UNIVERSITY OF CALIFORNIA
SANTA CRUZ

DELAY-TOLERANT ROUTING FOR EXTREME NETWORKING ENVIRONMENTS

A thesis submitted in partial satisfaction of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

Matthew Kenneth Bromage

September 2009

The thesis of Matthew Kenneth Bromage
is approved:

Professor Katia Obraczka, Chair

Professor J.J. Garcia-Luna-Aceves

Adjunct Assistant Professor Brad Smith

Lisa C. Sloan
Vice Provost and
Dean of Graduate Studies

Contents

List of Figures

List of Tables

Abstract

Delay-Tolerant Routing for Extreme Networking Environments

by

Matthew Kenneth Bromage

Delay-tolerant networking (DTN) provides a communications link in a heavily partitioned network that would otherwise be impossible to route across using existing technologies. Previously, routing solutions for mobile networks utilized an on-demand model such as AODV. These types of routing protocols do not provide a method for routing across partitions. The Epidemic routing protocol provides the best end-to-end throughput characteristics in many disrupted networks, providing the absolute smallest delay. The Epidemic routing protocol was developed to provide a solution to this problem, however Epidemic suffered from several performance deficiencies due to its indiscriminate methods of replicating messages in the network. The TAROT protocol, described in the second chapter of this thesis, was developed to minimize costly message overheads resulting from the Epidemic protocol, while at the same time providing comparable performance with regards to end-to-end delay.

The remaining two chapters of the thesis are dedicated to describing some of the contributions made to physical testbeds being developed at UC Santa Cruz to test the next generation of DTN, medium-access control (MAC), and MIMO radio protocols. When testing a DTN routing protocol, it is important to use real-world

scenarios and applications. The SCORPION and SlugTransit testbeds provide such an environment.

To my father, Bruce K. Bromage.

And my mother, Mary S. Bromage.

Acknowledgements

I would like to thank my family, for providing inspiration, insight, and support during these years. To my dad, Bruce, who has taught me the value of a positive attitude in life. To my mom, Mary, who has provided crucial planning and guidance. To my brother, Kevin, who has been a great inspiration to me as he works his way through his medical program. To my brother, Sean, I would like to thank for spending his summer with us in the iNRG lab working on the Scorpion project. His engineering talent and attention to detail have proved to be incredibly valuable to the success of the testbed. And for my wonderful girlfriend Zsofia, a tireless advocate of my work, and a supportive friend who has always been there for me.

I would like to thank my advisor, Katia, for her guidance and mentoring. I would also like to thank the countless faculty and staff at UCSC who have created a truly supportive environment for myself and others pursuing further education. It has been my pleasure to take part in such a wonderful community.

Finally I would like to thank my colleagues Vladi, Bruno, Matt, Dj, and James, for providing me with feedback on almost every aspect of my work, especially into the late hours of the night. I treasure the friendships they have provided me with over the years, and consider it one of the most valuable gifts from my time in this program.

Chapter 1

Introduction

Delay-Tolerant Networks (DTN) are designed to allow networking communications under the harshest conditions, including scenarios such as disaster relief, sparse networking infrastructure which is present in military operations and in developing countries, and interplanetary space communication. Delays may occur in wireless networks when transmission paths are blocked due to radio interference, physical obstructions, sheer distance between nodes, or mobile nodes that travel out of range. Conventional routing protocols such as AODV are unable to deal with large amounts of delay, and end up wasting precious bandwidth trying to correct the problem. DTN routing protocols are given the task of managing variable amounts of delay as part of the normal operation of a network. To do this in an efficient manner is the basis of this thesis.

Chapter 2 describes a novel DTN routing protocol named TAROT (Trajectory-Assisted ROuTing for Highly Partitioned Networks). TAROT is a delay-tolerant routing protocol. The novel aspect of TAROT is its ability to extract mobility patterns

exhibited by mobile nodes in real-time. These inferred patterns allow for a more efficient routing algorithm within a mobile network by selectively replicating messages in the network based on the likelihood of an encountered node reaching the destination, which is specified by a message. The result is an increase in efficiency compared with an Epidemic protocol that passes all messages in its queue to every node it encounters, a wasteful process indeed. The TAROT protocol defaults to an Epidemic message replication when it encounters nodes without established mobility patterns to take full advantage of all contact opportunities. This is important in highly partitioned networks where encounters are rare. Performance of the TAROT protocol was compared against that of a pure Epidemic protocol. In simulations conducted for this study, Tarot was able to match the message delivery ratio of Epidemic, however TAROT was able to reduced overhead by as much as 60%.

Chapter 3 describes SCORPION (Santa Cruz mObile Radio Platform for Indoor and Outdoor Networks), a mobile wireless networking testbed that comprises of over 100 nodes. The nodes consist of a small computer running Linux equipped with a set of wireless 802.11 radios. There are five different types of nodes (bus, briefcase, helicopter, airplane, and roomba) with various types of mobility schemes including pedestrian, vehicle, and aerial. Each node-type is able to communicate with every other node-type in the testbed, allowing for a truly heterogenous network of mobile nodes. It is important to test networking protocols using real-life scenarios, and this is especially true for disruption- and delay-tolerant network protocols that rely on the mobility of participating nodes. A custom-built management tool is used

to disseminate software updates, run experiments, and collect results.

SlugTransit, described in Chapter 4, is a real-time wireless tracking system designed for public transportation. The system is tested by outfitting the UC Santa Cruz's campus shuttles with computers running Linux and GPS tracking equipment. Wireless base-stations installed on the roofs of UCSC buildings provide an uplink for the shuttles to periodically report their location to a centralized server. This solution is less costly than one which relies on cellular networks to transfer location information. SlugTransit has been tested over the course of several months with a set of ten bus nodes, and has been very reliable. The node locations can be accessed using a web site from any web-connected computer or mobile smart phone.

Chapter 2

TAROT

2.1 Abstract

TAROT (Trajectory-Assisted ROuTing) is a DTN routing framework that detects and extracts structure in node movement in real-time. *TAROT* is motivated by the postulate that mobility, in particular human mobility such as is present in vehicular mobility, is seldom random and thus exhibits recognizable patterns. *TAROT*'s mobility pattern extraction capabilities transcends current solutions that rely on abbreviated (in some cases, instantaneous) snapshots of mobility history. *TAROT* is therefore able to predict future mobility with increased accuracy. Routing decisions are guided by node mobility patterns, ultimately resulting in more efficient routing and forwarding of messages. Our approach is capable of accommodating conditions where the best node may be one that is currently moving away from the destination. *TAROT* uses a "controlled epidemic" approach to route messages in which nodes will

only be "infected" with a message if their mobility pattern takes them closer to the destination.

TAROT's performance is evaluated through simulations using the QualNet network simulator. A side-by-side comparison against Epidemic Routing [50] under a variety of mobility and workload scenarios show that *TAROT* is able to match Epidemic's high data delivery guarantees at substantially reduced overhead (over 60% in some of our experiments). *TAROT*'s efficiency comes at the price of a slight increase in delivery delay (around 20% in our experiments). We argue that applications that use intermittently-connected networked environments are inherently tolerant of delay, and therefore favor slight increases in delay for increased efficiency and reduced resource consumption.

2.2 Introduction

Finding a path from a source to some destination, i.e., routing, is one of the core functions in a communications network. Many routing algorithms and protocols have been proposed targeting wired–, infrastructure-based wireless–, as well as multi-hop ad-hoc wireless networks (MANET). However, most "traditional" routing solutions assume that there exists an end-to-end route between communicating nodes, and therefore treat the absence of a route as a fault. When disconnections are detected in these networks, the underlying routing protocol recovers by trying to find an alternate path. However, these "traditional" techniques (including MANET routing

protocols such as AODV [39] and DSR [30]) are not designed to handle frequent, arbitrarily long-lived connectivity disruptions, and will subsequently suffer considerable performance degradation.

Networks exhibiting connectivity disruptions as part of their normal operation were first proposed in the context of long propagation delay environments such as the so-called "Interplanetary Networks" [25]. They were then referred to as "delay-tolerant networks", or DTNs. Later, these protocols were adapted for more terrestrial applications. In these scenarios, long delays were a result from connectivity disruptions which were caused by a variety of factors such as wireless channel impairments, node mobility, sparse deployments, duty-cycled node operation, or a combination thereof. This new type of network which came to be known as "disruption-tolerant networks," also DTNs, was designed to tolerate frequent, arbitrarily long disruptions in connectivity.

There are many interesting DTN applications which span from the creation of an "Interplanetary Internet" [25] to improving the quality of life for people living in the poorest parts of the world. For example, a DTN can leverage the transportation infrastructure (bicycles, scooters, busses) and effectively "bridge the digital divide" by connecting remote communities and rural villages to the Internet. This has been the subject of previous research efforts such as KioskNet [24] and UnitedVillages [49]. Communities in such areas may benefit from applications such as digital medical records synchronization, news and educational content dissemination, remote diagnosis services, multi-media communications, and local agricultural pricing indices.

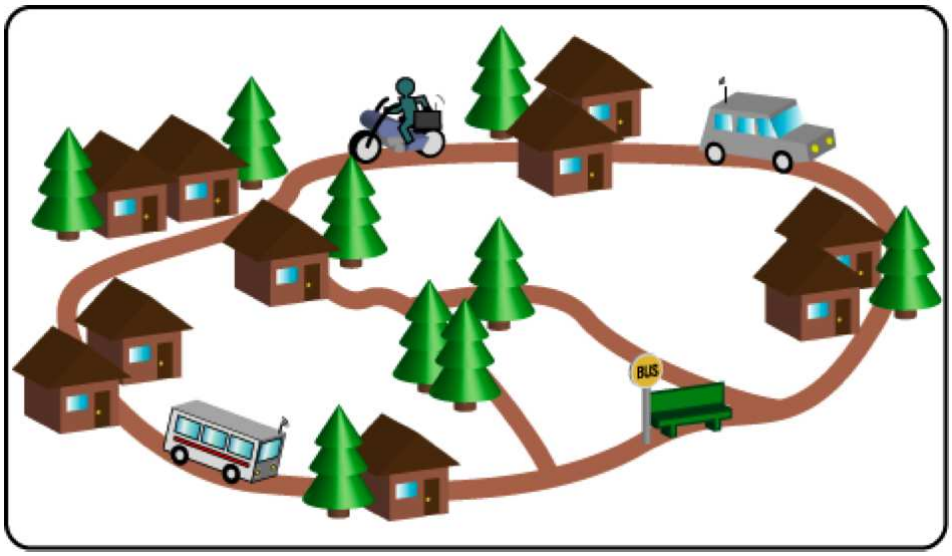

Figure 2.1: A sample village scenario with various types of mobility. The mobility of vehicles tend to exhibit patterns such as recurring paths. The path information can be leveraged to make more informed routing decisions in a hop-by-hop opportunistic manner.

There is also a growing demand for transcription services where paper documents are converted to digital copies which are then uploaded to the Internet for delivery. This application in particular could be a viable source of additional revenue for the villages.

