

SIMULATED DEPLOYMENT OF AN AD-HOC MESH NETWORK IN AN URBAN
METROPOLITAN AREA

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE
UNIVERSITY OF HAWAI'I IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

INFORMATION AND COMPUTER SCIENCES

AUGUST 2010

By
Andrew Wong

Thesis Committee:

Edoardo Biagioni, Chairperson
Henri Casanova
Scott Robertson

We certify that we have read this thesis and that, in our opinion, it is satisfactory in scope and quality as a thesis for the degree of Master of Science in Information and Computer Sciences.

THESIS COMMITTEE

Chairperson

©Copyright 2010

by

Andrew Wong

Acknowledgments

I would like to thank the following people for their support while writing this thesis:

Larry Ruckman, Jamal Rorie, James Kennedy, and Gary Varner, for the experience and confidence to code a large-scale programming project.

Janice Oda-Ng and Gerald Lau, two people in the department who have always had faith in me.

Curtis Ikehara and Michael-Brian Ogawa for being the inspiration that pushed me onto the thesis path.

David Schanzenbach, George Lee, Vern Wong, BJ Peter Dela Cruz, Michael Claveria, Aaron Kawamura, Lyneth Peou, Pei-Chia Chang, John Wu, and Ka Yee (Alice) Leung for being graduate students who provided me moral support.

Mona Livsey and Alicia Gomes-Figueira for being supportive from afar.

My committee members, Henri Casanova and Scott Robertson, for being on my committee.

My advisor, Edoardo Biagioni, for being able to have so much patience with me week-to-week.

And lastly, but not least, my parents for their support while I finished writing this document.

Abstract

Some bus delays may be inevitable, but a bus tracking system may be able to alleviate frustration with delayed or missed buses. This thesis looks at the feasibility and performance of a decentralized bus tracking system to evaluate whether such a network is adequate for this purpose. A simulator is executed using a timetable data model to answer this question.

A flooding algorithm is a simple method for disseminate data to a wide area. A source broadcasts data to nodes in range, and as nodes receive data, they rebroadcast it. This simple approach consumes unnecessary bandwidth, but improves the likelihood of successful transmission due to redundancy in data sent. This approach is attempted over a random rebroadcast ALOHA network, where nodes may transmit at any time with the risk of collision.

This algorithm is applied to a wireless network representing the bus system in Honolulu. Bus stops in the major areas are used as a starting point to build a mesh network to cover the city. Buses are set up to travel between the stops and broadcast their location at periodic intervals. The performance of the network is evaluated in terms of utilization and transmission reliability.

Findings include bandwidth availability being a significant factor affecting reliability of transmissions. Abundant bandwidth means that flooding is a viable approach to broadcasting data to a wide area. With limited bandwidth, flooding can still be considered as an option to broadcast data with reduced reliability.

Table of Contents

Acknowledgments	iv
Abstract	v
List of Tables	viii
List of Figures	ix
1 Introduction	1
1.1 Honolulu’s Current Bus System	3
2 Related Work	4
2.1 Current Tracking Systems	4
2.2 Current Technologies	6
2.2.1 Centralized versus Distributed	6
2.2.2 Current Wireless Technologies	7
3 Network Model and Analysis	9
3.1 Model	9
3.2 Network Nodes	10
3.3 Analysis of the System	11
3.4 Node Placement	12
4 The Simulator	14
5 Methodology	16
5.1 Bus Stop Topology	20
6 Results	23
6.1 Network Connectivity	23
6.1.1 Network subset	24
6.1.2 Full Network	24
6.2 Network Utilization	25
6.3 Transmission Reliability	27
6.3.1 Single Bus case	27
6.3.2 Multiple Bus case	28
6.4 Transmission Freshness	29
6.4.1 250kbps case	32
6.4.2 20kbps case	33
6.5 Reliability vs. Range	33
7 Evaluation	38
7.1 Cost	38
7.2 Performance	39
7.3 Limitations	40

8	Conclusions and Future Work	41
8.1	Future Work	42
8.1.1	Smarter broadcasting	42
8.1.2	Change of Media Access Control Layer	42
8.1.3	Comparison to TheBusHEA, the current bus tracking system	42
8.1.4	Other Applications	43
A	The Simulator	44
A.1	Network Nodes	44
A.1.1	Bus Stops	45
A.1.2	Buses	46
A.2	Configuration Files	46
A.2.1	vars.txt	47
A.2.2	stops.txt	47
A.2.3	routes.txt	48
A.2.4	buses.txt	48
A.2.5	nodegen.txt	49
A.3	User Interface	49
	Bibliography	52

List of Tables

<u>Table</u>		<u>Page</u>
2.1	Comparison of various wireless technologies.	8
6.1	Number of nodes generated for various range values for initial 289 nodes.	25
6.2	Number of nodes generated for various range values for full 438 node system.	25
6.3	Average and worst case freshness values for varying bandwidths, 75m range.	33

List of Figures

<u>Figure</u>	<u>Page</u>
2.1 Google Transit, available at http://www.google.com/transit , allows users to zoom in to a street in Honolulu and locate nearby bus stops. Users are able to click on a bus stop to retrieve information about routes serviced, as well as expected arrival times.	5
3.1 State diagram for nodes in the network.	10
5.1 Map outlining the boundaries of the area of study.	17
5.2 Plot of all stops with added nodes, 75m range.	22
6.1 Graph showing number of broadcasts received with a single bus in system.	28
6.2 Graph showing number of stops receiving a given number of broadcasts, 250kbps bandwidth.	30
6.3 Graph showing number of stops receiving a given number of broadcasts, 20kbps bandwidth.	31
6.4 Histogram showing freshness of data at bus stops for each bus in system, 250kbps bandwidth.	34
6.5 Histogram showing freshness of data at bus stops for each bus in system, 20kbps bandwidth.	35
6.6 Reliability of transmissions compared to range from originating nodes.	37
A.1 Bus system simulator, ad-hoc version. Display is showing the downtown area of Honolulu, and the connections between the nodes.	45
A.2 Bus system simulator, Statistics tab.	51

Chapter 1

Introduction

Public transit is known for being a cost-effective means of travel. In exchange for this savings in cost it introduces variability in arrival times to destinations. Unfortunately, this variability is a factor for many people when they are given a choice between public transportation or using their own vehicle to travel between two locations.

For instance, a person riding public transit may want to know when the next bus is coming. This question can usually be solved by using timetable data. However, unexpected circumstances may alter this schedule. For instance, something may suddenly happen to one of the buses on a route, such as an accident or bus malfunction, that would cause buses on a route to become delayed. A bus rider may want to know if the bus they are waiting for will be delayed in which case the rider may wish to change their intended route, if possible. Due to traffic, it is possible for many buses traveling along the same route to get delayed, causing a chain reaction of late arrivals.

There are often multiple ways to get to a destination. In many cases, these multiple route choices involve waiting at different bus stops, often separated by one or two minutes of walking distance. Furthermore, some routes have an express and non-express version that travels to the same destination, and it is sometimes unclear when the non-express route would be faster than the express route. This is because although the express bus almost always travels quicker than its non-express equivalent, the additional time waiting for an express bus could be significant enough to negate the advantages of an express bus.

In many cases, buses need to communicate with each other. This often happens when the transit station needs to contact a driver for various reasons, but could also be extended to notify buses travelling along a route of an unexpected detour. To communicate broadcasts to all operating buses, a wireless network that allows buses to connect to either another bus or a central station would

be helpful. Also, because bus location changes over time, the buses may not always be in range of each other. This property of having neighbors that change is what defines an ad-hoc network.

Wireless ad-hoc networks have been used previously in many sensor applications, such as the status of endangered species of plants [1], temperature, vehicular movement, noise level, and mechanical stress [2]. In this thesis, the goal is tracking vehicular movement in a large area using some sort of sensor to determine its location. A mesh network approach is considered due to the issue of scalability - as the network grows in terms of area coverage or number of buses, nodes can be added, as opposed to a centralized system, where the capacity of the central nodes would need to be increased instead.

Wireless networks have been around for over forty years. Initially starting as the ALOHA network [3], Abramson's paper introduced the idea of communication via electromagnetic waves. Wireless networks are convenient because they eliminate the expense of having to hook up wires to interconnect different components, as well as supporting improved mobility and flexibility [4]. However, this introduces the complication of multiple devices possibly competing for the same channels, which is commonly solved via Medium Access Control (MAC).

Likewise, the idea of mesh networks has been around for a little over fifteen years. In a mesh network, there are many static nodes which are connected and don't move. Mesh networks are largely suited to interconnecting devices in a user's home [4]; however, mesh networks can be extended to larger areas to track objects. The benefit of using a mesh network in this situation is that nodes can be added or subtracted from the network with little or no configuration changes needed from the user. In this application, it is unlikely that nodes will be changed after the network is installed except for network maintenance.

Mesh networking and mobile ad-hoc networking are two technologies that can be used to design a network. In a mobile ad-hoc network, nodes are allowed to move within the network, causing nodes to move in and out of range of each other. This introduces an additional level of complexity since a mobile node's neighbors constantly change as it moves through the system.

This thesis describes a simulation of a wireless ad-hoc mesh network deployed over an urban metropolitan area. The simulation uses the wireless ad-hoc network to track city buses in Honolulu to test the effectiveness of the network. Assuming that Honolulu is fairly representative of a metropolitan area, these results could be extended to other metropolitan areas. Simulations of the physical and data link layers are used to determine the feasibility of the network.

The rest of the thesis is structured as follows: Chapter 2 covers related work and existing technologies that are considered in the paper. Chapter 3 is where the model for the proposed

decentralized system is defined. Chapter 4 discusses how this model is implemented in the simulation. Chapter 5 describes the area considered for the study and the routes included in the simulation. Chapter 6 discusses results obtained with the simulator. Chapter 7 looks at the contrasting system, a centralized approach. Finally, chapter 8 presents conclusions based on the simulator results and discusses considerations for future work.

1.1 Honolulu's Current Bus System

Honolulu has had a public transportation system for over 100 years; however, the bus system didn't flourish until the 1960s, when Frank Fasi developed it during his mayoral term. Since then, it has been recognized for *America's Best Transit System* twice, in 1994-5 and 2000-1.

As of 2009, TheBus [5] sees a daily ridership of over 200,000 riders over more than 50 standard routes on the 531 buses in its fleet, in addition to 30 morning and afternoon express routes. These routes service approximately 4,200 bus stops scattered along streets in Honolulu.

Chapter 2

Related Work

2.1 Current Tracking Systems

Google Transit already does a significant amount of bus "tracking". It allows the end user to view a map of Oahu and choose any bus stop on the island. Google Transit is then able to determine which routes pass through that stop, as well as predict the time where buses will pass through that stop. The issue, however, is that all bus location information stored on this site is based on timetable data. This is adequate for simulation purposes and many normal day-to-day user inquiries, but it doesn't account for real-time events.

Regarding real-time tracking systems, there currently are two tracking methods in Hawaii, both available to the public.

Until recently, the only real-time tracking system available to the public involved calling TheBus's general information phone number, available daily from 5:30 am to 10:00 pm. A rider calls this phone number and asks for the location of the next bus, while also giving their current location. The operator then gets the location from the central office and/or contacts the buses directly and returns to the user. This process can take anywhere from three to five minutes or more, and often by the time the user is notified, a bus has already arrived.

In July 2009, TheBus deployed a website, designed specifically for mobile web browsers, that displays information locations about currently running buses. Called Honolulu's Estimated Arrivals (HEA) [6][7], it displays the location information for each bus in a table by showing previous and expected future arrivals. Current work by TheBus involves numbering each of the bus stops so that users can look up their bus stop on this system. Bus information is retrieved via a GPS device on each bus, and is updated every two minutes through a direct wireless link.

