2005

MV Routing and Capacity Building in Disruption Tolerant Networks

Brendan Burns
University of Massachusetts - Amherst

Oliver Brock
University of Massachusetts - Amherst

Brian Neil Levine
University of Massachusetts - Amherst

MV Routing and Capacity Building in Disruption Tolerant Networks

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

ABSTRACT

Disruption Tolerant Networks (DTNs) require routing algorithms that are different from those designed for ad hoc networks. In DTNs, transport of data through the network is achieved through the physical movement of the participants in the network. We address two fundamental problems of routing in DTNs: routing algorithms with robust delivery rates, and management of networks where demand for routes does not match with the movement of peers. For the first problem, we propose the *MV* algorithm, which is based on observed *meetings* between peers and *visits* of peers to geographic locations. We show that our approach can achieve robust delivery rates: 83% of the maximum possible delivery rate, as compared to 64% for fifo buffer management. The advantage remains significant as the offered load of the system is increased an order of magnitude. For the second problem, we propose to augment available routes and capacity in a DTN through autonomous agents (e.g., autonomous blimps or mobile robots). We propose a controller that moves the agent to where network needs are not being met by the movement of peers. Our controller is able to increase delivery between fifteen and twenty-five percent. Our experiments shows that the introduction of even a few agents can dramatically increase the reliability of the message ferrying network. Moreover, our techniques are compatible and offer a robust method of approaching the problems of DTNs.

Keywords: System design, Simulations, Routing, Wireless networks

I. INTRODUCTION

Many routing protocols exist to support end-to-end messaging in ad hoc wireless networks [11, 18]. Such protocols assume an end-to-end connection through a contemporaneous set of links via intermediary peers. As a result, if a path between two nodes in a network does not exist, communication is not possible, and the route creation process fails.

To adapt to situations where simultaneous links in the network are not practical or possible, a growing body of work is exploring techniques for moving network traffic over asynchronous paths. Such networks have varied names: *highly-partitioned networks* [6, 10], *message ferrying* [25, 24], *delay tolerant networks* [8], and *disruption tolerant networks* (DTNs) [7]. To enable routing end-to-end in a DTN (the term we choose for this article), peers are relied upon to store and carry messages of others. Whenever two members pass, they negotiate the exchange of messages, which are often called *bundles* in DTNs. There are many fundamental problems in designing a well-performing DTN. In this paper, we address two such problems.

First, we propose a protocol that controls *forwarding of bundles from a mobile source to a stationary destination*. We assume nodes have limited buffers for the storage of messages, and we assume they are able to determine their location in a cell-based geographic grid (e.g., using GPS). We assume nodes initially know nothing of the movement pattens of other nodes, and learn them only online through the mechanisms of the protocol. Our protocol keeps track of observed *meetings* between peers and *visits* to locations, and is called *MV*. It builds on our previous work [6], which kept track only of meetings between nodes. We compare our work to a first-in-first-out strategy for managing the buffer allocated for carrying the bundles of others. Our simulations show it performs significantly better, reaching 84% of maximum possible delivery rate, versus 64% for a first-in-first-out buffer management, and maintaining its significant advantage as the offered load in the system or the number of peers increases.

Second, we propose a method of *network design to address incongruence between the movement of peers and the flow of traffic*. The *bandwidth capacity* of a DTN is produced by the participants who move within it, while the *bandwidth requirements* are produced by the participants sourcing bundles. As in any network, when there is a mismatch between available capacity and demand the performance of the network suffers. In a traditional network such a mismatch would be resolved through additional network wiring or hardware. Unlike traditional networks, in a DTN which has no systematic infrastructure, additional capacity can only be added through the addition of participants in the network. These participants might be stationary nodes with large capacity located in strategic locations. However, due to the fluid nature of capacity in a DTN, we believe it is more effective to add additional mobile participants to the network whose movements can reactively increase capacity in areas of increased demand.