There are also many other exciting multi-disciplinary application domains for DTNs. For example, environmental and habitat monitoring systems such as ZebraNet [31] and DeerNet [17] map the mobility (among other things) of animals using wireless sensor network (WSN) devices. These projects can have significant scientific and societal impacts as we learn more information about animal migrations, inter-species interactions, and how to track emerging diseases.

In order to accomplish a level of connectivity in the above applications, DTNs require a new routing paradigm which no longer treats frequent and arbitrarily long-

lived connectivity disruptions as faults, but rather as a normal state of operation. As a result, an extension to traditional "store-and-forward" routing has been proposed for DTNs. The "Store-*carry*-and-forward" [20] paradigm was created to effectively bridge partitions in the network through the use of mobility. Here, a node stores and carries data for destinations to which it currently has no route until an opportunity to forward the data presents itself. This type of "mobility-assisted routing" therefore uses "contact opportunities" during node encounters to perform "opportunistic routing", a popular approach to DTN routing. A notable example is the Epidemic Routing protocol [50], which, at every contact opportunity, passes a copy of the message to the encountered node (replication). Clearly, in non-congested networks, epidemic will find the shortest-path to the destination if/when it exists. However, this is accomplished at considerably high overhead, i.e., excessive number of duplicate messages circulating in the network.

One solution to reducing the overhead is to employ a "controlled-replication" technique. The "Spray-and-Wait" protocol [45] is an example of a solution which seeks to limit the number of copies floating around in the network by selectively choosing the number of nodes which will receive duplicates. This is done by calculating an estimate of the total number of nodes in the network using "encounter-timers". A message can therefore be replicated to a node's neighbors until the number of copies reaches a fraction of the network nodes. This work was later extended [46] to allow nodes carrying replicas to forward those packets to other nodes in the network in order to increase the delivery ratio.

TAROT uses a "controlled-replication" scheme which utilizes an advanced metric to determine which nodes are candidate to receive copies. The metric is calculated using a distributed Path Detection (PD) algorithm (described in Section 2.4) which serves to extract the structure exhibited by the mobility of the nodes in the network. The structure information allows for an increased accuracy in the predictions of future mobility due to the fact that paths indicate a sequential time-series of movements, as opposed to simply keeping a history of the previously visited locations. These predictions in turn are directly linked to the decision for a node to carry a message towards some destination. If a node has a high probability of reaching that destination, it should request a copy of the message in order to provide timely delivery. For example, in the simple village scenario depicted in Figure 2.1, the mobility patterns of the nodes can be inferred and used to propagate messages efficiently across the network.

TAROT does not require global or local neighborhood state information to be disseminated or maintained, therefore consuming less resources in terms of message overhead, processing and storage requirements, and ultimately energy requirements. Of course the more global state each node is aware of, the more efficiently messages can be routed across the network. However, a study into the various levels of state information nodes are allowed access to, shown in [28], indicates diminishing returns.

This chapter is organized as follows: the next section provides an overview of the *TAROT* protocol, outlining each of the functional blocks. Section 2.4 discusses the Path Detection (PD) algorithm, followed by a discussion of the Routing Decision

Engine (RDE) in Section 2.5. Section 2.6 will then showcase the results of applying PD in combination with RDE in a variety of scenarios. Finally we summarize with related work and conclusions.

2.3 Overview

TAROT (Trajectory-Assisted Routing) uses inferred node trajectories (also referred to here as paths) to make informed routing decisions within the network. To accomplish this, *TAROT* uses two core algorithms: the **Path Detection (PD)** algorithm and the **Routing Decision Engine (RDE)**. The PD algorithm is the mechanism which extracts structure from the mobility of the node. The structure is stored in a "path library" which contains a list of commonly followed paths. Each node maintains its own path library, and paths are not exchanged in the network. The RDE, is responsible for making routing decisions such as the forwarding and replication of messages. The RDE uses the path library to make the comparative distance calculations, described further in Section 2.5.

Figure 2.2 depicts the process where a delay-tolerant application would utilize *TAROT*. The application sends the current location of the node to the PD algorithm in regular intervals (step 1) so that the PD algorithm can build the library of paths (step 2). In parallel, the application monitors for encounters with other nodes. During an encounter, the application sends a list of destinations sent by the encountered node to the RDE (step 3). The RDE uses the path library (step 4) to calculate the closest

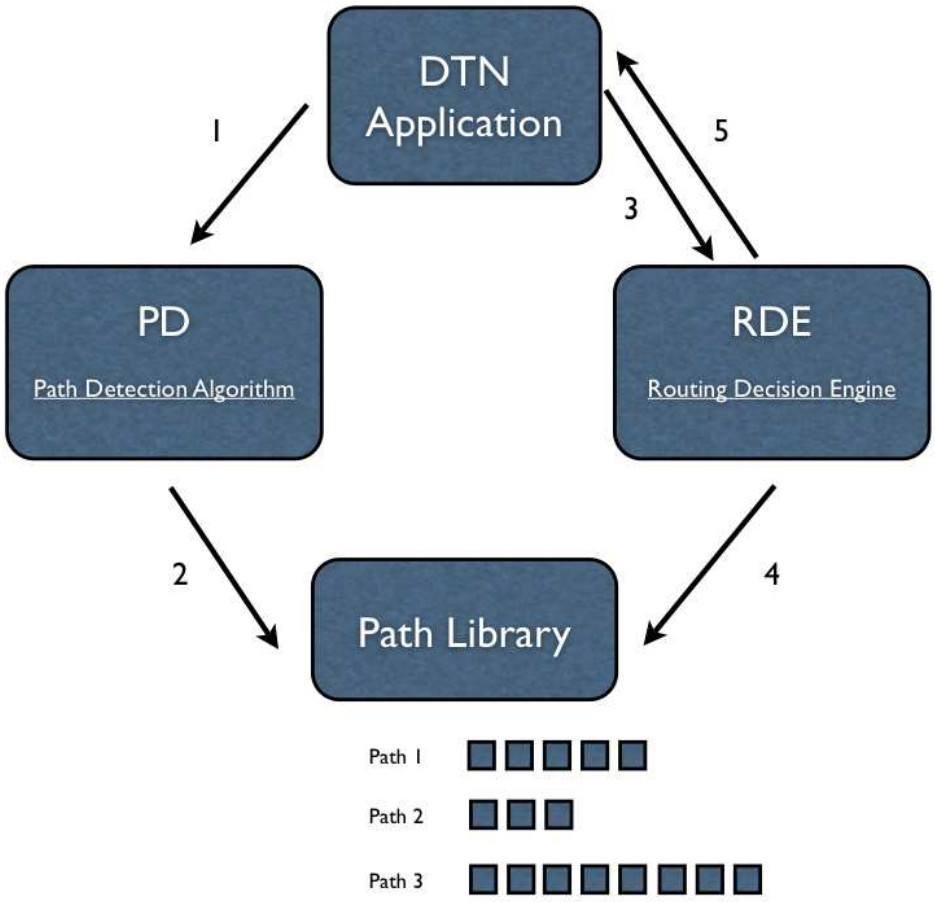

Figure 2.2: *TAROT* functional blocks. 1) The application sends the current location to the Path Detection (PD) algorithm. 2) The PD algorithm extracts paths and stores them in the path library. 3) When two nodes encounter, the application sends a list of message destinations to the RDE. 4) The RDE uses the path library to decide whether to forward or replicate. The squares under the path library indicate a time-series of coordinates for each path. 5) The decision is passed back to the application.

distance the node can get to each destination, and determines whether each packet should be requested for replication by the encountered node or simply ignored. The RDE then reports each decision back to the application (step 5).

2.4 Path Detection (PD)

At the core of the *TAROT* protocol is the Path Detection (PD) algorithm. Performing PD allows a node to predict its future mobility with increased confidence. We will use the words "path" and "trajectory" interchangeably.

Detecting a path involves making a determination about when the path should end. There are two ways to do this. The first method detects if a node has been following a "cycle", i.e., the node begins to re-trace a trajectory from the origin for more than three positions. The second way a path can be established is if the node is stationary for a period of three minutes or more. These parameters were selected based on field tests of vehicles in an urban setting which included stops at stop signs, traffic lights, bus stops, and other interruptions in mobility which should not account for a path to be ended. For example, a path that crosses over itself should not be treated as multiple paths; instead it should be identified as a path crossing, and the algorithm should continue building the path.[1] This will ensure that path files are not overly segmented, allowing for longer paths and better accuracy in predicting future mobility.

[1] We note here that the GPS devices used (Pharos iGPS-500) contain some inherent drift in accuracy, therefore a subset of the readings should be identified as one-time anomalies, or "outliers", and should be ignored.

The PD algorithm allows for nodes to be in one of two states: PATH-CREATION or PATH-FOLLOWING. The decision flow chart for each of these states is depicted in Figure 2.3. During PATH-CREATION, nodes used their sampled location to build a library of routinely followed paths.[2] Currently, we sample once every five seconds. The high sample rate avoids the need for interpolating between successive samples, which significantly simplifies the path creation process. In the PATH-FOLLOWING mode, nodes are able to track their progress using paths contained in the path library.

To create a new path, state variables and thresholds are required for each node. The **Current Working Path (CWP)** keeps track of which path in the path library is actively being edited during PATH-CREATION. The **Current Followed Path (CFP)** indicates which path the node is following when the node is in the PATH-FOLLOWING state. The **Follow Threshold (F-THRESH)** ensures a node is following a path, instead of crossing over a path. The **Outlier Threshold (O-THRESH)** is used during PATH-FOLLOWING to identify "outlier" locations. An outlier is defined as an "unexpected" node location that does not exist on the path the node is currently following. This could be caused by an anomalous GPS reading, in which case when the node returns to following the path. If the O-THRESH is reached, the node then switches to PATH-CREATION. The **Stop Threshold (S-THRESH)** identifies locations which are used to finalize a path; meaning no more new locations

[2]Currently there is no limit to the number of paths that can be stored in each node's "path cache". However, it would be relatively straightforward to age path table entries, e.g., by recency, to limit path storage requirements.

