Localizing Campus Shuttles from One Single Base Station Using LoRa Link Characteristics

Junyang Shi
Department of Computer Science
State University of New York at Binghamton
4400 Vestal Pkwy E, Binghamton, NY 13902
Email: jshi28@binghamton.edu

Mo Sha
Knight Foundation School of Computing and Information Sciences
Florida International University
11200 SW 8th St, Miami, FL 33199
Email: msha@fiu.edu
Telephone: +1-305-348-1717

***Abstract*—Today more and more bus companies are providing real-time bus locations to their riders to improve passenger experience and increase ridership. Most of the existing bus localization systems rely on the Global Navigation Satellite System (GNSS), such as the Global Positioning System (GPS). However, it is costly to install GNSS receivers and retrofit existing buses to power them, which prevents them to be adopted by those bus operators with tight budgets. There has been increasing interest in developing GPS-free localization schemes that leverage the wireless signals transmitted by the buses to localize them. Such schemes often require the received signal strength (RSS) measured at multiple base stations and therefore are not applicable to a small transportation service with a single base station, such as the shuttle service for a university campus. This paper presents a novel approach that leverages the LoRa link characteristics measured by a single base station and deep learning to localize a campus shuttle when it approaches a stop. Experimental results show that our solution provides a detection accuracy of no less than 92.07% and significantly outperforms all baselines without requiring new hardware and introducing additional communication overhead.**

***Index Terms*—Campus Shuttle Localization, LoRa, Link Characteristics, Deep Learning**

I. Introduction

Today more and more bus companies are providing real-time bus locations to their riders to improve passenger experience and increase ridership. For instance, the real-time location of a bus can be used to calculate the expected time of arrival (ETA) at different bus stops. Most of the existing bus localization systems rely on the Global Navigation Satellite System (GNSS), such as the Global Positioning System (GPS). For instance, buses in New York city are equipped with GPS receivers, and users can use the NYC bus checker app to get live bus countdowns and estimated arrival time at any of NYC's 15,000+ bus stops [1]. However, it is costly to install GPS receivers and retrofit existing buses to power them, which prevents them to be adopted by those bus operators with tight budgets. There exist some alternatives that leverage RFID [2] and Bluetooth Proximity Beacons [3] to provide accurate bus locations. Gunady et al. use Bluetooth Low Energy (BLE) proximity beacons to track the location of a bus and BLE detection devices are installed at all bus stops along the route [3]. Extra and expensive hardware is required both on buses and stops to support such designs. The bus operators with tight budget consequently have shown a marked reluctance to embrace them.

There has been increasing interest in developing new localization schemes that leverage the existing wireless signals transmitted by the buses to localize them. For example, Oguejiofor designs an outdoor localization system to automatically locate the position of target devices by measuring the signal strength at an appropriate number of sensor nodes [4]. Plets et al. present a performance comparison of signal strength based and signal arrival time differences based localization approaches in a public outdoor LoRa network [5]. The trigonometry calculation is used to obtain the object location relative to at least three reference points. Such schemes often require the received signal strength (RSS) to be measured at multiple base stations and therefore are not applicable to a small transportation service with a single base station. In this paper, we have developed a LoRa-based localization system by using a single base station and existing LoRa [6] traffic. We present a low-cost solution that leverages the LoRa link characteristics measured by a single LoRa base station and deep learning to localize a campus shuttle when it approaches a stop. Our solution does not require the installation of new hardware and operates solely with the existing LoRa link characteristics. Specifically, our paper makes the following contributions:

- We perform an empirical study that shows the feasibility of using the LoRa link characteristics measured by a single LoRa base station to localize campus shuttles;
- We develop a novel deep learning-based localization method that uses multiple deep neural networks (DNNs) to predict the shuttle location when it arrives at a stop;
- We implement our solution and test it using the real-world campus shuttle monitoring system. We have performed a 14-month empirical study to collect LoRa link traces to evaluate our localization method. Experimental results show that our approach can provide high localization accuracy and outperform several baselines.

The remainder of the paper is organized as follows. Section II introduces the background of our LoRa-based campus shuttle monitoring system. Section III shows our empirical study and Section IV presents the design of our campus shut-

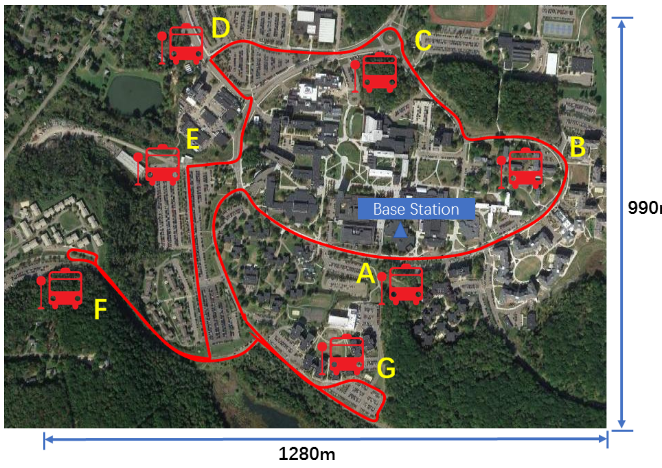


Fig. 1. Campus shuttle route, stops, and LoRa base station.

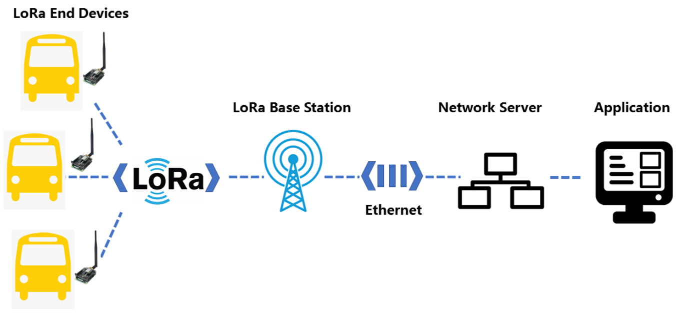


Fig. 2. LoRa network architecture.

tle localization approach. Section V evaluates our approach. Section VI reviews related work. Section VII concludes the paper.

II. Background of Our Campus Shuttle Monitoring System

In this section, we introduce our campus shuttle monitoring system, which relies on a LoRa-based wireless network to collect data from six campus shuttles in a real-time fashion [7]. As Figure 1 shows, all shuttles circle the campus of the State University of New York at Binghamton (a 1280m $\times$ 990m area) using a fixed counterclockwise route and stop at seven shuttle stops (marked as A, B, C, D, E, F, and G in Figure 1).

