Estimating Passenger Queue for Bus Resource Optimization using LoRaWAN-enabled Ultrasonic Sensors

Thanchanok Sutjarittham, Hassan Habibi Gharakheili, Salil S. Kanhere, and Vijay Sivaraman

***Abstract*—Anecdotal evidence has shown that bus stops around our University campus, specifically those serving express buses to city center, can get very crowded during certain times. This not only causes immense frustration to students who experience large variations in wait-time, but also creates challenges for the university and transport authority in knowing when to schedule extra buses. This paper outlines our efforts to instrument a main bus stop on our university campus with IoT sensors to monitor passengers queue length. Our specific contributions are as follows: (1) We begin by developing a LoRaWAN ultrasonic sensor for detecting people in the queue. The sensor emits an ultrasonic tone pulse every few seconds, and determines whether someone is in front of it based on the reflections received, if any. Ten sensor units are built, tested, and tuned in a lab environment to achieve optimum detection accuracy and data transmission rate; (2) Next, we install these sensors at a 6-meter interval along the campus fence bordering the bus-stop. We develop an algorithm to infer the number of passengers in the queue from sensor data and demonstrate that it achieves reasonable accuracy with a mean absolute error of 10.7 people (for a queue size of up to 100 people); and (3) we develop an optimization model to reschedule bus dispatching time, aiming to minimize total wait time of passengers. We show that a reduction of up to 42.93% in passengers' wait time can be achieved by adopting demand-driven bus scheduling.**

***Index Terms*—LoRaWAN sensors, bus queue monitoring, optimization**

I. Introduction

University campuses are essentially a microcosm of a city and can often encompass vast areas (*e.g.,* Stanford owns 8183 acres of land). They include a diverse range of facilities including residences, sports centres, offices, lecture theatres, and public transport stops. Universities are thus under constant pressure to improve efficiencies and offer better services to the various stakeholders including students, staff and visitors. On a typical weekday, thousands of people would arrive and depart the campus at various times of day. A vast majority of these would avail of public transport services for their daily commute. Transit stops at universities are notorious for overcrowding during peak hours [1]. It is not uncommon for commuters to wait for a second or even a third service to arrive before there is sufficient room to board.

Prolonged waiting is highly undesirable on multiple fronts including loss of productivity, fatigue, tiredness and discouragement for using public transport. Advanced transport system has provided myriad of opportunities to improve the experience of drivers and passengers [2]. In an effort to assuage this problem, many public transport providers have deployed real-time vehicle tracking services, whereby the arrival time of the next service is displayed in real-time on a display installed at transit stops or through mobile applications. During peak times, the passenger demand may outstrip the available capacity on the next arriving service, resulting in many passengers having to wait for longer. A system that can measure the length of the queue at transit stops in real-time would therefore be favorable for passengers, by aiding their decisions on transit stop selection and trip start time. The ability to estimate passenger demand would also enable a pragmatic approach to transit scheduling, which would lead to efficient transit resource management and thus an enhancement in commuter transit experience.

However, monitoring passenger queue length and wait time at transit stops still remains a challenge, especially in an outdoor scenario where power and wired connectivity are usually inaccessible, thus hindering sensor deployment efforts. Many studies have proposed mobile detection and crowd-sensing based approaches [3], [4] to measuring queue size or dwelling time of people, yet the solutions have only been tested and deployed in an indoor environment. Furthermore, such systems would require a fair fraction of queue members to actively participate in order to achieve a good accuracy. Existing works related to measuring crowding in an outdoor scenario are mostly camera based [5], and if applied to queue measurement can raise privacy concerns.

In this paper, we propose a novel end-to-end sensor-based system for measuring queue length in an outdoor setting. The solution is weather resistant, battery-operated, and communicates wirelessly, allowing outdoor deployment where access to power and communications ports are infeasible. We deploy our system to monitor one of the busiest bus stops on our University campus and show that our collected data can improve operational transit scheduling decision. Our key contributions are as follows: **(1)** we design and implement people detector devices using ultrasonic sensing. The device uses LoRaWAN for data communication by utilizing a public LoRaWAN base-station located on our campus. We test and tune the sensor and LoRaWAN-based parameters to achieve optimum detection sensitivity and superior data delivery ratio for our deployed environment while achieving low energy consumption. Our implementation choices enable ease of deployment in an

T. Sutjarittham, H. Habibi Gharakheili, and V. Sivaraman are with the School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, NSW 2052, Australia (e-mails: t.sutjarittham@unsw.edu.au, h.habibi@unsw.edu.au, vijay@unsw.edu.au).

S. Kanhere is with the School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia (e-mail: salil.kanhere@unsw.edu.au).

outdoor space; **(2)** We deploy ten people detector devices at one of the busiest bus stops on campus. The sensed data and empirical observations are used to develop an algorithm to infer queue length. Our method yields a reasonable accuracy with Mean Average Error (MAE) of 10.7 people for a queue size of up to 100 people; **(3)** We demonstrate that the data on actual transit demand can be used to optimize dispatching time of buses by formulating an optimization model, aiming to minimize total wait time of passengers at the bus stop. The experiments performed on the real-world deployment data show that up to 42.93% of passenger wait time can be reduced by reallocating the existing bus resources.

To the best of our knowledge, we are the first to employ an ultrasonic sensor-based solution using LoRaWAN to address the challenges in measuring queue length in an outdoor scenario. We use bus stops as an illustrative example for public transit stops. Our solution could also be applied to other transportation modes, and for any outdoor scenario (*e.g.,* stadiums, airports) where an orderly queue is formed.

The rest of this paper is organized as follows: Section §II describes relevant prior work. In §III, we present our design and implementation of people detector units. Our developed intelligent queue inference algorithm and deployment of the sensors are described in §IV. We then show the potential of collected passenger demand in optimizing transit bus scheduling in §V. Finally, this chapter is concluded in §VI.

II. Related work

Many existing works have proposed methods to estimate queue length [6], [7], waiting-time [3], [8], or crowd density in general [9], [10]. This section surveys the most commonly used approaches to deduce crowding information.

WiFi/Bluetooth Signals: Due to the prevalence of mobile devices that support wireless communication such as WiFi and Bluetooth, many researchers have attempted to leverage these signal traces in estimating human crowd information. For instance, Wang et al. [6] uses a single WiFi monitor to track human queue in retail environment by analyzing RSS trace from mobile devices, Shu et al. [8] uses WiFi positioning data to estimate queueing time at an airport, and several works [9], [11] estimate crowd density within an indoor area by exploiting WiFi and Bluetooth captures. These methods, require WiFi access points or at least a signal monitoring device to be installed at certain locations, which may not be suitable for an outdoor scenario.

Crowd-sensing or Crowd-sourcing: Crowd-sensing through the use of information obtained from smartphones has been explored to estimate people crowding information. Okoshi et al. [3] and Li et al. [4] use crowdsourced data from sensors embedded in smartphones such as accelerometer and magnetic compass to estimate queue waiting time in a retail environment. The proposed methods are able to distinguish queuers from non-queuers with good accuracy, yet high number of participants (who are willing to install and have the application running on their mobile phones) are required to achieve such good results. Elhamshary et al. [10] also leverages data sensed from users' smart phones in order to estimate crowding level in railway stations by exploiting a wider range of sensed data including individual's trajectory from entrance and ambient sound from smart phone's microphone. However, the method is not suitable for queue-specific detection, but instead can be used for monitoring of human crowd-level in general. An even more user-centric approach is using crowd-sourcing kiosks that asks users about their estimated waiting time through an interactive interface [12]. This method poses several drawbacks including a high likelihood of insufficient contributions from queuers and deployment difficulty.

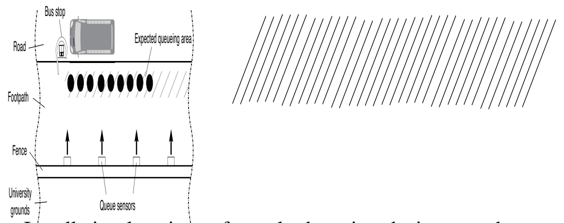


Fig. 1. Installation locations of people detection devices at a bus stop.

Camera: A more traditional approach for crowd detection is the use of camera-based systems [7], [13], [14], employing video or image analysis. However, this approach is expensive, endangers privacy, and its accuracy can largely depend on the camera's field of view. The camera-based approach also poses deployment difficulties, especially in an outdoor environment where power and communications are not easily available.

