Runtime Control of LoRa Spreading Factor for Campus Shuttle Monitoring

Di Mu, Yitian Chen, Junyang Shi, Mo Sha
Department of Computer Science
State University of New York at Binghamton
{dmu1, cyitian1, jshi28, msha}@binghamton.edu

***Abstract*—Traditionally, satellite and cellular technologies have been used in establishing the long-distance links that collect real-time data from running vehicles to the base station. However, the systems that implement those technologies are often too costly for use in small communities, such as monitoring shuttles that circle a university campus. Recently, LoRa has been used as a low-cost alternative that provides the capability of long-range data collection for low data rate applications. In this paper, we present a low-cost LoRa-based wireless network that collects real-time data from six shuttles circling our university campus and has operated in the real world for more than a year. The selection of the LoRa Spreading Factor (SF) poses a significant challenge because of its effects on two conflicting network performance metrics. A larger SF provides higher network reliability at the cost of lower throughput. To address this challenge, we develop a runtime SF control solution that employs the K-Nearest Neighbors (KNN) algorithm to adapt the SF configuration based on the current link condition. Experimental results show that our approach significantly increases the data collection throughput while meeting the application reliability requirement compared to the state of the art.**

***Index Terms*—LoRa, Spreading Factor Control, K-Nearest Neighbors, Quality of Service, Campus Shuttle Monitoring**

I. Introduction

Satellite and cellular technologies are traditionally used to collect real-time data from running vehicles to the base station through their long-distance links. For instance, LTE-based communication systems have been integrated into the urban transit systems [1], [2], while satellite links have been used to support communication among emergency vehicles [3]. However, such systems are often costly because they use expensive devices and licensed frequency bands, which prevents them from being used in many applications. As an emerging Low-Power Wide-Area Network (LPWAN) technology, LoRa is a low-cost alternative that can support long-range data collection for low data rate applications [4], [5]. For a small service area, such as a university campus, LoRa offers a cost-effective communication solution because one or a few base stations are enough to cover the area and battery-powered LoRa modules can inexpensively retrofit existing devices.

In this paper, we introduce the *ShuttleNet*, a LoRa-based wireless networking solution that collects real-time data from shuttles that circle a university campus using a fixed route. Multiple types of data, such as vehicle speed, the vehicle's operating condition, and the number of passengers, are collected to enhance the safety and efficiency of shuttle service and improve rider experience. For instance, the vehicle speed information is used to estimate the expected time of arrival (ETA) at each shuttle stop. The number of passengers is used to monitor the transit demand, which allows more shuttles to be dispatched when needed. Warnings or alarms are generated if the vehicle's operating condition degrades. The data needed to be collected falls into one of two categories: time-critical and non-time-critical. The time-critical data, such as the current vehicle speed and the number of passengers, needs to be collected in real time with high reliability because the data may become useless if it fails to be delivered in time. The non-time-critical data, such as the data reflecting the vehicle's operating condition (e.g., accelerating and braking performance), can be delayed because the operating condition of a vehicle does not change very frequently. Similar to the tradeoff between network reliability and throughput in cellular networks [6], the tradeoff also exists in LoRa networks, which results in a significant challenge on the selection of the LoRa Spreading Factor (SF). A larger SF provides higher reliability at the cost of lower throughput and vice versa. The fluctuations of low-power LoRa link quality resulting from the mobility of vehicles significantly amplify the challenge. Therefore, there exists a critical need for a new solution that makes good tradeoffs between network reliability and throughput for LoRa-based mobile wireless networks.

In this paper, we present a low-cost LoRa-based networking solution that employs a novel runtime SF control solution to maximize the data collection throughput from running vehicles while meeting the reliability requirement specified by the application. Specifically, this paper makes the following contributions:

- This paper presents a low-cost LoRa-based wireless networking solution that collects real-time data from six shuttles circling our university campus;
- To our knowledge, this is the first paper to investigate the SF selection for LoRa devices installed on running vehicles, distinguished from previous work designed for stationary devices. The study has been performed in the real world for more than a year;
- This paper provides a practical runtime LoRa SF control solution that employs the K-Nearest Neighbors (KNN) algorithm and meets network performance requirements

978-1-7281-6992-7/20/$31.00 ©2020 IEEE

with small computation overhead;
- Our experimental results show that our runtime LoRa SF control solution significantly outperforms the state-of-the-art methods.

The remainder of the paper is organized as below. Section II introduces the background of LoRa. Section III presents our design of ShuttleNet. Section IV describes our empirical study of LoRa SF configurations. Section V presents the design of our runtime SF control solution, and Section VI compares its performance against three baselines. Section VII reviews the related work. Section VIII concludes the paper.

II. Background

Recently, LoRa has emerged as a popular LPWAN technology that provides long-range communication. LoRa employs the Chirp Spread Spectrum (CSS) modulation, where a LoRa signal, namely a chirp, increases or decreases its operating frequency linearly through time and circularly sweeps through its predefined frequency band. The time duration of transmitting a chirp depends on the selections of the LoRa physical-layer parameters: SF and Bandwidth (BW). The reliability of a LoRa link is measured by the Packet Delivery Ratio (PDR), which depends on the Signal-to-Noise Ratio (SNR) and the Received Signal Strength (RSS) at the receiver. A larger SF allows the receiver to receive a packet with a lower RSS by exponentially expanding the duration of each chirp, which results in a lower data rate. A chirp that uses the SF value of x can represent x bits of data and takes the time duration of $2^x/BW$ to be transmitted [7]. With a fixed BW, the data rate of a chirp is proportional to $(x \cdot 2^{-x})$, which decreases nearly exponentially with x. There are six SF configurations available in the sub-GHz ISM bands (from $SF7$ to $SF12$). The data rates under $SF8$ to $SF12$ are 57.1%, 32.1%, 17.9%, 9.8%, and 5.4% of the one under $SF7$, respectively. The selection of SF decides the data rate and communication range of a LoRa link. The LoRa signal that carries a packet consists of a series of chirps, where the preamble chirps are followed by the data chirps. The preamble chirps are used by the receiver to identify the start of a LoRa packet. The data chirps contain up to 255 bytes including the packet header and payload sections. Both sections end with a Cyclic Redundancy Check (CRC) code to verify the integrity of the received data. The payload section contains the data bits with some inserted Hamming Code redundant bits for error correction. The Coding Rate (CR) of Hamming Code (e.g., "4/5") is defined as a fraction in which the numerator is the number of data bits and the denominator is the total number of data bits and redundant bit(s). The CR setting of a packet is stored in its header.

