MORA Routing and Capacity Building in Disruption-Tolerant Networks

Brendan Burns Oliver Brock Brian Neil Levine
Department of Computer Science
University of Massachusetts Amherst
{bburns,oli,brian}@cs.umass.edu

***Abstract*— Disruption-tolerant networks (DTNs) differ from other types of networks in that capacity is created by the movements of network participants. This implies that understanding and influencing the participants' motions can have a significant impact on network performance. In this paper, we introduce the routing protocol MORA, which learns structure in the movement patterns of network participants and uses it to enable informed message passing. We also propose the introduction of autonomous agents as additional participants in DTNs. These agents adapt their movements in response to variations in network capacity and demand. We use multi-objective control methods from robotics to generate motions capable of optimizing multiple network performance metrics simultaneously. We present experimental evidence that these strategies, individually and in conjunction, result in significant performance improvements in DTNs.**

I. Introduction

Many routing protocols exist to support end-to-end messaging in mobile ad hoc wireless networks. Such protocols assume an end-to-end connection through a contemporaneous set of links through intermediary peers [19], [27]. As a result, if a path between two peers in a network does not exist, communication is not possible, and the route creation process fails.

A growing body of work is exploring techniques for routing network traffic over asynchronous paths to adapt to situations where routes cannot be created from contemporaneous links. Such networks have varied names: highly partitioned networks [10], [17], message ferrying [41], [42], delay-tolerant networks [12], and disruption-tolerant networks (DTNs) [11]. To enable end-to-end routing in a DTN (the term we choose for this paper), network participants are relied upon to carry and deliver messages of others. Whenever two participants pass, they negotiate the exchange of messages. A message may be passed between a number of network participants before reaching its destination.

There are many scenarios in which DTNs are the most viable routing solution for several reasons. First, it is challenging to deploy an unpartitioned network using mobile nodes that roam a large geographic area. This can be the result of relatively short radio ranges; obstructions can also curtail the reach of devices, including buildings, trees, mountains, or water for amphibious devices (as can happen in wildlife monitoring [32]). Our own DTN, DieselNet [4] which is deployed using 802.11 on 30 buses, spans an area of 150 sq. miles, beyond the range of WiMax and our area's cellular phone coverage. In underwater acoustic networks [1], [26], DTNs are more critical as no fixed infrastructure exists and less than ten expensive autonomous underwater vehicles may roam an area hundreds of km in diameter carrying acoustics transducers that reach only 2–5 km. Second, battery-powered devices may adhere to energy management schemes that power down radios [31], [2], effectively decreasing the population of the network. Finally, DTNs can arise naturally from dense, infrastructure-based deployments. For example, a municipal mesh that is deployed to support radios on vehicles can provide robust, unpartitioned coverage. However, vehicles may leave the geographic coverage of the mesh; a DTN can extend networking to an area beyond the municipality's mesh by taking advantage of opportunities for transfer between vehicular nodes that arise intermittently.

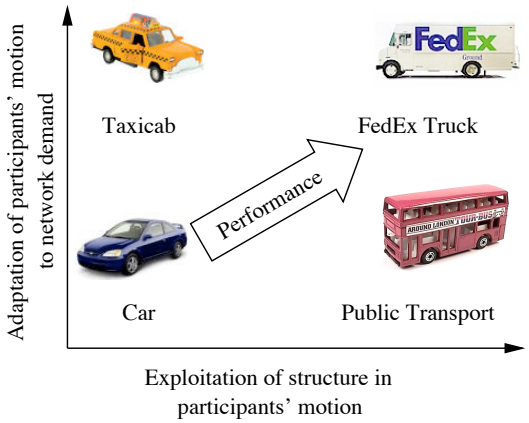


Fig. 1. Classification of routing methods for DTNs based on characteristics of participants' movement patterns.

The performance of disruption-tolerant networks depends on many factors, including the number of network participants, their storage capacity, communication capabilities, and movement patterns. In this paper, we focus on a performance factor unique to DTNs, namely the movement patterns of

This research was supported in part by National Science Foundation awards ANI-0133055, CNS-0519881, IIS-0545934, and EIA-0080199 and in part by NASA award NAG9-1445#1.

participants. We examine how movements can be exploited or controlled to improve performance in DTNs. We classify movements of network participants according to two independent properties: their inherent structure and their adaptiveness to the demand in the network (see Figure 1). In this context, structure refers to periodic patterns in peers' movements that can be exploited to estimate the probability of delivery for a specific message and peer.

For purposes of illustration, we relate this classification of participant's motion and associated routing protocols to everyday experience. The lower-left of the classification shown in Figure 1 corresponds to hitchhikers being picked up by randomly moving cars. In this scenario, cars move without periodicity and do not adapt to the route of the hitchhiker. The lower-right of the figure corresponds to public transport with fixed schedules. Independently operating taxicabs that pick up passengers in the street are represented in the top-left of the diagram. Finally, FedEx trucks are situated in the top-right corner. (Here, packages are transported, rather than people but the analogy holds.) FedEx trucks travel on structured daily routes, but only stop for scheduled pickups and deliveries, i.e., their coordinated routes are adjusted in response to demand.

From this description it is clear that performance metrics, such as bandwidth and latency, can be expected to improve for scenarios that can be classified towards the top-right of the diagram. To achieve these performance improvements, an increased amount of coordination among the network participants is required. Such coordination can exploit structure present in participant's motion patters to improve the efficiency of routing. A coordination of network participants that adapts the motion to the network's demands enables a better usage of the participants' capacity and thus can lead to increased bandwidth and reduced latency. The contribution of this paper lies in algorithms that take advantage of these insights to improve the performance of disruption-tolerant networks.

Contributions. To address the incongruence between the movement of network participants and traffic flow that may arise when movement pattern of participants do not match bandwidth requirements, we propose the introduction of autonomous agents as participants into the network. These agents can be ground-based [36] (see Figure 2), airborne [8], or underwater mobile robots [13], [14], [38], [34].

We propose methods for adapting the motion of such agents to bandwidth and latency requirements of a network. We call our approach *Multi-Objective Robotic Assistance* (MORA), which allows a set of autonomous agents in the DTN to enhance network performance by delivering packets. While the problem of choosing optimal motions for autonomous agents in this context is shown in this paper to be NP-hard, we propose techniques from robotic control that are able to obtain high-quality approximations to the optimal solution. We found experimentally that the addition of agents can have significant improvements to network performance; for example, in the case where it is paired with MaxProp [4], MORA increases average delivery rates from 67% to 88% while simultaneously reducing average delivery latency from 120 minutes to 76 minutes. MORA is independent of the underlying DTN routing protocol used by nodes, and our evaluations show that it improves the performance of random, FIFO, ME/DLE [5], [4], and MaxProp [4] protocols.

Fig. 2. The UMass Segway RMP, the subject of our previous work [36], is a robotic autonomous agent that can act as a network participant to improve performance in disruption-tolerant networks. Other platforms include robotic airborne vehicles [8] and underwater mobile robots [13], [14], [38], [34].

II. Related Work

Since this work is a synthesis of ideas from networking and robotic control, it has related work in both areas.

A. Networking

DTN routing has been studied by a growing number of researchers. As we stated in the introduction, we can taxonomize previous work based on their assumptions about the inherent structure of the network and the adaptiveness of peers to the demand in the network.

Along with the preliminary versions of this work [5], [6], our work is distinguished by its exploration of the adaptation of peer movements to meet communication needs of the network. We propose the use of robotic peers to improvement performance. Previous to our efforts, Li and Rus [23] proposed algorithms that minimally alter node movements for delivery of a single packet in a partially disconnected network. They assume peers have a communication channel that periodically broadcasts the locations of all other peers with bounded error though the channel does not allow the transfer of data. The authors point out that it is difficult to extend the algorithm to efficiently support multiple packets at once. Our work does not have such limitations.

We distinguish other work by the degree to which peer movements are fixed and well-known. At one end of the spectrum, Zhao, et al. [41], [42] proposed DTNs based on ferries, which are peers that have completely predictable movements through the geographic area (e.g., a city bus or

river ferry). Peers route message end-to-end by scheduling their movements to meet with the ferry. In later work, Zhao, et al. [43] propose a method for designing multiple fixed ferry routes. In comparison, our method dynamically adjusts routes based on perceived load, learned by agents from within the DTN itself. Also related to the paradigm of adding resources to a DTN is our work on throwboxes, which are solar-powered, stand alone computers with radios and storage [44], [2]. From one point of view, they play the role of stationary ferries.

At the other end of the spectrum is a series of papers that attempt to learn patterns in the movements of peers. In 2001, we proposed a routing algorithm for highly partitioned networks by exploring a number of different strategies for deciding which messages to exchange when two network participants meet [10]. Our algorithm, called *Drop-Least Encountered*, had peers keep track of the other peers they meet regularly over time. Peers initialize their estimate of the likelihood of message delivery to a moving peer as 0. When a peer A meets another peer B, the former sets the likelihood of delivering messages to B as 1. Then A takes a portion of B's likelihood of delivering messages to the other peers in the system. These values degrade over time, such that they are reinforced only if A and B meet periodically. Versions of this same algorithm, some more advanced, were subsequently proposed by others [24], [28], [16], with each paper showing a different analysis of the problem.

In our previous work, we have also proposed using acknowledgments of delivered data to remove stale data from network buffers [4]. Schemes also exist that use network coding [39], or restrict the number of copies of a packet [33], [30], [37], [25].

Also relevant to this paper is work by Jain et al. [18], who showed that networks that have a large number of connection opportunities require less intelligent forwarding algorithms. As resources become scarce, increasing availability of information about the network increases the performance of DTN routing.

There are other challenges within the subject of DTNs. For example, an information retrieval service can be a vital service in a DTN used by disaster management workers. In our previous work, we proposed a method of dividing up a database such that any small random subset of peers can answer queries with high accuracy even though each peer carries only a small fraction of the full database [17]. In our method, no routing is required, yet it is robust despite the movement of peers, who may change groups at any time.

