressions are obtained. Subscript x & y denote the X

and Y components of a vector respectively.

$$\|\vec{x} - \vec{p_c}\| - \|\vec{x} - \vec{p_i}\| + v(t_i - t_c) = 0, \qquad i = 1, ..., n-1 \tag{2.4}$$

$$\Rightarrow \sqrt{(\vec{x_x} - \vec{p_{c,x}})^2 + (\vec{x_y} - \vec{p_{c,y}})^2} - \sqrt{(\vec{x_x} - \vec{p_{i,x}})^2 + (\vec{x_y} - \vec{p_{i,y}})^2} + v(t_i - t_c) = 0, \qquad i = 1, ..., n-1 \tag{2.5}$$

Solving the value of $\vec{x}$ estimates the target device position on several hyperbolae. The intersection of these hyperbolae gives the target device position. Let the least required number of gateways being n= D+1, where D is the dimension of the location problem that is, two (lat,long) or three (lat,long,alt) [26]. The intersection depends mainly on the accuracy of the time stamping. Higher the timestamp generation error, higher will be the intersection error (position estimation error).

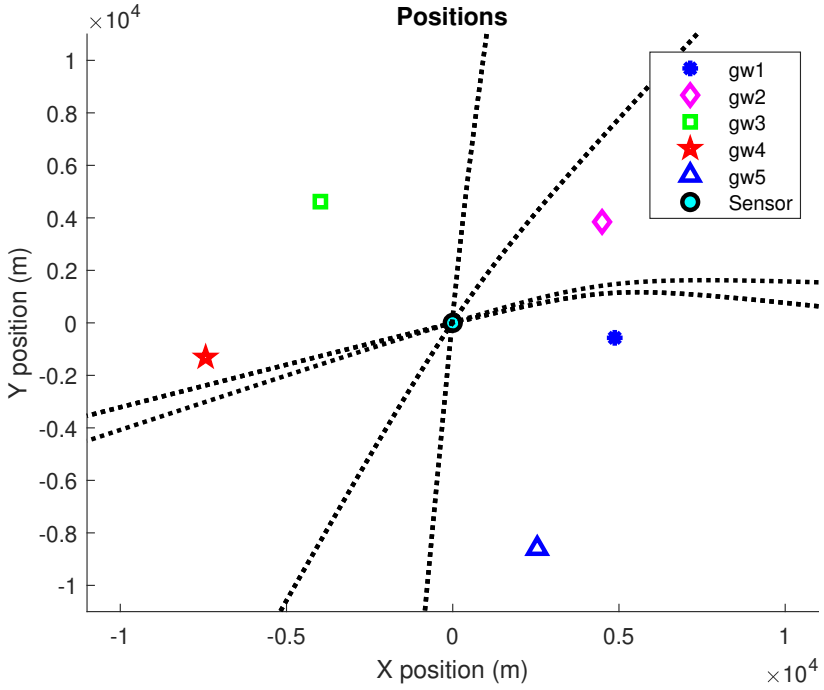


Figure 2.1 The figure shows, 4 hyperbolae plotted (with 5 GWs) using TDoA measurement technique with multilateration localization algorithm in MATLAB [21]

2.2 Sources of Error

In TDOA Implementation, there are three factors affecting location accuracy namely: Signal Processing considerations, Hardware requirements, Channel effects [18]. According to LoRaWAN Geolocation whitepaper [9], there are some important factors which can cause uncertainties in the accuracy of the geolocation position fix.

2.2.1 Factors affecting position accuracy fix

Propagation environment and multipath

In a Multipath-free environment, LoRaWAN geolocation performance is limited by the **gateway's clock accuracy**. However in presence of multipath, given the LoRa Bandwidth limitation of 125 KHz (at SF7/DR5), signal paths are often **indistinguishable**. Only the average channel delay can be estimated. In case of Non-line-of-sight measurement, a delay offset into the frame timestamps are introduced. Statistically, the timestamps are a bit late. But it is important to know that the timestamp errors may be negative, but they are never smaller than -1/bandwidth (i.e. never lower than the basic resolution of the system) [9]. There are several ways to reduce timestamp errors given below:

- Repetition of frames at different frequencies
- Antenna diversity at the gateway (typically two)
- Higher-density gateway deployment, which increases the number of available samples (uplink receptions) and increases the chance of line-of-sight measurements, thus increasing the accuracy of the TDoA
- Lower frame time stamping latency at the gateway
- Incorporation of out-of-band propagation error corrections to mitigate multipath issue

Gateway deployment geometry and density

The accuracy of position fix also depends on the end-node vs gateways geometry. The metric used to determine the quality of gateway placement is the Geometric Dilution Of Precision (GDOP), which is a measure of the "goodness" of the receiving gateway's relative geometry. Since TDoA measurement technique solves position using Hyperbolae (open curves), their intersection will lead to error amplification if the end-node and gateways are not well-positioned. GDOP captures this amplification factor, which depends solely on end-node vs gateways geometry. For position fixing in two dimensions, Horizontal DOP (HDOP) is computed. If **(HDOP== 1)**, then there is no amplification of error, for example, if the gateways show an uncertainty of 60 m on the timestamps (i.e., 200 nanoseconds), the final uncertainty will be 1x60 m= **60 m**. But If **(HDOP== 2)** the final uncertainty would be 2x60 m= **120 m**. HDOP could be lower than 1, if the gateway deployment density is higher. In order to have high geolocation accuracy (↑) and low HDOP (↓), gateway placements should be highly densified/ clustered in a regular gridded pattern. Position accuracy will be better at the middle of a square arrangement than with middle of a rectangle. Before deployment, it is highly recommended to compute HDOP and optimize gateway positions accordingly [9].

Other factors

- Position determination algorithm used by the geolocation solver

- Quality of gateway's time synchronization
- End-device dynamics and configuration

2.3 Optimization Approaches

In order to improve localization accuracy, Received Signal Strength Indication (RSSI) can be fused with the TDoA, i.e. **TDoA + RSSI** with the help Gaussian product method which is similar to measurement update step in Kalman filtering Algorithm. In the work of Ugur Bekcibasi et al. [1], a new approach for improving RSSI based localization has been implemented by introducing a **reference target device**, whose position is known, for example, a GPS sensor. Although, it requires an additional cost of reference node in the system, it improves the localization accuracy by calculating RSSI and η for each step, as a result the environmental dynamics are tolerated or eliminated. The simulation results are also depicted in form of tables and graphs which showed great improvement on using reference anchor method against the traditional three anchor method. The tracking algorithms such as Kalman filter, expectation maximization can also be used to improve the geolocation results. For RSSI, the finger printing method seems to be most popular in indoor localization applications. However it is also used for outdoor applications, it is complicated compared to localization using path loss model.

3 Theoretical Background

This chapter demonstrates the theory behind the Localization algorithm such as Triangulation, Trilateration, Multilateration and most importantly Time of Arrival (TOA) vs Time Difference of Arrival (TDoA) measurement techniques.

3.1 Geolocation

This chapter explains what is geolocation in general and how does it locate the objects/-target devices in real world co-ordinates.

Geolocation is the identification or estimation of the real-world geographic location of an object, such as a radar source, mobile phone, Internet-connected computer terminal or even an IoT end device. In its simplest form, geolocation involves the generation of a set of geographic coordinates and is closely related to the use of positioning systems, but its usefulness is enhanced by the use of these coordinates to pin-pointing at a meaningful location, such as a building in a street. The word geolocation also refers to the latitude and longitude coordinates of a particular location. [31].

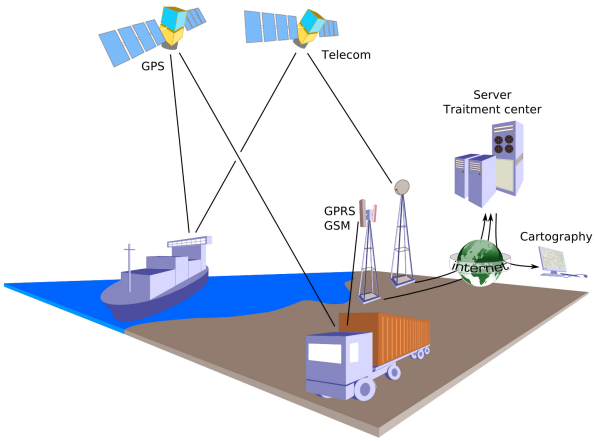


Figure 3.1 Principles of geolocation using GPS [10]

3.1.1 Wireless Geolocation Techniques

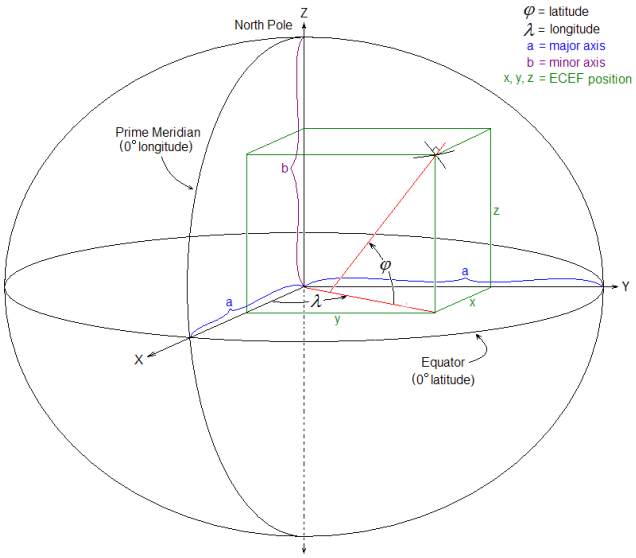
In order to geolocate a target device, the geolocation engine (Geolocation Resolver Server) often uses radio frequency (RF) positioning methods such as Angle of Arrival (AOA), Received Signal Strength (RSS), Time Difference of Arrival (TDoA), Time of Arrival (TOA) etc. Internet and computer geolocation can be performed by associating a geographic location with the Internet Protocol (IP) address, RFID, hardware embedded number (such

as MAC address, Device ID in LoRa), software embedded number (such as UUID, Device EUI in LoRa). [31].

3.2 ECEF Co-ordinate system

Representation of geodetic coordinates (latitude, longitude and altitude) in geocentric coordinates (x, y and z) are called Earth-Centered, Earth-Fixed (ECEF) coordinate system. The origin (0,0,0) is the centre of mass of the earth. The positive X axis passes through the (Latitude=0,Longitude=-90). The positive Y axis passes through the South-North pole. The positive Z axis passes through the prime meridian at Latitude=0 Longitude=0.For example, the exact ECEF coordinate of north pole is (0,6371,0) where 6371 is the radius of earth in km [29] [30].In geometry, a sphere is a set of points equally distanced from a single point called centre with a distance being radius r. A point on sphere could be mathematically represented as below,

$$x^2 + y^2 + z^2 = r^2$$



A sphere may be defined para-metrically in terms of (u,v) [3].

$$x = r * \cos(\phi) * \cos(\theta)$$
$$y = r * \cos(\phi) * \sin(\theta)$$
$$z = r * \sin(\phi)$$

where r is the radius of earth

Figure 3.2 ECEF Coordinate system [30]

Python provides a library called pyproj to convert latitude,longitude to x,y,z coordinates and vice-versa.
ECEF Syntax/Usage:
The syntax is pyproj.transform(FROM, TO, POINTS_TO_CONVERT,RADIANS/DEGREES)
The return has the form (X,Y,Z)
Example:

Listing 3.1 Python library for ECEF transformation

```
import pyproj
```

```
ecef = pyproj.Proj(proj='geocent', ellps='WGS84', datum='WGS84')
lla = pyproj.Proj(proj='latlong', ellps='WGS84', datum='WGS84')
x,y,z = pyproj.transform(lla, ecef, 12, 47, 500, radians=False)
lon,lat,alt = pyproj.transform(ecef, lla, x, y, z, radians=False)
print('x,y,z:',x,y,z)
#output: x,y,z: 4262795.32511108 906085.1174086552 4642130.465671182
print('lon,lat,alt:',lon, lat, alt)
#output: lon,lat,alt: 12.0 47.00000000000002 500.00000000279397
```

3.3 UTM Co-ordinate system

Although the ECEF converts the geodetic coordinates into Cartesian coordinates which is also three-dimensional, the more better to handle coordinate system is Universal Transverse Mercator (UTM) coordinate system. This coordinate system assumes earth is a ellipsoid surface. It basically converts the geodetic latitude, longitude and altitude into two-dimensional projection of the surface of earth (into a paper plane), where the earth map is divided into 60 zones, with each of them separated by 6 degrees in longitude. This coordinate is mostly used to plot in geographic global map and in military purposes. There is an library module available in python called **utm**. It could be imported by **import utm** command. It could convert the lat,long (3d) coordinates into x,y (2d) coordinates and vice-versa. Here the longitudes are called as eastings and latitudes are called as northings. Each point on earth has a unique easting coordinate (x), northing coordinate (y), Zonenumber and Zoneletter. It is a horizontal geolocation representation, without considering altitude. That is earth projected on a cylinder and that cylinder is unrolled as flat paper [34].

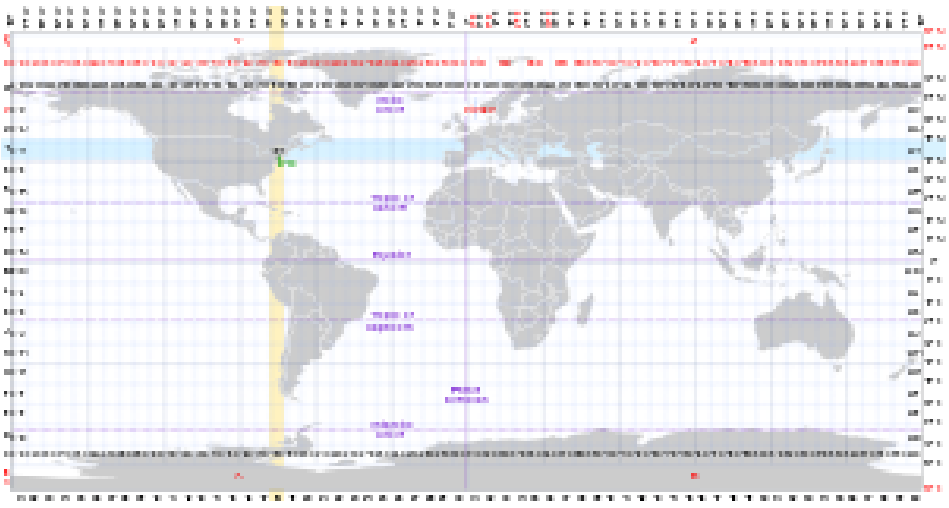


Figure 3.3 UTM coordinate system [34]

UTM Syntax/Usage:
The syntax is utm.from_latlon(LATITUDE, LONGITUDE)
The return has the form (EASTING, NORTHING, ZONE NUMBER, ZONE LETTER)
Example:

Listing 3.2 Python library for UTM transformation

```
import utm
UTMx, UTMy, ZoneNumber, ZoneLetter= utm.from_latlon(47.795918,12.118134)
lat,lon= utm.to_latlon(284188.0533790168, 5297639.812146321, ZoneNumber, ZoneLetter)
print('UTMx, UTMy, ZoneNumber, ZoneLetter:' ,UTMx, UTMy, ZoneNumber, ZoneLetter)
#output: UTMx, UTMy, ZoneNumber, ZoneLetter: 284188.0533790168 5297639.812146321 33 T
print('lat,lon:', lat,lon)
#output: 47.79591798243482 12.118132744776005
```

3.4 Popular Localization Algorithms: Triangulation vs Multilateration

This section discusses the overview of major localization algorithms and more emphasizes on Multilateration with TDoA technique which is used for this thesis.

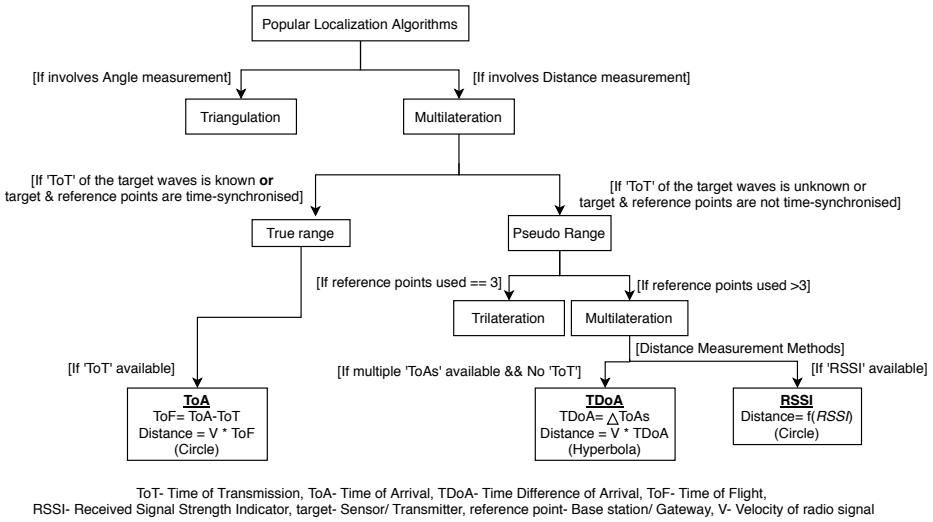


Figure 3.4 Block diagram of Popular Localization Algorithms [32] [33] [23]

3.4.1 Triangulation

In trigonometry and geometry, **triangulation** is a process of determining the location which uses **angles** to compute the position of target device (unknown) by forming triangles in relation to the reference points (known position). Triangulation today is used for many purposes, including surveying, navigation, metrology, astrometry, binocular vision, model rocketry and, in the military, the gun direction, the trajectory and distribution of fire power of weapons [33].

3.4.2 Multilateration

Multilateration is a method to determine the location of a movable vehicle or stationary point in space using multiple ranges (distances) intersecting at a same point in a certain geometry, for example, circle, hyperbola etc. The distance might be calculated from

the properties of Radio Frequency (RF) signal being transmitted (in this case, LoRa) such as signal propagation time, signal strength etc. This kind of RF based Localization methods are called Network-based Positioning [25]. In geometry, **trilateration** is defined as the process of determining locations of target devices (unknown) from a set of three reference points (known positions) through the **distance** measurement. The reference points may be any kind of receiver terminal such as gateways, base stations, WiFi routers etc. Based on the available RF signal information like ToT, ToA, TDoA, RSSI the **geometry of circles, spheres or triangles or hyperbola** will be chosen through which the position estimation would be done. Multilateration is same as trilateration but here the number of reference points are **more than three**. There are two types of multilateration: Pseudo range- & True range- Multilateration as shown in 3.4. **Pseudo range multilateration** is a navigation and surveillance technique based on measurement of the times of arrival (ToAs) of energy waves (radio, LoRa etc.) having a known propagation speed typically the speed of light (c). Prior to computing a solution, the time of transmission (ToT) of the waves is unknown to the receiver whereas in **True range multilateration** the ToT of the waves is known prior to computing a solution. For this thesis, the former has been used. Multilateration system can be described as measuring n + 1 ToAs using a clock with an unknown offset from target device. A ToA, when multiplied by the propagation speed, is termed a pseudo range. Using the measured ToAs, the user implements an algorithm that either: (a) determines the time of transmission (ToT) for the same clock and n target device coordinates; or (b) ignores the TOT and forms n time difference of arrivals (TDoAs), which are used to find the n target device coordinates [32]. Localization based on ToT and ToA information of RF signals are called Time of Flight (ToF) or Time of Arrival (ToA) measurement system which estimates true distance between the target and reference point, plotting all the possible target location in the form of a circle with radius $'r'$ being the estimated true distance. Localization based only on ToA information of received signals are called Time Difference of Arrival (TDoA) or also called hyperbolic measurement system, because they form a loci of hyperbola using range (pseudo) distance between target and reference points calculated from TDoA measurement technique. A TDoA, when multiplied by the propagation speed, is the difference in the true ranges between the target and the two reference points involved. For surveillance, a TDoA system determines the difference in the target distance to pairs of reference points at known fixed locations. For one reference point pair, the distance difference results in a set of possible target locations that satisfy the mathematical equations shown in literature review 2.1. When these possible locations are plotted, they form a hyperbolic curve. To locate the exact target on the curve, multilateration depends on multiple transmission of same uplink frame received by n reference points which leads to intersection of $n-1$ hyperbolae. These systems are relatively undemanding of the target receiver, as its 'clock' can have low-performance/cost and is usually not synchronized with the clock of reference points [32].

3.5 Localization Techniques in Multilateration Algorithm

3.5.1 ToA based Localization

The Time of Arrival Technique (ToA) is one of the most common techniques used for localization of target device. The target device (sender) sends the uplink data to three or more reference points (receiver) in which the clock of sender and receivers are well-synchronised which means the time of arrival of the uplink frames at the reference points (known position) are absolute (no offset) with respect to the sender. The possible sender locations are plotted as a **circle** per reference point at which the uplink frame is arrived. In the below figure 3.5, R1, R2 & R3 be the position of the reference points whereas the Point P be the target device where the distance is a true values forming multiple circles around each reference points used [23].

In the below equation 3.1, c' is the speed of light, $t_{arrival}$ is the Time of Arrival (ToA) and

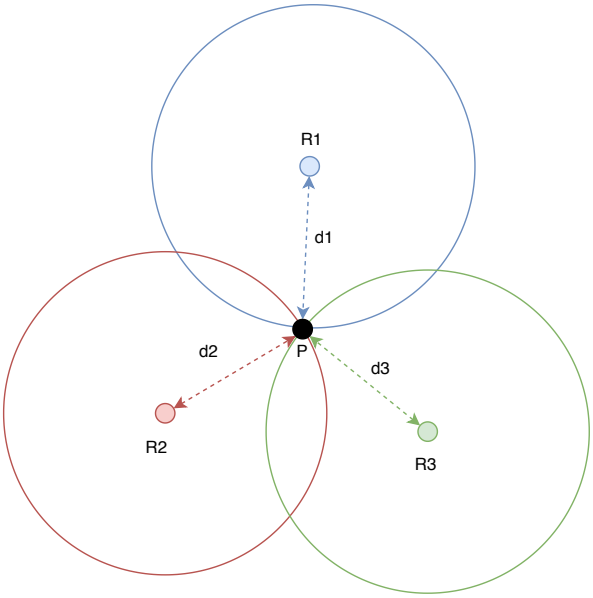


Figure 3.5 TOA Measurement technique

$t_{transmission}$ is the Time of Transmission (ToT). Using this equation, the distance 'd' which gives the possible target locations can be calculated. Since ToT is known.

$$d = c.\ (t_{arrival} - t_{transmission}) \tag{3.1}$$

Let the x_{ref} & y_{ref} be the x & y co-ordinates of the reference points respectively, 'd' be the calculated distance from equation 3.1 and x & y be the target co-ordinates.

$$d = \sqrt{(x_{ref} - x)^2 + (y_{ref} - y)^2} \tag{3.2}$$

3.5.2 TDoA based Localization

The Time Difference of Arrival Technique (TDoA) is quite easy to implement and the second-most popular position measurement techniques. It is more versatile and cheaper

than TOA. The target device sends the uplink frame to reference points in which the clock of the sender and the receivers are non-synchronised. This means the time of arrival of the uplink data at the reference points are relative with respect to sender (unknown offset). This will lead to a time difference of arrival of the same uplink frame at multiple reference points with synchronised clock. This $n-1$ TDoA between each pair of n reference points is used to localize the target device. Further leading to pseudo range distance between target and reference point. The possible target locations are plotted as a **hyperbola** per each pair of reference points (known position). So if four reference points are used, then three TDoAs are obtained. As a result only three hyperbolae are formed. In addition, this method reduces the cost of synchronization between sender's clock and receiver's clock. Especially, in case of low power consumption applications like LoRa it is an important measurement technique to be used since low-cost, battery-powered end-devices are used. In the below figure 3.6, TDoAs between three base stations are highlighted with two colors respectively. Each time difference Δd plots the possible sender on a branch of a hyperbola focused to the corresponding base stations (corresponding branches are differentiated with colors to their respective TDoAs). Multiple TDoAs leads to multiple hyperbolae. Now, the sender is located at one of the two intersections. Other navigation information can also be used to optimize intersection point [23].

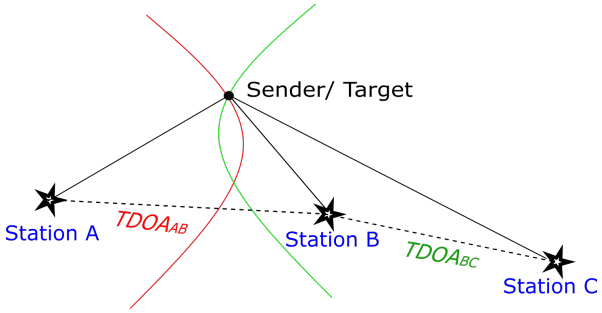


Figure 3.6 The three base stations are Stations A, B, C, whose locations are known. The times it takes for a RF signal to travel from the sender/target device to the stations are unknown, but the time differences are known from subtracting ToAs between each stations [11]

From the equation 3.3, the Δd can be calculated from Δt which is nothing but TDoAs between base stations (reference points)

$$\Delta d = c.\ (\Delta t) \tag{3.3}$$

From the equation 3.4, the x & y co-ordinates of the target device are calculated using the above calculated Δd and known co-ordinates of the two base stations $(x_1, y_1)(x_2, y_2)$

$$\Delta d = \sqrt{(x_2 - x)^2 - (y_2 - y)^2} - \sqrt{(x_1 - x)^2 - (y_1 - y)^2} \tag{3.4}$$

A simplistic demonstration relating to this thesis application is shown in figure 3.7 Lets assume the gateway 1 receives the signal at 16:00:02 and gateway 2 receives at 16:00:01.

The time difference Δt is 1 s. With signal speed being 3 m/s. Lets calculate the distance Δd,

$$\Delta d = Speed.\ \Delta t = 3\ m/s.\ 1\ s = 3\ m$$

Remember, this distance is constant for any point on the hyperbola. i.e. D1-D2 is always constant (Δd). According to properties of hyperbola, c is the distance between two gateways (foci). Lets take $c = 9\ m$. The foci formula is,

$$c^2 = a^2 + b^2$$

To find a^2:

$$a = \Delta d/2 = 3/2 = 1.5\ m$$
$$a^2 = 2.25\ m$$

To find b^2:

$$9^2 = 1.5^2 + b^2$$
$$b^2 = 78.75\ m$$

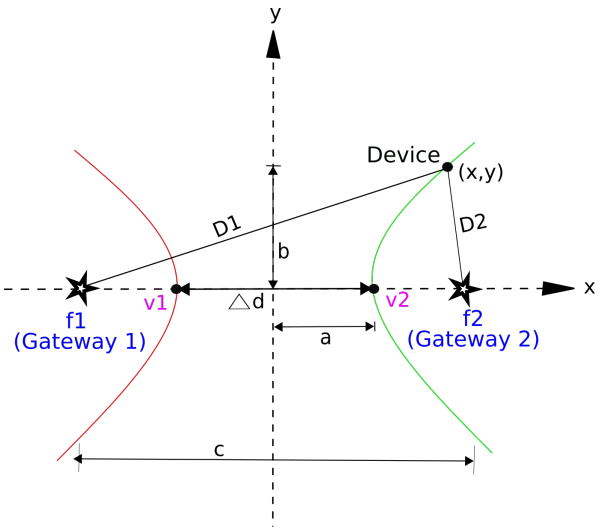


Figure 3.7 The two gateways are imagined as foci $f1$ & $f2$ respectively and any TDOA based device in-between lies on the hyperbolic curve around the foci (gateways) which are set of possible device locations between two foci and their x-intercepts are called vertices denoted as $v1$ & $v2$ respectively. The red hyperbola is imaginary which is mirrored green curve becomes real when distance D1<D2

The general representation of hyperbola (with known TDOA) is,

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1 \tag{3.5}$$
$$\frac{x^2}{2.25} - \frac{y^2}{78.75} = 1$$

If one co-ordinate is known, the other can be calculated using the formula 3.5. In order to solve this equation using linear algebra in python, one can use optimization methods such as least_squares method, Gaussian-elimination, LU decomposition etc. So far the time based localization techniques are explained, which calculates distance using time taken for the signal propagation. But in the next section, the signal strength based distance calculation will be explained.

3.5.3 RSSI based Localization

The Received Signal Strength Indication (RSSI) is the received signal power in milliwatts and is measured in dBm. It is the measurement of how "well" a receiver can hear a signal from a sender. The RSSI is measured in dBm and is negative value. The closer to 0 the better the signal is. Typical LoRa RSSI values are [22]: RSSI minimum = -120 dBm

- If (RSSI == -30 dBm): signal is strong.
- If (RSSI == -120 dBm): signal is weak.

The distance between target and reference point can be estimated by measuring the strength of the received signal at reference point. This distance-based technique needs at least three reference points to determine the two-dimensional location of a given target. RSSI systems are very interesting for urban and indoor geolocation systems, given that this technique is already available for cellular and WLAN networks. But in LoRa, a direct measurement of distance from the RSSI cannot be reliable, since the RSSI value mainly depends on the path-loss model that has been considered. It also depends on the channel characteristics. Therefore, RSSI based localization techniques are sensitive to channel parameters estimation [25]. Especially in LoRaWAN, the operating frequency changes dynamically i.e., frequency hopping. From the logged data observed huge change in RSSI between different frequencies and this affects the localization results [19], [35].Therefore, the RSSI values are collected and averaged in linear domain before passing it to the localization algorithm. As similar to multilateration it needs at least three or more base stations or gateways. For each gateways, a total of 400 RSSI readings were averaged. [19]

According to research article published by Hussein Kwasme [19], the received signal strength can be expressed as a function of frequency and antenna properties, but mainly distance:

$$P_r = f(d, f, Ap)$$

Where P_r is the received signal strength (power), d is the distance separating the Sender and Receiver, f is the operating frequency and A_p indicates antenna properties such as gain. The received signal may be affected due to several physical phenomenons shown in figure 3.9:
The RSSI can be modeled as [19]:

$$P_r = D + \psi + \alpha \tag{3.6}$$

Where D is the deterministic part of the signal which can follow many path-loss models (PLM) such as the single-slope model (log-normal model), stanford University Interim

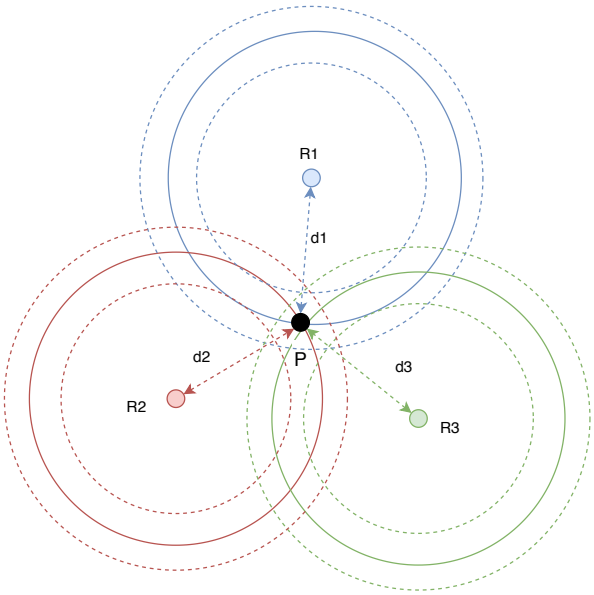


Figure 3.8 The dotted line in the figure indicates errors in RSSI measurements (uncertainty) [25]

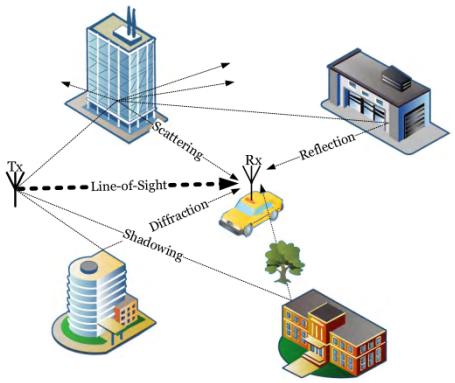


Figure 3.9 RF Received Signal Components [19]

(SUI) Model. Hata model, Okumura's Model etc.It is governed by the LOS component of the signal. ψ is called large scale fading (or Shadowing), it is modeled as a random variable predicting the variation in the received signal in an obstructed environment. α is the small scale fading (or Multipath) which is caused by the attenuation, diffraction, scattered and reflected copies of the signal arriving at the receiver. The Log-normal model is very common LOS PLM and is a single slope form which is given by [19]:

$$D|_{dBm} = K|_{dBm} - 10\gamma \log_{10}\left[\frac{d}{d_0}\right] \tag{3.7}$$

Increase in distance d (in meters), decreases deterministic received power. d_0 is some reference distance away from the transmitter at which the far-field transmission region of the antenna is considered. γ is the path-loss exponent (PLE) which is later explained in section 3.5.3. K is a constant that is governed by the operating frequency and the power

being transmitted by this antenna, and is given by [19]:

$$K_{dBm} = P_t|_{dBm} - C_{dB}. \tag{3.8}$$

Where

$$C_{dB} = 20\log_{10}(\frac{\lambda}{4\Pi d_0}) \tag{3.9}$$

Where P_t is the transmitted power, λ is the wavelength of the signal.

