2019

Optimal Gateway Placement in Low-cost Smart Cities

Oluwashina Madamori
University of Kentucky, shina.ct@gmail.com
Digital Object Identifier: https://doi.org/10.13023/etd.2019.472

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Oluwashina Madamori, Student

Dr. Corey E. Baker, Major Professor

Dr. Miroslaw Truszczynski, Director of Graduate Studies

Optimal Gateway Placement in Low-cost Smart Cities

THESIS

A thesis submitted in partial
fulfillment of the requirements for
the degree of Master of Science in
the College of Engineering at the
University of Kentucky

By
Oluwashina Madamori
Lexington, Kentucky

Director: Dr. Corey E. Baker, Assistant Professor of Computer Science
Lexington, Kentucky 2019

ABSTRACT OF THESIS

Optimal Gateway Placement in Low-cost Smart Cities

Rapid urbanization burdens city infrastructure and creates the need for local governments to maximize the usage of resources to serve its citizens. Smart city projects aim to alleviate the urbanization problem by deploying a vast amount of Internet-of-things (IoT) devices to monitor and manage environmental conditions and infrastructure. However, smart city projects can be extremely expensive to deploy and manage partly due to the cost of providing Internet connectivity via 5G or WiFi to IoT devices. This thesis proposes the use of delay tolerant networks (DTNs) as a backbone for smart city communication; enabling developing communities to become smart cities at a fraction of the cost. A model is introduced to aid policy makers in designing and evaluating the expected performance of such networks and results are presented based on a public transit network data-set from Chapel Hill, North Carolina and Louisville, Kentucky. We also demonstrate that the performance of our network can be optimized using algorithms associated on set-cover and Influence maximization problems. Several optimization algorithms are then developed to facilitate the effective placement of gateways within the network model and these algorithms are shown to outperform traditional centrality-based algorithms in terms of cost-efficiency and network performance. Finally, other innovative ways of improving network performance in a low-cost smart city is discussed.

KEYWORDS: smart cities, delay tolerant network, iot, placement optimization

Author's signature: Oluwashina Madamori

Date: December 20, 2019

Optimal Gateway Placement in Low-cost Smart Cities

By

Oluwashina Madamori

Director of Thesis: Corey E. Baker

Director of Graduate Studies: Miroslaw Truszczynski

Date: December 20, 2019

To Jesus Christ, for seeing me through this journey.

ACKNOWLEDGMENTS

I would first and foremost like to thank Dr. Corey Baker for being my advisor and giving me the opportunity to earn this degree. The hours spent brainstorming with Dr. Baker is what has resulted in this work. I would also like to thank Dr. Mirosław Truszczynski and Dr. Gregory D. Erhardt for being a part of my committee and providing valuable guidance during this work. My fellow lab mates have also been a great support to me and have made my time at the University of Kentucky and awesome experience. Thank you Esther Max-Onakpoya, Alyssa Donawa, Xava Grooms, Feyi Afolabi, and Abdulaziz Alhomaidhi. I also want to thank siblings Lola Adeyemi, Dayo Madmaori, Ronke Achi, and Wole Madamori. You all have been a constant blessing to me. Finally, I would like to thank Christian Enyoghasi for being a true brother in Christ. Meeting you here in Kentucky is one of the main highlights of my time in Kentucky.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

1.1 Background

Urbanization, which refers to the migration of people from rural areas to urban communities, has increased rapidly in the past few decades placing an extra burden on the infrastructure in urban areas. An urban area typically refers to a community where the population density exceeds 1,000 persons per square mile [1]. Currently, 55% of the world's population live in urban communities, and this percentage is projected to rise to 68% by 2050, with approximated 6.7 Billion people living in urban areas [1]. Policy makers are thus tasked with finding more effective methods for managing limited public resources to meet the needs of its citizens.

Smart city initiatives have been proposed as a way to alleviate the challenges arising from urbanization. Smart city projects involve the deployment of a vast number of Internet-of-things (IoT) devices across communities to monitor and manage environmental conditions and public infrastructure. These IoT devices, which includes devices like weather sensors, traffic monitors and parking meters/monitors generate a significant amount of data. The data generated these sensors typically needs to be collected and forwarded to remote servers located in the cloud for processing. The need for data retrieval in smart cities has thus necessitated research into the design of communication network architecture for smart cities that can forward data generated by sensors to the cloud in an efficient and reliable manner. Transforming a city into a smart city has also proven to be very expensive. For example, cities such as San Diego, New Orleans, London, and Songdo have either proposed or invested in smart city projects that cost between $30 Million and $40 Billion [2–5]. The high cost has made many city governments reluctant to invest in smart city projects [6, 7], especially when there aren't very strong guarantees about the sustainability and impact of such projects. Therefore, finding ways to significantly reduce the cost of transforming traditional cities into smart cities is critical.

Currently, most smart city solutions that make use of IoT devices (many of which may be outdoors) rely on cellular network connectivity for collecting the data generated by these sensors [8]. The reason for the popularity of cellular networks is that it guarantees data delivery with relatively little network latency. The main disadvantage of cellular connectivity however is the additional operating cost incurred as a result of the data subscription costs associated with each sensor [8]. This cost becomes significant when dealing with hundreds or even thousands of sensors to be deployed within a city. Also, since such sensors would be connecting to the same cellular base stations as those being used by mobile phones of citizens, these IoT devices can quickly lead to network congestion in cities and poor user experiences. Hence, other alternative and cost-effective methods of data retrieval in smart cities need to been explored.

In this thesis, we propose a low-cost network architecture that can serve as an alternative to cellular connectivity for data retrieval in smart cities. The architecture leverages the pre-existing mobility of people and vehicles in a city to create an opportunistic communication network for delivering IoT data to the cloud. Our network does not require subscription fees hence reducing the cost of maintaining a smart city.

1.2 Thesis Overview

Chapter 2 provides a detailed description of our proposed network architecture for low-cost smart cities and a mathematical model for estimating the upper bound for delivery latency. In Chapter 3, we develop a simulator for our network that uses real-world public transportation timetables to help in evaluating the various performance metrics. In Chapter 4, we consider several optimization problems present in our network architecture and we propose several of algorithms for solving these problems. Finally, in chapter 5 we discuss future directions of this work including classifying sensor data into categories, exploring pedestrian incentivization strategies, and integrating pedestrian mobility into network simulations.

Figure 1.1: An urban area

Chapter 2 Using Delay Tolerant Networks as a Backbone for Low-cost Smart Cities

2.1 Introduction

The term "delay-tolerant networking" or "delay-tolerant network", abbreviated as DTN, refers to a type of network architecture that relies on a store-carry-forward paradigm to transmit data packets from a source node to a destination node with the aid of forwarding/intermediary nodes in the network. A DTN typically consist of some mobile nodes, where communication between nodes is subject to frequent disruption and disconnection resulting in long delays and intermittent connectivity [9]. The first major research work to propose DTNs was by Kevin Fall in 2003 [10], which has motivated subsequent research in the field of DTN including applications in wild life management, agriculture, and vehicular communication [11–13].

In this chapter, we propose a low-cost DTN-based network architecture that could potentially serve as an alternative to cellular connectivity for IoT devices in smart cities. Our network architecture leverages the pre-existing mobility of pedestrians and public vehicles to create opportunistic communication networks for delivering IoT data to the cloud. The goal is to provide city planners with a viable and cheaper alternative to Internet connectivity for IoT devices in smart cities, by eliminating or reducing the associated Internet subscription fees.

However, previous research [14] has shown that the two main limitations of DTNs compared to cellular networks are: (i) low and unpredictable probability of delivery, and (ii) high and unpredictable network latency. Therefore, in order for a DTN-based architecture to serve as a backbone for smart city communication, the aforementioned limitations need to be addressed. Addressing these limitations would help city developers ensure that such a network meets quality of service (QoS) requirements [15].

To demonstrate that DTNs can serve as a backbone for smart city communication, we develop a mathematical model to characterize the entities participating in our

network and how these entities interact with one another. The model can also help policy makers understand how changes in their transit system with respect to transit scheduling, gateway deployment, sensor placement, ridership levels, and number of participating citizens can impact performance in their low-cost smart city. The rest of the chapter is structured as follows: Section 2.2 describes the tools used in this work to capture data-sets from real-world public transit systems. Section 2.3 describes the proposed model. Section 2.4 provides preliminary results. Finally, Section 2.5 describes the next steps in this work and the need for incentivization strategies that can be used to maximize delivery probability and minimize latency.

2.2 Understanding City Public Transportation

Many cities currently provide highly reliable information on the Internet about their public transit network. Popular sources include *OpenMobilityOrg* which contains General Transit Feed Specification (GTFS) data [16] and *NextBus* [17]. These data includes static information that specify bus routes, stops, and operating schedules. In addition, OpenMobilityOrg and NextBus provide real-time information about GPS location and expected arrival times of buses within various city transit networks. To validate the model presented in this work, the *NextBus* API was used to obtain data about the bus routes in the city of Chapel Hill, North Carolina. The data retreived contained descriptions of bus routes, trip directions, and stops per route for all buses in operation during the month of July, 2018. The characteristics of Chapel Hill public transportation are listed in Table 2.1.

