TAROT: Trajectory-Assisted Routing for Intermittently Connected Networks

Matthew K. Bromage
Computer Engineering, Baskin Engineering
University of California at Santa Cruz
California, USA
mbromage@soe.ucsc.edu

James T. Koshimoto
Computer Engineering, Baskin Engineering
University of California at Santa Cruz
California, USA
jkoshimo@soe.ucsc.edu

Katia Obraczka
Computer Engineering, Baskin Engineering
University of California at Santa Cruz
California, USA
katia@soe.ucsc.edu

ABSTRACT

We introduce *TAROT* (Trajectory-Assisted ROuTing), 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 vehicles, 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. In its current implementation, *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.

We evaluate *TAROT*'s performance through simulations using the QualNet network simulator. A side-by-side comparison against Epidemic Routing [26] 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.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless Communication; C.2.2 [**Network Protocols**]: Routing Protocols

CHANTS'09, September 25, 2009, Beijing, China.

General Terms

Algorithms, Performance, Reliability

Keywords

delay tolerant networks, routing, ad-hoc networks, intermittent connectivity

1. 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 [20] and DSR [12]) 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" [10]. 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" [10] 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

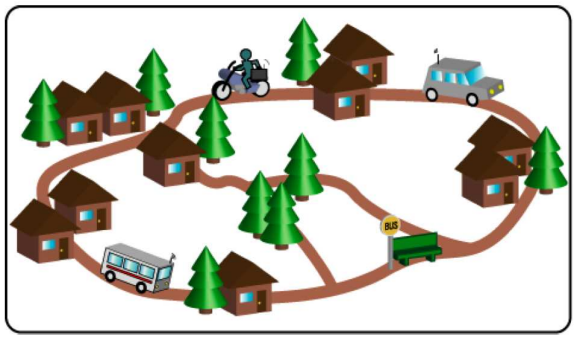

Figure 1: A sample village scenario with various types of mobility. The mobility of vehicles tend to exhibit patterns such as recurring paths. The path information can be leveraged to make more informed routing decisions in a hop-by-hop opportunistic manner.

KioskNet [9] and UnitedVillages [25]. 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. 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 [13] and DeerNet [7] 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" [8] 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 [26], 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 [23] 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 [24] 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 3) 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 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 [11], indicates diminishing returns.

The paper is organized as follows: the next section provides an overview of the *TAROT* protocol, outlining each of the functional blocks. Section 3 discusses the Path Detection (PD) algorithm, followed by a discussion of the Routing Decision Engine (RDE) in Section 4. Section 5 will then showcase the results of applying PD in combination with RDE in a variety of scenarios. Finally we summarize the paper with related work and conclusion.

2. 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 4.

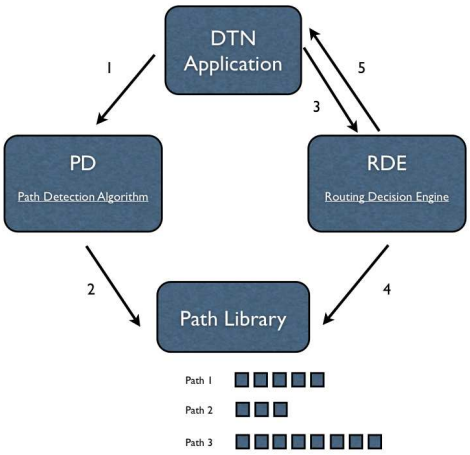

Figure 2: *TAROT* functional blocks. 1) The application sends the current location to the Path Detection (PD) algorithm. 2) The PD algorithm extracts paths and stores them in the path library. 3) When two nodes encounter, the application sends a list of message destinations to the RDE. 4) The RDE uses 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.

Figure 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 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).