B. Multi-Objective Control

In a disruption-tolerant network, robotic agents deliver packets by performing physical motions. During their motions, the agents encounter sources, sinks, and other agents. At any point in time, an agent may carry packets from multiple sources to be delivered to a variety of destinations. Given these destinations, an adequate motion of the agent has to be determined. This motion will have to satisfy multiple objectives determined by the destinations of network packets, the need to encounter other agents for message passing, and by desired performance metrics for the network. In Section III-A, we show that the computation of an optimal motion for agents in a disruption-tolerant network is an NP-hard problem. To overcome this computational complexity in a practical setting, we propose to determine near-optimal motions of agents using *multi-objective control methods* from robotics [7], [20].

In robotics, multi-objective control is used to generate complex behavior by composing several simple behaviors. These simple behaviors are represented and generated by *controllers* [15]. To apply multi-objective control to generate the motion of robotic agents in disruption-tolerant networks, we have to define controllers that generate motion to optimize network performance metrics. We then have to *compose* these simple controllers to achieve agent motion that ensures the satisfaction of multiple performance metrics.

A controller represents behavior using a *control function*. The domain of this function is a subset of all allowable states of the system. This subset includes the desired *goal state*, which represents the unique minimum of the control function. If the system is within the domain of the control function, the controller can follow gradient of this function to reach the goal state, since it represents its only minimum. The control function thus implicitly defines goal-directed behavior for a subset of the allowable state space. The control function can also be viewed as an error function. By descending the gradient of the control function, the controller continuously reduces the error until the goal state with zero error is reached. As long as the system remains within the domain of the appropriate control function, the associated controller will exhibit robust, goal-directed behavior. Control theory [15] offers a rich theoretical foundation for this method of behavior generation.

The appeal of controllers lies in their simplicity. For a simple behavior it is generally not difficult to specify and implement an appropriate control function. To generate more complex behavior, it would be possible to design more complex control functions. However, this approach quickly leads to instabilities in the overall system behavior. Instead, researchers in robotics have developed methods of composing simple controllers to generate more complex behavior. The two most commonly used methods rely on the *subsumption architecture* [3] and on *nullspaces* [20], a fundamental concept in linear algebra [22].

1) Subsumption: The subsumption architecture [3] can be applied in a broader context, but we restrict our discussion to controllers. The subsumption architecture proposes a layered approach to generating complex behavior. For the purpose of our discussion, each of these layers contains one or multiple controllers. Each controller in the subsumption architecture has access to state information and can issue commands that affect that state. This command is specified by the control function based on the current state information. Higher-level

controllers can access the state of lower-level controllers and can overwrite their commands. The process of overwriting a lower-level command is referred to as *subsumption*. The subsumption architecture allows the design and implementation of complex behavior in an incremental fashion, starting with the lowest level and the most basic behavior. It is noteworthy, however, that at any point in time a single controller, namely the highest-level controller, will determine a particular command to affect the system's state.

2) Nullspaces: The composition of controllers based on nullspace projections [7], [20], [35], [36] permits the concurrent execution of multiple controllers—and thus the attainment of multiple, non-conflicting objectives. To achieve this, controllers are ordered hierarchically. Within this hierarchy, nullspace projections ensure that the effect of a lower-level controller cannot interfere with the effect of a higher-level controller. To understand how this can be achieved, we begin by reviewing the notion of nullspace.

The nullspace of a linear mapping A consists of all vectors x such that $Ax = 0$. Here, the nullspace of a controller is considered to be the collection of control commands that, when performed *in addition* to the controller, do not affect its performance. An example: Imagine an object located as a point $p = (x, y)$ in the plane. A controller ϕ_1 is changing the state of the object to achieve $x = 0$. This is accomplished using a control function that has a unique minimum along the x-axis. A second controller ϕ_2 has the objective of achieving $y = 0$ with a similar control function. If ϕ_2 only causes motion of the object orthogonal to the x-axis, it will not affect the degree to which ϕ_1 has achieved its objective—it will not interfere with ϕ_1. We say that ϕ_2 operates in the nullspace of ϕ_1. The spaces within which the two controllers act are orthogonal to each other. Irrespective of the commands issued by ϕ_2, by projecting this command into the nullspace of ϕ_1, we can guarantee that all motions caused by ϕ_2 will be orthogonal to the x-axis.

Using this notion of nullspace projections, multi-objective control is obtained by arranging the controllers into a hierarchy and projecting the behaviors of a lower-level controller into the nullspace of a higher-level controller. At each level of the hierarchy the controller optimizes its actions within the nullspace of higher controllers. Since this optimization takes place in the nullspace of the higher controller, the choice of optimal action at a lower level is guaranteed not to affect the optimality of an action chosen earlier by the higher controller. This is in contrast to the subsumption approach, which achieves coordination through turning individual controllers on and off in a manner specified by the system designer.

Nullspace composition has been applied successfully in a variety of tasks [7], [20], including by Sweeney et al. [35], who used it to maintain network connectivity for distributed agents. In the work, agents maintain line of site (necessary for infrared communication) while pursuing the exploration of an unknown environment.

III. Autonomous Agents in DTNs

In this section, we describe how autonomous agents can be deployed in disruption-tolerant networks to increase network performance. This is accomplished by adapting the agents' motion to the demand in the network (Figure 1). We first show that determining optimal motions for agents is NP-hard, providing the justification for the approximation approach presented here. Then, we define methods to optimize particular network metrics. Subsequently, we define a multi-objective controller that coordinates the individual methods.

In our model, the network is composed of *peers*, which are mobile nodes that are the sources and destinations of packets. Peers carry computational resources including a wireless radio. The movement of peers is dictated by a model described below. We introduce *agents* into the network, which are systems that make online, autonomous decisions which are intended to enhance performance in a system based on a stated algorithm and the current available input. Our agents carry the same type of resources as peers. They are intermediaries on the paths from source to destination, but are never the source or destination of packets. Agents are nomadic and are able to navigate to selected location. We present the details of this model below.

A. Complexity of Scheduling Agent Movement

The problem of determining optimal motions for agents in a DTN is NP-hard. We show this by reducing the *dial-a-ride problem* [29] to our DTN agent motion problem. The dial-a-ride problem consists of dispatching a vehicle to service a request for an item to be transfered from one location to another. That problem is a generalization of the traveling salesman problem [9], and it is known to be NP-hard.

The reduction of some instance of the dial-a-ride problem to agents servicing a DTN is as follows. First, note that the graph representing the physical or geographical environment of a DTN is the same as in an instance of the dial-a-ride problem. We assume that at each peer in the graph there is a participant in the network, that each participant is far enough away from any other participant that no point-to-point communication is possible, and that each participant in the network is static.

Every request made to the dial-a-ride system for transport from a location A to a location B is exactly a message in the DTN sent from a peer statically located at location A to a peer statically located at location B. Since all of the participants in the network are static and incapable of communicating, the transport of the message from A to B must be accomplished by the agent. By optimizing the routing of messages by the agent we also obtain an optimal solution to the dial-a-ride problem.

Since the dial-a-ride problem is NP-hard and reducible to the problem of routing agents to assist DTN routing, the routing of agents is NP-hard as well. A problem closely related to the DTN agent motion problem has been shown to be NP-hard by Zhao et al. [42].

B. Performance Metrics

Our aim in deploying autonomous agents in a DTN is to improve a variety of network performance metrics. These metrics include the following:

- *Bandwidth:* This metric captures the total number of messages in the network at a given point in time. We want to schedule the motion of agents so as to increase the bandwidth of the network, ensuring that available space for the transport of messages is used effectively. To achieve this, agents have to consider their travel time for package delivery.
- *Unique Bandwidth:* This metric refers to the total number of *unique* messages that are currently active in the network (multiple copies of a message may exist in the network). Again, the desired motion of agents increases this metric so that available bandwidth is used most effectively to respond to the participants' transmissions. To maximize this metric, agents should prefer to transmit messages not already in transit.
- *Message Latency:* Latency captures the average amount of time that it takes for a message to be delivered. Message latency can be reduced by biasing the motion of agents towards peers which are sending or receiving many messages.
- *Peer Latency:* This metric refers to the average time since a peer was last visited by an agent. To prevent starvation, it is important that all of the participants in the network be visited intermittently.

C. Distributed Network State Maintenance

The evaluation of performance metrics described in Section III-B requires global state information. In a real network, this global information is unavailable. Each peer in the network has perfect information about its own state, but must estimate the network's overall state. Each agent accomplishes this by constructing an approximate model of the network from information obtained from other network participants encountered during its motion. To construct this model, we assume that all participants have (loosely) synchronized clocks.

Each agent maintains information about every participant in the network. This information is tagged with a global time, synchronized between all participants. When two agents in the network meet, they exchange packages according to the employed routing protocol (in Section IV we describe and compare the performance of four different routing protocols). Agents also update their state information about all network participants, provided the encountered agent has more recent information available. This information includes a list of carried packages for each participant and the GPS location and time stamp of the last encounter. This approximate global state of the network provides the agent with sufficient information to determine its motion in accordance to the desired performance metrics.

D. Movement Controllers for Performance Metrics

The motion of autonomous agents in a disruption-tolerant network should optimize the four metrics described in Section III-B. In Section III-A we showed that the corresponding optimization is NP-hard. In this section, we describe two methods for generating the agent's motion derived from multi-objective control in robotics. For each metric, we present a method of determining a motion that optimizes the metric in isolation. We refer to the algorithm that generates the agent's motion as a *controller*. The bandwidth controller directs the agent to act so as to maximize bandwidth, the latency controller acts to minimize latency, and so forth. We then show in the next section how these controllers can be combined to generate agent motion that optimizes all four of the performance metrics.

The details of the individual controllers for each of the performance metrics are as follows:

- *Total Bandwidth Controller ϕ_T:* Traveling to any peer A will increase the bandwidth of the network by the number of messages that the agent can obtain from peer A. The peer chosen by this controller is the peer that has the largest number of messages not yet seen by the agent, amortized by travel time. This choice is based on the information obtained through distributed state maintenance (see Section III-C).
- *Unique Bandwidth Controller ϕ_U:* The unique bandwidth controller chooses the peer that it believes to have the largest number of messages not present anywhere else in the network. This choice is made irrespective of the travel time to reach the peer.
- *Delivery Latency Controller ϕ_D:* The delivery latency controller chooses the peer whose average delivery time is the largest, according to the information available to the agent.
- *Peer Latency Controller ϕ_P:* To reduce the peer latency, this controller should choose the location n_i least recently visited by an agent. Since latency relates to time, we also would like to consider the travel time of the agent. Let Δt_{n_i} be the time elapsed since location n_i has been visited by an agent. The peer latency controller selects location $n = \operatorname{argmin}_{n_i} \Delta t_{n_i} + t_{n_i}$, where[1] t_{n_i} is the agent's travel time to location n_i. This resolves ties based on Δt_{n_i} in favor of locality. It ensures that traveling time to visit the least recently visited location does not actually cause an increase in the peer latency statistic.