Table 2.1: Chapel Hill Public Bus Characteristics

Numbers of routes	32
Distance of routes	$\mu = 15.70$ km, $\sigma^2 = 7.32$ km
Stops per route	$\mu = 72.6$, $\sigma^2 = 52.44$
Buses per route	$\mu = 1.79$, $\sigma^2 = 1.00$

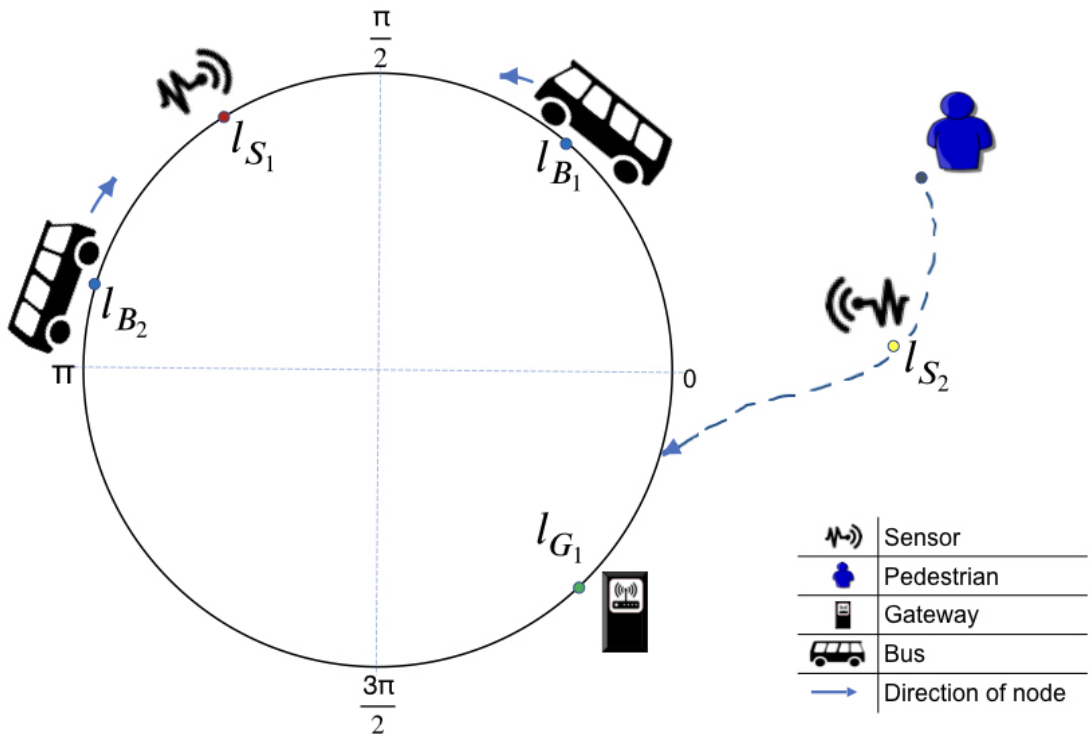


Figure 2.1: Opportunistic communication for low-cost smart cities

2.3 Modeling Low-cost Smart Cities

A smart city consists of many entities. The entities along with assumptions about their characteristics are as follows:

- **Bus routes** - are taken by buses. Due to the recurring nature of buses visiting the same stop, can be represented by circles, each with a circumference corresponding to the total distance for that route. In addition, the locations of all buses, bus stops, gateways, and on-route sensors are restricted to points on a circle representing a bus route.

- **Buses** - move along predefined routes on a fixed schedule, hence, the specific geographic position of any bus can be calculated at any time. Buses are also expected to move at a constant average velocity (including stops) throughout a route trip, thus assigning every bus a fixed round-trip time. Buses have buffer/queue sizes of infinity and do not use any type of drop-policy for stored data.

- **Pedestrians** - people who sign-up to join the low-cost smart city using their smartphones. Data packets from sensors are forwarded to their smartphones using Bluetooth 5 when they come within transmission range of a sensor. The data is stored on the phone until the pedestrian comes within transmission range of a bus or gateway and forwards the data to corresponding node.

- **Sensors** - broadly classified into two categories based on their location: on-route and off-route sensors. **On-route sensors** are those located at stop-lights, street-lights, and bus stops; and are within transmission range of buses travelling on a bus route. Any sensor that is not an "on-route sensor" is classified as an **off-route** sensor. Off-route sensors have an additional parameter associated with them called the "pedestrian arrival rate," which specifies the likelihood of a participating pedestrian coming in contact with the sensor. Although pedestrians are not essential for retrieving on-route sensor data, they are vital in the retrieval of off-route sensor data. In addition, individual data packets generated by sensors are considered to be small, allowing sensor buffer/queue size to be infinity and no drop policy needed.

- **Gateways** - stationary, always-on, always-connected devices that forward data directly to the cloud. Not all bus stops are gateways, but rather gateways are placed at select bus stops. Gateways act as the final destination for all data generated by sensors.

Assumptions

The model makes several assumptions about the DTN within a smart city. Each bus route has at least one gateway, and buses operate round the clock everyday. Furthermore, a connection is always established between the sensor and bus (or pedestrian) whenever a bus (or pedestrian) passes a sensor, and all data at the sensor is transferred to the bus (or pedestrian). In addition, there is no routing from pedestrian to pedestrian or from bus to bus.

As the model is further developed some of the aforementioned assumptions will be relaxed. Figure 2.1 summarizes the interactions between entities in the model.

Estimating delivery latency

On-route sensors: The expected delivery latency $T_{D\text{-on}}$ for any data produced at a sensor at an arbitrary time t is represented by:

$$T_{D\text{-on}} = T_{SB} + T_{BG} \tag{2.1}$$

$$T_{SB} = \frac{\textit{distance from bus location to sensor at } t}{\textit{average bus velocity}} \tag{2.2}$$

$$T_{BG} = \frac{\textit{distance from bus location to gateway at } t + T_{SB}}{\textit{average bus velocity}} \tag{2.3}$$

and the upper bound $\max(T_{D\text{-on}})$ for delivery latency will be:

$$\max(T_{D\text{-on}}) \approx \frac{2 \times \textit{circumference of route}}{\textit{average bus velocity}} \tag{2.4}$$

Off-route sensors: For off-route sensors, there are two additional stages between when the sensor data is generated and when it is delivered at the gateway.

$$T_{D\text{-off}} = E\big[T_{SP}\big] + E\big[T_{PB}\big] + T_{SB} + T_{BG} \tag{2.5}$$

Where $E\big[T_{SP}\big]$ is the expected time it takes a participating pedestrian to encounter the sensor after the data is generated in time t_0, and $E\big[T_{PB}\big]$ is the expected duration it takes for that pedestrian to board (or encounter) a bus. Also, the upper bound $\max(T_{D\text{-off}})$ for delivery latency will be:

$$\max(T_{D\text{-off}}) = \max(T_{SP}) + \max(T_{PB}) + \max(T_{D\text{-on}}) \tag{2.6}$$

2.4 Preliminary Results

Using the tools described in Section 2.2 to generate a data-set for Chapel Hill, North Carolina along with a 2016 Chapel Hill transit passenger survey [18], a light-weight

simulator[1] was developed to find the expected performance of transforming Chapel Hill into a low-cost smart city. The simulator is unique because it uses basic input parameters such as the number or routes, number of buses and stops, etc. for Chapel Hill (or any city) and returns an upper-bound of network performance. The input parameters for the simulation are listed in Table 4.3.

Table 2.2: Parameters used in the Simulation

Simulation seeds	0:1:99
Simulation duration	48 hours
Number of routes	32
On-route sensors per route	2 to 8
Gateways per route (G)	1 to 2
Number of bus stops per route (S)	$\mu_S = 72.6$, $\sigma_S^2 = 52.44$
Number of Buses per route	1 to 2
Bus velocity (including stops)	17.47 to 21.47 km/h
Sensor data generation rate	10 minutes to 2 hours
Number of buses	38
Number of off-route sensors	100
Circumference of routes	$\mu = 15$ km, $\sigma^2 = 7$ km
Pedestrian to sensor arrival delay	$\mu = 2$ hours, $\sigma^2 = 30$ mins
P of Pedestrian to Gateway delivery	$\frac{\max(G)}{\mu_S} \approx 0.03$
Sensor buffer/queue size	∞
Bus buffer queue/size	∞

The results in Figure 4.1 are for all messages generated and delivered within the simulation period of 48 hours for 100 simulations where the seed was changed from $0 - 99$. Regarding Figure 2.2a, half of the on-route sensor data had a delay of 10 minutes (median) or less before it made it from the sensor onto a bus. Half of the on-route sensor data then had a delay of 15 minutes (median) or less before it made it from a bus to the gateway. As expected, the delays of off-route sensor data in Figure 2.2b was larger as it requires two additional steps (sensor to pedestrian and pedestrian to bus stop) for message delivery. The median time for sensor to pedestrian was 30 minutes while the time from pedestrian to bus stop was 12.5 hours.