3. 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

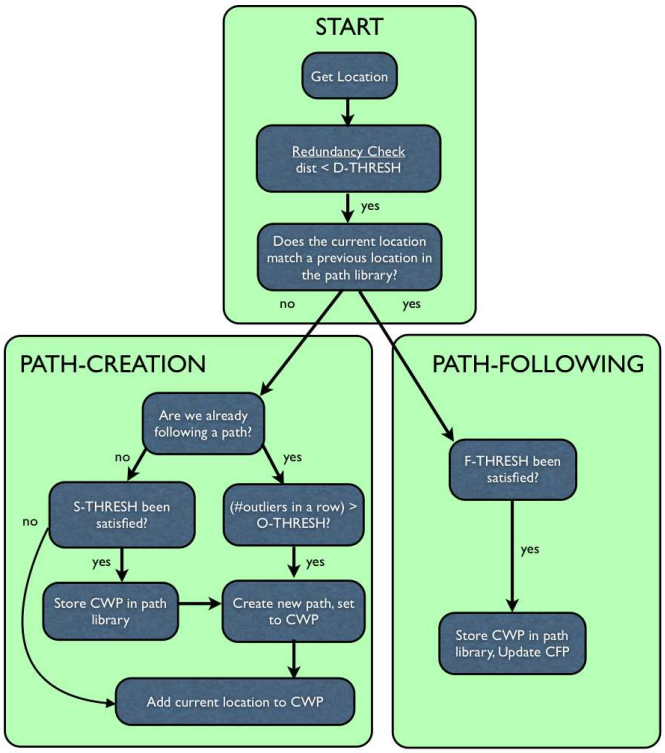

Figure 3: Flowchart for the Path Detection (PD) algorithm indicating the PATH-CREATION and PATH-FOLLOWING modes of operation. The process is repeated for each sampled coordinate.

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.

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 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

[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.

[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.

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 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.

3.1 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. 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 3.2).

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 4.

The decision process involved in PATH-CREATION is outlined in Figure 3. Figure 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.

Figure 4: UCSC campus shuttle system routinely followed paths. From left to right: upper campus shuttle, core shuttle, loop shuttle.

3.2 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.

4. 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.

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 [5].

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 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,

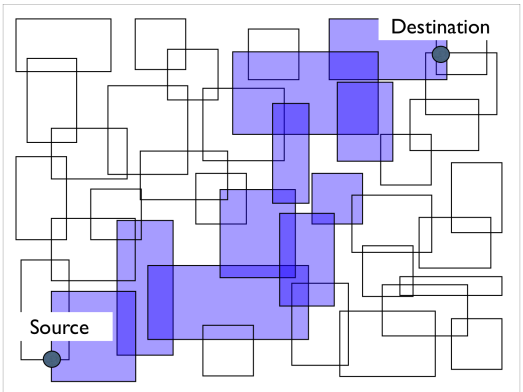

Figure 5: A sample route used by *TAROT*'s controlled epidemic forwarding approach. The light-shaded rectangles indicate structured mobility patterns detected by *TAROT*'s PDA. The dark-shaded rectangles indicate nodes selected to carry message replicas from *Source* to *Destination*.

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. 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.

4.1 Controlled Epidemic Forwarding

As illustrated in Figure 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 [14], 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:

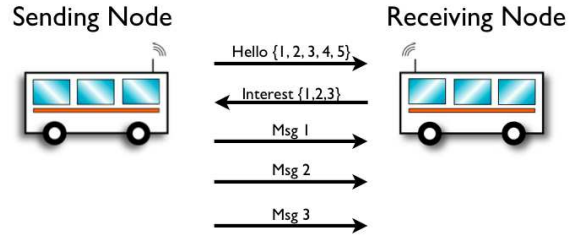

Figure 6: Handshake during an encounter. The *SendingNode* sends a *Hello Message* containing a summary vector of messages it carries. The *ReceivingNode* then sends an *Interest Message* for the messages it should relay. The *SendingNode* then forwards a copy of those messages.

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* are similar to the ones used in Epidemic Routing [26], 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.

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

4.2 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.

5. RESULTS

We test *TAROT* 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 [26] 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).

5.1 Simulation Setup

We use the Qualnet simulator 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 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.

5.2 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 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.

Table 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

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 5.5. 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.

5.3 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).

5.3.1 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.

5.3.2 Long-Lived Flows

In this experiment we steadily increase the number of

Table 2: Short-Lived Flow Results

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

Table 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%

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 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.

5.4 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 5.1). 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 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 8 and 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.