LoRaWAN is a Media Access Control (MAC) layer protocol designed for LoRa and defines three classes: Class A (by default), Class B, and Class C. The Class A communication frame is initiated by an end device and consists of an uplink window (time slot) followed by two downlink windows. To send data, the end device initiates an uplink window, selects a random channel, and transmits in pure ALOHA mode [8]. The first downlink window operates on the same channel as the last uplink window and the second downlink window operates on a predefined channel [9]. Classes B and C allow downlink data to be received in a timely fashion by increasing the time that end devices listen to the channel. Class B uses periodic beacons broadcasted by the base station to open extra downlink windows at end devices while Class C keeps end devices listening at all times. In many cases, the communication frames defined in LoRaWAN cannot provide optimal performance, such as when an application only requires end devices to upload data, leaving the downlink windows unused. Moreover, theoretically speaking, the maximum throughput that pure ALOHA can provide is only 18.4% of the channel capacity [10], which limits the throughput performance of LoRaWAN. To overcome the limitations, we design new time frame and channel assignment methods for ShuttleNet (see Section III-C). LoRaWAN specifies the Adaptive Data Rate (ADR) algorithm that selects SF based on the link quality measured on an end device [11]. Specifically, ADR first estimates the link quality using the maximum SNR values measured in the last 20 samples and then selects the SF based on the required SNR level for each SF [12]. Similar to the Class A frame, the ADR process is initiated by a request from an end device followed by a response from the base station with a new SF. If the response is not received by the end device before it times out, the end device increases the SF because the link is disconnected under the current SF. When an ADR request is received, the base station reduces the current SF to the most appropriate level if the estimated SNR is greater than the required SNR plus a safety margin. ADR is designed for stationary devices and does not work well on mobile devices (see Section IV), which motivates us to design a new runtime SF control solution for ShuttleNet.

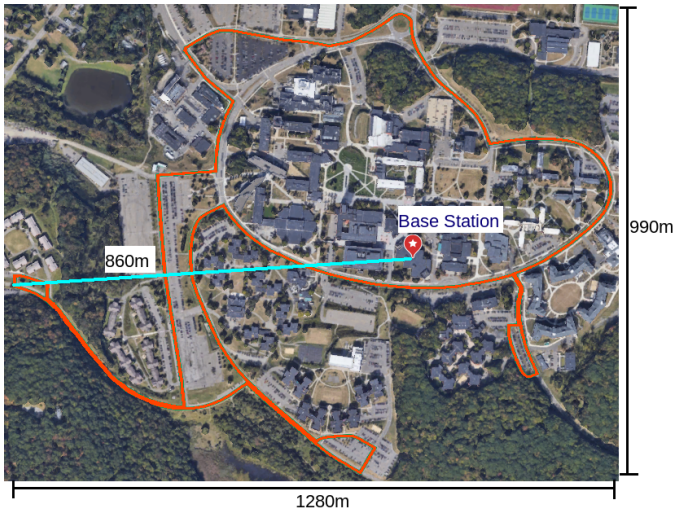
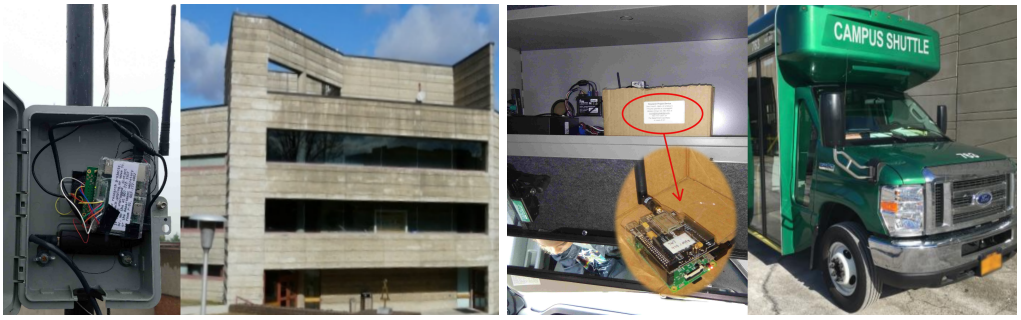


Fig. 1: Campus shuttle route.

III. ShuttleNet

In this section, we first present our hardware deployment and software architecture of ShuttleNet and then introduce our time frame and channel assignment designs.



(a) LoRa base station installed on a roof. (b) LoRa end device installed on a shuttle.

Fig. 2: Hardware deployment of ShuttleNet at the State University of New York (SUNY) at Binghamton.

TABLE I: Price List of Hardware Components

Device Name	Price (USD)	End Device	Base Station
Raspberry Pi 3	35	√	√
RN2903 Module	13	√	
RPi Connection Bridge	8	√	
iC980A Module	130		√
Power Adapter	5	√	√
Total (USD)	536	61 × 6	170

A. Hardware Deployment

ShuttleNet is designed to collect data from six shuttles that circle our university campus (a 1280m × 990m area) with a fixed route. Figure 1 shows the route. The distance between a shuttle and our LoRa base station can be up to 860 $meters$. Figure 2 shows our hardware deployment. The LoRa base station is placed in a weatherproof box on the roof of a three-floor building (Figure 2a) and a LoRa end device is installed in the glove compartment above the driver seat on each of six shuttles (Figure 2b). The LoRa base station and end devices are built by integrating commercial off-the-shelf (COTS) devices. The LoRa base station is an embedded computer (i.e., Raspberry Pi 3 Model B) integrated with an iC980A module provided by IMST, while the LoRa end device is a Raspberry Pi computer integrated with an RN2903 module operating in the 900/915 MHz band [13]. The iC980A module is the enhanced version of iC880A [14] operating in the 900/915 MHz band. We use iC980A to build our LoRa base station because it is capable of receiving packets from multiple end devices that use different SF configurations and up to eight channels in parallel. To maximize throughput, we configure different LoRa end devices to use distinct channels. The RN2903 module can only operate on a single channel in either transmitter or receiver mode at each time. Our goal is to provide a low-cost networking solution; thus we use the RN2903 module to build the LoRa end device. Table I summarizes the costs of our devices. Please note that LoRa operates in the free, unlicensed band. The total hardware cost of ShuttleNet is $536.

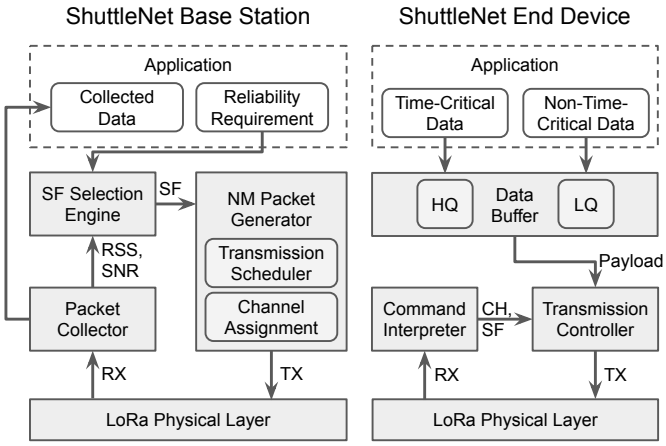


Fig. 3: Software architecture of ShuttleNet.

B. Software Architecture

Figure 3 plots the software architecture of ShuttleNet. On the LoRa base station, the **Packet Collector** forwards the received packets from the LoRa physical layer to the application and collects the link characteristics including RSS and SNR for the **SF Selection Engine**. The SF Selection Engine employs our runtime SF control solution that selects the best-suited SF configuration for each end device based on the given link characteristics and the reliability requirement specified by the application (see Section V). The **NM Packet Generator** broadcasts the Network Management (NM) packets, which carry the selected SF configuration, the assigned channel, and the transmission schedule for each end device (see Section III-C). On each LoRa end device, the **Command Interpreter** extracts and interprets the NM commands from the received NM packets. The **Transmission Controller** transmits the data generated from the vehicle on the assigned channel using the selected SF configuration. The **Data Buffer** maintains two data queues for the data collected from the vehicle: a high-priority queue (HQ) for time-critical data and a low-priority queue (LQ) for non-time-critical data. The transmission controller only transmits the data stored in the LQ when the HQ is empty.