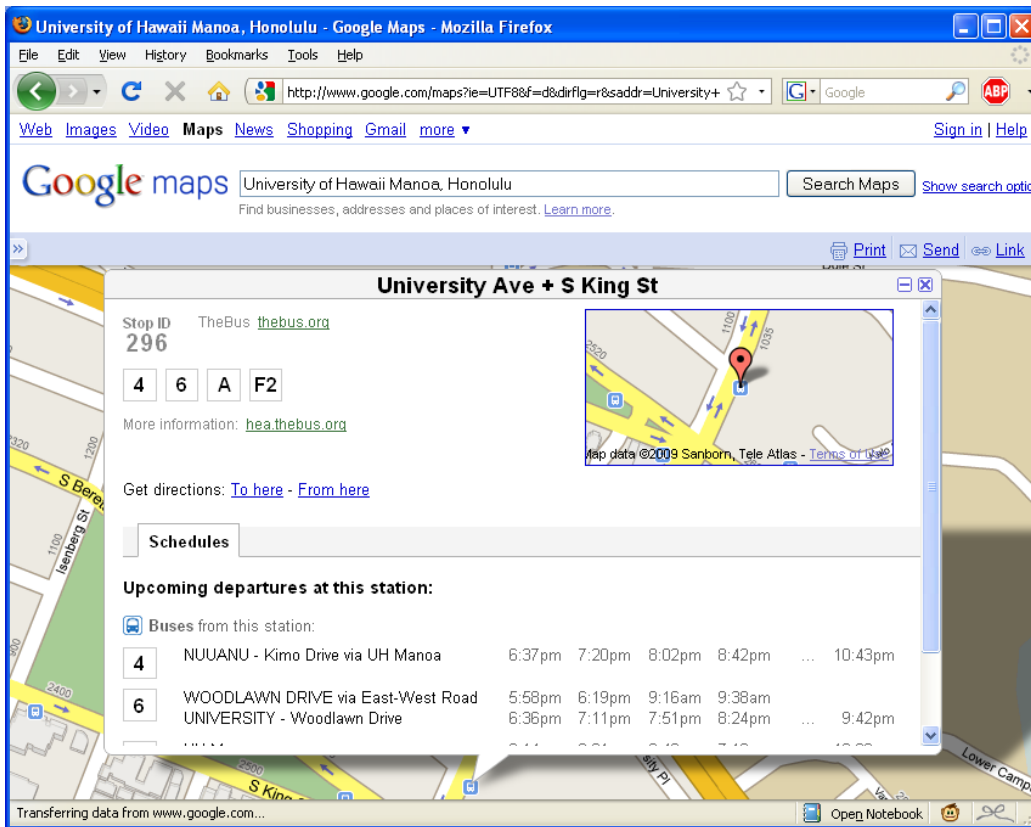


Figure 2.1. Google Transit, available at <http://www.google.com/transit>, allows users to zoom in to a street in Honolulu and locate nearby bus stops. Users are able to click on a bus stop to retrieve information about routes serviced, as well as expected arrival times.

A mesh network hardware implementation of a bus tracking system is in progress at the University of Massachusetts, Amherst (UMass). The Diverse Outdoor Mobile Testbed (DOME) [8] project attempts to deploy a mobile network to riders of the public transit system over seventeen square miles via 200 access points and forty buses, as well as track the location of these buses.

Also, NextBUS [9] is a company that specializes in deploying bus tracking systems in various cities in the United States. It is a centralized approach since all the tracking data is stored in a main data store, then distributed to its various applications.

2.2 Current Technologies

2.2.1 Centralized versus Distributed

This thesis considers two approaches to distributing data, centralized and distributed.

In a centralized network, one or more nodes are considered the ‘central’ portion of the network and controls the flow of data in the network. When data is updated, it is updated to these central nodes. Likewise, all requests for data also go to these nodes.

A disadvantage of the centralized approach is that the number of requests the network can handle is limited to the load the central nodes are able to handle. In order to increase the load capacity of the network, the load capacity of the central nodes would need to be upgraded, or more central nodes would need to be added.

In a decentralized network, no node is considered to be the center of the network. When data is updated, the node with the updated data broadcasts the data to all nearby nodes; likewise when these nodes receive updated data, they rebroadcast the data. Eventually the entire network receives the updated data. Since every node has every data update that is transmitted, it is only necessary to query any node in the system to request data. However, the bandwidth used to rebroadcast may be significant.

Another consideration about the deployment of such a network is that many projects are deployed incrementally - it is often the case that when a project receives funding, the entire system isn’t immediately deployed. In the case of the centralized network, the centralized nodes would definitely have to be part of the initial deployment, and because these central nodes would most likely be deployed at a transit center, it would limit the location in which the initial deployment could occur. A decentralized network, on the other hand, does not have this limitation - its initial test area could be deployed in any high-traffic area of interest. However, this doesn’t necessarily mean that a decentralized system is easier to deploy (and in many ways, it isn’t).

The main advantage of a decentralized network is the added reliability from the possible redundancy of nodes. With a centralized system, a path exists between two nodes through a central portion of the network. However, with a decentralized system, there are often two or more paths between two different nodes. Because of this, if one of the paths were to go down during the operation of the system, requests would simply be rerouted to another available path. In a centralized system, though, the central part of the network would need to be repaired before the network is operational again, or steps would need to be taken to reduce the downtime of the central portion of the network, such as UPS power or backup nodes.

When using a decentralized system, care needs to be taken in choosing a good rebroadcast algorithm. A poor algorithm could lead to excessive rebroadcasts causing unnecessary collisions or infinite loops, or the data not reaching every node in the network. Also, it should be considered whether or not having fewer costly nodes in a centralized system would be better than more cheap nodes in a decentralized one.

2.2.2 Current Wireless Technologies

This thesis looks at three wireless technologies. Each of these technologies possesses the following important physical properties:

- Transmit speed
- Transmit range
- Power consumption

Speed and range are two factors that many people consider in everyday wireless usage. Power consumption in this network is a consideration because there is a large number of nodes, and power at the stationary nodes may not be as readily available as on buses. Table 2.1 shows the transmit speed, range, and power consumption of the technologies that are compared in this study. Values for power consumption were derived by taking values given in Lee et al. [10] by multiplying voltage by the average transmit/receive ampere usage.

- 802.11b (WiFi): 11 MBit/s, 140 meters, 0.72 W. This is the standard WiFi that users are accustomed to. Only the b revision of this standard is considered because it features the largest range of all 802.11 standards, and full bandwidth (54 MBit/s) is most likely not needed.

Technology	Bandwidth	Range	Power Consumption	Cost per node
Wi-Fi	11 MBit/s	140m	~0.72 W	\$23.375
ZigBee	250 KBit/s	75m	~0.077 W	\$6.95
WiMAX	10 MBit/s	50km	?	\$30.60

Table 2.1. Comparison of various wireless technologies.

- ZigBee (802.15.4): 250 KBit/s, 50-75 meters, 0.77 W. Range affects power consumption - 1 mW corresponds to about 250 feet, while 1 watt corresponds to about 3,000 feet. ZigBee is a relatively new technology designed specifically for wireless mesh networks. Because of this, power consumed by this protocol is low compared to other protocols. In exchange, the throughput speed and range aren't as high as other protocols.
- WiMAX: 10 MBit/s, 50 kilometers. WiMAX [11] is a technology specifically designed for long range communications. A broadcaster can easily access an entire city from a centralized access point, making the decentralized network approach trivial. However, WiMAX does consume a large amount of power and has relatively expensive hardware. Also, it is not designed for mesh networks. Question marks represent the power consumption because comparable values were not found.

The purpose of considering these technologies is to look at the current physical capabilities and limitations of wireless transmissions. Although all of these technologies have additional functionality at the MAC layer, the emphasis is on each technology's physical properties, such as bandwidth and range.

Chapter 3

Network Model and Analysis

The focus of this thesis is to design a network to disseminate bus location information over the city of Honolulu. This network should be able to distribute a packet over a wide area, and should be able to distribute it in a fairly timely manner.

3.1 Model

As described in Section 2.2.1, the emphasis of this thesis is considering a distributed ad-hoc approach. It is chosen over centralized because of the scalability issue, as well as the potential higher reliability of a decentralized system since there is no single point of failure. Most projects are not implemented in a single step; rather, they are incrementally built. Both systems can be built incrementally, though the way they would be incrementally developed would differ. A centralized system would most likely be incremental by adding transmitters to buses in the system. In a decentralized system, all buses would likely have a less expensive transmitter and the areas of service for the network would likely be incrementally developed. Also note that both systems would use the same number of transmitters on buses, but the centralized system would require transmitters with a larger range to communicate with the central station.

For purposes of determining if this network is feasible, it would be very costly to implement this system, both in terms of financial cost as well as permissions from the city and county of Honolulu to set up a system. Instead, the network is simulated to evaluate these main questions of interest:

- Do broadcasts reach the entire network?
- If so, how long does a broadcast take to reach the entire network?
- Approximately how much would it cost to implement this system?

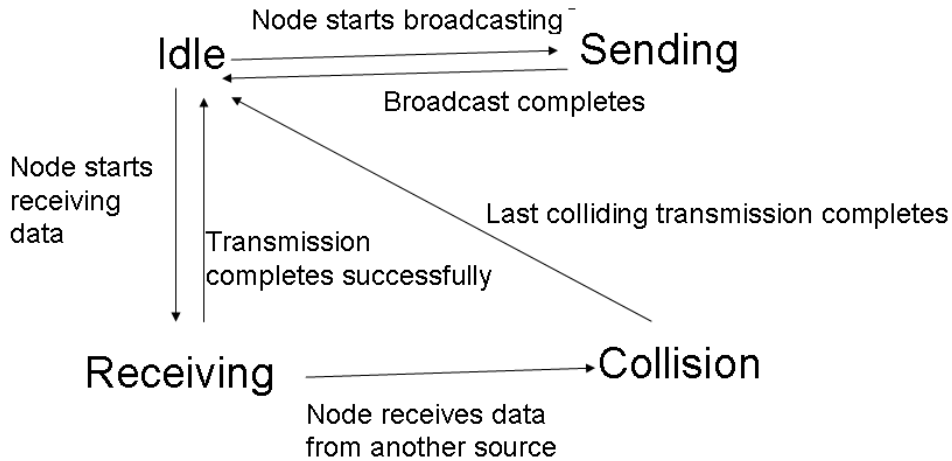


Figure 3.1. State diagram for nodes in the network.

For simulation purposes, it would also be costly in terms of execution time to fully implement each transmission byte-for-byte with processing done for each layer of the OSI networking model. Instead, a simpler approach is taken by making reasonable assumptions about the packets being transmitted, and the feasibility of this system is considered from the physical and datalink layers. This simulation focuses on bit of data being transmitted, using the datalink layer to control retransmissions when packets fail to be delivered.

3.2 Network Nodes

The MAC layer considered in this network is a simple protocol, the ALOHA MAC layer [3]. This is the MAC layer of choice because there is a large number of potential senders and the network is currently fairly randomly accessed. ALOHA is well-suited to this type of network, with also a low level of utilization [12]. Other considerations for different MAC layers are mentioned in Section 8.1.2.

In an ALOHA system, every node can both send and receive data, as depicted by the four possible states in the state diagram in Figure 3.1.

- Idle - The node isn't sending or receiving any data. Upon sending data, it transitions into the 'sending' state. Upon receiving data, it transitions into the 'receiving' state.

- Sending - The node is currently broadcasting. When the transmission finishes, it goes back to an idle state. This state cannot be interrupted since the node would not notice other nearby transmissions while it is sending.
- Receiving - The node is currently receiving data. It stays in this state until the transmission finishes, at which point it goes back to the 'idle' state. However, if the data updates the internal bus information table, the node will rebroadcast and go to the 'sending' state after a random delay. If another transmission is received while in this state, though, the node enters the 'collision' state.
- Collision - The node is experiencing a collision - this happens if it is receiving data from two or more nodes at the same time. This is problematic because partial transmissions aren't useful in any way, and thus all colliding packets are thrown out when a collision occurs. This state can only be waited out, though additional packets received while in this state can prolong the time needed to wait.