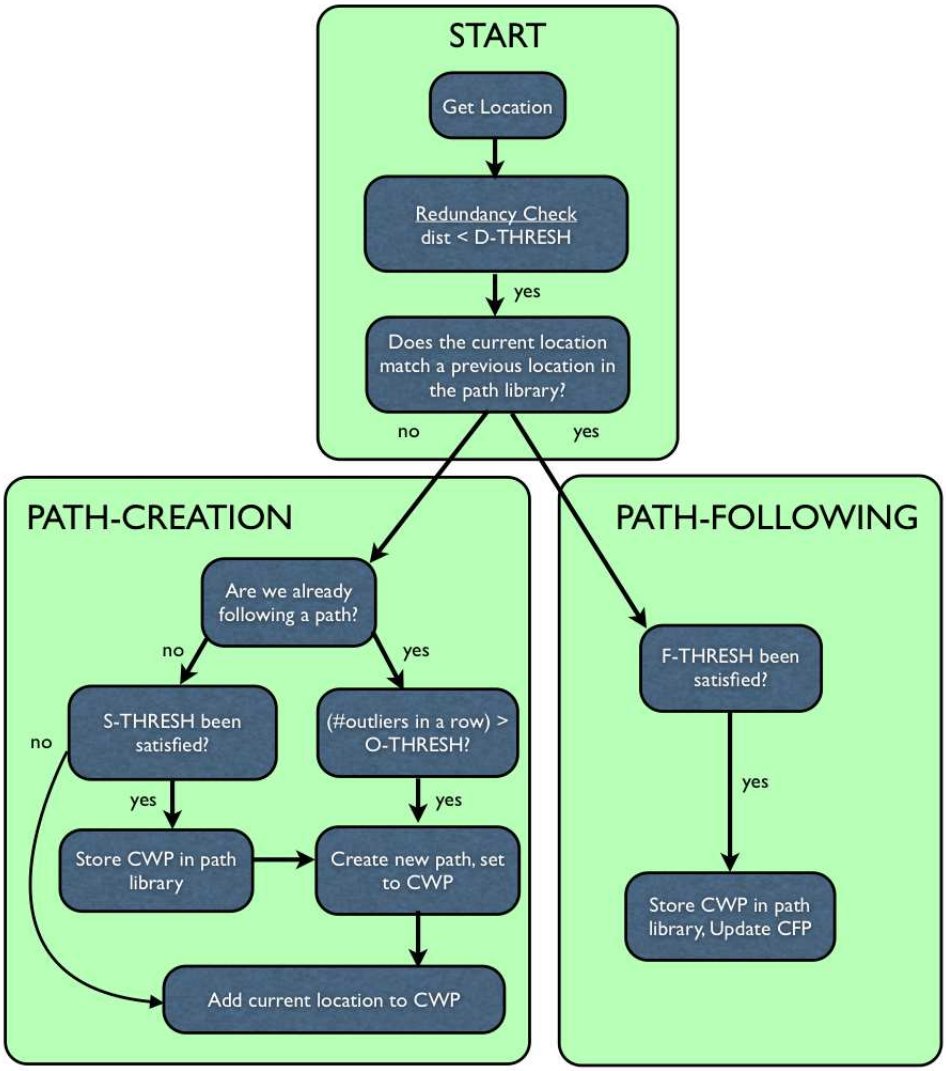

Figure 2.3: Flowchart for the Path Detection (PD) algorithm indicating the PATH-CREATION and PATH-FOLLOWING modes of operation. The process is repeated for each sampled coordinate.

will be added to the CWP. The S-THRESH is currently set to 3 minutes to account for events such as stop signs and traffic signals. Different values could be used depending on the types of mobility targeted. Finally, the **Distance Threshold (D-THRESH)** defines the maximum distance between two reported locations in order to be treated as the same location. The value is set at roughly 100 meters. This is to account for the drift associated with current GPS technology, as well as to address the fact that during PATH-FOLLOWING, location samples will not always line up precisely.

2.4.1 PATH-CREATION

In PATH-CREATION, a node actively creates a repository of routinely followed paths. There are three ways a node can enter the PATH-CREATION mode: (1) the reported location is new and the path library is empty, (2) during PATH-FOLLOWING, the number of outliers meets the O-THRESH, or (3) the CWP is finalized because the stop threshold (S-THRESH) has been reached and upon resuming, the node begins to build a new path. For example, this situation occurs when a node is parked in some location for longer than the S-THRESH. This would indicate that the location should be classified as a stopping point, ending the path file and starting a new one.

Each sampled location must pass a redundancy check in which the node verifies it is sampling the same location repeatedly, such as for a stop sign. A node will add the current location to the CWP if the point passes the redundancy check, the node is not currently following a path, and the S-THRESH has not been reached.

Figure 2.4: UCSC campus shuttle system routinely followed paths. From left to right: upper campus shuttle, core shuttle, loop shuttle.

If the S-THRESH has been reached, the node finalizes the CWP, adds the finalized path to the path library, and starts a new path file.

A path is finalized when either the S-THRESH has been reached (as described above), or if a node begins retracing its steps at any point on its CWP. Once the F-THRESH has been met, the node switches to PATH-FOLLOWING (Section 2.4.2).

Finalizing a path requires the node to calculate the total length of the path and the time to complete the path from start to finish. This information is stored as meta-data associated with the path and can be used by the routing decision engine as described in Section 2.5.

The decision process involved in PATH-CREATION is outlined in Figure 2.3. Figure 2.4 shows three example paths extracted from the mobility of campus shuttles using the PD algorithm. The shuttles followed different routes and were equipped with GPS devices to record mobility traces. The traces were then fed into the PD algorithm and a single path file was created for each bus to verify that their bus routes were correctly inferred.

2.4.2 PATH-FOLLOWING

Once a path has been entered into the path library, a node is capable of following that path using PATH-FOLLOWING. If a node's current location matches a previously seen location in a path file stored in the path library, and the F-THRESH has been satisfied for that path, the node is said to be following the corresponding path (the CFP). The node continues to follow that path until it encounters positions that do not correspond to the CFP (the positions do not need to be in order). If the positions do not match other paths in the path library, they are considered to be "outliers" and a new path is created.

PATH-FOLLOWING is crucial to the *TAROT* protocol. The path files provide a stronger confidence value that a node will visit some location within some amount of time in the future. In fact, if a node is following a path, it can make estimates on the distance and time required to visit any subsequent locations on that path.

There are several improvements that can be made to the PD algorithm. For example, dynamic thresholds and sample times can be used to account for various inferred types of mobility being exhibited (human, vehicle, aerial). Path aging can also be used for improved accuracy.

2.5 Routing Decision Engine (RDE)

The Routing Decision Engine (RDE) uses information on mobility patterns exhibited by nodes to make decisions on how to forward messages during node encounters, as shown in Figure 2.2.

Currently, the RDE uses a "controlled" epidemic approach to take advantage of epidemic routing's inherent redundancy. By controlling the number of relay nodes which receive a copy of the message, *TAROT* is able to reduce the network-wide overhead. Messages are limited to nodes exhibiting a high likelihood of either carrying or forwarding the message to the destination. Nodes with highly structured mobility patterns can make more informed decisions which are based on their future mobility predictions, and thus the likelihood they will encounter the destination. A node with no discernible structure in its mobility has a difficult time making any informed decision, and has previously been referred to as a "taxi" node [12].

TAROT is able to handle heterogeneous mobility scenarios, i.e., different degrees of mobility structure within the same network. For instance, it can accommodate highly structured mobile networks such as the one shown in Figure 2.5, networks with no mobility structure whatsoever (e.g., a network of "taxi" nodes), and varying degrees of structure present in the mobility of the nodes.. When the RDE encounters a node which exhibits no structure in its mobility, the algorithm will default to a purely epidemic approach to transferring messages. This is to ensure that opportunities to replicate are not missed, as they could be vital to forming a path to the destination.

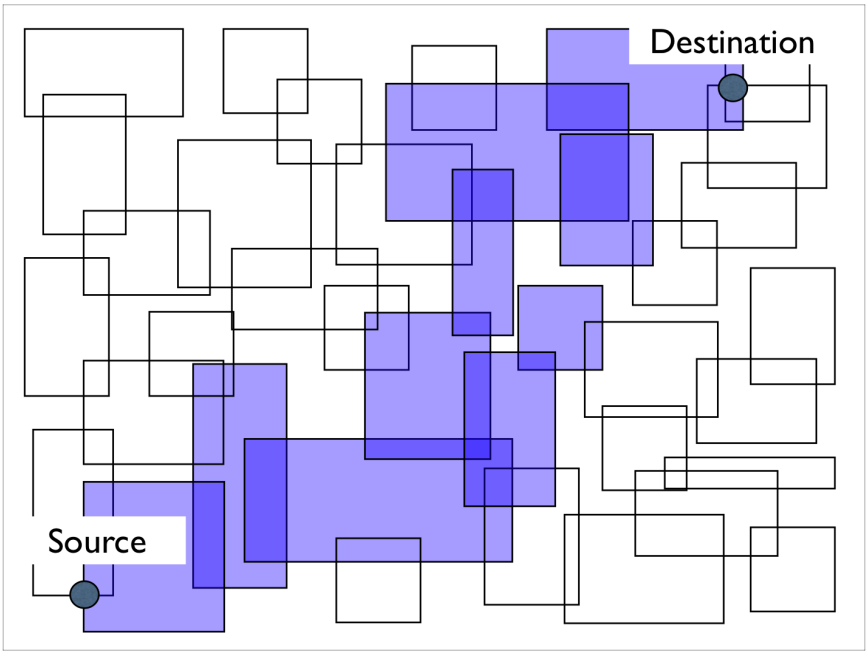

Figure 2.5: A sample route used by *TAROT*'s controlled epidemic forwarding approach. The light-shaded rectangles indicate structured mobility patterns detected by *TAROT*'s PDA. The dark-shaded rectangles indicate nodes selected to carry message replicas from *Source* to *Destination*.

Only nodes which have some degree of knowledge that they are not heading in the direction of the destination will not receive copies of the message intended for that destination. We plan to optimize this further in future work.

2.5.1 Controlled Epidemic Forwarding

As illustrated in Figure 2.6, when two nodes encounter one another, they perform an initial "handshake" to avoid unnecessary data exchange between nodes. The *SendingNode* sends a summary vector of locally stored messages in a *HelloPacket*,

which is received by the *ReceivingNode*. The *ReceivingNode* analyzes the *HelloPacket* and calculates a distance to each destination listed using its path library. If the *ReceivingNode* predicts that its future mobility will bring it closer towards the destination when compared to the *SendingNode*, it will request that message using an *Interest Message*, upon which the *SendingNode* will then replicate the message requested.

At this point, a few considerations are in order. We are currently considering static destinations whose locations are known. A location directory service could be used to provide destination location information. However in previous work, in particular geographic routing approaches such as GPSR [32], the location of the static destinations is assumed to be global knowledge for simplicity. We should also note that the replication scheme described above uses a "greedy" approach as, at every encounter, it tries to get "closer" to the destination. It is inevitable that a local minima may arise. Because our protocol replicates messages instead of only using forwarding techniques, local minima can be avoided by provided redundant copies of messages in the network.

Below, the messages exchanged during a node encounter are described in more detail:

Hello Messages These messages are transmitted once every second by every node in the network. They are used to locate neighbor nodes as well as to provide a list of messages IDs (message summary vector). The summary vectors used in *TAROT*

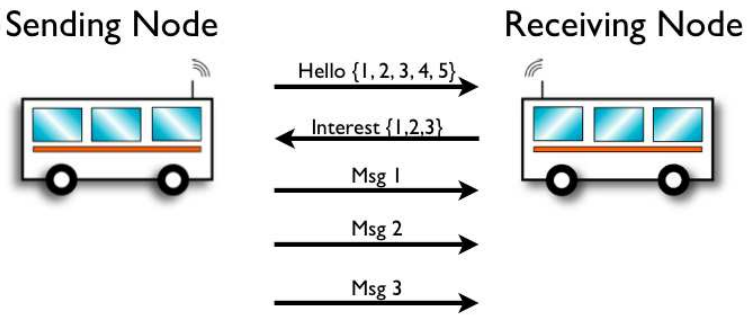

Figure 2.6: Handshake during an encounter. The *SendingNode* sends a *Hello Message* containing a summary vector of messages it carries. The *ReceivingNode* then sends an *Interest Message* for the messages it should relay. The *SendingNode* then forwards a copy of those messages.

are similar to the ones used in Epidemic Routing [50], but also include, for every message in the vector, (1) the destination ID and (2) the *SendingNode*'s distance to that destination.