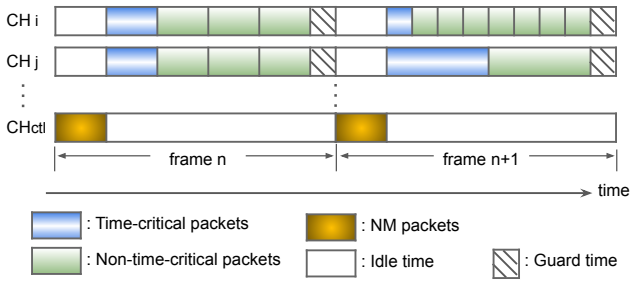
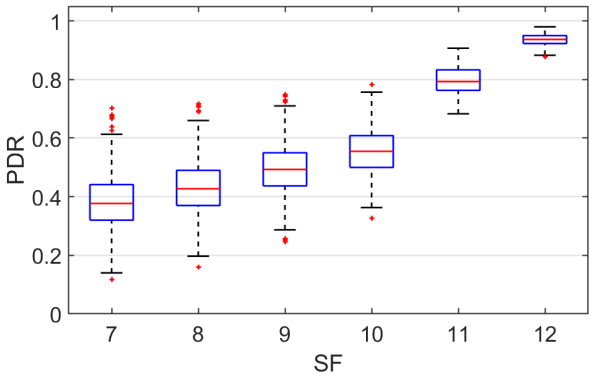


Fig. 4: An example timeline of transmissions on different channels within two consecutive time frames in ShuttleNet.

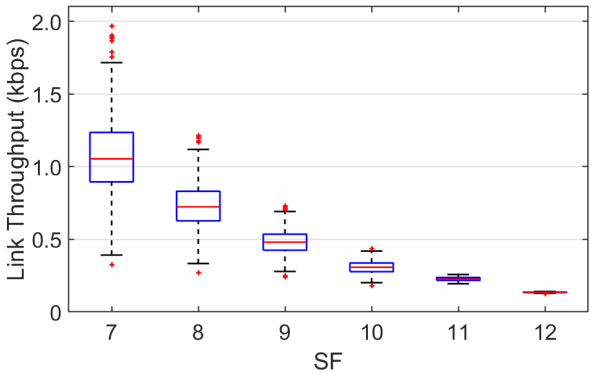
C. Time Frame and Channel Assignment

In ShuttleNet, the LoRa base station and end devices are time synchronized and share the notion of a time frame that repeats over time. At the beginning of each frame, the LoRa base station broadcasts a NM packet that synchronizes the clocks of all end devices. Each NM packet carries a unique frame ID and the network management information for this frame (e.g., the SF and channel assigned to each end device). ShuttleNet uses a fixed channel CH_{ctl} for downlinks (from the base station to shuttles) and assigns channels for uplinks (from shuttles to the base station) based on the number of shuttles in the network N_s. Assuming the LoRa base station can receive packets from N channels (CH_1 to CH_N) in parallel, the LoRa base station assigns a unique channel CH_i ($1 \leq i \leq N$) to each end device if $N_s \leq N$. Otherwise, the LoRa end devices share the N channels in a TDMA fashion. The NM packets carry the commands indicating which devices should transmit in each time frame. The LoRa base station always uses the physical layer parameters ($SF = 12$, $CR = 4/8$, and CRC enabled) to transmit the NM packets because the deliveries of NM packets are critical for maintaining the time frame.

Before a LoRa end device starts to transmit, it first listens to the downlink channel CH_{ctl} and waits for the NM packet. After receiving the NM packet, the LoRa end device searches for its ID to decide whether it can transmit in this frame. If the LoRa end device is allowed to transmit, it calculates the number of packets that can be transmitted in the current frame using the assigned SF configuration. A small guard time is reserved at the end of each frame to compensate for local clock drifting. The transmission starts with the time-critical packets followed by the non-time-critical packets using the assigned channel. After the LoRa end device finishes the transmissions in the current frame, it listens to CH_{ctl} and waits for the NW packet again. Figure 4 shows an example timeline of transmissions on different channels within two consecutive time frames. In the example, two LoRa end devices are assigned to use CH_i and CH_j, respectively. In frame n, both LoRa end devices transmit using the same SF configuration. In frame $n+1$, the LoRa end device using CH_i is assigned with a smaller SF while the other using CH_j is assigned with a larger SF. As discussed in Section II, a smaller SF provides higher throughput and allows more packets to be transmitted in a frame. In our implementation, we set the payload size of an uplink packet to 36 *bytes*, that of a NM packet to 12 *bytes*, the frame length to 5 *seconds*, and the guard time to 0.25 *seconds*.



(a) Box plot of link reliability. Central mark in box indicates median; bottom and top of box represent the 25th percentile (q_1) and 75th percentile (q_2); crosses indicate outliers ($x > q_2 + 1.5 * (q_2 - q_1)$ or $x < q1 - 1.5 * (q_2 - q_1)$); whiskers indicate range excluding outliers.



(b) Box plot of link throughput.

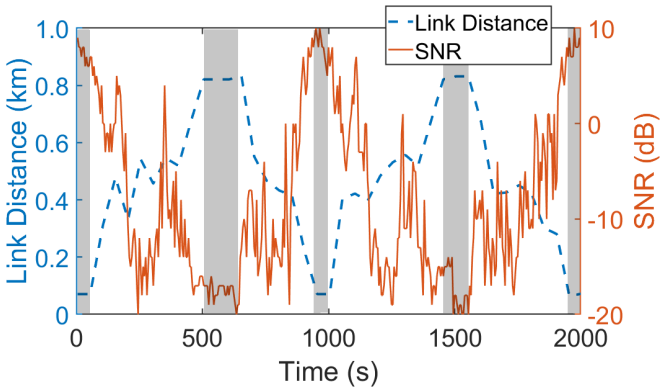
Fig. 5: Link performance when the LoRa transmitters use different SF configurations.

IV. An Empirical Study on SF Configuration

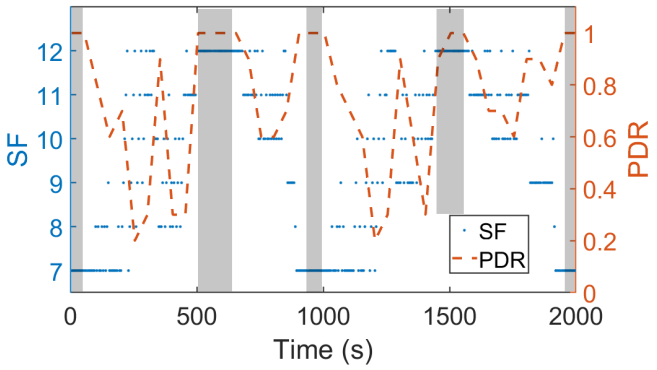
To investigate the impact of SF configuration on network performance, we have performed a 14-month empirical study from March 2018 to May 2019. We configure the LoRa end devices installed on the shuttles to use all six SFs ($SF7$ to $SF12$) for packet transmissions in a round-robin fashion. In other words, a LoRa end device switches to the next SF after transmitting a packet using the current SF. The LoRa base station has collected 3.18 million uplink packets generated by six shuttles during their real-world operations (3931 *loops* in total). We have identified all lost packets and their corresponding SFs by checking the sequence IDs carried by all received packets and discarded all corrupted packets with CRC errors.