LoRa is a low-power wide-area network modulation technique that has been initiated by Semtech [8] to build scalable wireless networks. A LoRa module that works at the sub-1 GHz bands can achieve a lifetime of up to 4.5 years with a 2000 mAh battery capacity [9]. The characteristics of LoRa make it a cost-effective solution to monitor the campus shuttles. Because the LoRa coverage range is 10 to 15 km, the characteristics of LoRa radio make it possible for a single LoRa base station to cover our entire campus. Figure 2 shows our LoRa network architecture, which consists of LoRa end devices, a LoRa base station, and a server. The LoRa end devices transmit uplink data messages to the LoRa base station through long-distance LoRa links and the LoRa base

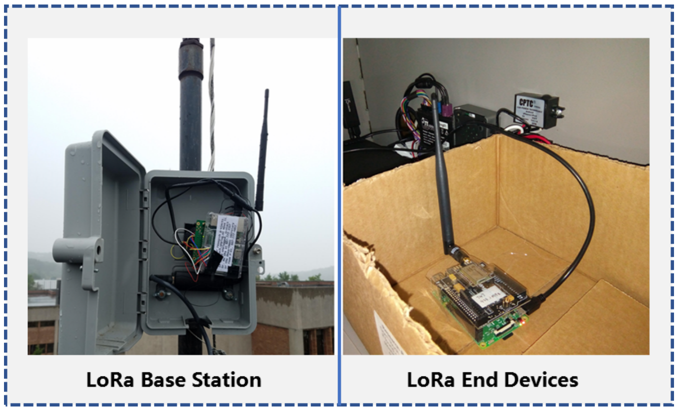


Fig. 3. Hardware deployment for our LoRa network.

station then forwards the messages to the server through an Ethernet connection. The LoRa base station measures each incoming LoRa packet's signal to noise ratio (SNR) and RSS values, periodically computes the packet delivery ratio (PDR) of each LoRa link, and forward the information to the server. The server is responsible for storing the data into a database and running campus shuttle monitoring applications. A user can access the database to view the historical data through an application terminal. The server also generates downlink application messages and passes them from right to left in Figure 2 until reaching the designated LoRa end device. The LoRa base station periodically broadcasts the network management packets that synchronize the time of all LoRa end devices on the shuttles. A fixed channel is used for downlink packets and six dedicated channels are assigned for different LoRa end devices to avoid conflicts.

Figure 3 shows the LoRa base station placed in a weatherproof box on the roof of a three-floor building and a LoRa end device installed in the glove compartment above the driver seat on a shuttle. The LoRa base station and end devices are built by integrating commercial off-the-shelf (COTS) devices:

- The LoRa base station contains a Raspberry Pi 3 Model B (an embedded computer), which is integrated with an iC980A module provided by IMST [10]. The iC980A module is an upgraded version of the IMST iC880A module, which operates in the 900/915 MHz band.
- The LoRa end device installed on the campus shuttle is a Raspberry Pi 3 Model B embedded computer, which is integrated with an RN2903 module [11]. The RN2903 module operates in the 900/915 MHz band that can be configured in either transmission or reception mode each time, and it can operate on a single channel for data transmission or reception. Please note that our LoRa devices operate in the free, unlicensed band.

LoRa Radios provide several configurable physical-layer parameters. The spreading factor (SF) and bandwidth (BW) determine the time duration of a single LoRa chirp transmission. The central carrier frequency (f_c) decides the LoRa signal central frequency for communication. The cyclic redundancy

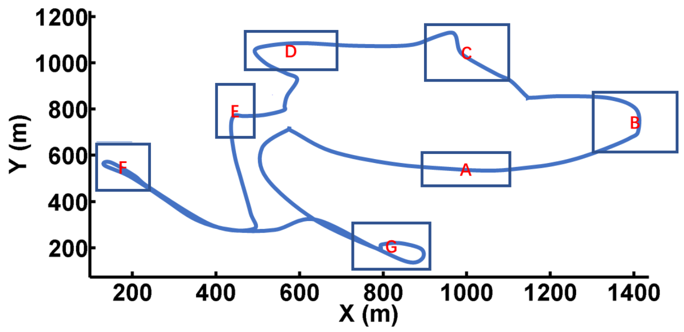


Fig. 4. Seven areas near shuttle stops.

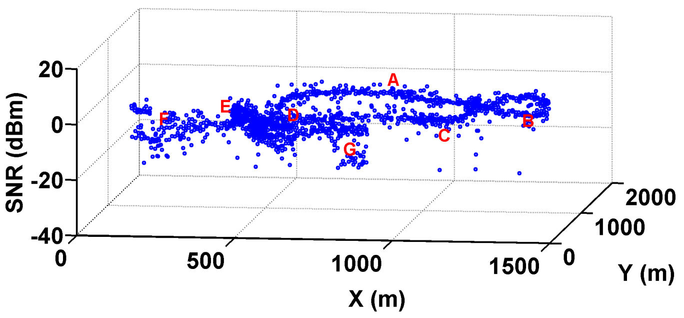


Fig. 6. SNR measurements when the shuttle is at different locations. The shuttle stops A, B, C, D, E, F, and G are marked in red.

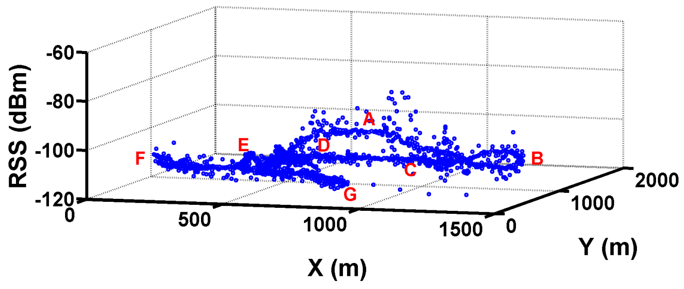


Fig. 5. RSS measurements when the shuttle is at different locations. The shuttle stops A, B, C, D, E, F, and G are marked in red.

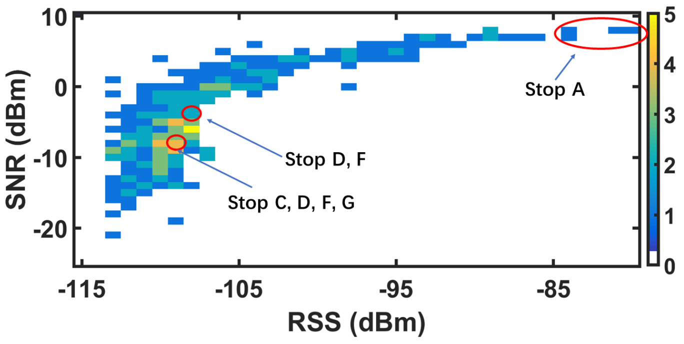


Fig. 7. Heatmap of the number of shuttle stops where we observe the same pair of RSS and SNR measurements.

check (CRC) can be used to verify the integrity of the received packets. The coding rate (CR) is the proportion of the data that carries the useful information, and LoRa uses the Hamming code [12] to provide data redundancy and detect errors. We configure our LoRa devices to use $SF = 9$, $BW = 500KHz$, $f_c = 910MHz$, and $CR = 4/8$, and enable CRC to achieve the best performance.