In the results, delivery rate is defined as the ratio between messages delivered and the total number of messages generated for each sensor within the simulation

[1]The code for the simulator is available on GitHub - `https://github.com/netreconlab/smartcomp19`

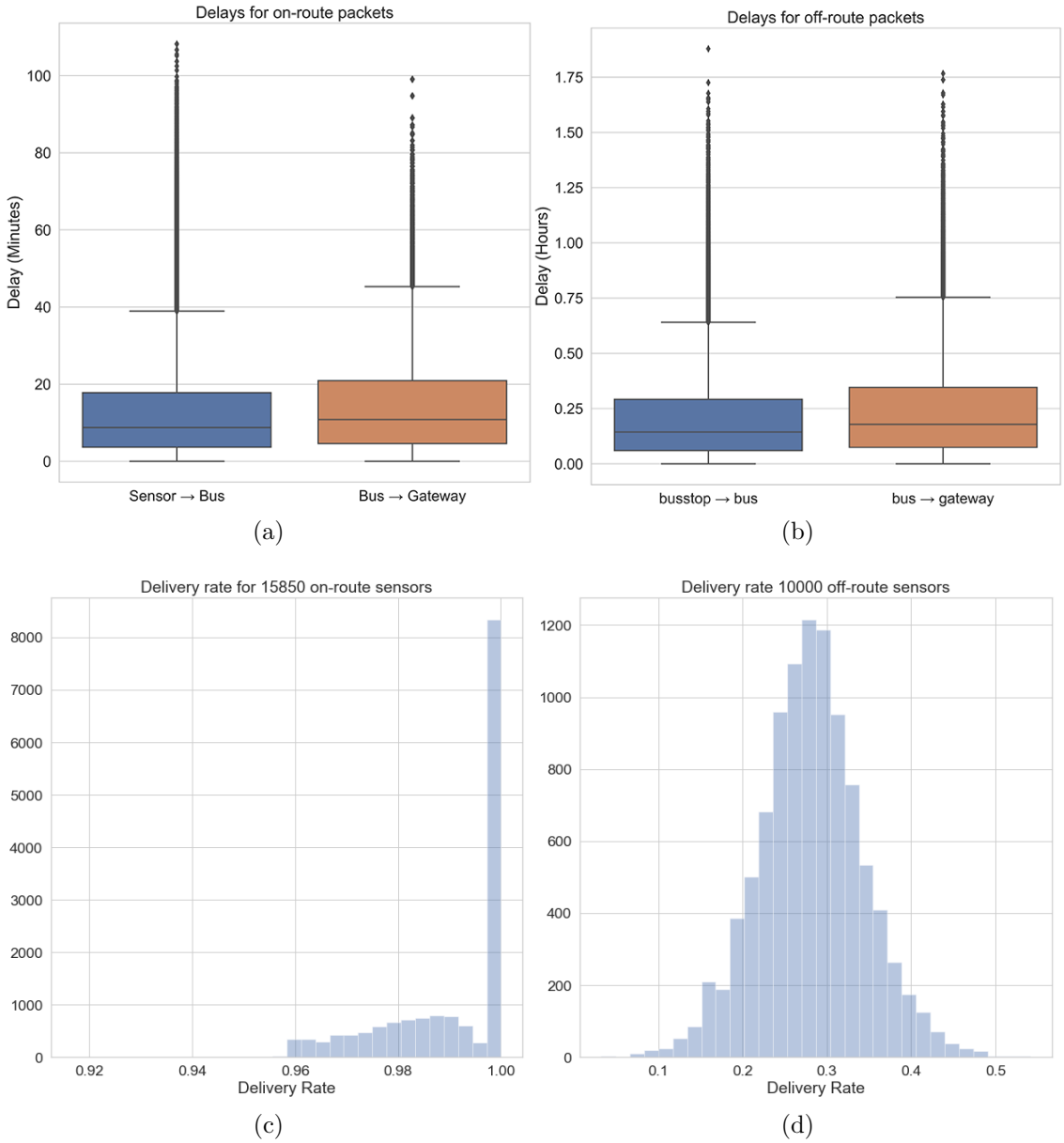


Figure 2.2: (a) Box plot of delivery delay for on-route sensors, (b) Box plot of delivery delay for off-route sensors, (c) Histogram of delivery rate for on-route sensors, (d) Histogram of delivery rate for off-route sensors.

duration. Figure 2.2c depicts that all of the on-route sensor data had a delivery rate of 0.96 or higher. This is due to an assumption in Section 2.3 that data can always be transmitted from sensor to bus once contact is made, regardless of the speed of the bus. The aforementioned assumption is built upon another assumption stated in Section 2.3 that stated data packets are small and can be delivered using Bluetooth 5. Figure 2.2d shows that most of the off-route sensors had a delivery rate of 0.5 or less. Low delivery rates of off-route sensor data is expected since it is dependent on pedestrians to pick up and deliver the data. Pedestrian mobility is not as predictable as bus mobility, and pedestrian transit ridership rate varies widely as described in the Chapel Hill Transit survey [17].

2.5 Future Work

Section 2.4 highlighted the poor network latency and low delivery probability of off-route sensors. Most sensors in a smart city will not have the benefit of being placed on-route. Improving off-route sensor performance is the most critical step for DTN's to serve as a viable option as a backbone in smart cities. Policy makers have significant influence over the number of active public transportation vehicles, schedules, and location of sensors and gateways. They have less control over the number of people who use public transportation, particularly when it comes to pedestrian inter-mobility between public transportation entities. To drive inter-mobility of pedestrians and non-public transportation vehicles such as ride-share vehicles and taxis to pickup off-route sensor data, the development of effective methods that strategically *incentivize* people is essential for increasing network performance in a low-cost smart city. Incentives can be generated from needs in network performances and used to drive data requests across the network using a human over-the-loop paradigm [19]. Classifying data types across the network to understand expected delays, and tuning the latency restrictions imposed by the infrastructure will also drive future research.

Chapter 3 A GTFS-based simulation tool

3.1 Introduction

In recent years, research in the field of vehicular communication networks (VCNs) has grown rapidly. Interest has been spurred by advancements in the fields of autonomous vehicles, electric vehicles, internet-of-things, and smart cities. VCNs refer to networks comprising of vehicular mobile nodes such as cars, buses and trains that communicate with other vehicles or embedded devices using device-to-device communication. Vehicles typically have to be equipped with a mini-computer and wireless radio to enable them to facilitate communication. Some VCNs may also be classified as a vehicular delay-tolerant network (VDTN). VDTNs are similar to Vehicular Ad-hoc Networks (VANETs) except for the fact that unlike VANETs, VDTNs can still forward data from source to destination even when no end-to-end path exists between the source and destination nodes at any particular time instance.

One current challenge when conducting research in VCNs and VDTNs involves how to effectively evaluate the performance of the network architecture. These networks are typically designed to work with a large number of nodes, and so in order to carry out real-world experiments, vasts numbers of IoT devices and other equipment would have to be deployed on a large scale, which is financially expensive [20]. Even when such real-world experiments are carried out, they are often short lived, lack sufficient variety of hardware, lack geographic diversity, and are very limited in scale [20]. One alternative to using real world experiments is strictly utilizing models and simulations. However, the validity of such simulations is highly dependent on how well the simulation environment represents real-world environments and creating such highly representative simulation environments is difficult.

In this chapter, we develop a novel simulation tool[1] that models a vehicular communication network consisting of data generators (sensors), intermediate carriers

[1]The code for the simulator is available on GitHub - `https://github.com/netreconlab/low_cost_smart_city_optimization`

(buses), and destination devices (gateways) within any real-world city, by directly using real transit network information provided in the General Transit Feed Specification (GTFS) format [21]. The simulation tool makes direct use of real transit network information provided in GTFS format [4]. Given that each city has a unique transport network, we are able to analyze the network performance on a city-by-city basis rather just a generalized analysis. In this chapter, we consider a model identical to that presented in Chapter 2, the difference being that in this model, we do not consider off-route sensors and pedestrians. This omission is due to extra complexity involved in integrating the unpredictable nature of pedestrian mobility into our simulation; and data retrieval for off-route sensors is highly dependent on pedestrian mobility. However, including off-route sensors and pedestrian mobility will be a future direction of this work.

Related Works

A number of test-beds and simulation tools have been developed by the research community and presented in literature to facilitate research in the field of VCNs and VDTNs. In H. Soroush et al [22] the authors create the DOME testbed, to give researchers access to transit buses already equipped with necessary equipment, so that external researchers can upload their communication protocols to the buses remotely and conduct experiments within a real-world environment without having to invest in additional infrastructure. In [23], the authors also design a adhoc testbed using busses. Although real-world deployments of various network architectures provide the most accurate results, oftentimes these results are not sufficiently generalized due to insufficient geographic diversity as well as the limited scale of the experiments. In addition, real-world deployments are typically financially expensive.

Hence, one popular alternative to real-world evaluations has been the use of mathematical models and simulations. Some of the advantages of this approach include: very little financial cost, the ability to easily scale up experiments, and the heterogeneity of geography and hardware parameters during experiments. Examples of some widely used open-source generic simulators in the field of vehicular communication

and DTN include the ONE simulator [24] and ns-3 [25]. However, it can be difficult to fully incorporate real-world data into the simulation environment of these generic simulators, or adapt the environment to match the designed network model. Hence, several works in literature use custom-built simulators for their models. In [26], the authors designed a simulation environment for VANETS and intelligent traffic lights to notify vehicles of traffic and warning messages using Adhoc On-Demand Distance Vector (AODV). In [27], the authors conduct a feasibility study by setting up a simulation framework that relies purely on opportunistic interactions between taxi cabs. The lower accuracy of simulation environments, when compared to real-world environments is a factor that often undermines the integrity and validity of results.
The major contribution of this chapter is the development a simulator for our network model that closely emulates the movement of real world transit vehicles, hence we are able evaluate the performance of our network across many cities, while also maintaining high result accuracy.

3.2 Dataset

Many public transit agencies serving various cities already provide highly accurate data that describes their transit networks. The data typically includes the location of routes, stops, and vehicle schedule information. Often, the data is also publicly accessible on Internet platforms such as *OpenMobilityOrg* [16]. Although transit data has been represented in many formats/specification, by far the most popular format is the GTFS. GTFS handles both static information such specify bus routes, stops, and trip timetables; and real-time information about the current location and speed of vehicles in the network [21].