Interest Messages These messages are sent by a *ReceivingNode* and received by the *SendingNode* during an encounter, and specify which messages the *SendingNode* should relay. If either the *ReceivingNode* or the *SendingNode* is not following a path, a forecast of future mobility cannot be made with high enough confidence, and therefore *TAROT* will default to using pure epidemic routing until a path has been established for both nodes. If, however, both nodes are following a path and the *ReceivingNode* can get closer than the *SendingNode* to a destination, the *ReceivingNode* requests all messages for that destination which it does not yet already have using an *InterestMessage*.

Data Messages When a *SendingNode* receives an *Interest Message* from a *ReceivingNode*, the *SendingNode* will replicate and transfer all messages listed in the *Interest Message*. *Data Messages* transfer the message for a single message ID.

2.5.2 Discussion

An intelligent combination of replication and forwarding of messages can be leveraged to reduce the overhead even further by forwarding the original message when a replication is not necessary (i.e. the confidence that the node will reach the destination is extremely high, or the message priority is low). Also, improvements can be made such as routing to mobile destinations, and providing a mechanism to disseminate location updates of destinations of interest in the network. We also plan to take into account a node's entire path library, in addition to looking at the current path, when making routing decisions.

2.6 Results

TAROT is tested using a variety of experiments including single flow, multiple flow, variable structure, and real mobility traces taken from the UCSC campus shuttles. To compare, a version of Epidemic [50] has been implemented. We use the following metrics as the basis for comparison: message delivery ratio, average delay (excluding messages still en route), and message overhead (calculated as the number of replicas created during the simulation).

2.6.1 Simulation Setup

The Qualnet simulator is used to verify the effectiveness of *TAROT*. The common simulation setup is as follows. Each simulation uses 60 nodes: 10 static destination nodes and 50 mobile nodes. The locations of the destination nodes are evenly spaced across the square boundary area sized $9.1908km^2$. There are two types of mobility modes a node may use: structured and unstructured. Structured nodes follow rectangle shapes of which the speed, location, and area are randomly generated for each node. The minimum rectangle size is $0.0891km^2$ and the maximum rectangle size is $0.9879km^2$, which approximates to 1% and 10% of the total boundary area respectively. Unstructured nodes use the Random Waypoint mobility model. Both structured and unstructured nodes travel at speeds ranging from $5 - 20km/hr$.

We use the IEEE 802.11 default radio stack in Qualnet, with the receive power decreased to $-85dBm$ and the transmit power reduced to $1mW$ resulting in a 100 meter range. These values were set using measured node encounter distances taken from moving vehicles equipped with 802.11 radios. Decreasing the transmit power and receive sensitivity also serves to decrease the connectivity of the network, so that nodes only communicate when they directly pass each other.

Each simulation in the experiment is repeated five times for averaging purposes. The node mobility files and source-destination pairs are randomly generated for each simulation. The simulations use Constant Bit-Rate (CBR) flows for their traffic. The number of flows and packets-per-flow varies with each experiment, and will be

Table 2.1: Single Flow Averaged Results

Experiment	Protocol	Delivery Ratio	Delay (sec)	Overhead (msgs)
Cold Start	Epidemic	60%	716	2733
	TAROT	54%	528	1608
Cold (ext)	Epidemic	100%	1116	5870
	TAROT	100%	1222	3913
Warm (ext)	Epidemic	100%	1066	5573
	TAROT	100%	1432	2111

discussed in more detail within each experiment below. Data packets are 512-bytes and sent in regular intervals of 20 seconds (within each flow). The notion of a *cold start* refers to flows which have started sourcing data before nodes are able to build their path libraries. A *warm start* refers to flows which start after the path libraries have been built and nodes are able to use the structure extracted. Finally, as we described previously, the location of the destination is assumed to be global knowledge.

2.6.2 Single Flow Experiments

The single flow experiments are designed to showcase how a single user might utilize a purely structured delay-tolerant network to send data (such as email) to a destination. This scenario uses a single CBR flow of 100 packets spaced 20 seconds apart. All 50 nodes in the network use structured mobility. The start and stop time of the flows is varied to get an idea of how well *TAROT* utilizes the extracted structure from the mobility of the nodes.

Table 2.1 shows the results of three different scenarios: a *cold start*, *cold start*

extended, and *warm start extended*. Here, the term "extended" refers to running the simulation for an extended amount of time after the flows have stopped sourcing data packets in order to allow for 100% delivery ratio. The *cold start* starts sourcing data at 0 seconds and ends the simulation at 2000 seconds. *Cold start extended* ends the simulation at 5000 seconds, and *warm start extended* starts sourcing data at 1000 seconds and ends the simulation at 5000 seconds.

During these simulations, *TAROT* was able to match the delivery ratio of Epidemic. The *cold start* experiment ended before the rest of the packets could reach the destination, which accounts for the lower delivery ratio. During *cold start*, *TAROT* was able to provide a 41% decrease in overhead while incurring a 26% *decrease* in delay. The decrease in delay could have been caused by excessive packet collisions during Epidemic which *TAROT* is able to avoid by using selective-replication, though this has not been thoroughly tested. An alternate explanation could be that due to the limited time for passing nodes to transfer data, the Epidemic protocol would be unable to transfer it's entire queue, whereas *TAROT*'s limited queue sizes allow for faster transfers during encounters. This effect seems to appear in Section 2.6.5. For the *cold start extended* scenario, there was a 33% decrease in overhead with a 10% increase in delay. The *warm start extended* scenario also showed good results with a 62% decrease in overhead and 34% increase in delay.

Table 2.2: Short-Lived Flow Results

	Delivery Ratio	Delay (sec)	Overhead (msgs)
Epidemic	100%	1203	19832
TAROT	100%	1488	12607
Difference	0%	+23%	-36%

2.6.3 Multiple Flow Experiments

These experiments use multiple source-destination pairs as well as vary the number of concurrent flows. The first experiment, short-lived flows, attempts to recreate a typical email exchange application where the traffic flows are sporadic in nature. The second experiment uses larger flows and varies the number of simultaneous flows present in the network. For each of these experiments, the nodes use fully-structured mobility and the flows receive a *warm start* (described above).

Short-Lived Flows

In this experiment, there are 10 flows each starting at a randomly chosen time within the range 1000-3000 seconds. Each flow then sources a random number of packets between 11-50 spaced 20 seconds apart. After all packets have been sourced, the simulations continue until 7000 seconds is reached.

These results are consistent with what we have observed in previous results. The overhead reduction is slightly lower compared with the cases where traffic offers a heavier load, which is to be expected.

Table 2.3: Long-Lived Flow Results (Comparison)

Flows/Pkts	Δ Delivery Ratio	Δ Delay (sec)	Δ Overhead (msgs)
5/100	-0.43%	+24.22%	-45.08%
10/50	0%	+16.95%	-41.45%
20/25	-1.05%	+17.68%	-44.57%
25/20	-1.30%	+16.32%	-39.73%
50/10	0%	+20.85%	-40.46%

Long-Lived Flows

In this experiment we steadily increase the number of simultaneous flows from 5-50. Effectively, this increases the number of nodes actively engaged as source-destination pairs. Each flow indicates a unique source and destination. We keep the total number of packets sourced in the simulation constant as the number of flows varies. This is so that the results are not affected by an increase in the total number of packets. Table 2.3 displays the difference in overhead, delay, and delivery ratio between Epidemic and *TAROT*.

As the number of concurrent flows increases, more packets are being injected into the network within a shorter span of time. This has the effect of scaling the effective "load" on the network (without saturating it). The average delay incurred is around 20% higher compared to Epidemic and the average overhead reduction was around 40%. The delivery ratios for both protocols were averaging near 100%. The average delay and overhead metrics were relatively consistent across each of the simulations.

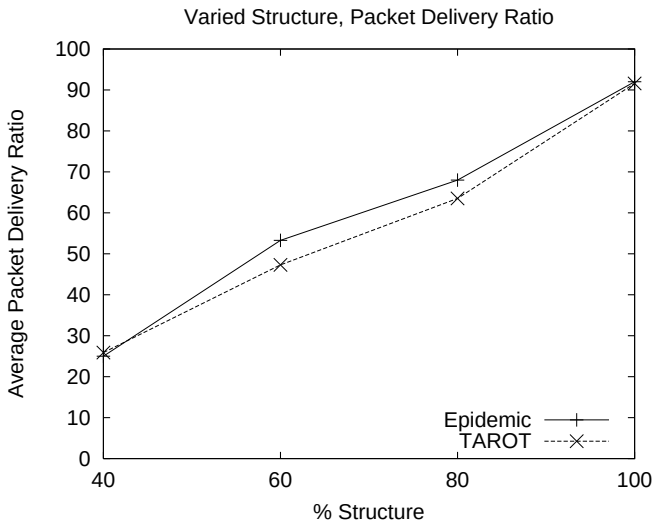

Figure 2.7: *TAROT* is able to maintain an almost identical delivery ratio compared with Epidemic as the amount of structure present in network varies.

2.6.4 Varied Structure Experiment

TAROT was developed to take advantage of networks exhibiting structure in mobility. We want to ensure, however, that *TAROT* performs well in both structured and unstructured mobility scenarios. This experiment was designed to control the amount of structure present in the network. Therefore, the nodes in the network are a mix of unstructured and structured nodes (as described in Section 2.6.1). All flows in these experiments use a *warm start*.

The traffic is as follows: there are 5 simultaneous flows with 100 packets per flow. The flows start at 1000 seconds and end at 3000 seconds. We run the simulation until 7000 seconds to allow packets to propagate through the network.

As can be seen from Figure 2.7, *TAROT* is able to maintain a delivery ratio

comparable to Epidemic down to just 40% structure exhibited by the network. Below 40%, the number of contact opportunities becomes increasingly rare due to the random mobility of the unstructured nodes, and the delivery ratio was too low to warrant any analysis for both Epidemic and *TAROT*. This can also be attributed to the fact that unstructured nodes do not cover the same area as structured nodes, and therefore limit the number of contact opportunities available during the simulation.

Figures 2.8 and 2.9 show graphs for the average delay and overhead. The general trend is a decreasing delay for both Epidemic and *TAROT*. Again, this is due to the increased contact opportunities given by increased structure exhibited by the network. *TAROT* is able to deliver messages end-to-end with an average delay spanning the range between 5% and 24% and an average overhead savings between 45% and 52%. The effectiveness of *TAROT* in reducing the overhead becomes more apparent as the structure in the network is increased. This is because nodes using unstructured mobility are unable to create an established path to aid routing decisions, and must default to Epidemic flooding.