III. Empirical Study

In this section, we present our empirical study that investigates the feasibility of using the LoRa link characteristics measured at a single base station to localize campus shuttles. To obtain the ground truth, we deploy a GPS receiver on the shuttle, which transmits its current location every 4.5 seconds through the LoRa link. We have performed a 14-month empirical study and collected 248,000 packets in total when six campus shuttle circled the university campus 2,185 times. The LoRa base station measures the RSS and the SNR of each incoming packet and computes the PDR using the sequence ID carried by each packet.

A. Mapping RSS and SNR to Shuttle Stops

Our goal is to localize a campus shuttle when it approaches a stop. Figure 4 plots seven areas, which surround our seven shuttle stops. We first study whether there exist unique RSS and SNR measurements in each area, which allows us to localize the shuttle. Figure 5 and Figure 6 plot the RSS and SNR measurements when the shuttle is at different locations. The RSS and SNR measurements vary considerably in the same area due to the shuttle movement. For example, the RSS measurements varies from -96 dBm to -77 dBm at the stop A and from -113 dBm to -102 dBm at the stop F. Similarly, the SNR measurements varies from -21 dBm to 1 dBm at the stop F and from -19 dBm to 1 dBm at the stop G.

Figure 7 plots the heatmap of the number of shuttle stops where we observe the same pair of RSS and SNR measurements. As Figure 7 shows, the RSS and SNR measurements with high values may help us localize the shuttle when it approaches the stop A, which is close to our LoRa base station.

Observation 1*: Some pairs of RSS and SNR measurements appear only at a single shuttle stop, and therefore can be used to localize the shuttle when it approaches that stop.*

However, most RSS and SNR measurements appear in more than two stops. For example, the combination of $RSS = -109dBm$ and $SNR = -8dBm$ appears in four stops (C, D, E, F) and the combination of $RSS = -108dBm$ and $SNR = -4dBm$ appears in two stops (D and F).

Observation 2*: Most RSS and SNR measurements appear at multiple shuttle stops. It is infeasible to localize the shuttle by simply mapping RSS and SNR measurements to different shuttle stops.*

B. Feasibility of Using Link Characteristics for Shuttle Localization

We further investigate the feasibility of using more link characteristics to localize the shuttle. Figure 8 plots the PDR, the RSS, the RSS variation (RSSV), and the SNR when the

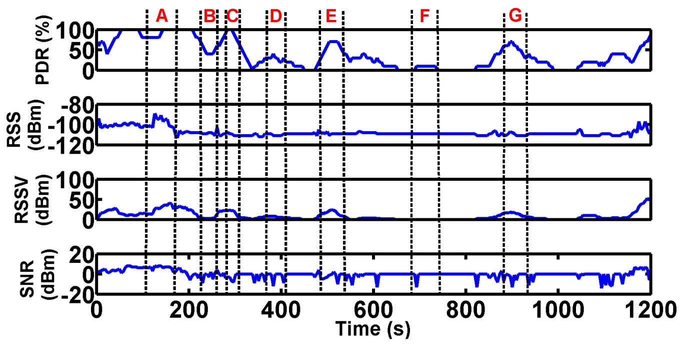


Fig. 8. PDR, RSS, RSSV, and SNR measurements when the shuttle circles the university campus. PDR and RSSV are computed using a 45-second moving window.

shuttle circles the university campus. We have identified all lost and corrupted packets by checking the sequence ID carried by each packet and used them to compute the PDR. The RSSV is computed by following the equation:

$$RSSV = \sum_{k=0}^{8} |rss_{i-k} - rss_{i-k-1}| \tag{1}$$

where rss_{i-k} and rss_{i-k-1} are the $(i-k)$th and $(i-k-1)$th RSS values measured by the LoRa base station. As Figure 8 shows, the PDR is high (100%) when the shuttle is close to the stop A, decreases when it approaches the stop B, and increases back to 100% when it gets close to the stop C. Similarly, the RSS, the RSSV, and the SNR also show some unique changing patterns when the shuttle approaches different stops.

Observation 3*: In addition to RSS and SNR, it is beneficial to use PDR and RSSV measurements to localize the shuttle.*

IV. System Design

In this section, we first present an overview of our campus shuttle localization system and then discuss each engine inside it.

A. System Overview

Figure 9 shows the design of our campus shuttle localization system that runs on the LoRa base station. When the system begins to operate, it enters the training phase, which requires the shuttles to transmit their ground truth locations together with their regular data traffic. The **Data Preprocessing Engine** gathers the link characteristics and the ground truth shuttle locations from the LoRa radio, preprocesses them, and forwards the data to the **Modeling Engine**, which generates models for the **Localization Engine** and the **Reset Engine**. After the deep learning models and reset model generation, the campus shuttle localization system then enters the operation phase and no longer requires the shuttles to collect location data. The Localization Engine and Reset Engine uses the LoRa link characteristics to localize a shuttle when it approaches a shuttle stop. We will next present the design of each engine.

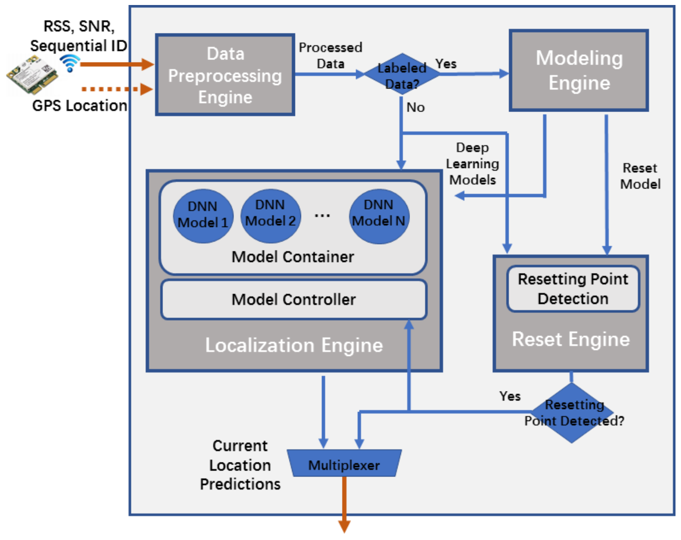


Fig. 9. Campus shuttle localization system.

B. Data Preprocessing Engine

The Data Preprocessing Engine measures the link characteristics from the LoRa base station when it receives an incoming packet. In the training phase, it labels the measured LoRa link characteristics (PDR, RSS, RSSV, and SNR measurements) with the ground truth shuttle location carried by the LoRa packet and then forward the data to the Modeling Engine that trains the localization models. In the operation phase, it forwards the measured LoRa link characteristics to the Localization Engine and the Reset Engine that use the localization models to localize a shuttle. Please note that the ground truth shuttle locations are needed only in the training phase.