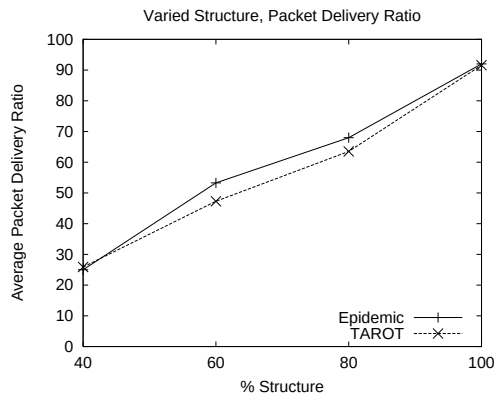

Figure 7: *TAROT* is able to maintain an almost identical delivery ratio compared with Epidemic as the amount of structure present in network varies.

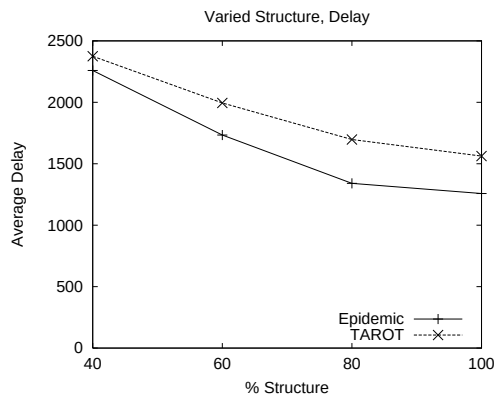

Figure 8: The average delay for successfully received data packets at the destination.

5.5 SCORPION Bus Traces

Using the SCORPION [3] 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 destinations positioned at five campus bus stops, the locations of which were randomly chosen around campus.

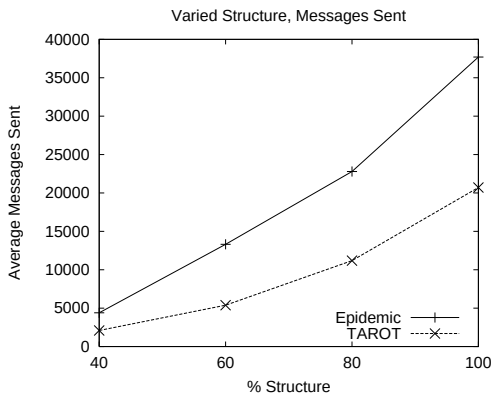

Figure 9: Average Overhead for Varied Structure Experiment

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.

6. 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, 2, 19, 1, 21, 16].

"Spray-and-Wait" [23] 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" [24] 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 [4] 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 [5] 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 [15] and GPSR [14], *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 [22]).

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 [17]. In the previously discussed GeOpps protocol [18], 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.

7. CONCLUSION

This paper 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. In its current implementation, *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 [26] 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.

As future work, there are a number of extensions planned for *TAROT*'s base algorithm. We plan to extend *TAROT*'s Path Detection to increase the accuracy and freshness of the extracted paths using techniques such as path aging. We also plan to address destination location dissemination and enable routing for mobile destinations.

8. 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.

9. REFERENCES

[1] A. Balasubramanian, B. N. Levine, and A. Venkataramani. Dtn routing as a resource allocation problem. In *ACM SIGCOMM*, 2007.

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

[3] 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, J. Garcia-Luna-Aceves, H. Sadjadpour, and B. Smith. Scorpion: A heterogenous wireless networking testbed. In *ACM SIGMOBILE Mobile Computing and Communications Review*, 2009.

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

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

[6] 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.

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

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

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

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

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

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

[13] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *ACM ASPLOS-X*, 2002.

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

[15] Y.-B. Ko and N. H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. *ACM Wireless Networks*, 6(4), 2000.

[16] J. Lebrun, C.-N. Chuah, D. Ghosal, and M. Zhang. Knowledge-based opportunistic forwarding in vehicular wireless ad hoc networks. In *Vehicular Technology Conference*, 2005.

[17] J. Leguay, T. Friedman, and V. Conan. Dtn routing in a mobility pattern space. In *ACM SIGCOMM Workshop on Delay-tolerant networking*, 2005.

[18] I. Leontiadis and C. Mascolo. Geopps: Geographical opportunistic routing for vehicular networks. In *WoWMoM AoC*, 2007.

[19] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. In *ACM SIGMOBILE*, 2003.

[20] 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.

[21] N. Sarafijanovic-Djukic and M. Grossglauser. Last encounter routing under random waypoint mobility. In *Networking*, 2004.

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

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

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

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

[26] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Technical report, Duke University, 2000.