Because these controller consider only a single performance metric, they may generate conflicting motions for the agents. For example, if there are two peers in spatial proximity that exchange a large number of messages, the total bandwidth controller ϕ_T would dispatch agents back and forth between the peers to maximize the bandwidth usage.

[1]Here, argmin represents the n_i that attains the minimum value of the given expression when all locations are considered.

However, this will prevent other peers from being visited, something that the peer latency controller ϕ_P is trying to ensure. In those cases, it will be necessary to coordinate the controllers to result in consistent and effective motion.

Independently of the coordination scheme, these controllers are responsible for selecting a peer as the new target for the agent. The agent then moves to the geographic location of the peer, as indicated by its global state information. Once it arrives at that location, it selects a new destination, regardless of whether the peer was encountered. While it is moving, the agent sends packets to all agents and peers that it comes in contact with. If, along the way, it makes contact with the peer it is trying to move to, it also selects a new destination.

E. Multi-Objective Control

In traditional wired networks, a network administrator balances a variety performance metrics through the specification of fixed network parameters. In the previous section, we introduced controllers that optimize specific performance metrics (or objectives) of a network by determining the motion of a robotic agent within that network. We now will propose to use multi-objective control methods from robotics [35], [40], [36] to balance multiple network performance metrics in disruption-tolerant networks that have been augmented with autonomous mobile agents.

The purpose of multi-objective control is to combine multiple controllers in such a way their individual objectives can be optimized concurrently. The simplest method of multi-objective control combines the output of the individual controllers to determine the overall behavior of the robotic agent. Such a combination can be achieved, for example, by adding the spatial motion vectors determined by each controller. For some applications in robotics, such as obstacle avoidance, the resulting motion will optimize multiple desired objectives (staying at a distance from multiple obstacles). In the context of disruption-tolerant networks, however, such a simple strategy would be meaningless: if the bandwidth controller identifies a particular network node as the next goal location but the latency controller identifies a different one, these two control decisions cannot be easily combined in a meaningful way.

We explore more sophisticated methods for multi-objective control, namely nullspace composition of controllers and the subsumption architecture. Within either of these frameworks, the network's performance can be optimized by selecting the parameter of each of the individual controllers and by determine an ordering of controllers. This ordering determines a hierarchy of importance, effectively prioritizing the performance metrics associated with the controllers.

1) Nullspace Composition: Informally, the *nullspace* of a particular controller ϕ_1 is defined as the set of actions that can be taken by the agent without affecting the performance of ϕ_1 (see Section II-B). For example, if n potential motions of an agent optimize the metric encoded by controller ϕ_1 equally well, these n motions represent the nullspace of ϕ_1. Choosing any of these n motions will result in an equally optimal overall behavior.

To optimize a second metric represented by controller ϕ_2, which we call the *subordinate* controller, one chooses among those n choices a motion that performs best with respect to the second metric. We say that ϕ_2 is optimized in the nullspace of ϕ_1. This permits an implicit ordering of metrics and corresponding controllers, in which the action taken by a subordinate controller never affects the performance of a superior one. The notion of nullspace together with an ordering provides a principled framework for the composition of the controllers introduced in Section III-D.

Motivated by the application of this framework to DTNs, we define the performance of controllers with respect to a threshold. This means, for example, that any motion of an agent achieving a specified minimum bandwidth requirement is said to perform equally well with respect to this metric. Such a definition of performance permits nullspaces of sufficient size to optimize multiple objectives.

An ordering of controllers captures the relative importance of the associated network metrics. The specification of such an ordering provides the network administrator with a simple way of specifying criteria for performance optimization. The ordering we use here is given by:

$$\phi_P \lhd \phi_D \lhd \phi_U \lhd \phi_T, \tag{1}$$

where $\phi_i \lhd \phi_j$ indicates that ϕ_j is more important than ϕ_i. We say that ϕ_i is optimized subject to ϕ_j, meaning that ϕ_i is optimized in the nullspace of ϕ_j, i.e., it is assured that ϕ_i only generates motions that do not allow the metric associated with ϕ_j to fall underneath its required threshold. (The notation $\phi_i \lhd \phi_j$ is taken from [7]).

The ordering of controllers given above was chosen based on the observation that the nullspace of equal total bandwidths is significantly larger than the nullspace for unique bandwidth, and so forth down the ordering. This means that the ordering offers more flexibility for controllers with lower priority.

It is important to note that the above ordering is not the only appropriate one. Future work may be warranted to explore the effects of different orderings either specified by administrators or learned automatically. Likewise, the choice of thresholds in the definition of nullspace is up to the end user. A network administrator can manipulate the performance of their network to suit its demands.

2) Subsumption: An alternative to the nullspace approach to multi-objective control is *subsumption* [3]. The difference is in how the controllers dominate one another; we still use the four controllers described in Section III-D, prioritized as described above. However, whereas in the nullspace approach subordinate controllers optimized the motion of an agent in the nullspace of superior controllers, in the subsumption approach the controller with the highest priority exclusively determines the motion of the agent until its performance threshold is achieved. Only when all superior controllers

achieve their performance metric, do subordinate controllers get a chance to optimize their performance. In the subsumption approach, the output of dominant controllers completely *subsumes* the output of the subordinate controllers.

Given the ordering shown in equation 1, this would mean that the total bandwidth controller ϕ_T determines the motion of the robotic agent until the desired threshold for total bandwidth is reached. Once this is the case, ϕ_T relinquishes control to the next controller with lower priority, in this case ϕ_U. Now ϕ_U determines the motion of the agent. If the minimum threshold for the unique bandwidth is satisfied in the network, ϕ_U passes control to ϕ_D. If however, during the execution of ϕ_U, the total bandwidth criterion is violated, the controller ϕ_T subsumes ϕ_U and takes control away. In other words, if the objective of a controller ϕ_j is satisfied, the controller passes control from right to left to controller phi_i, in accordance with the direction of $\lhd$ in $\phi_i \lhd \phi_j$. If the objective of controller ϕ_j is not satisfied, it takes control from all controllers ϕ_i for $i < j$.

3) Comparison of Nullspace Composition and Subsumption: In this section we provide some insight about the differences between nullspace composition-based multi-objective control and the subsumption-based approach. The main difference is in the method of combining multiple single-objective controllers to achieve multi-objective behavior. We will discuss how this difference can affect performance in the context of agent-augmented disruption-tolerant networks. For both approaches we assume that all performance metrics have been ordered according to their importance for the specific network configuration and overall desired performance objectives.

The main difference between nullspace composition and the subsumption-based approach is that the former executes multiple controllers concurrently while the latter only executes a single controller at a time. Multi-objective control uses nullspace projections to ensure that a subordinate controller does not affect the behavior of a superior controller. In practice, this means that the highest-priority objective will always be optimized to the best of the agent's abilities. This is a desirable property; the network administrator can pick the most important performance metric and the system will ensure it remains optimized. However, this puts subordinate objectives at a disadvantage. They can be optimized only in the nullspace of all objectives superior to them. This means that all subordinate controllers may be limited in their ability to optimize their own performance metrics. With each additional optimized performance metric, the nullspace could be reduced significantly. It is possible that the nullspace is empty for subordinate behaviors, leaving no possible action to further optimize the associated performance metric. This is the case when any action that could be taken by a subordinate controller would interfere with the optimization of a higher-priority objective. The nullspace based controller guarantees performance for the highest-priority performance metric but all other metrics are only optimized if the nullspace is sufficiently large.

In the subsumption approach, the highest-priority controller whose objective is not yet achieved assumes control, possibly interrupting a lower-priority controller. Once the highest-priority controller has reached its objective, a subordinate controller can take its turn to optimize its performance metric. However, in contrast to the nullspace-based approach, the subordinate controller does not take into account the objective of the superior controller; it is able to perform any motion it wants to optimize its objective, even if that motion interferes with the optimality of a superior performance metric. If the superior controller's performance goal is violated as a result of this action, the superior controller immediately re-assumes control.

Nullspace-based multi-objective control is suboptimal because it restricts subordinate controllers to operate in a reduced space of actions. The subsumption-based approach is suboptimal because the optimization of individual performance metrics is not coordinated carefully enough. By making these simplifications, both methods avoid solving the NP-complete optimization problem (see Section III-A). In spite of these simplifications, however, both methods are capable of producing reasonable approximations to the optimization problem in practice.

Due to the fact that the nullspace-based approach is choosing actions that optimize multiple performance metrics simultaneously, we expect its performance to be superior to that of the subsumption-based approach. In Section IV, we compare subsumption-based and nullspace composition-based coordination of the controllers to validate this hypothesis.

IV. Experimental Evaluation

In Figure 1, we classified routing algorithms for DTN according to the degree to which they exploit structure in the motion of network participants and according to the degree to which the participants adapt their movements to the network demand. We now present experimental evidence that when autonomous agents use MORA to adapt their movements, they are capable of enhancing network performance of many different DTN routing protocols. Using the analogy presented in Figure 1, MORA agents are like taxicabs added to a system of cars.

To evaluate our proposed algorithm MORA and the effect of introducing autonomous agents into a DTN, we ran a series of custom networking simulations. First, we evaluate the use of nullspace versus subsumption multi-objective controllers, and we find that nullspace controllers are moderately superior. With additional evaluations, we find that significant performance gains are possible using MORA, in terms of the message delivery rate and latency. We apply MORA to four routing protocols: random routing, the ME/DLE protocol [5], [4], the MaxProp protocol [4], and FIFO routing. We vary many scenario parameters, including the offered network load, node buffer size, the number of peers in the network,

and the size of each packet. Additionally, we show the accuracy of the controllers used by MORA agents.

A. Methodology

The success of DTN forwarding algorithms is tied to the movement pattern of peers. Traditionally, researchers have used the random waypoint model (RWM) in lieu of empirical models, though this is often criticized (see [21]). Such a movement model cannot be used for our evaluation of DTNs: if peers move randomly, then no peer is any better at delivering a message than any other. A successful routing algorithm exploits structured, distinguishable movements.