Each bus stop node possesses a bus information table. This is the internal storage for each node that stores which broadcasts have been received, what time the packet was generated by the bus, and the X-Y coordinates of the bus. It is stored as an array of entries with one entry for each bus in the system.

3.3 Analysis of the System

To decide if this network is initially feasible, a mathematical calculation with reasonable assumptions is performed.

A maximum of 1000 buses is assumed to accommodate the fleet of 531 buses, as well as future growth. A data size of 50 bytes is assumed to transmit a bus identifier ID with location information, with an additional 50 bytes of overhead, resulting in a total packet size of 100 bytes. If the buses broadcast once a minute, this would result in a total of $100 * 1,000 = 100,000$ bytes per minute, or 1,667 bytes per second. Each node would ideally receive and send each broadcast once from each bus, so this data rate would be doubled. Thus, a node would send and receive roughly 3,333 bytes of data per minute.

Finally, unslotted ALOHA [3] states that in a situation with random rebroadcast such as this, at most roughly 18% bandwidth effectiveness is achieved. Thus, in this system, roughly 6

times more available bandwidth would be needed to account for this, resulting in 20,000 bytes per second of data. This amounts for 160,000 bits per second of bandwidth needed for each node.

Current technologies support this amount of bandwidth, as mentioned in Table 2.1. In a low-power situation, however, the available bandwidth is usually reduced. In addition, 1,000 buses is assumed - Honolulu only has about half as many buses. It would be interesting to see what happens to the system when the amount of bandwidth used exceeds the bandwidth capability of the network, especially if a system the size of Sao Paulo, Brazil is used with a bus fleet of 17,000 buses [13]. The system most likely would not function at full potential, but would probably still function to an extent. However, it is also likely that bandwidth speeds will improve over time, so this system may be feasible in the future.

3.4 Node Placement

One difficulty encountered in creating a dataset is that there is no pre-entered public data available for bus stop data in Honolulu. Google Transit [14] does display bus stop data on its maps; however, bus stop data (as of the time of writing) is not downloadable to an immediately usable format.

A separate program was used to generate latitude/longitude values for these bus stops. Microsoft Streets and Maps 2006 was used to generate these location values. One important feature of Streets and Maps is that it can display the latitude and longitude value corresponding to the position of the mouse cursor; these values were determined and recorded for each bus stop in the list.

To generate the list of static nodes, bus stops were located on the Google Transit site and the corresponding point was found with Streets and Maps. This was done for each bus stop in both directions along each route; however, if two bus stops were in close proximity to each other (as in directly across the street from each other or reasonably close), one bus stop would be excluded.

These points from the existing bus stops comprise the initial set of static nodes. However, these bus stops are not immediately within range of each other. Thus, additional nodes needed to be placed in order to connect these nodes.

Each street within the area is a list of nodes that needs to be connected. These lists are stored in the data set. When the data set is loaded, it looks at these lists and makes sure that each street is connected. If they are not, intermediate nodes are added between them so that they are.

Streets are connected by considering each pair of nodes in the list. The process for connecting two nodes is as follows:

1. Calculate the distance between the two points.
2. Divide this distance by the broadcast range.
3. Round this value up to the nearest integer, and subtract one. This is the minimum number of intermediate nodes needed to connect the two nodes.
4. Linearly interpolate the points along the straight line connecting the two points. For example, if two nodes need to be placed, divide the line into thirds and determine the coordinates of the trisecting points. This is done by performing interpolation on the X, then Y coordinates.
5. Finally, determine if any of the existing nodes can be used instead of a newly generated node. This is checked by comparing if any existing nodes are within close proximity of the autogenerated nodes.

Moving buses are also implemented so that they broadcast their location to the network. Each bus corresponds to one bus in a timetable, traveling along a predetermined route. A real situation would also account for unexpected events, which are not covered in this simulation.

Also, users accessing data in the system are a consideration for this simulation, but were not implemented. This is because the requests could be implemented on a different wireless channel, and thus could run independently of this system.

More details about the area chosen is covered in Chapter 5. However, in general, the major transit centers in town are covered, as well as downtown Honolulu, Waikiki, and the University areas.

Chapter 4

The Simulator

To determine if this network design model is feasible, the network performance is simulated in software. The simulator is written in C++ as an event-driven simulation.

The simulator keeps track of all the buses and bus stops in the system. This includes the bus stops as non-moving nodes, buses as moving nodes, transmission times used as events, and collision handling.

Data transmissions are treated as events in the system. When a packet is sent, an event is generated for the sender as well as the receivers to signify when the transmission is finished. All of these events assume the time it takes for a packet to be sent is known, which is largely based on the bandwidth of the system, as well as data packet size.

The following are the actions handled by the simulator, which follow Figure 3.1.

- Sends can only be initiated from an idle state. Upon the transition from idle to sending, an event is added for the node where it returns to the idle state after the transmission time of the packet elapses, e.g. current time + transmit time.
- Receives also are initiated from an idle state; however, these are triggered from another node. An event similar to a send event occurs when a node starts to receive data. Upon a receive, the node internally notes that the transmission will finish at current time + transmit time; after this time, the data that is sent gets added to the node's data buffer.
- Collisions happen when a node is in a receiving state and another transmission is available. When this happens, any data in the buffer is invalid until the node returns to an idle state. In order to exit a collision state, the last received packet needs to finish transmitting - thus, from the beginning of the first transmit, the collision state lasts until current time + transmit time, and this value is extended anytime a new packet is received while in collision state.

Also, when a transmission is sent, the recipient(s) receive an event with the received data, and when the event completes, the data is available to the node. Likewise, a collision is handled if a recipient receives data and receives a second event, in which case then the data becomes garbled until the last transmit finishes.

A standard cycle in the simulator consists of the following:

- Update the current time. All the buses and stops are sequentially checked for their next event, and the earliest time is set in the next update. A bus's next time is determined by the broadcast time, while a stop's next time is determined by when a transmit begins or ends, or a rebroadcast occurs.
- Process any events for each node based on new time. At least one event will occur because of the way the next time was chosen. Each node may receive a transmission, and this transmission may trigger another send. It is impossible to skip over events because of the way the next chosen time is selected.

The technical details of the simulator are covered in greater detail in Appendix A.

Chapter 5

Methodology

Once the simulator was written, the first issue was inputting bus stop locations for a dataset. This involved plotting nodes along the major streets that run through the city of Honolulu. Many of the street choices are based on the chosen routes.

Ideally, one would simulate the entire island of Oahu. However, creating a dataset of the entire island proved to be time-consuming, largely due to the time spent inputting the routes. Thus, the area of interest was constrained to the following streets:

- Northern boundary: School Street/Wilder Avenue.
- Eastern boundary: University of Hawaii at Manoa/Kapiolani Park in Waikiki.
- Western boundary: Kalihi Transit Center, at the H-1 and Moanalua Freeway merge (Middle Street).
- Southern boundary: Nimitz Highway/Ala Moana Boulevard and Kalakaua Avenue.

Some exceptions were made to accommodate routes, largely if buses exited this area for a short period of time, or if the timetable ends near, but outside, a boundary. In some cases, timetable data was interpolated and routes were cut off once they exited this area of interest.

These are the routes considered in the dataset, as well as the portion that is implemented:

- Route A. Kalihi Transit Center to Sinclair Circle
- Route B. Entire route - Kalihi Transit Center to Kapiolani Park
- Route C. Dillingham offramp to Ala Moana Center
- Route E. Pali Highway offramp or Punchbowl onramp to Kapiolani Park



Figure 5.1. Map outlining the boundaries of the area of study.

- Route 1. Kalihi Transit Center to King/University
- Route 1L. Aala Park to King/University
- Route 2. Kalihi Transit Center to Kapiolani Park, including the turnaround loop
- Route 3. Vineyard ramp to Kapiolani/Date
- Route 4. Nuuanu/Kuakini (timepoint) to Waikiki Aquarium
- Route 5. Ala Moana Center to Wilder/Punahou
- Route 6. Pauoa (entire loop implemented) to University/Maile
- Route 7. School/Kamehameha IV Road to Likelike/School
- Route 8. Entire route - Ala Moana Center to Kapiolani Park
- Route 9. Dillingham offramp to Kapiolani/Date
- Route 10. Houghtailing/School to Nimitz/Sand Island Access Road
- Route 11. Vineyard ramp to Alapai Transit Center
- Route 13. Liliha/School to Campbell Avenue loop in Waikiki
- Route 15. Alapai Transit Center to Ward/Prospect
- Route 17. Pensacola/Wilder to Keeaumoku/Wilder
- Route 18. Ala Moana Center to Dole/East-West
- Routes 19, 20. Nimitz/H-1 to Kapiolani Park
- Route 22. Kuhio/Kalakaua to Monsarrat/Paki
- Routes 23, 24. Ala Moana Center to Monsarrat/Paki
- Route 40/40A. Dillingham offramp to Ala Moana Center
- Route 42. Dillingham offramp to Kapiolani Park
- Route 43. Dillingham offramp to Alapai Transit Center
- Route 52. Dillingham offramp to Ala Moana Center

- Routes 53, 54. Vineyard offramp to Alapai Transit Center
- Routes 55, 56, 57/57A. Pali/School to Ala Moana Center
- Route 62. Dillingham offramp to Ala Moana Center
- Route 65. Pali/School to Ala Moana Center, including Forrest Ave. loop

Also, the following remaining regular routes were not considered because they do not travel through the area of interest, or only stop at the transit center in the area.

- Route 14
- Route 16
- Route 31
- Route 32
- Route 41
- Route 44

After selecting these routes, the bus stops were placed along these streets by the method described in Section 3.4. Not all bus stops have a corresponding network node, since if two stops were very close together, one of the stops was placed and the other was excluded. After doing this process, a total of 438 stops were placed along these 38 routes.

Timetables were then inputted as buses - each row in a given timetable corresponded to one 'bus' in the system. Because of this, there are 2,638 bus entries in the simulation. This certainly is not the total number of buses in the system - only a maximum of 550 buses are in the system at any one time, since it is the number of buses TheBus owns and operates. This number could be reduced if it is known which bus driver corresponds to which timetable rows. However, this data was not readily available. The simulator also ignores any bus while the current time is outside the period when the bus is running its route.

All timetable data for the given routes was converted to 'buses' in the system - a weekday that was not Tuesday was chosen since route 22 does not run on Tuesdays. A full cycle for the timetables was considered, ranging from midnight until midnight on the next day, although some buses ran for a couple hours after that. However, the main emphasis is during the two rush hour periods during the day:

- Morning rush hour, from 5:00 am to 9:00 am
- Afternoon rush hour, from 3:30 pm to 6:30 pm

5.1 Bus Stop Topology

It is worth mentioning some details and features of the topology of the static nodes, since it has an effect on the performance of the network. This includes the size, potential bridges/bottlenecks, and number of redundant paths in the area (e.g. how much it looks like a 'mesh', or grid).

The network is largely linear - the north-south distance from School Street to Nimitz Highway is approximately one mile, while the east-west distance from Kalihi Transit Center to East-West Road at the University is 5.5 miles. Also, the distance from Kapiolani Boulevard to Kapiolani Park is 2.0 miles. This makes the network fairly simple to flood a broadcast since it would simply travel in a line, but also limits the redundancy of the network, since a square mesh network would possess many path options for a node.

A visual graph of the area can be seen in Figure 5.2, with each dot representing a node. This figure includes placed nodes for a 75m range, to show where streets lie in the system. Important features of the network should be noted:

- Kalihi area (northwest, above $Y = -1000$): This area looks very close to a mesh network. There are two dead ends on this portion of the map - at the end of Vineyard Boulevard, and the end of Nimitz Highway once it runs underneath the H-1 Freeway.
- Downtown area (origin): This area is also very much like a mesh network. One important point to note is that the area around Aala Park (around -100, -400) is a bottleneck - it is the only path that exists between the Kalihi area and the rest of the map. Also, the end of Hotel Street at Richards is a dead end (500,500), and although nodes could be added along Richards Street, no bus stops are along this street.
- North of Downtown (northeast of 0, 0): There are a few dead ends here - up Nuuanu Avenue, Pali Highway, and Queen Emma Street. Queen Emma becomes Lusitana Street which leads to the turnaround loop for the route 6 bus.
- Remainder of town (X from 0-4000, Y from 0-3000): This is also a fairly good mesh network. A couple side streets aren't fully placed (Ward Avenue, Pensacola Street), but otherwise it has a good amount of redundancy.