GTFS provides public transit agencies with a standard format for publishing their transit data in an inter-operable manner so that it can be interpreted by various third-party applications. GTFS is an open-source data specification that has been widely adopted and used by hundreds of transit agencies all over the world. It is also used by navigation software such as *Google Maps* to provide users with static and real-time information about transit routes. Thus, by building a simulation tool that

incorporates GTFS static information, we are able to evaluate the performance of our low-cost smart city model for hundreds of cities around the world.

3.3 Simulation Design

In order to be able to analyze the overall viability of our network model, as well as compare the performance of various gateway placement optimization algorithms discussed in Chapter 4 we developed a light-weight simulation tool written in *Python*. The tool takes advantage of the real-world public transit timetables to create a more realistic environment for evaluating the network. In this section, we describe various components of our simulation tool.

Transit feed to graph conversion:

We convert the GTFS transit feed data for a transit agency into a directed graph. The conversion is done using an open-source library called *peartree* [28]. The graph contains: (i) Nodes representing stops, and with each node containing the departure times for all vehicles from that stop, (ii) Edges representing a bus path from one stop to another. The edge weight is the average time it takes for a vehicle to get from source node to destination node.

Sensor Placement:

Sensors are placed at randomly selected stops in the transit network. Each stop has a maximum of one sensor and the total number of sensors to be placed in the network is defined for each simulation. In addition, each sensor is assigned a time value representing the frequency at which data is generated by the sensor. This frequency value is randomly assigned to the sensor based on a uniform distribution.

Gateway Placement:

Gateways are placed in the network using a intuitive algorithm. First, each bus stop is ranked based on the number of routes being serviced by that stop. Gateways are

then placed at each stop in order of decreasing rank until all routes have a gateway located on at least on of its stops. More intricate ways of gateway placement is further discussed in Chapter 4.

Compute data delivery delay:

For each data packet generated at a sensor, the shortest duration it takes for the data packet to get to get to a gateway is computed. First, a subgraph (consisting of only stops in a single route) is extracted from the main graph for each route. Next, we iterate through each route the sensor is on and compute the shortest path length from the sensor to any gateway on that route. Since the edge weight is the average travel time for vehicles between a node pair, the shortest path length computed represents the estimated time taken for a vehicle forward the data to a gateway after departing from the sensor. We calculate the total estimated delay by adding the shortest path length to the waiting time between when the data is generated and the next vehicle for the route arrives at the stop where the sensor is located. After iterating through all routes for the sensor, a path is selected based on the path with the shortest total estimated delay as the path through which the data will travel and thus obtain its path length as the estimated end-to-end delay for the data packet generated. Algorithm 1 presents the pseudocode for computing the delay for each data packet generated.

Algorithm 1 Pseudocode for Computing data delivery delay

```
1:  procedure ComputeDelay(routes, sensor, time)
2:      delay ← ∞
3:      for r ∈ routes do
4:          G_r ← GetRouteSubgraph(r)
5:          for gateway ∈ r.gateways do
6:              length ← DijkstraShortestPath(G_r, sensor, gateway)        ▷
7:              if length then
8:                  wait ← TimeTillNextArrival(sensor, route, time)        ▷
9:                  if wait then
10:                     delay ← min(delay, length + wait)
11:     return delay
```

Storing results for analysis:

For each simulation, important information such as generation time, delivery time, delivery path, vehicle wait time, and vehicle travel time for each data packet generated during simulation is recorded and stored in JavaScript Object Notation (JSON) file. This helps with carrying out post-simulation analysis after the simulation has ended without having to re-run simulations.

3.4 Results

Simulations were conducted using GTFS feeds provided by four transit agencies in the United States namely; (i) Chapel Hill Transit (CHT) in Chapel Hill, (ii) King County Metro (KCM) in Seattle, (iii) Metropolitan Transportation Authority in New York, and (iv) Southeastern Pennsylvania Transportation Authority (SEPTA) in Pennsylvania.

Table 3.1: Simulation Parameters

Random generator seeds	0:1:9
Simulation start time	00:00:00
Simulation end time	23:59:59
Number of Sensors	$5\% \times \lvert stops \rvert$
Sensor data generation frequency (minutes)	$U(1, 120)$

For each transit network, a simulation is ran for a 24-hour time period for the busiest day of the week based on its GTFS feed. The simulation parameters used for each simulation is shown in table 3.1. Figure 3.1 shows a snapshot of an interactive map of Chapel Hill overlaid with route, sensor, and gateway locations. The black lines represent routes in the network, while red circles and blue squares represent sensors and gateways, respectively. Clicking on any sensor of interest can give a summary of all relevant information regarding that sensor for the duration of the simulation. Figure 3.2 represents the distribution of the average (mean) delivery delay per data packet generated for various transit agencies.

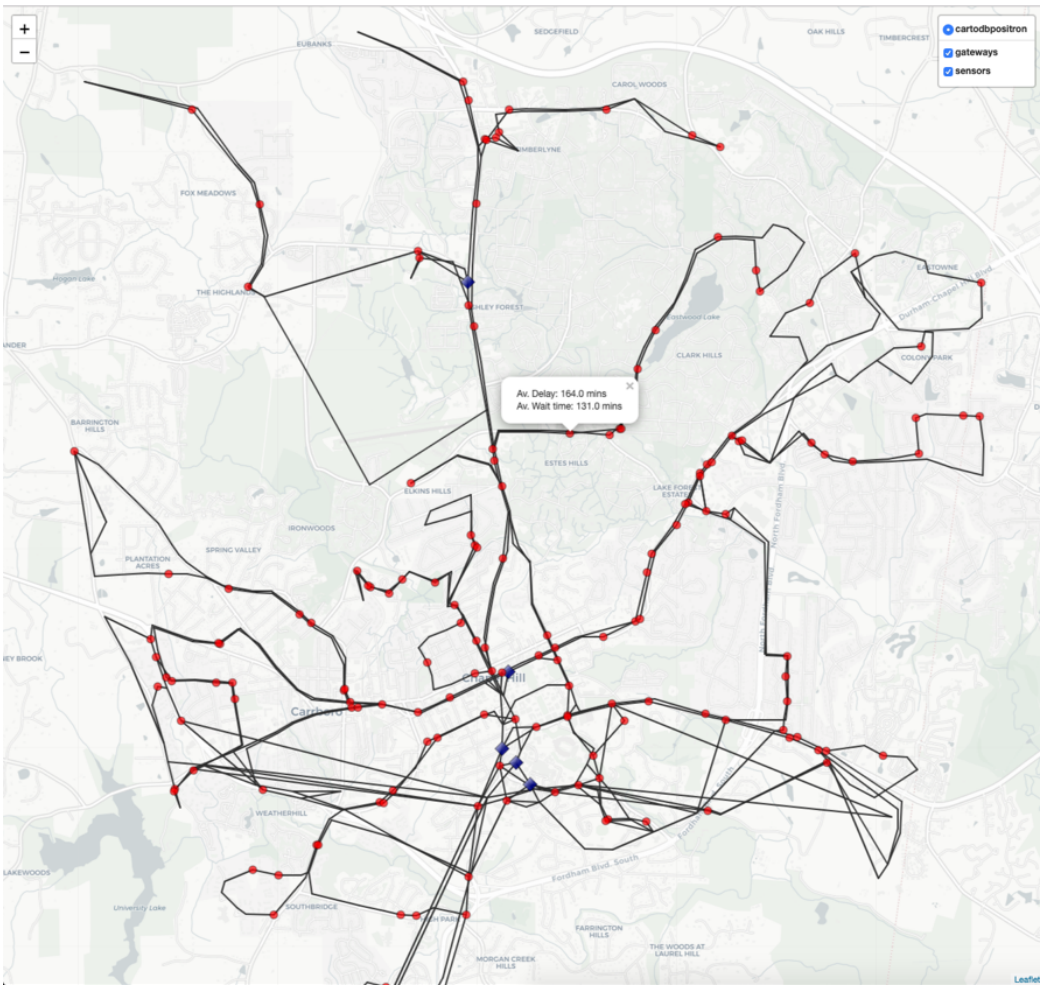


Figure 3.1: Snapshot of an interactive map overlaid with route, sensor, and gateway locations

3.5 Conclusion

In this chapter, we presented a tool that utilizes GTFS static feeds and *graph theory* to run simulations for VDTNs within smart cities. Results are stored using the JSON format for future analysis. City feeds from Chapel Hill, Seattle, Pennsylvania and New York for demonstrative simulations. In future works, more analyses will be completed to support the validity of simulation results. We will also implement other gateway placement optimization techniques based on different criteria. Finally, we will investigate what impact various characteristics of a transit network have on the performance of DTNs in our network model.