Specifically, the Data Preprocessing Engine performs the following tasks:

- Identifying missing packets: All received packets are sorted based on their sequence IDs. Based on the sequence IDs carried by the revised LoRa packets and the predetermined packet transmission interval, the Data Preprocessing Engine can identify all missing packets at the LoRa base station. The missing packets' RSS and SNR values can be represented by the most recent measurements.
- Computing categorical values: The shuttle route is divided into seven blocks, which cover seven shuttle stops. As Figure 4 shows, A, B, C, D, E, F, and G are the seven blocks. All blocks $(Block_i)$ can be represented as a categorical value that is between $[1, 7]$. The categorical values (labels) are used by the Modeling Engine for training.
- Feature scaling: The Data Preprocessing Engine uses min-max normalization method to preprocess all LoRa link characteristics by following the equation:

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{2}$$

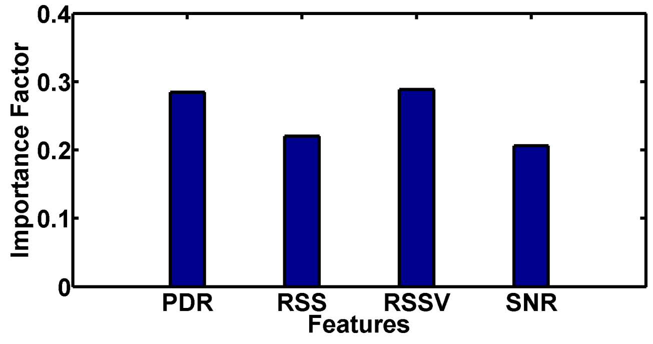


Fig. 10. Importance factors of different link characteristic features when using tree-based feature selection method [13].

where x is the original value, and x' is the normalized one. The Eq. 2 can convert link values into the range between $[0, 1]$. The feature scaling helps to weigh all the features equally and makes the deep learning model converge faster.

C. Modeling Engine

1) Problem Formulation: Our localization system aims to localize a shuttle when it approaches a stop with the following requirements:

- As shown in Figure 1, the campus shuttle runs following a fixed route with seven stops. Our localization system must detect the current location when the campus shuttle approaches a stop as accurate as possible;
- Our localization system must be able to correct its prediction without human involvement;
- Our localization system must rely on a single one base station ;
- Our localization system must introduce no additional overhead and operate with the existing LoRa traffic.

To predict the current shuttle location, we formulate the shuttle localization task as a machine learning problem. Let $\mathbf{x} = concatenation(PDR, RSS, RSSV, SNR)$ denote the given network link characteristics and $\mathbf{y} = P \in (A, B, C, D, E, F, G)$ denote the current shuttle location. Our goal is to learn a nonlinear mapping $f_\theta(\cdot) : \mathbf{x} \rightarrow \mathbf{y}$, which can correctly predict the current shuttle location with the measured LoRa link characteristics as input.

2) Feature Selection: We perform a feature selection study to identify important LoRa link characteristics. The feature importance can represent how useful the input features can contribute to predict the target variable. We have defined four input features (LoRa link characteristics) and one target variable (shuttle location). Figure 10 plots the importance factors of four input features when we use the tree-based method [13] to compute the score. The input features, which are selected at the top of the trees, are more important than those at the lower level of the trees. We plot the normalized importance factors and the sum of all importance factors equals to one. The importance factors are 0.2848, 0.2202, 0.2887, and 0.2064 for PDR, RSS, RSSV, and SNR, respectively. The feature importance scores are in the same order of magnitude. The results indicate all four different inputs can contribute to predict the target variable (i.e., shuttle location). Please note that some input features can be affected by the shuttle running speed. For instance, the PDR is relatively high when the shuttle stops or drives at low speed. The PDR decreases when the shuttle is moving.

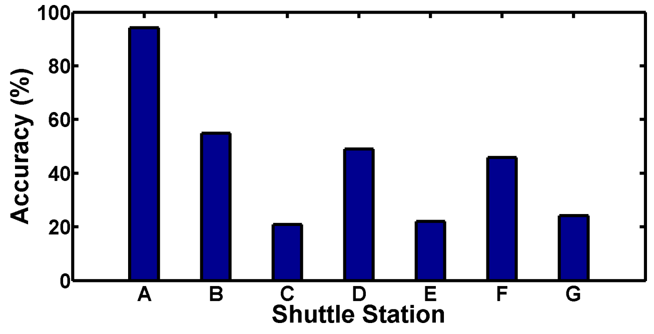


Fig. 11. Location detection accuracy by using DNN classification.

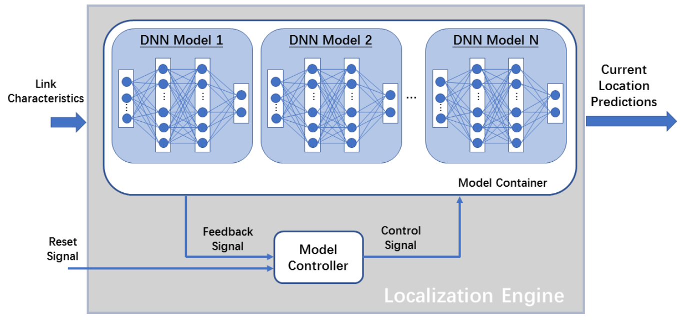


Fig. 12. Multiple DNN models and model controller in the Localization Engine.

3) Simple Classification: The primary goal of our system is to learn a classifier to identify the shuttle locations based on input features. We first try to identify each shuttle stop by using a deep learning model to solve the classification problem. Multilayer Perceptron (MLP) [14] is used to design the architecture of a deep learning model: there are 128 and 64 neurons in the first two hidden layers, and eight neurons in the output layer to represent the seven shuttle stop blocks and transition block. Figure 11 shows the accuracy of shuttle location prediction when the shuttle approaches different stops. The prediction accuracy is 94.20%, 54.84%, 20.90 %, 48.97%, 21.97%, 45.84%, and 24.26% when the campus shuttle arrives at shuttle stop A, B, C, D, E, F, and G, respectively. The average accuracy is 44.43%. The simple approach cannot accurately identify shuttle locations without considering the sequential order of shuttle arrival ($A \rightarrow B \rightarrow ... \rightarrow F \rightarrow G$). For instance, if campus shuttle approaches stop E, then the next prediction should be F instead of other locations.

4) Enhanced Deep Learning Model: To improve the overall performance, the Modeling Engine builds multiple models to detect different shuttle stops and the model controller in the

Localization Engine selects the correct model for prediction (See Section IV-E). Figure 12 shows multiple DNN models computed by the Modeling Engine, which are further stored in the model container. In the training phase, our Modeling Engine trains each DNN model to detect each shuttle stop, independently. Multilayer Perceptron (MLP) is used to design the architecture of each deep learning model: there are 128 and 64 neurons in the first two hidden layers, and two neurons in the output layer to represent the detection status of current stop. The Modeling Engine forwards all models to the localization engine for online localization. We introduce the Localization Engine in Section IV-E.