Most of existing methods for measuring crowding and queue information are only applicable to an indoor setting such as retail environments or airports. Works that have been deployed in an outdoor settings either aim to estimate overall crowd density [15] or focus on tracking individuals [16], [17]. To the best of our knowledge, none of these works attempt to estimate the length of an orderly queue and waiting time in an outdoor scenario.

III. Sensing People in Queue: Design, Implementation, and queue inference algorihtm

In this section, we propose a novel end-to-end sensor-based system for human queue length measurement in an outdoor space. We first describe the design and implementation of a people detector unit (PDU), which serves as a sensor unit in our system. Additionally, we demonstrate the performance of the sensors through experimentation and parameter tuning. We then introduce an algorithm that infers queue length information from the sensed data. Note that we obtained appropriate ethics clearance (UNSW Human Research Ethics Advisory Panel approval no. HC180359) prior to conducting this research work.

A. Design Decisions

Our bus stop of interest is located outside our university campus which serves express bus service to the city center. The formation of the queue is in a straight line, along the campus fence. Fig. 1 shows the presence of passenger queue relative to the road, bus stop, campus ground, and fence. The expected area where the queue forms is approximately at a distance of 2-3 meters from the fence and can on occasion extend beyond the edge of the fence line. The goal of each PDU is to detect and indicate whether there is a queue formation in front of

TABLE I
SUMMARY COMPARISON OF SENSOR TECHNOLOGIES FOR BUS QUEUE MONITORING.

Technology	**Contact based**	**Passive/ Active**	**Spacial**	**Privacy intrusive**	**Unit cost (USD)**	**Relative power draw**	**Other limitations**
Scannable QR codes	no	passive	single	yes	10^{-1}	None	Low participation, high abuse.
Pressure pads	yes	active	either	no	10^{2}	Low	Council permission, tripping hazard.
Passive infra-red	no	active	single	no	10^{1}	Low	Only useful at night.
Camera	no	active	multi	yes	10^{2}	High	Large data volumes.
Laser time-of-flight	no	active	single	no	10^{2}	Low	Expensive & niche.
Ultrasonic time-of-flight	no	active	single	no	10^{0}	Low	
WiFi session logs	no	active	multi	yes	0	None	Unreliable, only covers certain WiFi users (not all queue members).

it. Since power and communications points are not available within the proximity of the bus stop, we chose to use an array of small single spatial sensors rather than a single multi-spatial sensor (such as camera) due to portability and battery life requirements. Our choices of sensors and communications technologies are discussed next.

1) Sensing Choices: We now compare several sensing technologies for people detection as summarized in Table I. Criteria used for our comparison include whether the sensor is contact-based (require physical contact with queue members to take measurement), whether the sensor is passive (require conscious effort from queue members to record readings), collected data size, privacy, cost, and power draw.

Scannable QR code is a form of passive sensor that requires queue members to scan unique QR code posters placed along the length of a bus queue. Despite being a cheap and flexible option, sufficient frequency of participation is required to make the solution effective.

Pressure-pads are flat devices placed on the ground to detect the weight of users standing on them. Unfortunately their installation on a footpath is complicated, requiring extensive approvals, mechanical protection (vandalism, weather, vehicles), and safety risk mitigation (tripping hazards).

Passive infrared sensors detect the infra-red light emitted by all warm entities (humans). However, these sensors cannot discriminate between people and other sources of heat (*e.g.,* vehicles, sunlight), making them inappropriate for outdoor roadside installation and operation, especially during the day.

Camera sensors: A low-resolution of 640×480 pixels camera already requires the transmission of 300,000 samples, which even with compression will still require significant time and energy. Additionally these devices collect personally identifiable information, which would require more complex data handling and approval arrangements.

Existing WiFi infrastructure produces session logs that can potentially be used to determine queue occupancy without additional capital costs, however this method has severe accuracy and validity limitations. Only users of the university WiFi systems will be measured. Moreover, it is difficult to determine exactly where users are relative to each access point (*e.g.,* in a bus queue or in a nearby cafe) and WiFi coverage can often be poor at bus stops as they are usually located at the edge of the campus.

Ultrasonic distance sensors and **laser time-of-flight** sensors take distance measurements between themselves and objects in front of them. They operate by emitting a small amount of sound or light, and measure how long it takes for this wave to bounce off a nearby object and come back. Ultrasonic waves are sound waves above human hearing, typically chosen to avoid irritating humans and to allow smaller physical sensor size. As sound waves are many magnitudes slower than light waves, ultrasonic distance sensors are typically much simpler, cheaper, and available in more varieties than their laser counterparts.

Summary: We chose ultrasonic distance sensors because of their low power requirements, low cost, low implementation complexity and high likelihood of detecting the presence of individuals in a queue while not being adversely impacted by other objects in the environment. Compared to light and infra-red solutions, they do not suffer interference problems (sunlight, car headlights, etc), and compared to solutions such as WiFi they are less likely to miss a fraction of queue members. Ultrasonic, whilst not a perfect sensing method, avoids most of the disadvantages of other sensing technologies.

2) Communication Choices: Data collected by sensor units must be communicated back to a central location for permanent storage and analysis. Real-time streaming of collected data permits immediate analysis and reporting. Wired communication is infeasible due to its high infrastructure costs and its inflexibility for making positional adjustments. Thus we only consider wireless communications solutions for which we have a set of requirements including low power draw, simple to operate, long-range, low-cost, and multiple devices should be able to directly communicate with a gateway. Note that high data-rate is not a requirement for our PDU, as the distance measurement data collected is at the scale of a few bytes per minute. In the following, we discuss several wireless communication options in detail. A summary of our comparison is available in Table II.

WiFi (802.11) is a large collection of standards optimized for high data-rates and persistent sessions. Notable amount of time, along with exchanging of control messages is required to initiate wireless connections; hence demanding high software complexity and high power draw.

Bluetooth is a collection of standards optimized for master-slave communications. While this is optimal for operating simple devices such as microphones and speakers, it is not effective or simple for the operation of an arbitrary number of PDUs simultaneously. Some methods allow for multiple client devices ("Bluetooth Piconet"). However, this requires further synchronization complexity, and imposes a (low) total device limit of 7.

TABLE II
SUMMARY COMPARISON OF COMMUNICATION TECHNOLOGIES FOR BUS QUEUE MONITORING.

Technology name	Data-rates	Intended Range(Max)	Primary topology	Usage Complexity	Power draw	Cost
WiFi	Low to high	Medium	Star, P2P	High	High	Low
Bluetooth	Low to med	Short	Master-slave	High	Low	Low
Wireless broadband	Low to high	Long	Star	High	Low	High
Zigbee, NRF24, etc	Low to med	Short	Various	Low	Low	Low-Moderate
LoRaWAN	Low	Long	Star	Low	Low	Moderate

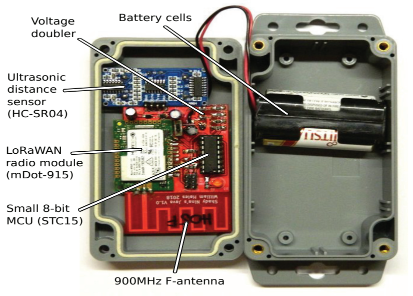

Voltage doubler
Battery cells
Ultrasonic distance sensor (HC-SR04)
LoRaWAN radio module (mDot-915)
Small 8-bit MCU (STC15)
900MHz F-antenna

Fig. 2. Interior of a PDU.

Wireless broadband systems such as 3G, 4G, and LTE can be optimized for low power signaling and sleeping after a connection is initially negotiated, however compliant radio modules are typically very expensive and complex to operate. Many vendors require signature of non-disclosure agreements and external Linux-running microprocessors with proprietary drivers for these devices to function.

Proprietary low-power communications systems and modules such as **IEEE 802.15.4** (Zigbee) and **NRF24** provide low-cost, simple-interfaced and low-power solutions. Unfortunately their range is very limited, requiring the installation of local (line-of-sight) reception towers near instrumented bus stops.

LoRaWAN and **LoRa** are a set of low-power and long range communications standards. Radio modules are easily available with simple UART-style interfaces and communications can be performed over kilometers in non-line-of-sight urban environments. The typical network topology is star, with an arbitrary number of low-power devices transmitting to a centralized (infrastructure) base-station for data collection. Base-stations themselves can either be shared (public) resources run by third parties or private infrastructure [18], in both cases the standards use an advanced encryption standard (AES) implementation to provide message security [19]. Unfortunately LoRa is implemented on a wide variety and disparate set of frequency standards [20], with many LoRaWAN radio modules on the market only supporting a single frequency band. LoRaWAN radio modules that meet local regulatory requirements are also not necessarily cheap.