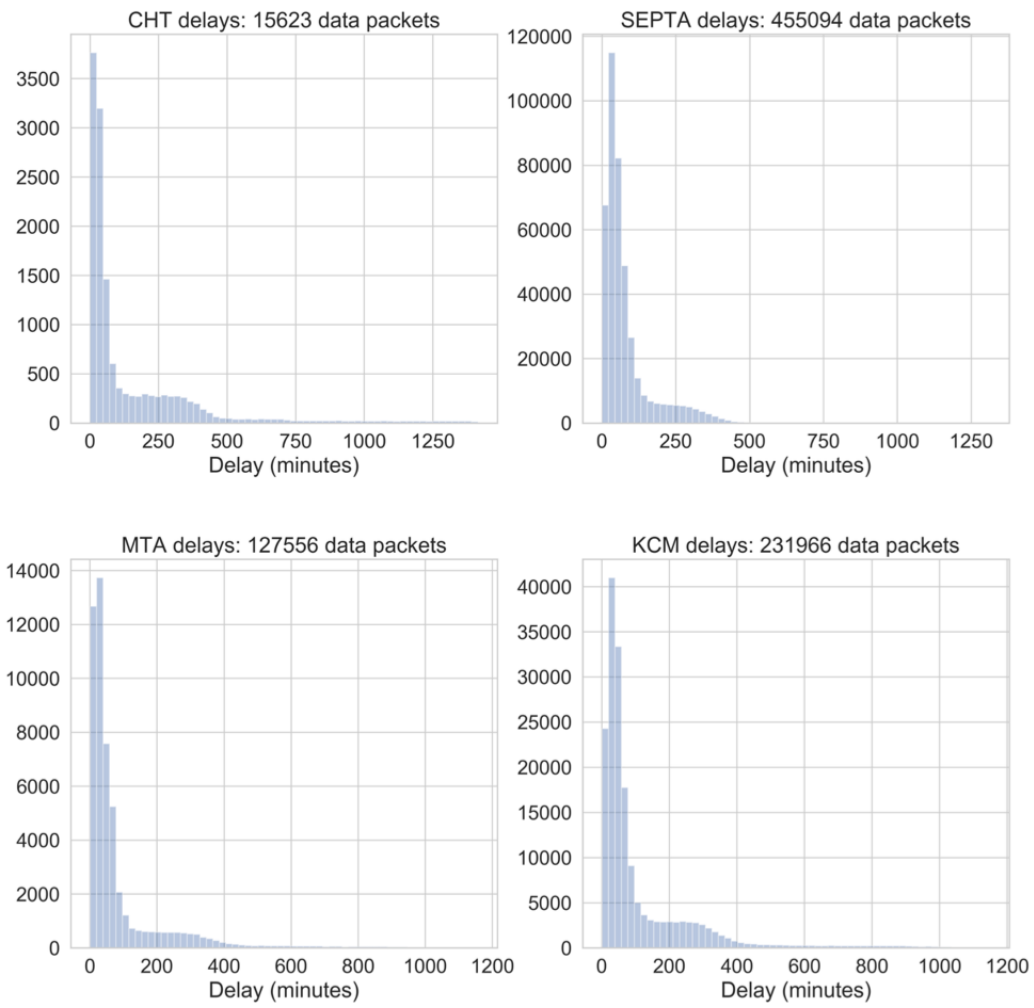


Figure 3.2: Distribution of average delivery delay per sensor for various transit networks

Chapter 4 Optimal Gateway Placement in Low-cost Smart Cities

4.1 Introduction

The decision of where to place gateways in a city is a significant problem in a low-cost smart city. It is significant because gateways serve as the only medium through which any sensor data is sent to the cloud. Furthermore, gateways are required to be always-on, always-connected devices, and in some cases even have edge computing capabilities for processing data before forwarding to the cloud. This means that a power source may be needed at the location of the gateway. Direct high-speed internet access also needs to be available at the gateway location, as well as other expensive resources including CPU, RAM, and persistent storage. All of aforementioned gateway requirements means that for each additional gateway, an extra cost is required. Since the primary objective of the proposed network model is to enable smart cities at a fraction of the cost, the fewer the number of gateways required in the network to achieve desirable network performance, the lower the setup and maintenance cost will be.

In this chapter we: (i) formulate three gateway placement optimization problems in our network architecture, (ii) identify optimization problems in literature that are special cases of our optimization problem, (iii) develop several algorithms for solving the optimization problems highlighted, and finally (iv) compare the results of our algorithms with traditional network centrality measures

4.2 Related Works

Placement optimization algorithms are typically specific to each network architecture and model. Some previous works on placement optimization for networks similar to ours include Krause et al. [29] which exploited the principle of submodularity to tackle the problem of sensor placements in water distribution networks. In [30, 31], the authors explore several optimization techniques for access point placement in

wireless mesh networks.

The problem of gateway placement is similar in a few ways to finding the most significant stops in a transit network. In the field of network analysis, there exist several popular centrality measures. These centrality measures are usually computed by a real-valued function and reflects a node's significance or importance within the given network [32]. The centrality measures have been used in many kinds of networks including the Internet, social networks, biological networks, and transportation networks. Unfortunately, centrality measures work best with simple static networks [33] and not dynamic networks. Since our network model is more complex, containing not just nodes (stops) and edges(trips), but also vehicle schedule information across each node, centrality measures may not prove very useful for optimizing gateway placement. Hence, for the optimization problems considered in this chapter, we explore other solutions. We also compare our solutions to two centrality measures, "betweenness centrality" and "in-degree centrality."

4.3 Problem Formulation

In this chapter, we consider two gateway optimization objectives relevant to enabling low-cost smart cities. We assume that the cost associated with installing a gateway is constant regardless of the location being considered.

Maximal Sensor Coverage (MSC)

The first optimization problem considered is Maximizing Sensor Coverage (MSC). The objective is to find the minimum/smallest set of locations to install gateways such that there is at least one direct path in the network from all possible sensor locations to a gateway. This would ensure that all sensors, regardless of where they are placed on the transit network will have a chance of having its data delivered.

A typical bus transit network consists of several routes through which vehicles move. Each route comprises of a set of stops. These stops are locations where vehicles (buses) on the route stop to pick up passengers. It is very common for two or

more routes share similar stops. Therefore, we can define our MSC problem as such: Given a set of routes R, and a list of stops, L, each with a subset of routes, $R_l \subseteq R$, that use that stop (each element of R is associated with at least one stop), find the minimal set of stops required to cover all routes. This problem can be reduced to a minimal set cover problem. In the set cover problem, we are given a universal set U, such that $|U| = n$, and a family of subsets $S_1, ..., S_k \subseteq U$. A set cover is a collection C of the subsets $S_1, ..., S_k$ whose union is the is the universal set U. Formally, C is a set cover if $\bigcup_{S_i \in C} S_i = U$. To find the minimal set cover, the objective is to minimize $|C|$.

The reduction is fairly intuitive. In our case the universal set is the set of all routes $R = U$, and the family of subsets are the set of routes each stop services, $L = S$. Given a decision variable, $x_l \in \{0, 1\}$, which indicates whether a stop in L is picked, the ILP formulation is thus:

$$\text{minimize} \sum_{l \in L} x_l \text{ st} \tag{4.1}$$

$$\sum_{l:r \in L} x_l \geq 1 \quad \forall\, r \in R \tag{4.2}$$

$$x_l = \{0, 1\} \quad \forall\, l \in L \tag{4.3}$$

The problem of finding the optimal set cover solution is NP-Complete (one of Karp's 21 NP-complete problems). Nevertheless, the greedy approach is able to find a solution close to the optimal set cover. It is bounded above by a $O(log_e n)$ approximation to optimal solution of the set cover problem, where n is the number of routes in the network. The greedy MSC algorithm is described in Algorithm 2. At each iteration, we find the gateway candidate that provides the largest increase in the number of routes covered, and add it to the gateway set. This process is repeated until all routes have been covered by the gateway set.

Minimal Delivery Delay (MDD)

The next optimization problem considered is the Minimal Delivery Delay problem (MDD). Here, the objective is to select the set of locations to place gateways such that the average network latency for data generated across the network is minimized,

Algorithm 2 Maximal Sensor Coverage (MSC)

Input — routes **R**, route subsets **S**
Output — selected gateway set **G**

1: **procedure** GREEDY-MSC(R, S)
2: $X \leftarrow R$
3: $G \leftarrow \emptyset$
4: **while** $X \neq 0$ **do**
5: `Select an` $S_i \subseteq R$ `that maximizes` $|S_i \cap X|$
6: $X \leftarrow X \setminus S_i$
7: $G \leftarrow G \cup S_i$
8: **return** G

without exceeding a budget constraint. In considering network latency, we account for both delivered and undelivered data. We assign a penalty value to data that is undelivered, as explained in Section 4.3. The budget constraint refers to the number of gateways that can be added to the network. Also, the gateway selection process is conducted without prior knowledge of the stops in which the sensors will placed in the city.

In formulating the problem, we consider each potential gateway location to possess a influence value, which describes its impact on the network latency if it is added to an existing set of gateways. Hence, our problem objective is to select a set of gateways below a specified cardinality, that together decreases the network latency the greatest. Given a decision variable, $y_l \in \{0, 1\}$, which indicates whether a stop was selected as a gateway, a budget constraint, B and an latency impact parameter, $I_l \in I$ associated with every stop $l \in L$, the ILP formulation is thus:

$$\text{maximize} \sum_{l \in L} y_l I_l \text{ st} \tag{4.4}$$

$$\sum_{l \in L} y_l \leq B \quad \forall\ l \in L \tag{4.5}$$

$$y_l = \{0, 1\} \quad \forall\ l \in L \tag{4.6}$$

In developing our algorithm for this problem, we use a technique similar to the Influence Maximization (IM) problem. IM problems typically involve the task of selecting a small subset of nodes in a network to serve as seeds for the propagation of some entity (such as information or product adoption) so that the entity spreads

to the largest number of nodes in the network [34]. Influence Maximization (IM) algorithms have been applied in social network analysis such as viral marketing, disease modeling, and public health interventions [35, 36]. Kempe et al. (2003) [34] formalized the IM problem as: Given a network with n nodes and given a propagation process on that network, choose a set of nodes called the *seed set* S of size $k < n$ that maximizes the number of nodes in the network that are ultimately influenced. Solving this problem turns out to NP-Hard. Over the years, researchers have developed several approximate solutions. However, our problem differs from the traditional IM problem because the set of nodes we want to select are not seed/source nodes but destination nodes (gateways). Therefore, in solving our problem, we define our own influence function below and then modify two well-known approximation algorithms; (i) *Greedy* and *Cost-Effective Lazy Forward* (CELF) [37].