Path-Loss Exponent

The Equation 3.7 is a linear equation in the dB domain where γ is the slope of this line, referred to as PLE. This slope represents how "fast" the signal is attenuated (loss increase). This exponent is environment dependent and it's either determined imperially or from typical values tables for different environments. A method to determine γ is the linear Least Square Error (LSE). LSE is a curve fitting method that minimizes the squared error between the actual values (received power readings-RSSI) and the fitted curve (line) giving us the slope of this line (γ).
Once the distance is estimated using RSSI, then multilateration algorithm computes the geolocation co-ordinates using below algebraic equations [19]:
In 2-D space

$$(x - x_i)^2 + (y - y_i)^2 = d_i^2 \quad (i = 1, 2, ...n). \tag{3.10}$$

In 3-D space

$$(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = d_i^2 \quad (i = 1, 2, ...n). \tag{3.11}$$

where $x, y, and\ z$ are the unknown coordinates of the target. $xi, yi, and\ zi$ are the known coordinates of the ith reference points. di is the estimated distance between the target and the ith reference points. n is the number of reference points used. More reliable and practical way to implement RSSI based distance estimation was demonstrated in the articles [1] [15] as shown below:
Compute the RSSI with known distance d to the reference node & with experimented/ derived η value:

$$RSSI = -(10.\eta.\log(d) - A) \tag{3.12}$$

Where,
RSSI- Received signal strength (in dBm)
d- distance between reference node and gateway (in meters)
A- Absolute value of RSSI for the distance of 1 m (in dBm)
η (aka 'n')- Environmental co-efficient or path-loss exponent (no unit)
log denotes log 10

Substitute calculated RSSI value from (3.12) in below equation and compute environmental co-efficient η:

$$\eta = \frac{A - RSSI}{(10.\log(d))} \tag{3.13}$$

Using the above η, one can estimate more accurate distance value with RSSI values transmitted by at least three anchor nodes. In order to compute the distances of the target device, the following equation is used.

$$d = 10^{\left[\frac{A-RSSI}{10.\eta}\right]} \tag{3.14}$$

The distance d from 3.14 will be passed to Multilateration algorithm for position estimation. There are some experimental dataset available on article [15] to substitute for A & η values, which were empirically sampled according to various operating modes in LoRa standard as shown in below table 3.1 [15].

Table 3.1 LoRa channel characterization: A and η values for different LoRa modes. Specifically, mode 1 features BW=125 KHz, CR=4/5, and SF=12; on the other side, mode 10 has BW=500 KHz, CR=4/5, and SF=7 and Mode 5 is a trade-off among these two operating modes, where CR means Code Rate, BW means Bandwidth, SF means Spread Factor [15].

Environment	Mode 1		Mode 5		Mode 10	
	A	η	A	η	A	η
LOS indoor	-43.0	2.2	-41.0	1.9	-33.7	2.1
NLOS indoor	-45.3	3.1	-31.8	3.9	-26.1	4.2
LOS outdoor	-38.3	2.1	-35.6	2.1	-26.4	2.4
Combined LOS in and outdoor	-45.2	1.8	-41.6	1.8	-34.0	2.0

4 Implementation

In this chapter, the implementation steps will be demonstrated with the help of block diagrams and texts in detail.

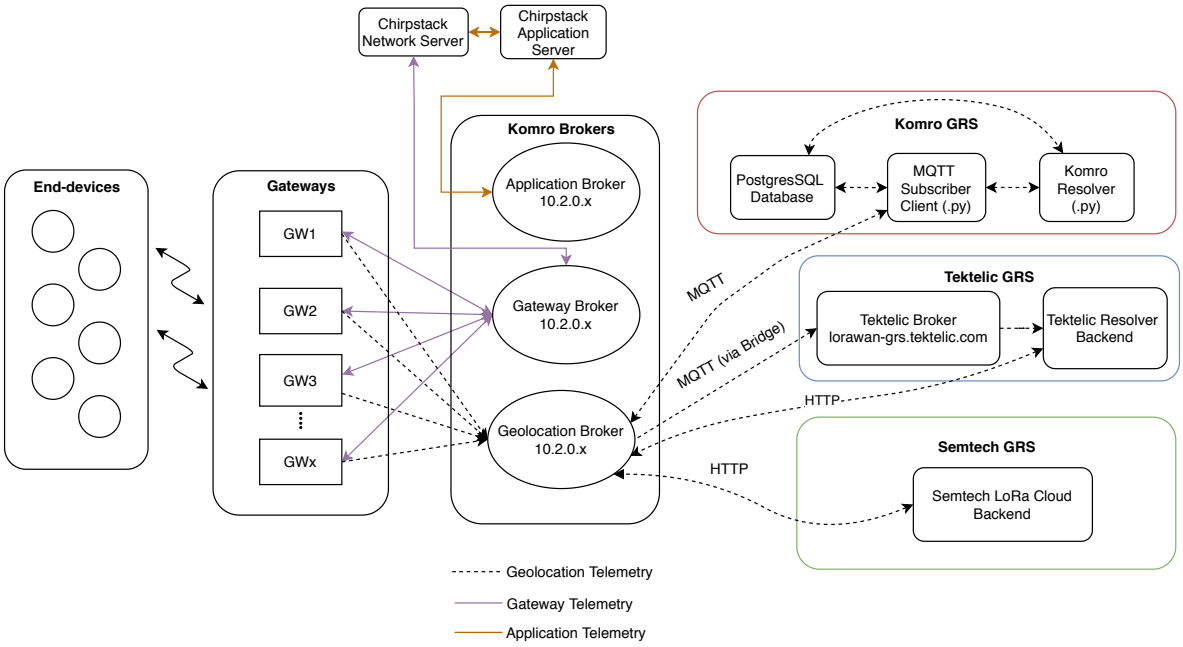


Figure 4.1 Block diagram of Komro GeoLoc Broker Architecture

4.1 Geolocation using Tektelic GRS

Implementation steps & challenges faced for Tektelic GRS will be shown here with detailed explanation. The Tektelic GRS is composed of three main components namely [12],

1. Fine Timestamp and Metadata Collector (FTMC)
2. Geolocation Resolver Back-end (GRB)
3. Graphical User Interface (GUI)

One can use **GRB** directly without **FTMC** by using RESTful API over **HTTP** Protocol which follows Request/Response model.

4.1.1 Installation of Geolocation Packages on Gateway

Usually, for every LoRaWAN version, there will be new Board Support Packages available (BSP) for upgrade in Tektelic Gateways. After a BSP upgrade, one should install the

following installation packages (ipks). The first four are to be installed for all gateways and the fifth one (i.e., the gl ipk) is special for geolocation featured gateways.

- packet forwarder
- mqtt bridge
- fe-fpga
- gpio-fpga
- gl ipk

4.1.2 Configuration of Gateway Bridges

In order to forward the TOA packets to the Tektelic Geolocation resolver server (GRS), one should configure the mqtt-bridge config file on gateway as shown below in 4.1 as **key:value** pairs. However, in real config files they are written without colon **(:)**

Tektelic GW Bridge config for direct connection

$$client\_id : tekmqttbridge005B92 \tag{4.1}$$
$$gateway\_mac : 647xxxxxxx005b92 \tag{4.2}$$
$$ns\_host : 10.x.x.x \tag{4.3}$$
$$ns\_type : brocaar \tag{4.4}$$
$$gw\_user : Txxxx5B92 \tag{4.5}$$
$$gw\_password : xxxxxxxxxxx \tag{4.6}$$
$$geoloc\_host : lorawan - grs.tektelic.com \tag{4.7}$$
$$geoloc\_topic : gateway/geoloc \tag{4.8}$$
$$gw\_geoloc\_user : 4gETxxxxxxxxuSuYofa \tag{4.9}$$
$$gw\_geoloc\_password : xxxxxxxxxxxxxxx \tag{4.10}$$
$$semtech\_udp\_listen\_port : 1700 \tag{4.11}$$
$$snmp\_agent : 127.0.0.1 \tag{4.12}$$
$$snmp\_community : set \tag{4.13}$$

In the config 4.1, the value in 4.1 is MQTT client ID parameter. Its is a Required parameter. It should be unique. The line 4.2 is Gateway MAC address (Required). The line 4.3 is Network Server host address. It is an Optional parameter. The line 4.4 is Network Server type (Optional). For Chirpstack Network Server, it should be **brocaar**. The line 4.5 is gateway username obtained from the Network Server (Optional). The line 4.6 is gateway password obtained from the Network Server (Optional). The line 4.7 is the IP address of the destination server, here it is IP of the Tektelic GRS Server. The line 4.8 is just a mqtt topic for each public networked gateways with a format **gateway/geoloc**. The line 4.9 & 4.10 are generated by Tektelic GRS GUI after adding the gateway and it is copied to the gateway bridge. The line 4.11 is UDP bridge listen port (Optional). Valid

ports are in (0, 65535] range. The line 4.12 is SNMP agent address (Optional). The line 4.13 is SNMPv1 community (Required). By implementing this config file in gateways, the gateways will send the TOA packets directly to Tektelic GRS server (via public internet). Then, by accessing Tektelic GRS web GUI, one can see all the received TOA packets & estimate geolocation of devices.

4.1.3 Tektelic GRS via MQTT

Tektelic Web-GUI

The Tektelic GRS has a web-based GUI platform where one can add an user account with E-Mail ID & Password and send activation link to that concerned E-Mail ID. Before that one has to purchase a subscription plan from Tektelic. After that, tektelic will add one user and send the first activation link through which multiple accounts would be added and activated over an E-Mail link (# depends on the subscribed package).

Log in to Tektelic GRS :*https* : *//lorawangrs.tektelic.com/home*
UserName :*xxxxxx*
Password :*xxxxxx*

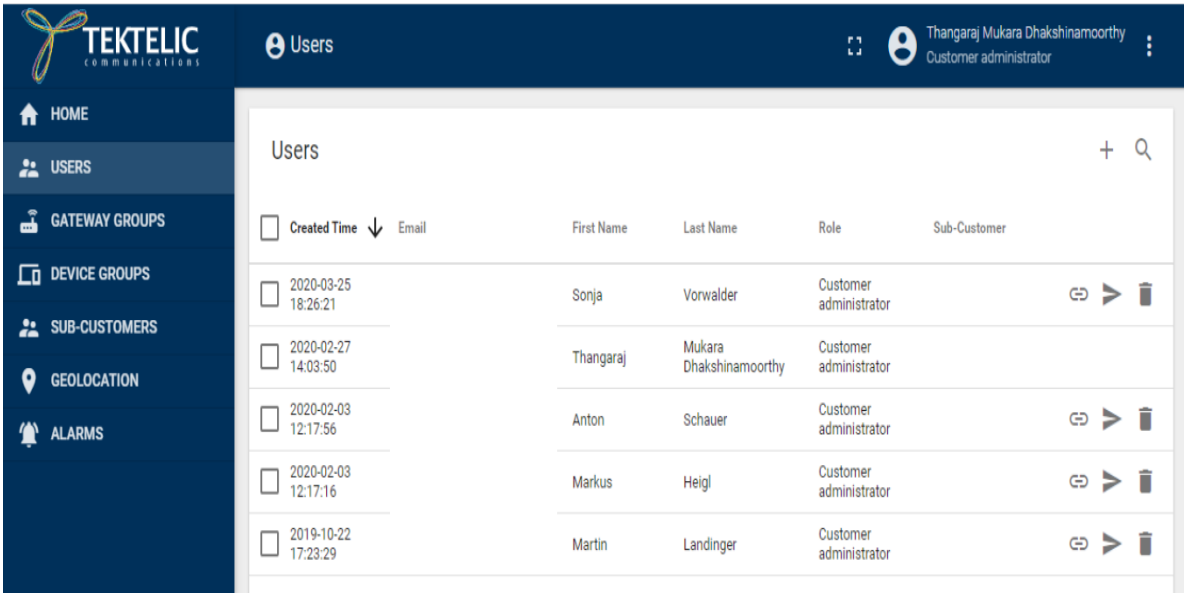


Figure 4.2 User Page [12]

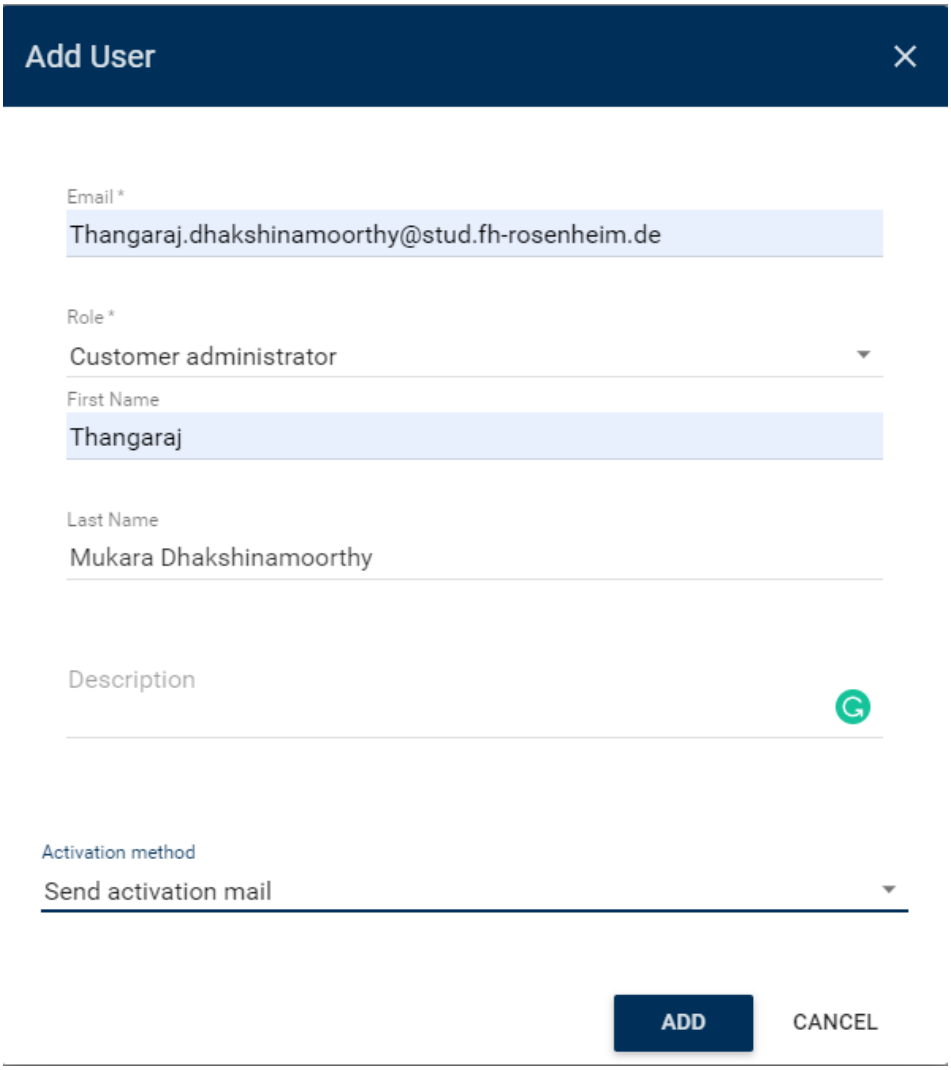
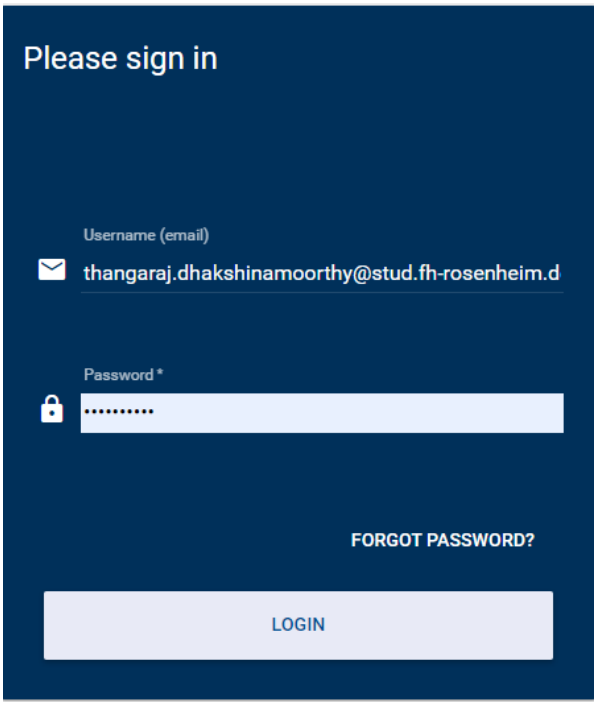
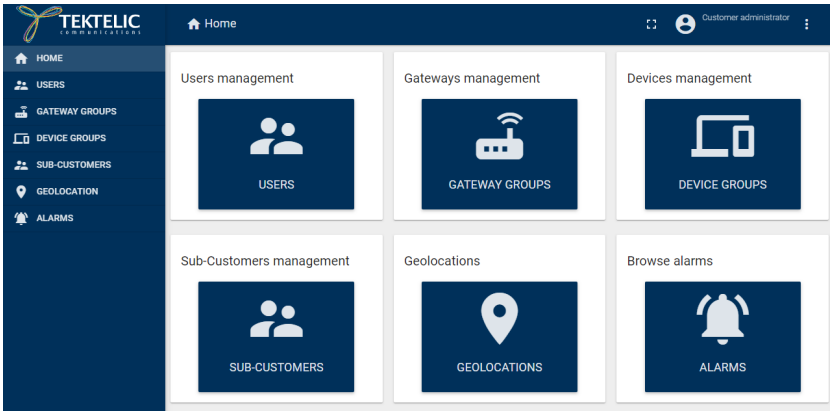


Figure 4.3 Add new user's E-Mail, Name and send activation link to his/her E-Mail ID through which one can set password and log-in [12]

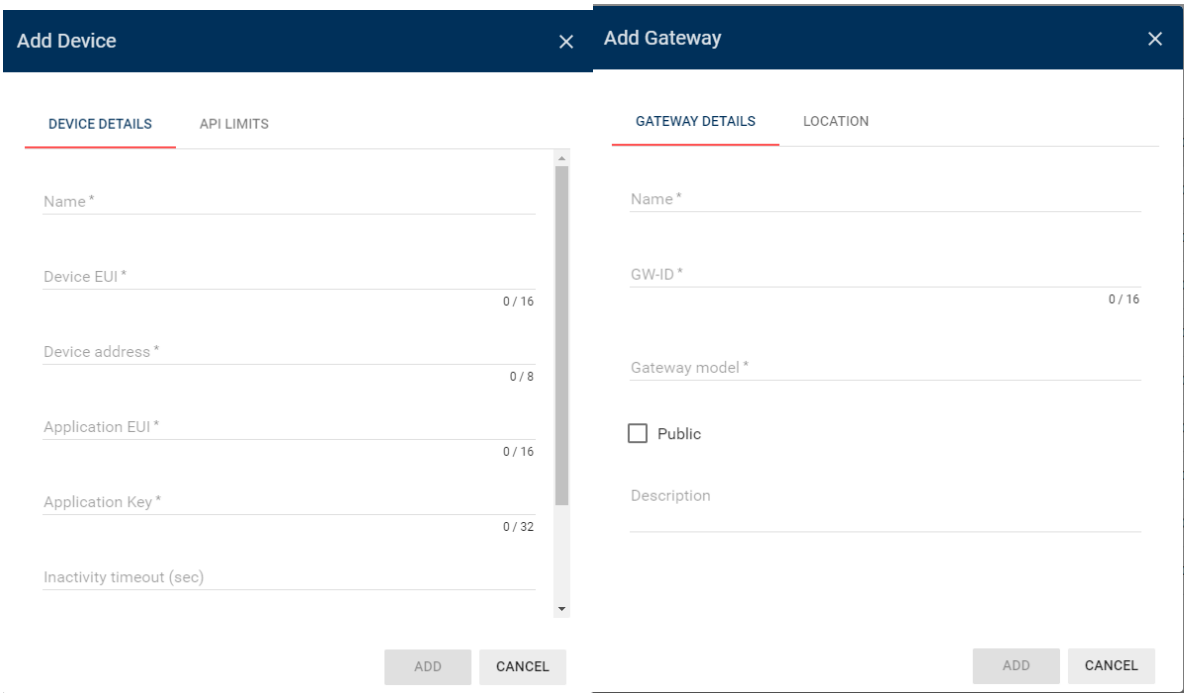


(a) Log-In Page



(b) Homepage

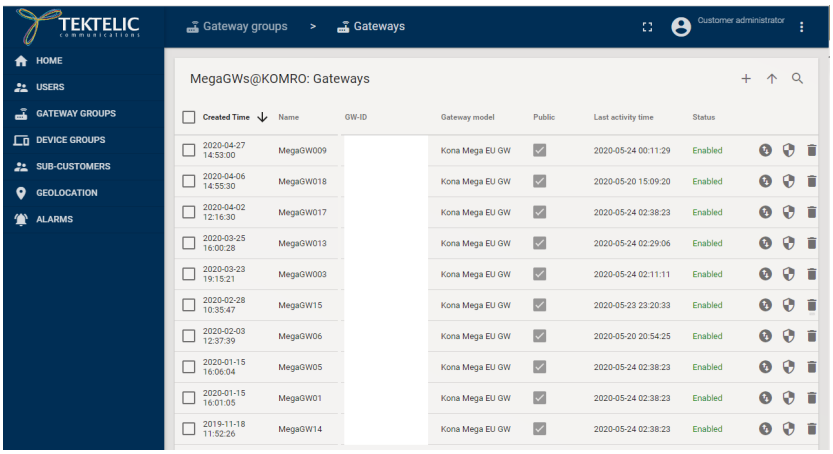
Figure 4.4 Tektelic GRS GUI Homepage [12]



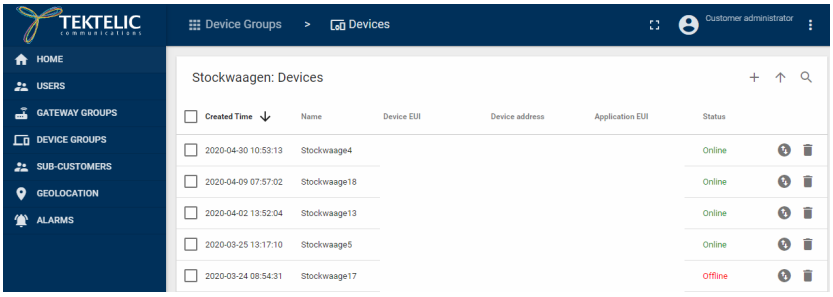
(a) Commission some devices on Tektelic GRS to estimate the geolocation

(b) Commission some devices on Tektelic GRS to estimate the geolocation

Figure 4.5 Add devices & gateways [12]

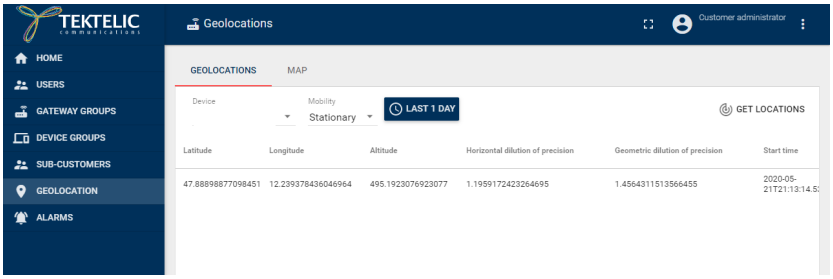


(a) Densify gateway networks in GRS as much as possible

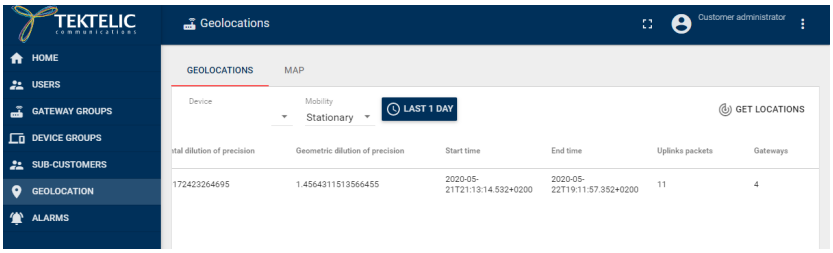


(b) Densify devices and create groups for each device types in GRS

Figure 4.6 Densify gateways & devices [12]



(a) Geolocation tab_stationaryDevice_1



(b) Geolocation tab_stationaryDevice_2

Figure 4.7 Select the device EUI and its mobility status and press "Get Location" button on top right corner. Here the mobility status is "Stationary" where it gathers multiple UL packets from multiple gateways over a range of time and resolves the geolocation

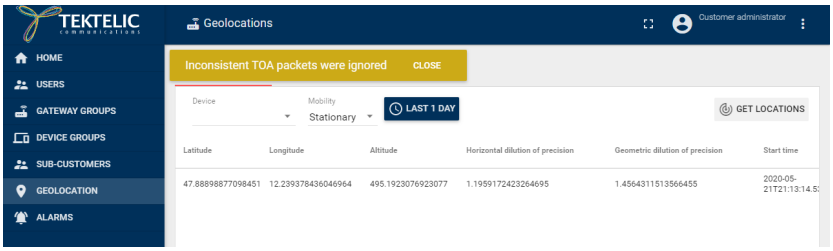
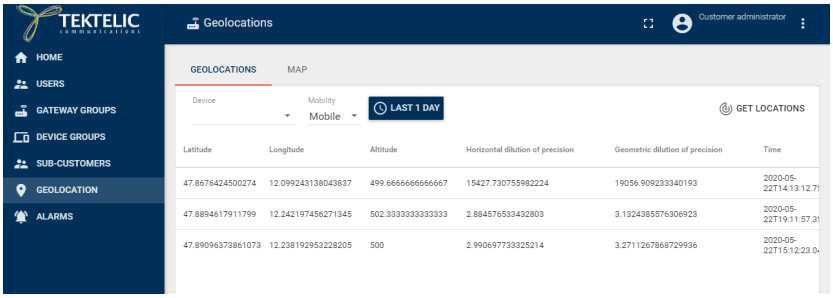
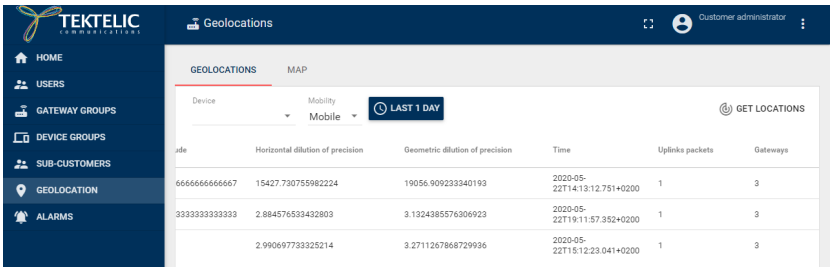


Figure 4.9 Estimate Geolocation for Stationary Device by ignoring Inconsistent TOA packets [12]



(a) Geolocation tab_MobileDevice_1



(b) Geolocation tab_MobileDevice_2

Figure 4.10 Select the device EUI and its mobility status and press "Get Location" button on top right corner. Here the mobility status is "Mobile" where it takes single UL packets from 3 gateways at a time instance and resolves the geolocation for that particular time instances

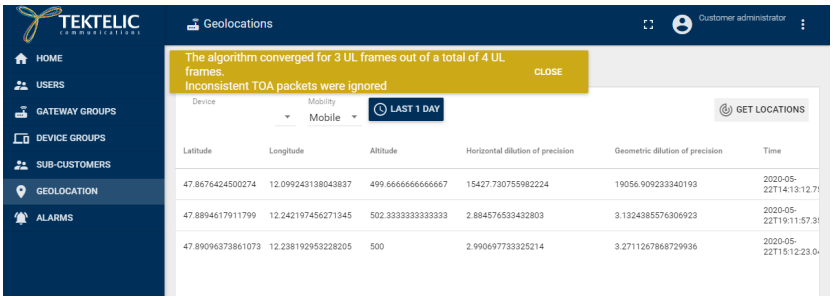


Figure 4.12 Estimate Geolocation for Mobile Device by ignoring Inconsistent TOA packets [12]

Mosquitto Broker Bridge solution

Tektelic GRS is on public internet. But most of the gateways in Komro are in private networks. As a result only few gateways could reach the tektelic GRS directly over public internet, as a result not enough gateways available on tektelic GRS. So that a broker bridge is built between private networked gateways and local broker (komro broker) upto Tektelic GRS. The below configuration [4.1.3] [4.28] are for communication from one gateway upto Tektelic GRS. The same will be repeated for n number of private networked gateways.

Tektelic GW Bridge config for indirect connection

Inside **root@lora-gw018: # cat /etc/default/mqtt-bridge.conf**

$$client\_id : tekmqttbridge005BB4 \tag{4.14}$$
$$gateway\_mac : 647xxxxxxx005bb4 \tag{4.15}$$
$$ns\_host : 10.x.x.x \tag{4.16}$$
$$ns\_type : brocaar \tag{4.17}$$
$$gw\_user : Txxxx5BB4 \tag{4.18}$$
$$gw\_password : xxxxxxxxx \tag{4.19}$$
$$geoloc\_host : 10.x.x.x \tag{4.20}$$
$$geoloc\_topic : gateway/geoloc/647xxxxxxx005bb4 \tag{4.21}$$
$$gw\_geoloc\_user : bJ0W8KxxxxxjXHjwJ3 \tag{4.22}$$
$$gw\_geoloc\_password : xxxxxxxxxxxxxx \tag{4.23}$$
$$semtech\_udp\_listen\_port : 1700 \tag{4.24}$$
$$snmp\_agent : 127.0.0.1 \tag{4.25}$$
$$snmp\_community : set \tag{4.26}$$
$$snmp\_mib\_dir : /usr/share/snmp/mibs \tag{4.27}$$

In the configuration 4.1.3, the value in 4.14 is MQTT client ID parameter. Its is a Required parameter. It should be unique. The line 4.15 is Gateway MAC address (Required). The line 4.16 is Network Server host address. It is an Optional parameter. The line 4.17 is Network Server type (Optional). For Chirpstack Network Server, it should be **brocaar**. The line 4.18 is gateway username obtained from the Network Server (Optional). The line 4.19 is gateway password obtained from the Network Server (Optional). The value in line 4.20 is the IP address of the destination server, here it is IP of the central komro broker. The line 4.21 is just a mqtt topic for each private networked gateways with a format **gateway/geoloc/$gatewayEUI**. The line 4.22 & 4.23 are generated by Tektelic GRS GUI after commissioning a device and it is copied to the gateway bridge. The line 4.24 is UDP bridge listen port (Optional). Valid ports are in (0, 65535] range. The line 4.25 is SNMP agent address (Optional). The line 4.26 is SNMPv1 community (Required). The line 4.27 is Additional MiB directory (Optional). By implementing this config file in gateways, the gateways will send the TOA packets directly to Tektelic GRS server (via public internet). Then, by accessing Tektelic GRS web GUI, one can see all the received TOA packets & estimate geolocation of devices.

Komro (mosquitto) broker bridge config

Inside **komro@komro:/etc/mosquitto/conf.d$ cat tekmquttbridge005BB4.conf**

$$connection : tekmqttbridge005BB4 \tag{4.28}$$
$$try\_private : false \tag{4.29}$$
$$cleansession : true \tag{4.30}$$
$$notifications : false \tag{4.31}$$
$$bridge\_attempt\_unsubscribe : false \tag{4.32}$$
$$local\_clientid : bridge\_local\_id \tag{4.33}$$
$$local\_username : Txxxx6A78 \tag{4.34}$$
$$local\_password : xxxxxxxx \tag{4.35}$$
$$remote\_username : jf1ZxxxxxxrU8y9j94 \tag{4.36}$$
$$remote\_password : xxxxxxxxxxxxxxx \tag{4.37}$$
$$address : lorawan - grs.tektelic.com \tag{4.38}$$
$$topic : "" \ out \ 0 \ gateway/geoloc/647xxxxxxx005bb4 \ gateway/geoloc \tag{4.39}$$

In configuration 4.28, the line 4.28 is connection name and it must have a unique name. The line 4.29 is is try_private flag. If try_private is set to true, the bridge will attempt to indicate to the remote broker (Tektelic GRS) that it is a bridge not an ordinary client. If successful, this means that loop detection will be more effective and that retained messages will be propagated correctly. The line 4.30 is clean session variable. When it is set to true, when the bridge disconnects for any reason, all messages and subscriptions will be cleaned up on the remote broker. When set to false, the subscriptions and messages are kept on the remote broker, and delivered when the bridge reconnects. The line 4.31 is notification variable. If set to true, publish notification messages to the local and remote brokers giving information about the state of the bridge connection. The line 4.32 is bridge attempt unsubscribe variable. If a bridge has topics that have "out" direction, the default behaviour is to send an unsubscribe request to the remote broker on that topic. This means that changing a topic direction from "in" to "out" will not keep receiving incoming messages. Sending these unsubscribe requests is not always desirable. The line 4.33 is local clientID. Set the clientid to use on the local broker. If not defined, this defaults to local.<remote_clientid>. The line 4.34 is local username variable. Configure the username to be used when connecting this bridge to the local broker. This may be important when authentication and ACLs are being used. The line 4.35 is local password. Configure the password to be used when connecting this bridge to the local broker. This may be important when authentication and ACLs are being used. The line 4.36 is a username for the bridge. The line 4.37 is a password for the bridge. The line 4.38 is to specify the address and optionally the port of the bridge to connect to. This must be given for each bridge connection. If the port is not specified, the default of 1883 is used. The line 4.39 is topic remapping variable, defines a topic pattern to be shared between the two brokers. The patter is topic pattern [[[out | in | both] qos-level] local-prefix remote-prefix] [5].