- University area (northeast at 5000,1000): This area contains a loop, but two paths out of the loop. Only the part that continues up University Avenue is a dead end.
- Waikiki area (southeast of 3000, 3000): Two paths travel through Waikiki (Kalakaua and Kuhio Avenue), and the path ends at two loops: one at Campbell Avenue (route 13), and one around Kapiolani Park (route 2).

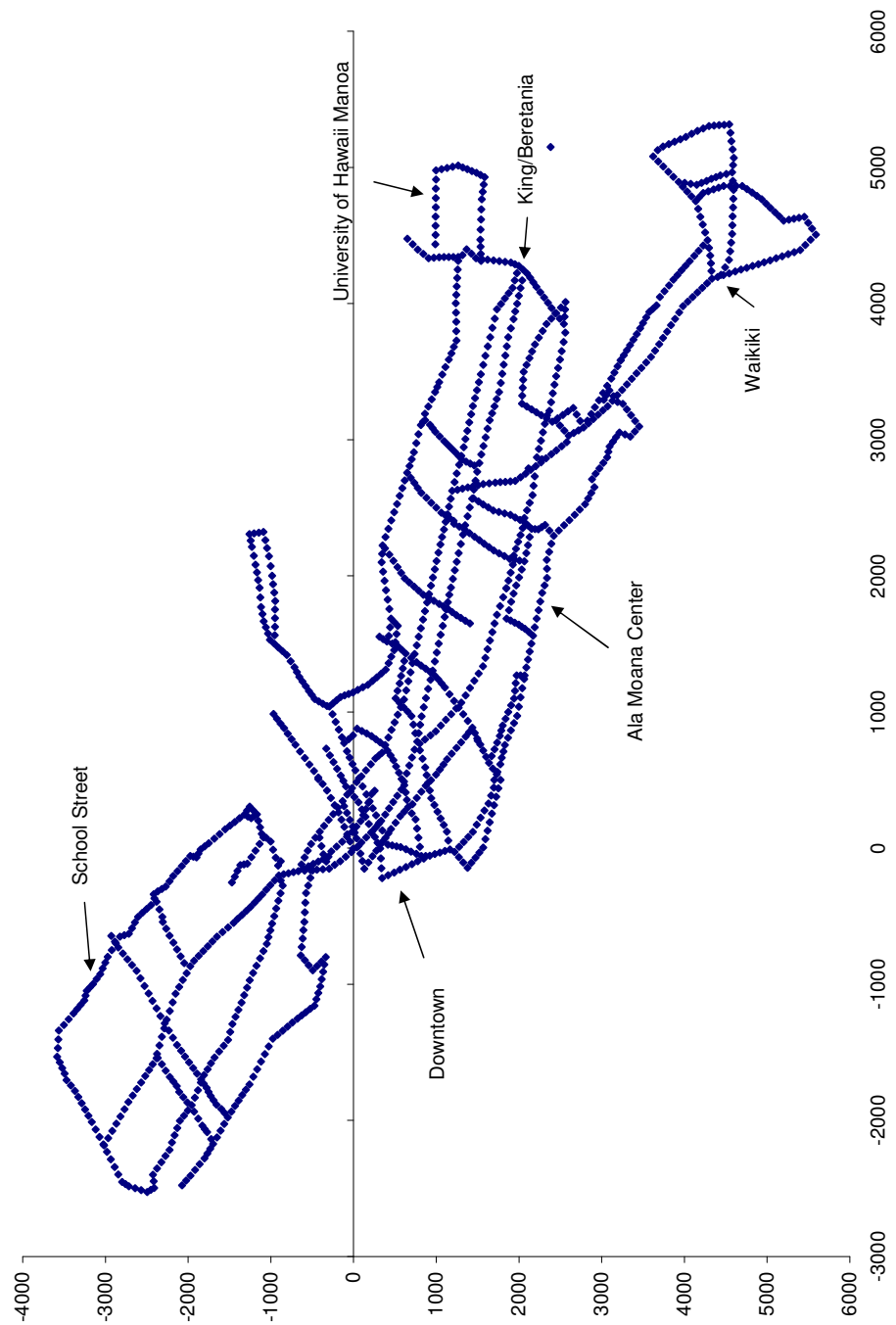


Figure 5.2. Plot of all stops with added nodes, 75m range.

Chapter 6

Results

The goal of this thesis is to determine whether flooding using an ALOHA rebroadcast algorithm is sufficient to distribute data for tracking city bus locations. To determine this, a few metrics are considered:

- Network Connectivity. Given the initial configuration of bus stops in the system, how many additional nodes are necessary in order to fully connect the stops?
- Network Utilization. Given the bus stops and all the buses running, how much of the available bandwidth is being used by the system?
- Transmission Reliability. When a bus broadcasts, how many of the other nodes in the system receive the given broadcast?
- Transmission Freshness. When an entry is stored in a node's bus information table, how out-of-date can the bus info entry become?

6.1 Network Connectivity

The first question in determining whether this flooding approach is acceptable. In order to consider flooding or any other networking method, all of the nodes need to be connected, either directly or through other nodes. Without these interconnections, a broadcast would only be able to reach a portion of the network.

In order to determine the connectivity of the network, the stops were loaded into the simulator without placing any additional nodes. Connections between the nodes are shown on the simulator display with lines between the nodes. From this display, network connections within the

network can be seen. The range parameter of the network can be adjusted to change the proximity required for nodes to be within range of each other. It was determined that a range of 400 meters was necessary to ensure every node had a path to any other node in the network, without additional nodes. It should be noted that with this range, there isn't any ensured redundancy since there may still be only a single path between two nodes in the system.

Node paths were created containing a sequence of nodes that need to be connected in a particular order, in order to accommodate the likely possibility that the network is not initially connected. These paths are selected by listing sequences of nodes along streets with bus stops in the system. A sequence exists for every street in the system using the file described in A.2.5. The simulator reads these paths and generates additional nodes between the nodes along this path to connect the nodes at the bus stops, as described earlier in Section 3.4. However, the simulator also takes into account the possibility that an existing node may already be nearby (on a different path), and skips the generation of a node.

For node placement, two cases are considered: a smaller subset consisting of routes that mostly traveled near the university, and the full system of all streets included in routes in the given area.

6.1.1 Network subset

For the range of 802.11b (140 meters), 244 additional nodes were needed. For ZigBee's range of 75 meters, 594 additional nodes were needed. If the range is further reduced to 50 meters, the number of additional nodes nearly doubles at 957. Values for other range values are given in Table 6.1.

6.1.2 Full Network

Afterwards, nodes are placed in the full 438-node system for a few likely 802.11b and ZigBee ranges. The results are given in Table 6.2.

The most important thing to note here is that the ratio of new nodes to initial nodes does not significantly change in either system. One explanation is because the full system contains approximately 2/3 of the partial system. The other explanation is that bus stops within the system are most likely spaced with a consistent distance between them within town.

In order to deploy a network based on existing bus stops, the number of nodes would need to be a little more than quadrupled for 50m, tripled for 75m, and between doubled and tripled for

Range (m)	New Nodes	
10	5298	
25	2041	
50	957	+331%
75	594	+206% (ZigBee)
100	414	+143%
125	295	
140	244	802.11b
150	211	
175	168	
200	134	
250	68	802.11n
300	38	
350	22	
400	16	
500	4	

Table 6.1. Number of nodes generated for various range values for initial 289 nodes.

Range (m)	New Nodes	%
50	1484	+339%
75	913	+208%
100	621	+142%

Table 6.2. Number of nodes generated for various range values for full 438 node system.

100m. For the remainder of the experiments conducted in this thesis, 75m is used to accommodate the upper range of ZigBee.

6.2 Network Utilization

To determine utilization, two values need to be calculated. First, the amount of data transmitted by each node - both sent and received data - is counted. Secondly, the amount of bandwidth available to each node is counted. The ratio of the data transmitted divided by the bandwidth of the medium is the utilization percentage. For example, suppose a network node has a bandwidth of 1000 bits per second, sends an average of 20 bits per second, and receives an average of 80 bits per second. The total amount of sent and received data is $20 + 80 = 100$ bits per second, and the percentage is $100/1000 = 10\%$ utilization.

Collision rate is determined by counting the total number of collisions and successful transmissions. The number of collisions is divided by the sum of collisions and transmits, and is converted to a percentage. For instance, if there are 10 collisions and 90 successful transmits, the collision percentage is $10/(10+90) = 10/100 = 10\%$ collisions.

An initial simulation was performed without doing any rebroadcast, and the data packets did not travel a significant distance - they only traveled an average of twenty-five nodes before a collision, which was determined by a roughly 4% collision percentage. Thus, a simple rebroadcasting algorithm was added to the nodes. This rebroadcasting algorithm tells the node to rebroadcast after a random interval that is a multiple of the total transmission time of the packet. For simplicity, it currently triggers when the sender detects a collision, though in reality a sender would not be able to sense this. In a more realistic simulation, it would trigger when the node detects that a transmission did not go through when it fails to receive a passive acknowledgement, e.g. when it fails to over overhear a neighbor forwarding the packet. In any case, implementing a rebroadcast algorithm improved the number of successful transmits, and thus distance a broadcast travels, in the system.

Runs were performed at 75m range, 3:30 to 4:00 pm, using all stops and buses. Values representing the ZigBee spectrum for bandwidth were used, from 250 kbps down to 20 kbps, to determine if bandwidth available played a significant role in utilization. Both cases used the same dataset for stops and buses, so the number of broadcasts is constant: 3,277. This comes out to 1.8 broadcasts per second, or 109 broadcasts per minute, which is the average number of buses in the system during this period. Also, the initial broadcast time for each bus is the same for all cases.

For the highest bandwidth case (250kbps), the simulation ended with a utilization value of 1.35%, though it did peak at 1.38% at 3:52 pm. It is interesting to note that the utilization does fluctuate within the first minute - this is due to the time it takes for a signal to reach the entire network, as outlined in Section 6.3. Also, there were 130,801 collisions out of 10,737,983 total transmits, resulting in a 1.22% collision rate.

For the lowest bandwidth case (20kbps), the simulation ended with a utilization value of 6.43%. It did peak at 6.49% at the same time as the previous simulation (3:52 pm), so it is possibly due to both cases having the same initial broadcast times, as well as same positions. Also, there were 160,532 collisions out of 4,182,838 total transmits, resulting in a 3.84% collision rate. Although there were only 22.7% more collisions, the total number of transmits in the system is less than half of the previous case, which significantly increased the collision rate.

Intermediate values (40kbps and 100kbps) produced utilization values between the other two cases, as expected. 40kbps had a utilization of 5.01%, while 100kbps was utilized 2.91%.

6.3 Transmission Reliability

To determine if the information gets to its intended destination, the bus information tables are checked to see if they received any data regarding the buses in the system. The simulator provides a function to output the bus information tables to a text file which lists each bus stop with the buses it possesses information about, as well as the time of the last packet received from each bus. These text files are processed to determine which stops know about which buses.

6.3.1 Single Bus case

A simple test is done first to determine if broadcasts are being disseminated through the entire system. The bus stop data with generated nodes is kept, and a single bus in the system is kept. The bus broadcasts and the bus information tables are checked every second after the broadcast. Also, to ensure other broadcasts don't interfere, the transmission interval is set to a length much greater than the actual time to broadcast. At the time of broadcast, the bus was on King street between Bishop and Alakea streets in downtown Honolulu.