2.6.5 SCORPION Bus Traces

Using the SCORPION [8] testbed at UCSC, traces have been collected detailing the interactions between busses on campus over the span of a day, in which 10 campus shuttles were outfitted with GPS tracking devices. These traces were then imported into Qualnet driving simulations running *TAROT* and Epidemic Routing. The simulations used 5 flows with 100 packets per flow. Messages were sent to des-

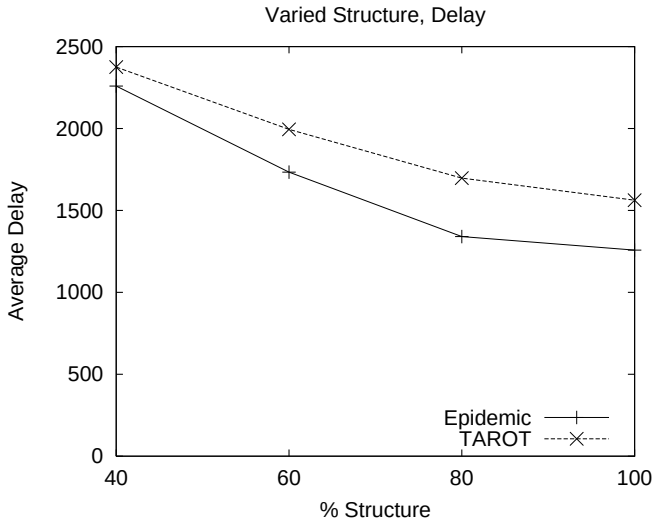

Figure 2.8: The average delay for successfully received data packets at the destination.

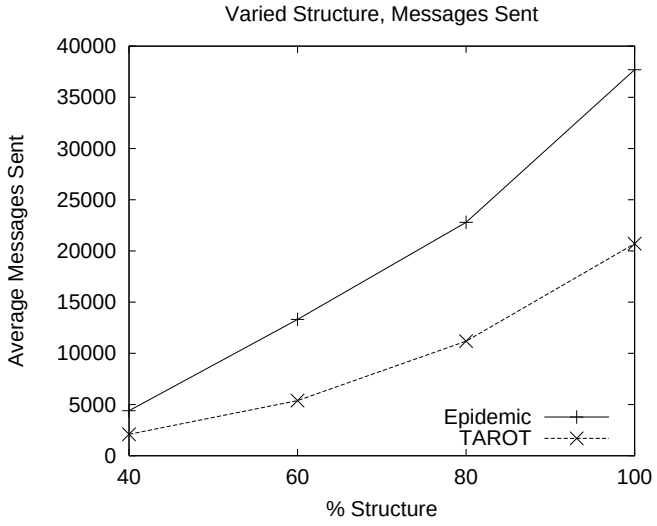

Figure 2.9: Average Overhead for Varied Structure Experiment

tinations positioned at five campus bus stops, the locations of which were randomly chosen around campus.

We ran the simulations until 100% delivery ratio. Overall, there was an 11.67% improvement in overhead compared with Epidemic, while the *TAROT* delay in fact *decreased* by 6.02%. We attribute the decrease in delay to the fact that there were very few opportunities to replicate. Few opportunities coupled with fast-moving busses results in very little time to pass information. Therefore, *TAROT* was able to deliver more data because it was not burdened by transferring all of its messages such as in Epidemic. The overhead did improve slightly, however this was hindered by the fact that there is only one road on campus for the shuttles to move. Therefore every shuttle is able to reach every destination, and the greedy algorithm can only do slightly better than Epidemic.

2.7 Related Work

DTN routing has been the subject of extensive research in the past few years. In particular, there have been a number of efforts proposing different strategies for controlling the "spread" of epidemic routing in order to reduce the amount of message replicas circulating in the network. One such strategy uses historical node contact information (e.g., "age of last encounter", encounter frequency) to determine the *utility* of a node as a relay to a destination. Some notable examples include [6,7,13,35,38,43].

"Spray-and-Wait" [45] also proposes a variation of epidemic routing where,

"in the first hop", the message originator "sprays" a certain number of copies of a given message to encountered nodes depending on their "age of last encounter" with the message's destination. The number of replicas "sprayed" is calculated based on the number of nodes in the network. The "infected" nodes must then deliver the message directly to the destination. "Spray and Focus" [46] has been proposed as an extension which allows "infected" nodes in the first hop to forward their replicas to other nodes who have seen the destination relatively recently.

The MV routing algorithm [11] keeps track of historical information such as node encounters and visited locations. This information is used to decide which messages to replicate during encounters with other nodes. Later, the MORA [12] protocol was designed to use autonomous agents whose mobility is utilized to create contact opportunities. The agents can be programmed to optimize the network for certain metrics such as bandwidth and delay.

Similar to geographic routing approaches such as LAR [33] and GPSR [32], *TAROT* assumes that nodes have access to destination location information as well as their own location (e.g., GPS devices, or by means of some other localization technique, for example [44]).

In many real world applications, location history often serves as a good indication of a node's future position. For example, *TAROT* specifically looks for "paths" or "corridors" that are routinely followed by nodes. More generally, nodes can be categorized into routinely visited "spaces" such as those described in [36]. In the previously discussed GeOpps protocol [37], paths extracted from the navigation systems

of automobiles were used to selectively route messages using a greedy forwarding protocol. This is similar to our approach except that GeOpps assumes the knowledge of user-entered navigation information and therefore has near-perfect knowledge of future node mobility. We extend this work by extracting patterns in real-time, allowing for a broader set of applications.

2.8 Conclusion

This chapter introduced the *TAROT* (Trajectory-Assisted ROuTing) DTN routing framework, which bases its routing and forwarding decisions on movement patterns extracted from node mobility in real-time. Using past and current mobility patterns, *TAROT* is able to predict future mobility with increased accuracy resulting in more efficient routing and forwarding of messages. *TAROT*'s main contribution is that it goes beyond current solutions that rely on abbreviated (in some cases, instantaneous) snapshots of mobility history. *TAROT* uses a "controlled epidemic" approach to route messages where nodes will only be "infected" with a message if their mobility pattern takes them "closer" to the message destination

We evaluated *TAROT*'s performance using the QualNet network simulator. A side-by-side comparison against Epidemic Routing [50] under a variety of mobility and workload scenarios show that *TAROT* is able to match Epidemic's high data delivery guarantees at substantially reduced overhead (over 60% in some of our experiments). *TAROT*'s efficiency comes at the price of a slight increase in delivery delay

(averaging around 20% in our experiments). We argue that applications that use intermittently-connected networked environments are inherently delay-tolerant, and therefore favor slight increases in delay for increased efficiency and reduced resource consumption.

2.9 Acknowledgments

This work has been partially supported by the Army Research Office (ARO) under a MURI project named DAWN, NSF grants ANI 0322441 and CNS 0534129.

Chapter 3

SCORPION

3.1 Abstract

The SCORPION (Santa Cruz mObile Research Platform for Indoor and Outdoor Networks) project is a testbed for conducting research and testing protocols for delay-tolerant networks. Creating such a testbed is a difficult endeavor. This is due to the fact that in order to create real-world ground-based delay, you require node mobility in the network. Generating repetitive mobility is difficult because the it needs to be reproducible for subsequent experiments and averaging purposes, as well as be done with minimal human intervention for scaling purposes.

For this reason, the SCORPION testbed uses an autonomous airplane to regulate mobility and fly patterns set by the operator. The testbed also is able to make use of helicopters to provide limited local mobility. Briefcase nodes are used to represent human mobility. iRobot Create nodes are used to simulate a fleet of robots

capable of responding to emergency situations, such as fire or earthquake scenarios. Finally, bus nodes are used to represent vehicular mobility. Every node-type can communicate with every other node-type to provide a truly heterogeneous network of mobile nodes.

Management of the testbed is another topic entirely, and we have made large strides towards developing a platform which efficiently does this task. This chapter discusses the design, implementation, and experiences with the SCORPION testbed.

3.2 Introduction

During the last decade, the success and popularity of wireless standards such as IEEE 802.11 have drawn the attention of the research community to wireless networks. A great amount of effort has been invested into research in this area, most of which relies heavily on simulation and analysis techniques. However, simulations do not precisely control hardware interrupts, packet timing and real physical and MAC layer behaviors. As a result, simulation results need to be validated by real implementations, which is evident by the change in focus of research activities increasingly moving towards real implementations, including the deployment of testbeds as a main tool to analyze network protocol functionality. Under this context, we present an overview of *SCORPION* (Santa Cruz mObile Radio Platform for Indoor and Outdoor Networks), a heterogeneous wireless networking testbed that includes a variety of nodes ranging from ground vehicles to autonomous aerial vehicles. The purpose of SCORPION to

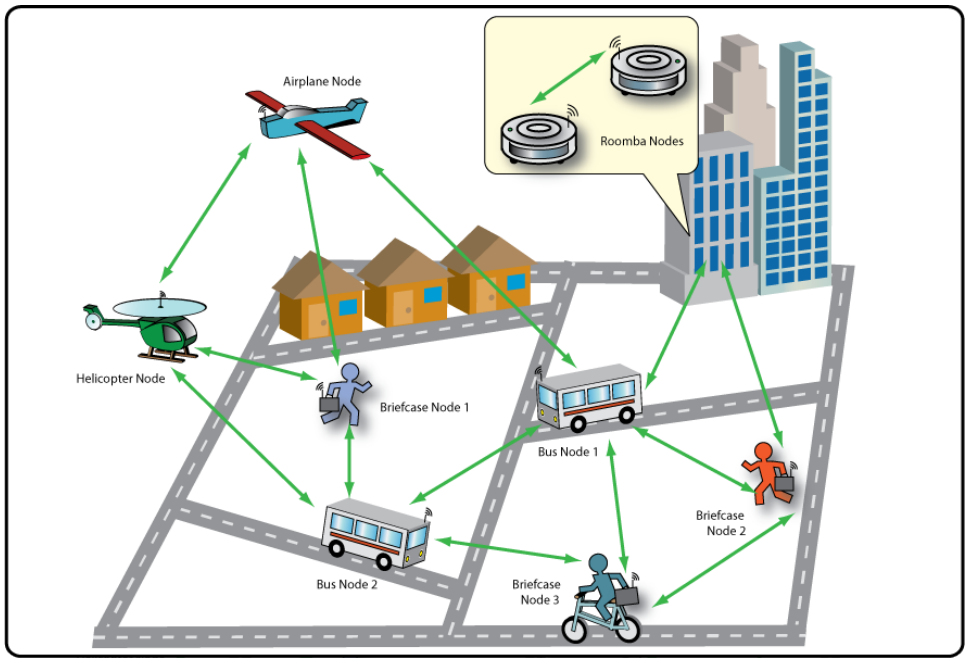

Figure 3.1: Its diverse set of nodes makes SCORPION suitable for experiments and evaluations under the most vast set of mobility applications.

is to deploy and investigate nascent networking protocols using a variety of mobile platforms utilizing structured as well as unstructured mobility patterns.