Algorithm 1: Reset Model Construction Algorithm

```
    Input  : $RSS, SNR, Block_i$
    Output: $Table[][]$
1   $Table[][]$={};
2   if $Table[RSS][SNR]$ is empty then
3       Allocate space for $Table[RSS][SNR]$ ;
4       $Table[RSS][SNR] = \{Block_i\}$ ;
5   end
6   else if $Table[RSS][SNR]$ is not empty then
7       if $Block_i$ is not in $Table[RSS][SNR]$ then
8           Insert $Block_i$ into $Table[RSS][SNR]$;
9       end
10  end
```

5) Reset Model: The Modeling Engine also needs to construct the reset model and forwards the model to the Reset Engine. Basically, the reset model can be used correct the current model prediction. Algorithm 1 shows the reset model construction algorithm. The RSS and SNR measurements are input link characteristics, and $Block_i$ indicates the current shuttle blocks. $Table[][]$ is a lookup table. We implement it as a two-level hash table to map the input (RSS and SNR) to an array of $Blocks_i$. Line 2 to Line 5 of Algorithm 1 initialize the lookup table if it is empty, and allocate space for the current $Block_i$. Line 6 to Line 10 of Algorithm 1 check whether the current $Block_i$ existed in the lookup table. If $Block_i$ is not inside the table, and it will allocate space for the current $Block_i$ and add into $Table[RSS][SNR]$. The Algorithm 1 runs in the training phase and build a map from (RSS, SNR) to the blocks. This map can be used to set rules for the RSS and SNR threshold to detect reset points.

D. Reset Engine

Based on the reset model generated by the Modeling Engine, the Reset Engine can set up RSS and SNR threshold for the reset points. Those reset points possess special values which only appear in specific shuttle stations. If the resetting points are detected, the Reset Engine outputs the current location, which bypasses the Localization Engine's output (See Section IV-E). The Reset Engine also needs to signal (S_{reset}) the model controller in the Localization Engine to reset its current deep learning model. Please note that the same reset points can be detected only when the minimum time of shuttle round interval is satisfied to prevent the repeat correction or error correction.

E. Localization Engine

Algorithm 2: Model Controller Algorithm

```
    Input  : $RSS, SNR, S_{reset}, S_{feedback}$
    Output: $Model_{new}$
1   $Model_{cur} = -1$;
2   if $Model_{cur} == -1$ then
3       //Initialization
4       if $S_{reset}! = -1$ then
5           $Model_{cur}$ = $S_{reset}$ ;
6           Output $(Model_{cur} + 1)\%N$;
7       end
8   end
9   else
10      //Runtime update
11      if $S_{reset}! = -1$ then
12          if Minimum time of shuttle round interval
             satisfied then
13              $Model_{cur}$ = $S_{reset}$ ;
14              Output $(Model_{cur} + 1)\%N$;
15          end
16      end
17      if $S_{feedback}$ == True then
18          $Model_{cur}$ = $Model_{cur} + 1$ ;
19          Output $(Model_{cur} + 1)\%N$;
20      end
21  end
```

The Localization Engine has two major modules, which include the model container and model controller, as shown in Figure 12. The model container stores multiple DNN models and the model controller activate the correct DNN model for the location prediction. Algorithm 2 illustrates the model controller algorithm. The input values are RSS, SNR, resetting signal (S_{reset}), and feedback signal ($S_{feedback}$). $S_{feedback}$ is a boolean value, which represents whether the current stop has been detected or not. When $S_{feedback} == True$, it means the current stop has been detected. Then the value of model selector will increase by one. Line 2 to Line 8 of Algorithm 2 initializes the model selector $Model_{cur}$. If a resetting point is detected (Line 4), $Model_{cur}$ will be updated and the model controller signals the Localization Engine to activate the next DNN model $(Model_{cur} + 1)\%N$, where N is the total number of DNN models in the Localization Engine. Please note that S_{reset} can be an integer value in the range of $[0, N-1]$, which represents different shuttle stops. If the resetting point is detected again and the minimum round time interval is satisfied, the model controller can correct the current prediction (Line 11 to Line 16). If the current stop is detected, $S_{feedback}$ will be true, and the next deep learning model can be activated (Line 17 to Line 20). Please note that

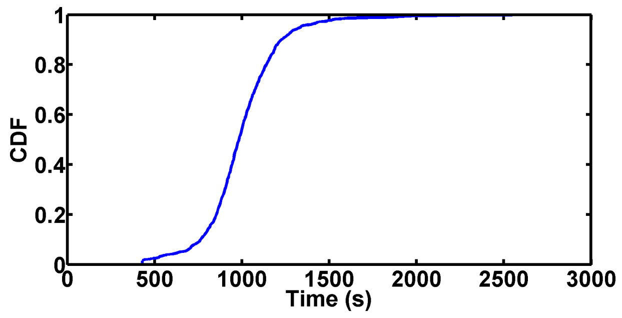


Fig. 13. Campus shuttle round interval.

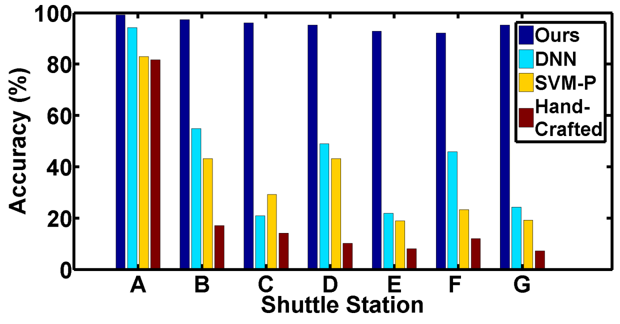
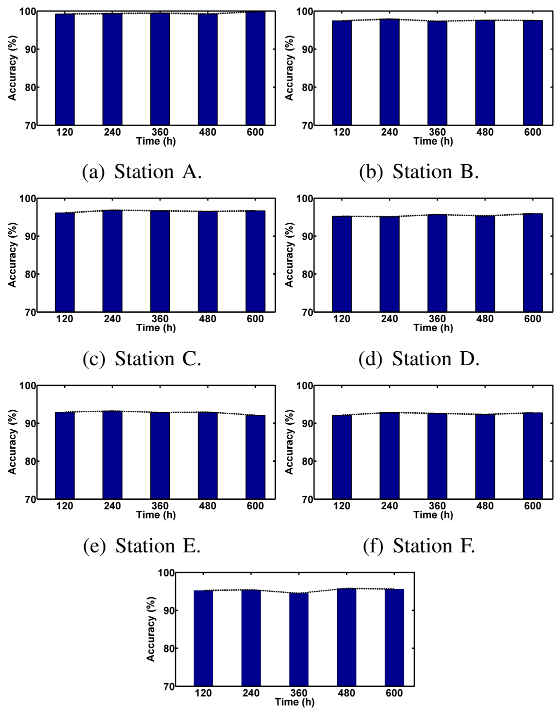


Fig. 14. Detection accuracy under different approaches.



(a) Station A. (b) Station B. (c) Station C. (d) Station D. (e) Station E. (f) Station F. (g) Station G.