Seven stops did not receive this broadcast:

1. Kapiolani/Waiialae. This is to be expected, since this stop was added to accommodate the route 1 timetable. This stop is not a part of the node paths, and therefore is not connected to the network.
2. Old Pali/Mamalahoa. This was also expected for the same reason as above.
3. Dillingham/Middle. This stop should have received the broadcast - it is possible that both neighbors may have transmitted at the same time on both the initial broadcast and rebroadcast.
4. Punahou/Dole. This stop also should have received the broadcast - in this case, though, the stop is very close to a pair of stops at the intersection of Punahou and Wilder.
5. Three autogenerated nodes.

It took a total of 68 seconds for a broadcast to disseminate through the network from Downtown - from 3:54:24 PM to 3:55:32 PM. A graph showing the number of nodes that received the data over the 68 seconds is shown in figure 6.1. It is interesting to note that the slope is fairly linear until it tapers off after most of the system has received the broadcast.

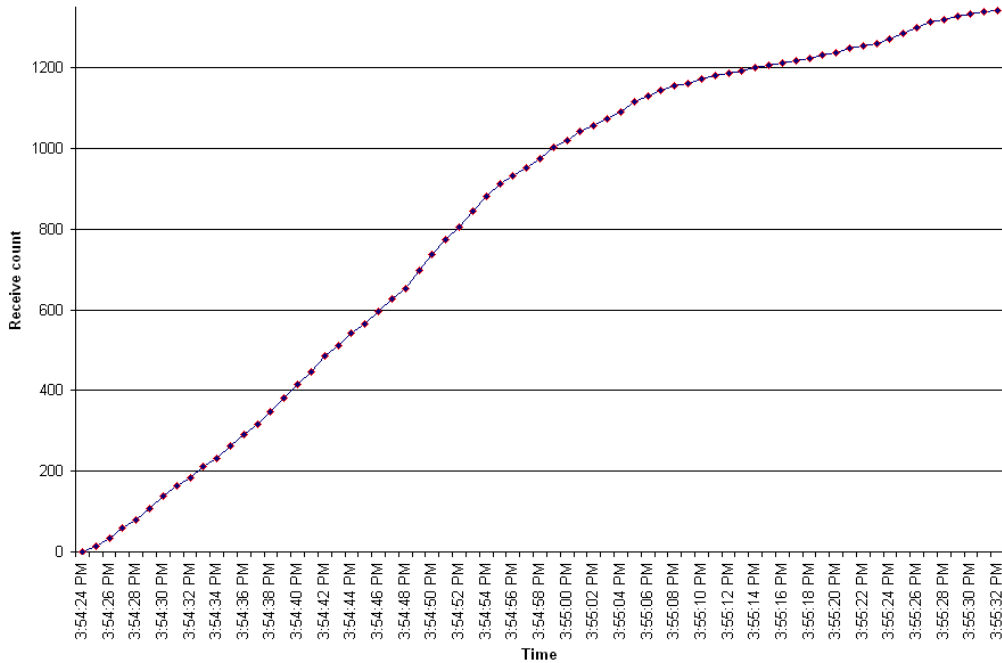


Figure 6.1. Graph showing number of broadcasts received with a single bus in system.

The simulation was continued past the second rebroadcast, and all the nodes received at least one of the two broadcasts. Therefore, without interference from other buses, transmissions are able to flow through the entire system.

6.3.2 Multiple Bus case

The simulation was then executed from 3:30 pm until 4:00 pm, using the aforementioned parameters: 75m range, 1 min broadcasts. The same bandwidth values were tested as in the previous simulations. For this 30 minute period, a total of 3,277 broadcasts (as before) were performed by the 191 buses that performed broadcasts that were received by at least one stop.

For the 250kbps case, the bandwidth was adequate enough so that every stop had location data on nearly every bus in the system. In the worst case, a stop received broadcasts from 188 out of the 191 distinct buses (98.4%). It is interesting to note which stops did not receive all of the bus data:

- 188 receives: Route 6 loop in Pauoa. This is apparently because there is only a single path for broadcasts to travel to get to this loop. All 40 stops with 188 bus entries are along this loop - existing bus stops along with added nodes.
- 189 receives: Latter half of Kuhio Avenue in Waikiki, a few stops in the Route 6 loop, one stop in the Campbell Avenue loop.
- 190 receives: Other stops in Waikiki, two stops on Beretania Street, some stops on Nimitz Highway.

The stops with less than perfect reception do appear to be at the borders of the map, especially the route 6 loop and the Waikiki loop. Therefore, this result is not surprising. A distribution of the counts for this case can be seen in Figure 6.2. To improve this, additional redundant paths should be considered to avoid bottlenecks such as this.

The 20kbps simulation was certainly a lot more interesting - with less available bandwidth, not all the transmissions were able to completely flood the network. In the best case, a stop was able to receive information about 178 buses out of the 191 in the system (93%). In the worst case, 151 transmissions were received by a single stop (79%). On average, a stop had information about 169.76 (88.9%) of the buses in the system. A distribution for this case can be seen in Figure 6.3. Surprisingly, the nodes with the most receives were situated around the Ala Moana area - this may be simply because many routes operate at or near the shopping center.

It should be noted that these percentages only state that the stops received at least one broadcast - it is possible that a stop received a broadcast at the beginning of the simulation and never receives another one for the rest of the simulation.

6.4 Transmission Freshness

From the previous case, it is known that each bus stop knows about most of the buses in the system. However, how up-to-date is the information that the bus stops have at the end of the simulation?

The contents of the bus information tables were analyzed. However, an added difficulty exists, since buses can enter and leave the system - and thus, it is possible that a bus may leave the system in the middle of the simulation. Therefore, to decide how old data is within the system, the time a bus exits the system needs to be considered.

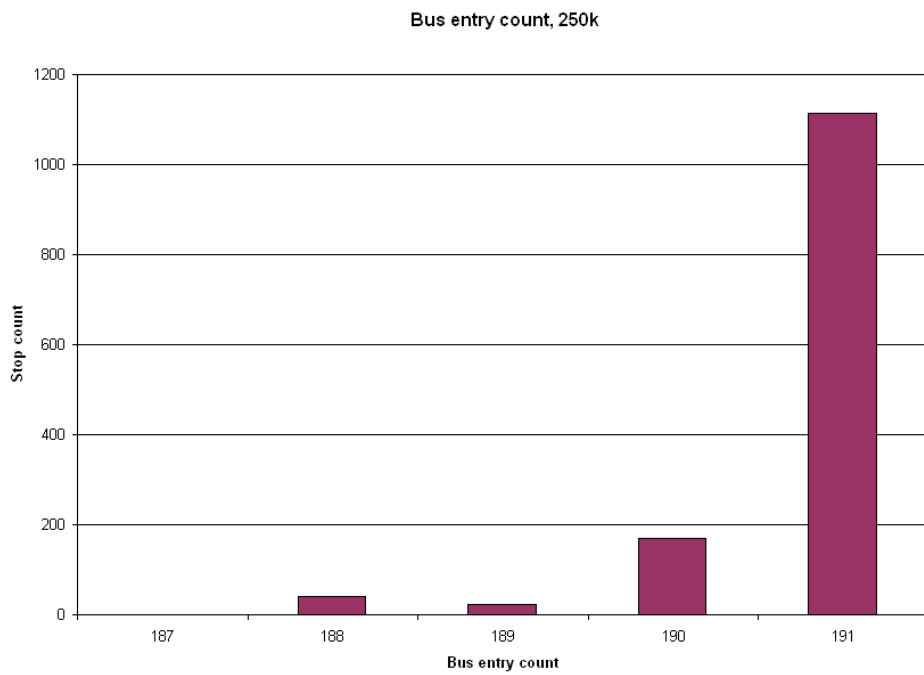


Figure 6.2. Graph showing number of stops receiving a given number of broadcasts, 250kbps bandwidth.

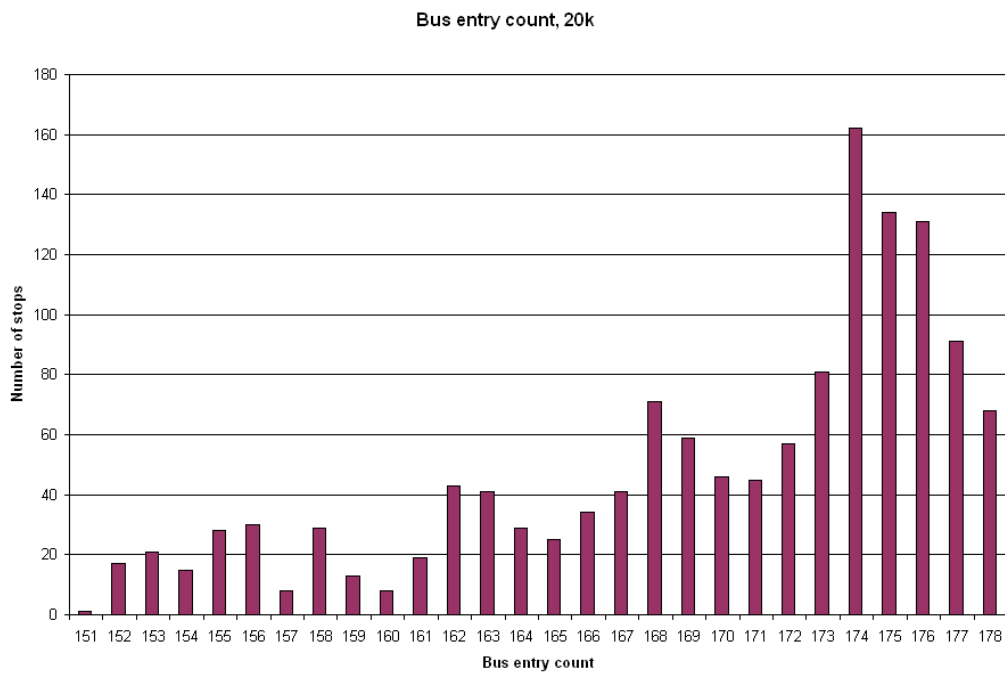


Figure 6.3. Graph showing number of stops receiving a given number of broadcasts, 20kbps bandwidth.

Other studies estimate an acceptable wait time for a bus to be approximately 5.3 minutes [15]. Although this is a slightly different measure from what is being looked at here, a value of 5.3 minutes will be used as a value before a person waiting at a stop wonders where their bus is, and thus how out-of-date location information is allowed to be.

To determine the freshness of the data, the broadcasts for each bus are considered. Each stop keeps track of received broadcasts, as well as the time of the last broadcast received for any given bus in the system. The average of the times for the last received transmission for each bus and each stop in the system is computed. For instance, suppose the system contained one bus with three stops. The three stops contain broadcasts from the bus from 3:53.24, 3:54.24, and 3:56.24. Each time is converted to number of minutes past an arbitrary time (in this case, 3:00) to the values 53.4, 54.4, and 56.4; then are averaged to get the resulting value of 54.73.

However, the time in which a bus exits the system needs to be considered, since if a bus leaves the system at 3:40, there will obviously be no broadcasts between 3:40 and 4:00. In the above case, if the bus left the system at 3:57, the last broadcast would be at 3:56.24, and the average freshness for stops that received the transmission would be $56.4 - 54.73 = 1.67$ minutes.

6.4.1 250kbps case

The average freshness for the 250kbps case is very acceptable - the average within the system is 0.27 minutes out-of-date. This means that if a bus stop is queried for a bus in the system, the data would on average be 0.27 minutes old, or 16.5 seconds. (It is interesting to note how this is also significantly faster than the 68 seconds from the single bus case - this is largely due to the greater amount of bandwidth available here.) If a bus travels 10 miles per hour, this translates to about 0.046 miles the bus has traveled, or 74 meters, or approximately one node in the system. Because an average of 2 nodes were added for each existing bus stop, this 5 node distance translates to about one to two bus stops.

Surprisingly, the worst freshness of the buses in this system occurred on route 1 buses, largely because the route endpoint at Kapiolani/Waiialae isn't connected to the rest of the network, so the last few minutes before a route 1 bus exits the system, it isn't in range of other nodes. Excluding these buses, the worst freshness was 3.108 minutes.