3.2.1 Goal

Node diversity in terms of mobility and capabilities (e.g., processing, storage, and communication) makes the SCORPION testbed well-suited for testing and evaluating a variety of wireless network protocols, including multi-radio, multi-channel medium access control, multi-hop wireless ad-hoc routing, as well as disruption-tolerant routing and message delivery protocols for networks with varying connectivity.

In its current implementation, SCORPION includes nodes outfitted with three different types of *autonomous* vehicles, namely: (1) iRobot Create ground robots

[27], (2) remote controlled airplanes equipped with Paparazzi autopilots [48], and (3) self-stabalizing remote-controlled helicopters [19].

Additionally, two non-autonomous mobile nodes complement the testbed. The first is a briefcase node that will be carried by people in order to mimic human mobility patterns. The second is a bus node which will provide structured mobility patterns for the testbed. The bus nodes are deployed in the UC Santa Cruz campus bus shuttles.

The testbed currently has four airplane nodes, four helicopter nodes, 20 iRobot Create nodes, 40 briefcase nodes, and 40 bus nodes. The iRobot Create nodes are equipped to carry briefcase nodes, allowing for a total of 88 nodes to be active at any one time during testbed operation. It should be stressed that each node has the capability to communicate with every other node in the testbed (bus to bus, bus to airplane, helicopter to briefcase, etc.).

The varied mobility patterns exhibited by these nodes allow for unique and innovative ways to test network protocols for current as well as next-generation network applications. SCORPION's management suite facilitates updating system and protocol software running on the nodes as well as monitoring the current status of nodes. Conditions that the management tool reports include whether the node is up or down, as well as the versions of the operating system, software and protocols are running. The management tool also includes scheduling capabilities for running multiple protocols during each deployment.

3.2.2 Related Work

The recent proliferation of efforts aimed at building wireless network testbeds is clear evidence that testing and evaluating wireless network protocols under realistic conditions is critical. One notable example is Rutgers University's ORBIT project which focuses on creating repeatable experiments in a wireless testbed setting [22].

Harvard's MoteLab [51] is a sensor network testbed that utilizes Berkeley Motes [52] equipped with light, temperature, and humidity sensors. Developers are able to upload software, test protocols, and receive results using their desktop. MoteLab specifically targets static, connected, indoor sensor network scenarios.

While testbeds like MoteLab [29] embody static topology paradigms, others utilize mobile nodes to create a more dynamic network environment. The University of Utah's Emulab is one example. Emulab's remote-controlled robots create dynamic topologies which are ideal for testing wireless sensor network applications. Each mobile robot has sensory capabilities and can communicate through inanimate motes. While this approach to wireless networking takes testing to the next level, it still lacks heterogeneity.

SCORPION focuses on emulating a diverse network scenario consisting of nodes with different capabilities and mobility patterns. Airplanes nodes, helicopter nodes, human nodes, and bus nodes each carry SCORPION hardware, making it an effective tool for protocol deployment in a dynamic, heterogeneous network environment that approximates real-world deployments more closely.

In [15] authors describe a miniaturized 802.11b-based, mulli-hop wireless network testbed called MINT. This testbed occupies a significantly small space, and reduces the effort required for setting up a multi-hop wireless network used for wireless application/protocol testing and evaluation. An interesting feature described in [15] is the hybrid simulation functionality. Under this platform ns-2 simulations can be executed where the link, MAC and physical layer in the simulator are replaced by real hardware. They also compare experimental results on their testbed with similar experiments conducted on pure simulation plataforms.

3.3 Nodes

In its current version, SCORPION consists of 88 nodes: 40 bus nodes, 20 briefcase nodes, 20 iRobot nodes, and 8 aerial vehicle nodes.

3.3.1 Bus Nodes

Bus nodes are equipped with a mini-ITX [47] computer running Linux. Each bus node has three 802.11a/b/g radios [4], a GPS tracking device [41], and a 900MHz radio [1]. The 900MHz network formed by bus nodes is used to collect tracking information about the buses. This information will be used by UCSC's Transportation Department to better manage the campus bus network. Buses transmit their GPS location to base stations deployed throughout campus, which in turn relay location information to a server. The server publishes bus location information on the Internet which can be viewed in real-time through a Google-Maps based graphical user

interface. The tracking information is logged and can be used in the form of mobility traces.

There are approximately thirty to forty buses running on the UCSC campus at any given time. The bus nodes use the 802.11 a/b/g radios to create an ideal platform for testing delay-tolerant networking (DTN) protocols. Additionally, the bus nodes collect rich GPS data sets for further analysis of different types of protocols (e.g., find where and when a connection was established).

3.3.2 Briefcase Nodes

All 20 briefcase nodes are equipped with three 802.11a/b/g radios [4]. They are also equipped with GPS receivers [41] for accurate location updates. The main-board is a mini-ITX computer [47] running Linux Debian Etch. Each briefcase node is equipped with a laptop battery [21]. The electronics are encased using foam inside a waterproof Pelican case.

The idea is to distribute the nodes to people (e.g., students on the UCSC campus) who go about their daily routine, roaming the testbed area while constantly communicating with other nodes in the vicinity, regardless of the type. Using their GPS data, node location can be tracked (e.g., to determine whether specific nodes follow a particular mobility pattern).

3.3.3 Autonomous Ground Nodes

Similar to the briefcase nodes, the autonomous ground nodes use a mini-ITX computer running Linux Debian with three 802.11a/b/g radios. This hardware is carried by iRobot Create robots that can roam both in- and outdoors using various mobility patterns. One pattern is random waypoint, where the robots move in a chosen direction until they reach an obstacle then change direction. More specific mobility patterns can be created where the nodes follow predefined paths. The nodes can move in and out of contact with each other closely emulating mobility scenarios in real life ad-hoc networks in order to thoroughly test protocols.

3.3.4 Aerial Vehicles

Aerial vehicles add several nice features to the testbed. They are able to navigate over any terrain providing network connectivity to otherwise disconnected nodes, and they can cover wider areas in shorter periods of time since they can achieve higher speeds and avoid ground-based obstacles.

Four remote controlled, self-stabalizing helicopters and four autonomous airplanes have been integrated into SCORPION. The planes are equipped with autopilot hardware and software from the open source Paparazzi project [48]. The software allows the plane to circle GPS waypoints, fly between two waypoints, or survey an area defined by four waypoints. The aerial nodes are equipped with 802.11 radios that allow them to communicate with any other node in the network as well as bridge disconnected sections of the testbed.

3.4 Management

For management purposes, the network is divided into *nodes* and *gateways*.

All nodes run a *management agent* that runs as a background process listening and executing commands transmitted by a gateway. Commands are encoded in the form of character strings at the gateway end. When a command is issued by the gateway, it is sent to a list of IP addresses corresponding to the testbed nodes over a UDP socket connection. After the management agent receives one of these commands at the node, it systematically checks the string against a list of registered commands and executes the statements associated with the matching case. Currently the agent will respond to status and power cycle requests.

Status requests are periodically issued by the gateway to ascertain which nodes it can communicate with. When the node receives a status request, it collects information pertaining to its system state that will help with diagnostics at the gateway. Information that the agent gathers includes IP address and host information, uptime, system temperature, kernel version, and versions of software running on the node. This information is funneled into an output file and transmitted over the same UDP socket connection.

Both the bus and briefcase nodes are given software updates using shell scripts accessing an SVN repository. When the nodes start, a bootup script is launched which invokes a series of scripts. Each child script will start a different background process, including management programs as well as networking protocols. Additionally, there

is a scheduler script which has the ability to run software at specified time intervals. This lets the testbed to run several protocols during the length of a single deployment, allowing several researchers access simultaneously.

3.5 Conclusion

In this chapter, an overview of the *SCORPION* (Santa Cruz mObile Radio Platform for Indoor and Outdoor Networks) was given. *SCORPION* is a heterogeneous wireless networking testbed that includes a variety of mobile nodes ranging from ground vehicles to autonomous aerial vehicles. An integral component of SCORPION is its management suite which automates node configuration, monitoring, troubleshooting and testbed scheduling. The testbed is unique due to its node heterogeneity and consequently wide range of mobility scenarios that try to closely model real-world situations (e.g., applications that are prone to frequent, long-lived connectivity disruptions). SCORPION's distinct set of varying node mobility types allows for thorough testing of innovative protocols in the field of wireless networking.

Chapter 4

SlugTransit

4.1 Abstract

SlugTransit is an on-line location-based system which allows public transportation systems to be managed in a cost-effective manner while improving overall system's usability and user satisfaction. *SlugTransit* vehicles report their location in real-time to *SlugTransit* base stations. Vehicle location information is then uploaded and stored in a database. Software updates and log file retrievals are done through *SlugTransit* gateway's. Through a visual, Web-based interface, system operators as well as users have access to current location information in real-time. This capability enables operators to better manage the transportation network, allowing for improved efficiency.

4.2 Introduction

The increasing availability of portable computing devices as well as ubiquity of wireless communication infrastructure have sparked the development and deployment of a wide-range of new applications and services. "Location-based" applications are a notable example of these new class of applications which use location information to provide a variety of services. *SlugTransit* is an on-line location-based system which allows cost-effective management of public transportation services while improving their overall usability and user satisfaction. *SlugTransit* vehicles report their location in real-time to *SlugTransit* base stations; vehicle location information is then uploaded and stored in a database. Software updates and log file retrievals are done through *SlugTransit* gateway's. Through a visual, Web-based interface, system operators as well as users have access to current location information in real-time. This capability enables operators to better manage the transportation network, improving efficiency and, at the same time, user satisfaction. Users are also able to look up real-time bus schedule information available on-line which provides them with a higher-quality user experience.

The name *SlugTransit*[1] refers to the current instantiation of the system which is being used to manage the UC Santa Cruz campus transportation service. Currently, *SlugTransit* consists of ten bus nodes, five base station nodes, and one gateway node.

SlugTransit is also part of *SCORPION* [8], a heterogeneous, disruption-tolerant, wireless networking testbed that can be used by researchers to test their pro-

[1]The name SlugTransit was inspired by UC Santa Cruz's mascot, the banana slug.