Fig. 15. Detection accuracy of our solution over time.

the minimum time of shuttle round interval can ensure that the reset signal will not be triggered many times within a single round. Taking our campus shuttle system as an example, the resetting point at stop A can be detected when RSS and SNR measurements are above the -90dBm and 0dBm threshold. The system contains seven DNN models ($N = 7$) in the Localization Engine, which can make predictions based on current link characteristics. By identifying the current shuttle stop ($S_{feedback}$ is true), the current model indicator is updated ($Model_{cur} = Model_{cur} + 1$), then the Control Engine signals the localization engine to activate the next one. The model selection process repeats when the campus shuttle circles the campus.

V. Evaluation

A. Experimental Setup

We perform a series of experiments to validate the efficiency of our method to locate the shuttle when it arrives at different stop. We compare our method against three baselines: (i) A single DNN model for classification [14]; (ii) SVM-P method [15]; and (iii) hand-crafted threshold-based localization method. As introduced in Section IV-C, we consider a single DNN model to perform localization. The DNN model has 128 and 64 neurons in the first two hidden layers, and 8 neurons in the last layer to represent different shuttle stops and transition blocks. The rectified linear unit (ReLU) is used to activate each hidden layer and softmax is employed to the output layers, respectively. We use the Adam optimizer with a learning rate of 0.01. For the SVM method, we have considered the polynomial kernel (SVM-P) for classification. In our hand-crafted method, we set different rules (i.e., RSS, SNR, and PDR threshold or range) to detect different shuttle stops. The RSS and SNR threshold for stop A is -90 dBm and 0 dBm. The PDR range for stop B, C, D, E, F, and G is [40%, 60%), [90%, 100%], [30%, 40%), [60%, 70%), [10%, 30%), and [70%, 90%], respectively. Our method has seven DNN models in total, each of which has 128 and 64 neurons in the first two hidden layers, and two neurons in the output layer to represent the detection status of current stop. A total number of 100 training epochs have been trained on each DNN model. A different number of training data sets has been used to evaluate our approach against different baselines. We use 60% of our total data collection to test the detection accuracy of our model.

B. Detection Accuracy of Our Solution

In this set of experiments, we measure the shuttle stop detection accuracy of our solution and three baselines. Figure 13 shows the cumulative distribution function (CDF) of round time intervals when the campus shuttle circles the campus. The time interval ranges between 430 seconds to 2718 seconds. The median time interval is 990 seconds. By leveraging the minimal round interval information (See Algorithm 2) and different DNN models to detect the campus at different stops, Figure 14 shows the comparison on the detection accuracy among different solutions. The detection accuracy of our solution is 99.16%, 97.41%, 96.11%, 95.19%, 92.91%, 92.07% and 95.19% at different shuttle stops, respectively. The average accuracy is 71.96% under our solution based on calculation of

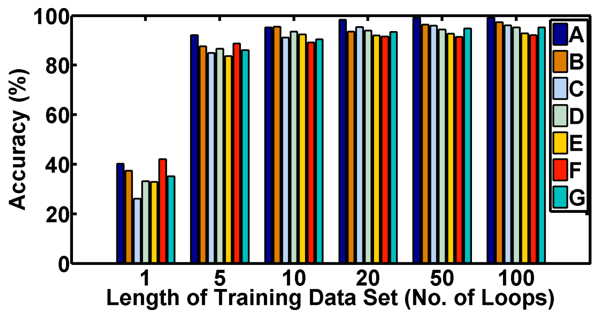


Fig. 16. Performance when using a different number of training data sets.

conditional probability. In comparison, the detection accuracy of a single DNN model is 94.20%, 54.84%, 20.90%, 48.97%, 21.97%, 45.84% and 24.26% at different shuttle stops, respectively. The average accuracy is 44.43%. The SVM-P and hand-crafted solutions have lower average detection accuracy, which are 37.15% and 21.52%, respectively. The detection accuracy of SVM is 82.91%, 43.17%, 29.29%, 43.10%, 18.99%, 23.26% and 19.30% at different shuttle stops, respectively. The results show that our solution consistently outperforms the baselines. The performance improvement benefits from the multiple DNN models and the control engine selection algorithm. We then sort our data by their chronological order and use a time window of 437 rounds to test our solution over time. Figure 15 shows the detection accuracy of our solution over time under different shuttle stations. When the time increases from 120 hours to 600 hours, the accuracy of our solution to detect stop A is 99.01%, 99.24%, 99.08%, 98.63%, 99.54% (as shown in Figure 15(a)). Small changes in accuracy can also be observed at other stations. The results indicate our solution can be applied under different time windows and the detection accuracy does not decrease or vary considerably over time.

C. Performance under Different Amounts of Training Data

We further evaluate the effectiveness of our solution with the different amount of training data. Figure 16 shows the detection accuracy of our solution under a different number of training data sets. When the training data set is one loop, the detection accuracy of A is 40.19%. It further increases to 92.07%, 95.19%, and 99.08% under 5, 10, and 100 rounds of training data, respectively. The detection accuracy increases dramatically when the number of training data sets increases from one to five loops. For instance, the detection accuracy increases by 51.88% at stop A when the number of training data increases from one to five loops. The detection accuracy is 98.25%, 93.59%, 95.42%, 93.90%, 91.91%, 91.53% and 93.36% at different shuttle stops when the training data set is 20 loops. The results show that our solution performs well with a small number of training data sets and the cost of training DNN models is low.

D. Importance of Reset Engine

We perform a study to demonstrate the importance of the Reset Engine to improve the robustness of the system.

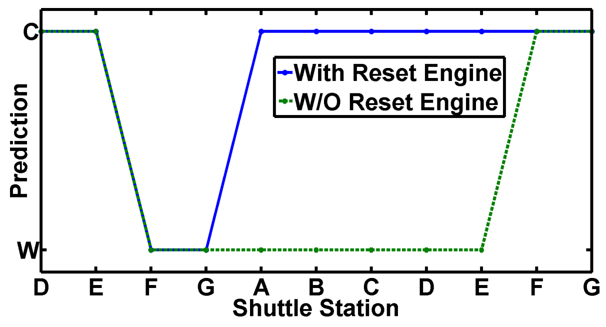


Fig. 17. A micro-benchmark measurement on location prediction with or without the reset engine.

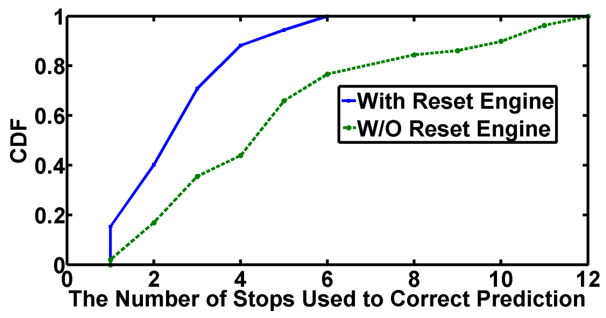


Fig. 18. Recovery overhead.

Figure 17 shows a micro-benchmark measurement on location prediction with or without the reset engine. The C and W on the y-axis represent correct and wrong predictions, respectively. It can be observed that our method can quickly correct the prediction with the reset engine. After the wrong prediction on location F and G, it can correctly predict the location A. However, it takes a whole round for the one without the reset engine to correct its location prediction. Our experimental results demonstrate the importance of the reset engine to help our localization system quickly recover from the wrong prediction.