Mosquitto Broker Bridges

The Mosquitto broker bridge for each gateway has been created on Komro GeoLoc Broker (10.x.x.x) & all the Mosquitto Broker Bridges shown in figure 4.13 were stored inside them (docker environment)

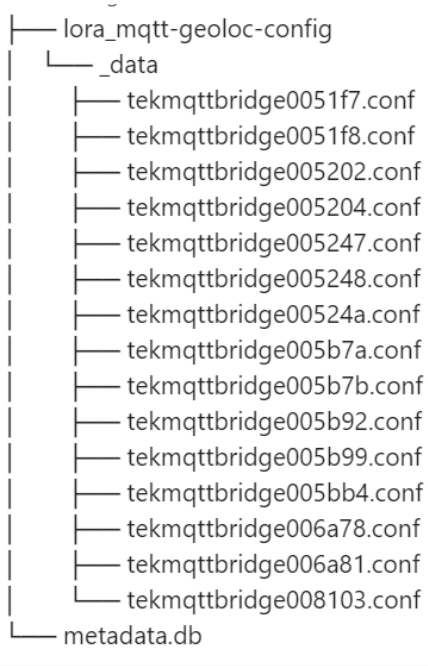


Figure 4.13 A pile of Mosquitto Broker Bridges created for each gateways for topic remapping and attaching the gateway specific $Username & $Password to reach Tektelic GRS

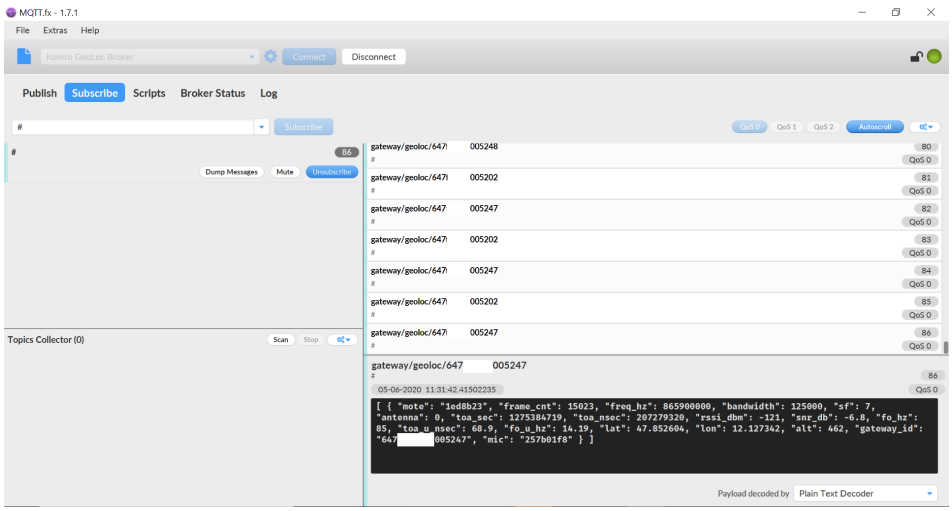


Figure 4.14 Gateway specific geoloc packets on Geoloc Broker Host- subscribed from MQTT.Fx GUI Tool [16]

Topic Remapping

The gateway packets are differentiated from each other using geoloc topics and in order to reach Tektelic GRS, they need to be transformed into different form attached with $Username and $Password generated in Tektelic GRS during commissioning.

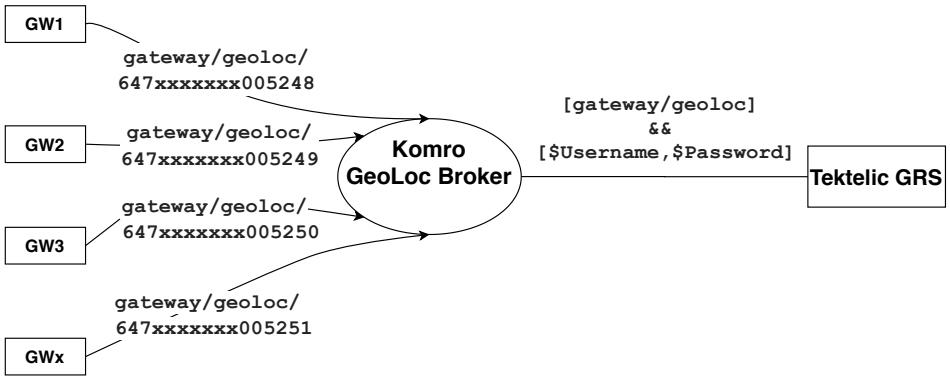


Figure 4.15 The Topic from each gateways are remapped to *gateway/geoloc* along with gateway specific Username & Password indicated by prefix $

4.1.4 Tektelic GRB via HTTP

The Tektelic Georesolver Backend (GRB) can be used directly through HTTP protocol without storing the geolocation packets on Tektelic's server but it needs some implementations at local server like Request & Response frameworks. It is achieved by **GRB API** provided by Tektelic (upon request). The End-Point URL used for this HTTP request is `https://lorawan-grs.tektelic.com/api/geo/localization/gr`. This chapter contains three parts

1. Request headers
2. Request body
3. Response JSON

Request headers:

The Request header must contain the following format:

- Content-Type: application/json
- Authorization: Basic **<credentials>**

where **<credentials>**is the base64 encoding of username and password joined by a colon. To obtain a username/password pair, create a device group in the GRS and use its credentials for this API:

- Username: Device group username
- Password: Device group password

This is a temporary method of obtaining credentials for the GRB API and will change in the future [12].

Request message:

The request message is carried in the body of the HTTP request and is a JSON string with the following format.

Listing 4.1 Geolocation packets in JSON format

```
{
  "dev_id": device ID ("DevEUI", "DevAddr", etc.),
  "mobility": mobility state of the end-device (0:stationary, 1:mobile),
  "toa_packets": [
    {
 "frame_cnt": UL frame count,
 "freq_hz": channel frequency in Hertz,
 "bw_hz": bandwidth in Hertz,
 "sf": spreading factor,
 "rssi": RSSI in dBm,
 "snr": SNR in dB,
 "toa_sec": the seconds part of the TOA,
 "toa_nsec": the nanoseconds part of the TOA,
 "fo_hz": estimated frequency offset of the received packet in Hertz,
 "toa_u_nsec": uncertainty in the TOA estimate in units of nanoseconds,
 "fo_u_hz": uncertainty in the frequency offset estimate in units of Hertz,
 "gw_id": "gateway EUI",
 "ant_ind": antenna index (0 or 1),
 "ant_lat": GPS latitude of the antenna,
 "ant_lon": GPS longitude of the antenna,
 "ant_alt": GPS altitude of the antenna
    }
  ]
}
```

The "geolocation packets" field is an array of geoloc packet objects. Each geoloc packet object is a structure that contains the fine timestamp (i.e., toa_nsec) and other metadata corresponding to a particular UL frame received by a particular antenna of a particular gateway. The elements of this array can belong to multiple UL frames. The resolver will automatically match and group all TOA packets that belong to the same UL frame. The "mobility" status field determines how the case of multiple UL frames is treated by the resolver. If the mobility is set to 1 (i.e., mobile end-device), the resolver generates one location estimate per each UL frame. However, if the mobility is set to 0 (i.e., stationary end-device), the resolver generates a single location estimate using the TOA packets from all UL frames.

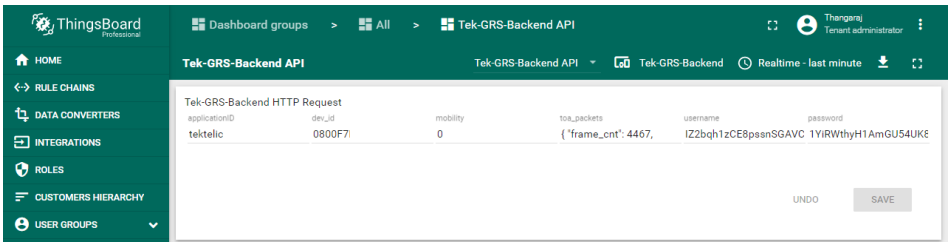
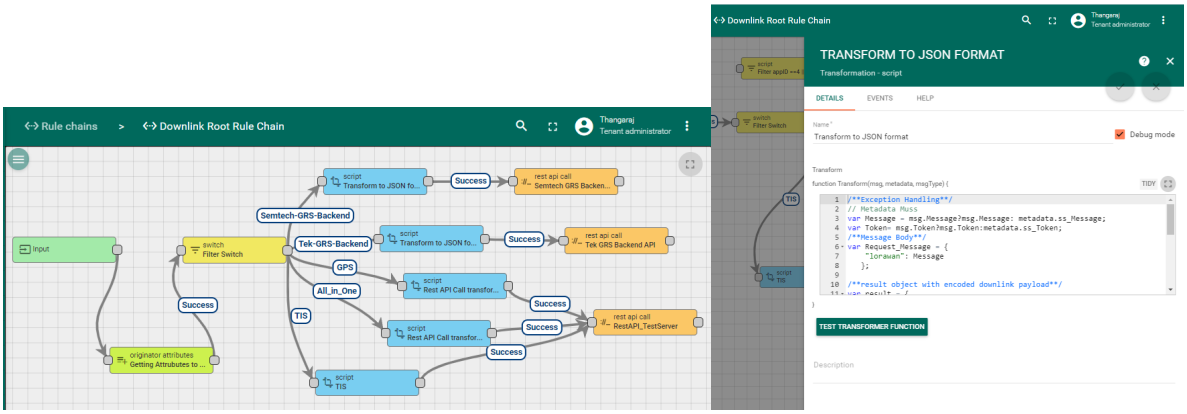


Figure 4.16 Post HTTP Request from Dashboard [12]



(a) Thingsboard_Downlink Rule Chain **(b)** Thingsboard_HTTP Request Encoder

Figure 4.17 Request side of Application Platform (Thingsboard) [12]

Response message:

The HTTP request message is posted as shown above, now the corresponding response message should be received in the same application platform where the request was posted and the body of HTTP response is decoded using the Uplink Data Converter. The Tektelic GRB generates two different response messages depending on the mobility status of end-device in the corresponding request message.

Response message for a mobile end-device:

The response message for a mobile end-device is a JSON string with the following format.

Listing 4.2 HTTP Response message for mobile device in JSON format

```
{
  "dev_id": device ID ("DevEUI", "Devaddr", etc.),
  "loc_est": [
    {
 "lat": estimated latitude of the end-device,
 "lon": estimated longitude of the end-device,
 "alt": estimated altitude of the end-device,
"ts_msec": timestamp of the UL frame used for calculating this location
    estimate in units
of milliseconds,
 "hdop": horizontal dilution of precision at the estimated location,
 "gdop": geometric dilution of precision at the estimated location,
```

```
"number_of_ul_packets_used": the number of UL frames used by the resolver to
calculate the solution. It is always equal to 1 in the case of a mobile device,
 "number_of_gateways_used":  the number of different gateways used by the resolver to
calculate the solution.
    }
  ],
  "errors":[
 "error message"
  ],
  "warnings":[
 "warning message"
  ]
}
```

The "loc_est" field is an **array of location estimate objects**. Each location estimate object contains an estimate of the end-device location (lat, lon, alt), the time at which the end-device was estimated to be in that location (ts_msec) and some other information. The number of elements in the loc_est array is less than or equal to the number of UL frames sent in the request message (through the automatic matching and grouping process).

Response message for a stationary end-device:

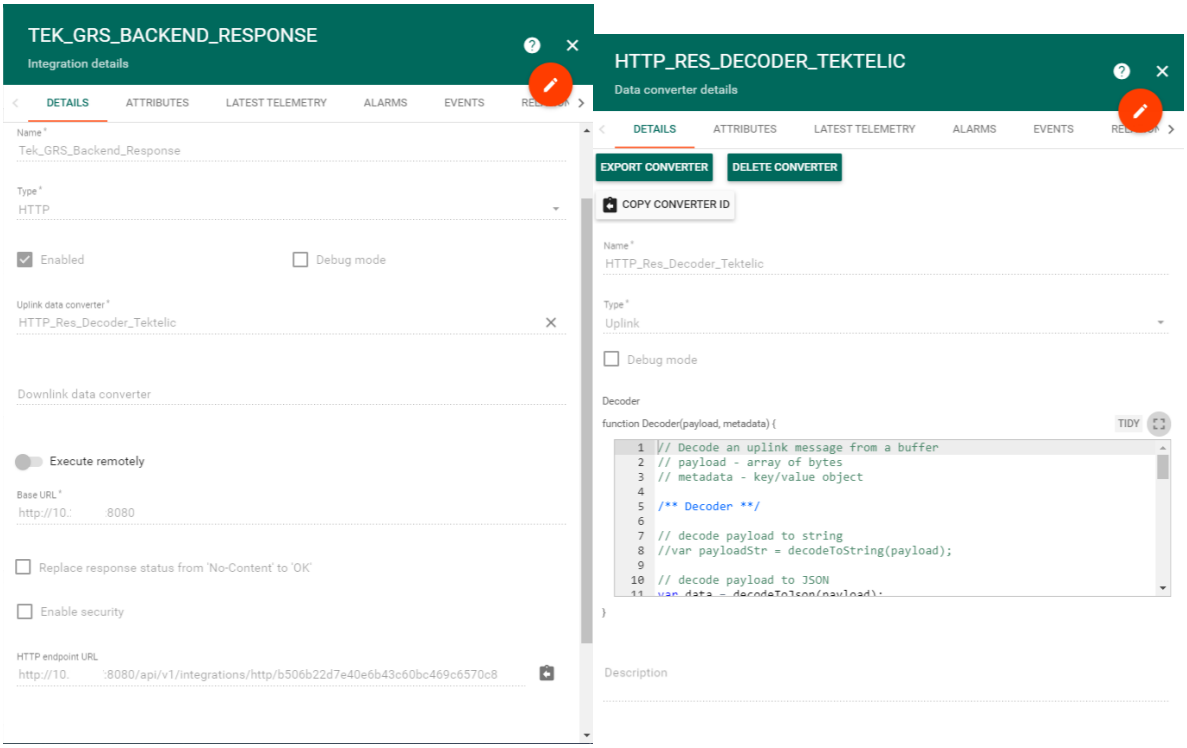
The response message for a stationary end-device is a JSON string with the following format.

Listing 4.3 HTTP Response message for stationary device in JSON format

```
{
  "dev_id": device ID ("DevEUI", "Devaddr", etc.),
  "loc_est": {
 "lat": estimated latitude of the end-device,
 "lon": estimated longitude of the end-device,
 "alt": estimated altitude of the end-device,
 "ts_msec_start": timestamp of the earliest UL frame used for calculating this location estimate in units of milliseconds,
 "ts_msec_end": timestamp of the most recent UL frame used for calculating this location estimate in units of milliseconds,
 "hdop": horizontal dilution of precision at the estimated location,
 "gdop": geometric dilution of precision at the estimated location,
 "number_of_ul_packets_used": the number of UL frames used by the resolver to calculate the solution,
"number_of_gateways_used": the number of different gateways used by the resolver to calculate the solution.
  },
  "errors":[
 "error message"
```

```
  ],
  "warnings":[
 "warning message"
  ]
}
```

The “loc_est” field, in this case, is a **single object** (as opposed to an array of objects in the mobile case). This object contains an estimate of the end-device location (lat, lon, alt) using all TOA packets (from all UL frames) in the request message. It also contains the timestamp (in milliseconds) of the earliest UL frame found in the request message, the timestamp (in milliseconds) of the latest UL frame found in the request message, and some other information.



(a) Subscribe HTTP Responses from Tektelic GRB **(b)** Decode Requested HTTP message into JSON format accepted by Tektelic GRB

Figure 4.18 Response side of Application Platform (Thingsboard) [12]

HTTP POST Request to Tektelic GRB using python:

The same above implementation can be realised using python, without a need for application platform like Thingsboard.

Listing 4.4 HTTP POST Request to Tektelic GRB using python

```
import requests                    #http library
import paho.mqtt.client as client #mqtt library
# Required HTTP parameters: End points url, username, password
```

```
#End points url
url_tektelicGRB= 'https://lorawan-grs.tektelic.com/api/geo/localization/gr'
#Username from tektelic GRS device group
tek_username = 'IZ2bxxxxxxxVC'  #base64 string to be copied from tektelic
                                                        #GRS account
#Password from tektelic GRS device group
tek_password = '1Yixxxxxxxx82'  #base64 string to be copied from tektelic
                                                      #GRS account

# MQTT Topics to publish geolocation results from tektelic GRB
topic_response = "geolocation/httptektelic/jsonres/" + str(device_addr)

# Extract/filter dataframe acc. to data filters
request_body = extract_df(dev_addr, filter_mode, num_of_frames, split_date)

# HTTP headers
# check the header parameters with tektelic GRS account, because it is
                                                         #dynamic
basicAuthCredentials = ('Basic'+tek_username+':'+tek_password)
HEADERS = {'Content-Type': 'application/json', 'Authorization':
    basicAuthCredentials}

# Send HTTP POST request to server and attempt to receive a response
r = requests.post(url_tektelicGRB, headers=HEADERS, data=request_body)
response_json = json.dumps(r.json())

# publish the JSON response to Geoloc Broker
client.publish(topic_response, response_json, 1, 1)
```

4.2 Geolocation using Semtech GRS

Implementation steps & challenges faced for Chirpstack NS and Semtech GRS will be shown here with detailed explanation. The Chirpstack NS uses Semtech GRS Back-end. It could be integrated with Chirpstack NS (in MQTT). Also, HTTP integration is possible seperately provided that the request/response arrangements are made.

4.2.1 Semtech GRS under Chirpstack NS via MQTT

This is currently not supported by Tektelic gateways, therefore only the architecture of Semtech GRS in Chirpstack NS is shown below in figure 4.19.

4.2.2 Semtech GRB via HTTP

In order to use Semtech GRS in HTTP protocol, an API key (token) to authenticate the user with each HTTP query is required. For that an user account should be created in

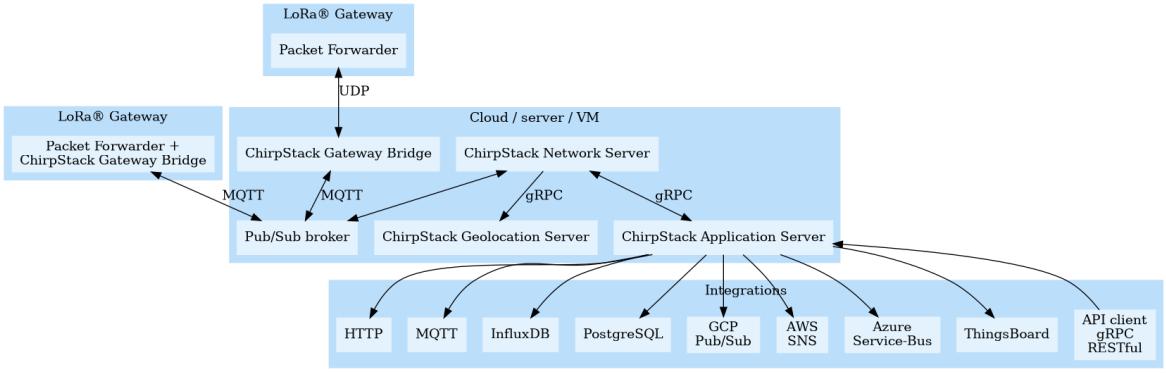


Figure 4.19 Geolocation architecture in chirpstack network server [4]

Semtech Portal under `https://www.loracloud.com/portal/geolocation` [6]

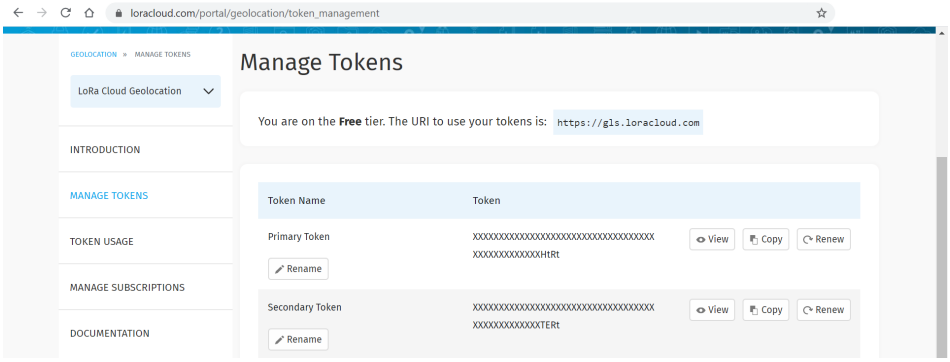


Figure 4.20 API Token should be copied and sent with each Geolocation query (HTTP request) [4]

HTTP Request & Response Format

The Semtech LoRa Cloud Back-end accepts two types of request according to API v3 (LoRa® TOA/RSSI). They are Singleframe & Multiframe Requests. As seen in Tektelic HTTP formats, here also there are three parts such as Request headers, Request body & Response JSON [6].

Request Headers

Listing 4.5 HTTP POST- Request Headers

```
1 Accept: application/json
2 Ocp-Apim-Subscription-Key: Required. Access token
```

Request Body

Listing 4.6 HTTP POST- Request Body

```
1 # Singleframe
2 {
3    "gateways": ARRAY,  // Required. Array of Gateway objects.
```

```
   "frame":    OBJECT, // Required. Frame object.
   "params":   OBJECT  // Optional. Parameters object.
}
# Multiframe
{
   "gateways": ARRAY,  // Required. Array of Gateway objects.
   "frames":   ARRAY,  // Required. Array of Frame object.
   "params":   OBJECT  // Optional. Parameters object.
}
# Parameters (params)
{
  "locAlgType":        STRING,  // "TDOA_ALG" or "RSSI_ALG"
  "doRssiTdoaCombine": BOOLEAN
}
```

Response JSON

Listing 4.7 HTTP POST- Response Body

```
# Singleframe & Multiframe JSON response
{
     "result":{
        "numUsedGateways":7,
        "HDOP":0.3,
        "algorithmType": "RssiTdoaCombined",
        "locationEst":{
           "latitude":51.50053,
           "longitude":-0.14842,
           "toleranceHoriz":92
        }
     },
     "warnings":[]
  }
```

HTTP POST Request to Semtech GRB using python:

The HTTP POST request implementation can be realised using python, without a need for application Platform like Thingsboard.

Listing 4.8 HTTP POST Request to Semtech GRS (LoRa Cloud) using python

```
import requests                    #http library
import paho.mqtt.client as client #mqtt library
# Required HTTP parameters: End points url for both single- & multi-frame
      #formats in v3 API (supports TDOA + RSS Algorithms) & API Token
#End points url
url_v3_singleframe = 'https://gls.loracloud.com/api/v3/solve/singleframe'
url_v3_multiframe = 'https://gls.loracloud.com/api/v3/solve/multiframe'
```

```
#API Token, a base64 string to be copied from semtech lora cloud account
API_Token = 'AQEACklyxxxxxxxxxxxxxxxxxxxxvMPSw7pdHtRt'

# MQTT Topics to publish geolocation results from semtech GRS (LoRa Cloud)
topic_singleframe_response = "geolocation/httpsemtech/v3singleframe/jsonres
                                                    /" + str(device_addr)
topic_multiframe_response = "geolocation/httpsemtech/v3multiframe/jsonres/"
                                                      + str(device_addr)

# Extract/filter dataframe acc. to data filters
request_body = extract_df(dev_addr, filter_mode, num_of_frames, split_date)

# HTTP headers, check the header paramaters with semtech lora cloud (GRS)
                                          #website, because it is dynamic
HEADERS = {'Content-Type': 'application/json', 'Ocp-Apim-Subscription-Key':
                                                              API_token}

# Send HTTP POST request for singleframe which takes only one UL frame
                                                            #(suffix s)
rs = requests.post(url_v3_singleframe, headers=HEADERS, data=request_body)
response_json = json.dumps(rs.json())

# publish the JSON response to Geoloc BRoker
client.publish(topic_singleframe_response, response_json, 1, 1)

# Send HTTP Post request for multiframe which takes more than one UL frames
                                                          #(suffix m)
rm = requests.post(url_v3_multiframe, headers=HEADERS, data=request_body)
response_json = json.dumps(rm.json())

# publish the JSON response to Geoloc Broker
client.publish(topic_multiframe_response, response_json, 1, 1)
```

4.3 Geolocation using Komro GRS (local)

So far the third party cloud-based Geolocation resolvers were used and now the local backend Geo-Resolver will be built and used for geolocation estimation. The local georesolver will be called as Komro GRS. The expansion of Komro GRS is Komro Geolocation Resolver Server. It consists of three components namely [4.1] [5.1],

1. MQTT Subscriber (geoloc packet collector)
2. PostgreSQL Database
3. Resolver

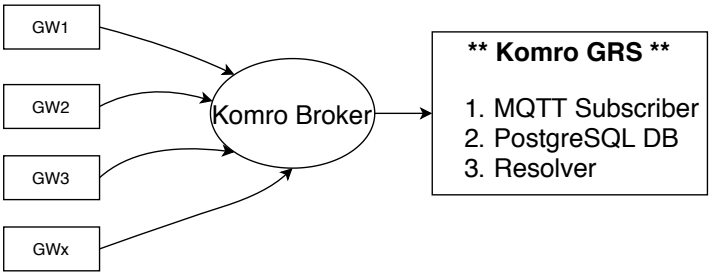


Figure 4.21 Block diagram of Komro GRS components [4.1] [5.1]

4.3.1 MQTT Subscriber (geoloc packet collector)

The MQTT Subscriber is similar component as FTMC in tektelic GRS. As described in the flowchart 4.22, the python library paho mqtt [28] is used to subscribe incoming mqtt messages in the komro broker. The python library psycopg [28] is used as adapter for connecting to PostgreSQL DB. The mqtt parameter used are

Listing 4.9 MQTT parameters to publish the results

```
import paho.mqtt.client as mqtt # import library
# MQTT parameters to publish the results
geoloc_host = '10.x.x.x'
mqtt_port= 1884
qos = 0
retain = False
# mqtt connect function to establish a connection with mqtt broker
def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))
client = mqtt.Client()
client.on_connect = on_connect
client.connect(geoloc_host,mqtt_port, 60)
client.loop_start()
```

4.3.2 PostgreSQL DB

The PostgreSQL DB [24] should be installed and the respective tables with columns of pre-defined data types as shown in figure 4.24 should be created either using python psycopg2 module or directly using pgadmin web-GUI tool as shown in figure 4.24. The project tree of the PostgreSQL DB is shown below in figure 4.23, it contains three tables called **geoloc_packets**, **geoloc_results** and **miscellaneous** under Geolocation_DB.

Listing 4.10 PostgreSQL parameters to connect, query & store geoloc packets

```
import psycopg2        # import library
# Postgres DB parameters
postgres_host= '10.x.x.x'
postgresql_port = '5432'
DB_NAME= 'Geolocation_DB'
```

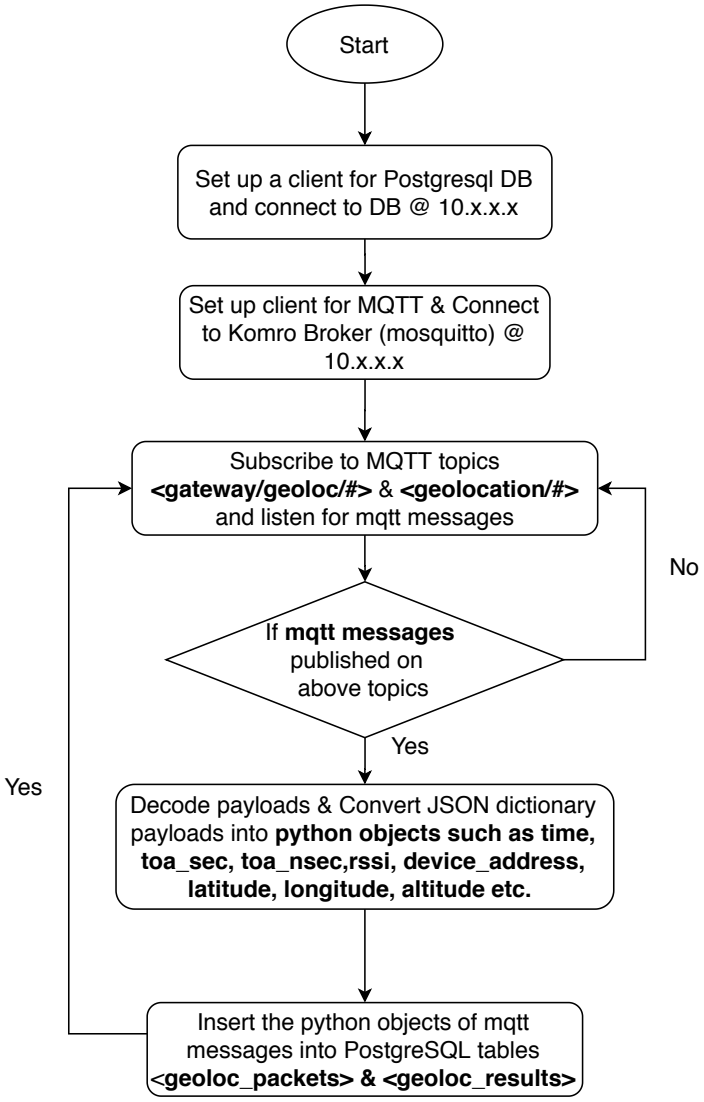


Figure 4.22 Block diagram of geoloc packet collector

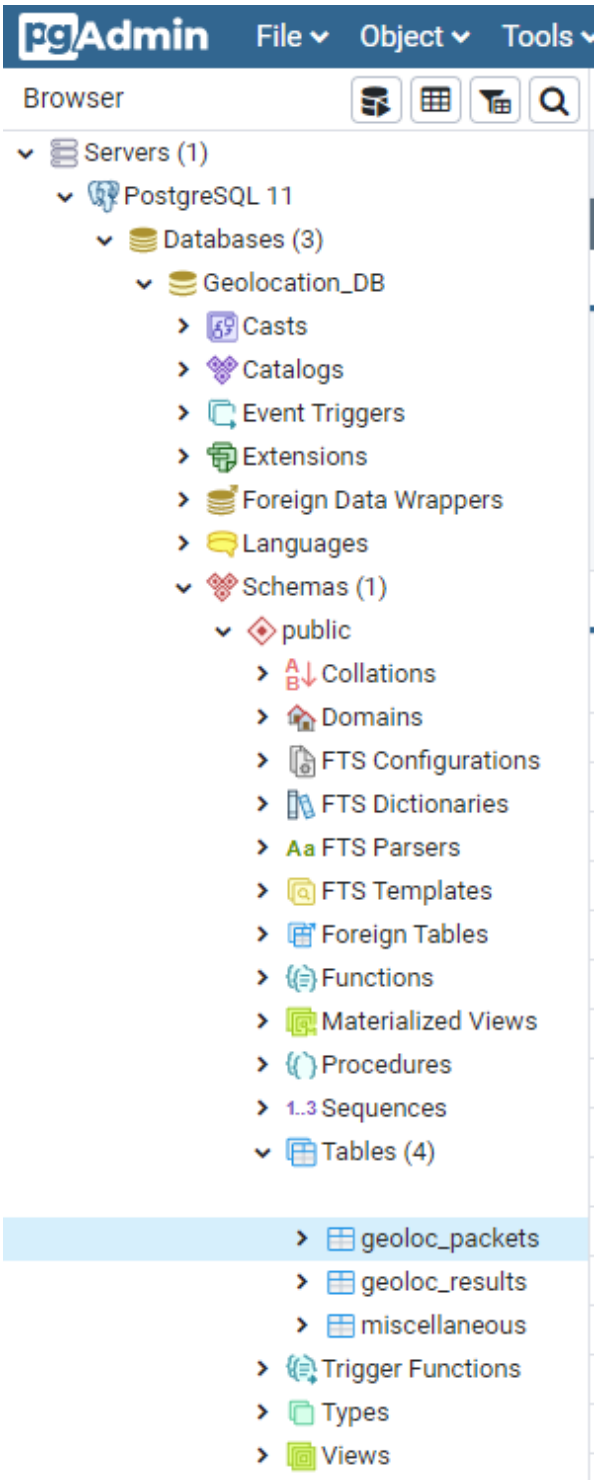


Figure 4.23 Table hierarchy of geolocation DB in PostgreSQL DB

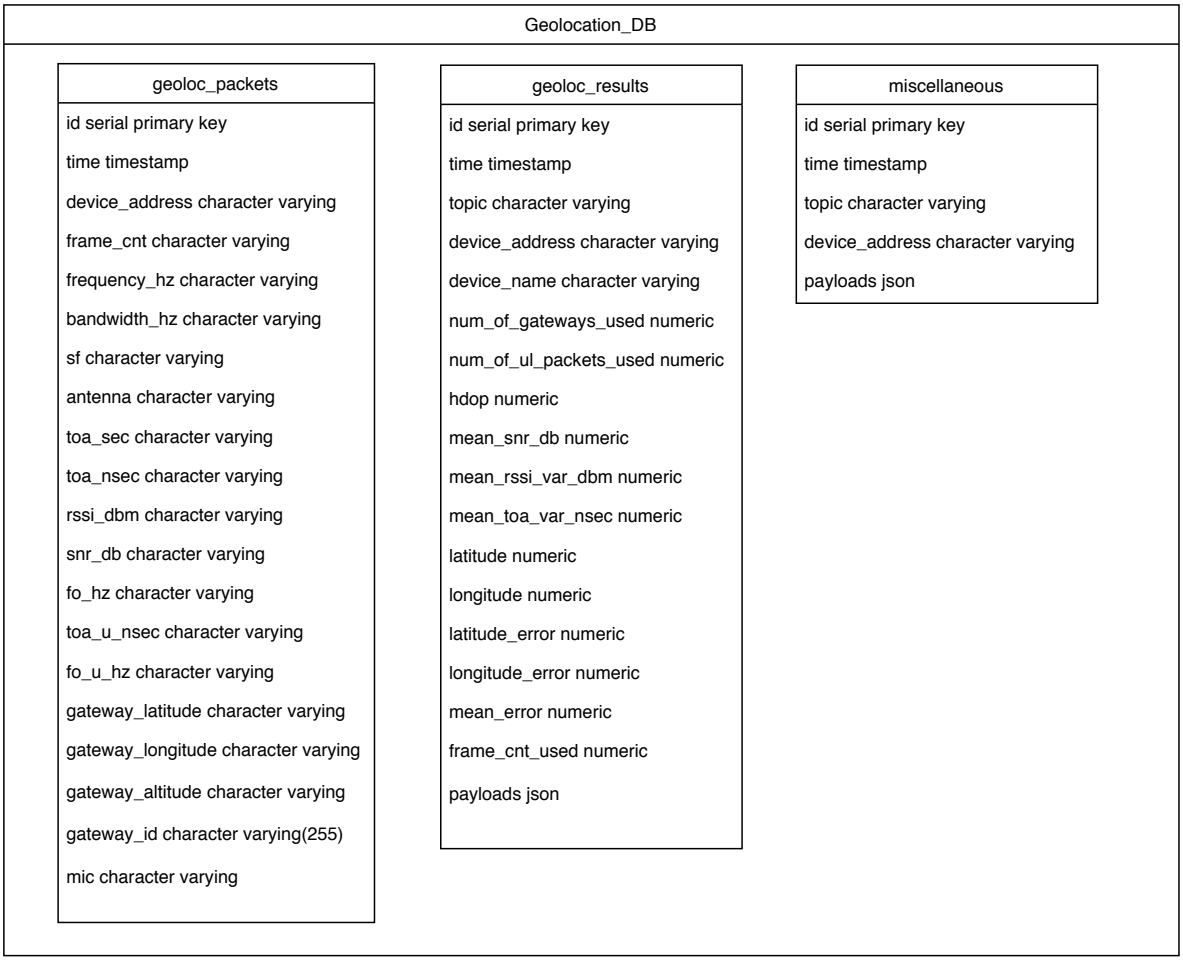


Figure 4.24 Tables and columns of geolocation database. The units & data types of the table columns are mentioned following the column name with underscore & spaces respectively