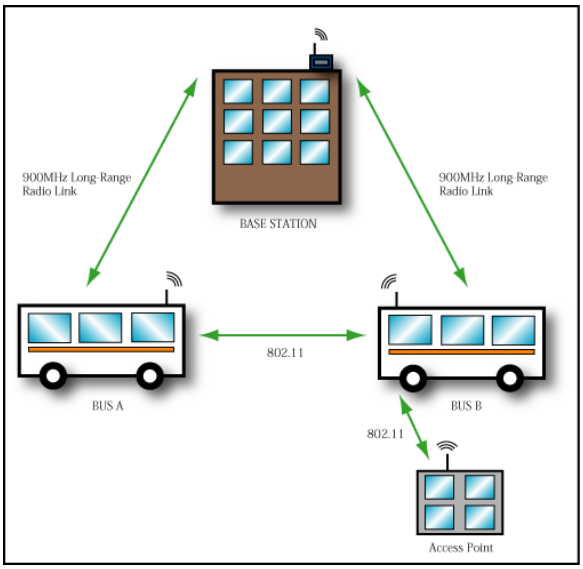

Figure 4.1: *SlugTransit* system overview: location information from vehicle nodes is sent to the base station through a 900MHz network. A separate 802.11 network is used for vehicle-to-vehicle communication.

tocols in a real-world environment under realistic conditions. To this end, *SlugTransit* nodes can communicate among themselves through a separate wireless network. Mobility traces collected from *SlugTransit* vehicles can also be used as realistic traces to drive simulations when evaluating disruption-tolerant network (DTN) protocols.

4.3 Related Work

There has been significant work on location-based services for a variety of applications. In the specific case of transportation management, one notable service is NextBus, a system used by the Bay Area Rapid Transit (BART) [18]. NextBus provides time-of-arrival using current location information. It uses cellular communication to transmit vehicle location. Each bus is equipped with a cellular modem and at specific

intervals, it sends its location to a centralized server. Although cellular technology provides wide coverage, the cost of maintaining a cellular link per vehicle can get prohibitively expensive. In fact, based on the contract between NextBus and BART, the total cost for a seven year contract connecting four BART stations is estimated to cost approximately $1,000,000 [2]. *SlugTransit*, on the other hand, does not require any maintenance fee; its one-time cost is for setting up the 900MHz infrastructure and to install the tracking device on to the buses.

Another system to be pointed out is UMass's DieselNet [10] whose main purpose was to serve as a networking research testbed. DieselNet vehicles relay location information through 802.11b WiFi radios in a disruption-tolerant manner. This means that location information traverses from bus to bus until it gets to some centralized location. This is one of the main differences between DieselNet and *SlugTransit*, where location information is transmitted through a dedicated 900MHz network where connectivity is guaranteed. This ensures minimum delay in providing up-to-date vehicle location information to operators and users.

4.4 System Overview

Public transportation can be disturbed by several factors including traffic situations, vehicle breakdowns, route changes, "hot spots", etc. *SlugTransit* aims at providing users and system operators with on-line up-to-date vehicle location information to ensure efficient and high-quality service.

SlugTransit vehicles are outfitted with nodes equipped with a GPS tracking device and a 900MHz long-distance radio to track vehicles on their routes. Location data is collected using the GPS device and beaconed out through the 900MHz radio. Location data is then received by base stations, which also uses a 900MHz network to relay it to a centralized server.

This data is then made available through a visual Web-based interface; we have also made it available on portable mobile devices, specifically the iPhones, using the Google Maps API [23] [2]. Through the *SlugTransit* interface, users and operators can access exact location of vehicles from nearly anywhere and plans can be made accordingly even when problems occur.

SlugTransit vehicles are also equipped with 802.11 WiFi radios providing a separate network dedicated to vehicle-to-vehicle communication. This network is intended as a way to test and evaluate wireless network protocols, specifically protocols designed to operate in connectivity-challenged network scenarios. It can also be used to communicate with other nodes in the *Scorpion* [8] testbed. Additionally, *SlugTransit* vehicles can collect GPS-rich data sets to be used in trace-driven simulations of wireless network protocols as well as statistical studies of user mobility. The 802.11 network is also used for pushing software updates and collecting log files through a gateway node (described in detail in Section 4.7).

In summary, *SlugTransit* is able to relay up-to-date vehicle location information reliably and thus reflect the dynamics of the day-to-day operation of a public

[2]The Google Maps API provides the ability to embed Google Maps into Web sites using JavaScript; it also comes with abundant map manipulation utilities.

transportation service. It is instrumental at helping system operators provide high-quality service to users in a cost-effective way.

4.5 Graphical User Interface

The graphical Web-based user interface (shown in Figure 2) is a key part of the system. *SlugTransit*'s Web site is available to anyone with Internet access. However, a portion of the Web site is meant to be accessible only by service operators and is thus password protected. *SlugTransit* maps are also accessible using smart phones such as the iPhone.

4.5.1 Transportation System User Interface

Users access real-time vehicle location information through a Web site created using the Google-Maps API. In the current implementation, the Web site refreshes every 2.5 seconds to offer up-to-date location of vehicles. Location information is displayed on the map in color-coded dots depending on which route the vehicle is on. This is the same on an iPhone. The server will detect the iPhone and direct the user to the appropriate page and display the map with the current location of all of the vehicles. Figure 3 shows the iPhone interface.

4.5.2 System Operator Interface

System operators are also provided with a Web site and given administrator privileges to, e.g., configure route information for each vehicle. These updates show up

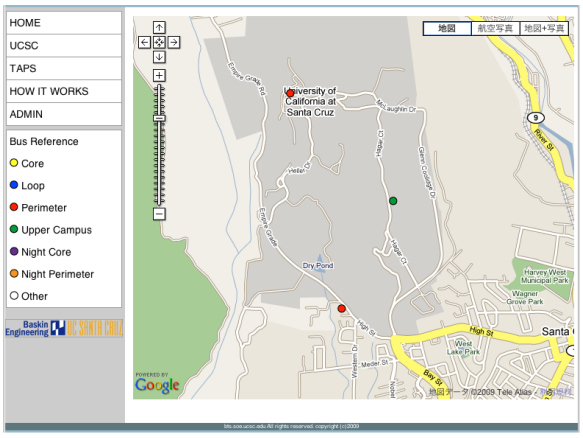

Figure 4.2: *SlugTransit* user interface based on the Google Maps API

on the main Web site (accessible to users of the transportation service) in real-time.

4.6 Hardware

SlugTransit hardware is based on a Mini-ITX [47] computer running Linux and depending on the type of node, it is equipped with different external devices as follows:

- **Vehicle Node:** GPS tracking device, 900MHz radio, and three 802.11a/b/g WiFi radios.
- **Base Station Node:** 900MHz radio.
- **Gateway Node:** 802.11a/b/g WiFi radio.

Figure 4.3: *SlugTransit* User Interface for the iPhone

4.6.1 900MHz Radio

AC4790 900MHz transceiver radios from Aerocomm [1] are used to populate the 900MHz network. The AC4790 uses frequency hopping spread spectrum in the 900MHz ISM band to provide high-performance communication, while following the FCC part 15.247 regulations. Therefore, it does not have FCC licensing issues allowing *SlugTransit* to be usable by any public transportation provider/operator in the United States.

4.6.2 GPS Tracking Device

Pharos' iGPS-500 [41] receiver collects vehicles' GPS coordinates. The NMEA-0183 standard protocol is used and the 'GGA' message containing 3-Dimentional location and accuracy data, is extracted from the output message to provide the longitude and latitude corresponding to the vehicle's current location. This device connects through USB and thus can be connected through an available USB port.

4.6.3 802.11a/b/g Radios

Atheros 802.11a/b/g WiFi radios [5] are used to enable communication between vehicles and *Scorpion* testbed nodes. The radios are also used for software updates and collecting log files. They are configured to be on the same subnet to enable communication between them. Figure 4 shows a snippet of a GPS log file collected on June 10th 2009.

```
...
1243005797 36.989944 -122.066994
1243005798 36.989922 -122.067055
1243005799 36.989891 -122.067123
1243005800 36.989861 -122.067184
1243005801 36.989830 -122.067253
1243005802 36.989796 -122.067337
1243005803 36.989758 -122.067398
1243005804 36.989719 -122.067467
1243005805 36.989681 -122.067513
1243005806 36.989651 -122.067558
1243005807 36.989635 -122.067596
1243005808 36.989620 -122.067612
1243005809 36.989613 -122.067612
...
```

Figure 4.4: Snippet of a GPS log file on June 10th 2009.

4.7 Current Deployment

Currently, *SlugTransit* has been deployed in the UCSC campus and used by UCSC campus transportation service. As of now, there are a total of ten buses outfitted with vehicle nodes, five base stations that are deployed to provide a campus wide coverage, and a gateway node installed at the bus depot that performs software updates and log retrieval.

4.7.1 Vehicle Node

Vehicle nodes are installed in the cabinet located above the bus driver's seat. Since the Mini-ITX computer has one of the smallest form factors available in the market, it allows easy installation on the bus without cramming the bus driver's area. The power is connected through three input wires on the M3-ATX power supply [14]:

Battery +, Battery -, and ignition. This power supply is designed for use in a vehicle and it can handle an input voltage from 6V - 24V to accommodate engine cranks, under-voltage and over-voltage situations. It also takes care of power challenges when installing a computer in a vehicle. Generally when vehicles are powered off, the computer connected to the battery still consumes power and drains the battery. To avoid this, this power supply has an 8-bit micro-controller unit that cuts-off the power completely after some pre-defined time.

4.7.2 Base Station Node

Base station nodes are deployed throughout campus to provide a 900MHz coverage over all bus routes. The current deployment allows maximum coverage around campus with minimal number of base stations. The approach used to choose the best location to deploy base stations depended on how high the building is, if the site had Internet connectivity, and the range of the 900MHz radio. Each site's 900MHz range was checked around campus using a 900MHz transceiver to determine the approximate range in order to provide optimal coverage. Additionally, the deployment was done in such a way as to allow a small overlap between neighboring base stations to avoid drop outs by allowing smooth hand-offs between base stations. Figure 5 shows the approximate 900MHz coverage.

This infrastructure also allows the bus operators to use the system without any monthly maintenance fee. Compared to coverage using a cellular technology, where each communication device usually comes with a monthly service charge, the

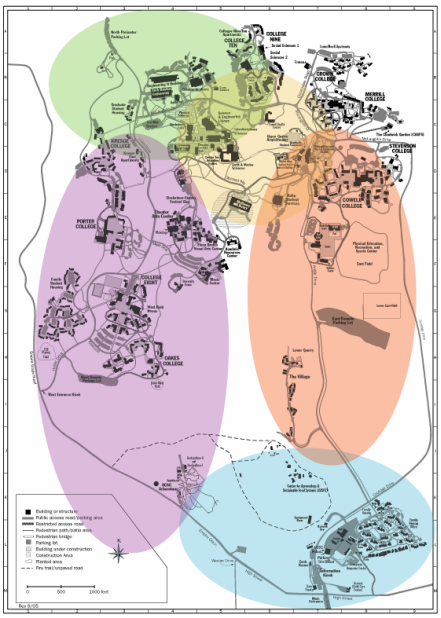

Figure 4.5: Approximate 900MHz coverage.

maintenance fee scales with the number of participating vehicles and thus can get overwhelmingly expensive. In *SlugTransit*, once the infrastructure is installed, no additional maintenance fees are needed.

4.7.3 Gateway Node

A gateway node has been installed in an office near the gas station in the bus depot. During a normal day, a bus comes into the bus depot to fill up their gas at least once a day. When the bus is turned on, a series of scripts are activated and start communicating with the gateway node. (This is discussed in further detail in Section 4.8.) Since every bus comes into the bus depot, it allows prompt updates of software and log files to be uploaded.