Leveraging our data trace, we first relate SF configuration to network performance. Figure 5 plots the box plots of the link reliability and throughput when the LoRa transmitters use different SF configurations. The link reliability in terms of PDR and the link throughput is computed every 25 $minutes$. Figure 5a and 5b clearly present the tradeoff between link reliability and throughput when the LoRa transmitters use different SF configurations. The PDR increases and the throughput decreases when the LoRa transmitters use a larger SF. Specifically, the median PDRs are 0.38, 0.43, 0.49, 0.56, 0.80, and 0.94, while the median throughput values are 1.05 $kbps$, 0.72 $kbps$, 0.48 $kbps$, 0.31 $kbps$, 0.23 $kbps$, and 0.14 $kbps$ when the LoRa transmitters use $SF7$, $SF8$, $SF9$, $SF10$, $SF11$, and $SF12$, respectively. More importantly, the link reliability increases more significantly when the LoRa transmitters use large SFs ($SF11$ and $SF12$), while the link throughput decreases dramatically at small SFs. This indicates that increasing SF when the current SF is large may significantly enhance the link reliability at the cost of slightly reducing the link throughput, while slightly decreasing SF when the current SF is small may significantly improve the throughput without introducing too much damage on the link reliability. Those observations motivate our designs in Section V.

As discussed in Section II, ADR specified in LoRaWAN is designed to select SF based on the measured link quality. Unfortunately, our empirical study shows that ADR is ineffective when the LoRa end devices are in motion. For instance, Figure 6 shows the PDR changes when a shuttle circles around the campus twice. To understand the ineffectiveness of ADR, we install a GPS device onto that shuttle. Figure 6a plots the distance and the SNR of the link between the shuttle and the LoRa base station and Figure 6b plots the SF configurations selected by ADR and the resulting PDR over time. The shuttle made five stops around $0s$, $500s$, $950s$, $1450s$, and $1950s$ during that $2000s$ measurement. From the link distances plotted in Figure 6a, we can observe that the first, third, and fifth stops are close to the LoRa base station, while the second and fourth stops are far away. As Figure 6b shows, ADR provides very good link reliability (almost 1) by selecting appropriate SFs when the shuttle was not in motion. However, the link reliability decreases significantly when the shuttle begins to move. For example, the link reliability drops to 0.21 at $255s$ and 0.24 at $1195s$. From Figure 6b, we can observe that the SF configurations selected by ADR fluctuate when the shuttle moves, resulting in unstable link reliability. There are two main reasons that cause the ADR's failure to maintain good link reliability when the LoRa end device moves quickly. First, ADR uses the maximum SNR of the last 20 SNR samples to estimate the link quality. Although it works well on stationary devices, it fails on those devices in motion because the SNR changes very fast, as Figure 6a shows. This motivates us to consider the most recent link characteristics when selecting SF (see Section V). Second, ADR uses a set of theoretical SNR thresholds for SF selection rather than using the actual link reliability measured by the device. A recent study shows that the best-suited SNR thresholds for SF selection should be selected based on the specific physical-layer configurations of the LoRa device (e.g., packet payload length and coding rate) [15]. Moreover, we observe that using SNR measurements alone is insufficient to accurately predict the packet receptions because the receiver sensitivity of a LoRa device also depends on the RSS [16]. This motivates us to use both SNR and RSS measurements when selecting SF (see Section V).



(a) Link distance and SNR.



(b) SF selected by ADR and resulting PDR.

Fig. 6: The link reliability changes under ADR ($window = 20, margin = 10dB$) when a shuttle circles the campus twice. The grey areas indicate the time when the shuttle stopped.

V. Runtime SF Control

In this section, we present the design of our runtime SF control solution that runs on the LoRa base station and selects the SF configuration for each LoRa end device to meet network performance requirements. The design goal of our solution is to maximize the data collection throughput while meeting the reliability requirement specified by the application.

A. Overview of the Solution

Our design goal is to maximize the data collection throughput while maintaining the application-specific link reliability by employing best-suited SF continuously. The SF configuration is selected based on the given link reliability requirement and link characteristics based on a machine learning model.

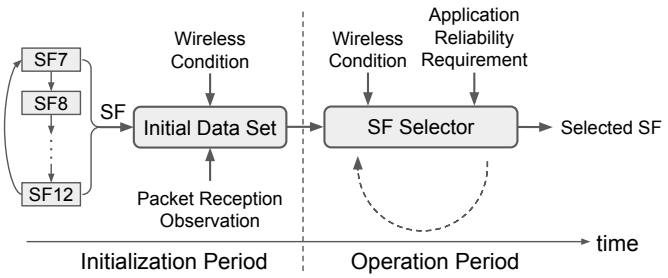


Fig. 7: Initialization Period and Operation Period.

After a LoRa end device begins to operate, our runtime SF control solution guides it through two periods: **Initialization Period** and **Operation Period**, as shown in Figure 7. In the Initialization Period, our runtime SF control solution controls each LoRa end device to transmit packets using all SF configurations in a round-robin fashion. It also controls the LoRa base station to measure the link characteristics when receiving every packet and observes the packet receptions under each SF configuration. This allows the LoRa base station to create the **Initial Data Set** S, in which each data element includes three pieces of information: the link characteristics (RSS, SNR, and the averaged RSS over the last 10 time frames), the SF configuration, and the packet reception result (success or failure). We empirically choose the time length of the Initialization Period that provides the best performance (see Section VI-A). After collecting enough data for S, our runtime SF control solution begins to periodically predict the best-suited SF configuration in the Operation Period based on a non-linear mapping between the link characteristics ($\mathbf{x}$) and the best-suited SF configuration (sf), i.e., $f : \mathbf{x} \rightarrow sf$, which is learned by our machine learning model using S. We will next present the design of our SF selector that runs in the Operation Period.

B. KNN-based SF Selector

The primary task of the SF selection algorithm is to identify the best-suited SF in each time frame as described in Section V-A. We define a classification problem that periodically predicts the packet reception result (success or failure) when using each SF configuration under the given link characteristics. In each frame, our solution selects the smallest SF predicted with a successful packet reception to deliver the maximum throughput. While considering runtime computational efficiency, we employ the KNN algorithm [17] for classification. We adjust the parameters in the KNN algorithm at runtime to meet the reliability requirement (see Section V-C). Algorithm 1 presents our KNN-based SF selection algorithm. The input of Algorithm 1 includes the link characteristics ($\mathbf{x}$) and the output is the selected SF configuration for the next frame (sf). $\mathbf{x}$ is an array of three integers (x_1, x_2, and x_3) representing the link characteristics (RSS, SNR, and averaged RSS). The Initial Data Set S is used to relate the link characteristics to the packet reception results under each SF

Algorithm 1: KNN-based SF Selection Algorithm

Input : $\mathbf{x}$
Output : sf
Data Set : S
Parameters: k, $\mathbf{v_t}$

```
1  d = 0
2  N = [ ]
3  while N.length < k do
4      X' = list_link_conditions (x, d)
5      for x' in X' do
6          N.add (S[x'_1][x'_2][x'_3] )
7      end
8      d += 1
9  end
10 for sf in [7...11] do
11     v_p = 0
12     for n in N do
13         v_p += n[sf]
14     end
15     if v_p / N.length > v_t[sf] then
16         return sf
17     end
18 end
19 sf = 12
```

configuration. In our implementation, S is implemented as a 3-D array that uses three array indices for the link characteristics and each array element is implemented as a linked list, which stores the observed packet reception results (1 for success and 0 for failure) using all SF configurations. The parameter k is a constant, which denotes the number of samples to be searched for in S. Based on the size and density of S, we empirically set k to 20 in our implementation. The parameter $\mathbf{v_t}$ is an array, which stores the voting thresholds for each SF. To simplify our presentation, we assume that all arrays in Algorithm 1 are free of bounds and can be accessed using any integer.