Our evaluations are based on traces of the movements of the UMass DieselNet DTN [4], which is a testbed of 30 buses in Amherst, MA that we have constructed. To take traces of movement, we installed GPS devices on each bus. The GPS devices had to be placed on the buses at an angle that prevented continuous reception of GPS signals (i.e., they lacked a complete view of the sky). Therefore, while the experiment was always able to record data transfers, the location of each bus was available only during certain periods of time. However, we constructed a simple trace generator using GPS data to reconstruct the typical movement and timing information of 9 bus routes that roam an area 8.24 km by 14.70 km (about 121 km^2). Each route is modeled on representative trace days that had accurate GPS data. An advantage of the trace generator is that we are able to vary the number of buses in the simulation[2].

The default values for each simulation are an unlimited buffer, an offered mean load of 36 pkts/hour, 10-Kbyte packets, and 9 buses, each on distinct routes. Peers generate messages with inter-arrival times from a uniform distribution. We allowed peers to have unlimited buffers because in our real testbed each bus carries a 40GB drive; this is more than sufficient given the actual transfer bandwidths we observed between peers. In our experiments where we did limit the buffer, the buffers at peers are stated in terms of the number of messages they could store, which we varied from 50 to 450. In all experiments, each peer had an unlimited buffer for storing messages for which it was the source. To determine the amount of data transfered at each opportunity, we multiplied the length of time peers are in radio range by the mean transfer rate reported in the trace data, which is 120 KB/s. Each simulation is run for 10 simulated hours.

In each experiment, our default scenario is a comparison of 9 buses running a specific protocol (MaxProp, ME/DLE, random, or FIFO) to 9 buses and 3 autonomous agents running MORA. Obviously, adding robotic agents increases the amount of buffer resources in the system. Therefore, in the interest of a fair comparison, in the simulations of MORA, 3 randomly chosen buses have no buffer space allocated for packets generated by others. In other words, the MORA protocol simulations do not introduce additional storage in the network. Note also that MORA agents never generate packets.

[2]Future traces of DieselNet will solve this GPS data collection problem.

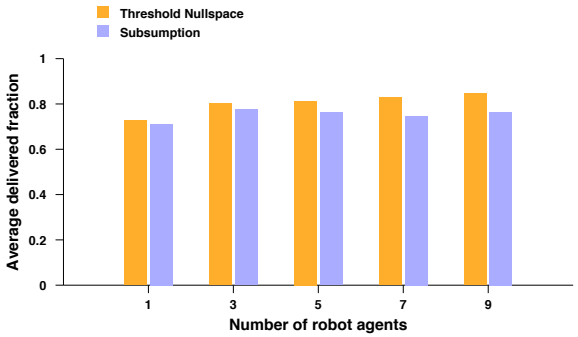


Fig. 3. Simulation experiments comparing the effect of subsumption vs threshold nullspace control on **packet delivery rate** as a function of offered load for ME/DLE. A statistical test shows the difference of the means are not significant for a 95% confidence interval.

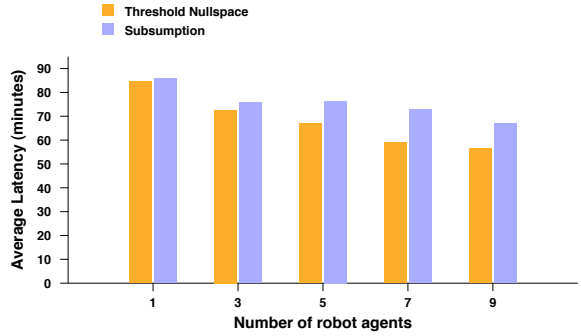


Fig. 4. Simulation experiments comparing the effect of subsumption vs threshold nullspace control on **delivery latency** as a function of offered load for ME/DLE. A paired t-test shows the means are statistically different for a 95% confidence interval for all cases except 1 agent.

Each point on the graph represents the average value of 10 simulations with 10 different random seeds (totaling thousands of packets). Error bars on graphs that report the fraction of delivered packets represent the semi-interquartile range (i.e., 25% and 75% of data). The SIQR numbers were too large to be useful for latency measurements—this is because the mean latency between each of the source-destination pairs is very different, and so measuring the variance of the latency is not illuminating or appropriate. To address this problem, for all latency graphs, we computed the *paired t-test* of robotic versus non-robotic versions of each protocol. Each simulation contains the same delay between new packets and the same movement patterns by buses. Therefore, to perform the test, we pair corresponding mean delivery times of each source-destination combination. The mean latencies reported in the graphs are the means of all source-destination means. Unless specifically denoted in the caption of each figure or in the text, the results always represent a significant difference between paired robotic and non-robotic protocols for a 95% confidence interval.

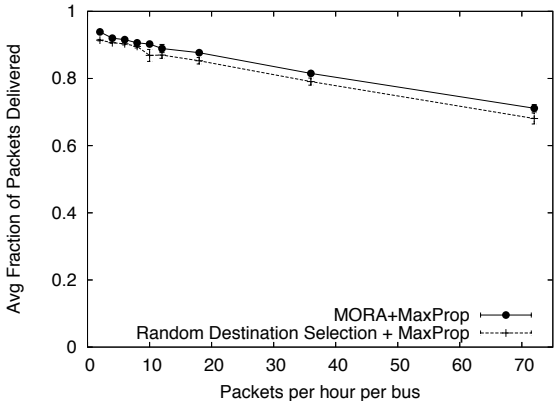


Fig. 5. **A comparison of MORA to movement using random selection of destinations** — Packet delivery rate with increasing load in the context of MaxProp and Random routing strategies.

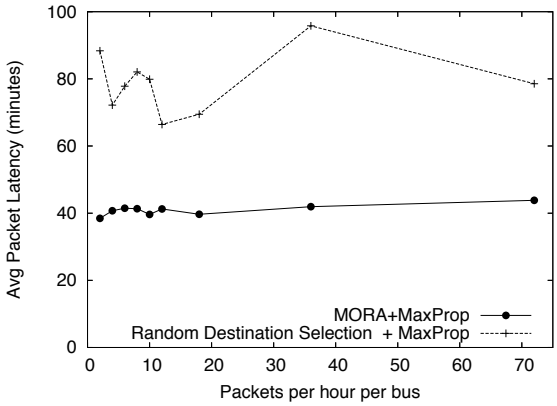


Fig. 6. **A comparison of MORA to movement using random selection of destinations** — Delivery latency with increasing load in the context of MaxProp and Random routing strategies.

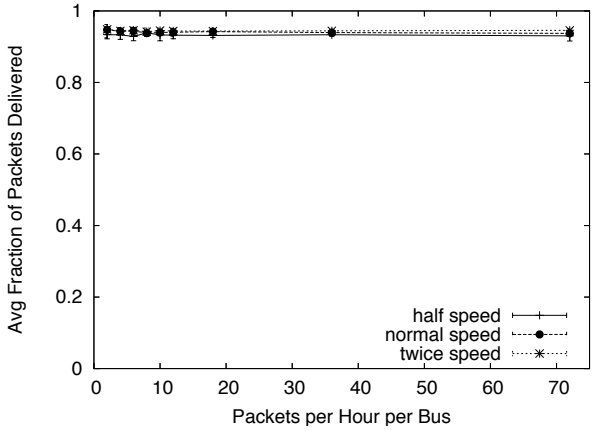


Fig. 7. Packet delivery rate with increasing load — **a comparison of MORA+MaxProp when agents move at varied speeds**.

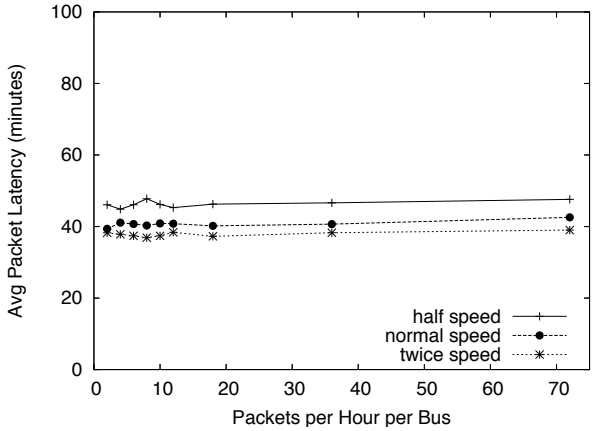


Fig. 8. Delivery latency with increasing load — **a comparison of MORA+MaxProp when agents move at varied speeds**. (Normal speed is statistically the same as twice when packets per hour per bus is 2.)

1) Protocols: Our design of MORA is independent of and oblivious to an underlying routing protocol used by non-robotic peers in the system. Therefore, we evaluate MORA over four different routing protocols, which we describe below.

The limited resource in our simulations is the bandwidth available between peers during transfer opportunities. At each opportunity, peers do not know the amount of data that can be transfered. These four protocols differ in how a peer A decides which packets to send—and what order to send them—to a peer B during a transfer opportunity.

- **Random** delivers packets destined for B first and then selects and schedules packets randomly from those stored at A.
- **MaxProp** [4] uses several mechanisms to provide routing. Packets are first assigned a score based on their destination and B; the score is based on the likelihood that B meets the destination directly or through a series of intermediaries. Packets with lowest scores are scheduled first; however, new packets are given a higher priority. Additionally, reports of successful packet delivery is epidemically flooded across the network to clear out buffers of intermediaries. Finally, packets maintain a list of previous hops to avoid transmitting to an intermediary that has already received the packet. Details of MaxProp are presented in our previous work; to our knowledge no proposed protocol performs better.
- **ME/DLE** is described in our previous work [4] as the Most Encountered / Drop Least Encountered protocol. ME/DLE is an extension of our previous work, the MV routing protocol (which appears in the preliminary version of this paper [5]). MV was designed for environments where buffer space is the limited resource and transfer opportunity bandwidth is unlimited. For devices with very small buffers compared to their radio resources, this makes sense. However, for the bus network we have constructed, MV is not appropriate. While MV's only mechanism is to drop packets with destinations that were least encountered, ME/DLE adds a mechanism that prioritizes during transfer opportunities the packets with destinations that are most encountered.