Summary: We chose LoRaWAN as the communications platform for this work because of its longer range, lower power draw, and simpler implementation complexity compared to other alternatives.

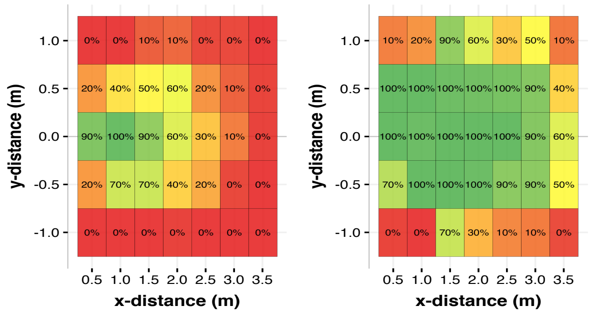
B. Implementing People Detector Units

We designed and assembled ten people detector units (PDUs) using Ultrasonic distance sensors and LoRaWAN radio modules, covering the entire fence length in our deployment environment with approximately ten people queuing between two consecutive sensors. Sensor units were designed to measure the presence of a bus queue as shown in Fig. 1. An array of small single-spatial sensors was chosen over a single multi-spacial sensor (such as a camera) due to portability and battery life requirements, as discussed in §III-A1. Each component of the PDU and important design decisions taken during the implementation are described below:

1) Control board: A red circuit board containing a small 8-bit micro-controller and miscellaneous periphery components was used to coordinate each PDU. We adopted an "wake-and-return-to-sleep" design to maximize battery life. Each PDU spends as much time as possible sleeping, only waking to take brief measurements or transmit measured data. PDUs are also configured to stop transmitting if they are unable to take any successful measurements. This greatly reduces the amount of transmissions when there is low occupancy at the stop, both increasing battery life and reducing interference to external devices using the same radio spectrum. Furthermore, the message payload is optimized to reduce message length. Distances are stored as 8-bit values with the resolution of distance measurement of 2 cm (permitting a theoretical measurement range of 0-510 cm). No checksum or other metadata are included in the payload, instead we rely upon the LoRaWAN protocols for reliability and error detection.

2) Ultrasonic Sensor: The HC-SR04 ultrasonic distance sensor is a low-cost off-the-shelf greymarket part[1], most variants of which can provide practical distance measurements from 0.01 m to 3 m with better than 0.02 cm of resolution depending on the target object being observed. Our testing reveals most HC-SR04 sensors are unreliable for detecting soft and uneven objects such as humans beyond approximately 1 meter, where successful measurement rates fall below 50%. A failed measurement ("infinite distance") indicates that either a queue member in front of a sensor has been missed (false negative) or that there is no detection within 3.5 meters (true negative). False positives (incorrectly detecting people in empty queue) are rare events for our sensors. In order to address the issue of unreliable detection, we configure the ultrasonic sensor to repeatedly make up to 5 measurement attempts (with a millisecond gap) until a valid (non-infinite) measurement is obtained. This configuration allows us to afford a high accuracy while maintaining low false-positive and -negative rates. The sensors are configured to measure and record the distance every 10 seconds, and transmit (via their LoRaWAN interface) a batch of 6 measurement records every minute.

[1]"Greymarket" parts are manufactured by a variety of vendors with no central authority and often no relevant/accurate data-sheet. Use of these parts is common in most manufactured electronics today, however the increased variance in quality & performance demands more attention be paid by users to their testing & implementation.



(a) Unmodified ultrasonic sensor. (b) Modified ultrasonic sensor.

Fig. 3. Probability of detecting a human at various proximities of the sensor (the sensor is located at origin).

Multiple PDUs operating in the same location could potentially pick up each other's ultrasonic pulses, providing incorrect measurements. To statistically minimize this effect, we design the sensor units to be as non-deterministic in their timing as possible; where small changes in measurement success (num of cycles), transmit time and clock speed would prevent PDUs from synchronizing their behavior. Units are RC-oscillator controlled (rather than crystal controlled) and do not contain real-time clock. Even if all units are turned on simultaneously they naturally spread out their timings.

3) LoRaWAN Radio: Multitech mDot radios[2] are chosen as they were the only market-available option at the time of implementation that was compatible with the local (legal) LoRa frequency plan, available off-the-shelf (in-stock) and provided a well-documented UART interface for simple operation. We choose to utilize a public LoRaWAN base station already provisioned by our campus Estate Management. The use of a public tower necessitated the use of an intermediary organization, The Things Network (TTN)[3], in our data collection.

4) Power supply: Two 1.5V AA alkaline cells are used due to the concern about the longevity of rechargeable solutions in enclosures placed in direct sunlight. The 3V supplied is enough to operate the micro-controller and radio, however it is not sufficient for the ultrasonic distance sensor, which requires 5V. We use a discrete voltage doubler (operating at a fixed frequency by the micro-controller) as a simple solution to this problem. It has near-zero quiescent power draw and was later modified (see §III-C3) to provide extended battery life operation.

C. Performance Evaluation and Tuning

In this subsection, we evaluate the performance of our PDUs in terms of detection range of the ultrasonic sensor, communication reliability of the LoRaWAN radio, and energy consumption of the entire unit.

1) Detection Range: Our PDU utilizes a HC-SR04 ultrasonic sensor to return the distance to the closest object from the sensor. We first evaluate the detection ability in our laboratory. Our initial testing verified that the sensor performed well in returning the distance to large solid objects, however detection of people was erratic. We found that the sensor underperformed when the human subject moved more than 1.5 meters away from it. Also, the detection ability varied depending on the type of clothing (e.g., soft long sleeved top) worn by the subject. The cause of such poor detection ability with people is due to ultrasonic waves not being reflected strongly off people to be detected at the sensor. Soft clothing is also not a good reflector of ultrasonic waves when compared to bare human skin.

[2]https://www.multitech.com/brands/multiconnect-mdot

[3]Public-use LoRaWAN base stations: https://www.thethingsnetwork.org/

TABLE III
LoRaWAN DATA RATE SETTINGS AND ASSOCIATED THEORETICAL TIME-ON-AIR FOR 6-BYTE PAYLOAD.

Data rate	DR0	DR1	DR2	DR3	DR4	DR5
Spreading Factor	12	11	10	9	8	7
BW (kHz)	125	125	125	125	125	125
Time-on-air (ms) (for 6-byte payload)	1319	741	330	185	103	51

Emil [21] has done an in depth analysis of the HC-SR04 performance and has reverse engineered it. The author found that the band-pass filter used on the receiver circuit for detecting the reflected ultrasonic wave is centered at 18kHz. As the wave emitted by the sensor is 40kHz, the reflected wave back to the sensor is also expected to be 40kHz and having the received wave filtered by a band-pass centered at 18kHz reduces the receivers sensitivity. Therefore, as suggested by Emil, we modified two resistors used in the filter circuit of the HC-SR04, and hence enhanced the sensitivity of the receiver.

We then evaluate two groups of sensors namely three units of "unmodified" and one unit of "modified" HC-SR04's. For each sensor, the human subject stood at cells on a clearly marked grid map. We collected 100 readings at each position, then the average was used for each group of sensors. The results of this experiment are shown in Fig. 3, where the sensor is located at coordinate (0,0) facing right. Each cell value indicates the probability of human detection (green cells show high detection and red cells show low detection). It is clearly seen that the modified sensor (Fig. 3b) has a better field of view, detecting the test subject in most situations even if they are not standing directly in front of the sensor. The detection range is also greatly increased. We also observe that the detection ability of the unmodified sensors tapers off after 1.5 meters whereas the detection ability of the modified sensors starts to taper off at a further distance of 3 meters. The results of the experiment is conclusive in demonstrating that the modified sensor has significantly better ability in detecting people than the original version.