Lines 1 and 2 of Algorithm 1 initialize the distance integer d and the neighbor sample array $\mathbf{N}$. The loop from Line 3 to Line 9 searches for and stores the "nearest neighbors" in the array $\mathbf{N}$, where the nearest neighbors are the samples in S within the shortest distances to the input link characteristics ($\mathbf{x}$). We use the Euclidean distance to measure the distance between $\mathbf{x}$ and $\mathbf{x}'$:

$$d(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_i (x_i - x'_i)^2} \quad (1)$$

where x_i and x'_i ($i \in [1, 2, 3]$) are the elements of $\mathbf{x}$ and $\mathbf{x}'$. Line 4 lists all integer-valued link characteristics with the rounded distance (d) to $\mathbf{x}$ and stores them in the set $\mathbf{X}'$. The loop from Line 5 to Line 7 uses the link characteristics in each $\mathbf{x}'$ from $\mathbf{X}'$ as the indices of S and adds all samples from the indexed S elements into $\mathbf{N}$. The distance d is incremented (Line 8) until k samples have been added to $\mathbf{N}$ (Line 3). All samples with the distance d are used when a tie of distance

exists. KNN uses the votes of the neighbors for classification. The loop from Line 10 to Line 18 counts the votes for each SF from $SF7$ to $SF11$ and returns the selected SF based on the voting result. For each SF configuration, v_p counts the positive votes from each neighbor **n** in **N**, where a neighbor with a successful reception under the SF ($n[sf] = 1$) contributes a positive vote (Line 13). The resulting positive voting rate (v_p/**N**.*length*) is compared with the voting threshold of the SF ($v_t[sf]$). If the positive voting rate is greater than the threshold, the classification result is positive for the SF (Line 15), predicting a successful packet reception. The algorithm returns with the smallest possible SF when a positive classification result is found (Line 16). The array of voting thresholds $\mathbf{v_t}$ is adjusted at runtime in response to the reliability requirement (see Section V-C). If there is no positive classification result from $SF7$ to $SF11$ or the link characteristics are not available at the moment, $SF12$ is selected (Line 19) to maintain the connection.

C. KNN Voting Threshold Adjustment

The array of voting thresholds ($\mathbf{v_t}$) is a set of parameters that are critical to meet the reliability requirement specified by the application. Each element in $\mathbf{v_t}$ is a voting threshold for an individual SF because each SF may require a different threshold to meet the reliability requirement. A higher threshold requires more positive votes to predict a successful packet reception, which reduces the chance of false positive predictions.

Algorithm 2: Voting Threshold Adjustment

Inputs : R_r, R_c, $\mathbf{R_{sf}}$, $\mathbf{v_t}$
Output : $\mathbf{v_t}$
Parameters: α, β, γ

```
1  if R_c < R_r then
2      for sf in [7...11] do
3          if R_sf[sf] < R_r then
4              v_t[sf] += α
5          end
6      end
7  end
8  if R_c > R_r + γ then
9      for sf in [7...11] do
10         if R_sf[sf] > R_r + γ then
11             v_t[sf] -= β
12         end
13     end
14 end
```

When the system begins to operate, the voting thresholds for $SF7$ to $SF11$ are initialized to 0.5. Algorithm 2 is designed to adjust the voting thresholds and is triggered when a new PDR measurement is generated or a new reliability requirement is input by the application. Given the application reliability requirement (R_r), the current link PDR (R_c), and an array of PDRs when using each SF configuration ($\mathbf{R_{sf}}$), there are two options for adjustments: (1) increasing the voting threshold for sf by α if both R_c and $R_{sf}[sf]$ are less than R_r (Line 1 to 7) and (2) decreasing the voting threshold for sf by β if both R_c and $R_{sf}[sf]$ are higher than $R_r + \gamma$ (Line 8 to 14). Algorithm 2 keeps the previous voting thresholds if no change is needed. In our implementation, we set α to 0.1, β to 0.05 which allows a fast response when the actual reliability fails to meet the requirement and a slow adjustment when the actual reliability is high. While increasing the voting thresholds helps increase the link reliability, decreasing the voting thresholds is designed to increase the link throughput because the selector has a greater chance to select smaller SFs with higher data rates. The value of γ is set to 0.05 in our implementation which aims to maintain the link reliability between R_r and $R_r + 0.05$.

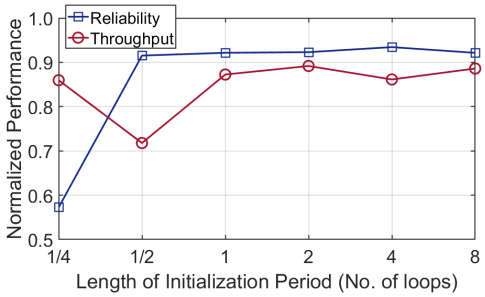


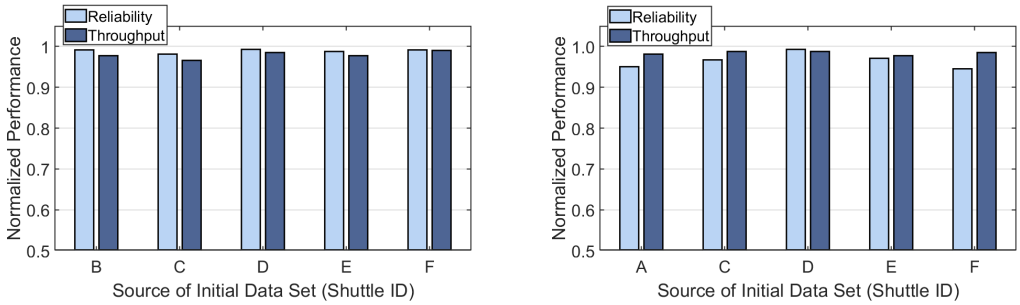
Fig. 8: Performance when using the Initial Data Set with different sizes. Performance is normalized to the one using the optimal selections.

VI. Evaluation

To validate the efficiency of our runtime SF control solution in maximizing the link throughout while meeting the application-specified reliability requirement, we performed a series of experiments. We first empirically identify the best-suited length of the Initialization Period. We then examine whether our solution can consistently meet the reliability requirement specified by the application. We evaluate our solution's effect on increasing the data collection throughput, and compare its performance against three baselines. Finally, we evaluate the runtime efficiency of our solution by measuring its execution time on a Raspberry Pi computer. To ensure a fair comparison between our solution and baselines, we apply all solutions on the same data trace collected from our 14-month empirical study with six shuttles (see Section IV). We also identify the optimal SF selections by analyzing the entire data trace. Please note that the optimal solution cannot be implemented at runtime and is only for comparison purposes.