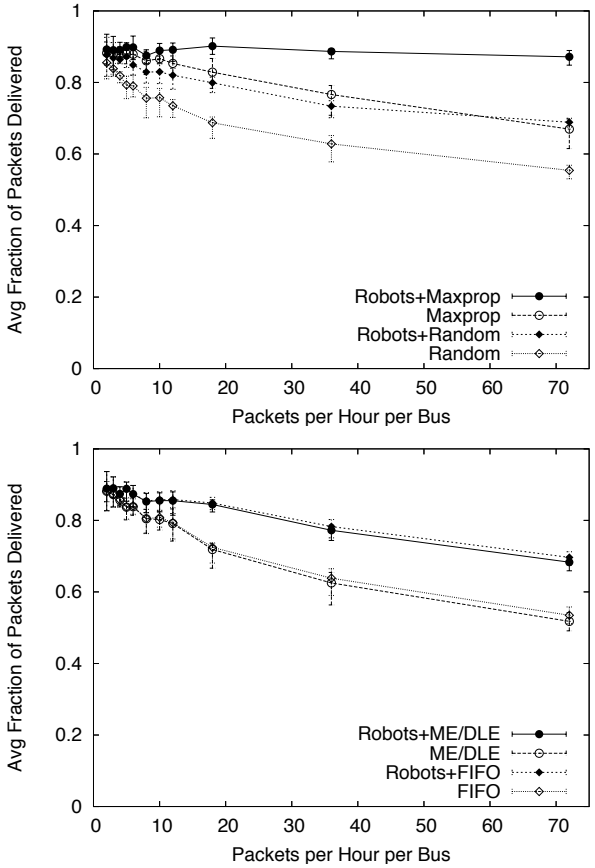


Fig. 9. Packet delivery rate as **network load** increases.

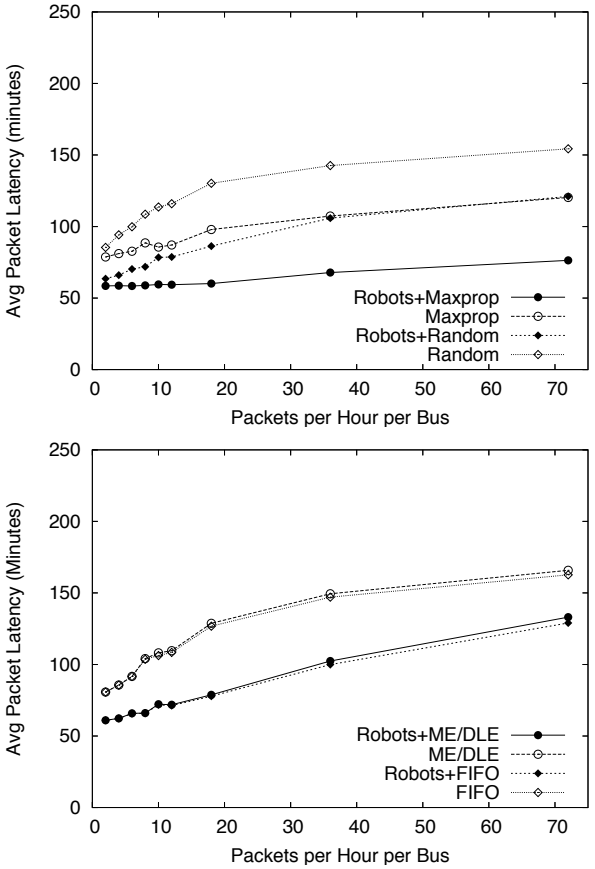


Fig. 10. Packet latency as **network load** increases.

In particular, ME/DLE uses the scoring mechanism from MaxProp (i.e., it is exactly MaxProp without the additional mechanisms).

- **FIFO** is a simple strategy where the packets are ordered for transmission based upon the FIFO order at which they were received at the bus node. Packets are also deleted in FIFO order when necessary. When buffers are large, packets leave the queue only when they are successfully delivered to their destination. Because no acknowledgments are reported, nodes can learn only directly from peers which packets have already been delivered by others.

B. Comparing Subsumption and Nullspace

To determine the appropriate multi-objective controller to use for autonomous agents in a DTN, we compared the performance of the two algorithms proposed in Section III-E by simulation.

Figures 3 and 4 compare the performance of subsumption to threshold-nullspace for varying numbers of robots with a fixed network load of 18 packets per bus per hour using the ME/DLE protocol. These experiments show that threshold nullspace control is better at merging the various control objectives of the primitive controllers. While the average delivery fraction for threshold nullspace is better (Figure 3), the confidence interval of the differences (not shown in the graphs) indicates that the means are in fact not statistically different. However, as Figure 4 shows, the mean latency of threshold nullspace is lower and statistically significant.

Since it achieves lower latency, for all subsequent experiments we choose nullspace-based multi-objective control for the autonomous agents. Below, we describe how the introduction of agents affects networks performance.

To evaluate the quality of agents' chosen movement destinations, we compared MORA against agents that move to randomly selected destinations. Note that the agents do not practice brownian motion or the random waypoint model. The peer they chose to move to is randomly selected (c.f., the description of controller in Section III-D). This requires some resources but is still trivial to implement and is more fair than random waypoint motion. Figure 5 shows the packet delivery rates for these two strategies when using MaxProp and random routing and three agents. The delivery rates are unchanged, but this is expected as the randomly moving agents are bound to meet some bus and that is always helpful. Since buses never leave the simulation, as long as the agents are meeting buses, packets will get delivered. Figure 6 shows the improvements due to MORA over randomly moving agents are in terms of delivery latency. MORA decreases

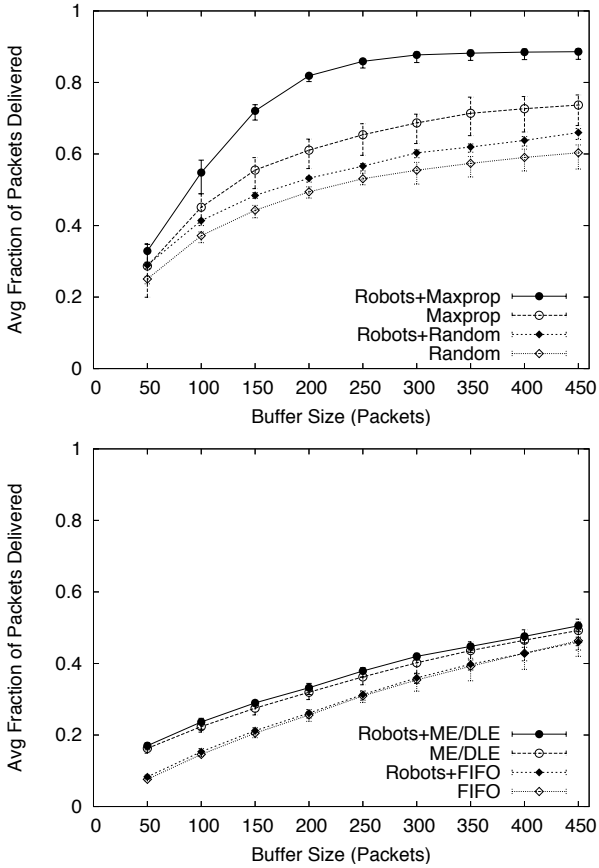


Fig. 11. Packet delivery rate as local **buffer sizes** increase.

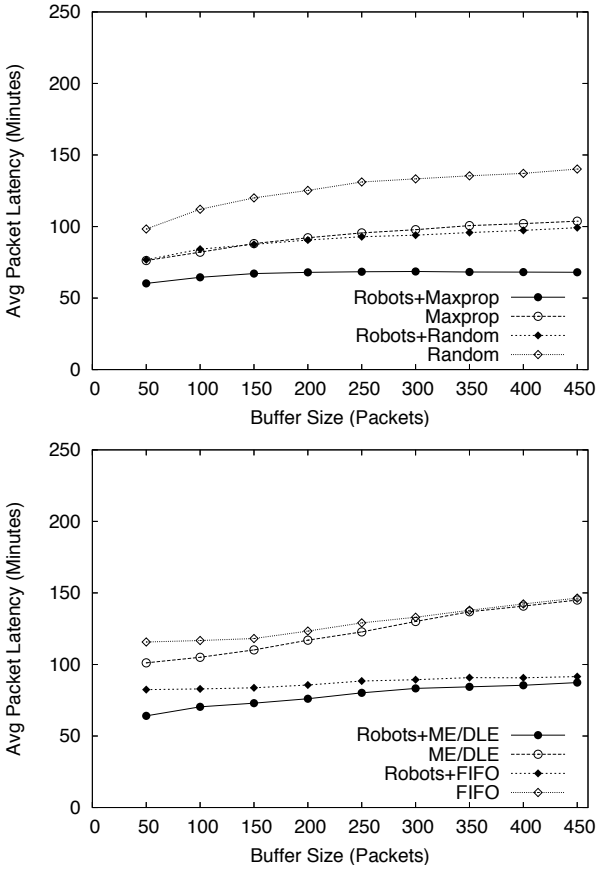


Fig. 12. Packet latency as local **buffer sizes** increase.

packet delivery latency by about half and it is less varying with increasing load. We can conclude that our algorithm is making decisions that benefit performance.

We also examined how agent movement speed affected MORA performance. Speeds half as fast and twice as fast did not affect delivery rate, as shown in Figure 7. However, as might be expected, in our experiments, delivery latency is affected by agent speed, but only slightly, as shown in Figure 8.

C. DTN Performance with Agents

In our next set of experiments, we evaluate MORA under varied network load, buffer space, and the number nodes in the system. For each, we show the resulting packet delivery rate and packet delivery latency. In general, we see that MORA always improves the fraction of packets delivered and consistently decreases average delivery latency.

Note that in Figures 9 through 16, delivery rate graphs appear in the left column and delivery latency graphs appear in the right column; MaxProp and Random are always in the top row, and ME/DLE and FIFO are always in the bottom row; The original protocols use unfilled symbols (e.g.,"○"), and robot-enhanced protocols use filled symbols (e.g., "●").

Varying Network Load. Figure 9 compare the delivery rates of the four protocols with and without MORA as the offered load increases in the network. Figure 10 shows the corresponding delivery latency. In all four protocols, the use of MORA robots significantly increases mean delivery rate. For the highest loads, we see that MORA provides an increase from 67% to 87% for MaxProp, from 55% to 69% for Random, from 52% to 68% for ME/DLE, and from 53% to 70% for FIFO. In all cases, we also see that MORA decreases average latency in all cases, and this difference increases slightly as load increases.