2) LoRaWAN Communications Reliability: We quantify the reliability of LoRaWAN by using packet delivery ratio (PDR) as a performance metric. PDR is defined as the fraction of packets received by the gateway out of the total number of packets sent. Reliability of LoRaWAN notably depends on the deployment environment as well as settings on LoRaWAN physical layer. There are three main parameters that can be tuned including spreading factor, SP (number of bits used to represent a symbol, which directly impacts bit rate and thus time-on-air of packets); bandwidth, BW (defines the range of frequencies over which LoRa signal spreads, higher bandwidth allows for faster data transmission rate but reduces receiver sensitivity and communication range); and transmission power, TXP (amount of energy used to transmit a packet). LoRaWAN introduces a new variable called data rate (DR), which is

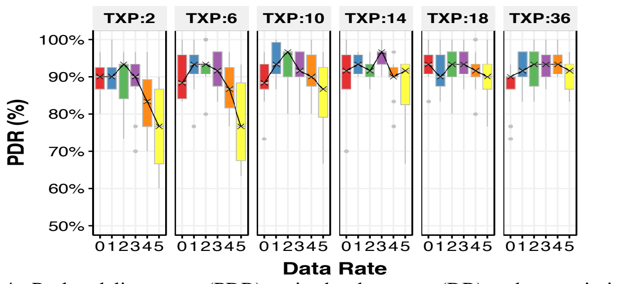


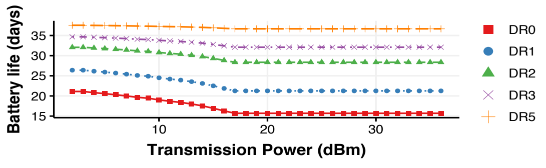
Fig. 4. Packet delivery rate (PDR) varies by data rates (DR) and transmission power (TXP).

an arbitrary configuration parameter used to define different combinations of SP and BW settings. The respective configurations (available in our country) that we evaluated are shown in Table III. Note that BW configurations are constant across all data rates, but the spreading factors vary from 12 (low data rate of DR0) to 7 (high data rate of DR5). In theory, lower data rate decreases the signal to noise ratio (SNR) limit at the receiver gateway (causing the receiver gateway to become more sensitive to the received signal), thus creating a more reliable LoRaWAN link. However, this will also raise time-on-air of the transmitted packet, which in turn increases the chance of collision.

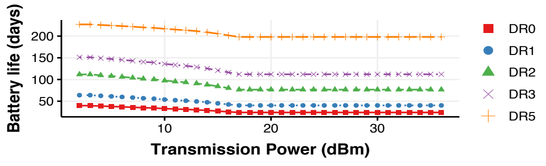
To evaluate the performance of various configurations of DR and TXP, we wrote a script to randomly cycle the setting combinations for all of our 10 PDUs. We consider six data rates (DR0-DR5) and six transmission power settings (TXP values of 2, 6, 10, 14, 18, and 36 dBm), thus a total of 36 combinations. The PDUs were set to transmit a 6-byte data packet every minute, and a total of 30 samples were collected for each setting combination per PDU. This yields a total sample size of 10,800 transmissions (from ten units). The random cycling of settings helps reduce sampling bias caused by environmental factors such as temperature, humidity, and line-of-sight that can impact the reliability of LoRa links. Furthermore, repeated measurement attempts (explained in §III-B2) were made during this phase of experiment to introduce random delays between successive transmissions, and thus preventing PDUs from getting synchronized, thus minimizing collisions. We installed our 10 PDUs on the fence (at the bus stop) to collect data for reliability analysis.

Fig. 4 shows PDR (y-axis) for various combinations of DR settings (x-axis) and transmission power settings (facets). We observe that for transmission power up to 10 dBm, PDR has a general declining pattern in data rate (from DR0 to DR5) and is widely spread in values between 70% and 90% – this behavior complies with other experimental studies on LoRaWAN performance [22], [23], where a low data rate was reported to result in more reliable link and thus gives a better PDR. By increasing the transmission power (TXP of 14 and 18 dBm), no obvious pattern is observed across various DR values, suggesting that the delivery rate is less susceptible to the changes in LoRaWAN data rate setting. This is due to the fact that higher transmission power yields stronger signals that are less prone to attenuation caused by the environment.

3) Energy Consumption: We evaluate the energy consumption of our PDUs by computing the estimated battery life. Battery lifetime (in days) can be computed by dividing the usable energy available in the batteries (found from manufacturers data sheet [24]) by the total daily energy consumed. We start by computing the energy consumed for one packet. In our PDUs, the LoRaWAN radio and the ultrasonic sensor contribute to the majority of energy consumption – the total energy consumed per packet can be calculated by adding energy consumed by LoRaWAN radio and by ultrasonic sensor. We experimentally measure the energy usage of both components using a small resistive shunt and an oscilloscope.



(a) Worst case.



(b) Best case.

Fig. 5. Estimate of battery life with different LoRaWAN parameter settings.

As mentioned in §III-B2, our ultrasonic sensors are configured to collect and transmit the distance data at regular intervals, but each sensor may attempt up to 5 times before they record a valid measurement. Therefore, the total energy consumed by the sensor can vary depending on the number of measurement attempts. The best case scenario is when the first measurement in a session successfully detects the queuer and the worst case is when all five measurement attempts are committed. With the five retries available, we found the energy consumed by the ultrasonic sensor under best and worst cases to be 0.04J and 0.29J respectively. The LoRaWAN radios energy consumption varied depending on the configured data rate and transmit power. Lower data rates consume more energy due to longer time on air, and high transmit power settings (from 17dBm to 36dBm) consume the maximum amount of energy for a given data rate – it saturates due to limitations of our radio. The best case for our radio's energy consumption is with the highest data rate and lowest transmit power (0.01J), whereas worst case is lowest data rate and highest transmit power (0.43J).

Once we calculate the energy consumed per packet, the total daily energy consumption of a PDU can be obtained by multiplying this value by the maximum number of packets sent per day. As a result, battery lifetime (in days) can be computed by dividing the usable energy available in the battery by the total daily energy consumed. Fig. 5 represents the estimated battery life for each combination of LoRaWAN parameters with the best and worst case scenarios (for ultrasonic detection) respectively. The trends show that the data rate has a greater impact on the battery life compared to the transmit power. Ultrasonic energy dominates for the worst case scenario leading to a compressed range (*i.e.,* 15 to 38 days in Fig. 5) of battery life. Our PDU's energy consumption is dominated by the ultrasonic sensor due to its retry property when no valid measurement is returned, suggesting high energy consumption

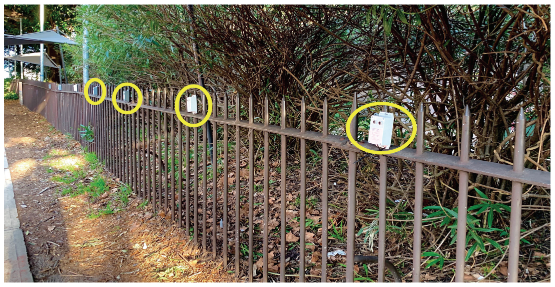


Fig. 6. Sensors mounted on the campus fence bordering the bus stop.

arises from an empty queue.

LoRaWAN Parameters Decision: For our deployment scenario, we have found that: an acceptable packet delivery ratio (more than 85% on average) can be achieved across all data rates when transmission power is configured to a value above 14 dBm (§III-C2); higher data rate setting can notably enhance battery life, for instance 208-day battery life can be achieved for DR5 setting while only 28-day life for DR0 setting when the transmission power is set at 14 dBm (§III-C3). Furthermore, based on the theoretical time-on-air calculation per data rate setting shown in Table III, faster data rate (*e.g.,* DR5) reduces air time for our sensors, which is limited to 30 seconds per-day according to TNN guidelines. In summary, data rate DR5 and transmission power of 14dBm are chosen because of acceptable packet delivery (reliability), longer battery life, and shorter time-on-air.

D. Queue inference algorithm

Data obtained from each PDU indicates the distance between the fence and the closest object in front of it. Since PDUs are deployed along the footpath which also serves as a pedestrian walkway, there is a chance that the PDUs detect people who are walking by or standing within proximity but are not part of the passenger queue. We therefore develop a queue inference algorithm (Algorithm 1) to convert the measurement data into queue length information while also eliminating any false positive or false negative readings.

The algorithm consists of three steps. The first step aims to address the time synchronization of the PDUs since data from each PDU is recorded at different time instances as explained in §III-B2. In order to synchronize the records, we map the received data into its corresponding time bin of t minute interval.