A. Impact of Initialization Period Length

As discussed in Section V-A, our runtime SF control solution first controls the LoRa end devices installed on the shuttles to use all six SFs ($SF7$ to $SF12$) for packet transmissions in a round-robin fashion to generate the Initial Data Set S. The network may experience poor reliability and low throughput in the Initialization Period. Thus, it is beneficial to keep the



(a) Using the initial data collected from different shuttles for Shuttle A.

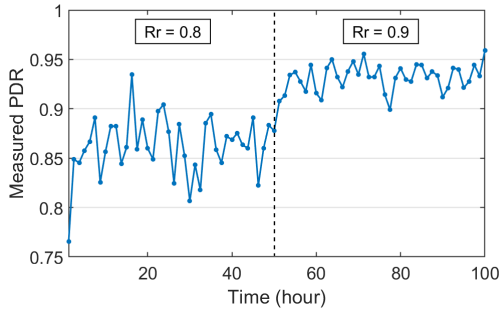
(b) Using the initial data collected from different shuttles for Shuttle B.

Fig. 9: Performance when using the Initial Data Set collected from one shuttle on another. Performance is normalized to the one when using the initial data collected from the same shuttle. The Initial Data Set includes one loop of data. $R_r = 0.8$.

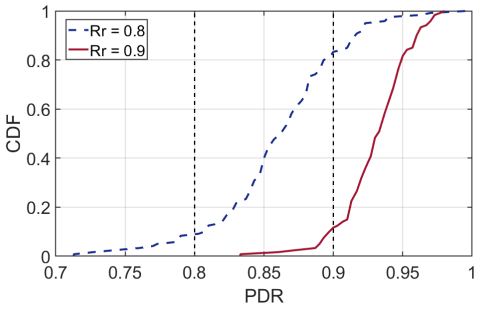
Initialization Period as short as possible. We run experiments to study the impact of the length of the Initialization Period on network performance. Figure 8 shows the network reliability and throughput when our runtime SF control solution uses S with different sizes. All results are normalized to those using the optimal selections. As Figure 8 shows, the normalized reliability is very low (0.57) when S has the collected data when a shuttle traveled the first quarter of its route. After the shuttle traveled the first half of its route, the normalized reliability increases to 0.92, but the normalized throughput is only 0.72. The normalized throughput increases to 0.87 and the normalized reliability is 0.92 when the shuttle traveled the whole route once. More data in S does not provide much help on improving the throughput and reliability. The results show that collecting one loop of data to create the Initial Data Set is enough for our KNN-based SF selector to provide good SF selections at runtime.

B. Sharing the Initial Data among Shuttles

We run experiments to explore the feasibility of sharing the Initial Data Set collected from one shuttle with other shuttles. Figure 9 plots the reliability and throughput performance when using the initial data collected from other shuttles on Shuttles A and B. The results are normalized to the one when using the initial data collected from the same shuttle. As Figure 9a shows, the normalized reliability ranges from 0.98 to 0.99 when using the initial data collected from Shuttle B, C, D, E, and F for Shuttle A, while the normalized throughput ranges from 0.96 to 0.99. Similarly, the normalized reliability and throughput are not less than 0.94 and 0.97, respectively, when using the initial data collected from different shuttles for Shuttle B. The absolute reliability is not less than 0.87 and 0.83 when using the initial data collected from different shuttles on Shuttles A and B, respectively, which meets the reliability requirement specified by the application ($R_r = 0.8$). The results show that using the Initial Data Set collected from one shuttle for other shuttles only slightly degrades the performance of our runtime SF control solution when the shuttles follow the same route; therefore it is feasible to share the Initial Data Set among different shuttles, which significantly reduces the initialization overhead.



(a) An example data trace of PDR measurements. The application changes its PDR requirement from 0.8 to 0.9 at the 51st hour. The PDR measurement is computed in every 1.25 hours.

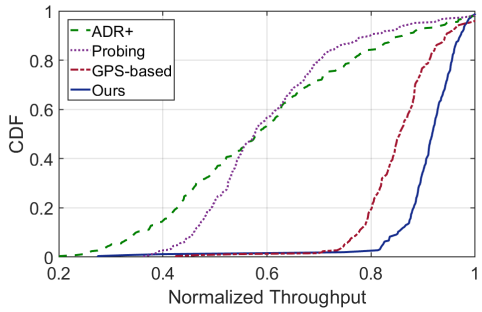


(b) CDF of PDR measurements.

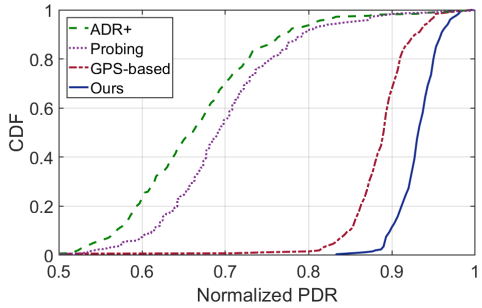
Fig. 10: Performance under different application reliability requirements.

C. Effectiveness of our Runtime SF Control Solution

We perform a series of experiments to examine whether our runtime SF control solution can consistently meet the reliability requirement specified by the application. We configure the application to input different reliability (PDR) requirements and measure the actual PDRs at the LoRa base station. Figure 10a plots the example PDR measurements collected from a shuttle for more than 100 $hours$. In this example, the



(a) CDF of normalized throughput.



(b) CDF of normalized PDR.

Fig. 11: Performance comparisons between our solution and three baselines. Performance is normalized to the one using the optimal selections.

application inputs 0.8 as the PDR requirement at the beginning of the Operation Period and then changes the requirement to 0.9 at the 51st hour. As Figure 10a shows, our runtime SF control solution can always meet the application reliability requirement except for the first measurement. The slightly lower reliability (0.76) in the first measurement is caused by the voting threshold adjustments performed at the beginning of the Operation Period. More importantly, our runtime SF control solution only takes 117 μs to select a new SF to accommodate the reliability requirement changes issued by the application. This demonstrates the time efficiency of our SF selections. Figure 10b plots the Cumulative Distribution Function (CDF) of PDR measurements under different application reliability requirements. The PDR measurement is computed every 25 $minutes$. In more than 95.8% and 90.2% of the time, our runtime SF control solution provides good SF selections, which successfully meet the application reliability requirements, 0.8 and 0.9, respectively.