Note that MORA adds stability to MaxProp, maintaining a delivery rate of about 90% consistently as load increases (Figure 9-top). Interestingly, ME/DLE and FIFO perform no better than random (and therefore are very poor designs for DTNs), but all four protocol protocols benefit from the use of MORA robots.

Varying Storage. Figures 11 and 12 show the delivery rate and latency, respectively, for experiments where we varied a limited amount of buffer space carried by each bus and agent. In these simulations, the load was fixed at 36 pkts/hour per bus. (A comparison of Figures 11 and 9 shows that MaxProp nodes require buffers of only about 450 packets.)

In all cases, MORA-versions of all protocols have delivery rates that are no worse than without MORA, and we see the strongest improvement with MaxProp. For all sized buffers, MORA improves packet delivery latency.

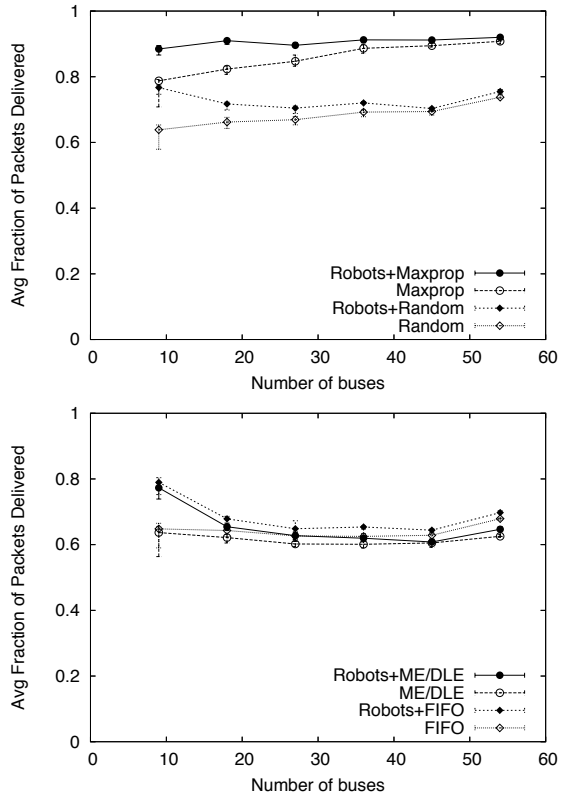


Fig. 13. Packet delivery rate with an increasing **numbers of buses**.

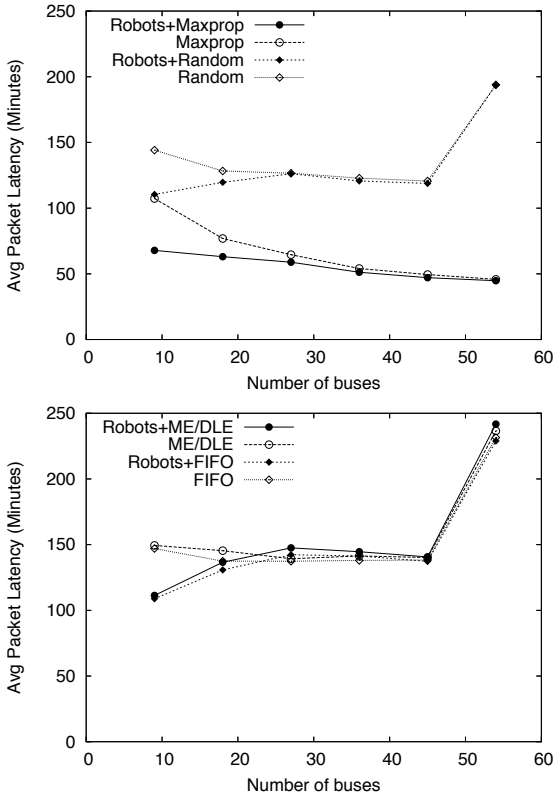


Fig. 14. Packet latency with an increasing **numbers of buses** in the network. (Means lack statistical difference after $x = 27$.)

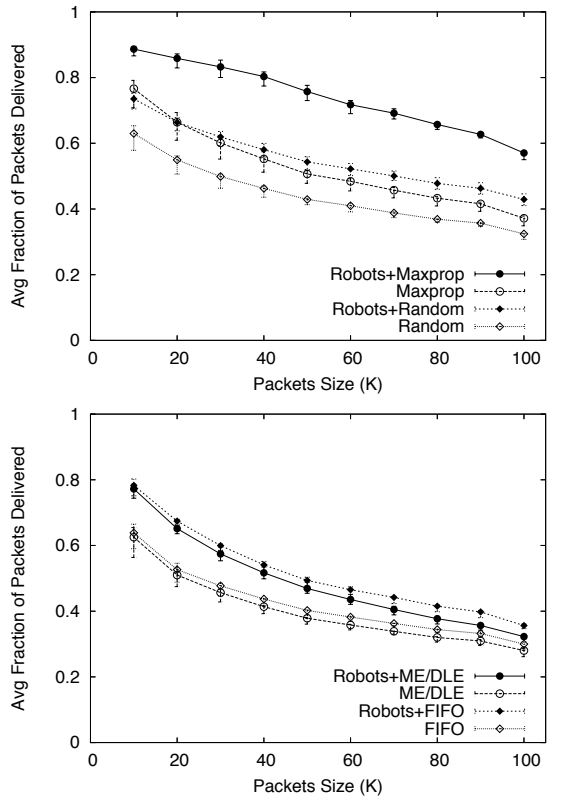


Fig. 15. Packet delivery rate as **packet size** increases.

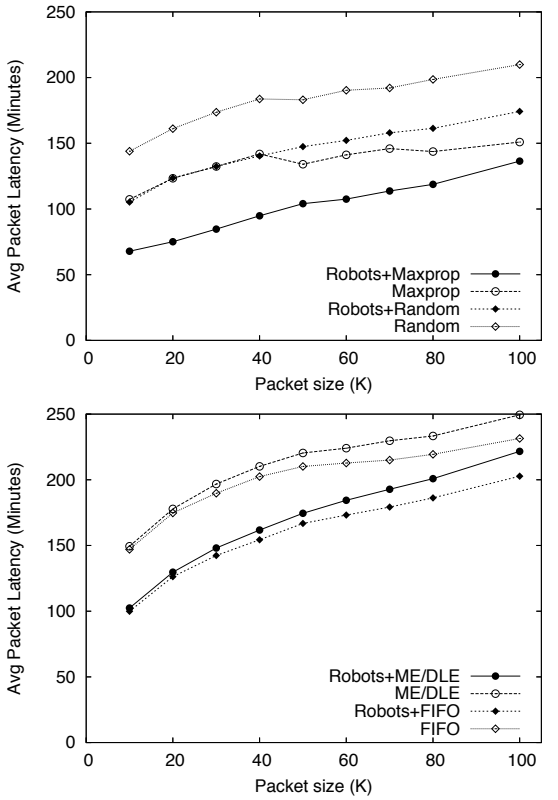


Fig. 16. Packet latency as **packet size** increases.

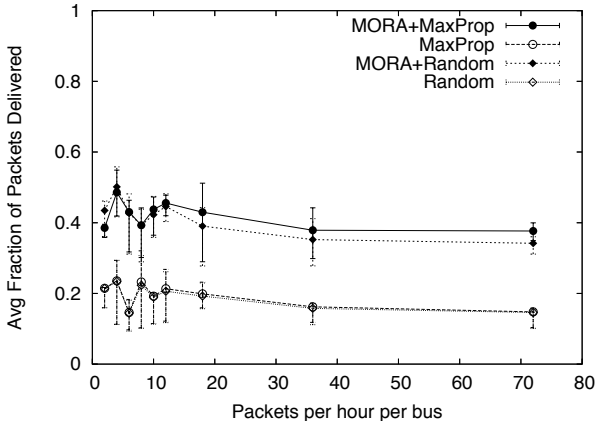


Fig. 17. **Synthetic mobility model** — delivery rate as packet load increases.

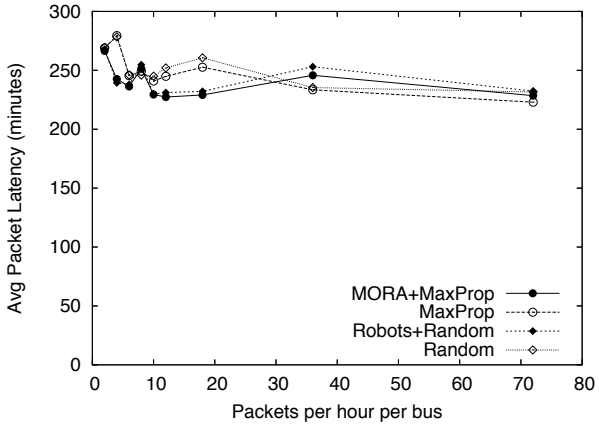


Fig. 18. **Synthetic mobility model** — latency as packet load increases. (Few points are statistically different for either protocol.)

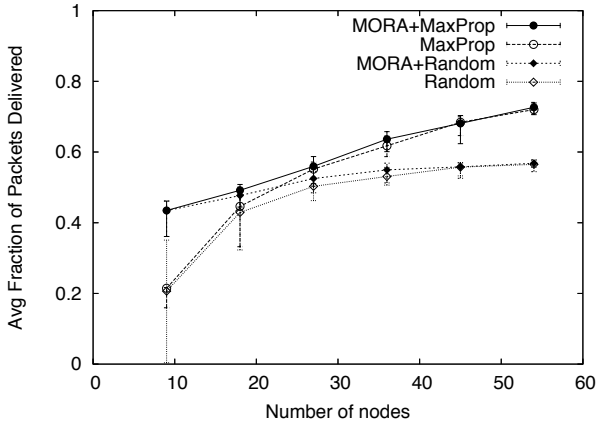


Fig. 19. **Synthetic mobility model** — delivery rate as the number of nodes increases.