For each time interval, the second step decides whether a PDU detects the existence of a queue or not. If the fraction of positive measurements (detection within 2-3 m) over the time interval is greater than a pre-defined threshold percentage (*th*), we deem that the PDU detects a queue. Executing this step yields a vector of 10 binary values (1: queue detected and 0: not detected), corresponding to the ten PDUs. Since there is typically no substantial gap in the middle of the queue, the derived binary vector may contain false negative values which causes a disjoint sequence of 1's in the vector. The last step therefore aims to fix such errors if present by employing the concept of hamming distance to correct the measured vector (code) to one of the following valid codes:

Algorithm 1 Queue Inference Algorithm

```
Input: Sensor Data, valid codes
 1: Split sensor data into subsets of $t$ minute intervals
 2: for each subset do
 3:     Initialize array
 4:     for each sensor do
 5:         if % of distance between 2-3 m $> th$ then
 6:             sensor state = ON
 7:         else
 8:             sensor state = OFF
 9:         end if
10:         append sensor state to array
11:     end for
12:     for each v in valid codes do
13:         compute Hamming distance between array and v
14:     end for
15:     Set v with the lowest hamming distance as final array
16:     detected = sum of elements in array
17:     Queue length = detected * number of people
18: end for
Output: Queue length
```

$$valid\ codes \in \left\{ \begin{bmatrix}1\\0\\0\\0\\ \vdots\\0\end{bmatrix}, \begin{bmatrix}1\\1\\0\\0\\ \vdots\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\\0\\ \vdots\\0\end{bmatrix}, \ldots \begin{bmatrix}1\\1\\1\\1\\ \vdots\\1\end{bmatrix} \right\}$$

The selected valid code is then used to derive passenger queue length. Depending on the approximated number of people standing between any two adjacent PDUs, the queue length can be deduced by multiplying the number of detected bits in the vector by the number of people.

IV. System deployment

In this section, we first illustrate the deployment of our PDUs at an on-campus bus stop for queue length detection. We evaluate the efficacy of our algorithm against ground-truth data obtained from the field. The experiment shows that our algorithm can achieve good accuracy with mean absolute error value of less than 11.

A. Experimental Setup & Challenges

We deployed ten PDUs at the main bus stop of our university campus that serves as the first stop for a non-stop bus service to the city center. There are no intermediate stops, therefore all passengers head to the same destination. Since this is one of the busiest bus stops on the campus, waiting queue may grow to over 100 passengers at a time while the smallest bus offers a capacity of 68 passengers. Hence, it is possible for commuters to wait longer than the next service arrival time in order to board a bus. The sensors were affixed to the campus fence adjacent to the pedestrian footpath (about 3 meters in width) at a regular spacing of 6-meter, where each unit was installed 1.2-meter above ground. Fig. 6 shows our deployment of PDUs. We observed that typically passengers would form a queue along the footpath at a distance of about

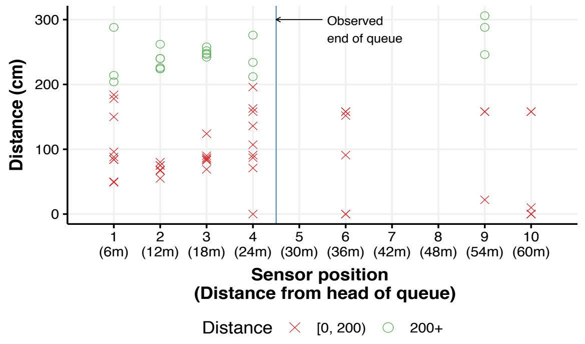


Fig. 7. Raw sensed distance measurements at each PDU position during a 2-minute interval snapshot (4.10pm-4.12pm on 24 Sep 2019). Green circles indicate distance where queue is expected and red crosses indicate detection that is not part of the queue.

2-3 meters from the PDUs. The arrangement of our PDUs in relative to the queue formation is shown earlier in Fig. 1.

We encountered several challenges during the deployment of our detection solution. The first challenge is associated with the installation requirement where each PDU requires a base object such as fence or a pole to attach it to. Therefore, the queue length measurement range in our experiment was limited to the length of the fence, since beyond the fence is an open area. Another challenge we observed is the inconsistency of queue formation during inclement weather such as rain, where passengers tend to crowd together under the shaded area. Since these areas are only available at certain parts of the queue, this creates a discontinuous queue formation which can affect the accuracy of our queue detection system. Furthermore, occasionally people who are not passengers may stand within the detection range, preventing the PDU from producing valid readings and reducing the accuracy of the queue length output. This issue can be partially solved by our queue inference algorithm mentioned in §III-D.

In order to evaluate the accuracy of our queue inference algorithm, we manually collect ground truth data by recording timestamps when each passenger arrives at the queue and boards the bus. This data is used to compute the actual queue length over the measurement period. This data was collected during peak hours (3-6 pm) on a weekday.

B. Sensed Data

Fig. 7 shows a 2 minute snapshot of collected raw measurements (infinite values as mentioned in §III-B2 are not shown here), where y-axis shows the distance (in cm) measured by each PDU and the x-axis denotes the position of each PDU with 1 and 10 denoting the head and tail of the queue, respectively. Measurements over 200cm are denoted as green while those that are lower are marked red. This is because the former are highly likely to be passengers in the queue, while the latter are likely passers-by. During the time period of this snapshot, between 40-58 queuers are observed, this is shown by a vertical blue line which indicates the approximate end of queue corresponded to the actual queue length. In the other word, the first 4 (or possibly 5) PDUs are expected to detect the queue while the rest are expected to report no detection. Examining the sensed data, we see that all PDUs at positions 1 to 4 have successfully detected the existence of the queue,

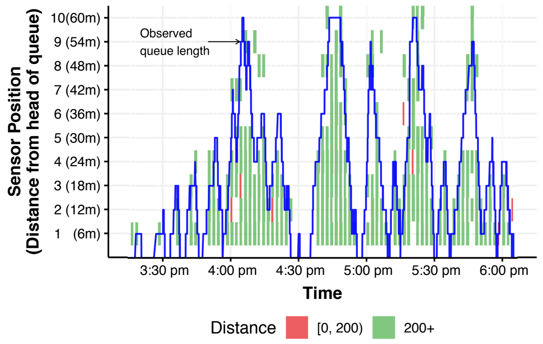


Fig. 8. Time-trace of measurements from PDUs overlaid by ground-truth queue information.

though some of the measured distances are not within the expected range (possibly from people walking pass by on the footpath). A false positive can also be seen at the position 9, where an object is detected within the range of the queue.

To further visualize the sensed data, we plot a time trace of PDU measurements and ground truth queue length in Fig. 8, where x-axis denotes time at 2 minute resolution and y-axis represents the position of each deployed PDU. Green cells indicate detection of the queue by the corresponding PDU at the corresponding time while red cells indicate detection of object that is not part of the queue. We can see that PDUs are able to fairly track the dynamics of the queue, evidenced by the true position detections (green fills that lies under the blue ground truth trajectory). Nonetheless, several false detections can be observed including false positives where a queue is detected when there is no queue formation (*e.g.,* position 8 at 4.14pm) and false negatives where queue detection is missed (*e.g.,* positions 5 and 6 between 5:44pm and 5:48pm). These false detection can be easily fixed using the algorithm we introduced in Section III-D.

C. Algorithm Evaluation and Tuning

From the aforementioned description of the queue inference algorithm in Section III-D, there are two tunable parameters, namely time bin t (in minutes) and detection threshold th (in percentage), that are needed to be decided upon. In order to select the optimum parameter values, we apply our algorithm to the sensor data collected during the 3 hour field experiment (where ground truth occupancy was obtained) and observe the impact of varying t and th. Root Mean Square Error (RMSE) is used as the performance metric. Fig. 9 shows RMSE (y-axis) as a function of detection threshold (x-axis). Different line graphs shown the impact of varying the time bin duration. Note that a time interval of 2 min (solid green line) achieves the lowest RMSE. Also, choosing the detection threshold at 0.2 gives an acceptable RMSE of 13.25 and MAE of 10.75. The output queue length calculated by our algorithm, using the optimal values of t and th, is visualized in Fig 10, where ground-truth is shown by dotted red lines and the calculated queue occupancy is shown by solid blue lines. We can see that our algorithm is able to closely track the real queue with low error. We note that our solution can only measure the queue up to the end of fence area (as noted earlier), hence the calculated queue length is capped at 100 people. Note that

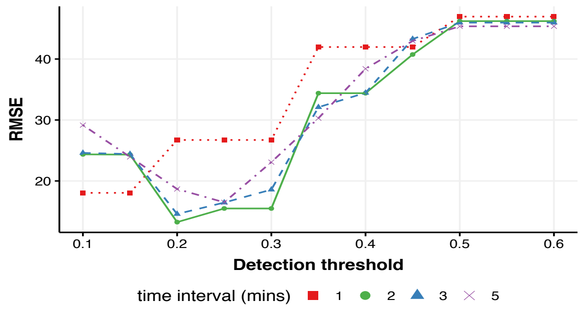


Fig. 9. Comparison of RMSE across different tuning parmeters.

this is an artifact of our deployment and does not impact the algorithm accuracy.