Our runtime SF control solution is designed to maximize the link throughput. We compare the throughput provided by our solution against three baselines: ADR+, Probing, and GPS-based. ADR+ is an enhanced version of ADR, which takes input from the average (instead of maximum) SNR of the last 20 packets and selects SFs based on the required SNR for each SF configuration [12]. Probing is a LoRa transmission parameter selection algorithm based on the measured link Packet Reception Ratio (PRR) [18]. GPS-based is a baseline that we

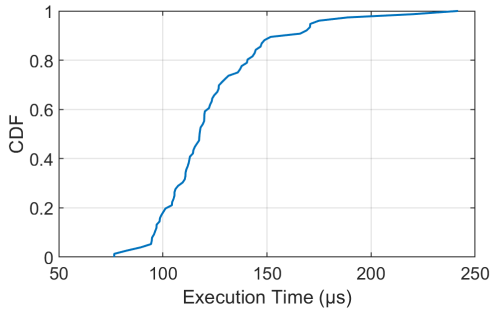


Fig. 12: The execution time of SF selections on a Raspberry Pi computer.

create by installing GPS devices on the shuttles and using GPS coordinates to select SF configurations. Figure 11 plots the comparisons among four solutions, where all results are normalized to the optimal values. Figure 11a shows the CDF of normalized throughput. The median throughput normalized to optimal is 0.58, 0.57, 0.86, and 0.92, when the LoRa base station runs ADR+, Probing, GPS-based, and our solution, respectively. Figure 11b shows the CDF of normalized PDR. The median normalized PDR is 0.66, 0.69, 0.89, and 0.93 when the LoRa base station runs ADR+, Probing, GPS-based, and our solution, respectively. As Figure 11a and 11b show, our runtime SF control solution consistently provides the highest throughput and best reliability among all solutions. The result also indicates that the measured link characteristics can be used reliably to select good SF configurations and there is no need to install additional GPS devices which are cost and time inefficient.

D. Time Efficiency of our Runtime SF Control Solution

Our KNN-based SF selection is designed to be lightweight. We measure the time duration taken by our KNN-based SF selector to select the best-suited SF configuration. We record the time of the events when the input is fed into the selector and the output (i.e., SF configuration) is generated. For this experiment, we repeat the measurement 50,000 times on the Raspberry Pi computer with a 900MHz quad-core ARM CPU and the RAM of 1GB. Figure 12 plots the CDF of the time duration of each run. As Figure 12 shows, the median execution time is 117 μs. 90% and 99% of the SF selections finish within 165 μs and 241 μs, respectively. These results demonstrate the good time efficiency of our SF selector as well as the advantage of using the KNN algorithm.

VII. Related Work

Satellite and cellular technologies are traditionally used to collect real-time data from moving vehicles through long-distance links. For instance, vehicular satellite links have been used to connect emergency vehicles to information headquarters in disaster areas [3], while LTE-based communication systems have been integrated into the urban transit systems [1], [2]. LTE-based vehicle-to-everything (V2X) communication is currently being standardized by 3GPP [19]. Unfortunately,

those satellite or cellular based systems are often very costly because they use expensive devices and licensed frequency bands, which limits their applications. In recent years, LoRa, an emerging LPWAN technology, has been used as a low-cost alternative that provides the capability for long-range data collection to low data rate applications. For instance, Islam et al. proposed a LoRa link scheduling algorithm and tested it in city environments [4]. Liando et al. conducted large-scale measurements on the performance of a campus-wide LoRa network and studied the impact of LoRa transmission parameters on link performance [20]. More recently, LoRa has been employed to support vehicular communication. For example, Santa et al. developed a LoRa-based vehicular monitoring platform [21]. Salazar-Cabrera et al., Boshita et al., and Guan et al. presented prototypes of public vehicle tracking systems that use LoRa to transfer the real-time locations and operating conditions of vehicles [22]–[24]. Bertoldo et al. deployed LoRa end devices on public transportation vehicles to transfer environmental sensor readings [5]. Ouya et al. proposed a LoRa-based communication protocol that allows electric vehicles and charging stations to exchange information on energy demand and availability [25]. The existing studies have demonstrated the feasibility of using LoRa to reliably collect data over long distances. To our knowledge, our work is the first to investigate the SF selection for LoRa end devices installed on running vehicles, distinguished from previous work using static SF configurations. Our yearlong empirical study provides valuable insights on selecting SF configurations for the LoRa end devices with mobility. Our work is therefore orthogonal and complementary.

In the literature, several approaches have been proposed to configure SF for LoRa networks based on link quality. For instance, the ADR algorithm, specified in LoRaWAN, estimates the link quality using the maximum SNR in 20 historical samples and selects the SF configuration based on the required SNR for each SF [11]. ADR+ is an enhanced version of ADR which replaces the maximum SNR with the average SNR to estimate the link quality [12]. The Probing algorithm makes use of the measured PRR to configure SF and gradually approaches the optimal configuration [18]. Unfortunately, those SF control approaches designed for stationary LoRa end devices do not work well for mobile devices. As presented in Section VI-C, our runtime SF control solution significantly outperforms the existing solutions. There also exist some approaches that estimate the link quality and select the SF configuration based on the GPS locations of LoRa end devices [26]–[28]. However, those GPS-based approaches significantly increase the system cost and power consumption of LoRa end devices. Our experimental results presented in Section VI-C show that the measured link characteristics can be used reliably to select good SF configurations and there is no need to install those GPS devices.

KNN is a classical machine learning technique that uses the nearest neighbors in the collected data set to determine the class or value of a query example [17]. KNN has been demonstrated as an efficient and effective algorithm when applied to solve many wireless communication problems. For instance, Yu et al. and Arya et al. used KNN for indoor and outdoor localization based on RSS measurements, respectively [29], [30]. Li et al. employed KNN to detect network intrusions in Wireless Sensor Networks (WSNs) [31] while Pan et al. used KNN to estimate the missing data during data transfers in WSNs [32]. Donohoo et al. used KNN to predict the energy demand of mobile devices [33]. Ma et al. employed KNN to predict the PRR of 802.11 links based on SNR measurements [34]. To our knowledge, our work is the first to use KNN to select SF configuration for mobile LoRa end devices. Our experimental results presented in Section VI demonstrate the effectiveness and efficiency of our proposed solution.

VIII. Conclusion

Satellite and cellular technologies are traditionally used to collect real-time data from running vehicles to the base station through their long-distance links. However, such systems are often costly because of their use of expensive devices and licensed frequency bands, which prevents them from being used in many application scenarios. As an emerging LPWAN technology, LoRa has been used as a low-cost alternative that provides capability for long-range data collection to low data rate applications. In this paper, we present a low-cost LoRa-based wireless network, ShuttleNet, that collects real-time data from six shuttles circling our university campus and has operated in the real world for more than a year. When implementing ShuttleNet, we find that the selection of LoRa SF poses a significant challenge because of its effects on two conflicting QoS metrics. To investigate the impact of SF configuration on network performance, we have performed a 14-month empirical study. Our empirical study shows that a larger SF provides higher network reliability at the cost of lower throughput. More importantly, we observe that the link reliability increases more significantly when the LoRa transmitters use large SFs ($SF11$ and $SF12$), while the link throughput decreases dramatically at small SFs. This indicates that increasing SF when the current SF is large may significantly enhance the link reliability at the cost of slightly reducing the link throughput, while slightly decreasing SF when the current SF is small may significantly improve the throughput without introducing too much damage on the link reliability. Those observations motivate us to develop a runtime SF control solution that employs the KNN algorithm to adapt the SF configuration based on the link characteristics. We compare our solution against three state-of-the-art baselines and observe that ours consistently provides the highest throughput and the best reliability among all solutions.

Acknowledgment

This work was supported by the NSF through grant CRII-1657275 (NeTS) and the Transportation and Parking Services at SUNY at Binghamton through a grant. We thank our IT specialists Mr. Robert L Mess and Mr. Dave Hall for assisting us in deploying and maintaining the LoRa base station.