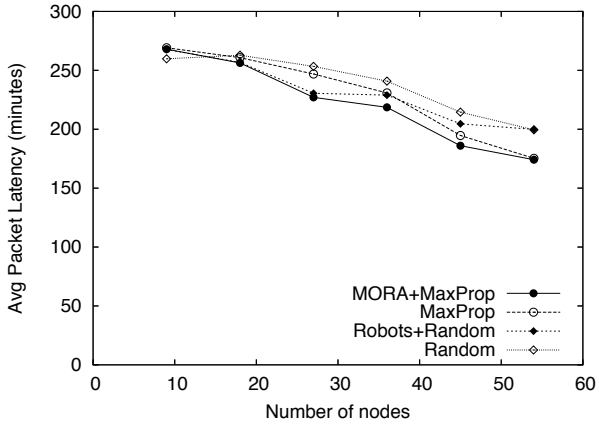


Fig. 20. **Synthetic mobility model** — latency as the number of nodes increases. (Both MaxProp and Random are not statistically different from MORA versions at $x = \{9, 18, 54\}$.)

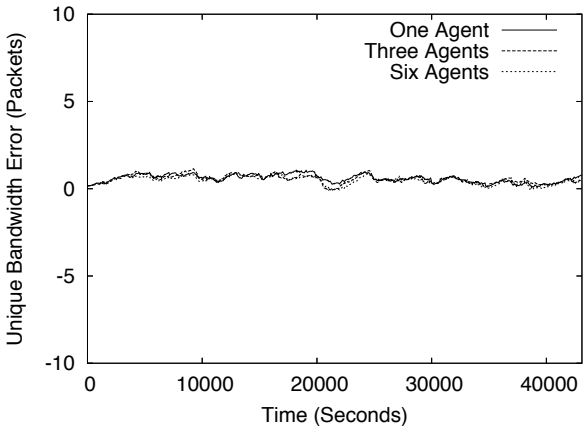
The best performing protocol is MORA-MaxProp, achieving delivery of 88% of packets with a buffer for only 300 packets and also achieving the lowest latency. And while MORA does not improve the delivery rate of ME/DLE and FIFO, it does improve delivery latency for those protocols. We can infer that MORA in general requires sufficient buffer in order to most improve performance, however, when MORA is coupled with better performing routing protocols, its improvements are pronounced even when storage buffers are limited.

Varying Nodes. Figures 13 and 14 compare increasing numbers of nodes on each bus route. Recall that we simulated nine separate routes each with 1 bus; in the figures, we increase the number of buses per route, with the total number in the system shown on the x-axis. The number of agents is fixed at 3 for all cases. As the network increases in density, and the robot agents comprise a smaller fraction of the population, we see diminished improvement between MORA and the original protocols. We can conclude that MORA agents must be a reasonable portion of the node population if significant improvements are desired. It is difficult for us to state conclusively from these experiments what percentage is appropriate for an arbitrary network.

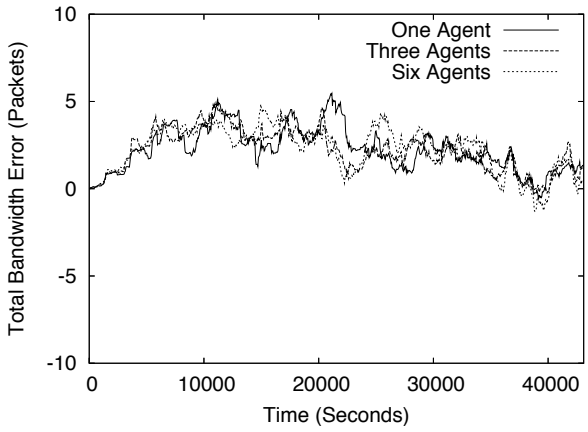
As the networks increase in size, the load also correspondingly increases. For network greater than 45 buses, the load is too high for network capacity, and the latency spikes for Random, ME/DLE, and FIFO. MaxProp and MaxProp+MORA do not suffer this congestion collapse since the protocol clears out delivered packets and priorities data for closer destinations, lowering latency for delivered packets.

Varying Packet Size. Figures 15 and 16 compare performance as the size of the packets increase. This effectively decreases the bandwidth available during each transfer opportunity. MORA-MaxProp again performs best in terms of the percentage of delivered packets as well as latency, though all see improvement. This experiment shows that MORA's improvements are present for a variety of application-required packet sizes in the DTN, the improvement drops as resources become scarcer.

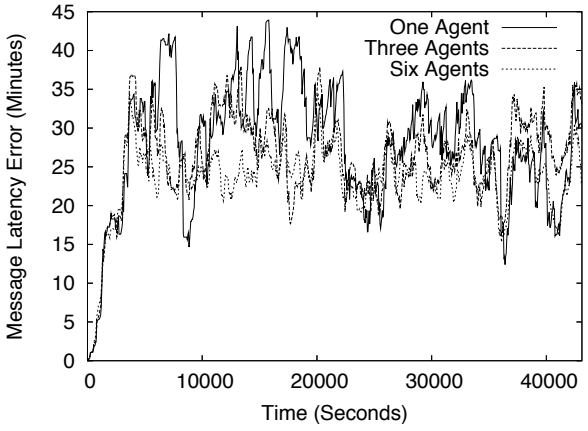
Evaluation Against a Synthetic Mobility Model. In our final set of comparisons, we show the results of using MORA in a purely synthetic mobility model that we first introduced



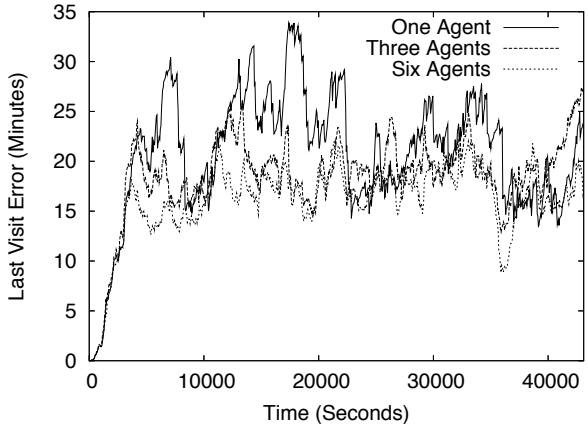
(a) Bandwidth Error



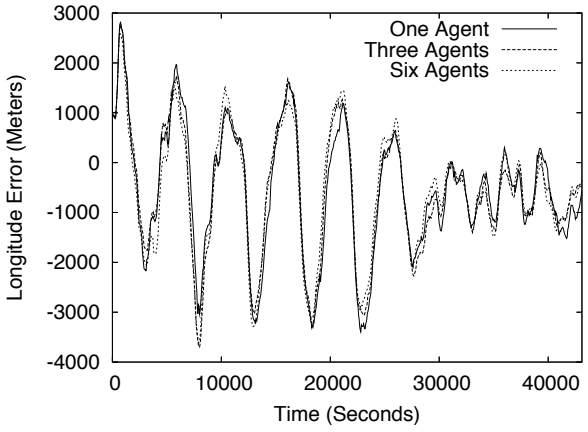
(b) Bandwidth Error



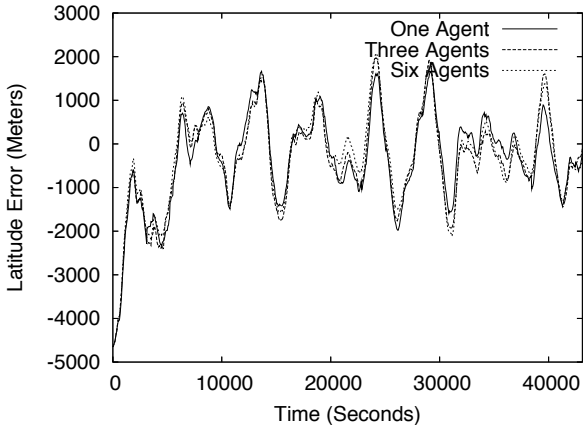
(c) Latency Error



(d) Last Visited Error



(e) Agent Longitude Error



(f) Agent Latitude Error

Fig. 21. Accuracy of the distributed status information over time.

in our previous work [5]. In the model, peers move periodically between three randomly chosen geographic locations: a home location and two remote locations. Peers move only among the three points, with the home location being chosen 50% of the time, and the remote points visited 25% each.

We evaluate the performance of MaxProp and Random, with and without MORA, as load increases and as the number of nodes simulated grows (with the number of agents held at 3). In Figure 17 we see that MORA doubles the delivery rate, and we see in Figure 18 there is little improvement in

latency. Similarly, MORA loses it's advantage as the number of nodes in the network increases, as shown in Figures 19, and it has only a small improvement in latency as shown in Figure 20. In this synthetic mobility model, the network is closer to the movements of the random waypoint model, and therefore, there is little structure for MORA to take advantage of. Although nodes move to distinct locations, the locations themselves are even distributed throughout the geography. Most of the advantages of MORA are likely from the increased number of connection opportunities it offers rather than intelligent delivery. Therefore, we conclude that MORA has the most to offer in scenarios where movement is structured and not random.

D. Evaluation of Distributed Network Statistics

The control strategies are dependent upon a quality estimate of the state of the network. In a DTN, the quality of the estimate is affected greatly by the disruptive nature of the network since peers find it hard to exchange information. To better understand this, we monitor the percentage error of each of the network statistics over time. Additionally we measure the accuracy of an agent's estimation of where a peer is geographically and the agent's actually location; longitude and latitude are displayed separately. The graphs of these results are shown in Figure 21.

The results shown are for 3 buses per route (27 total) and 24 simulated hours of continuous operation. The accuracy of each agent is recorded every 50 seconds. The error reported is measured relative to the ground truth in the simulation. The plots represent the averages of 10 independent simulations with different random seeds. We show results for simulations with one, three and six agents in the system. We use the same default parameters described earlier.

These results show that the agents are able to consistently estimate the unique bandwidth in the system, and they have a relatively accurate view of the total bandwidth. We believe this is because bandwidth and unique bandwidth is discrete and changes more slowly, so the approximation is more consistent. The agents are constantly moving and location is continuous, therefore there is much greater error. In other words, bandwidth only ever changes when peers meet, and when peers meet they have an immediate opportunity to update the distributed statistics. On the other hand, location is constantly changing and that information doesn't propagate out. From the perspective of the distributed statistics, peers jump from one location to the next. Even so, each agent's estimation of the location of a peer that it is moving towards stabilizes as the simulation progress, but always displays a small amount of error compared to the size of the simulated movement of 8.24 km by 14.70 km. These numbers could be improved if MORA nodes used dead reckoning to estimate where nodes will be in the future, rather than moving to their last known location. The most problematic estimators are message latency and last visit error. However, as these have the least priority in our nullspace controller, we do not believe the effect is pronounced. The estimates increase in accuracy with the number of agents in the system since they are able to pass information to each other. We expect that if nodes in the system were modified to pass such information to agents, the accuracy would be even greater—however, we did not evaluate this scenarios as our goal for MORA is to show its improvement without modifying other nodes in the network. This is enforced by the notion that MORA performs quite well in the MaxProp and Random cases.