Influence Function

Current IM algorithms require an influence function that simulates the propagation process and computes the marginal influence that each potential seed has on the overall propagation. Therefore for our algorithm, we have to develop an influence function (σ) that computes the expected latency across the network for any potential set of gateways.

The influence function makes use of the network simulation described in Section 3.3 to calculate the network latency. In addition, given that there is no prior knowledge of what stops sensors would be located when selecting gateways, we compute the network latency over multiple sensor placement scenarios rather than just one, through a Monte Carlo simulation. This method increases the likelihood of the influence function returning an unbiased estimation of network latency for a specific set of gateways. Hence, we generate a set of sensor placement scenarios, in which each scenario is essentially a simulation environment consisting of the transit network graph, where n sensors are randomly placed at various stops, and each sensor randomly assigned a message generation frequency.

We define the influence of a set of gateways as the degree to which network latency

is reduced by the set. Therefore, the lower the value of the latency, the greater the influence value of the gateway(s). The influence value calculated by subtracting the latency from the simulation's upper bound value for latency. The upper bound value is the maximum latency possible in the network, and in our case we pick a fixed value outside the range of simulation window. This upper bound value is also the penalty value for data that is undelivered.

Influence Function Submodularity

We postulate that our influence function is a monotonically increasing sumbmodular function.

Definition: Given a finite ground set, U, and a set function $f : 2^U \rightarrow \mathbb{R}$, f is a submodular function if $\forall\ A \subseteq B \subseteq U,\ a \in U \setminus B$.

$$f(A + a) - f(A) \geq f(B + a) - f(B) \tag{4.7}$$

Let the ground set be the set of latency impact parameters (stop influence), I, associated with each stop, $l \in L$, such that $I_l \geq 0\ \forall\ I_l \in I$. All possible gateway placement sets can be represented by $P \subseteq U$ such that the budget is met, $|P| \leq B$. Also, let $f(P)$ be the maximum influence, that can be acquired from a gateway placement set, P. It is easy to see that the $f(\emptyset) = 0$ because, the maximum influence acquired from the empty set is 0. Hence, f is normalized. Next, we ask the question: *Is $f(P)$ monotonically increasing?*

Let A and B be possible placement sets such that $A \subseteq B \subseteq U, \forall\ A\ f(A) \leq f(B)$. Suppose $f(B) < f(A)$, then there has to be an element, which when added to the gateway placement set, A, results in a decreased $f(A)$. Since $I_l \geq 0\ \forall\ I_l \in I$, no such element exists. Thus, it is monotonically increasing. Since it is monotonically increasing, we proceed to ask the question: *Is $f(P)$ submodular?*

Contradiction: Assume that $f(A + a) - f(A) < f(B + a) - f(B)$. Then there has to be an element, a, for which the marginal increase in influence derived from adding a to set A is less than that of adding a to set B. This is not possible because $A \subseteq B$ and is monotonically increasing, thus the entire influence of A is captured in

B. As a result, if an element causes a marginal influence increase of x when added to set B, the marginal increase on set A has to be greater or equal to x. Hence, it is submodular.

Greedy Algorithm

Since the problem is reduced to the maximization of a monotone submodular function, the greedy algorithm for IM provides a $(1 - 1/e)$ - approximation [37]. Hence, our greedy algorithm is theoretically guaranteed to choose a gateway set whose network latency will be at least 63% of the network latency of the optimal gateway set. Our greedy algorithm is described in Algorithm 3. It starts with an empty gateway set $S = \emptyset$. In each iteration, the greedy heuristic chooses a new gateway u from the non-gateway nodes $V \setminus S$ with largest (marginal) influence gain $\sigma(S \cup v) - \sigma(S)$ and adds u to S. The algorithm stops after selecting k gateways.

Algorithm 3 Greedy Minimal Delivery Delay

Input — network graph **N**, influence function σ, budget **k**
Output — selected gateway set **G**

1: **procedure** GREEDY-MDD(N, σ, k)
2: $G \leftarrow \emptyset$
3: **while** $|G| < k$ **do**
4: $u \leftarrow arg\ max_{\ v \in V \setminus G}\ \sigma(G \cup v) - \sigma(G)$
5: $G \leftarrow G \cup \{u\}$
6: **return** G

CELF Algorithm

Although the greedy algorithm is much quicker than a brute-force approach, greedy algorithm is still very slow when used on realistically sized transit networks. CELF algorithm has a significantly reduced the running time despite returning the exact same set of gateways as our greedy algorithm [38].

CELF exploits the submodular property of our influence function. CELF eliminates the need to compute the marginal influence value of all potential gateways at each iteration. In the first round, we calculate the influence for all gateway candidates

(like Greedy), select the gateway with the greatest influence, and store the influence values the other candidates in a max heap. In subsequent iterations, the marginal influence for the top gateway candidate in the heap is computed and added back to the heap. If the gateway candidate remains at the top of the heap, then it must have the highest marginal influence of all gateway candidates due to the sub-modular property of the influence function. If a different candidate is on top of the heap, the process continues until a candidate remains on top after two iterations, after which that candidate is added to the gateway set. This process is repeated until the gateway budget has been met.

Algorithm 4 CELF Minimal Delivery Delay (CELF-MDD)

Input — network graph **N**, influence function σ, budget **k**
Output — selected gateway set **G**

1: **procedure** CELF-MDD(N, σ, k)
2: $G \leftarrow \emptyset$
3: $Q \leftarrow \emptyset$
4: **for** $v \in N.nodes$ **do**
5: $u \leftarrow v$
6: $u.gain = \sigma(\{v\})$
7: `add` u `to` Q `in descending order of` $u.gain$
8: **while** $|G| < k$ **do**
9: $u \leftarrow Q.top$
10: **if** $u.flag = |G|$ **then**
11: $G \leftarrow G \cup \{u\}$
12: $Q \leftarrow Q \setminus u$
13: **else**
14: $u.gain \leftarrow \sigma(G \cup \{u\}) - \sigma(G)$
15: $u.flag \leftarrow |G|$
16: `Re-sort` Q `in descending order of` $u.gain$
17: **return** G

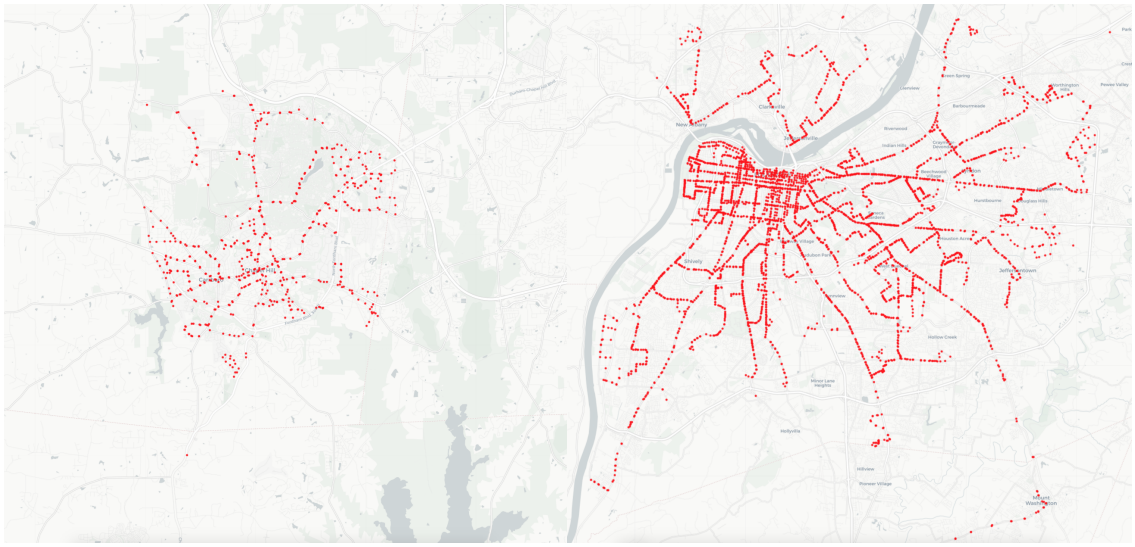
4.4 Numerical Evaluation

In order to evaluate the accuracy of our optimization algorithms, we make use of the GTFS data of two bus transit agencies; Chapel Hill Transit (CHT) in Chapel Hill, North Carolina, and Transit Authority of River City (TARC) in Louisville, Kentucky [16]. Chapel Hill is a relatively small town with measuring 55 km sq (21.3

sq miles) and an estimated population of about 60,988. Louisville, on the other hand, is a much larger city with a population of 620,118 and land area of 171.70 km sq (66.29 sq miles) [39]. The difference in city size also translates to the public transport network present in both cities as highlighted in Table 4.1 and Figure 4.1a.