D. Deployment results

We deployed our PDUs at the bus stop for a period of five days, starting from 3pm on Monday (23 Sep 2019) until the end of Friday (27 Sep 2019). The tuned algorithm described in §IV-C is applied to the collected data (from our deployment) to infer the temporal profile of queue length. We compute the arrival rate (every minute) of passengers by integrating the queue length (deduced from our sensors data) with a publicly available (and real-time) data of bus schedules in the form of General Transit Feed Specification (GTFS) [25]. This GTFS data reports the actual arrival/departure time of buses to/from transit stops as well as buses type and model, allowing us to estimate the capacity of each dispatched bus. The computed rate of passengers arrival is highly fluctuating. Therefore, we apply moving average (15-minute window) to smooth the arrival rate data. Fig. 11 illustrates the smoothed rate of passengers arrival per minute (black line) across the five days of our deployment. The bus arrival times are depicted in the same figure as vertical red lines.

It is clearly seen in Fig. 11 that bus arrivals are consistent across all weekdays (static scheduling) despite variations in the arrival pattern of passengers. We observe that the passenger arrival pattern varies across different days especially on Friday where the peak occurs at 4:30pm as opposed to 5pm for other weekdays. However, buses are statically scheduled to be dispatched more frequently around the 5pm mark, suggesting a mismatch between supply and demand and thus inefficient use of resources. This observation suggests for a more optimal bus scheduling to be considered, so that the passengers experience is improved by reducing their waiting time at the bus stop but without adding extra bus services.

V. Optimizing Bus Schedules Based on Demand

We saw in §IV-D that existing static bus schedules are not particularly efficient. In this section, we develop an optimization model that schedules the arrival of buses based on the actual passenger demand in order to minimize the total wait time of passengers at the stop. We formulate the problem with an assumption that the passengers demand is given and known. Note that the future demand may be predicted by way of modeling the pattern of historical data, but predicting the demand is beyond the scope of this paper given the limited data collected from our deployment.

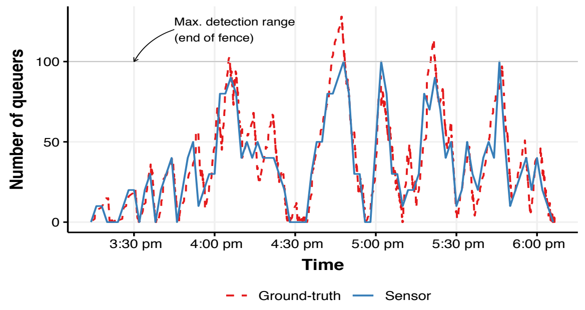


Fig. 10. Real-time queue length inferred from sensor data vs. ground-truth.

A. Optimization Formulation

Several research works [26], [27] have developed dynamic bus scheduling schemes to improve the transit experience of passengers, with the common objective being minimizing passenger wait times. However, many existing models estimate passenger wait time without considering the limited capacity of buses [28], [29]. During peak hours particularly, certain passengers (depending on the queue length and their location in the queue) may need to wait for more than two buses before they get on board. Our formulation, therefore, aims to incorporate the additional wait time of those passengers who may get left behind due to overloaded buses.

For our optimization problem, let there be B buses available for use in a day. The arrival time and seating capacity of bus i are respectively denoted by d_i and C_i where $1 \leq i \leq B$. Passengers demand, captured by their rate of arrival to the stop as shown in Fig. 11, is denoted by $\lambda(t)$. The total daily wait time of passengers is obtained by adding the following components:

(a) Time spent (within a day) by those passengers who board their first arriving bus since they joined the queue. We borrow from [30] the model of this factor of wait time for B buses during a day, which is given by:

$$W^{first} = \sum_{i=1}^{B} \int_{d_{i-1}}^{d_i} (d_i - t)\lambda(t)\ dt \tag{1}$$

(b) Time spent by waiting passengers who missed to board previous bus(es) due to capacity limit. For each arriving bus i, N_i^{left} denotes the number of leftover passengers that arrived before and missed the previous bus $i-1$. The total wait time of leftover passengers can be formulated as:

$$W^{left} = \sum_{i=1}^{B} N_i^{left}(d_i - d_{i-1}) \tag{2}$$

Given $N_1^{left} = 0$, the number of leftover passengers can be calculated recursively by:

$$N_i^{left} = N_{i-1}^{left} + \int_{d_{i-2}}^{d_{i-1}} \lambda(t)\ dt - C_{i-1} \tag{3}$$

The objective of our optimization formulation is to minimize the total daily wait time of passengers:

$$\min \quad W^{first} + W^{left} \tag{4}$$

In our optimization model, we assume an ordered set of buses, each with certain capacity. Our decision variable is the

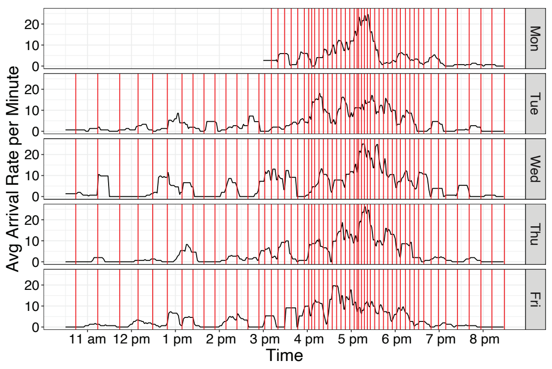


Fig. 11. Static schedule of buses dispatch time.

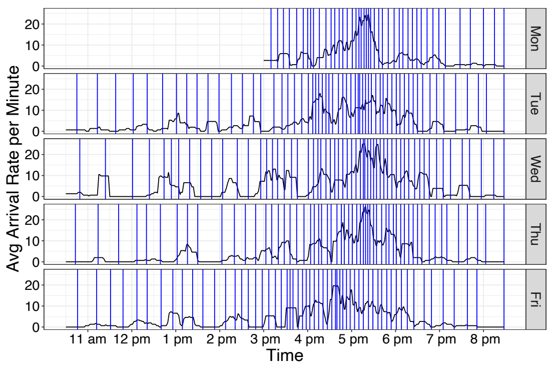


Fig. 12. Optimal schedule of buses dispatch time.

arrival time of these buses. There are a couple of constraints to be considered:

(a) Time interval between consecutive buses (headway) need to be less than an hour (H_{max}) in order to maintain a minimum frequency of bus services (as observed in existing time-tables). We also note that the headway cannot be less than a minute (H_{min}), since the resolution of our computation is every minute (minimum) and buses are not scheduled simultaneously. Therefore, our first constraint is given by:

$$H_{min} \leqslant d_i - d_{i-1} \leqslant H_{max} \tag{5}$$

(b) The last bus (of a given day) needs to be scheduled at the end of the daily optimization period (t_e), consistent with the current bus schedules (say, 8:30pm).

$$d_B = t_e \tag{6}$$

B. Algorithms for Solving Optimization Problem

The scheduling problem we formulated in §V-A is a nonlinear and non-convex combinatorial optimization problem [31], which makes it difficult to obtain the global optimal solutions. Therefore, heuristic algorithms and non-heuristic algorithms such as Genetic algorithm are commonly employed to solve such problems. In particular, GA is one of the most well-studied algorithm used in the field of mass transit optimization [26], [30] and has proven to be effective in solving combinatorial optimization problems [32]. In this paper, we adopt GA and Hill Climbing heuristic search to solve our scheduling problem and compare their performance.

GA is inspired by the process of natural evolution. The algorithm selects optimal solutions that yield the best fitness (based on the objective function) for reproduction in the next generation, hoping to produce better offspring (solutions with better fitness scores than the parents). We use the optimization objective in Eq. 4 (minimizing the total daily wait time) as the fitness function of our genetic algorithm. For our chromosome design, we let genes within a chromosome represent headways of a fixed ordered set of buses. Therefore, each chromosome is expressed as $[H_1, H_2, ..., H_{B-1}]$ where H_i (a gene) denotes the headway between bus $i-1$ and bus i. Note that the last available bus on a day needs to be scheduled at the end of daily window according to the second constraint in Eq. 6, Hence there are total number of $B-1$ genes for the chromosome. Note that we use bus headways instead of dispatching time in order to narrow down the search space.