4.3.3 Resolver

The scientific python packages such as pandas, numpy, scipy, paho-mqtt, psycopg, pykalman, Tkinter (GUI) were used to build the Komro resolver and seaborn & matplotlib were used for performance evaluation [28]. For future deployments, docker platform has been chosen. There are two localization methods which could be used for geolocating LoRaWAN devices namely TDOA & RSSI localization techniques. The Resolver(TOA, GW_coordinates) takes those two parameters and an initial guess (i.e. nearest gateway with minimal TOA) about unknown position of end-device (x,y). The TDOA Multilateration equation given in 2.4 is used to geolocate the end-device using python library function scipy.optimize.least_square [28] which solves the linear algebraic equations being formulated. This function minimizes the residuals in the equation by iterative process starting from reasonable initial guess position till the iteration becomes stable/remains unchanged.

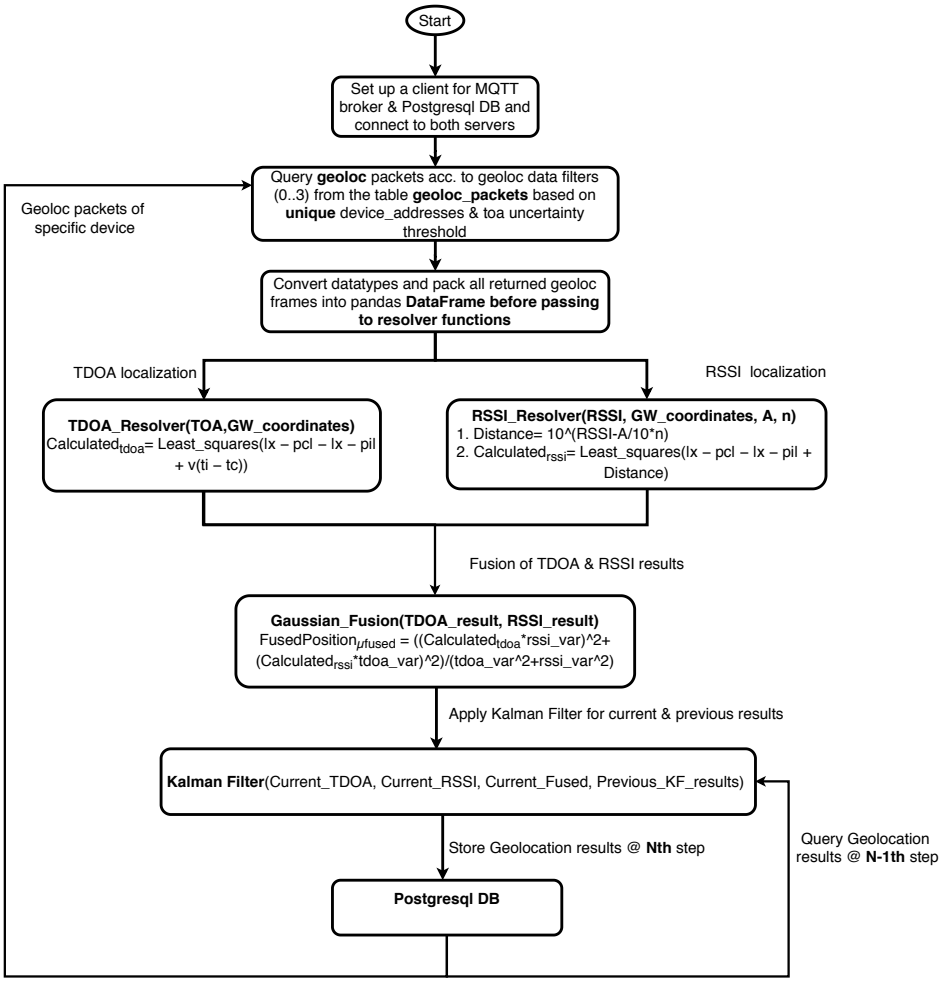


Figure 4.25 Block diagram of Komro Resolver Data (flow) pipeline

Algorithm 1: Algorithm for TDOA Geolocation Calculation

Result: Latitude and Longitude of the Device (x, y in UTM/ ECEF co-ordinates [meters])

Input parameters: GW Coordinates (x_i, y_i in meters), Time of Arrivals (TOA in nanoseconds);

for $i \leftarrow 1$ **to** $length(TOA)$ **do**

Linear_algebraic_equation=

$$\sqrt{(\vec{x_x} - \vec{p_{c,x}})^2 + (\vec{x_y} - \vec{p_{c,y}})^2} - \sqrt{(\vec{x_x} - \vec{p_{i,x}})^2 + (\vec{x_y} - \vec{p_{i,y}})^2} + v(t_i - t_c) = 0 \quad 2.4$$

Solver ⇒ scipy.optimize.least_squares(Linear_algebraic_equation)

end

Let the TOA data available be 'n' (geoloc packet rows), the algorithm runs for 'n-1' times, finding the difference in TOAs and multiplying with velocity of light which is approximately equal to propagation speed of LoRa waves.

Algorithm 2: Algorithm for RSSI Geolocation Calculation

Result: Latitude and Longitude of the Device (x, y in UTM/ ECEF co-ordinates [meters])

Input parameters: GW Coordinates (x_i, y_i in meters), Received Signal Strength Indicator (RSSI in dBm), Reference RSSI for 1 m (A in dBm), Path loss exponent (η);

for $i \leftarrow 0$ **to** $length(RSSI)$ **do**

$$d_i = 10^{\left\lfloor \frac{A - RSSI}{10.\eta} \right\rfloor} \quad 3.14$$

Linear_algebraic_equation= $(x - x_i)^2 + (y - y_i)^2 - d_i^2 = 0$ 3.10

Solver ⇒ scipy.optimize.minimize(Linear_algebraic_equation)

end

The scipy.optimize.minimize function [28] is similar to least_squares function of python scipy package. The distance equation used here is derived in 3.14 and the geolocation co-ordinates are obtained from 3.10. Let the RSSI data available be 'n' (geoloc packet rows), the algorithm runs for 'n' times, finding the absolute distances from each unique gateways found in the geoloc packets used for calculation to the device of interest. The latitude and longitude in degrees are converted into meters using two co-ordinate system converters (specific python libraries available) namely UTM [3.3] & ECEF [3.2].

While computing overall mean 'A' & 'η' values, an initial 'A' value should be initiated from the table 3.1 according to mean LoRa operating mode [3.1] of all devices, since it is tested & trusted experimental data. By sampling through the below algorithm, with an initial 'A' value as -26 dBm, the mean 'A' derived is -20.34 dBm and mean η derived is 2.89, where all the considered devices were taken into account.

Compute mean A & η

Algorithm 3: Algorithm for computing overall mean A & η

Result: mean A & η
Input parameters: Initial A value, Device coordinate (tx), device_dataset (Geolocation Dataset of each device contains gateway coordinates (GW), RSSI etc.)
global A, η
// initiate a value for A in order to calculate first η
$A \leftarrow [-26\ to\ -45]\ dBm\ (a\ constant\ value\ from\ table\ 3.1)$
while *True* **do**
  for $i \leftarrow 0$ **to** $length(device\_dataset)$ **do**
    import ith device_dataset
    for $i \leftarrow 0$ **to** $length(Unique\_Gateways)$ **do**
      Calculate distances (d) between tx & GW
      η ←(A-rssi)/(10.log10(d))
      A ←η.(10.log10(d))+rssi
      A_mean_GWx.append(statistics.mean(A)) `// for ith GW`
      η_mean_GWx.append(statistics.mean(η)) `// for ith GW`
    end
    A_mean_txx.append(statistics.mean(A_mean_GWx)) `// for ith device`
    η_mean_txx.append(statistics.mean(η_mean_GWx)) `// for ith device`
  end
  A_mean_Overall.append(statistics.mean(A_mean_txx))
  `// Overall mean` A `from all sampled device datasets`

  η_mean_Overall.append(statistics.mean(η_mean_txx))
  `// Overall mean` η `from all sampled device_datasets`
end

Fusion of TDOA & RSSI

A product of two Gaussian distributed model will result in new Gaussian distributed model with lesser uncertainty (narrowed distribution). Therefore, after TDOA & RSSI algorithms estimated the geolocation, the results are fused using mean (μ) & variance (σ^2) of the two estimates. The variance of these geolocation results can be obtained from the toa_uncertainty & rssi values of the geoloc frames being used for estimation. [27]

$$\mu_{fused} = \frac{\mu_1\sigma_2^2 + \mu_2\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \tag{4.40}$$

$$\sigma^2_{fused} = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \tag{4.41}$$

The equation 4.40 shows the fused mean of the two Gaussian mean values μ_1 & μ_2. The equation 4.41 shows the fused variance of the two Gaussian variance values σ_1^2 & σ_2^2 [27].

Listing 4.11 Algorithm for fusing TDOA & RSSI ('df' means geoloc dataframe)

```
# fusion function definition (similar to kalman filter update step)
def fusion(mean1, var1, mean2, var2):
    sum = var1+var2
    pr_s = mean1*var2 + mean2*var1
    new_mean =1/(sum) * pr_s
    product = var1*var2
    new_var = product/sum
    return [new_mean, new_var]
# noise modelled from toa uncertainty and rssi variance
# variance data extraction
toa_var = statistics.mean(df.toa_u_nsec)
# calculate rssi variance per each gateway_id, bcz b/w a gateway & device
   the rssi should be normally distributed
grouped_df_temp = df.groupby(df.gateway_id)
unique_gw_id = df.gateway_id.unique()
rssi_var_per_gw_list = []
for i in range(len(unique_gw_id)):
    grouped_df = grouped_df_temp.get_group(unique_gw_id[i])
    var_of_each_gw_id = grouped_df.rssi_dbm.astype(float).var()
    rssi_var_per_gw_list.append(var_of_each_gw_id)
rssi_var = statistics.mean(rssi_var_per_gw_list)
toa_var = toa_var/3                    # 3 nanoseconds = 1 meter inaccuracy
# TOA variance
toa_var = np.array([0.5*toa_var,0.5*toa_var]).astype(float)
# RSSI variance
rssi_var = abs(rssi_var/A)   # A in dBm = 1 meter inaccuracy
rssi_var = np.array([0.5*rssi_var,0.5*rssi_var]).astype(float)
# function call
fused = fusion(Position_rssi, rssi_var, Position_tdoa, toa_var)
```

GUI Resolver

Input Parameters

In order to geolocate a device using Komro GRS GUI Resolver which is built using python Tkinter package [28], one should pass the following parameters so that the resolver queries the user-defined geoloc (geolocation) data from PostgreSQL DB. The geoloc packets contain a lot of uncertainties and to ignore them, the SQL query has been used. But sometimes if the noisy measurements are ignored with a thresholding as shown here, there wont be enough geoloc frames to perform multilateration.

Listing 4.12 TOA uncertainty column name 'toa_u_nsec' is filtered with a threshold range from -500 to 500 nanoseconds using below function

```
import psycopg2
# filter out more uncertain values
```

```
def query_less_noisy_toa(device_address, low, high):
    # df is noise filtered dataframe
    sql_query = '''SELECT * from public.geoloc_packets
                   WHERE (device_address = %s AND CAST(toa_u_nsec AS
                      decimal(18, 2)) BETWEEN %s AND %s);'''
    conn = psycopg2.connect(user='postgres',
                            host=postgres_host,
                            port='5432',
                            password=postgres_password,
                            database=DB_NAME)
    cur = conn.cursor()
    cur.execute(sql_query, (device_address,low, high,))
    SQL_Query = cur.fetchall()
    # execute the sql query with pandas dataframe
    df = pd.DataFrame(SQL_Query, columns=['id'...'mic'])
    toa_column = df['toa_sec'].astype(str).str.cat(df['toa_nsec'].astype(
        str), sep='.').copy()
    df['toa'] = toa_column.copy()
    # set datetime index
    df['time'] = pd.to_datetime(df['time'])
    df = df.set_index('time').copy()
    # sort geoloc frames based on time
    df = df.sort_index().copy()
# filter the geoloc dataframe & pass the df to komro resolver function as
    shown in [Listing 4.11]
device_address = 'abcd123'
low_limit = -500 [in nanoseconds]
high_limit = 500 [in nanoseconds]
# call the functions
df = query_less_noisy_toa(device_address, low_limit, high_limit)
```

- **Device Address**- Device address assigned by network server (Ex.0167xxxx)
- **Num of UL Frames**- Number of geoloc frames to be passed for geolocation estimation (Ex. 4...10)
- **Filter Multi-frame?**- If set to 1, multiple UL frame groups are passed to resolver function, otherwise passes only the last-best UL frame group. (Boolean)
- **Calculate TOA?**- If set to 1, the TOA will be calculated using ground truth of device. Just for simulating the ideal TOA. (Boolean)
- **Plot Hyperbola?**- if set to 1, the hyperbola will be generated acc. to multilaterate algorithm (Boolean)
- **Filter Mode**- Essential parameter, it is used to choose one of the following data filter modes

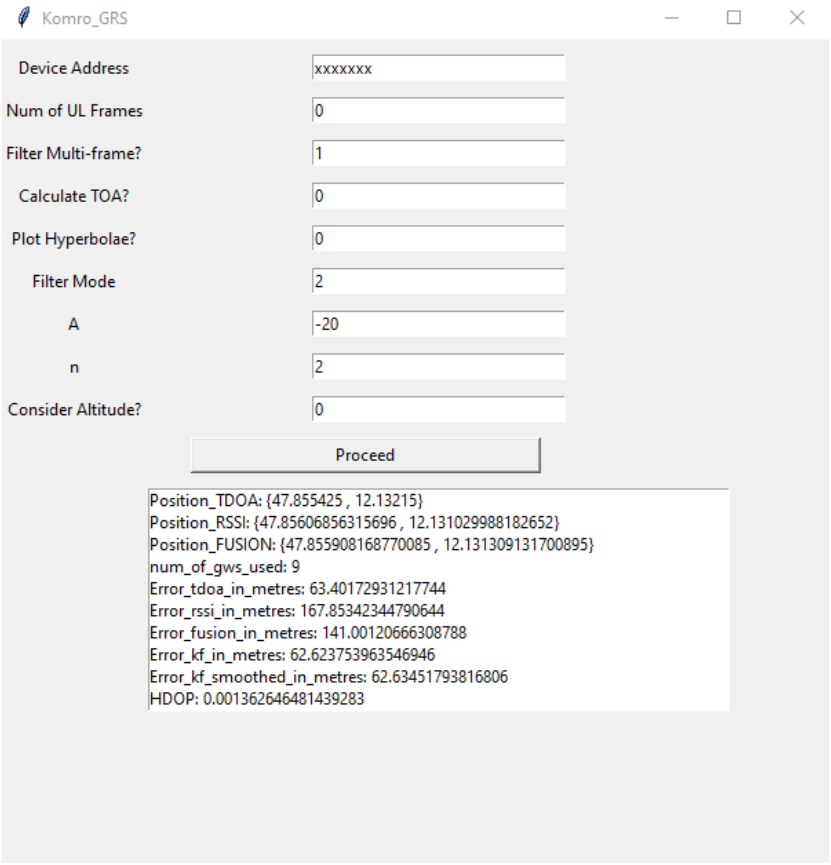


Figure 4.26 Komro GRS GUI Resolver User Input

- **A**- Absolute RSSI value for 1 meter (in dBm)
- **n**- RSSI Path loss exponent or environment coefficient (no unit)
- **Consider Altitude?**- If set to 1, the gateway altitudes (in ECEF coordinates) will be used for geolocation calculation, otherwise uses only latitude & longitude (in UTM coordinates). (Boolean)

Geoloc Data Filter

It is important to know how the geoloc data are filtered from SQL query return before passing to resolver function. This is the essential parameter of the resolver client since each filter extracts geoloc data on different basis, leading to different geolocation estimation for every time/ UL instance although the device was stationary during the whole time, this will be evaluated and improved on evaluation phase. The most important & reliable Filter Mode is 2, it extracts the geoloc frames depending on the FCntUp (i.e. uplink frame count) in descending order, such that an UL group has more than 2 unique Gateway_EUIs **AND** TOA noises between -500 ns to 500 ns. This filter mode works in two ways, either it passes only single UL group (**singleframe**) or multiple UL groups (**multiframe**) depending on additional essential control flag called **'Filter Multi-frame?'** (see annexure A). In addition, it gives us an idea which geoloc data filter or which GUI

parameter gives better accuracy closest to the true location (for stationary devices). In the figures [4.28] [4.30] [4.31] shown the geolocation of same device at different data filter modes. **Note:** No matter which filter mode has been used, positions will be resolved for each unique UL frame count (i.e. FCntUp) present in the filtered geoloc dataframe. There are four data filter modes namely:

- **Mode 0**- filters based on number of frames
- **Mode 1**- filters based on time delta (ex. last hour)
- **Mode 2**- filters based on UL frame counts (singleframe/multiframe- mostly used)
- **Mode 3**- filters based on constant date or time (geoloc data after certain date/time)

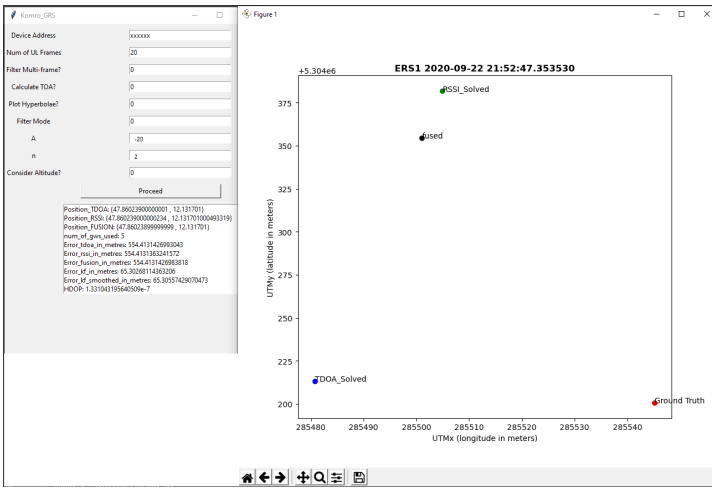


Figure 4.27 Komro GRS GUI Resolver takes Input parameters and displays the geolocation of corresponding device for filter mode 0

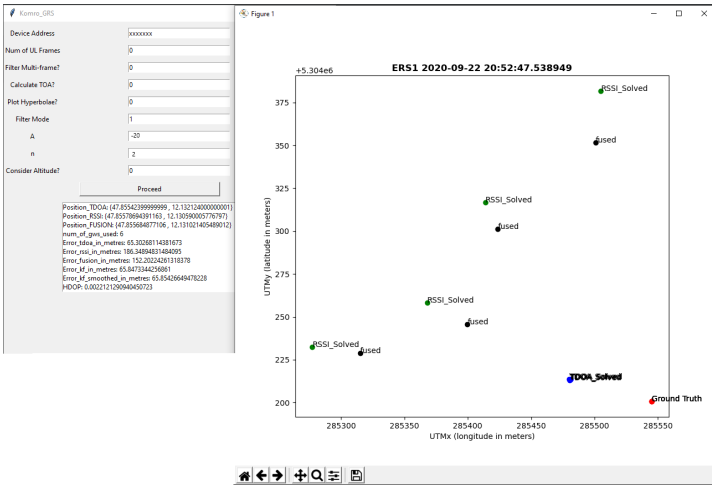


Figure 4.28 Komro GRS GUI Resolver takes Input parameters and displays the geolocation of corresponding device for filter mode 1

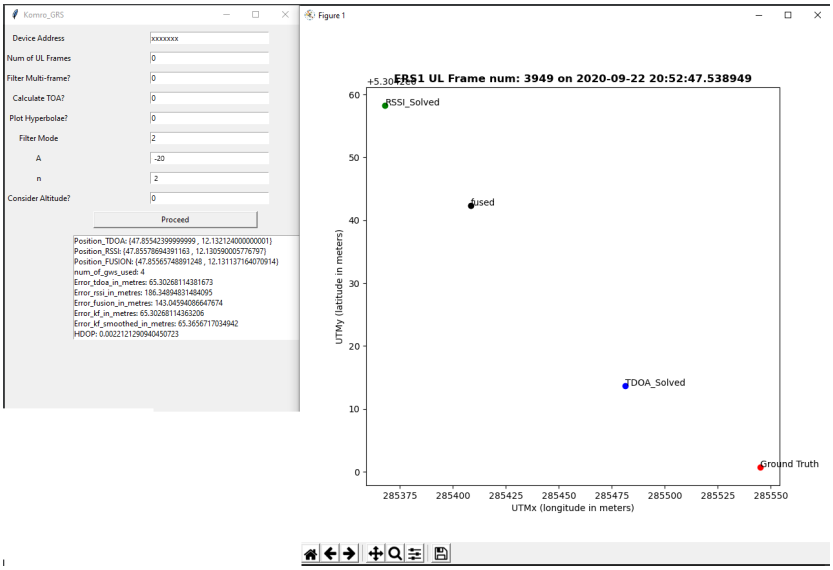


Figure 4.29 Komro GRS GUI Resolver takes Input parameters and displays the geolocation of corresponding device for filter mode 2 in mobile mode (using last singleframe)

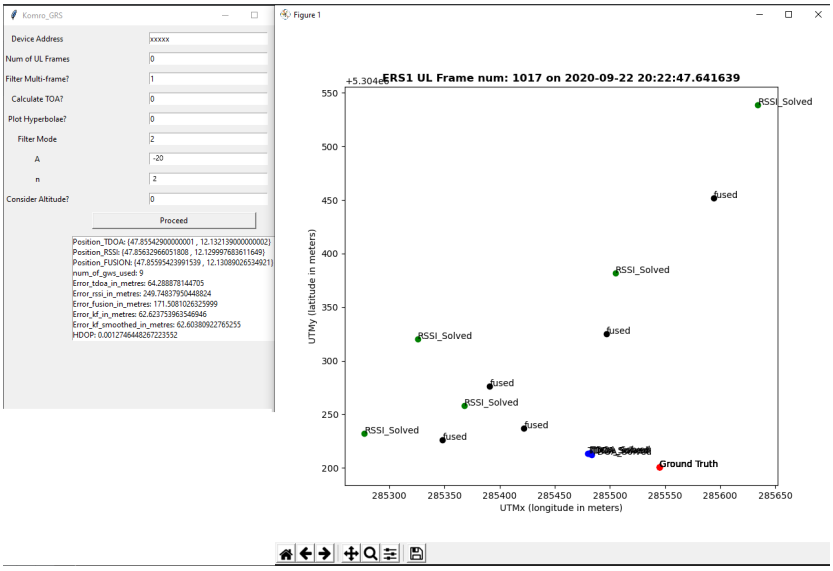


Figure 4.30 Komro GRS GUI Resolver takes Input parameters and displays the geolocation of corresponding device for filter mode 2 in stationary mode (grouping multiframes& resolve individually in a sequence)

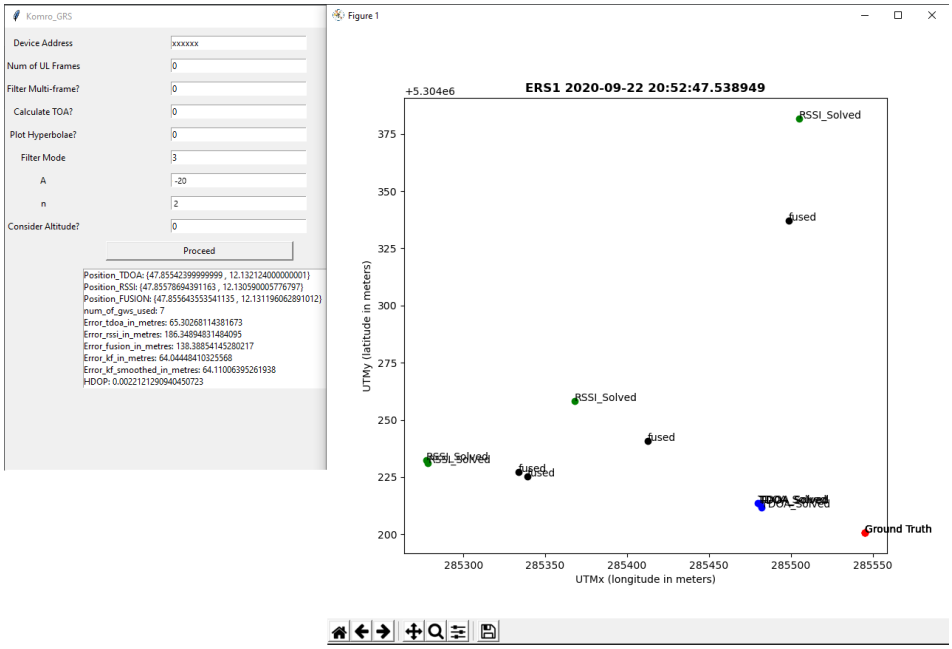


Figure 4.31 Komro GRS GUI Resolver takes Input parameters and displays the geolocation of corresponding device for filter mode 3

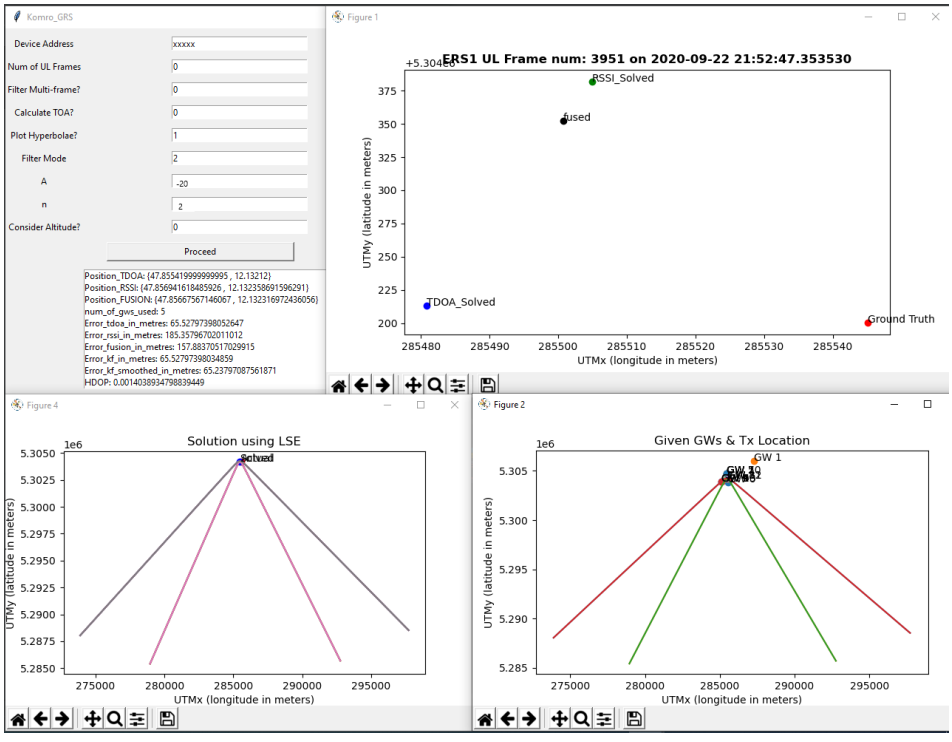


Figure 4.32 Given the true location of a device, the hyperbolae can be plotted by calculating the TOA of signal from a device to gateways i.e. where the gateways are widely spaced around the device

Headless Resolver

So far the GUI resolver has been shown. In order to run the resolver in Host Server periodically (say for every one hour), a headless resolver has been created. This resolver works same as GUI resolver but without GUI and unlike GUI resolver it resolves for multiple devices (ex. pre-selected device address list) periodically (hourly) in every execution cycle which would be comfortable for most IoT use cases (monitoring). Since there is no relation between number of gateways & accuracy, the condition is constructed such that, only if the geoloc data of the end-device contains more than 2 gateways, it could be resolved by komro resolver otherwise the loop just passes to next device_address. **Note:** It is important to know even with less than 2 gateways some devices randomly shows better accuracy due to LOS & antenna diversity (0 or 1). The extract_geoloc_frames() function queries the appropriate geoloc data from DB and gets the dataframe (df) which will be passed to komro_resolve() function, that further returns various geolocation results and are published to geoloc broker in the following topics, which will be stored in PostgreSQL DB under geoloc_results table:

Listing 4.13 Komro Resolver Structure & MQTT topics on which the resulting geolocation estimates are being published [28]

```
# import libraries
import paho.mqtt.publish as publish
import pandas as pd
import numpy as np
# read the true_location.csv file (ground truth)
df_trueloc = pd.read_csv('true_location.csv', sep=';', encoding='latin-1',
    header= 'infer', engine='python')
while True:
    # resolve for selected device group only
    device_addr_list = ['1exxx4c', '1d3xx70', '14xxdc', '8xxxe', '11xx5b9',
        '16xxxfe', 'f5xx4', 'f2xxxd', '1axxx6', '2cxx94','11xx073', '5
       exxd4', '19xxx1', '17xx86', '1dxxx8', 'b7xxx53', '166xxx7a', '16
       xxx6', '1xxxcf', '1exxxa']
    for dev_addr in device_addr_list:
        # query geoloc packets acc. to filter modes
        df = extract_geoloc_frames(dev_addr, filter_mode, num_of_frames,
            split_date, bmultiframe, set_fcnt, low_limit, high_limit)
        # group dataframe acc. to UL frame counts
        grouped_df_temp = df.groupby(df.frame_cnt)
        # get the unique UL frame counts present in dataframe
        unique_fcnt = df.frame_cnt.unique()
        # last 'n' ULs to resolve (in descending order)
        num_of_UL_frames_to_resolve = 5
        if len(unique_fcnt)<num_of_UL_frames_to_resolve:
            num_of_UL_frames_to_resolve = len(unique_fcnt)
        # resolve & publish for each UL seperately in descending order for
            the device, from -1 to -n (num_of_UL_frames_to_resolve)
```

```
        for i in range(num_of_UL_frames_to_resolve):
            print('UL_DF_Index:', -(i))
            # get the grouped dataframe & resolve for individual UL
            single_UL_df = grouped_df_temp.get_group(unique_fcnt[-i])
            print('Mean_Time_of_UL_DF:', single_UL_df.index.mean())
            # resolve only if the end-device reached more than 2 gateways
            total_unique_gw_id = len(df.gateway_id.unique())
            if total_unique_gw_id >2:
                result = komro_resolve(single_UL_df, A, n, df_trueloc,
                    dev_addr, bcalculatedTOA)
                publish("geolocation/calculation/tdoa/komro/"+str(dev_addr)
                    ,str(tdoa_result),qos,retain)
                publish("geolocation/calculation/rssi/komro/"+str(dev_addr)
                    ,str(rssi_result),qos,retain)
                publish("geolocation/calculation/fused/komro/"+str(dev_addr
                    ),str(fused_result),qos,retain)
            else:
                print('Due to less than three gateways, the LOOP passed for
                     '+str(dev_addr))
                pass
    sleep(3600)
```

The third party applications or IoT platforms like Thingsboard can subscribe to this topics according to the device address and display the estimated geolocation.