We further measure the number of stops used to correct prediction with or without the reset engine. Figure 18 shows the CDF of recovery overhead with or without the reset engine, respectively. By enabling the reset engine, the maximum and median recovery overhead are six stops and three stops, respectively. In contrast, the maximum and median recovery overhead are 12 stops and five stops by disabling the reset engine. The results indicate that the reset engine can make the localization system more stable and robust.

E. Time Consumption of Our Solution

We measure the training time of our DNN models. Figure 19 shows the box plot of training time for each DNN model. We run the modeling algorithm on a Dell Linux laptop with the 2.8GHz Intel Core E3-1505M for ten different times. On average, the modeling time is 60.38 seconds, 58.32 seconds, 61.10 seconds, 57.43 seconds, 62.20 seconds, 55.24 seconds, and 60.92 seconds for location A, B, C, D, E, F, and G, respectively. The time consumption of DNN training is

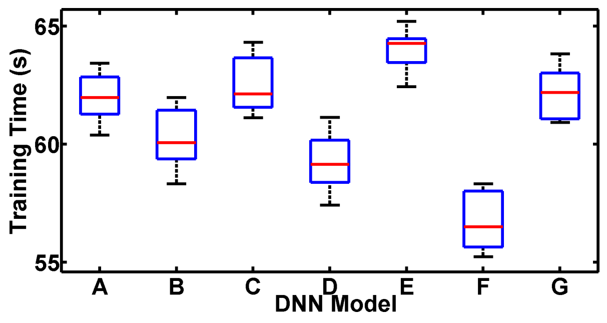


Fig. 19. Training Time of Different DNN models (offline training).

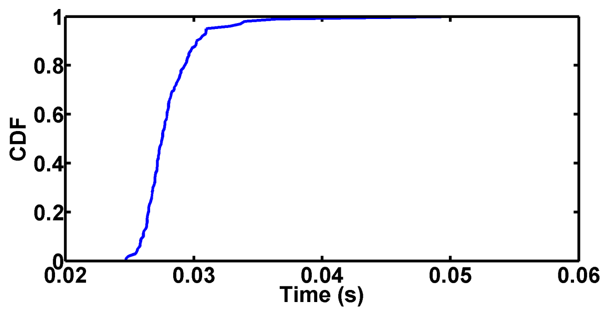


Fig. 20. CDF of execution time for online localization.

moderate, and the modeling time is a one-time expense. Once the model is established, no further training is required. Our solution's detection accuracy over time has been evaluated in Section V-B. Figure 20 shows the CDF of execution time of DNN models with the model selection algorithm in the online localization stage. On average, the execution time is 0.028 seconds. The execution time is far less than the LoRa packets sending interval (4.5 seconds). The short training time and execution time demonstrate the high runtime efficiency of our solution. Our model can provide real-time location predictions with low execution time.

VI. Related Work

GPS combined with other technologies are traditionally used to identify moving bus locations and provide arrival time prediction. For instance, Jisha et al. propose an IoT-based tracking system to estimate school bus arrival time by using a combination of GPS, GSM, and RFID technologies [16]. A Kalman filtering-based prediction algorithm has been used to estimate the arrival time of a school bus. Lee et al. design a navigation system that supports autonomous driving through the use of GPS/DR [17]. The system uses GPS/DR error estimation based on a lane detection algorithm to improve the localization performance. However, it is costly to install localization devices and retrofit existing buses to power them. The GPS module is power-hungry in the continuous navigation mode [18], which is unacceptable for resource-constrained IoT devices.

There has been increasing interest in developing GPS-free localization schemes that leverage the wireless signals transmitted by the IoT devices to locate them. Those practical localization techniques are based majorly on Time of Arrival (ToA), Time Difference of Arrival (TDoA), RSS, and Angle of Arrival (AoA) [19]. Multiple base stations or APs are required to locate the target object. For instance, Xiong et al. develop an AoA based localization method, which uses antennae arrays at the receiver side to estimate the device location [20]. Two monitors are required in a two-dimensional space, and three monitors are required in a three-dimensional space to track wireless clients. Lam et al. place six anchor nodes and filter out the nodes strongly affected by noise to identify the location of target device by using LoRa technology [21]. Thaljaoui et al. design a method for identifying a BLE device by using three BLE beacons. The method operates in two stages: the distance estimation stage and iRingLA localization stage [22]. Shirehjini et al. propose a RFID based localization system. The mobile device is equipped with an RFID reader, and it reads the information from multiple RFID tags on the carpet and then uses the sensor information to calculate the device's relative position [23]. Kyritsis et al. present a low-cost, threshold-based localization approach and design an algorithm that takes into account both the RSS of the bluetooth low energy beacons and the geometry of the rooms the beacons are placed in [24]. Artificial neural networks (ANNs) are also used to perform device localization. The location prediction is often formulated as a classification problem. Altini et al. design a deep learning system based on multiple neural networks to identify the target object location [25]. The DNN is trained using the Bluetooth RSS values in the offline training stage with labels. Once the model is trained, then it can be used in the online localization stage. Most of the previous work often requires the measurements performed at multiple base stations and therefore are not applicable to a single base station scenario. Recently, Blanco et al. propose a single base station ToA/AoA localization method [26]. By using the LTE sounding reference signal, the distance between the target object and base station is calculated through the ToA estimation. The AoA is measured by leveraging the multi-signal classification algorithm [26] and the system has been evaluated in the office scenario. Such a design can greatly benefit from the technologies such as Massive Multiple Input Multiple Output (MIMO) [27] systems to correctly predict the target object location. In a practical urban environment, the multiple base stations method used for localization can rarely be met [28]. Tsalolikhin et al. address the problem of mobile station localization using a single base station approach, which attempts to build a statistical model of urban propagation conditions [29]. The main idea of the proposed localization approach is to formulate the mobile station localization problem in the target classification framework and to use the statistical model of the urban propagation conditions to locate the target object without any hardware modifications. Porretta et al. propose a deterministic localization method with a single base station [30]. This method approximates the urban environment in the base station proximity by a sentinel function and achieves good localization performance. The method requires exact knowledge of the urban environment in the base station proximity. A new mobile station localization approach based on Ring of Scatterers (ROS) is proposed in response to the Non-Line-of-Sight (NLOS) environments [31]. By exploiting the geometrical relations among the mobile station, scatterers, and the single base station, Tian et al. present a Geometric Characteristics Based (GCB) localization algorithm with ROS model that provides conditional information for accurate location estimation of mobile station and scatterers [31]. Simulation results illustrate the superior performance of the proposed algorithm in typical NLOS environments. In contrast to previous studies, our paper investigates the feasibility of a single base station localization by using low-cost and low-power LoRa networks without introducing extra network traffic. Our approach leverages the LoRa link characteristics measured by a single LoRa base station and deep learning method to localize a campus shuttle, and it is therefore orthogonal and complementary.