For cross-over operation where two parent solutions exchange genes, we adopt a local arithmetic cross-over [33]. This operator takes weighted sum of the two parents to create a new gene where weights are randomly selected for each gene. This method is chosen to increase the diversity in generated offspring. In other words, the objective is to create new genes which are not present in parents [34]. Lastly, for mutation operation, we adopt a uniform random mutation (for simplicity) where genes are selected randomly from a uniform distribution over a range $[H_{min}, H_{max}]$, the lower and upper bounds in Eq. 5. We use the GA library in R [35] to implement our genetic algorithm for solving the optimization problem. The default probability values of 0.8 and 0.2 are used respectively for cross-over and mutation operations. We run the algorithm for 1000 generations, each consisting of a population of 50 individuals. The search halts when the best solution score remains unchanged for 100 generations.

On the other hand, Hill Climbing is a heuristic search that uses the greedy approach to progressively select the best neighboring solutions that optimize the cost function. The algorithm runs until no further improvement can be yielded from the neighbors. Using the previously defined set of head ways as our decision variables, we define neighboring nodes of the current solution as solutions with 1 minute addition or subtraction to any one of the headway variable $[H_i]$. Similarly, the optimization objective in Eq. 4 is used as an evaluating function in Hill Climbing.

C. Optimization results

We use the passenger demand data from our deployment (described in §IV-D) as input to our optimization model. The optimal bus dispatching times using GA (for each day) are illustrated in Fig. 12. Comparing the two schedules, *i.e.,* static (in Fig. 11) with optimal (in Fig. 12), we observe that they match to a great extent from Monday to Thursday, displaying a higher frequency pattern during the peak time (5:00pm-5:30pm). Also, some minor variations can be observed, specially during times closer to the peak. For example, an additional peak period is seen between 4pm and 4:30pm on Tuesday, prompting 40% more buses to be scheduled for that time compared to other weekdays. The most notable difference between the two schedules is observed on Friday, where buses are scheduled more frequently by our optimal policy between 4.30pm and 5pm (as shown in Fig. 12) to fulfill the actual peak demand. To better visualize the shift in the peak demand,

TABLE IV
AVERAGE WAIT TIME (MIN) OF PASSENGERS: STATIC VERSUS OPTIMAL (DYNAMIC) SCHEDULING.

	All day					Peak hour				
date	static	optimal (HC)	optimal (GA)	reduction (HC)	reduction (GA)	static	optimal (HC)	optimal (GA)	reduction (HC)	reduction (GA)
Mon	4.43	3.52	3.22	20.06%	27.21%	4.52	3.34	3.16	26.10%	30.12%
Tue	6.55	4.62	5.07	29.49%	22.72%	4.75	3.20	3.90	32.68%	17.90%
Wed	10.22	6.88	6.78	32.64%	33.66%	6.57	3.92	5.39	40.34%	17.91%
Thu	6.79	5.06	6.04	25.50%	10.98%	6.84	4.84	5.92	29.18%	13.43%
Fri	7.69	5.44	4.39	29.33%	42.93%	6.80	4.43	3.38	34.82%	50.23%

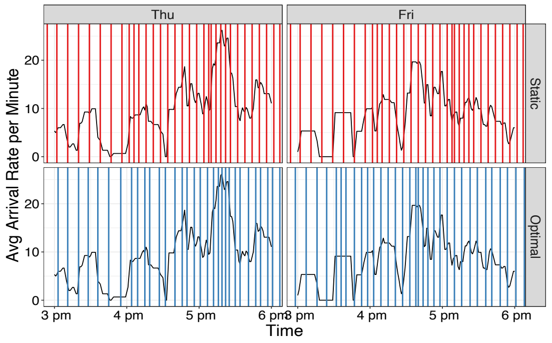


Fig. 13. Comparing static schedule with optimal schedule of buses for two representative days, Thursday and Friday.

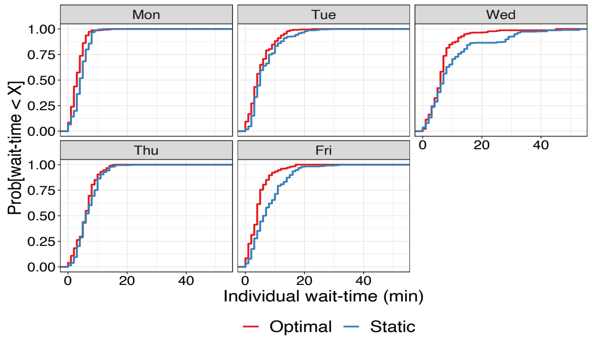


Fig. 14. CDF of passengers wait-time: Optimal versus Static schedules, from 5-day trial period.

we illustrate in Fig. 13 a comparison of static schedule (top) with the optimal schedule (bottom) for Thursday (left) and Friday (right) during 3-6pm. It is clearly seen that static and optimal schedules highly correlate on Thursday with 7 buses scheduled during the peak (5pm-5:30pm). On the other hand, on Friday, the optimal schedule has two additional services during the revised peak period (4:30pm-5pm) but two fewer buses during 5pm-5:30pm.

We compute average wait-time per passenger to compare the performance of static and optimal bus schedules obtained from genetic algorithm (GA) and hill climbing algorithm (HC). Table IV summarizes the average wait-time across each day of our 5-day trial. As expected, the optimal approaches yield lower wait times compared to the initial static bus schedule across all 5 days. Optimal bus schedule obtained from GA yields the highest reduction in wait time across three days (Mon, Wed, and Fri) with the largest improvement of 42.93% observed on Friday. Focusing on Friday, the average wait-time from the static bus schedule is 7.69 minutes which can be reduced to 4.39 minutes and 5.44 with the optimal bus schedule produced from HC and GA respectively. This relatively large gap highlights the inefficiency of existing static bus scheduling which falls short in accommodating the shift in peak demand. In contrast, lower reductions are observed for the rest of weekdays due to less variation from the expected demand wherein a minor peak is expected at 4 pm and a major peak at 5 pm according to the static schedule. On average, both optimization algorithms provide a relatively similar wait time reduction of 25.7% compared to static scheduling averaged over the 5 days. One of the reasons a heuristic algorithm like hill climbing performs relatively well compared to genetic algorithm, despite being a local search algorithm (which is prone to produce a local optima outcome), is because the initial solution provided for the algorithm is the existing static schedule that has been designed with some planning by the transport authority.

Lastly, we plot in Fig. 14 the cumulative distribution function (CDF) of the wait-time of individual passengers. It can be seen that optimal and static curves exhibit very minor differences on Monday, Tuesday and Thursday, with the former slightly shifted to the left (slightly lower wait times overall). In contrast, we observe a more noticeable difference on Wednesday and Friday. On Friday, the static schedule requires more than 80% of the passengers to wait up to 12 minutes. Instead, the optimal schedule reduces the wait time to 7 minutes. On Friday, the longest wait time is 32 minutes with the static schedule, while this reduces to 17 minutes for the optimal scheme. The difference between the two curves on Wednesday can be attributed to the unexpected higher demand during the non-peak time period 11:30am-1:00pm (see Fig. 11). The optimal scheme adapts to this variation by scheduling additional buses during this time period (see Fig. 12).

The experiments on real world data demonstrate that dynamic bus scheduling based on actual demand can reduce passengers' wait time by up to ≈43%. This shows that real world passenger demand information can be used to inform bus transit scheduling decisions that can significantly improve passenger experience at transit stops.

VI. CONCLUSION

In this paper, we proposed an end-to-end system for queue length measurement using ultrasonic sensors and LoRaWAN for data communications. Our proposed solution can be used in a variety of outdoor settings (e.g. stadiums, airports) where an orderly queue is formed. We first described the implementation of our people sensing devices that are battery-operated, able to communicate wirelessly, and have water-proof exterior, allowing the solution to be easily deployed in an outdoor environment. Next, we developed an algorithm to infer queue length information from the collected data. Our algorithm achieved a decent accuracy with MAE of 10.7 people for a queue size of up to 100 passengers. Finally, we showed how our system and method for estimating the queue length in real-time can be used to optimize bus scheduling, revealing a potential wait time reduction of 42.93% over a day by adopting a demand-driven bus scheduling.