4.3.4 Kalman Filter

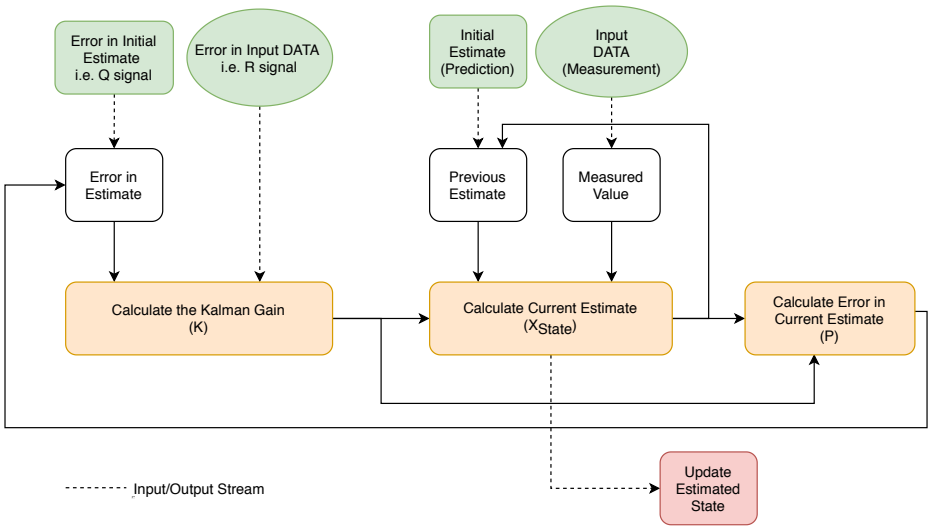


Figure 4.33 Block diagram of KF Algorithm (working principle)

A Kalman Filter is a time discrete, recursive & optimal (with minimal error variance or 'least squares') estimator for the state of a dynamic system, which has control inputs and

is subject to stochastic noise. KF combines/ enhances raw measurements (observations) with knowledge of a system's dynamics (process model) and knowledge of the statistical properties of error sources. Although geolocation in LoRaWAN Infrastructure is not meant for mobile purposes(i.e. continuous tracking), KF could be implemented to filter out the measurements noises (R) associated with the observations such as TDOA, RSSI & Fused. KF application is quite similar to time averaging of previous (KF) results along with current geoloc results (dead-reckoning). In Pykalman, a python implementation of Kalman filter [13], there is a method called **em()**, also called expectation maximization method/algorithm which will assume the measurements are normally distributed & finds the local minimum of the distribution in iterative manner. The Expectation Maximization (EM) is popularly used statistical approach to fit mixture-of-Gaussian models and draw confidence ellipsoids for multivariate models. The model parameters which are traditionally specified by hand can also be learned by the implemented EM algorithm without any labeled training data (auto tuning of parameters such as transition covariance & observation covariance as shown in the sample code Listing 4.12) [13]. The most significant part on applying kalman filter is to extract variance of RSSI & TOA parameters from geoloc frames being used to solve the position, these measurement noises (R) are stored along with each results not only for future evaluation, but also for passing to KF (over&over). Here measurements provided are tdoa, rssi, fused, previous kf estimates and measurement noises (R) provided are toa uncertainty and rssi variance which are specified by hand, rest of the internal elements and transition elements are extrapolated by EM algorithm by default. In short, if the passed measurements are consistent & more likely to each other, the obtained output from KF will be more certain or optimized corresponding to the ground truth due to optimized kalman gain (K).

Listing 4.14 Python library for Kalman Filter [Pykalman [13]]

```
# import pykalman library
from pykalman import KalmanFilter
# define the KF function (necessary subsets of pykalman library)
def kf_estimate(kf, measurements):
    kf.initial_state_mean = measurements[0]
    # call expectation maximization (em) subclass of KF object/class which
       iteratively auto-tunes the KF parameters
    kf = kf.em(measurements, n_iter=5)
    (filtered_state_means, filtered_state_covariances) = kf.filter(
       measurements)
    (smoothed_state_means, smoothed_state_covariances) = kf.smooth(
       measurements)
    return filtered_state_means, smoothed_state_means
# concatenate current geolocation results of TDOA, RSSI & Fused & previous
   KF estimates as measurement variables
current_measurements = [TDOA_result, RSSI_result, Fused_result]
previous_KF_results = [kf_result0, kf_result1, kf_result2]
# append the current_measurements to measurements, by default
for i in range(len(current_meas)):
    measurements.append([current_meas[i][0], current_meas[i][1]])
```

```
# append the previous KF results to measurements, if enabled (for devices where RSSI & TDOA results are more close to each other)
if bCallPrevKFResults:
    for i in range(len(previous_KF_results)):
        measurements.append([previous_KF_results[i][0], previous_KF_results[i][1]])
# Initialize KF object
# covariance matrix [P]
initial_covariance = 1 * np.array([[1, 0], [0, 1]])
kf = KalmanFilter(n_dim_state=2,
                  n_dim_obs=2,
                  transition_matrices=[[1, 0], [0, 1]],
                  observation_matrices=[[1, 0], [0, 1]],
                  initial_state_covariance = initial_covariance,
             em_vars=['transition_covariance', 'observation_covariance'])
# tuning params (transition_covariance [Q], observation_covariance [R])
# concatenate variance of kf measurements [R]
mean_rssi_var_m = abs(mean_rssi_var_dbm / A)  # convert dBm to meters
mean_toa_var_m = abs(mean_toa_var_nsec / 3)  # convert nsec to meters
mean_R = np.mean([mean_rssi_var_m, mean_toa_var_m])
kf.observation_covariance = mean_R * np.array([[1, 0], [0, 1]])
# call the kf_estimate function
kf_result, smoothed_kf_result = kf_estimate(kf, measurements)
# publish the results to geoloc broker
publish("geolocation/estimation/kf/komro/" + str(dev_addr),str(kf_result),qos,retain)
publish("geolocation/estimation/smoothed_kf/komro/" + str(dev_addr),str(smoothed_kf_result),qos,retain)
```

4.3.5 GDOP Calculation

The Geometric Dilution Of Precision (GDOP) is a metric to determine the quality of gateway deployment or its relative geometry with end-device [2].
Let us consider,

$$a_n, x = x$$
$$a_n, y = y$$
$$a_n, z = z$$

Where x,y,z be the co-ordinates of gateways & end-device (ground truth or resolved position), either in ECEF (x,y,z) or UTM (x,y only) system, 'n' represents no. of gateways + 1 (end-device) and 'a' denotes a vector component in Line Of Sight (LOS) Matrix 'A' [2]. LOS Matrix 'A' has ones components in last column, because the time domain is neglected

here.

$$A = \begin{bmatrix} a_{1,x} & a_{1,y} & a_{1,z} & 1 \\ a_{2,x} & a_{2,y} & a_{2,z} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,x} & a_{n,y} & a_{n,z} & 1 \end{bmatrix}$$

$$A^T = \begin{bmatrix} a_{1,x} & a_{2,x} & \dots & a_{n,x} \\ a_{1,y} & a_{2,y} & \dots & a_{n,y} \\ a_{1,z} & a_{2,z} & \dots & a_{n,z} \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

Now compute covariance matrix [2]:

$$COV(X) = (A^T.A)^{-1} = \begin{bmatrix} (\sigma_x)^2 & \sigma_x.\sigma_y & \sigma_x.\sigma_z & \sigma_x.\sigma_t \\ \sigma_x.\sigma_y & (\sigma_y)^2 & \sigma_y.\sigma_z & \sigma_y.\sigma_t \\ \sigma_x.\sigma_z & \sigma_y.\sigma_z & (\sigma_z)^2 & \sigma_z.\sigma_t \\ \sigma_x.\sigma_t & \sigma_y.\sigma_t & \sigma_z.\sigma_t & (\sigma_t)^2 \end{bmatrix}$$

Finally, extract DOPs from the covariance matrix [2]:

$$GDOP = \sqrt{(\sigma_x)^2 + (\sigma_y)^2 + (\sigma_z)^2 + (\sigma_t)^2}$$
$$TDOP = \sqrt{(\sigma_t)^2}$$
$$PDOP = \sqrt{(\sigma_x)^2 + (\sigma_y)^2 + (\sigma_z)^2}$$
$$HDOP = \sqrt{(\sigma_x)^2 + (\sigma_y)^2}$$
$$VDOP = \sqrt{(\sigma_z)^2}$$

5 Evaluation

The performance and accuracy of device positioning in the above implementations will be analysed and the results will be depicted here.

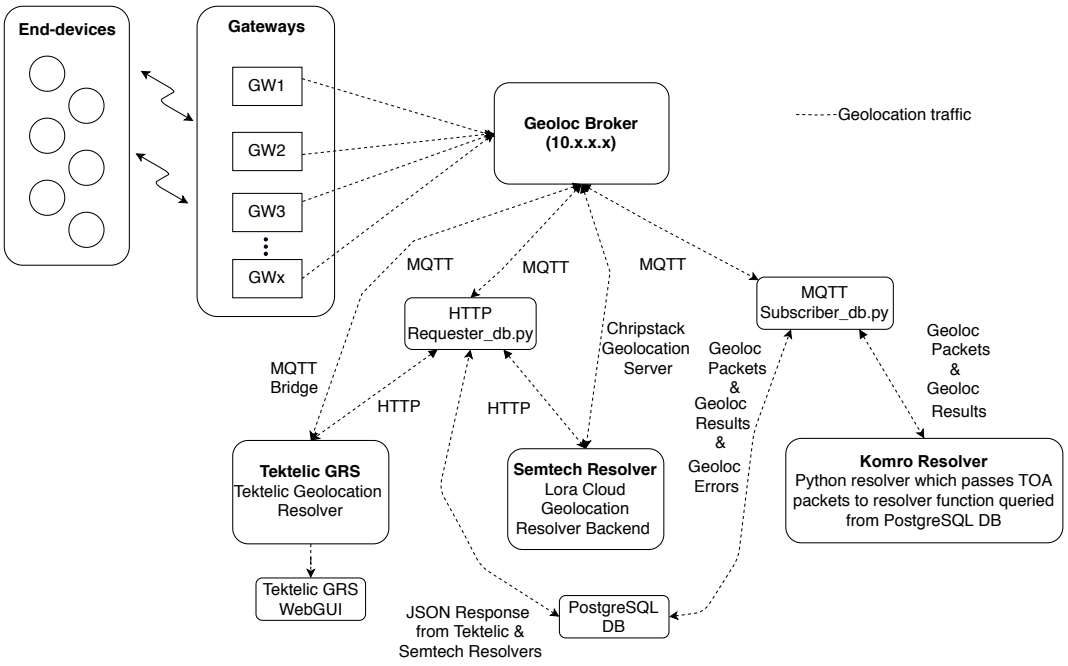


Figure 5.1 Block diagram of Overall Implementation [4.1] [4.21]

5.1 Comparison of Various Implementations

Comparison of various implementations will be shown in [4.1] [4.21] [5.1]. This includes performance analysis, comparison of error statistics etc. For evaluation, 21 end-devices (LoRa sensors) were selected, the ground truths were shown in tables & maps. In maps, the red color indicates gateways & blue indicates end-devices. The statistics are applied on the geolocation result dataset collected from last 10 days. Since the evaluation is done around Rosenheim district, **geo-fencing** has been implemented, to eliminate unwanted or illogical results (ex. hidden noise variations mislead the estimated position towards other countries like small change in latitude alone pinpoints estimation at Africa), which might poorly influence the variance or standard deviation of error distribution. The word 'overall' here denotes consolidating all devices. The best/minimal errors achieved are evaluated from the entire error dataset collected since the beginning of this thesis to show the maximum achievable accuracy and corresponding device location.

Table 5.1 Ground Truth of Komro LoRa devices to be evaluated. The devices namely LuG_FWI21, LuG_FW111 & ERSN11 were placed in basements (@ lower altitudes)

True Location of Komro Devices			
Device Name	Latitude	Longitude	Description
ERS1	47.855329	12.132985	komro
ERSN2	47.82637	12.094212	Trafostation Pang
ERSN7	47.839692	12.09008	Trafostation Brucklach
ERSN11	47.855567	12.126387	Trafostation P&C
ERSN16	47.842413	12.133832	Trafostation Am Kobel/Kaltwies
LuG_FW4	47.849838	12.094099	Grubholzer Str. 2 - DM
LuG_FW111	47.855995	12.127162	Max-Josefs-Platz/Bergmeister
LuG_FW20	47.852551	12.105238	Äußere Münchner Str.
LuG_FWI21	47.79429	12.12221	Raubling
DZG_Zaehler001	47.855329	12.132985	komro
SensoneoWastesensor006	47.857962	12.132537	MBB
SensoneoWastesensor037	47.829013	12.115934	Grünfeldstraße
SensoneoWastesensor022	47.88023	12.125959	Laurentius/Langenpfunzen
SensoneoWastesensor029	47.860527	12.122631	Meraner Straße Fürstatt
Stockwaage3	47.878361	12.371056	Sassau
Stockwaage4	47.8534	12.16	Schlossberg/Stephanskirchen
Stockwaage8	47.85449	12.287833	Ratzinger Höhe
Stockwaage10	47.867681	11.941739	Bruckmühl, 83052
Stockwaage18	47.8434	12.16000	Stephanskirchen
TIS1	47.83973	12.11313	Mangfall/Miesbacher Str.
TIS15	47.830801	12.114622	Asamstr. 72

Table 5.2 Ground Truth of Komro Gateways

True Location of Komro Gateways				
gateway id	Latitude	Longitude	Altitude	Description
5B99	48.033791	12.206955	484	Freiham, Eiselfing
51F8	47.866111	12.10856	463	TH Rosenheim
8103	47.854657	12.131994	406	Innstraße 30, Rosenheim
6A81	47.85189	12.132987	426	Minigolf im Mangfall
6A78	47.883146	12.286356	492	Bienenzuchtverein Bad Endorf
5B7B	47.795936	12.118171	478	Redenfelden, Raubling
51F7	47.883115	12.286383	490	Bienenzuchtverein Bad Endorf
5202	47.851307	12.091761	468	Oberaustraße 12, Rosenheim
5B7A	47.872332	12.155551	524	Stephanskirchen
5BB4	47.852391	12.114261	465	Eduard-Rüber-Straße 5, Rosenheim
5247	47.852605	12.127323	462	Parkhaus P1 Zentrum, Rosenheim
524A	47.860205	12.131756	493	Pettenkoferstraße 10, Rosenheim
5204	47.747946	12.250003	1555	Frasdorf, 83112
5248	47.855416	12.132148	466	Am Innreit 2, Rosenheim

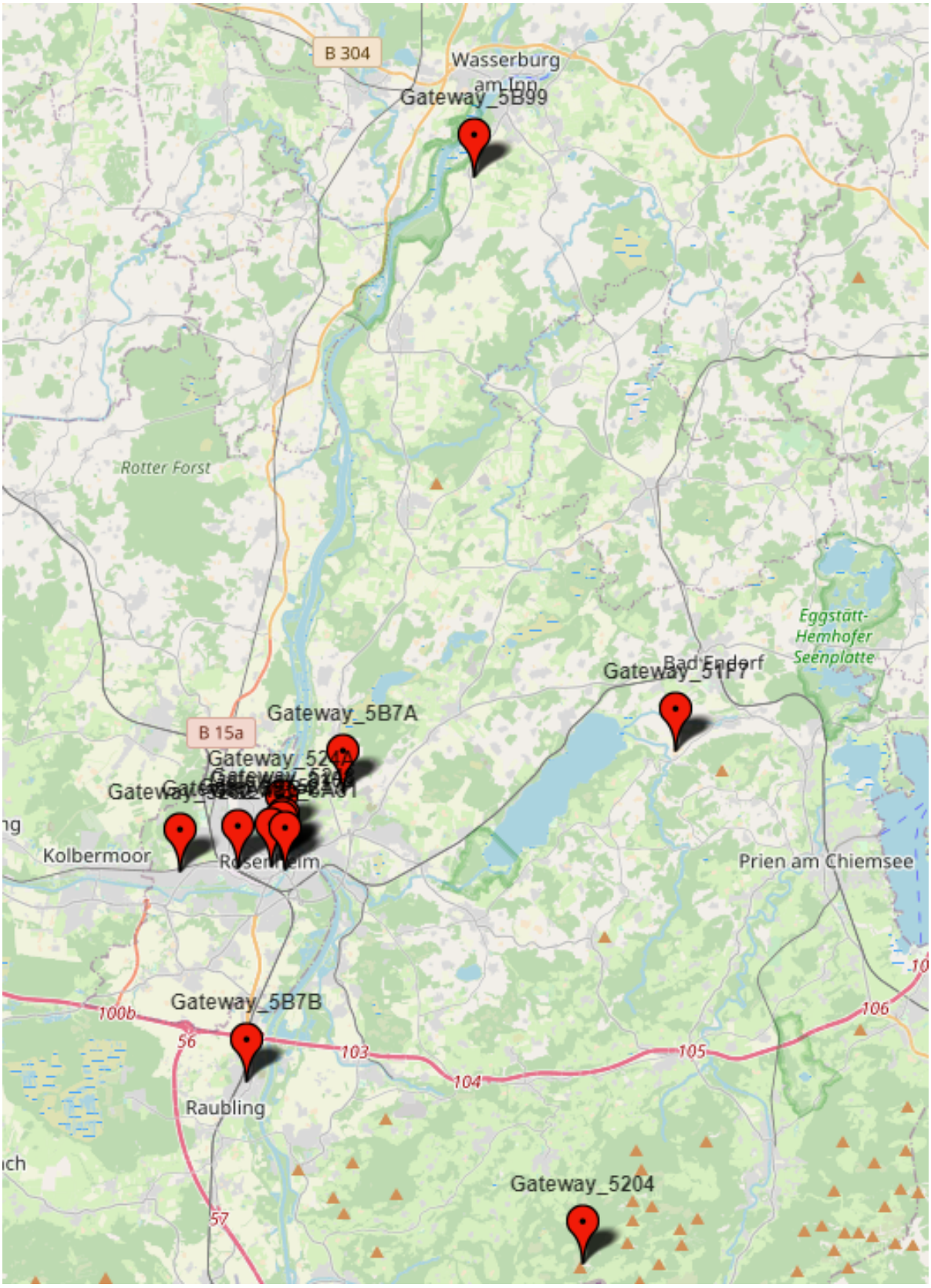


Figure 5.2 Overall Geolocation Gateways

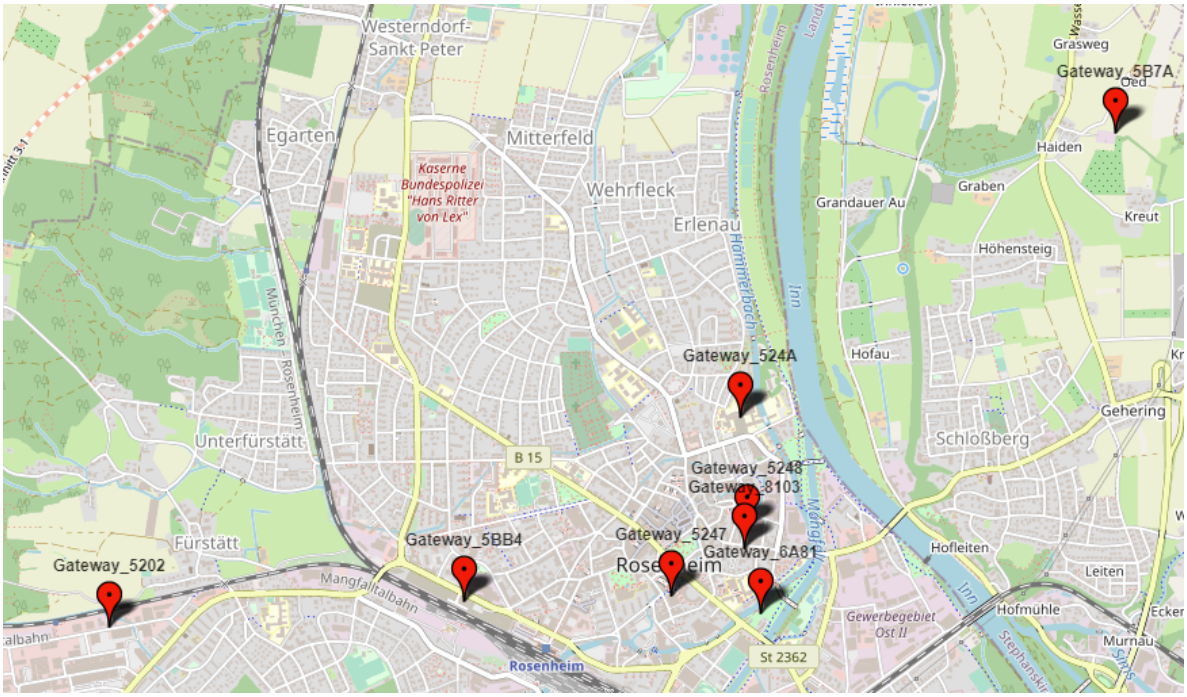


Figure 5.3 Urban Geolocation Gateways

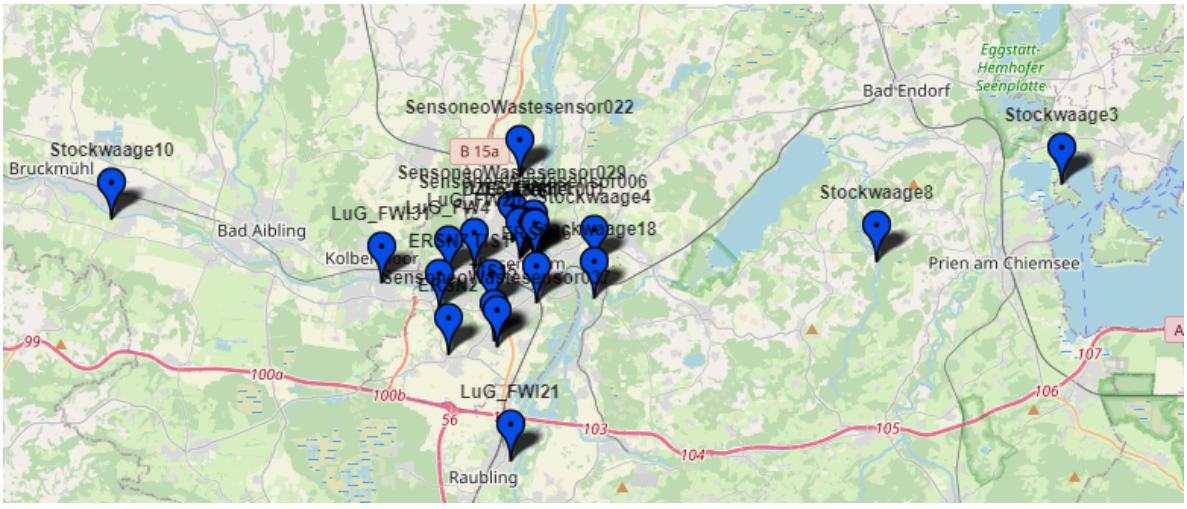


Figure 5.4 Overall Geolocation Devices

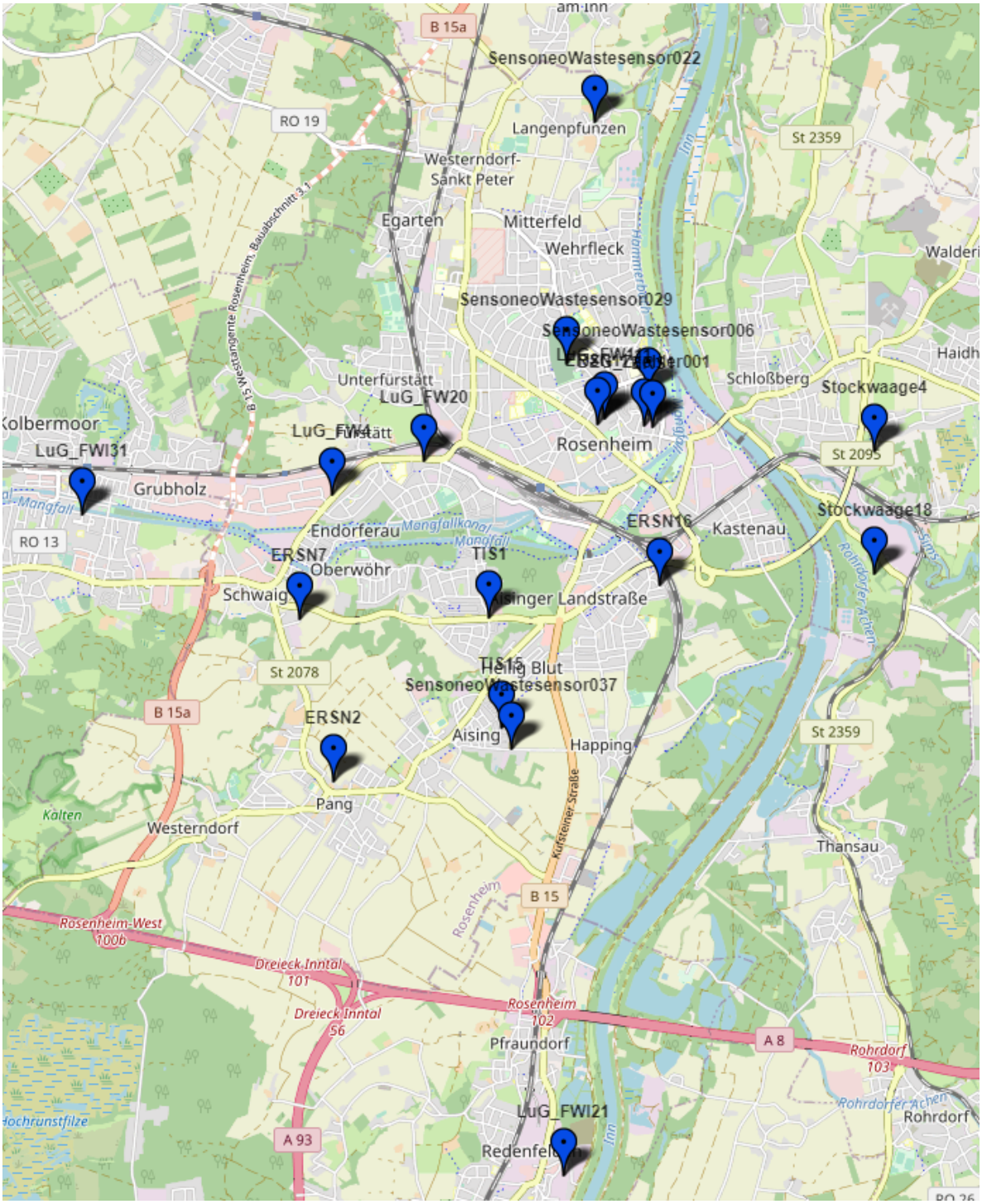


Figure 5.5 Urban Geolocation Devices

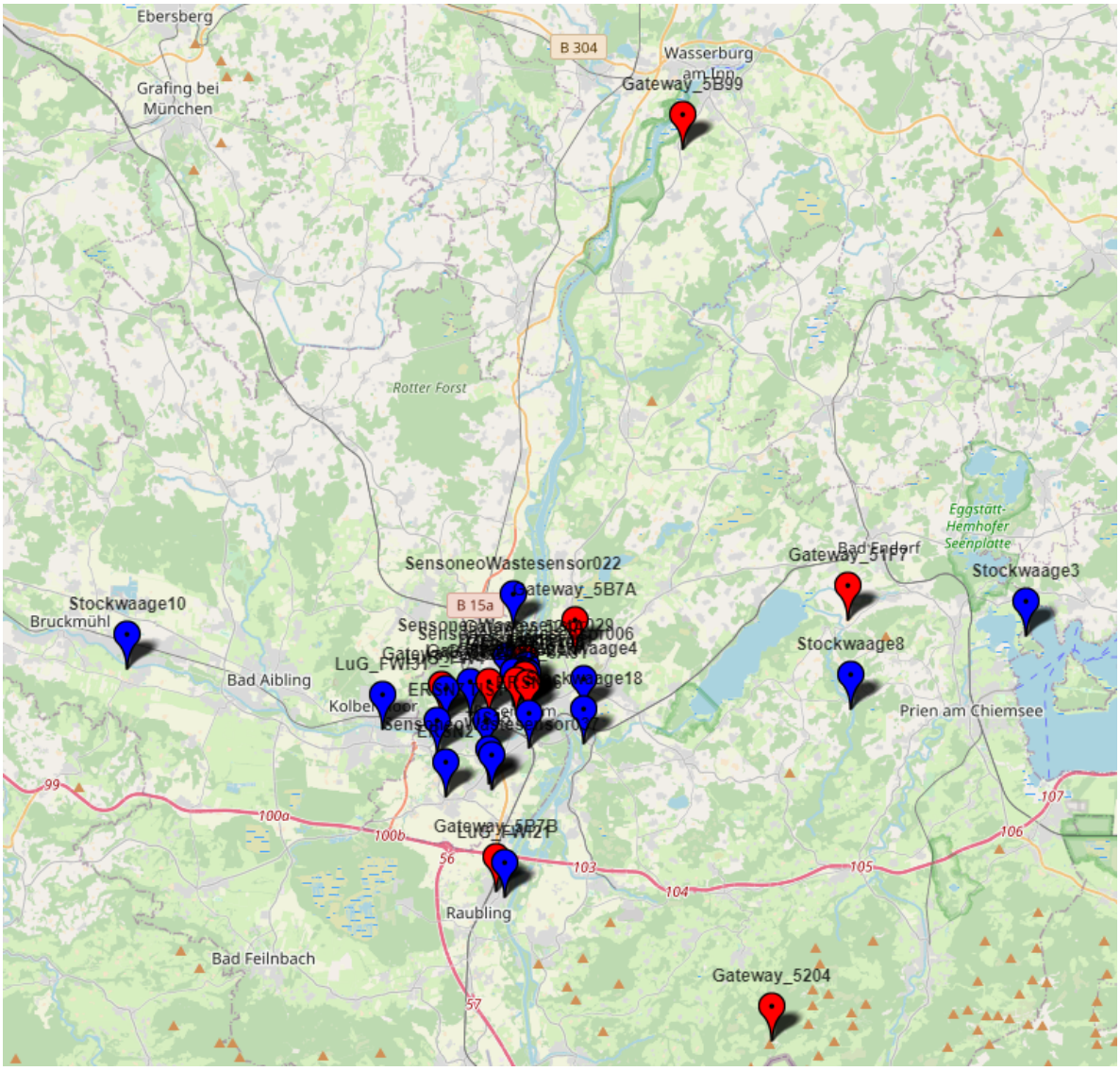


Figure 5.6 Overall Gateways vs Devices

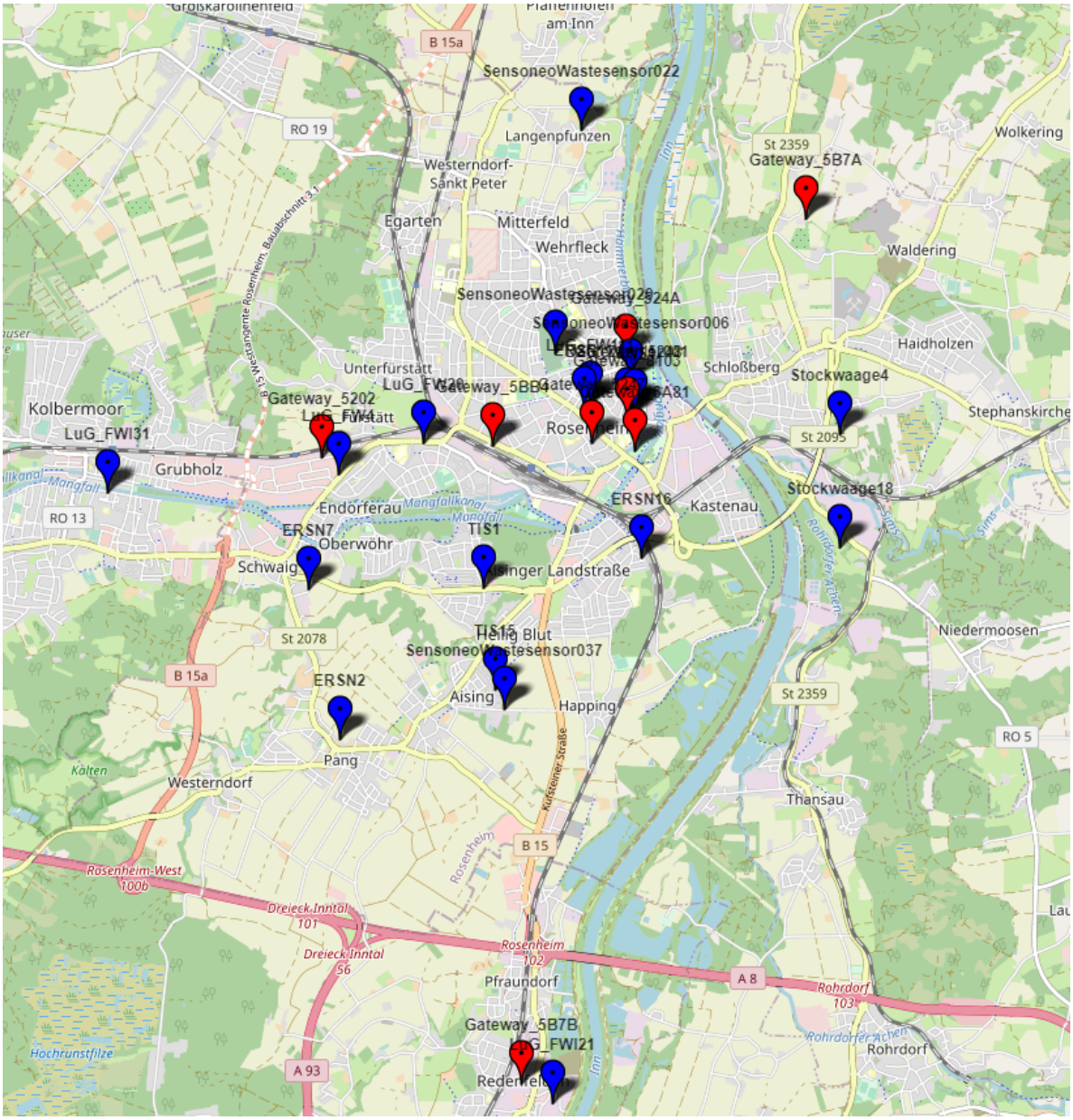


Figure 5.7 Urban Gateways vs Devices

5.1.1 GRS: Tektelic vs Semtech vs Komro

In this subsection, the implementation comparison of different Geo-resolvers (GRS) along with their geolocation accuracy were shown.