It should be noted, though, that even if the data is not completely fresh, the packets are still timestamped. This means that even if an entry is retrieved that is five minutes old, its location may be fifteen stops away, so the bus may be expected to arrive when the packet is retrieved.

Bandwidth	Average	Worst Case
20kbps	2.866 min	15.21 min
40kbps	1.243 min	6.45 min
100kbps	0.380 min	3.21 min
250kbps	0.275 min	3.14 min

Table 6.3. Average and worst case freshness values for varying bandwidths, 75m range.

6.4.2 20kbps case

However, when the bandwidth is greatly reduced, the freshness of the bus data suffers. On average, bus stops contain data 2.87 minutes out of date with the bus broadcasts. This translates to 0.478 miles, 765 meters, 10.20 nodes, or 3-4 bus stops in this system. Although it is not nearly as desirable as the previous case, it is usable since a user would still have data about an upcoming bus.

The worst latency was 15.21 minutes, which was a route 6 bus. This bus ended the simulation at 4:00 pm within the loop, which was mentioned earlier as having only one path to the loop. It is very likely that the transmission, upon trying to exit the loop, collided with other transmissions trying to enter the loop, causing the data to never exit the loop. Prior to this, the bus was within the downtown area, so those transmissions were able to get through. However, once it entered the loop with a single exit, these earlier transmissions were never replaced, resulting in a high delay in value transmission.

Also, of the 191 buses, 26 of the buses had an average freshness of greater than 5.3 minutes among all bus stops in the 20kbps case.

Histograms for both cases can be seen in Figures 6.4 and 6.5. Also, a table with values between these two cases can be seen in Table 6.3.

6.5 Reliability vs. Range

One observation is that as a signal travels farther from the original broadcasting node, the chance of collision increases. How far can a signal travel before it is unlikely that the signal reaches additional nodes?

For this case, only the 20kbps case was considered with a 75m range. The entire thirty minutes of simulation were executed; however, the last few minutes of the simulation is the focus for this case. Since the buses broadcast once per minute, each bus broadcasts exactly once between 3:58 and 3:59. Also, from Section 6.3, it takes 68 seconds for a single broadcast to flood the

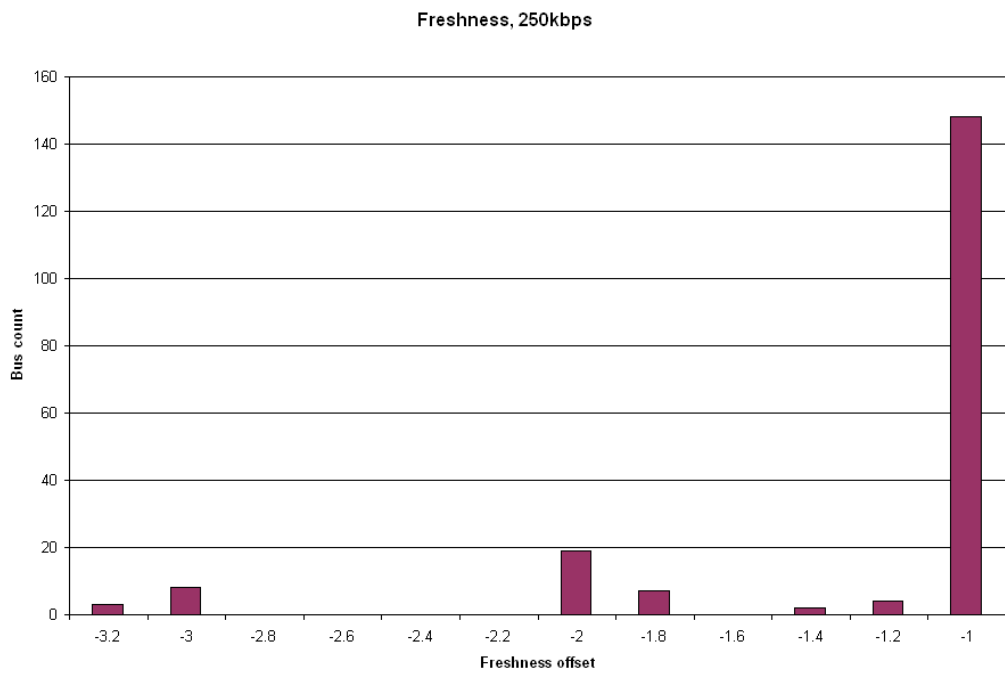


Figure 6.4. Histogram showing freshness of data at bus stops for each bus in system, 250kbps bandwidth.

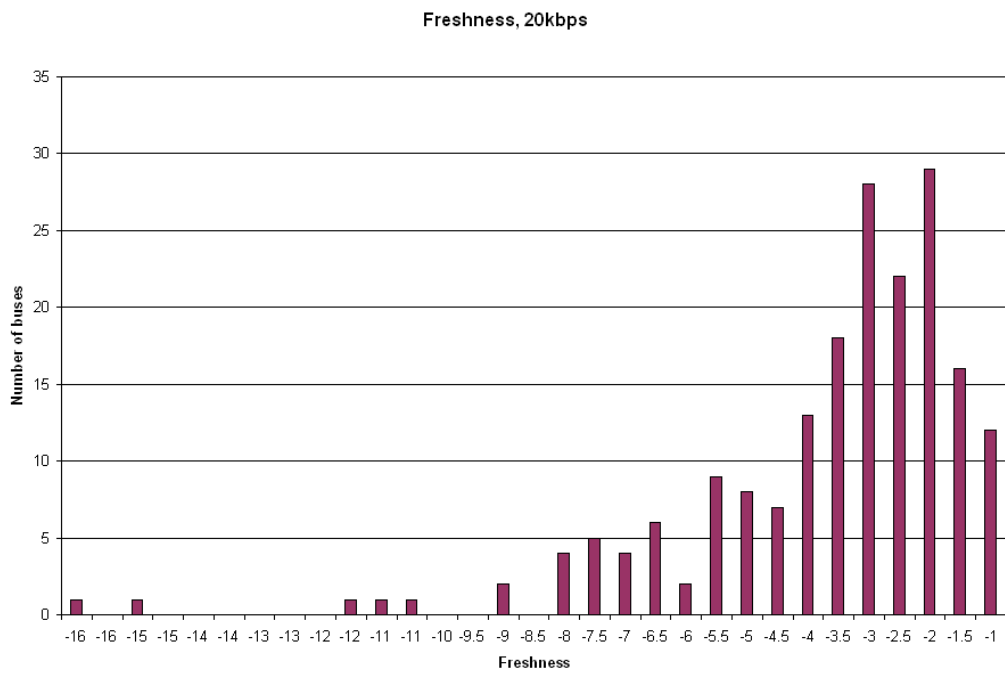


Figure 6.5. Histogram showing freshness of data at bus stops for each bus in system, 20kbps bandwidth.

entire network at this bandwidth value. Thus, this broadcast in most cases should (assuming no interference) reach the entire network by the end of the simulation at 4:00.

The following is how reliability based on a range from originating broadcast is determined:

1. Determine where each bus is between 3:58 and 3:59, at the time the broadcast was executed.
2. For each bus, find all bus stops within range of that bus.
3. From that list of bus stops, determine the percentage of bus stops that received an updated packet from that bus. In this case, the simulation only executed until 4:00, so a bus stop received an updated packet if its bus information table has location information for that bus timestamped at either 3:58 or 3:59.
4. Repeat this procedure for each bus. This results in a reliability % for each bus.
5. Take the average of all the buses. This is the overall % reliability for the system.

A graph of the results can be seen in Figure 6.6. The interesting note here is that this graph is exponential - for each additional 100m, the percentage of received transmissions is 98-99% of the previous value. This suggests that each transmission is independent of the previous ones. Also, for small ranges (500 meters or less), over 90% reliability is achieved, although this value never reaches 100%.

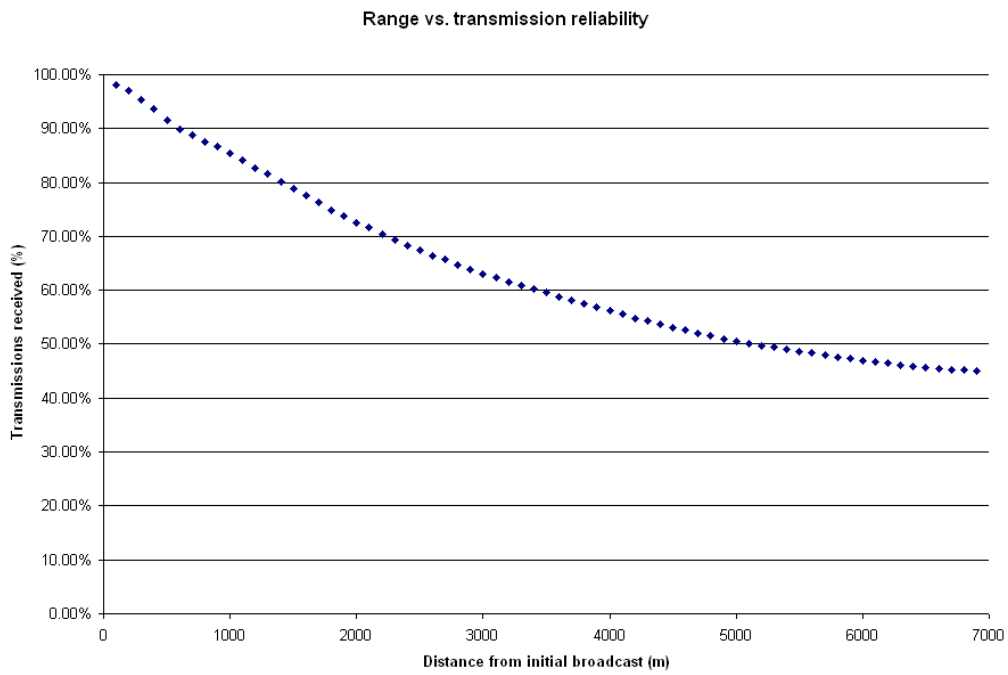


Figure 6.6. Reliability of transmissions compared to range from originating nodes.

Chapter 7

Evaluation

One large question that remains is if this approach is better than a centralized approach. In order to do a comparison, a model needs to be established for a centralized system.

For the centralized system, it is assumed that there is also a set of moving (buses) and non-moving (bus stop) nodes. However, the number of non-moving nodes is much smaller in a centralized system, and possibly only at major locations, such as transit centers. It is also assumed that there exists a mechanism to retrieve location information from these locations such as a web page.

7.1 Cost

Two factors are considered for the cost: transmitters for the buses, and transceivers for the bus stops.

For a centralized system, the advantage is that there aren't as many non-moving nodes in the system. The disadvantage, though, is that because there isn't a mesh network in the city, there is no way to relay a message from a nearby stop to the central station. Because of this, the range of each transmitter would need to be large enough to transmit directly to at least one central node on each transmit. This can add significantly to the deployment cost since the transmitters would need to have a large range.

It is also possible to get around this limitation by using a cellular network to transmit data. This would reduce the need to have a large enough transmitter, but does add the upkeep of having to provide a cellular network, which would need to either be deployed at an additional expense, or contracted out to another company at an additional periodic charge.

To estimate the cost of deploying either system, the former (non-cellular) system is assumed. Also, the following assumptions are made:

- WiMAX is used for the centralized system, since it is the most comparable technology that supports the given range.
- A total of 1,000 buses is assumed to accommodate the future growth of the bus fleet.
- For the decentralized system, a range of 50 meters is assumed - this in effect requires roughly 2,000 static nodes.
- For the centralized system, a single static node at the centralized location is assumed.

A WiMAX transceiver costs \$30.60 from DigiKey, a supplier of hardware online. A ZigBee transceiver by the same brand costs \$6.95. This comes out to a cost of \$30,600 for a centralized system, while the decentralized system costs \$20,850 for the networking components. For upkeep of the system, the ZigBee transceivers could conceivably be solar powered; the WiMAX transceivers could also be solar powered, but most likely would not need to be since they would be situated on buses. The cost of solar panels for the static nodes would introduce an additional cost for static nodes in the decentralized case. It should also be noted that the cost of the centralized system is directly proportional to the number of buses. Thus, for the current bus fleet of 500 buses, the cost would be roughly halved, which would be cheaper than a decentralized system.