Table 4.1: Bus Network Characteristics CHT and TARC

Statatistics	**CHT**	**TARC**
Routes	26	46
Stops	571	4391
Total trips	1252	1917
Betweenness centrality avg.	0.04896	0.00829
In-degree centrality avg.	0.00210	0.00025



(a) (b)

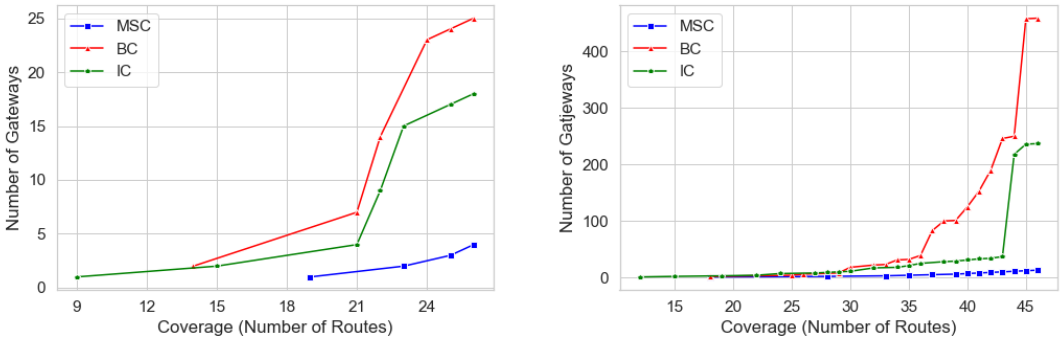
Figure 4.1: (a) Map of CHT bus-stop locations in Chapel Hill, (b) Map of TARC bus-stop locations in Louisville.

Maximizing Coverage

Figures 4.2 and 4.2 give insight into the performance of the MSC Algorithm described in 4.3 when compared to two traditional graph centrality measures namely the results from betweenness centrality (BC), and in-degree centrality (IC). For the centrality measures, we picked each gateway in order of decreasing centrality, until all routes

were covered. Figure 4.2 indicates the rate of increase in route coverage compared to the number of gateways selected. We see that MSC outperforms BC and IC significantly for both CHT and TARC. For CHT, MSC requires just 4 gateways to cover all routes in the network, compared to 18 and 25 gateways in IC and BC, respectively. For a larger network like TARC, MSC requires 13 gateways to cover all routes in the network, compared to 257 and 459 gateways in IC and BC, respectively.

Table 4.2 also highlights the delivery ratio for each algorithm. Since the three algorithms (MSC, BC, and IC) cover all routes, the delivery ratio is essentially the same. Delivery ratio is less than 100% because delivery also depends on a bus arriving at a sensor within the simulation time window. This means that for some data packets generated, especially in the later stages of the simulation, there may no longer be buses moving on that route causing the data packet to remain undelivered.



(a) (b)

Figure 4.2: (a) MSC performance for CHT, (b) MSC performance for TARC.

Table 4.2: Results of MSC, BC and IC

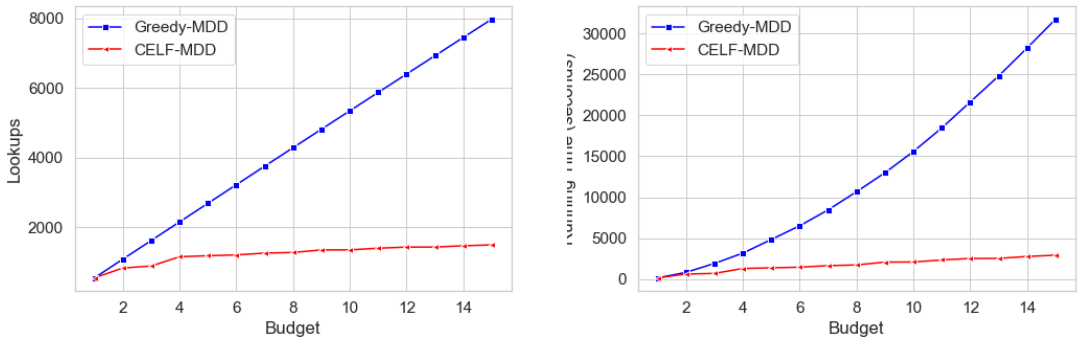
	CHT		TARC	
Algorithms	Cost	Delivery ratio (%)	Cost	Delivery ratio (%)
MSC	4	86.0512	13	87.0903
BC	25	84.4760	459	87.8831
IC	18	85.9515	237	88.4032

Table 4.3: Simulation and Scenerio Parameters

Random generator seeds	0:1:9
Simulation start time	1:00:00
Simulation end time	24:00:00
Latency upper bound (hours)	25
Number of sensors	$30\% \times \|stops\|$
Sensor data generation frequency (minutes)	$U(1, 120)$
Number of sensor scenarios	5
Number of sensors for scenarios	$U(30, 40)$

Minimizing Latency

We evaluate the effectiveness of the CELF-MDD algorithm (described in Section 4.3) at minimizing the overall message delay in the network model. We first compare the difference in run-time efficiency between Greedy-MDD and CELF-MDD. Figures 4.3a and 4.3b show the average number of lookups and running-times for a gateway budget of up to 15 when working with the CHT network. The number of lookups at each budget refers to the number of times the influence function needs to be computed before a gateway was selected at that stage. Greedy-MDD grows linearly because for each iteration, the influence gain for all gateways that haven't been selected has to be computed. However for CELF-MDD, the number of lookup grows much slower due to it leveraging results from past computation as discussed in Section 4.3.



(a) (b)

Figure 4.3: (a) Graph showing the number of lookups for Greedy-MDD and CELF-MDD, (b) Graph showing the number of running time for Greedy-MDD and CELF-MDD

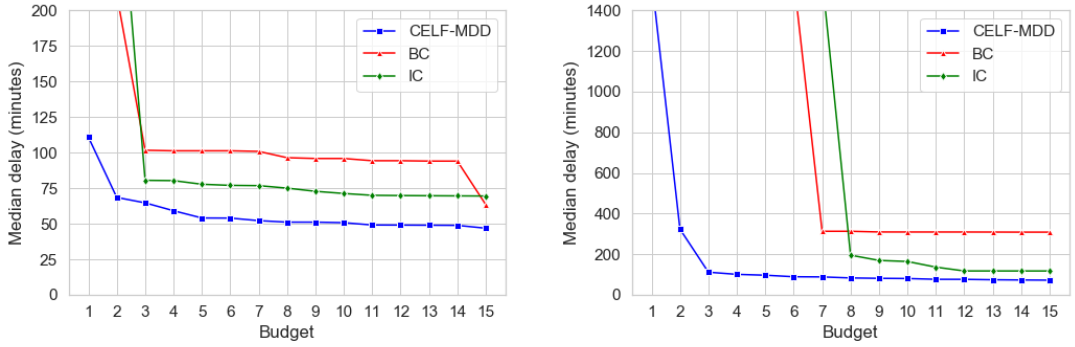


Figure 4.4: (a) Average network latency for CHT, (b) Average network latency for TARC.

Finally, we compare the performance of CELF-MDD to BC and IC in terms of network latency minimization. For each algorithm we select the first k set of gateways generated by the algorithm where k represents the budget. Simulations are then run with that gateway set using the simulation parameters outlined in Table 4.3. Figure 4.4a and 4.4b illustrates the average network latency at various budgets for CHT using each algorithm. The simulations were run using the parameters specified in Table 4.3. Similar to our influence function in 4.3, we assign the same upper bound value specified in Table 4.3 as the penalty value (delay) for undelivered data.

For CHT, we observe that CELF-MDD consistently outperforms both BC and IC by $\approx$ 20 minutes or higher. We also observe very little decrease in delay after first 5 gateways have been selected when using CELF-MDD. For TARC, we observe that CELF-MDD consistently outperforms both BC and IC by 45 minutes or higher. We also observe very little decrease in delay after first 3 gateways have been selected when using CELF-MDD.

4.5 Conclusion

In this chapter, the problem of how to efficiently place gateways in low-cost smart cities was addressed. We introduced several optimization algorithms and compared them to traditional network centrality measures. We carried out our experiment

using public transport networks in two cities in the United States; Chapel-Hill and Louisville. Finally, we presented results which show our algorithms outperforming traditional centrality measures for two performance metrics, coverage at minimal cost, and network latency.

Chapter 5 Conclusion and Future Works

In this thesis, we have presented a network model that leverages the existing mobility of people and vehicles within a city to form an opportunistic and delay-tolerant network for collect data from wireless sensors in a cost efficient manner. First we grouped sensors in smart cities into two types: on-route sensors and off-route sensors, and showed that for on-route sensors, the network latency and delivery probability is tolerable. Next we designed a tool and takes full advantage of available public transit timetables of many cities to provide estimates on the network latency of our proposed model for on-route sensors.

Future directions of this research include modelling pedestrians and transit riders and integrating it into simulation environments to make the models more robust. Also, due to the poor network performance of off-route sensors in a low-cost smart city, the development of effective methods of incentivization for people is essential for DTN's to serve as a viable option as a backbone in smart cities. Classifying data types across the network to understand expected delays and tuning latency restrictions imposed by the infrastructure will also drive future research.