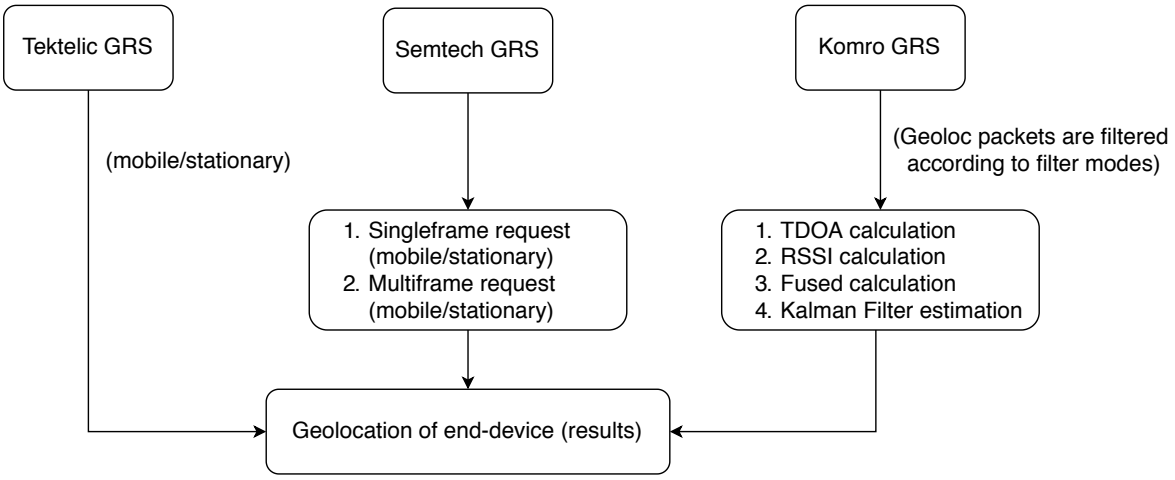


Figure 5.8 Block diagram of various geolocation resolvers and their data pipeline through which the geolocation result is calculated

The figure 5.8 shows the different GRS implemented in this thesis work and their different localization methods. The public geo-resolvers like Tektelic GRS & Semtech GRS have not revealed how their algorithm work, but just published the API formats in their homepages. Tektelic GRS accepts the geolocation estimation request in either in MQTT or HTTP protocol given the appropriate geoloc packets of the specific device. As a result Tektelic GRS can give only one geolocation result in both protocols. It is important to note, that entire available geoloc dataframe of a device has been sent to Tektelic GRS over HTTP. Semtech GRS takes only two HTTP request formats such as singleframe & multiframe formats. The singleframe format is nothing but the frame formatted with single UL message group, whereas in multiframe format multiple UL message groups are used to form semtech specified HTTP request body (API). As a result semtech GRS can give 2 seperate geolocation results. The private geo-resolver Komro GRS filters the geoloc packets according to 4 filter modes and then pass the filtered packets to 4 localization methods/ functions namely TDOA, RSSI, Fused & Kalman Filter which gives 4 seperate geolocation results. The filter_mode 2 with singleframe is the most reliable filter in Komro GRS. However Tektelic & Semtech GRS estimates geolocation based on a major parameter called 'mobility status' i.e. If the device is 'stationary' the estimation returns only one result with grouping or averaging or else if the device is 'mobile' the estimation returns multiple different result without grouping or averaging. If the device is 'mobile' the GRS will estimate geolocation for each UL group separately (ex. for ULs in last 24 hours) and for 'stationary device' the GRS will calculate geolocation for several UL groups seperately and estimate the final geolocation.

The table 5.3 shows the common comparison between Tektelic, Semtech and Komro Resolver Implementation.

The following tables and figures show the error statistics of Tektelic, Semtech and Komro resolvers. The units of each parameters in a table are mentioned followed by '_' in each column names or with a square brackets ([]).

The table 5.7 shows, overall best or minimum RMS error achieved using three different resolvers.

Table 5.3 Comparison of Implementation

General Comparison of GRS			
Factors	Tektelic GRS	Semtech GRS	Komro GRS
Average Accuracy	High	Low	Moderate
Protocol	HTTP & MQTT	HTTP	MQTT
Public availability	yes	yes	no
Need a cloud account?	yes	yes	no
Server Location	USA	France	Germany
Implementation Difficulty	High (incl. BSP installations)	High	Very High
Storage History in MQTT	Last 1000 messages per device	No MQTT	All the geoloc ULs

To evaluate the geolocation of LoRa devices, the mean distance between calculated position & ground truth are computed. The error is calculated for each devices under various resolvers namely: Tektelic, Semtech, Komro. Especially, the Komro resolver have 4 localization methods such as TDOA, RSSI, Fusion/Fused, Kalman Filter (KF). The evaluation statistics are applied on errors of all devices under various resolving methods and tabulated below. The error distributions are shown with the help of distribution plot from python seaborn library (sns.distplot()) where the density of the error distribution is shown. In order to focus on the trends & proportions, the rarely occurring errors beyond a threshold (ex. 5000 m) are ignored while plotting. For some categorical datapoints, the quantities are randomly distributed, therefore they are plotted using boxplots from python seaborn library (sns.boxplot()) where the median errors according to different data quantities will be shown in each boxes with short horizontal line at the middle.

Error calculation

The formula for error & root mean squared error (RMSE) were shown below,

$$Error = \sqrt{(calculated\_position - actual\_position)^2} \quad (M)$$

$$RMS \;\; Error = \sqrt{\sum_{i=1}^{n} \frac{(calculated\_position - actual\_position)^2}{n}} \quad (M)$$

The RMS error will be used to find the overall mean error of a specific device, an indication of precision level as tabulated in [5.8] [5.9] [5.14].

Table 5.4 Statistics of Tektelic GRS Results. Here 'overall' denotes, consolidating all devices

Overall Geolocation Statistics of Tektelic GRS					
Statistics	num_of_gateways_used	hdop	mean_snr_db	mean_toa_u_nsec	error_m
count	235.000000	235.000000	235.000000	235.000000	235.000000
mean	5	0.980700	-2.036819	177.654672	801.060940
std	1	1.215201	3.503592	44.460337	865.974647
min	3	0.143293	-7.083784	67.357484	44.060951
25%	5	0.208843	-4.698464	172.273643	91.843546
50%	6	0.473353	-1.849440	186.778139	264.623931
75%	6	1.229616	-0.926246	205.748658	1588.136948
max	9	4.119843	6.140586	235.487706	2457.999038

Table 5.5 Statistics of Semtech GRS Results. Here 'overall' denotes, consolidating all devices

Overall Geolocation Statistics of Semtech GRS					
Statistics	num_of_gateways_used	hdop	mean_snr_db	mean_toa_u_nsec	error_m
count	581.000000	101.000000	581.000000	581.000000	581.000000
mean	3	2.306931	-1.859155	152.534578	1686.906153
std	1	2.608635	4.420420	54.028193	1157.408975
min	1	0.510000	-10.268000	67.357484	37.277132
25%	3	1.170000	-5.001451	87.704507	724.839210
50%	3	1.360000	-1.408163	172.059811	1548.728132
75%	4	2.050000	-0.471544	201.475000	2607.984916
max	8	10.840000	6.140586	235.487706	4991.893441

Table 5.6 Statistics of Komro GRS Results. Here 'overall' denotes, consolidating all devices. It is obvious that HDOPs of Public resolver differs from Komro GRS, this is due to use of different co-ordinate system in each resolvers where the degrees are converted into meters

Overall Geolocation Statistics of Komro GRS					
Statistics	num_of_gateways_used	hdop	mean_snr_db	mean_toa_u_nsec	error_m
count	9406.000000	9406.000000	9406.000000	9406.000000	9406.000000
mean	3	0.131881	-1.098208	152.353575	1277.359607
std	1	0.495342	4.708273	59.406542	1153.024737
min	3	0.000107	-12.120000	26.533333	25.447052
25%	3	0.000661	-5.021863	87.605653	249.815081
50%	3	0.001104	-1.505070	162.899492	1019.459273
75%	4	0.002678	4.087377	203.000000	2100.581184
max	8	3.779490	9.022222	346.883333	4999.544883

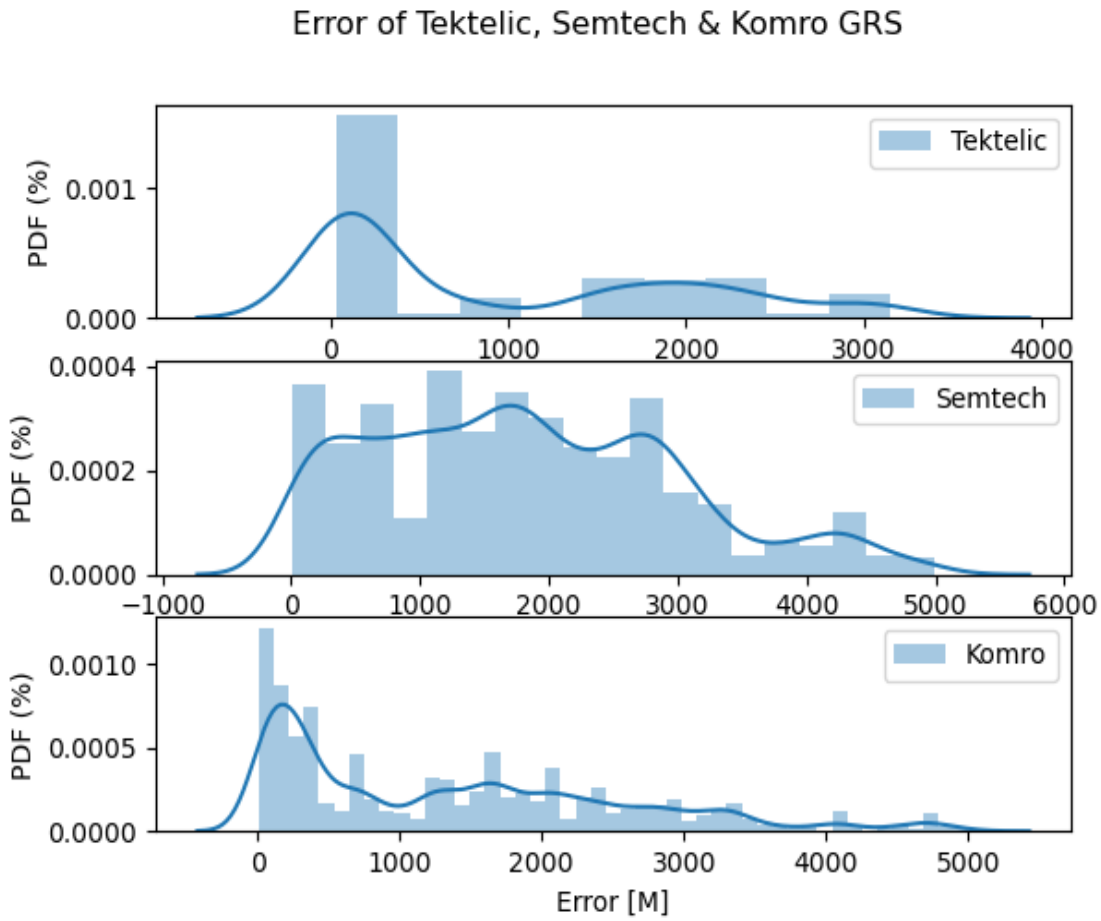


Figure 5.9 Error of Tektelic, Semtech & Komro GRS shows that tektelic has better distribution compared to other GRS although, it has only few datapoints. Tektelic resolver has many constraints so that many devices are not getting valid results via HTTP request/response method (shows an error message 'The end-device location(s) cannot be estimated. The algorithm did not converge'). The numerical values of this figure is shown in tables [5.4] [5.5] [5.6]

Table 5.7 Comparison of minimum error achieved by Tektelic, Semtech & Komro GRS with respect to type of geoloc data filter used 4.3.3

Minimum Errors achieved [M]			
Type of geoloc data filter used	Tektelic	Semtech	Komro
Filter mode 2 & Singleframe/ Mobile	68	11	21
Filter mode 2 & Multiframe/ Stationary	72	266	62

Table 5.8 Geolocation Error for each devices using Tektelic GRS is shown here.'-' denotes that the algorithms have not converged for these devices, meaning that no valid HTTP geolocation response were received. Here 'RMS Error' shows the precision & 'Min. Error' shows the maximum accuracy achieved [5.1.1]. Here 'no. of gateways used' is specific for min. error achieved. The devices namely LuG_FWI21, LuG_FW111 & ERSN11 were placed in basements (@ lower altitudes). It is important to note, while requesting geolocation via HTTP, all the available geoloc packets of a device is sent, regardless of geoloc data filter mode used (as shown in 4.3.3), because tektelic pre-processing will be decided based on the Boolean flag 'mobile' or 'stationary' which is mandatory for each geolocation requests as shown in 4.1.4. As one can see, most devices have minimal error almost close to RMS error, indicating the consistency or precision level of Tektelic GRS.

Geolocation error for each devices using Tektelic GRS			
Device Name	No. of gateways used	Min. Error [M]	RMS Error [M]
ERS1	3	68	68
ERSN2	8	678	1974
ERSN7	2	-	-
ERSN11	6	42	56
ERSN16	3	1526	1547
LuG_FW4	-	-	-
LuG_FW111	-	-	-
LuG_FW20	6	28	45
LuG_FWI21	1	-	-
DZG_Zaehler001	3	55	55
SensoneoWastesensor006	6	128	193
SensoneoWastesensor037	6	984	2310
SensoneoWastesensor022	5	2328	2828
SensoneoWastesensor029	5	1917	1942
Stockwaage3	1	-	-
Stockwaage4	9	80	93
Stockwaage8	1	-	-
Stockwaage10	9	16345	17175
Stockwaage18	8	5633	5864
TIS1	5	144	211
TIS15	-	-	-

Table 5.9 Geolocation Error for each devices using Semtech GRS is shown here.'-' denotes that the algorithms have not converged for these devices, meaning that no valid HTTP geolocation response were received. Here 'RMS Error' shows the precision & 'Min. Error' shows the maximum accuracy achieved 5.1.1. Here 'no. of gateways used' is specific for min. error achieved. The devices namely LuG_FWI21, LuG_FW111 & ERSN11 were placed in basements (@ lower altitudes). As one can see that some best results were obtained even with 1 or 2 gateways, which shows the resolver constraint level of Semtech GRS as opposed to Tektelic & Komro GRS. However it is important to note, while requesting geolocation via HTTP [4.2.2], the geoloc packets of a device is sent based on geoloc data filter mode used (as shown in 4.3.3), because the HTTP request body is expected/formatted like that

Geolocation error for each devices using Semtech GRS			
Device Name	No. of gateways used	Min. Error [M]	RMS Error [M]
ERS1	6	7	493
ERSN2	3	2849	4303
ERSN7	2	1318	2196
ERSN11	3	39	406
ERSN16	3	1360	1722
LuG_FW4	1	235	1239
LuG_FW111	5	724	724
LuG_FW20	4	32	1245
LuG_FWI21	1	-	-
DZG_Zaehler001	3	101	101
SensoneoWastesensor006	5	10	465
SensoneoWastesensor037	1	1063	2656
SensoneoWastesensor022	2	2037	2403
SensoneoWastesensor029	4	248	1767
Stockwaage3	1	-	-
Stockwaage4	8	28	2339
Stockwaage8	1	-	-
Stockwaage10	4	12307	14867
Stockwaage18	6	2454	3003
TIS1	7	267	1840
TIS15	3	2787	2924

5.1.2 Komro GRS: TDOA vs RSSI vs Fused vs Kalman Filter Results

In this subsection, the errors (inaccuracies) of different geolocation methods (of komro GRS) were shown. Since the difference in geolocation results for each filter modes are negligible, only the singleframe & multiframe results are mainly evaluated in komro GRS. The following tables and figures show the error statistics of Komro resolver and its different methods. The units of each parameters in a table are mentioned followed by '_' in each column names.

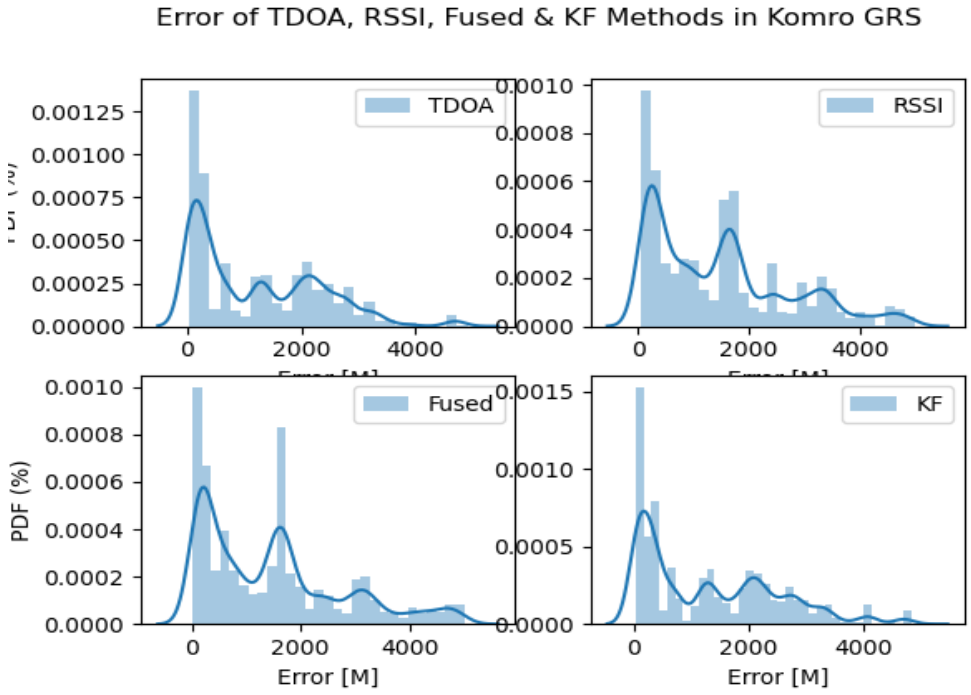


Figure 5.10 Error of TDOA, RSSI, Fused & KF methods were shown here. By neglecting errors above 1000 m, the density of fused & KF errors seem to be better than others. The kalman filter is expected to be more narrower than other methods in Komro GRS, but unfortunately due to lot of uncertainties in every UL (especially in RSSI method accuracy is arbitrary or course, which will also be a current measurement for KF), the error distribution is not yet better. Hence for a LoRa class A device it is always better to trust the TDOA results than others until kalman filter gets better/ stable results. The numerical values for this figure has been shown in tables [5.10] [5.11] [5.12] [5.13]

Table 5.10 Statistics of Komro GRS Results (TDOA method)

Overall Statistics of Geolocation Results of Komro GRS (TDOA method)					
Statistics	num_of_gateways_used	hdop	mean_snr_db	mean_toa_u_nsec	error_m
count	2105.000000	2105.000000	2105.000000	2105.000000	2105.000000
mean	3	0.122451	-1.158166	151.741124	1216.672718
std	1	0.542439	4.746106	58.213848	1136.843165
min	3	0.000107	-12.120000	26.533333	25.447053
25%	3	0.000649	-5.021863	87.704507	147.154532
50%	3	0.001064	-1.437500	162.899492	738.990235
75%	4	0.002408	4.087377	200.587714	2130.036820
max	8	3.779490	9.022222	295.957143	4777.597438

Table 5.11 Statistics of Komro GRS Results (RSSI method)

Overall Statistics of Geolocation Results of Komro GRS (RSSI method)					
Statistics	num_of_gateways_used	hdop	mean_snr_db	mean_toa_u_nsec	error_m
count	2206.000000	2206.000000	2206.000000	2206.000000	2206.000000
mean	3	0.135432	-1.158687	151.855850	1393.003702
std	1	0.564928	4.796722	62.116819	1176.912991
min	3	0.000107	-12.120000	26.533333	57.586491
25%	3	0.000757	-5.021863	87.521357	337.828044
50%	3	0.001139	-1.726928	162.899492	1177.353648
75%	4	0.002911	4.088226	203.448395	1904.977824
max	8	3.779490	9.022222	346.883333	4916.095629

Table 5.12 Statistics of Komro GRS Results (Fused method)

Overall Statistics of Geolocation Results of Komro GRS (Fused method)					
Statistics	num_of_gateways_used	hdop	mean_snr_db	mean_toa_u_nsec	error_m
count	2056.000000	2056.000000	2056.000000	2056.000000	2056.000000
mean	3	0.135259	-1.177732	149.146256	1332.395019
std	1	0.555891	4.875763	60.132487	1212.918813
min	3	0.000107	-12.120000	26.533333	37.991455
25%	3	0.000711	-5.024413	87.497457	207.700668
50%	3	0.001104	-1.706761	152.988818	949.357151
75%	4	0.003029	4.120040	201.176730	1975.279255
max	8	3.779490	9.022222	346.883333	4999.544883

Table 5.13 Statistics of Komro GRS Results (KF method)

Overall Statistics of Geolocation Results of Komro GRS (KF method)					
Statistics	num_of_gateways_used	hdop	mean_snr_db	mean_toa_u_nsec	error_m
count	4329.000000	4329.000000	4329.000000	4329.000000	4329.000000
mean	3	0.197868	-1.168696	154.230492	1274.720593
std	1	0.566601	4.666809	59.477739	1160.834683
min	3	0.000107	-12.120000	26.533333	25.447052
25%	3	0.000707	-4.970811	87.771302	209.247384
50%	3	0.001139	-1.584426	171.091486	1059.851888
75%	4	0.003129	4.080682	201.700000	2129.993681
max	8	3.731004	9.022222	346.883333	4964.306564

Table 5.14 Geolocation Error for each devices using Komro GRS is shown here. The 'RMS Error' shows the precision, 'Min. Error' shows the maximum accuracy achieved [5.1.1] & 'method used for best result' shows which method is more accurate. Here 'no. of gateways used' is specific for min. error achieved. The devices namely LuG_FWI21, LuG_FW111 & ERSN11 were placed in basements (@ lower altitudes)

Geolocation error for each devices using Komro GRS				
Device Name	No. of gateways used	Min. Error [M]	RMS Error [M]	Method used for best result
ERS1	4	43	142	Kalman Smoother Method
ERSN2	4	444	4304	Kalman Smoother Method
ERSN7	3	134	1253	TDOA Method
ERSN11	4	48	383	Fusion Method
ERSN16	3	1231	1462	Kalman Smoother Method
LuG_FW4	4	26	874	Kalman Smoother Method
LuG_FW111	5	127	318	Fusion Method
LuG_FW20	4	25	706	Kalman Smoother Method
LuG_FWI21	1	-	-	-
DZG_Zaehler001	3	53	129	TDOA Method
SensoneoWastesensor006	5	8	544	Fusion Method
SensoneoWastesensor037	5	192	3199	TDOA Method
SensoneoWastesensor022	3	650	2525	Fusion Method
SensoneoWastesensor029	3	507	1673	Kalman Smoother Method
Stockwaage3	1	-	-	-
Stockwaage4	5	28	1884	TDOA Method
Stockwaage8	1	-	-	-
Stockwaage10	3	11372	17546	TDOA Method
Stockwaage18	3	1094	2458	Kalman Smoother Method
TIS1	3	255	2770	Kalman Smoother Method
TIS15	3	1659	3615	Kalman Smoother Method

Table 5.15 Comparison of minimum error achieved by Komro GRS in various methods with respect to type of geoloc data filter used [4.3.3]

Minimum Errors achieved [M]				
Type of geoloc data filter used	TDOA	RSSI	Fused	Kalman Filter
Filter mode 2 & Singleframe/ Mobile	63	121	67	21
Filter mode 2 & Multiframe/ Stationary	62	90	96	86

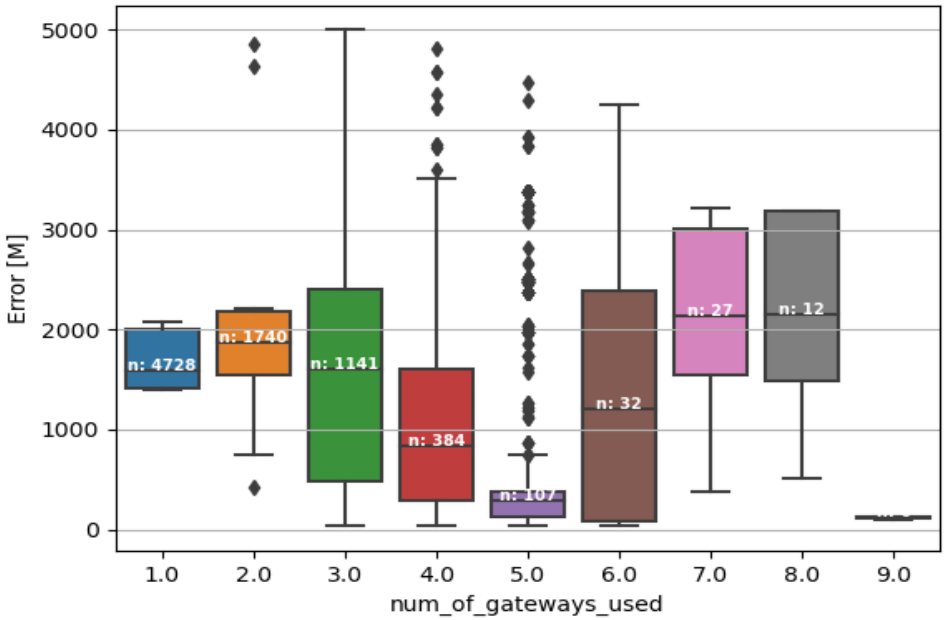


Figure 5.11 Boxplot of Number of gateways reached vs Error (no linear proportion) is shown here. 'n' is the number of observations/ datapoints. As one can see the error decreases till 5 gateways & then again increases. The median error (short horizontal line in boxes) gives an idea according to the quantity of data points available. **Note:** As a general rule, komro GRS resolves only if the nth UL geoloc dataframe has more than 2 gateway_ids

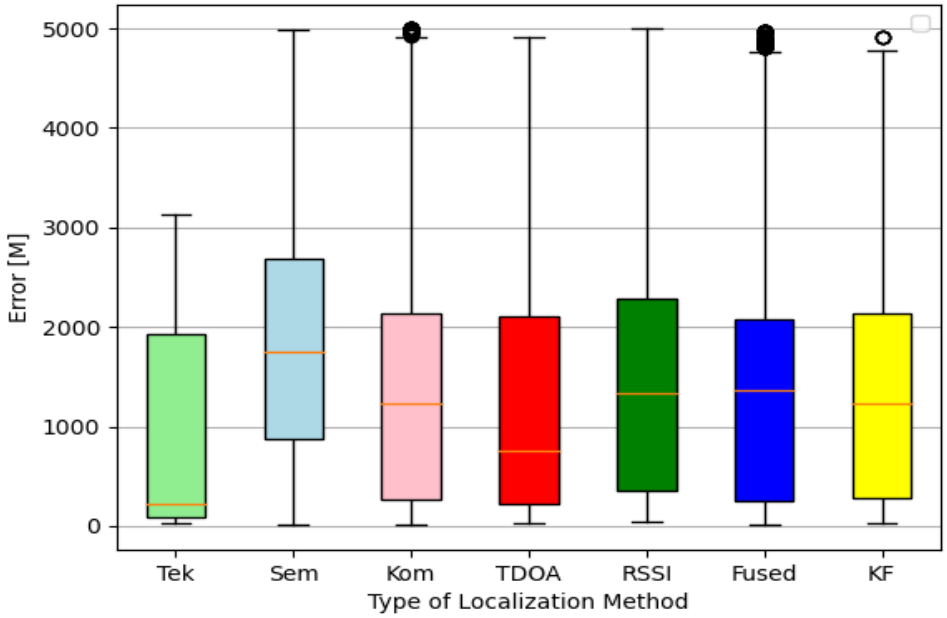


Figure 5.12 Boxplot of Error for various resolvers namely Tektelic, Semtech & Komro and various localization methods such as TDOA, RSSI, Fused & KF is shown here. The median error (short horizontal line in boxes) gives an idea according to the quantity of data points available