VII. Conclusions

Today most of the existing bus localization systems rely on the GNSS, such as GPS. However, it is costly to install GPS receivers and retrofit existing buses to power them, which prevents them to be adopted by those bus operators with tight budgets. There also exist some localization methods that leverage the wireless signals transmitted by the buses to locate running objects. Such methods often require the RSS measurements to be performed at multiple base stations and therefore are not applicable to a small transportation service with a single base station. In this paper, we present our 14-month empirical study that investigates the feasibility of using the LoRa link characteristics measured at a single base station to localize campus shuttles. Based on our findings, we develop a novel solution that uses the LoRa link measurements to localize a shuttle when it approaches a stop. We implement our solution and test it on our campus shuttle monitoring system. Experimental results show that our solution provides the detection accuracy of no less than 92.07% and significantly outperforms all baselines.

Acknowledgment

This work was supported in part by the National Science Foundation under grant CNS-1657275 and CNS-2150010.

References

[1] NYC Bus Checker. [Online]. Available: http://www.buschecker.com/app/NYC/

[2] M. Anu, D. Sarikha, G. Keerthy, and J. Jabez, "An RFID Based System for Bus Location Tracking and Display," in *ICIICT*, 2015.

[3] S. Gunady and S. L. Keoh, "A Non-GPS based Location Tracking of Public Buses using Bluetooth Proximity Beacons," in *WF-IoT*, 2019.

[4] O. Oguejiofor, "Outdoor Localization System using RSSI Measurement of Wireless Sensor Network," *International journal of Innovative technology and exploring Engineering (IJITEE)*, vol. Volume 2, pp. Pages 1–6, 01 2013.

[5] D. Plets, N. Podevijn, J. Trogh, L. Martens, and W. Joseph, "Experimental Performance Evaluation of Outdoor TDoA and RSS Positioning in a Public LoRa Network," in *IPIN*, 2018, pp. 1–8.

[6] LoRa. [Online]. Available: https://lora-alliance.org/

[7] D. Mu, Y. Chen, J. Shi, and M. Sha, "Runtime Control of LoRa Spreading Factor for Campus Shuttle Monitoring," in *ICNP*, 2020.

[8] Semtech. [Online]. Available: https://www.semtech.com/

[9] P. Mayer, M. Magno, T. Brunner, and L. Benini, "LoRa vs. LoRa: In-Field Evaluation and Comparison For Long-Lifetime Sensor Nodes," in *IWASI*, 2019.

[10] IMST iC980A. [Online]. Available: https://wireless-solutions.de/products/lora-solutions-by-imst/radio-modules/im980a-l/

[11] MICROCHIP RN2903. [Online]. Available: https://www.microchip.com/wwwproducts/en/RN2903

[12] R. W. Hamming, "Error Detecting and Error Correcting Codes," *Bell System Technical Journal*, vol. 2, no. 29, 1950.

[13] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts, "Understanding Variable Importances in Forests of Randomized Trees," in *NIPS*, 2013.

[14] M.-C. Popescu, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, "Multilayer Perceptron and Neural Networks," *WSEAS Transactions on Circuits and Systems*, vol. 8, no. 7, p. 579–588, 2009.

[15] Chih-Wei Hsu and Chih-Jen Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, March 2002.

[16] R. C. Jisha, A. Jyothindranath, and L. S. Kumary, "Iot Based School Bus Tracking and Arrival Time Prediction," in *ICACCI*, 2017.

[17] Lee, Byung-Hyun and Song, Jong-Hwa and Im, Jun-Hyuck and Im, Sung-Hyuck and Heo, Moon-Beom and Jee, Gyu-In, "GPS/DR Error Estimation for Autonomous Vehicle Localization," *Sensors*, vol. 15, no. 8, pp. 20 779–20 798, 2015.

[18] A. Carroll and G. Heiser, "An Analysis of Power Consumption in a Smartphone," in *USENIX ATC*, 2010, p. 21.

[19] A. Yassin, Y. Nasser, M. Awad, A. Al-Dubai, R. Liu, C. Yuen, R. Raulefs, and E. Aboutanios, "Recent Advances in Indoor Localization: A Survey on Theoretical Approaches and Applications," *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 1327–1346, 2017.

[20] J. Xiong and K. Jamieson, "ArrayTrack: A Fine-Grained Indoor Location System," in *USENIX NSDI*, 2013.

[21] K. Lam, C. Cheung, and W. Lee, "LoRa-based localization systems for noisy outdoor environment," in *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2017, pp. 278–284.

[22] A. Thaljaoui, T. Val, N. Nasri, and D. Brulin, "BLE localization using RSSI measurements and iRingLA," in *ICIT*, 2015.

[23] A. A. N. Shirehjini, A. Yassine, and S. Shirmohammadi, "An RFID-Based Position and Orientation Measurement System for Mobile Objects in Intelligent Environments," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 6, pp. 1664–1675, 2012.

[24] A. I. Kyritsis, P. Kostopoulos, M. Deriaz, and D. Konstantas, "A BLE-based probabilistic room-level localization method," in *2016 International Conference on Localization and GNSS (ICL-GNSS)*, 2016, pp. 1–6.

[25] M. Altini, D. Brunelli, E. Farella, and L. Benini, "Bluetooth indoor localization with multiple neural networks," in *IEEE 5th International Symposium on Wireless Pervasive Computing 2010*, 2010, pp. 295–300.

[26] A. Blanco, N. Ludant, P. J. Mateo, Z. Shi, Y. Wang, and J. Widmer, "Performance Evaluation of Single Base Station ToA-AoA Localization in an LTE Testbed," in *PIMRC*, 2019.

[27] L. Lu, G. Y. Li, A. L. Swindlehurst, A. Ashikhmin, and R. Zhang, "An Overview of Massive MIMO: Benefits and Challenges," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, pp. 742–758, 2014.

[28] Y.-T. Chan, W.-Y. Tsui, H.-C. So, and P. chung Ching, "Time-of-arrival based localization under NLOS conditions," *IEEE Transactions on Vehicular Technology*, vol. 55, no. 1, pp. 17–24, 2006.

[29] E. Tsalolikhin, I. Bilik, and N. Blaunstein, "A Single-Base-Station Localization Approach Using a Statistical Model of the NLOS Propagation Conditions in Urban Terrain," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 3, pp. 1124–1137, 2011.

[30] M. Porretta, P. Nepa, G. Manara, F. Giannetti, M. Dohler, B. Allen, and A. Aghvami, "A novel single base station location technique for microcellular wireless networks: description and validation by a deterministic propagation model," *IEEE Transactions on Vehicular Technology*, vol. 53, no. 5, pp. 1502–1514, 2004.

[31] Z. Tian, Y. Shu, Y. Li, M. Zhou, and Z. Li, "Ring of Scatterers Based Localization Using Single Base Station," in *Wireless Internet*, C. Li and S. Mao, Eds. Cham: Springer International Publishing, 2018, pp. 232–238.