Bibliography

[1] United Nations, "World Urbanization Prospects: The 2018 Revision, Key Facts," p. 2, 2018. [Online]. Available: https://esa.un.org/unpd/wup/publications/Files/WUP2018-KeyFacts.pdf

[2] T. C. of San Diego, "San Diego Deploys the World's Largest Smart City Platform," 2019. [Online]. Available: https://www.sandiego.gov/sustainability/energy-and-water-efficiency/programs-projects/smart-city

[3] G. Technology, "New Orleans Wants to Be a Smart City, But at What Cost?" [Online]. Available: http://www.govtech.com/applications/New-Orleans-Wants-to-Be-a-Smart-City-But-at-What-Cost.html

[4] Smart London Board, "Smart London Plan: Using the creative power of new technologies to serve London and improve Londoners' lives," p. 54, 2013. [Online]. Available: https://www.london.gov.uk/sites/default/files/smart_london_plan.pdf

[5] L. Garfield, "Songdo, South Korea has an eco-friendly design," 2018. [Online]. Available: http://uk.businessinsider.com/songdo-south-korea-design-2017-11

[6] Deloitte, "The challenge of paying for smart cities projects," Tech. Rep., 2018. [Online]. Available: http://dx.doi.org/10.14257/ijfgcn.2014.7.1.15

[7] SmartCitiesWorld, "Smart cities : understanding the challenges and opportunities," Tech. Rep.

[8] J. Paradells, C. Gómez, I. Demirkol, J. Oller, and M. Catalan, "Infrastructureless smart cities. Use cases and performance," *2014 International Conference on Smart Communications in Network Technologies, SaCoNeT 2014*, 2014.

[9] V. Cerf, S. Burleigh, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-Tolerant Networking Architecture," pp. 1–35, 2007. [Online]. Available: https://tools.ietf.org/html/rfc4838

[10] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2003, pp. 27–34.

[11] T. Liu, C. M. Sadler, P. Zhang, and M. Martonosi, "Implementing software on resource-constrained mobile sensors: Experiences with impala and zebranet," in *Proceedings of the 2nd international conference on Mobile systems, applications, and services*. ACM, 2004, pp. 256–269.

[12] H. Ochiai, H. Ishizuka, Y. Kawakami, and H. Esaki, "A DTN-based sensor data gathering for agricultural applications," *IEEE Sensors Journal*, vol. 11, no. 11, pp. 2861–2868, 2011. [Online]. Available: http://ieeexplore.ieee.org.

[13] P. C. Cheng, K. C. Lee, M. Gerla, and J. Härri, "GeoDTN+Nav: Geographic DTN routing with navigator prediction for urban vehicular environments," *Mobile Networks and Applications*, vol. 15, no. 1, pp. 61–82, 2010.

[14] J. Morgenroth, W.-B. Pöttner, S. Schildt, and L. Wolf, "Performance issues and design choices in delay-tolerant network (dtn) algorithms and protocols," in *Advances in Delay-Tolerant Networks (DTNs)*. Elsevier, 2015, pp. 225–250.

[15] C. E. Baker, A. Starke, T. G. Hill-Jarrett, and J. McNair, "In vivo evaluation of the secure opportunistic schemes middleware using a delay tolerant social network," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 2537–2542.

[16] OpenMobilityOrg, "OpenMobilityData - Public transit feeds from around the world." [Online]. Available: https://transitfeeds.com/

[17] CUBIC, "Nextbus Public XML Feed." [Online]. Available: https://www.nextbus.com/xmlFeedDocs/NextBusXMLFeed.pdf

[18] E. Institute, "Chapel Hill Transit Passenger Survey," Tech. Rep., 2016. [Online]. Available: https://www.townofchapelhill.org/home/showdocument?id=33401

[19] A. Graham, Y. Liang, L. Gruenwald, and C. Grant, "[research paper] formalizing interruptible algorithms for human over-the-loop analytics," in *2017 IEEE International Conference on Big Data (Big Data)*, Dec 2017, pp. 4378–4383.

[20] R. Iziduh, J. Jackson, and H. Sueing, "The Design of a Simulation for the Modeling and Analysis of Public Transportation Systems as Opportunistic Networks," Tech. Rep. 2. [Online]. Available: http://www.cscjournals.org/csc/manuscript/Journals/IJCN/volume2/Issue4/IJCN-69.pdf

[21] "GTFS Static Overview — Static Transit — Google Developers." [Online]. Available: https://developers.google.com/transit/gtfs/

[22] H. Soroush, N. Banerjee, A. Balasubramanian, M. D. Corner, B. N. Levine, and B. Lynn, "DOME: a diverse outdoor mobile testbed," *Proceedings of the 1st ACM International Workshop on Hot Topics of Planet-Scale Mobility Measurements*, pp. 1–6, 2009.

[23] J. Paradells, C. Gomez, I. Demirkol, J. Oller, and M. Catalan, "Infrastructureless smart cities. use cases and performance," in *Smart Communications in Network Technologies (SaCoNeT), 2014 International Conference on*. IEEE, 2014, pp. 1–6.

[24] A. Keränen, J. Ott, and T. Kärkkäinen, "The one simulator for dtn protocol evaluation," in *Proceedings of the 2nd international conference on simulation tools and techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, p. 55.

[25] T. R. Henderson, M. Lacage, and G. F. Riley, "Network Simulations with the ns-3 Simulator," in *SIGCOMM'08*, 2008, p. 527. [Online]. Available: http://www.isi.edu/nsnam,

[26] C. T. Barba, M. A. Mateos, P. R. Soto, A. M. Mezher, and M. A. Igartua, "Smart city for vanets using warning messages, traffic statistics and intelligent traffic lights," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE, 2012, pp. 902–907.

[27] M. Bonola, L. Bracciale, P. Loreti, R. Amici, A. Rabuffi, and G. Bianchi, "Opportunistic communication in smart city: Experimental insight with small-scale taxi fleets as data carriers," *Ad Hoc Networks*, vol. 43, pp. 43–55, 2016.

[28] K. Butts, "Peartree: A library for converting transit data into a directed graph for sketch network analysis." 2018. [Online]. Available: https://github.com/kuanb/peartree

[29] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos, "Efficient sensor placement optimization for securing large water distribution networks," *Journal of Water Resources Planning and Management*, vol. 134, no. 6, pp. 516–526, 2008.

[30] F. Xhafa, C. Sánchez, A. Barolli, and M. Takizawa, "Solving Mesh Router Nodes Placement Problem in Wireless Mesh Networks by Tabu Search Algorithm," Tech. Rep.

[31] F. Xhafa, C. Sánchez, and L. Barolli, "Locals Search Algorithms for Efficient Router Nodes Placement in Wireless Mesh Networks," 2009.

[32] M. Newman, *Networks: An Introduction*. New York, NY, USA: Oxford University Press, Inc., 2010.

[33] W. Chen and S.-H. Teng, "Interplay between Social Influence and Network Centrality: A Comparative Study on Shapley Centrality and Single-Node-Influence Centrality." [Online]. Available: http://dx.doi.org/10.1145/3038912.3052608

[34] D. Kempe, J. Kleinberg, and Tardos, "Maximizing the spread of influence through a social network," *Theory of Computing*, vol. 11, pp. 105–147, 2003.

[35] C. Wang, W. Chen, and Y. Wang, "Scalable influence maximization for independent cascade model in large-scale social networks," *Data Mining and Knowledge Discovery*, vol. 25, no. 3, pp. 545–576, 2012.

[36] W. Chen, Y. Yuan, and L. Zhang, "Scalable influence maximization in social networks under the linear threshold model," Tech. Rep., 2010.

[37] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions-I," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978. [Online]. Available: https://www.researchgate.net/publication/242914003

[38] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. Vanbriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007, pp. 420–429.

[39] U.S. Census Bureau, "U.S. Census Bureau QuickFacts: UNITED STATES," 2016. [Online]. Available: https://www.census.gov/quickfacts/fact/table/US/PST045218#qf-headnote-a%0Ahttps://www.census.gov/quickfacts/fact/table/US/PST045216

Vita

Shina Madamori

Education

2010 - 2014 **B. Eng. Electrical and Electronic Engineering**
Federal University of Technology, Minna

Appointments

Jul 2019 - Aug 2019 **Software Engineer Intern**
Autodesk Inc., Portland, OR

Jan 2019 - May 2019 **Research Assistant**
Department of Computer Science, University of Kentucky

Aug 2017 - Dec 2018 **Teaching Assistant**
Department of Computer Science, University of Kentucky

Aug 2016 - July 2017 **Software Engineer**
Bincom ICT Solutions, Lagos

Apr 2013 - Oct 2013 **Network Support Intern**
Cobranet, Lagos

Publications

Conference Proceedings

1. E. Max-Onapkoya, **O. Madamori**, F. Grant, R. Vanderpool, M. Chih, D. Ahern, E. Aronoff-Spencer, and C.E. Baker, "Augmenting cloud connectivity with opportunistic networks for rural remote patient monitoring," in IEEE International Conference on Computing, Networking, and Communications (ICNC), 2020 (In-press, Acceptance: 24.9%)

2. **O. Madamori**, E. Max-Onapkoya, C. Grant, and C.E. Baker, "Using delay tolerant networks as a backbone for low-cost smart cities," in 5th IEEE International Conference on Smart Computing (SMARTCOMP), 2019

Poster Sessions

1. **O. Madamori** and C.E. Baker, "Opportunistic device-to-device communication for low-cost smart cities," in ACM Richard Tapia Celebration of Diversity in Computing (TAPIA), 2019

Honors and Awards

- University of Kentucky Lexington Herald-Leader Graduate Fellowship, $6,000 plus semester tuition, 2018