7.2 Performance

A centralized system could perform very well if the broadcasts are synchronized to avoid collisions of data. The broadcasts could be scheduled so that each bus transmits at a certain portion of each cycle. Synchronization would need to be maintained to ensure that each bus possesses the same time.

Using the simulation parameters from earlier (100 byte packets), if each bus were to broadcast once per minute, $1,000 \times 100 = 100K$ data every minute. This amount of bandwidth is easily satisfied in WiMAX, at least by the typical case.

The main issue with a centralized approach is handling requests from users in the system. In the decentralized system a user could query a nearby bus stop, but in this system, a request would need to somehow be routed from their device to the central station. This could be a connection over the internet that goes to the central node, or may require some device to connect. This request system could also be done over a separate wireless channel so that it would not affect the broadcasting portion of the system.

In terms of redundancy, multiple central nodes could be placed within the system, and the bus broadcasts would, at least in theory, reach all of these nodes. A user could then access the nearest central node and retrieve their data.

7.3 Limitations

The apparent limitation of the decentralized system is the cost of the static portion of the network. As mentioned in Section 6.1, a significant investment of additional nodes would be necessary to provide coverage to Honolulu.

A centralized system, on the other hand, would be adequate for the given area of study. With a range of 50 km, this is certainly enough to encompass the urban area of Oahu without considering WiMAX's line-of-sight requirement. The length of Oahu is slightly larger than this at 71 km, so a broadcast would likely encompass most of the island, although the entire island being covered is not always guaranteed. Deploying this in a larger area could present a challenge, however.

One consideration to get around these limitations would be implementing a hybrid system: smaller areas covered by a centralized network while these smaller networks are interconnected to each other in some sort of decentralized network. This gets around the disadvantage of the decentralized network requiring too many smaller nodes, and allows the range of a wireless technology to be extended. However, this is outside the scope of this thesis.

Chapter 8

Conclusions and Future Work

It is difficult to determine which system between the centralized and distributed system is better. Each system has its own merits and setbacks that need to be taken into consideration before making a verdict.

The main advantage of a centralized system is that it is much less complicated to implement. One central node (or network) would be used to handle all incoming requests from other buses to keep track of the data. However, with this simplicity, there is always the risk of unexpected downtime when the central network goes down, since no other nodes in the system would be able to handle the transmissions from the buses.

The decentralized system, on the other hand, would be able to adjust in the case of unexpected failure. If a path goes down, as long as there is some redundancy in the network, requests can be rerouted to another portion of the network. The downside of this approach is the cost for all the static nodes in the network, which isn't necessarily trivial since it is necessary to double or triple the number of bus stops in order to have a fully connected network.

The flooding approach is adequate if there are only a small number of buses in the system. If there are more buses, it is apparent that more bandwidth is necessary to make the network to become more reliable. However, even with decreased bandwidth, the network does still function if the data is requested close to the bus in question. This approach may be good for an application where broadcasting nodes frequently broadcast and only nearby areas need to know about the broadcast, since transmission reliability is good for nearby nodes.

8.1 Future Work

8.1.1 Smarter broadcasting

One apparent limitation is that singular path bridges in the system reduce the overall reliability of this network. This could be improved upon in two ways:

First, certain paths could be prioritized for broadcasting. For instance, if a bus along the route A broadcasts, routing algorithms could prioritize the broadcast along the route A route before sending it to other nodes within the system. Secondly, the network could detect bridges within itself and alert someone setting up the network that only a single path exists, and possibly also suggest additional paths that could be placed within the network to improve the redundancy.

8.1.2 Change of Media Access Control Layer

In this study, a simple ALOHA network was implemented using the network topology. Medium Access Control protocols have significantly evolved since then - currently, Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) is used in everyday WiFi. This protocol improves the rate of successful transmits because it senses whether the channel is being utilized before it decides whether or not to transmit or wait.

The other method would be to consider attempting more rebroadcasts upon a failed transmission. The current worst utilization in a run is roughly one-third of the total theoretical utilization for random rebroadcast, so it may be possible for nodes in the network to rebroadcast multiple times to ensure transmissions to go through.

Both approaches have potential for improving the performance of the network presented in this thesis.

8.1.3 Comparison to TheBusHEA, the current bus tracking system

It would be interesting to compare both of these approaches to the current bus tracking system in place, TheBusHEA. It currently tracks the position of city buses via GPS. This is known because on TheBusHEA's web site, sometimes buses will be listed as "scheduled (no GPS)".

It would be interesting to find out technical details about the current system for a few reasons. First, is TheBusHEA distributed or centralized? Does the system perform acceptably, or could it be improved? Secondly, if information about the packets sent could be known, it could be determined whether or not the 100-byte assumption for packets is sufficient. Finally,

information about the wireless technology could be acquired, which would determine whether or not similar wireless protocols are being used for both systems.

No response has been received from the attempts to contact TheBus on the technical details behind this system. However, it is speculated that it is a centralized system since the tracking system has only been in effect for one year, and centralized systems are usually easier to deploy than decentralized ones.

8.1.4 Other Applications

Finally, the extensibility of a network like this should be considered. A bus user may want their route information to be available everywhere in the network to query it, but a nearby accident alert system for cars may be useful, where drivers may want to know if an accident is nearby so they can go onto a side street to avoid it. This could also be useful for emergency response, where EMS drivers would benefit greatly if they do not need to travel on heavily congested streets to reach their destinations.

Appendix A

The Simulator

Development was done using the wx-Dev-C++ development environment [16], which is similar in many ways to Microsoft's Visual Studio IDE. It has a couple of bugs, but it is free, so it is easily obtainable. Unfortunately, the development environment is only released in Windows right now.

The main loop of the program keeps track of the current time in nanoseconds, and allows objects in the system to execute their computation when a transmission succeeds or fails. Because not every nanosecond has an event, it isn't necessary to iterate over every possible nanosecond value. Instead, for each iteration, the loop determines at what time the next event occurs, and skips forward to that time, since the previous times require no change in the network state.

Following are the various types of objects in the system.

A.1 Network Nodes

The basic unit of a network - the network node - is the smallest object in the system. This class handles all the network traffic that is generated via events, either from buses broadcasting or bus stops rebroadcasting. A network node can have one of the following states:

- **Idle** - The node is not doing anything.
- **Sending** - The node is sending data.
- **Receiving** - The node is receiving data.
- **Jammed** - The node is currently experiencing a collision.
- **Idle Wait** - The node is not doing anything, but is waiting to transmit.

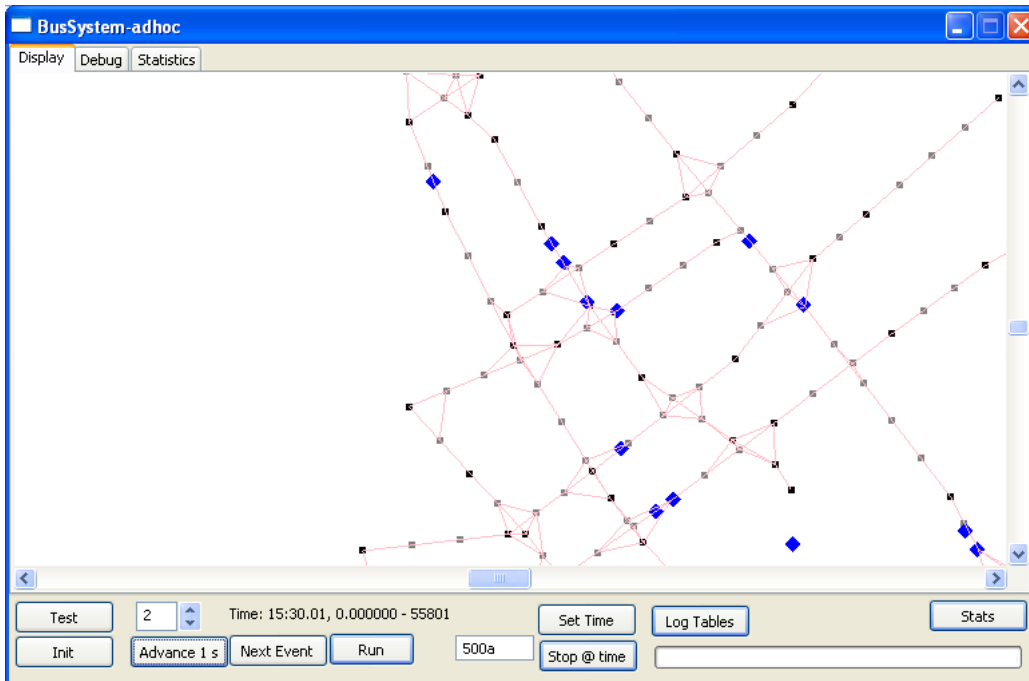


Figure A.1. Bus system simulator, ad-hoc version. Display is showing the downtown area of Honolulu, and the connections between the nodes.

A node can enter any other state from the idle state. While sending or receiving data, a node cannot receive any other data - if it gets a receive request while receiving, it enters a jammed state, and stays in that state until the last transmission finishes. A node will continue to send while another node is transmitting nearby, though. Also, if a node is in an idle wait state but is receiving data when it wants to transmit, the transmission is further delayed. This delay is equal to a time between one and `REBROADCAST_MAX` (currently set to 25) times the transmission time for a single packet, uniformly distributed. When waiting for the retransmission, the packet is queued in the node.

Each of the buses and bus stops in the system uses a network node to communicate.

A.1.1 Bus Stops

Bus stops are shown in the simulator as black squares. Placed (autogenerated) stops are shown as gray squares. Lines show connections between these stops if they are within range of each other.

The bus stops in the system are implemented as a network node with a bus information table as mentioned in Section 3.2. The table is updated as transmissions are received, either by

discarding the packet if the information is old, or updating the information table and rebroadcasting if it is newer.

Aside from this, bus stops would serve the other purpose of handling user requests, but that isn't currently implemented in the simulation.

A.1.2 Buses

Buses are shown in the simulator as blue diamonds.

The buses in the system also contain a network node; however, they don't keep a bus information table, since they don't receive data. However, buses do move within the system, so a method to simulate bus movement needs to be implemented.

When buses travel in the city, it is noted that while in service, they are always traveling between bus stops. In effect, a bus's movement is noted by bus routes, which are paths between existing bus stops. Schedules are modified routes which add a time component - specifying a certain stop the bus will be at in a certain time for various stops - and are the basis of defining each bus's movement in the system.

When the simulator sets a new time, all buses move to their expected position based on their schedule. This is done by finding the closest point before and after the current time, and performing a linear interpolation between these two points to find the expected current position. This is not completely faithful, but is sufficiently accurate for this simulation.

A.2 Configuration Files

A dataset for the simulator consists of five files.

- **vars.txt** - General variables which can be changed to simulate different physical layer constraints.
- **stops.txt** - Information pertaining to static nodes in the system.
- **routes.txt** - Information pertaining to routes taken in the system.
- **buses.txt** - Information about specific buses in system.
- **nodegen.txt** - Information pertaining to paths in system that must be connected.

A.2.1 vars.txt

This file contains variables relating to the simulation, are scalar in nature, and are the values that get changed between runs. This file is unique in the sense that it is the only file that doesn't represent objects; rather, they are different parameters.

The allowed variables in this file are:

- **TransmissionRange** defines the range of each network transmitter on each node, in meters. This also affects the autogeneration process of nodes.
- **TransmissionSpeed** defines how fast wireless signals travel, in meters per nanosecond. This is set to the speed of light, 0.3 meters per nanosecond, and doesn't change between simulator runs.
- **TransmissionRate** defines the bandwidth available in the network, or the rate in which bytes travel. This is defined in megabits per second.
- **PacketSize** defines the size of the data packet, in bytes. This is used to calculate how long it takes to do a transmission after propagation delays.
- **AckThreshold** defines a level of acknowledgement in the protocol, but isn't used at this time.
- **XmitInterval** defines how often buses should do a broadcast of their location, in minutes. It should be noted that each bus's initial broadcast is randomly generated based on this value.
- **StartTime** defines the starting time of the simulation, in number of minutes from 12 am. This also affects the initial broadcast time of each bus.