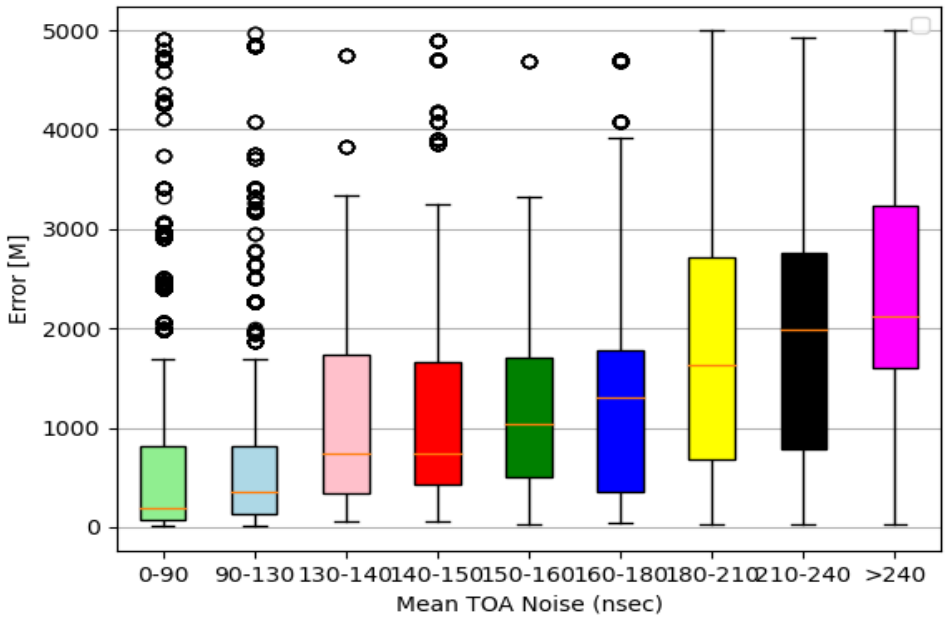


Figure 5.13 Boxplot for TOA Uncertainty vs Error is shown here. The linear proportion between TOA noises & errors were depicted. Therefore it is important and recommended to ignore more uncertain packets for instance above 500 ns

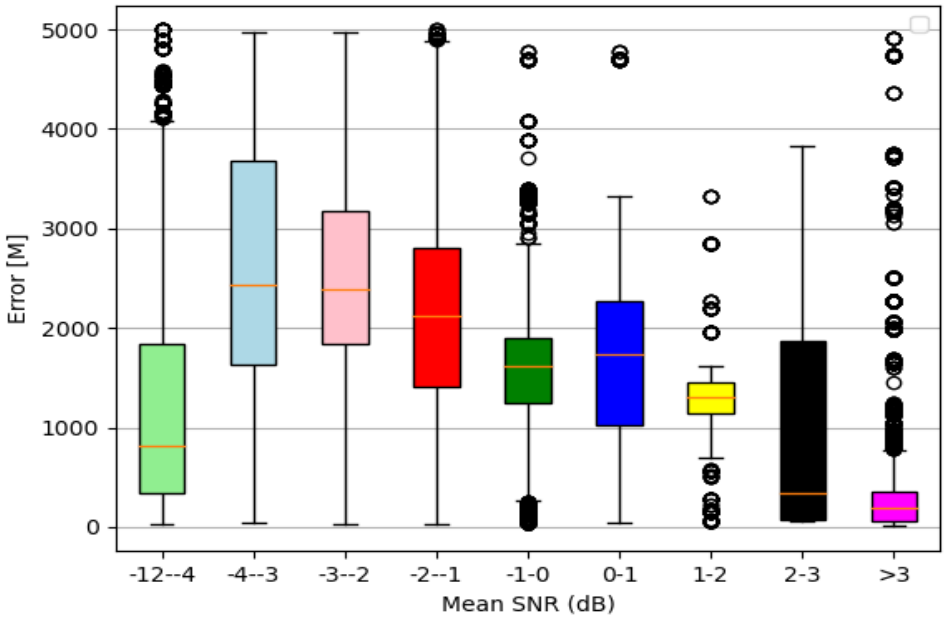


Figure 5.14 Boxplot for SNR vs Error. Partially it is obvious that higher SNR shows lesser error. So this could be one of the reliability indicators for trusting geolocation results

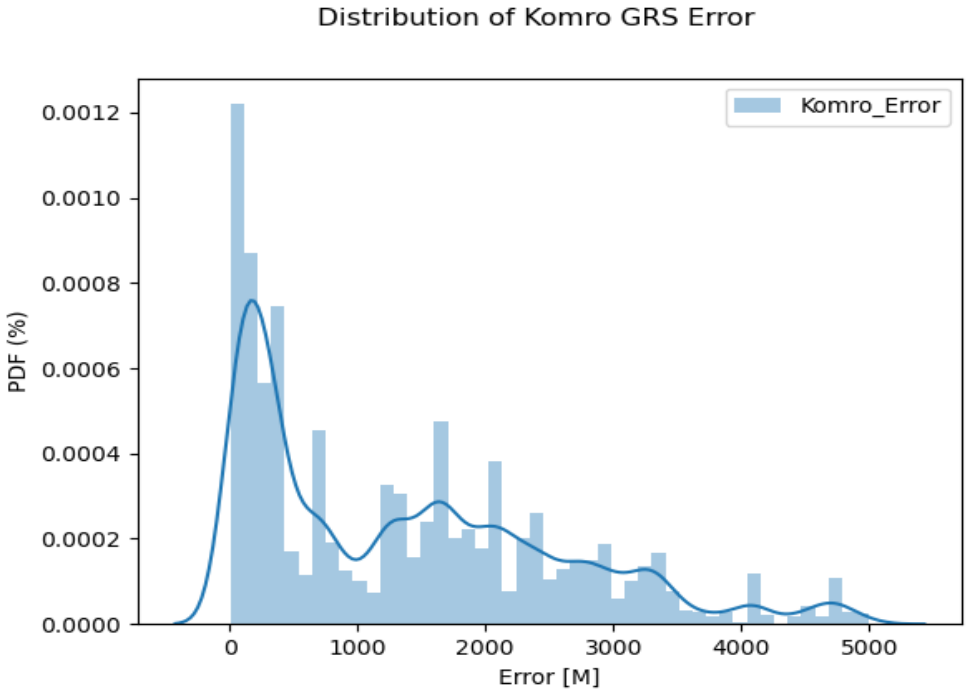


Figure 5.15 Distribution of Komro GRS Error

Table 5.16 Comparison of GRS with respect to devices with minimal error is tabulated here. These are the optimal device locations where the absolute distance between device and gateways is moderate and their relative gateway geometry is very good as shown in figure 5.21, which seems to be suitable for respective resolver algorithms and thereby giving much better accuracy (less than 100 m) favourable for Komro GmbH as well. Although this accuracy happens only for 10% of the time, with constant observations and continuous improvements this accuracy could be maintained

Favourably located devices with minimal error [<100 M]			
Tektelic	Semtech	Komro	Locations
ERS1	ERS1	ERS1	komro office
DZG_Zaehler001		DZG_Zaehler001	komro office
ERSN11	ERSN11	ERSN11	Trafostation P&C
		LuG_FW4	Grubholzer Str. 2 - DM
LuG_FW20	LuG_FW20	LuG_FW20	Äußere Münchner Str.
Stockwaage4	Stockwaage4	Stockwaage4	Schlossberg/Stephanskirchen
	SensoneoWastesensor006	SensoneoWastesensor006	MBB

Table 5.17 Noise indicators in UL signals for different devices located at same place. Notice the relation between mean_snr_db & mean_toa_u_nsec against error_m (RMS). Although the geolocation result data collected for DZG_Zaehler001 is less compared to ERS1 (see count), this comparison could be trusted since the quantity of data points does not influence any trends so far

Noise indicators in UL signals for Stockwaage4 @Stephanskirsche-Schlossberg					
Statistics	num_of_gateways_reached	hdop	mean_snr_db	mean_toa_u_nsec	error_m
count	1815	1815	1815	1815	1815
mean	5	0.034015	-1.867715	216.853651	1692.419773
Noise indicators in UL signals for Stockwaage18 @Stephanskirsche					
count	779	779	779	779	779
mean	4	0.110648	-3.023759	248.814352	2458.966781
Noise indicators in UL signals for ERS1 @Komro building (@ third floor)					
count	6852	6852	6852	6852	6852
mean	4	0.056747	5.488810	82.962189	133.726187
Noise indicators in UL signals for DZG_Zaehler001 @Komro building (@ ground floor)					
count	526	526	526	526	526
mean	3	0.514048	0.784864	91.293112	130.966396

5.1.3 Sources of Errors in TDOA & RSSI Methods

In this subsection, the sources of errors in the main multilateration methods such as TDOA & RSSI were shown.

Source of Error	
TDOA	RSSI
Hardware defect (Timestamp error)	Hardware defect (RSSI error)
Multipath fading delay	Multipath fading mitigation
Interference noise delay	Interference noise mitigation
Conventional TDOA Model	Conventional Path loss (log-distance) Model

The difference between generated TDOA & calculated TDOA (diff_tdoa_rmse) is considered not only as proof for timestamp error (error source in TDOA method) but also the need for multiple single frame resolving instead of collected multiframe resolving. The differential TDOA Error has been calculated for singleframe & multiframe TOA data and are published to geoloc broker in the topic 'geolocation/diff_tdoa_rmse'. From observations, the single-frame tdoa differences are better than multiframes grouped together, because lot of noises are accumulated in this case. The diff_tdoa_rmse will be lesser for accurate TOA data. It is obvious to see that in singleframe TOA data shown in figure 5.16 the diff_tdoa_rmse is less compared to multiframe TOA data shown in figure 5.17, even ideal values like 0.0 ns were seen in singleframe TOA data. That is the errors are less when using singleframe, therefore it is recommended to resolve for multiple singleframes separately and estimate from individual results using kalman filter. **Note:** Multiple singleframes are different from multiframe all together, however in proprietary GRS terminology, the term 'multiframe' is often used to indicate multiple singleframes (calculate solutions separately for multiple singleframes and then estimate).

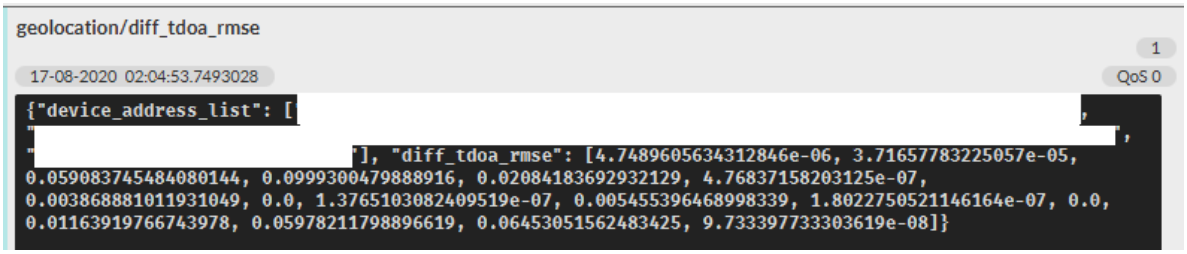


Figure 5.16 Differential TDOA Error for singleframe dataframe

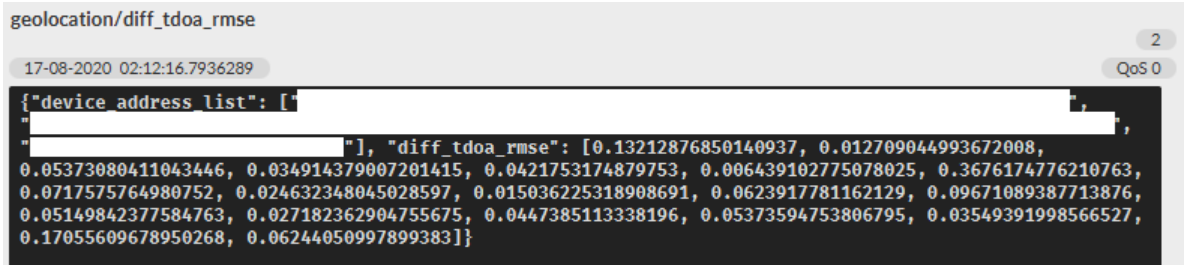


Figure 5.17 Differential TDOA Error for collected multiframe dataframe

5.2 Device-Gateway Geometry for Better Geolocation

Case study of device-gateway geometry in order to achieve best position accuracy will be shown here.

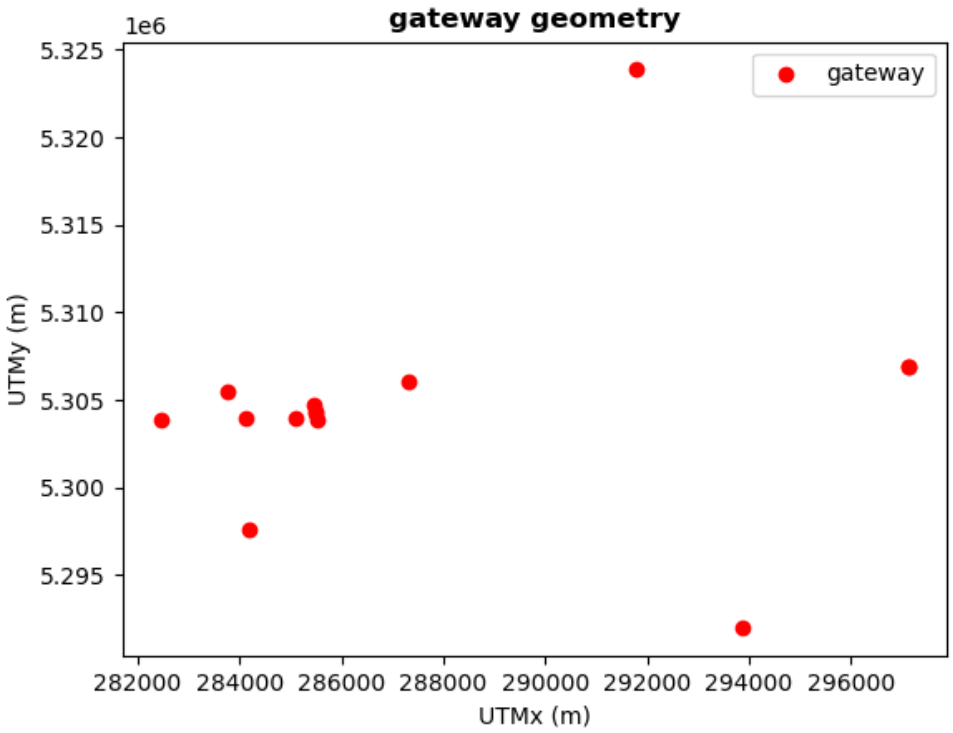


Figure 5.18 Gateways were placed wherever the place is available without specific grid pattern

As shown in figure 5.18, the gateways are placed all over Rosenheim wherever place is available, without any particular strategy or grid pattern. Therefore, some end-devices could not reach more than three gateways in an UL. The device to gateway geometry is qualified by a factor called Dilution of Precision (DOP). The DOP can be classified as Geometric DOP (GDOP), Horizontal DOP (HDOP) & Vertical DOP (VDOP). In 2-dimension, only HDOP is used to measure the quality of device to gateway geometry whereas in 3-dimension, all the DOP mentioned above will be used to measure the quality of device to gateway geometry. The calculation steps are explained in chapter 4.3.5. The DOP range varies from co-ordinate to co-ordinates. For example HDOP in ECEF system, shows a range of values which may completely differ from that of in UTM system. But the proportion will remain, whatever the co-ordinate system is. From observations, the altitude consideration through ECEF co-ordinate does not improve the accuracy compared to UTM co-ordinate. In fact, altitude consideration will slightly decrease the accuracy. Therefore, HDOP (in UTM co-ordinate) is chosen to evaluate the relative geometry of gateways & devices.

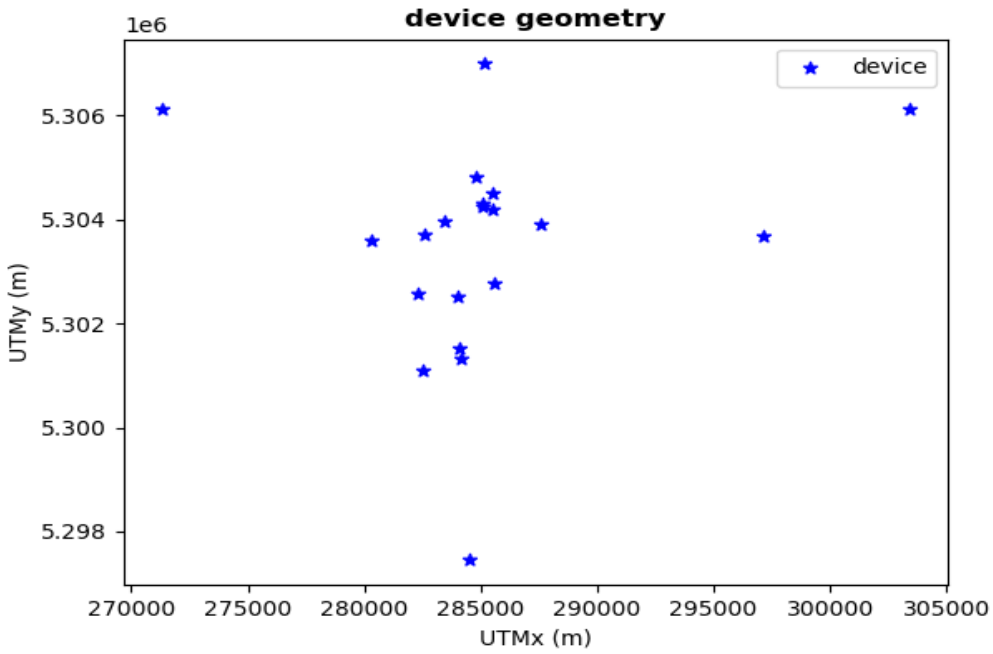


Figure 5.19 End-devices were placed at customer's location arbitrarily around Rosenheim

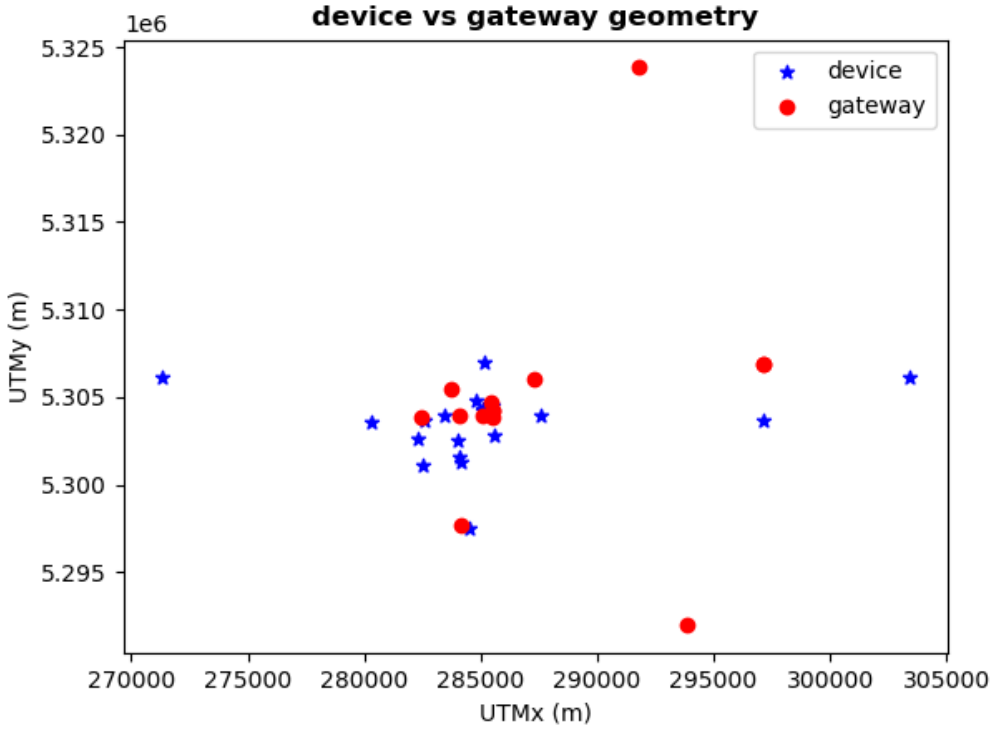
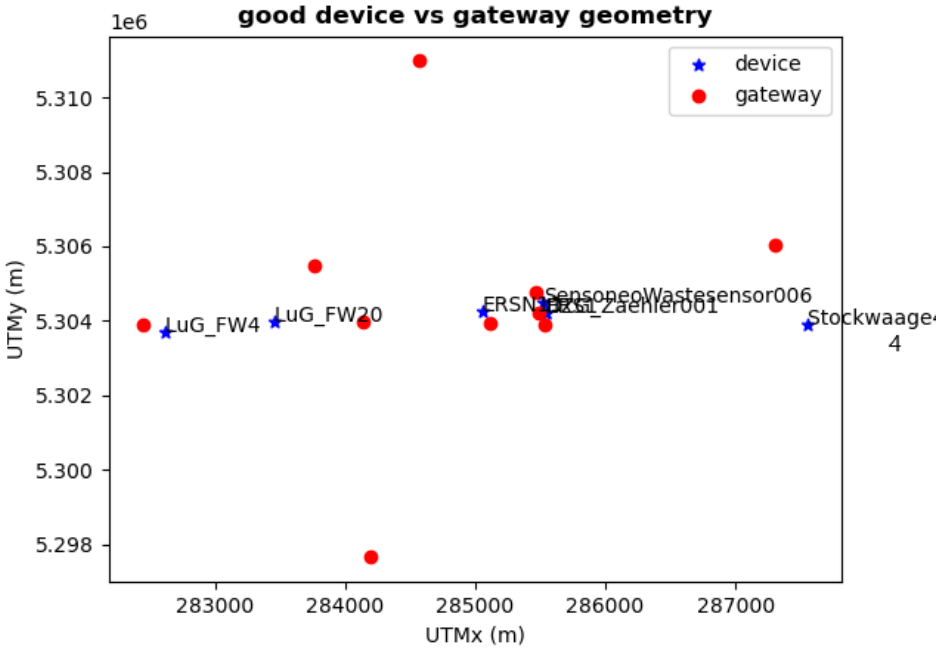
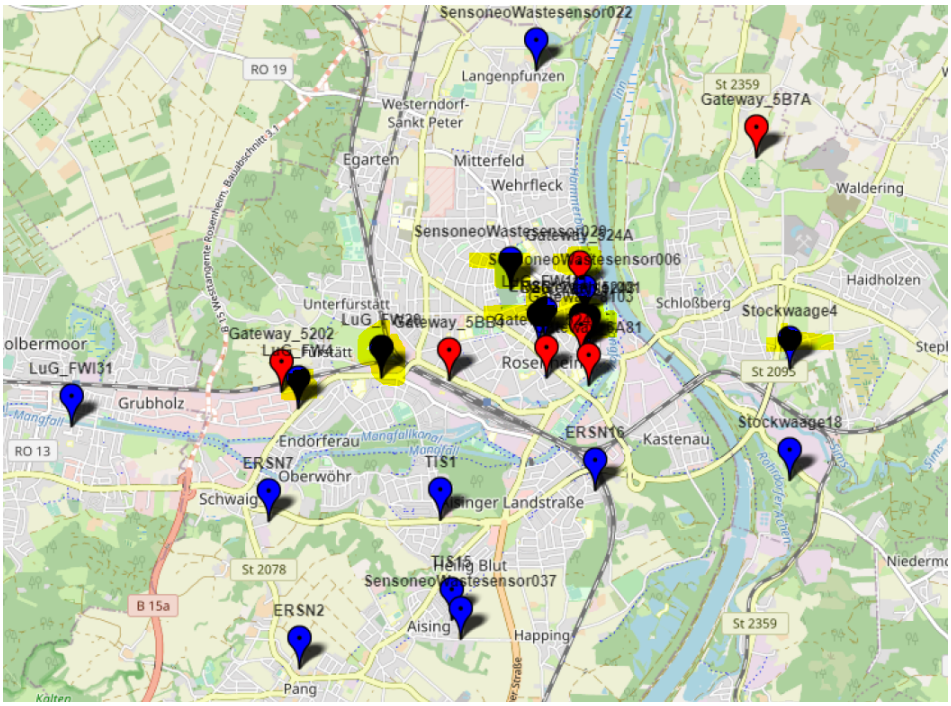


Figure 5.20 End-devices and Gateways were placed without specific grid pattern since it is publicly deployed at various customer locations and free terraces

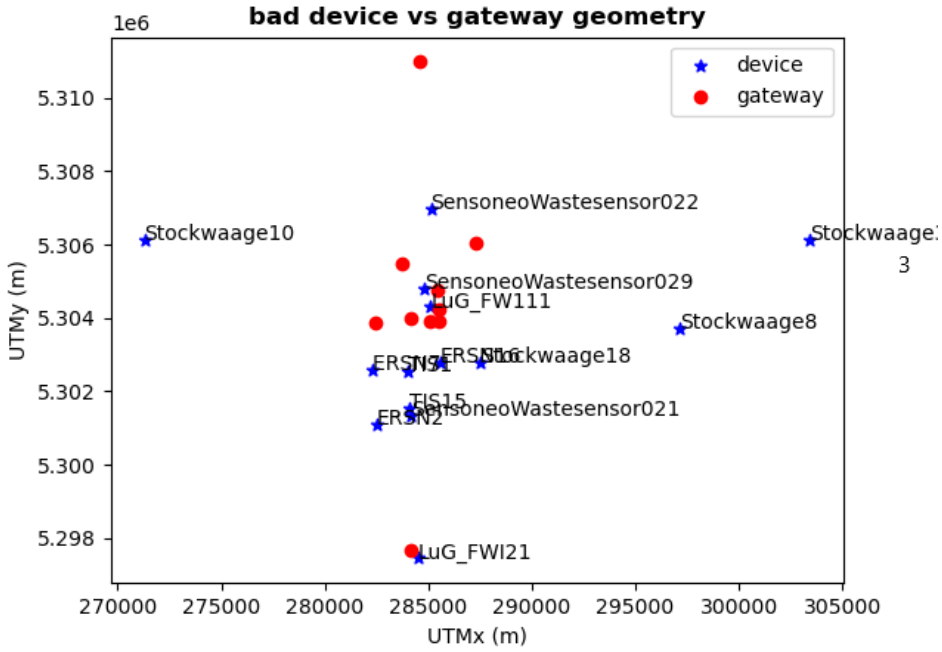


(a) Good devices shown in UTM map

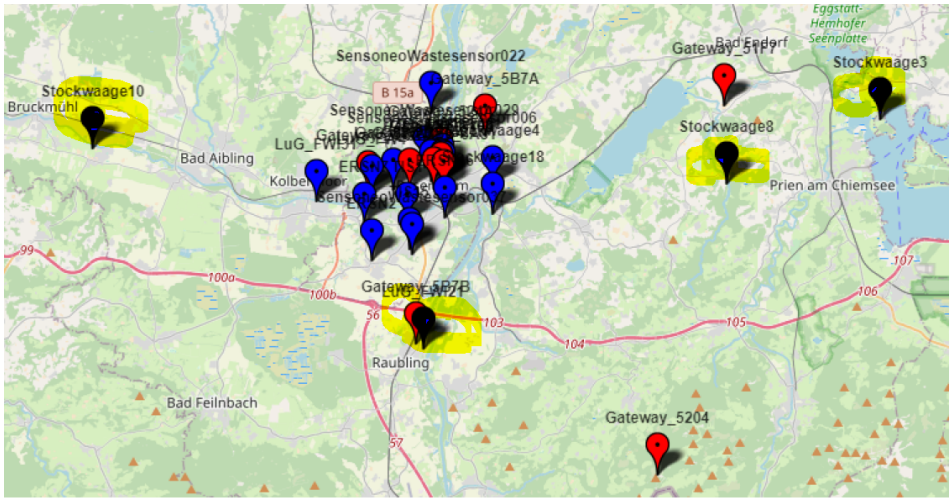


(b) Good devices shown in geographic map. The good devices were darkened & highlighted

Figure 5.21 End-devices with good accuracy and their surrounding gateway geometry were shown here. Here 'good' indicates good accuracy achieved (less than 100 m)



(a) Bad devices shown in UTM map. In order to zoom-in the devices, three gateways on the left half were not plotted in this figure



(b) Bad devices shown in geographic map. The bad devices were darkened & highlighted

Figure 5.22 End-devices with bad accuracy and their surrounding gateway geometry were shown here. Here 'bad' indicates bad accuracy achieved

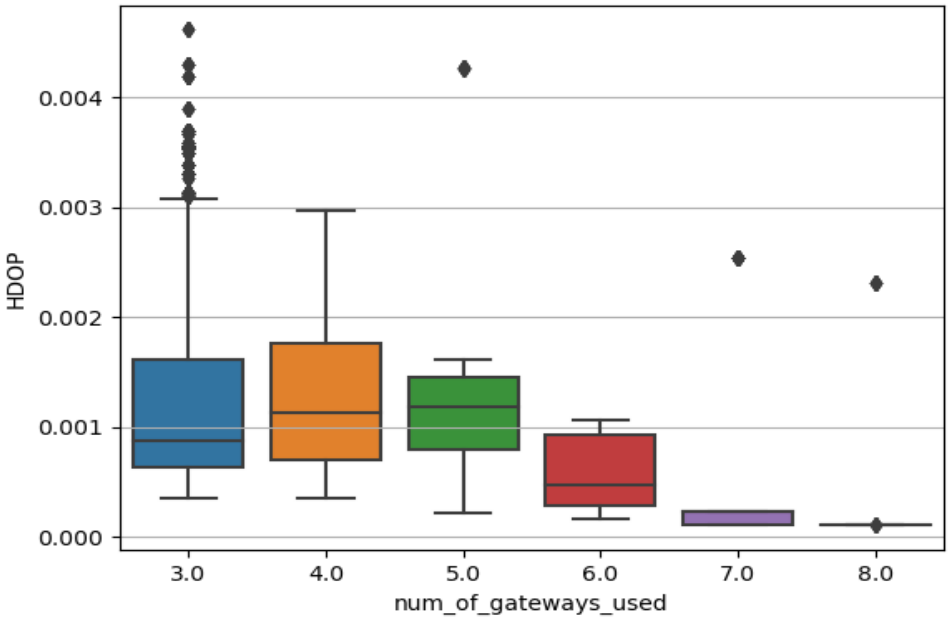


Figure 5.23 Boxplot of Number of gateways used vs HDOP shows that HDOP decreases as the number of gateways increases. The median error (short horizontal line in boxes) gives an idea according to the quantity of data points available

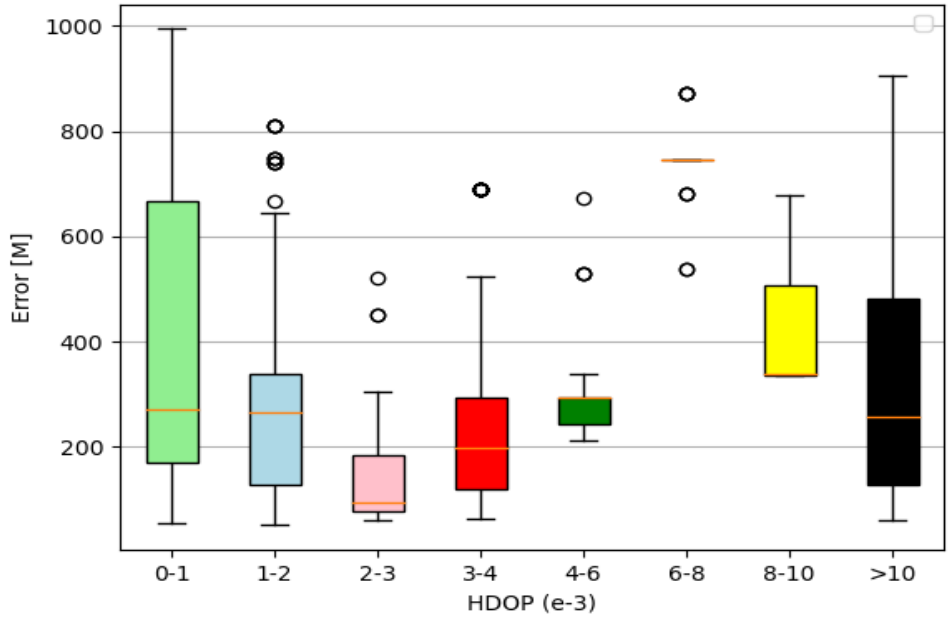


Figure 5.24 Boxplot of HDOP (e-3) vs Error shows that there is no linear proportion between Error and HDOP. In theory, the HDOP may affect the accuracy, but it is not the one & only factor. The median error (short horizontal line in boxes) gives an idea according to the quantity of data points available

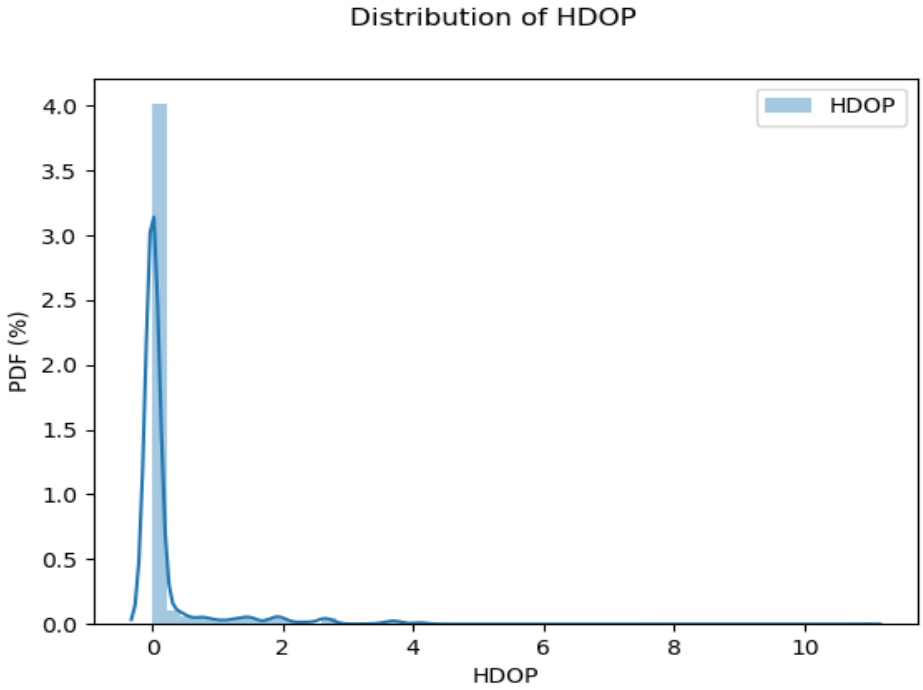


Figure 5.25 Distribution of HDOP shows that the komro gateways were deployed very good in terms of geometry. According to theory, HDOP<1 means, very good deployment and accuracy will be better. This is because all the gateways were placed wide apart from each other (well spread deployment)

5.3 LoRa Parameters for Consistent Geolocation

Experimentation and tuning of the LoRa parameters in order to achieve consistent uplink transmission will be shown here.