V. Conclusions

Disruption-tolerant networks require routing algorithms that are different from those designed for ad hoc networks. The capacity of a DTN is provided solely by the motion of its participants. For a routing algorithm to ensure performance under such conditions, it has to explicitly account for this motion in its strategy of forwarding messages. In this paper, we introduce a classification of routing algorithms for DTNs based on this observation. We differentiate routing protocols based on the degree to which they exploit structure in the movement patterns of network participants to improve performance metrics. Along a different dimension, we differentiate them based on the degree to which participants adapt their motions to network demand.

The exploitation of structure in the network participants' movement patterns improves performance in DTNs. We introduce the routing protocol MORA, which maintains a movement model of the network participants and uses this information to perform routing of messages on the network. It estimates the probability of a particular message being delivered by a given peer, and thus is capable of making informed routing decisions. We present experimental evidence that routing messages in DTNs using MORA results in significant performance improvements over other techniques in achieving higher delivery rates and lower delivery latency. These improvements continue even as traffic on the network increases by an order of magnitude.

The adaptation of network participants' motion to network demand permits additional performance improvements for DTNs. For this purpose, we propose the introduction of autonomous agents into the DTN. By adapting their motion, these agents are able to compensate for a mismatch between available capacity and demand. We propose multi-objective control algorithms from robotics to control the motion of autonomous agents in order to optimize network performance metrics. These algorithms permit for a simple prioritization among network metrics by network administrators. Experimental results demonstrate that multi-objective control methods are successful at improving network performance by adapting the movements of autonomous agents introduced into a DTN.

We have shown that employing a routing strategy based on multi-objective control for autonomous agents in a DTN has the most significant performance improvements. This indicates that it is desirable for routing protocols in DTN to exploit the structure present in movement patterns of network participants to route messages as well as to change

the movement patterns of participants in accordance with network demands.

REFERENCES

[1] I. Akyildiz, D. Pompili, and T. Melodia. Underwater acoustic sensor networks: Research challenges. *Elsevier Ad hoc Networks Journal*, March 2005.

[2] N. Banerjee, M. D. Corner, and B. N. Levine. An Energy-Efficient Architecture for DTN Throwboxes. In *Proc. of IEEE Infocom*, May 2007.

[3] R. A. Brooks. A robust layered control system for a mobile robot. *International Journal of Robotics and Automation*, RA-2(1):14–23, 1986.

[4] J. Burgess, B. Gallagher, D. Jensen, and B. Levine. MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks. In *Proc. IEEE INFOCOM*, April 2006.

[5] B. Burns, O. Brock, and B. Levine. *MV* routing and capacity building in disruption tolerant networks. In *Proc. IEEE INFOCOM*, pages 398–408, March 2005.

[6] B. Burns, O. Brock, and B. Levine. Autonomous Enhancement of Disruption Tolerant Networks. In *IEEE International Conference on Robotics and Automation*, May 2006.

[7] J. Coelho and R. Grupen. A control basis for learning multi-fingered grasps. *Journal of Robotic Systems*, 14(7):545–577, 1997.

[8] Georgia Tech Aerial Robotics. Available at: http://controls.ae.gatech.edu/gtar/.

[9] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.

[10] J. Davis, A. Fagg, and B. Levine. Wearable Computers and Packet Transport Mechanisms in Highly Partitioned Ad hoc Networks. In *Proc. Intl. Symposium on Wearable Computers*, October 2001.

[11] BAA04-13: Disruption Tolerant Networking (DTN), May 2004. Technical POC: Preston Marshall, DARPA/ATO.

[12] Delay-tolerant networking research group. http://dtnrg.org.

[13] M. Dunbabin, P. Corke, I. Vailescu, and D. Rus. Data Muling over Underwater Wireless Sensor Networks using an Autonomous Underwater vehicle. In *Proc. Intl Conf on Robotics and Automation (ICRA)*. IEEE, May 2006.

[14] M. Dunbabin, J. Roberts, K. Usher, G. Winstanley, and P. Corke. A Hybrid AUV Design for Shallow Water Reef Navigation. In *Proc. Intl Conf on Robotics and Automation (ICRA)*. IEEE, April 2005.

[15] G. F. Franklin, J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Addison-Wesley, third edition, 1994.

[16] M. Grossglauser and M. Vetterli. Locating Nodes with EASE: Mobility Diffusion of Last Encounters in Ad hoc Networks. In *IEEE Infocom*, 2003.

[17] K. M. Hanna, B. Levine, and R. Manmatha. Mobile Distributed Information Retrieval For Highly-Partitioned Networks. In *IEEE ICNP*, Nov 2003.

[18] S. Jain, K. Fall, and R. Patra. Routing in a Delay-Tolerant Network. In *Proc. ACM Sigcomm*, August 2004.

[19] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[20] O. Khatib. A unified approach to motion and force control of robot manipulators: The operational space formulation. *International Journal of Robotics and Automation*, 3(1):43–53, 1987.

[21] D. Kotz, C. Newport, R. Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental Evaluation of Wireless Simulation Assumptions. In *Proc. ACM/IEEE Intl Symp on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pages 78–82, Oct 2004.

[22] D. C. Lay. *Linear Algebra and its Applications*. Addison Wesley, 2nd edition, 1997.

[23] Q. Li and D. Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 44–55, New York, NY, USA, 2000. ACM Press.

[24] A. Lindgren, A. Doria, and O. Schelén. Poster: Probabilistic routing in intermittently connected networks. In *Proceedings of The Fourth ACM International Symposium on Mobile Ad hoc Networking and Computing (MobiHoc 2003)*, Annapolis, MD, June 2003.

[25] M. Motani, V. Srinivasan, and P. Nuggehalli. Peoplenet: engineering a wireless virtual social network. In *ACM MOBICOM*, pages 243–257, 2005.

[26] J. Partan, J. Kurose, and B. N. Levine. A Survey of Practical Issues in Underwater Networks. In *Proc. ACM International Workshop on UnderWater Networks (WUWNet)*, pages 17–24, September 2006.

[27] C. E. Perkins and E. M. Royer. Ad hoc On-Demand Distance Vector Routing. In *IEEE Workshop on Mobile Compting Systems and Applications*, pages 90–100, February 1999.

[28] N. Sarafijanovic-Djukic and M. Grossglauser. Last Encounter Routing under Random Waypoint Mobility. In *Networking 2004, The Third IFIP-TC6 Networking Conference*, May 2004.

[29] M. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research*, 6(4):285–305, 1985.

[30] T. Small and Z. J. Haas. Resource and performance tradeoffs in delay-tolerant wireless networks. In *Proc. ACM Workshop on Delay-tolerant networking (WDTN)*, pages 260–267, 2005.

[31] J. Sorber, N. Banerjee, M. D. Corner, , and S. Rollins. Turducken: Hierarchical power management for mobile devices. In *Proc. Intl Conf on Mobile Systems, Applications, and Services (MobiSys),*, June 2005.

[32] J. Sorber, A. Kostadinov, M. Brennan, M. Corner, and E. Berger. eFlux: Simple Automatic Adaptation for Environmentally Powered Devices (poster/demo). In *IEEE workshop on Mobile Computing Systems and Applications (HotMobile/WMCSA)*, April 2006.

[33] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proc. ACM Workshop on Delay-tolerant networking (WDTN)*, pages 252–259, 2005.

[34] R. Stokey et al. Enabling Technologies for REMUS Docking: An Integral Component of an Autonomous Ocean-Sampling Network. *IEEE J. Oceanic Eng.*, 26(4):487–497, Oct. 2001.

[35] J. Sweeney, H. Li, R. A. Grupen, and K. Ramamritham. Scalability and Schedulability in Large, Coordinated, Distributed Robot Systems. In *Proc. Intl. Conf on Robotic Applications (ICRA)*, 2003.

[36] B. Thibodeau, S. W. Hart, D. R. Karuppiah, J. D. Sweeney, and O. Brock. A Cascaded Filter Approach to Multi-objective Control. In *Proceedings of the International Conference on Robotics and Automation*, pages 3877–3882, New Orleans, USA, April 2004.

[37] Y. Wang, S. Jain, M. Martonosi, and K. Fall. Erasure-coding based routing for opportunistic networks. In *Proc. ACM Workshop on Delay-tolerant networking (WDTN)*, pages 229–236, 2005.

[38] D. Webb, P. Simonetti, and C. Jones. SLOCUM: an underwater glider propelled by environmental energy. *IEEE J. Oceanic Eng.*, Oct. 2001.

[39] J. Widmer and J.-Y. L. Boudec. Network coding for efficient communication in extreme networks. In *Proc. ACM Workshop on Delay-tolerant networking (WDTN)*, pages 284–291, 2005.

[40] Y. Yang, O. Brock, and R. Grupen. Exploiting Redundancy to Implement Multi-Objective Behavior. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3385–3390, 2003.

[41] W. Zhao and M. Ammar. Message Ferrying: Proactive Routing in Highly-Partitioned Wireless Ad hoc Networks. In *IEEE Workshop on Future Trends in Distributed Computing Systems*, May 2003.

[42] W. Zhao, M. Ammar, and E. Zegura. A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad hoc Networks. In *Proc. ACM Mobihoc*, May 2004.

[43] W. Zhao, M. Ammar, and E. Zegura. Controlling the Mobility of Multiple Data Transport Ferries in a Delay-Tolerant Network. In *IEEE Infocom*, March 2005.

[44] W. Zhao, Y. Chen, M. Ammar, M. D. Corner, B. N. Levine, and E. Zegura. Capacity Enhancement using Throwboxes in DTNs. In *Proc. IEEE Intl Conf on Mobile Ad hoc and Sensor Systems (MASS)*, pages 31–40, Oct 2006.