A.2.2 stops.txt

This file consists of bus stop locations in the system. Each bus stop location is represented by an X-Y coordinate pair where the positive directions are represented by the east and south ordinal directions. Each line in this file consists of three values:

Description, X-coordinate, Y-coordinate

The description is not used in the simulation. It serves namely for debugging purposes, and is usually supplied a value similar to the way stops are referred to in Honolulu: major/minor street name. Each intersection consists of two streets, so those two streets are used. The bus is also traveling along one of those two streets to stop at that stop; this is the major street in this pair.

(Note that, for example, Punchbowl/King is **not** equivalent to King/Punchbowl.) Also note that the description field cannot contain a comma, or else it would signify the end of the field.

A.2.3 routes.txt

This file consists of lists of stops which buses will travel between in order to 'simulate' their movement. The syntax is as follows:

Description, first-node, second-node, ..., last-node

Each node is a number greater than zero, and refers to a line in stops.txt. Also, a number may be preceded by an asterisk (*) - this means that this stop is also referred to as a 'timetable point', meaning there is a specific time the bus will be at this point. It is also a point that the other non-timepoint stops are interpolated against in order to find intermediate positions in the system. Also, the first and last nodes should always contain asterisks.

Note that timetable data is not located in this file. This is because each bus corresponds to one entry in the timetable.

Description is also optional, but serves as a user note when inputting routes. Because it is possible for a single route to have multiple start and end points, this field is helpful to avoid confusion when editing routes later.

A.2.4 buses.txt

This file consists of buses that correspond to the existing routes. While a route says where to go, a bus associates a time factor to it. The syntax is:

description, route, timepoint1, timepoint2, ...

Route corresponds to a row in route.txt. Each of the timepoint values corresponds to a starred value in the corresponding route data file.

To determine intermediate positions, an interpolation process is performed. The timepoints before and after the current position are considered. The distance along the route between these two points are calculated, and compared to the time-wise fraction between the two points. This is converted to a distance, and the point which is that distance from the earlier timepoint is the interpolated position.

A.2.5 nodegen.txt

This file was added to perform the task of auto-generating nodes, and is intended to contain paths of single streets that should be connected. Unlike a bus route which can skip some nodes on the path, this file must list them in order, since skipping nodes may cause additional unnecessary nodes to be generated.

The format is a list of whitespace-delimited numbers corresponding to bus stop nodes. Since this file is parsed using multiple fscanf calls, a delimiter is needed to specify the end of a list - in this file, the sentinel value -1 is used.

A.3 User Interface

The user interface isn't overly complicated - it consists of a visual display of the system with control buttons. The interface controls are as follows:

- **Tabs.** Three tabs are shown: Display, Debug, and Statistics. Display shows a visual display of the current network. Debug shows useful debug messages regarding the network when executing - like when a node broadcasts, successfully receives a broadcast, or is jammed. Output on the Debug tab is disabled when executing simulations, though, due to the amount of time I/O takes. Statistics is used to retrieve statistics regarding the network, but stat collection mode needs to be enabled first.
- **Network Display.** The window shows the buses and bus stops in the network. Bus stops are shown as black squares, autogenerated nodes are shown as gray, and buses are shown as blue diamonds on the display. The display supports a coordinate mapping of 20,000 meters in either direction, centered at 0. Each meter is one pixel when zoomed in, and the display can be zoomed out. When a node is receiving data, the node turns to green; if a collision occurs, it turns to red. However, this is difficult to note on the display since most transmits happen in a very short period of time.
- **Test.** This does a very simple test of the NetworkNode class. It isn't used in the simulation itself.
- **Init.** This button loads the network defined by the 'honolulu' dataset.
- **Advance 1s, Next Event, Run, Set Time.** These buttons control the time of the simulation. 'Next Event' advances the simulation to the next event, while 'Advance 1s' advances the

simulation until one second elapses. 'Run' is the equivalent of continuously pressing the 'Advance 1s' button. 'Set Time' sets the current time of the simulation, though currently isn't working properly due to the way the initial broadcast time for each bus is set.

- **Time text.** The current time of the simulation is shown here. The time is shown with three values - a human readable value in 24h format, a decimal between 0 and 1 representing the fraction of a second portion of the current time, and an integer representing the number of seconds from 12:00 am.
- **Stats.** A toggle button that needs to be toggled to enable collection of data. Data is not collected while this button is unpressed.
- **Spinner.** (set to 2 in Figure A.1) This represents the zoom of the display area. At zoom level 1, each pixel represents one meter in the system; increasing the value zooms out so the user can view a larger portion of the system at once.
- **Stop @ time.** When pressed, it sets a flag in the simulator that if the specified time is reached and the Run button is pressed, stop the simulation and set the Run button back to normal. Note that if the time in the text box is changed after this button is pressed, the stop time is **not** updated.
- **Log Tables.** When pressed, the simulator outputs the bus information tables at 30 second intervals of simulation time.
- **Progress Bar.** This bar shows how far into the current second has elapsed in the simulation. This bar is only used while the Run button is pressed.

Figure A.2 shows the statistics tab of the program, which is used to display data regarding the network. Note that both utilization and collisions are not tracked unless the "Stats" button is pressed.

- **Utilization.** This displays the utilization of the network. Utilization is determined by tracking how many time units a node is idle vs. sending or receiving, and taking that percentage over all nodes in the system.
- **Collisions.** This displays the number of collisions, as well as successful transmits, in the system. A transmit is a collision if a node is receiving data and senses another broadcast nearby; a transmit is successful if a receive finishes without a collision.

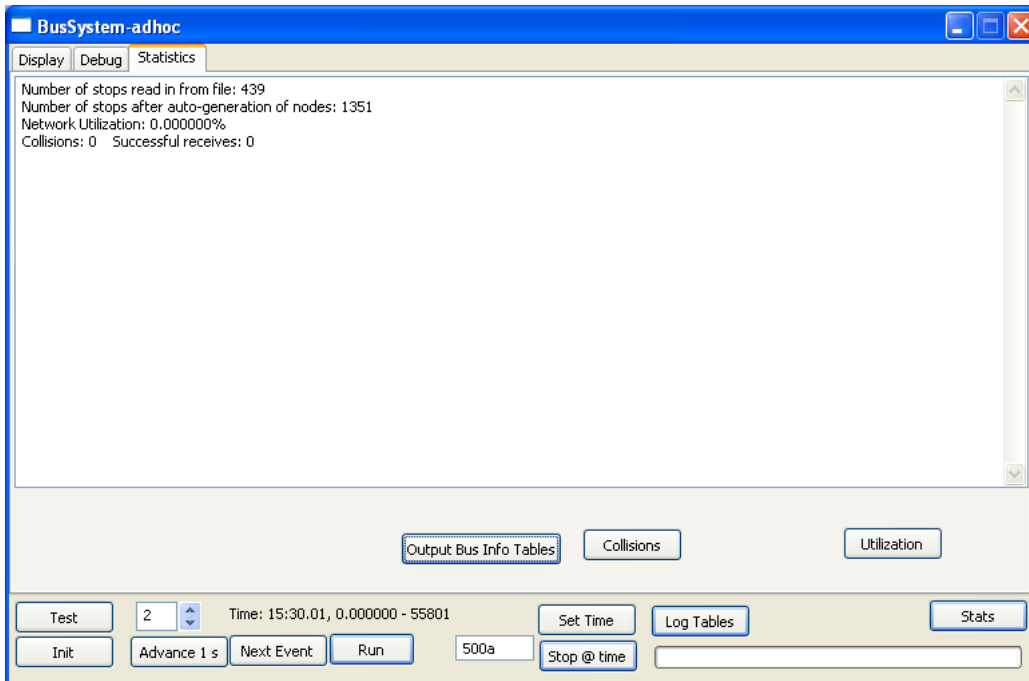


Figure A.2. Bus system simulator, Statistics tab.

- **Output Bus Info Tables.** This outputs the bus information tables as they currently stand.

Bibliography

- [1] Edoardo S. Biagioni and K.W. Bridges. The Application of Remote Sensor Technology To Assist the Recovery of Rare and Endangered Species. *International Journal of High Performance Computing Applications*, 16(3):315–324, 2002.
- [2] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.
- [3] Norman Abramson. The aloha system: another alternative for computer communications. In *AFIPS '70 (Fall): Proceedings of the November 17-19, 1970, fall joint computer conference*, pages 281–285, New York, NY, USA, 1970. ACM.
- [4] Ian F. Akyildiz, Xudong Wang, and Weilin Wang. Wireless mesh networks: a survey. *Computer Networks*, 47(4):445–487, March 2005.
- [5] TheBus, Retrieved May 2010. <http://www.thebus.org>.
- [6] TheBusHEA, Retrieved May 2010. <http://hea.thebus.org>.
- [7] KHON2.com. New Technology Helps Honolulu’s Bus Riders, Aired on August 7, 2009. <http://www.khon2.com/content/news/developingstories/story/New-Technology-Helps-Honolulus-Bus-Riders/z3VyPLsSt0qmTeqh-COAlg.csp>.
- [8] Hamed Soroush, Nilanjan Banerjee, Aruna Balasubramanian, Mark D. Corner, Brian Neil Levine, and Brian Lynn. DOME: A Diverse Outdoor Mobile Testbed. In *Proc. ACM Intl. Workshop on Hot Topics of Planet-Scale Mobility Measurements (HotPlanet)*, June 2009.
- [9] NextBUS, Retrieved October 2009. <http://www.nextbus.com/corporate/works/index.htm>.

- [10] Jin-Shyan Lee, Yu-Wei Su, and Chung-Chou Shen. A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi. In *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*, pages 46–51, Nov. 2007.
- [11] IEEE Computer Society. *WiMAX Specifications (802.16)*, 2004. <http://standards.ieee.org/getieee802/download/802.16-2004.pdf>.
- [12] Sunil Kumar, Vineet Raghavan, and Jing Deng. Media access control protocols for ad hoc wireless networks: A survey. *Ad Hoc Networks* 4, 4(3):326–358, 2006.
- [13] Wikipedia: Sao Paulo, Buses section, Retrieved December 2009. http://en.wikipedia.org/wiki/S%C3%A3o_Paulo\#Buses.
- [14] Google Transit, Retrieved May 2010. <http://www.google.com/transit>.
- [15] Daniel Hess, Jeffrey Brown, and Donald Shoup. Waiting for the bus. *Journal of Public Transportation*, 7(4):67–84, 2004.
- [16] wx-Dev-C++, a development environment based on wxWidgets, Retrieved May 2010. <http://wxdsgn.sourceforge.net/>.
- [17] Andrew Wong. Case Study: Simulated deployment of a mesh network in Honolulu. In *Proceedings of the 43rd Annual Hawaii International Conference on System Sciences (CD-ROM)*. Computer Society Press, January 2010.
- [18] The Network Simulator ns-2, Retrieved May 2010. <http://www.isi.edu/nsnam/ns/>.
- [19] wxWidgets, Cross-Platform GUI Library, Retrieved May 2010. <http://www.wxwidgets.org>.
- [20] S. Xu and T. Saadawi. Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks? *Communications Magazine, IEEE*, 39(6):130–137, Jun 2001.
- [21] Frederic Dreher. Simulating media access control protocols in wireless networks. Master’s thesis, Swiss Federal Institute of Technology Zurich, 2007.
- [22] ZigBee Alliance. *ZigBee Specifications*, April 2005. version 1.0.
- [23] IEEE Computer Society. *WiFi Specifications (802.11)*, 2007. <http://standards.ieee.org/getieee802/802.11.html>.

[24] P. Kinney. ZigBee Technology: Wireless Control that Simply Works. White Paper, October 2003.