4.8 Management

Vehicle nodes are updated via bash scripts along with the usage of an SVN repository. When a vehicle starts its engine, the vehicle node detects that it was turned on and initiates a boot up script. Once the boot up script launches, it invokes a series of other scripts that are responsible for executing different tasks. One of the script's main task is to start the update process, which is done by making an update call to the SVN repository. The update procedure is quite simple and this script executes a command to compare its revision number against the SVN repository's revision number to check for any updates. If there exists files with a newer revision number, the bus node will download the new files through the gateway node and then finishes the update procedure.

Since these nodes do not have continuous access to the Internet, the updates are done only when it has contact with the gateway node. A gateway node acts as a bridge to the Internet for vehicles. In order to check if vehicles are in contact with the gateway node, one of the scripts make an attempt to ping the gateway node on boot up and if it is successful, the script goes on to the update procedure. Otherwise, it skips updating and continues booting up the node. The ping command is useful to avoid any delays when vehicles turn on because vehicles are not in contact with the gateway node at all times.

4.9 Conclusion

In this chapter, the *SlugTransit* was introduced, which is an on-line location-based system which allows cost-effective management of public transportation systems while improving their overall usability and user satisfaction. *SlugTransit* vehicles report their location in real-time to *SlugTransit* base stations; vehicle location information is then uploaded and stored in a database. Software updates and log file retrievals are done through *SlugTransit* gateway's. Through a visual, Web-based interface, system operators as well as users have access to current location information in real-time. This capability enables operators to better manage the transportation network, improving efficiency and, at the same time, user satisfaction. Users are also able to look up real-time bus schedule information available on-line which provides them with a higher-quality user experience.

We have also described the current *SlugTransit* deployment aiming at helping manage UCSC's campus transportation services. As future work, we plan to expand the current deployment to include other vehicles that are part of UCSC's transportation network (e.g., disability vans, bicycle shuttles, etc.). We also plan to deploy *SlugTransit* in the Santa Cruz Metropolitan region.

4.10 Acknowledgement

Thank you to the following members of the UCSC community: Bruno Nunes, Vladislav Petkov, Brad Smith, Tracy Freeman, Larry Pageler, John Steele, Jim Warner,

Andrew Kong, Tony Yu, Uladzimir Karneyenka, Eduardo Villafana, Stephen Petersen, and Kip Laws.

Chapter 5

Conclusion

This thesis covered three different areas of wireless mobile networks. Chapter 2 discussed a novel delay-tolerant routing algorithm called TAROT which is able to provide a communications link in a heavily partitioned network which would otherwise be impossible to route across using existing technologies such as AODV. TAROT is able to match the 100% delivery ratio of the Epidemic routing protocol, while providing a solution to reducing the overhead generated by Epidemic's overuse of replicating messages in the network. The TAROT protocol was able to keep end-to-end delay to within 20% of Epidemic, while reducing the overhead by as much as 60%.

The SCORPION project was introduced, which aims to develop a testbeds at UC Santa Cruz to test the next generation of DTN, medium-access control (MAC), and MIMO radio protocols. SCORPION uses a variety of node-types including human mobility, aerial nodes, vehicular mobility, and disaster response nodes. SCORPION bridges an important part of research on wireless networks by providing real-world

environments to test and analyze the functionalities of the protocols.

Finally, the SlugTransit project was introduced. SlugTransit is being developed to provide public transit systems with a method of tracking vehicles in a cost-effective manner. This is accomplished by establishing a network of base-stations which are used by the vehicles as location uplinks. Because there is no recurring or monthly charge for using these off-the-shelf base-stations, costs are kept at a minimum and the system scales. Our experience with SlugTransit over the past several months has shown that the system is reliable and accurate.

Bibliography

[1] Aerocomm AC4790-1000. 1watt 900mhz radio. http://www.aerocomm.com/, Last visited in 2008.

[2] ACTransit. Action memo. http://www.pharosgps.com/, Last visited in 2009.

[3] National Marine Electronics Association. Nmea. http://www.nmea.org, Last visited in 2009.

[4] Atheros. 802.11 radio. http://www.atheros.com/, Last visited in 2008.

[5] Atheros. 802.11 radio. http://www.atheros.com/, Last visited in 2009.

[6] Aruna Balasubramanian, Brian Neil Levine, and Arun Venkataramani. Dtn routing as a resource allocation problem. In *ACM SIGCOMM*, 2007.

[7] J. Boice, J. J. Garcia-Luna-Aceves, and K. Obraczka. Disruption-tolerant routing with scoped propagation of control information. In *IEEE ICC*, 2007.

[8] S. Bromage, J. Koshimoto, C. Engstrom, M. Bromage, V. Petkov, B. Nunes, H. Taylor, K. Obraczka, S. Dabideen, M. Hu, R. Menchaca-Mendez, D. Nguyen, D. Sampath, JJ Garcia-Luna-Aceves, H. Sadjadpour, and B. Smith. Scorpion: A heterogenous wireless networking testbed. In *ACM SIGMOBILE Mobile Computing and Communications Review*, 2009.

[9] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *IEEE INFOCOM*, 2006.

[10] John Burgess, Brian Gallagher, David Jensen, and Brian Neil Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. *In Proc. IEEE INFOCOM*, 2006.

[11] B. Burns, O. Brock, and B. N. Levine. Mv routing and capacity building in disruption tolerant networks. In *IEEE Infocom*, 2005.

[12] B. Burns, O. Brock, and B. N. Levine. Mora routing and capacity building in disruption-tolerant networks. In *Elsevier Ad Hoc Networks Journal*, 2008.

[13] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, and R. Gass. Impact of human mobility on the design of opportunistic forwarding algorithms. In *IEEE Infocom*, 2006.

[14] Ituner Networks Corp. M3-atx automotive power supply. http://www.mini-box.com/M3-ATX-DC-DC-ATX-Automotive-Computer-car-PC-Power-Supply, Last visited in 2009.

[15] P. De, A. Raniwala, S. Sharma, and T. Chiueh. Mint: a miniaturized network testbed for mobile wireless research. *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 4:2731–2742 vol. 4, March 2005.

[16] L. F. M. de Moraes and B. A. A. Nunes. Calibration-free wlan location system based on dynamic mapping of signal strength. In *ACM MobiWac*, 2006.

[17] DeerNet. www.wu.ece.ufl.edu/projects/DeerNet/DeerNet.html.

[18] San Francisco Bay Area Rapid Transit District. Bart - about bart. http://www.bart.gov/about/index.aspx, Last visited in 2009.

[19] Draganfly. Self stabilizing helicopter. http://www.rctoys.com/rc-toys-and-parts/DF-SAVS/INDUSTRIAL.html, Last visited in 2008.

[20] DTNRG. http://www.dtnrg.org.

[21] Valence Electronics. Vnc-130. http://www.valence.com/, Last visited in 2008.

[22] S. Ganu, H. Kremo, R. Howard, and I. Seskar. Addressing repeatability in wireless experiments using orbit testbed. *Testbeds and Research Infrastructures for the Development of Networks and Communities, 2005. Tridentcom 2005. First International Conference on*, pages 153–160, Feb. 2005.

[23] Google. Google maps api - google code. http://code.google.com/intl/en-EN/apis/maps/, Last visited in 2009.

[24] S. Guo, M. H. Falaki, E. A. Oliver, S. Ur Rahman, A. Seth, M. A. Zaharia, and S. Keshav. Very low-cost internet access using kiosknet. In *ACM Sigcomm*, 2007.

[25] A. J. Hooke. Towards an interplanetary internet: A proposed strategy for standardization. In *SpaceOps*, 2002.

[26] NextBus Inc. Nextbus/how nextbus works. http://www.nextbus.com/corporate/works/index.htm, Last visited in 2009.

[27] iRobot. irobot create programmable robot. http://www.irobot.com/sp.cfm?pageid=305, Last visited in 2008.

[28] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *ACM SIGCOMM*, 2004.

[29] D. Johnson, T. Stack, R. Fish, D. M. Flickinger, L. Stoller, R. Ricci, and J. Lepreau. Mobile emulab: A robotic wireless and sensor network testbed. *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, April 2006.

[30] D. B. Johnson, D. A. Maltz, Y. C. Hu, and J. G. Jetcheva. Dynamic source routing in ad hoc wireless networks. Mobile Computing, 1996.

[31] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li-Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *ACM ASPLOS-X*, 2002.

[32] B. Karp and H. T. Kung. Gpsr: greedy perimeter stateless routing for wireless networks. In *ACM Mobicom*, 2000.

[33] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. *ACM Wireless Networks*, 6(4), 2000.

[34] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Nonnumerical algorithms and problems—geometrical.

[35] J. Lebrun, Chen-Nee Chuah, D. Ghosal, and Michael Zhang. Knowledge-based opportunistic forwarding in vehicular wireless ad hoc networks. In *Vehicular Technology Conference*, 2005.

[36] Jeremie Leguay, Timur Friedman, and Vania Conan. Dtn routing in a mobility pattern space. In *ACM SIGCOMM Workshop on Delay-tolerant networking*, 2005.

[37] Ilias Leontiadis and Cecilia Mascolo. Geopps: Geographical opportunistic routing for vehicular networks. In *WoWMoM AoC*, 2007.

[38] Anders Lindgren, Avri Doria, and Olov Schelen. Probabilistic routing in intermittently connected networks. In *ACM SIGMOBILE*, 2003.

[39] C. E. Perkins, E. M. Royer, and S. R. Das. Ad hoc on-demand distance vector routing. IETF Internet Draft http://www.ietf.org/internet-drafts/draft-ietf-manetaodv-03.txt., 1999.

[40] Pharos iGPS-500. http://www.pharosgps.com.

[41] Pharos. Gps receiver. http://www.pharosgps.com/, Last visited in 2009.

[42] Qualnet Simulator. http://www.scalable-networks.com.

[43] Natasa Sarafijanovic-Djukic and Matthias Grossglauser. Last encounter routing under random waypoint mobility. In *Networking*, 2004.

[44] Y. Shang, H. Shi, and A. Ahmed. Performance study of localization methods in ad-hoc sensor networks. In *IEEE Mass*, 2004.

[45] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *ACM WDTN*, 2005.

[46] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility. In *IEEE PERCOM*, 2007.

[47] Logic Supply. Mini itx board. http://www.logicsupply.com/products/, Last visited in 2009.

[48] Paparazzi Team. Paparazzi, the free autopilot. http://paparazzi.enac.fr, Last visited in 2008.

[49] UnitedVillages. http://www.unitedvillages.com.

[50] Amin Vahdat and David Becker. Epidemic routing for partially-connected ad hoc networks. Technical report, Duke University, 2000.

[51] G. Werner-Allen, P. Swieskowski, and M. Welsh. Motelab: a wireless sensor network testbed. *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 483–488, April 2005.

[52] Alec Woo. Mote documentation and development information. *http://www.eecs.berkeley.edu/ awoo/smartdust/*, 2000.