REFERENCES

[1] R. Daniels *et al.*, "The paradox of public transport peak spreading: Universities and travel demand management," *International Journal of Sustainable Transportation*, vol. 7, no. 2, pp. 143–165, 2013.

[2] F. Lyu *et al.*, "Intelligent context-aware communication paradigm design for IoVs based on data analytics," *IEEE Network*, vol. 32, no. 6, pp. 74–82, 2018.

[3] T. Okoshi *et al.*, "Queuevadis: Queuing analytics using smartphones," in *Proc. ACM IPSN*. Seattle, USA: ACM, 2015, pp. 214–225.

[4] Q. Li *et al.*, "Queuesense: Collaborative recognition of queuing on mobile phones," in *Proc. IEEE SECON*. Singapore, Singapore: IEEE, 2014, pp. 230–238.

[5] J. Weppner *et al.*, "Bluetooth based collaborative crowd density estimation with mobile phones," in *Proc. IEEE PerCom*. San Diego, CA, USA: IEEE, 2013, pp. 193–200.

[6] Y. Wang *et al.*, "Tracking human queues using single-point signal monitoring," in *Proc. ACM MobiSys*. New Hampshire, USA: ACM, 2014, pp. 42–54.

[7] D. Aubert, "Passengers queue length measurement," in *Proc. IEEE ICIAP*. Venice, Italy: IEEE, 1999, pp. 1132–1135.

[8] H. Shu *et al.*, "Queuing time prediction using wifi positioning data in an indoor scenario," *Sensors*, vol. 16, no. 11, p. 1958, 2016.

[9] L. Schauer *et al.*, "Estimating crowd densities and pedestrian flows using wi-fi and bluetooth," in *Proc. ACM Mobiquitous*. London, UK: ACM, 2014, pp. 171–177.

[10] M. Elhamshary *et al.*, "Crowdmeter: Congestion level estimation in railway stations using smartphones," in *Proc. IEEE PerCom*. Athens, Greece: IEEE, 2018, pp. 1–12.

[11] X. Tang *et al.*, "Indoor crowd density estimation through mobile smartphone wi-fi probes," *IEEE transactions on systems, man, and cybernetics: systems*, 2018.

[12] J. Goncalves *et al.*, "Crowdsourcing queue estimations in situ," in *Proc. ACM CSCW*, San Francisco, California, USA, 2016, pp. 1040–1051.

[13] H. Ma, C. Zeng, and C. X. Ling, "A reliable people counting system via multiple cameras," *ACM TIST*, vol. 3, no. 2, pp. 1–22, 2012.

[14] O. Masoud *et al.*, "A novel method for tracking and counting pedestrians in real-time using a single camera," *IEEE transactions on vehicular technology*, vol. 50, no. 5, pp. 1267–1278, 2001.

[15] J. Ma *et al.*, "Atrous convolutions spatial pyramid network for crowd counting and density estimation," *Neurocomputing*, vol. 350, pp. 91–101, 2019.

[16] T. Kulshrestha *et al.*, "Real-time crowd monitoring using seamless indoor-outdoor localization," *IEEE TMC*, vol. 19, no. 3, pp. 664–679, 2019.

[17] A. Musa *et al.*, "Tracking unmodified smartphones using wi-fi monitors," in *Proc. ACM SenSys*. Toronto, Canada: ACM, 2012, pp. 281–294.

[18] S. E. Al, "Lorawan 1.0.3 specification," https://lora-alliance.org/sites/default/files/2018-07/lorawan1.0.3.pdf, 2018.

[19] "Lorawan security, full end–to–end encryption for iot application providers," https://lora-alliance.org/sites/default/files/2019-05/lorawan_security_whitepaper.pdf, 2017.

[20] "Lorawan frequencies overview," https://www.thethingsnetwork.org/docs/lorawan/frequency-plans.html, 2019, accessed: 20/10/2019.

[21] Emil, "Making a better hc-sr04 echo locator," https://uglyduck.vajn.icu/ep/archive/2014/01/Making_a_better_HC_SR04_Echo_Locator.html, 2014, accessed: 19/05/2019.

[22] M. Cattani *et al.*, "An experimental evaluation of the reliability of lora long-range low-power wireless communication," *Journal of Sensor and Actuator Networks*, vol. 6, no. 2, p. 7, 2017.

[23] R. Sanchez-Iborra *et al.*, "Performance evaluation of lora considering scenario conditions," *Sensors*, vol. 18, no. 3, p. 772, 2018.

[24] F. Corporation, "Technical information: alkaline manganese batteries aa size / lr6g07 premium," https://media.digikey.com/pdf/Data\%20Sheets/FDK/LR6G07.pdf, accessed: 15/10/2019.

[25] "Tfnsw open data," https://opendata.transport.nsw.gov.au, 2019, accessed: 23/10/2019.

[26] Z. Huang *et al.*, "A novel bus-dispatching model based on passenger flow and arrival time prediction," *IEEE Access*, vol. 7, pp. 106 453–106 465, 2019.

[27] X. Feng *et al.*, "Design of intelligent bus positioning based on internet of things for smart campus," *IEEE Access*, vol. 6, pp. 60 005–60 015, 2018.

[28] B. Chen, R. Bai, J. Li, Y. Liu, N. Xue, and J. Ren, "A multiobjective single bus corridor scheduling using machine learning-based predictive models," *International Journal of Production Research*, pp. 1–16, 2020.

[29] K. Gkiotsalitis and R. Kumar, "Bus operations scheduling subject to resource constraints using evolutionary optimization," in *Informatics*, vol. 5, no. 1. Multidisciplinary Digital Publishing Institute, 2018, p. 9.

[30] X. Luo *et al.*, "Dynamic bus dispatching using multiple types of real-time information," *Transportmetrica B: Transport Dynamics*, 2018.

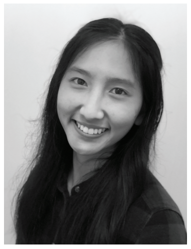
[31] B. H. Korte *et al.*, *Combinatorial optimization*. Springer, 2011, vol. 1.

[32] S. Khuri *et al.*, "An evolutionary approach to combinatorial optimization problems." in *ACM Conference on Computer Science*, 1994, pp. 66–73.

[33] D. Dumitrescu *et al.*, *Evolutionary computation*. CRC press, 2000.

[34] G. Pavai and T. V. Geetha, "A survey on crossover operators," *ACM Comput. Surv.*, vol. 49, no. 4, Dec. 2016.

[35] L. Scrucca *et al.*, "GA: a package for genetic algorithms in r," *Journal of Statistical Software*, vol. 53, no. 4, pp. 1–37, 2013.



Thanchanok Sutjarittham received her B.Eng. degree in Electrical Engineering and Telecommunications from UNSW Sydney in 2016. She is currently pursuing her Ph.D. in Electrical Engineering at UNSW. Her primary research interests include Internet of Things, sensor data analytics, and applied machine learning.

Hassan Habibi Gharakheili received his B.Sc. and M.Sc. degrees of Electrical Engineering from the Sharif University of Technology in Tehran, Iran in 2001 and 2004 respectively, and his Ph.D. in Electrical Engineering and Telecommunications from UNSW in Sydney, Australia in 2015. He is currently a Senior Lecturer at UNSW Sydney. His current research interests include programmable networks, learning-based networked systems, and data analytics in computer systems.

Salil S. Kanhere received his Ph.D. from Drexel University. He is a Professor in the School of Computer Science and Engineering at UNSW. His research interests include the Internet of Things, pervasive computing, blockchain, crowdsourcing, sensor networks, and security. He has published 200 peer-reviewed articles and delivered over 30 tutorials and keynote talks. He is a Senior Member of ACM. He is a recipient of the Humboldt Research Fellowship.

Vijay Sivaraman received his B. Tech. from the Indian Institute of Technology in Delhi, India, in 1994, his M.S. from North Carolina State University in 1996, and his Ph.D. from the University of California at Los Angeles in 2000. He has worked at Bell-Labs as a student Fellow, in a silicon valley start-up manufacturing optical switch-routers, and as a Senior Research Engineer at the CSIRO in Australia. He is now a Professor at the University of New South Wales in Sydney, Australia. His research interests include Software Defined Networking, network architectures, and cyber-security particularly for IoT networks.