The more frequent the geolocation packets, the more reliable the geolocation results will become with the help of kalman filter, it is pretty much similar to time averaging. Number of singleframe geolocation results are passed to kalman filter as measurements and the filter will narrow the calculated results to a mean value using Gaussian distribution. For continuous or consistent transmission from end-device, the DR & Tx Power parameters need to be adjusted accordingly. In order to achieve consistent Tx/UL with best data rate & power consumption, LoRaWAN protocol provides a solution with „Automatic data rate decay mechanism“ from the device side. When detected that the device is disconnected from the Network server, the device starts decrementing the data rate step by step from DR5 to DR0 until it connects back to the network server. This mechanism is called **’Adaptive Data Rate’ (ADR)**. By switching the ADR ’ON’, an end-device will adapt its spread factor and data rate according to the downlink acknowledgement from the network server and signal to noise ratio (SNR) of the device. The mechanism of ADR can be classified into two types such as device side & network server side which have been illustrated in [5.26] [5.27].

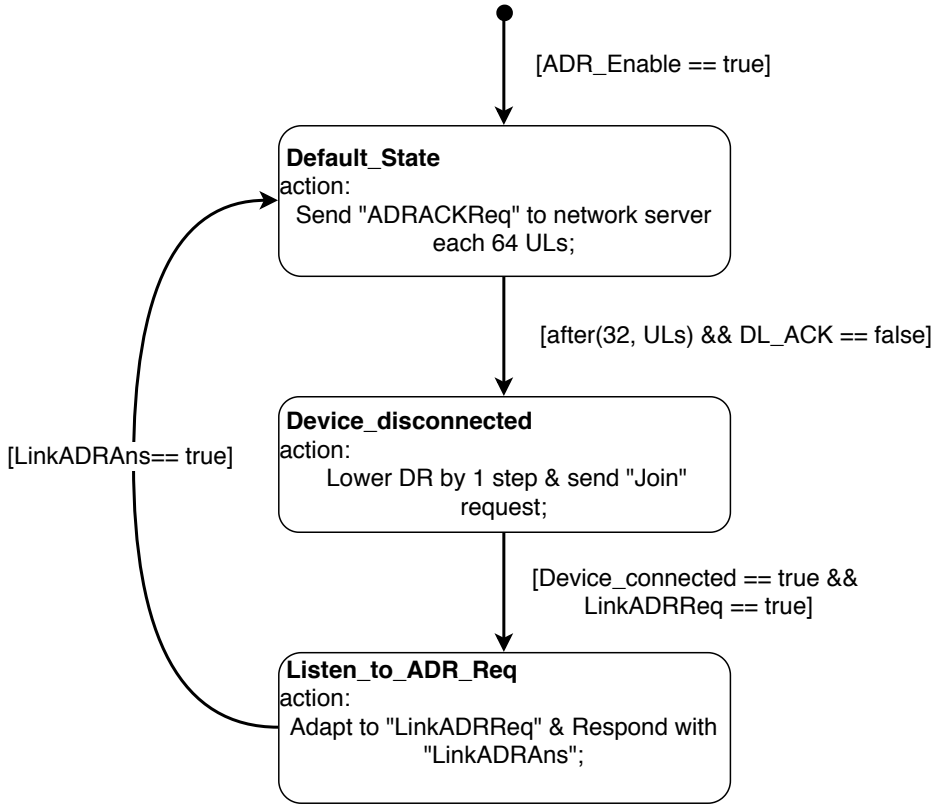


Figure 5.26 Stateflow chart of ADR mechanism on Device Side

Listing 5.1 Device Side ADR Functionality

```
#1. Wait for 'ADR Acknowledgement' from NS and act accordingly
If (ADR(node)==1):
{
Periodically validates that the network is receiving uplink messages with
    the help of frame counter
If (ADR_ACK_CNT == 64):
{
ADRACKReq= 1; (i.e. ask the network whether it is receiving the uplink
    messages or not)
If (ADR_ACK_DELAY (32 UL frames sent) ==1 && No Downlink/ACK message==1):
{
Decrease it's data rate step by step. (make the transmission slower)
}}}
```

Note: Remember the change in Data Rate will reflect from next uplink message. Data Rates of downlink messages (receiver window 01 and 02) are always fixed and controlled by nodes. Data Rates of receiver windows are function of uplink data rates. The ADR feature is enabled for all the evaluated devices in this thesis work.

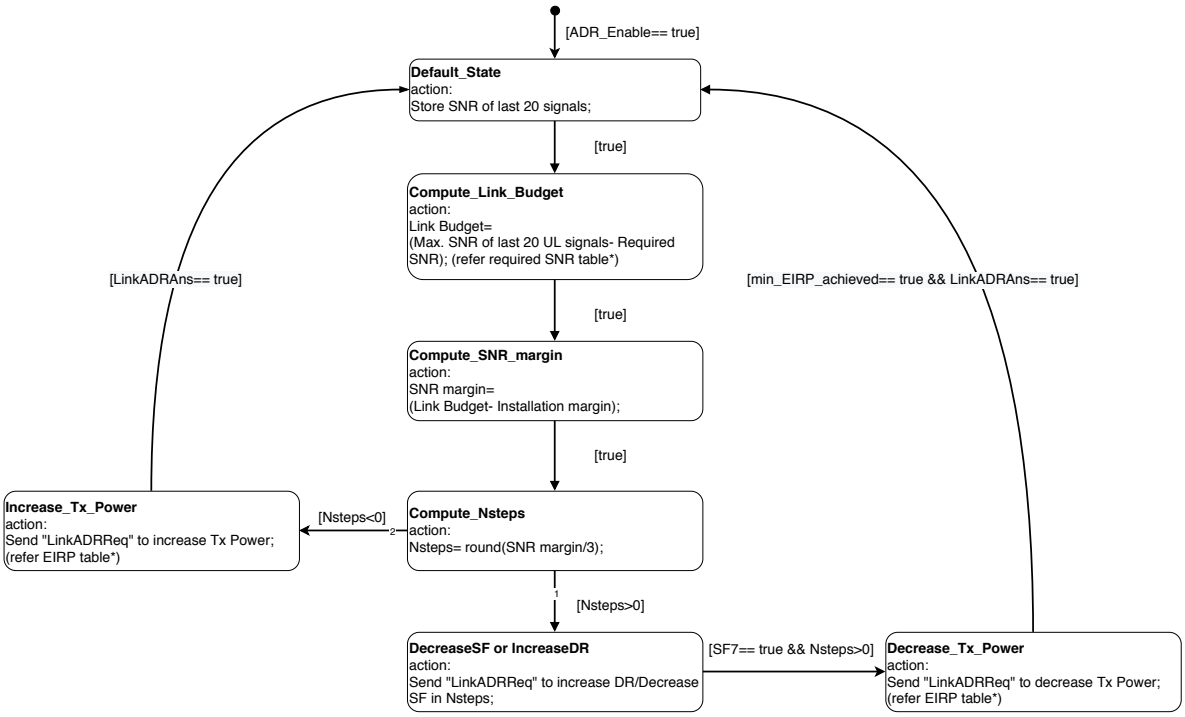


Figure 5.27 Stateflow chart of ADR mechanism on Network Server Side

Listing 5.2 Network Server Side ADR Functionality

```
#1. compute 'max. SNR' of last 20 signals/Rx
If (ADR(node)== 1):
{
```

```
GW/ Network server starts computing the max. SNR (SNRmax) of last 20
    received signals
}
#2. compute 'SNR margin'
SNR_margin = Link_Budget-Installation_margin
Where,
Installation_margin = 10 by default (GW > .conf file)
Link_Budget = SNRmax-requiredSNR
SNRmax = max. SNR of 20 received signals
# Syntax- DR/SF: SNR (dB)
requiredSNR = {DR0/SF12: -20, DR1/SF11: -17.5, DR2/SF10: -15,
                  DR3/SF09: -12.5, DR4/SF08: -10, DR5/SF07: -7.5}
#3. Compute 'Nstep' parameter and act accordingly
Compute Nstep = round(SNR_margin/3)
Ex: Nstep= round(6.3/3)= 2 (for a good signal strength)
If (Nstep < 0):
{
Transmission power is incremented in each step by 3 until max. transmission
     power is reached (Tx max = 14 dBm)
}
If (Nstep > 0):
{
DR is incremented (make the transmission faster) in each step until it
    reaches DR5 and then transmission power is decremented until it reaches
     min. transmission power (Tx min = 2 dBm)
}
#4. Send 'LinkADRReq' MAC command as DL message to end-device (to change Tx
     parameters)
Finally, network server put all these information in LinkADRReq MAC command
     in next downlink message and request end-node to change it's
    transmission parameters.
```

6 Discussion

The use cases, individuals opinion, advantages and disadvantages of the above implementations and experimentation will be discussed here.

6.1 Use Cases

The geolocation in LoRaWAN could be useful for energy efficient applications where the addition of GPS module leads to inefficient & expensive. More important thing is, that LoRa is suitable for normally stationary objects. For real-time mobile objects, accuracy won't be that good even though there is frequent ULs and available parameters such as frequency offset & uncertainties. Some of the most desirable applications are described below:

- **Asset management:** Locating waste containers in municipal corporations is a major application, where municipal workers monitor trash level & act accordingly. Especially, these waste containers need special care while taken for washing, where one may misplace different container instead of expected one, which may mislead a municipal worker who listens only to the sensor data (waste level) & its corresponding original ground truth which would not be true anymore since someone misplaced it in different place. As a result, waste of resources like time, labour & fuel could occur. Not only for waste containers, but also for similar applications where remote monitoring is trusted/implemented. For example, Beehive scale.

- **Pet Tracking:** Locating pet animals, livestock, herds etc. that too with energy efficient class-A LoRa sensor (assuming that the UL happens event based).

- **Theft Tracking:** Locating stolen public LoRa sensors or properties etc.

- **Logistics:** Indoor vehicle localization in an industry, where material flow happens through AGVs equipped with LoRa sensor. Or tracking of municipal vehicles (ex. garbage clearing vehicles)

6.2 Miscellaneous Problems

- **Problem with gateways:** A few downlink acknowledgement messages of NS were treated as uplink geoloc messages of LoRa device by tektelic mega gateways. Although this happens rarely, accuracy will be badly affected due to this. But this has been promptly fixed by appropriate HAL upgrades (BSP) provided by tektelic.

6.3 Advantages

- Localization capability without GPS module or additional BOM cost.
- Geo-Localization feature with low energy consumption, which will be economical for clustered deployments.

6.4 Disadvantages

- Arbitrary accuracy & poor precision over time (inconsistent unlike GPS).
- Not well suited for high dynamic mobile tracking applications (ex. real-time tracking)
- Accuracy highly depends on UL signal noises such as gateway density, multipath fading, interference etc.
- Difficult to deploy gateways in specific grid patterns to cover large group of sensors & gain better accuracy. (especially in urban city applications)
- The least squares algorithm will not converge, if the gateways around a device are far apart from one another (ex. more than 5 kms), because calculated result remains unchanged (settles down) at the gateway which is closest to the device, regardless of device's reach-ability.
- If the device is placed at lower altitudes covered by buildings, then the RSSI and SNR will be very poor leading to poor geolocation results. For instance, the device **DZG_Zaehler** is installed at Komro building in ground floor below the device **ERS1**, although it delivers a good fine timestamps, it does not deliver good RSSI values, as a result it returns an accuracy of around 60 m in TDOA Method similar to ERS1 device and nothing in other methods due to less number of frames per UL (typical example. 4 messages/rows per UL covering 3 gateways can't be used for RSSI geolocation since the result derived would not be reliable. This in turn spoils fusion results), i.e. possibility for having antenna diversity would be less and weak signal strength could be seen from TOA uncertainty, RSSI & SNR values 5.17. Especially, for devices underground/ basements, this method could not be completely reliable.

6.5 Exceptional scenarios

- Although ERSN11 is placed in basement, due to better gateways reach & ADR feature it delivers better accuracy of 50 m [5.14]. Similarly, the device LuG_FW111 is also reaching 5 gateways on average, achieving a minimal error of 127 m.
- Although Stockwaage10 is located far away from densed gateway region (could be referred as 'away from hotspot'), it is able to reach more gateways with the help of ADR feature [5.22] [5.8].

7 Conclusion and Future Work

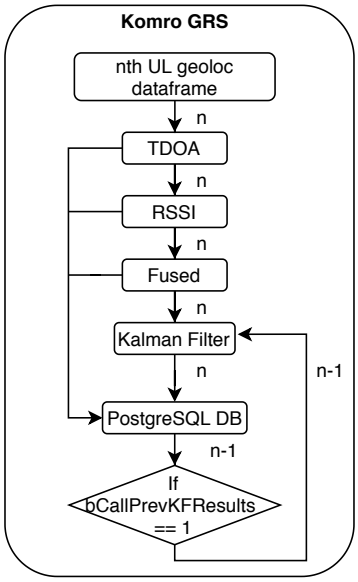


Figure 7.1 Block diagram of Outline of Komro GRS algorithm. Here 'n' denotes time step. Not necessary to be n-1, it could also be n-5 meaning that last 5 KF estimates, especially for stationary devices, this will also improve accuracy or likelihood of the device location. When the sequentially passed ULs are consistent (last 5 ULs), the uncertainty will reduce gradually over time through this filtration structure under filter mode 2 (multiframe) for stationary devices. Currently, last 5 ULs are resolved sequentially in multiframe mode and along with each UL frame 3 previous kf results were passed to kalman filter simultaneously (this could be termed as 5:3 resolver ratio). The control flag 'bCallPrevKFResults' could be enabled if KF results were close to ground truth, otherwise it should be disabled in order to estimate optimal location for each UL frame from TDOA, RSSI & Fusion results @ nth time instance

In this thesis, a novel method has been proposed to localize a LoRa device using Gaussian mixer model (fusion). TDOA and RSSI localization methods are the original inherent geolocation methods upon which the Gaussian fusion is applied in order to reduce the associated uncertainties. The TDOA and RSSI localization methods are based on multilateration technique where the RF signal properties are converted into distances, pseudo range distance in TDOA method (hyperbolas intersection) and true range distance in RSSI method (circle intersection). Having calculated the intersection points (results) of TDOA and RSSI methods, the results are highly subjected to normally distributed noises, in fact the gateway itself delivers uncertainty of TOA data, additionally a statistics is applied on RSSI data to find variance of RSSI between the device of interest (target) and each gateways it covers. The major problem while performing RSSI variance calculation is that number of frames/rows/messages sent from a device to gateway per UL is sometimes

just one, where the statistics operation won't be possible, in this situation the overall variance of RSSI data between a device and all of its gateways are taken for variance calculation, which might generalize the uncertainty characteristics of RSSI of a device. But on the good side, it occurs rarely and furthermore the rssi variance does not differ significantly between both aforementioned scenarios, so its not a big problem for fusing the TDOA and RSSI results. As a result of Gaussian fusion, a more reliable or narrower result can be obtained using noise information (used as weights). The fusion step is similar to measurement update step in kalman filter.

Afterwards, the three results namely TDOA, RSSI, Fused are passed to kalman filter as measurements along with their mean variances (noises), this could be considered as dead reckoning process. The kalman filter further applies expectation maximization algorithm, which reduces the uncertainty of the three measurements in iterative manner given the observed noise information [R]. The kalman filter object is initialized above the 'while' loop where the cyclic execution happens every hour. The KF object updates the measurement and noise for each device and over time it generalizes the Kalman gain (K) which acts as weight (knowledge of noise characteristics) to compensate between uncertainties. When K and variance characteristics become consistent especially in multiframe mode where multiple ULs are resolved sequentially, the KF will produce better or more likely results than other three methods. More importantly, if the overall noise distribution is wider, then the KF will take more time to attain stability or at worst case it may not even converge at all. Currently, the KF is effective only if the measurements are more likely to each other since the ULs are infrequent and no transition matrix is modelled using velocity vector. At some cases, TDOA & RSSI results vary very much where the noise information is also very high leading to wide distribution and coarse estimation (poor). The extraction of noise characteristics is also tricky part, while applying KF or Fusion algorithms. So far only the TOA uncertainty is used as inherent noise information (generated) which is delivered by gateway itself, whereas the RSSI variance is calculated from RSSI data present in UL frames being used for geolocation. Although RSSI is said to be coarse method for geolocation, it has converged up to 57 m accuracy (for SensoneoWastesensor006 [5.14]). By default the resolver uses filter mode 2- multiframe [4.3.3], because all the sensors are stationary. For some devices the previous result influences kalman filer accuracy very much, so it is important to experiment the 'optimal length' (last 5 results) and 'optimal method' (among tdoa, rssi, fused & kf) to be passed as previous results when the control flag '**bPrevCallKFResults**' is enabled. Currently, resolver queries last 3 KF estimates and pass them as previous results parameter along each UL frame simultaneously @ nth time instance as shown in [7.1] & [4.3.4]. Besides the implementation & analysis, a bench marking of public and private GRS were done (Tektelic, Semtech, Komro) to see the differences in accuracy (error statistics) and to reverse engineer the algorithms used by public GRS through comparison.

7.1 Outlook and Future Work

- In the presented thesis work, only TOA and RSSI data were used to model the localization algorithm. But in order to provide the resolver with velocity information

of end-device, it is essential to incorporate inherent RF signal properties such as Frequency Offset (fo_hz) and RMS Frequency Uncertainty (fo_u_hz) into multilateration model (FDOA) and further into kalman filter as **velocity measurements** which will improve localization algorithm especially for moving devices in rural deployment (under mild multipath) due to Doppler effect, that could be the future work [12]. In this thesis work, only noisy **position measurements** of stationary devices along with their uncertainty information were used.

- Since all the evaluated devices are LoRa class A devices, the ULs are not frequent rather their Tx interval (periodicity) is 1 hour, therefore in this scenario the kalman filter object needs longer time to attain stability where the resolver needs to be running for hours/days in singleframe filter mode. A faster stability could be attained if resolver ran in multiframe filter mode or if the resolver is ran for frequent ULs from LoRa class C device, so that kalman filter gains stability faster or even better option would be if the resolver is ran for just one device thereby focusing on the noise behaviour of just one device at a particular place for longer period [4.3.4]. Also, precautions need to be taken while tuning resolver ratio (as described in caption of figure 7.1) in order to avoid over interpretation of results. All of the above would be applicable only if the TDOA & RSSI results are better i.e. when the ULs are less noisy.

- Although the number of gateways does not influence the localization accuracy in greater extend, it could be interesting to implement a weight-based algorithm by using number of gateways used along with noise indicators such as mean SNR (dB), TOA uncertainty (nsec) that are stored alongside each geolocation results which enables observation of trends or relations to error (ex. num_of_gateways_used, mean_toa_u, mean_snr etc.), thus leading to modelling a weight-based algorithm step by step through long-time observations.

- Even though two devices were placed at same place but in different altitudes, the accuracy may vary due to TOA uncertainty, RSSI & SNR differences. In such cases, the LoRa parameters namely DR/SF & Tx Power (i.e. operating modes) have to be tuned/adjusted in order to reach as many gateways as possible with less noises. This would be done automatically by Adaptive Data Rate Mechanism (ADR) [5.26]. Therefore, it is always recommended to enable ADR for all devices in general, especially for devices located in underground basements or at lower altitudes in order to achieve better accuracy, for instance take the scenario with DZG-Zaehler001 vs ERS1 [5.17], where lower altitude device (DZG-Zaehler001) delivered better accuracy with the help of ADR feature. If enabling ADR doesn't enhance Tx consistency (i.e. tendency to reach more gateways with less interference), then the gateway density around that particular place should be increased, for instance check the devices with only one gateway coverage (such as. Stockwaage3, Stockwaage8 & LuG_FWI21) in table 5.14 and their relative gateway geometry in figure 5.22.

- Although the accuracy might slightly depend on other factors such as HDOP, number of UL frames used to resolve (in multiframe filter mode) [4.3.3], type of geoloc data filter used (singleframe/multiframe) etc., it majorly depends on closely deployed

gateway density surrounding the device (in short distances), possibility of LOS environment and noise indicating signals such as SNR & TOA uncertainty. In case of mobile devices, it also depends on frequency offset uncertainty (out of scope in this thesis).

- Compared to ECEF co-ordinates solution (with altitude consideration), the UTM co-ordinates solution (without altitude consideration) shows better accuracy on average.
- Although Tektelic shows better accuracy only for few devices, it has lesser RMS Error, meaning it is more precise and consistent compared to other GRS. This might be due to lot of constraints applied on geoloc packets passed, especially based on the noise indicating signals. This would be further studied and incorporated in Komro GRS as well.

A Annexure

Listing A.1 Singleframe Filtering Function in Geoloc Data Filter

```
#import library
import pandas as pd
#for mobile device: filtering last/recent (also best, with more than 2
    gateway ids) Fcntup instance
def filter_last_fcntup(df):
    global f_index, frame_count, grouped_df, total_unique_gw_id
    f_index = -1
    frame_counts = df.frame_cnt
    unique_frame_counts = frame_counts.unique()
    total_unique_fcnts = len(unique_frame_counts)
    print('total_unique_fcnts:', total_unique_fcnts)
    total_unique_gw_id = len(df.gateway_id.unique())
    no_of_occurences= df['frame_cnt'].value_counts()
    recent_total_frames= no_of_occurences[unique_frame_counts[f_index]]
    # groupby fcntup
    grouped = df.groupby(df.frame_cnt)
    skipped_frames = 0
    total_filtered_frames = 0
    total_unique_gw_id = 0
    if (total_unique_fcnts > 1 or recent_total_frames < 3):
        while (skipped_frames <= total_unique_fcnts and total_unique_gw_id
            < 3):
            f_index = f_index-1
            grouped_df = grouped.get_group(unique_frame_counts[f_index])
            total_unique_gw_id = len(grouped_df.gateway_id.unique())
            total_filtered_frames = len(grouped_df)
            skipped_frames = skipped_frames + 1
    else:
        grouped_df = grouped.get_group(unique_frame_counts[f_index])
        total_filtered_frames = len(grouped_df)
    if bPrint_debug:
        total_unique_gw_id = len(df.gateway_id.unique())
        print('no. of unique gw_ids:', total_unique_gw_id)
        print('no. of unique frame counts:', total_unique_fcnts)
        print('unique_frame_counts:', unique_frame_counts)
        print('no_of_occurences:', no_of_occurences)
        print('recent_total_frames:', recent_total_frames)
```

```
36          print('grouped_df:', grouped_df)
37          print('frame_cnt_used:',grouped_df.frame_cnt.unique())
38          frame_count = grouped_df.frame_cnt.unique()
39      print('frame_cnt_used:', grouped_df.frame_cnt.unique()[0])
40      return grouped_df
```

Listing A.2 Multiframe Filtering Function in Geoloc Data Filter

```
1  #import library
2  import pandas as pd
3  #for stationary device: filtering/grouping last two or more Fcntup
       instances (also best)
4  def filter_multi_fcntup(df, set_fcnt):
5      # always set set_fcnt to 4, so that geolocation of all the unique
           frames will be requested
6      frame_counts = df.frame_cnt
7      unique_frame_counts = frame_counts.unique()
8      total_unique_fcnts= len(unique_frame_counts)
9      total_unique_gw_id = len(df.gateway_id.unique())
10     no_of_occurences = df.frame_cnt.value_counts()
11     # groupby fcntup
12     grouped = df.groupby(df.frame_cnt)
13     unique_frame_counts = grouped.frame_cnt.unique()
14     # filter grouped dataframes acc. to 'set_fcnt'
15     if set_fcnt < 2 or total_unique_fcnts < 2:
16         print('no grouping with one unique frame group, so just returning
               the entire df')
17         grouped_df = df
18         grouped_df = grouped_df[len(grouped_df['gateway_id'].unique())>2]
19         unique_frame_counts = grouped_df.frame_cnt.unique()
20     elif set_fcnt == 2 or total_unique_fcnts == 2:
21         grouped_df= pd.concat([group for (name, group) in grouped if
22                                name in [unique_frame_counts[-1],
                                           unique_frame_counts[-2]]])
23         grouped_df = grouped_df[len(grouped_df['gateway_id'].unique())>2]
24         unique_frame_counts = grouped_df.frame_cnt.unique()
25     elif set_fcnt == 3 or total_unique_fcnts == 3:
26         grouped_df = pd.concat([group for (name, group) in grouped if
27                                 name in [unique_frame_counts[-1],
                                            unique_frame_counts[-2],
                                            unique_frame_counts[-3]]])
28         grouped_df = grouped_df[len(grouped_df['gateway_id'].unique())>2]
29         unique_frame_counts = grouped_df.frame_cnt.unique()
30     else:
31         # assemble all the unique frame counts (ULs) present in device df
32         print('set_fcnt is more than 3, so all ULs frames are grouped &
```

```
            filtered')
33          grouped_df = grouped.filter(lambda x: len(x['gateway_id'].unique())
                >2)
34          unique_frame_counts = grouped_df.frame_cnt.unique()
35      if bPrint_debug:
36          print('no. of unique gateway IDs:', total_unique_gw_id)
37          print('no. of unique frame counts:', total_unique_fcnts)
38          print('unique_frame_counts:',unique_frame_counts)
39          print('no_of_occurences:',no_of_occurences)
40          print('grouped:',grouped_df)
41      print('len_of_unique_frame_counts_after_grouping_best_fcnts:', len(
            unique_frame_counts))
42      return grouped_df
```

Bibliography

[1] Ugur Bekcibasi and Mahmut Tenruh. Increasing rssi localization accuracy with distance reference anchor in wireless sensor networks. *Acta Polytechnica Hungarica*, 11(8):103–120, 2014.

[2] Nuria Blanco-Delgado, Fernando Duarte Nunes, and Gonzalo Seco-Granados. On the relation between gdop and the volume described by the user-to-satellite unit vectors for gnss positioning. *GPS Solutions*, 21(3):1139–1147, 2017.

[3] Paul Bourke. Spheres, equations and terminology, 1992.

[4] Orne Brocaar. Chirpstack geolocation server. `https://github.com/brocaar/chirpstack-geolocation-server`, 2019.

[5] Mosquitto Broker. Mosquitto.conf homepage, 2020. [Online; accessed 25-May-2020].

[6] Semtech LoRa Cloud. Semtech lora cloud documentation, 2020. [Online; accessed 14-July-2020].

[7] LoRa Alliance Technical Committee et al. Lorawan back-end interfaces specification. `https://lora-alliance.org/resource-hub/lorawanr-back-end-interfaces-v10`, 2017.

[8] LoRa Alliance Technical Committee et al. Lorawan specification. `https://lora-alliance.org/resource-hub/lorawanr-specification-v103`, 2018.

[9] LoRa Alliance™ Strategy Committee et al. Lora alliance geolocation whitepaper. `https://lora-alliance.org/resource-hub/lora-alliance-geolocation-whitepaper`, 2018.

[10] Wikimedia Commons. File:geolocation.png — wikimedia commons, the free media repository, 2014. [Online; accessed 12-March-2020].

[11] Wikimedia Commons. File:hyperbolic navigation.svg — wikimedia commons, the free media repository, 2019. [Online; accessed 19-March-2020].

[12] Tektelic Communications. Tektelic lorawan gateways, 2020. [Online; accessed 19-March-2020].

[13] Pykalman Contributors. Pykalman organization. `https://pykalman.github.io/`, 2020.

[14] Mehmet Ali Ertürk, Muhammed Ali Aydın, Muhammet Talha Büyükakkaşlar, and Hayrettin Evirgen. A survey on lorawan architecture, protocol and technologies. *Future Internet*, 11(10):216, 2019.

[15] Emanuele Goldoni, Luca Prando, Anna Vizziello, Pietro Savazzi, and Paolo Gamba. Experimental data set analysis of rssi-based indoor and outdoor localization in lora networks. *Internet Technology Letters*, 2(1):e75, 2019.

[16] MQTT FX Homepage. Mqtt fx gui tool website. *Web resource: https://mqttfx.jensd.de/*, 2020.

[17] Michael Jurasovic. Multilateration. `https://github.com/jurasofish/multilateration`, 2019.

[18] Kevin J Krizman, Thomas E Biedka, and Theodore S Rappaport. Wireless position location: fundamentals, implementation strategies, and sources of error. In *1997 IEEE 47th Vehicular Technology Conference. Technology in Motion*, volume 2, pages 919–923. IEEE, 1997.

[19] Hussein Kwasme and Sabit Ekin. Rssi-based localization using lorawan technology. *IEEE Access*, 7:99856–99866, 2019.

[20] l33tech pte ltd. Lorawan architecture vs lora architecture, 2019. [Online; accessed 19-March-2020].

[21] MATLAB. *Time Difference Of Arrival Positioning Using PRS (LTE Toolbox)*. The MathWorks Inc., Natick, Massachusetts, 2020.

[22] Mobilefish.com. Lora/lorawan tutorial, 2019. [Online; accessed 25-March-2020].

[23] Brian O'Keefe. Finding location with time of arrival and time difference of arrival techniques. *ECE Senior Capstone Project*, 2017.

[24] Behandelt PostgreSQL. Postgresql. *Web resource: http://www.PostgreSQL.org/about*, 1996.

[25] A. Roxin, J. Gaber, M. Wack, and A. Nait-Sidi-Moh. Survey of wireless geolocation techniques. In *2007 IEEE Globecom Workshops*, pages 1–9, 2007.

[26] Harri Saarnisaari and Timo Bräysy. Systematic errors and location accuracy in wireless networks. *EURASIP Journal on Advances in Signal Processing*, 2006(1):081787, 2006.

[27] Prof. Dr.-Ing. Birger Mysliwetz & Prof. Dr.-Ing. Markus Stichler. *Kalman Filtering in Control- & Communication Applications*. Rosenheim Technical University of Applied Sciences, 2020. A course book in master's degree, Automation & Control Technology.

[28] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[29] WrldUnity Website. Wrld-unity, 2020. [Online; accessed 23-May-2020].

[30] Wikipedia contributors. Ecef — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=ECEF&oldid=935356790`, 2020. [Online; accessed 29-March-2020].

[31] Wikipedia contributors. Geolocation — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Geolocation&oldid=942085396, 2020. [Online; accessed 12-March-2020].

[32] Wikipedia contributors. Multilateration — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Multilateration&oldid=945041031, 2020. [Online; accessed 12-March-2020].

[33] Wikipedia contributors. Triangulation — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Triangulation&oldid=941922807, 2020. [Online; accessed 12-March-2020].

[34] Wikipedia contributors. Universal transverse mercator coordinate system — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Universal_Transverse_Mercator_coordinate_system&oldid=941474749, 2020. [Online; accessed 29-March-2020].

[35] Tuo Xie, Hanjun Jiang, Xijin Zhao, and Chun Zhang. A wi-fi-based wireless indoor position sensing system with multipath interference mitigation. *Sensors*, 19(18):3983, 2019.