REFERENCES

[1] T. Tang, K. Dai, Y. Zhang, H. Zhao, and H. Jiang, "Field Test Results Analysis in Urban Rail Transit Train Ground Communication Systems of Integrated Service Using LTE-M," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2016.

[2] A. Khayat, M. Kassab, M. Berbineau, and A. Belghith, "LTE Based Telecommunication System for Urban-Guided Transports," in *Transport Research Arena (TRA)*, 2014.

[3] G. Iapichino, C. Bonnet, O. del Rio Herrero, C. Baudoin, and I. Buret, "Advanced Hybrid Satellite and Terrestrial System Architecture for Emergency Mobile Communications," in *International Communications Satellite Systems Conference (ICSSC)*, 2008.

[4] M. T. Islam, B. Islam, and S. Nirjon, "Duty-Cycle-Aware Real-Time Scheduling of Wireless Links in Low Power WANs," in *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2018.

[5] S. Bertoldo, L. Carosso, E. Marchetta, M. Paredes, and M. Allegretti, "Feasibility Analysis of a LoRa-Based WSN Using Public Transport," *Applied System Innovation*, vol. 1, no. 4, 2018.

[6] B. Soret, P. Mogensen, K. I. Pedersen, and M. C. Aguayo-Torres, "Fundamental Tradeoffs among Reliability, Latency and Throughput in Cellular Networks," in *IEEE Globecom Workshops (GC Wkshps)*, 2014.

[7] LoRa Modulation Basics. [Online]. Available: https://www.semtech.com/

[8] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the Limits of LoRaWAN," *IEEE Communications magazine*, vol. 55, no. 9, 2017.

[9] F. Delobel, N. El Rachkidy, and A. Guitton, "Analysis of the Delay of Confirmed Downlink Frames in Class B of LoRaWAN," in *IEEE Vehicular Technology Conference (VTC Spring)*, 2017.

[10] A. A. Syed, W. Ye, J. Heidemann, and B. Krishnamachari, "Understanding Spatio-Temporal Uncertainty in Medium Access with ALOHA Protocols," in *Proceedings of Workshop on Underwater Networks (WuWNet)*, 2007.

[11] LoRaWAN Adaptive Data Rate. [Online]. Available: https://www.thethingsnetwork.org/

[12] M. Slabicki, G. Premsankar, and M. Di Francesco, "Adaptive Configuration of LoRa Networks for Dense IoT Deployments," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2018.

[13] RN2903 Provided by MICROCHIP. [Online]. Available: https://www.microchip.com/wwwproducts/en/RN2903

[14] iC880A Provided by IMST. [Online]. Available: https://www.wireless-solutions.de/products/radiomodules/ic880a.html

[15] O. Afisiadis, M. Cotting, A. Burg, and A. Balatsoukas-Stimming, "On the Error Rate of the LoRa Modulation with Interference," *Transactions on Wireless Communications*, 2019.

[16] SX1272 Datasheet Provided by Semtech. [Online]. Available: https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1272

[17] P. Cunningham and S. J. Delany, "k-Nearest Neighbour Classifiers," *Multiple Classifier Systems*, vol. 34, no. 8, 2007.

[18] M. Bor and U. Roedig, "LoRa Transmission Parameter Selection," in *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2017.

[19] S. Chen, J. Hu, Y. Shi, Y. Peng, J. Fang, R. Zhao, and L. Zhao, "Vehicle-to-Everything (V2X) Services Supported by LTE-Based Systems and 5G," *IEEE Communications Standards Magazine*, vol. 1, no. 2, 2017.

[20] J. C. Liando, A. Gamage, A. W. Tengourtius, and M. Li, "Known and Unknown Facts of LoRa: Experiences from a Large-Scale Measurement Study," *ACM Transactions on Sensor Networks*, vol. 15, no. 2, 2019.

[21] J. Santa, R. Sanchez-Iborra, P. Rodriguez-Rey, L. Bernal-Escobedo, and A. F. Skarmeta, "LPWAN-Based Vehicular Monitoring Platform with a Generic IP Network Interface," *Sensors*, vol. 19, no. 2, 2019.

[22] R. Salazar-Cabrera, Á. Pachón de la Cruz, and J. M. Madrid Molina, "Proof of Concept of an IoT-Based Public Vehicle Tracking System, Using LoRa (Long Range) and Intelligent Transportation System (ITS) Services," *Journal of Computer Networks and Communications*, 2019.

[23] T. Boshita, H. Suzuki, and Y. Matsumoto, "IoT-Based Bus Location System Using LoRaWAN," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[24] P. Guan, Z. Zhang, L. Wei, and Y. Zhao, "A Real-Time Bus Positioning System Based on LoRa Technology," in *IEEE International Conference on Smart Grid and Smart Cities (ICSGSC)*, 2018.

[25] A. Ouya, B. M. De Aragon, C. Bouette, G. Habault, N. Montavont, and G. Z. Papadopoulos, "An Efficient Electric Vehicle Charging Architecture Based on LoRa Communication," in *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2017.

[26] M. Asad Ullah, J. Iqbal, A. Hoeller, R. D. Souza, and H. Alves, "K-Means Spreading Factor Allocation for Large-Scale LoRa Networks," *Sensors*, vol. 19, no. 21, 2019.

[27] S. Demetri, M. Zúñiga, G. P. Picco, F. Kuipers, L. Bruzzone, and T. Telkamp, "Automated Estimation of Link Quality for LoRa: a Remote Sensing Approach," in *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2019.

[28] N. Benkahla, H. Tounsi, S. Ye-Qiong, and M. Frikha, "Enhanced ADR for LoRaWAN Networks with Mobility," in *IEEE International Wireless Communications & Mobile Computing Conference (IWCMC)*, 2019.

[29] F. Yu, M. Jiang, J. Liang, X. Qin, M. Hu, T. Peng, and X. Hu, "5G WiFi Signal-Based Indoor Localization System Using Cluster k-Nearest Neighbor Algorithm," *International Journal of Distributed Sensor Networks*, vol. 10, no. 12, 2014.

[30] A. Arya, P. Godlewski, and P. Mellé, "Performance Analysis of Outdoor Localization Systems Based on RSS Fingerprinting," in *International Symposium on Wireless Communication Systems (ISWCS)*, 2009.

[31] W. Li, P. Yi, Y. Wu, L. Pan, and J. Li, "A New Intrusion Detection System Based on KNN Classification Algorithm in Wireless Sensor Network," *Journal of Electrical and Computer Engineering*, 2014.

[32] L. Pan and J. Li, "K-Nearest Neighbor Based Missing Data Estimation Algorithm in Wireless Sensor Networks," *Wireless Sensor Network*, vol. 2, no. 02, 2010.

[33] B. K. Donohoo, C. Ohlsen, S. Pasricha, Y. Xiang, and C. Anderson, "Context-Aware Energy Enhancements for Smart Mobile Devices," *IEEE Transactions on Mobile Computing*, vol. 13, no. 8, 2013.

[34] Y. Ma, "Improving Wireless Link Delivery Ratio Classification with Packet SNR," in *IEEE International Conference on Electro/Information Technology (EIT)*, 2005.