The second part of this paper is concerned with the design and evaluation of control algorithms that direct the motion of physical agents in the DTN to increase network performance metrics. The control algorithm provides a reactive, distributed monitor for the DTN. It provides information about the network's state and acts to improve regions where demand is greater than current capacity. The design of control strategies assumes the use of autonomous agents which can move to arbitrary locations in the physical environment of the DTN. In practice such agents might be ground-based mobile robots, such as the UMass Segway RMP [21] or airborne robots [4]. Some examples of these robots are shown in Figure 2.

We show that the task of choosing the appropriate path for an agent is NP complete; however, we propose that the approach of nullspace composition [13] taken from robotic control [3, 12, 16] can obtain a quality approximation of the optimal solution. Experimentally we found that the addition of these agents can improve bandwidth and latency metrics for the network by up to 20%.

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 forwarding has been studied by a growing number of researchers. Vahdat and Becker have proposed a flooding algorithm called *epidemic routing* [23] that assume infinite buffer. In 2000, we proposed routing algorithm for highly-partitioned networks by exploring a number of different strategies for deciding which bundles to exchange when two network participants meet [6]. Our algorithm, called *Drop-Least Encountered*, had peers keep track of the other peers they meet regularly over time. Peers initialize their likelihood of delivery a bundle to a moving peer as 0. When a

Figure 1: The UMass Segway Robotic Transporter [21]

Figure 2: The Georgia Tech Robotic Helicopter [4]

peer A meets another peer B, the former sets the likelihood of delivering bundles to B as 1. Then A takes a portion of B's likelihood of delivering bundles to the other nodes 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 were subsequently proposed by others [14, 19, 9], with each paper showing a different analysis of the problem.

Zhao, et al. [24, 25] have proposed DTN networking based on ferries, which are nodes that have completely predictable routes 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. That work is similar to the work we present here; the difference is that in Zhao, et al., peers adapt their own movements to a fixed-scheduled ferry. In contrast, we propose the ferries adapt their movements to the routing demand that is unmet by the movement of peers.

Similar to Zhao, et al. is work from the IRTF Delay Tolerant Network Research Group [8], where the focus again is on predictable, but non-contemporaneous routing[1].

[1]Also apparently related but was not available to the public is: S.

Rumor routing [1] is an approach to networking in sensor networks which avoids the costs of doing flood routing. In rumor routing each node passes a message on to each of its neighbors with a probability p. In this way a message is probabilistically insured to travel from source to destination without an explicit route. Rumor routing is focused on networks with stationary nodes, but the spread of bundles through the network is very similar to that achieved by DTN routing.

Finally, in our previous work, we showed that for some applications, disruption and partition can be survived without routing or forwarding. 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 peers carries only a small fraction of the full database [10]. 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

The controller we propose for agents in a DTN is derived from works in the field of multi-objective control [3, 12, 16]. In general the goal of multi-objective control is to coordinate a collection of controllers with individual goals to achieve a desired global behavior. It is generally straightforward to specify individual controllers that obtain local atomic goals which are pieces of a larger behavior. The job of the multi-objective controller is to find a coordinated composition of these individual controllers such that the globally desired behavior (in our case the improvement of the DTN) is obtained.

Numerous algorithms for multi-objective control have been proposed, here we discuss those which are directly related to our proposed controller.

In the subsumption architecture [2] each individual controller is a finite state machine with inputs and outputs which may be connected to other controllers or real world sensors/actuators. These controllers are ordered into a layered hierarchy. Multi-objective control and coordination is achieved by having higher controllers modify the inputs or inhibit the operation of lower level controllers.

The notion of the nullspace [13] from linear algebra has also been used to construct multi-objective controllers [3, 12, 16, 22]. The nullspace of a mapping A consists of all vectors x such that $Ax = 0$. Here, the nullspace of a controller C is considered to be the collection of control commands that, when performed *in addition* to the controller do not affect the performance of C. Using the nullspace, multi-objective control is obtained by arranging the controllers into a hierarchy, projecting the possible 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 does not affect the optimality of an action chosen earlier by the higher controller. This is the important distinction between nullspace composition and subsumption. Subsumption achieves coordination through turning individual controllers on and off in a manner designed by the system engineer. In contrast, nullspace composition relies on mathematics to provide a solution which allows all controllers to act simultaneously while coordinating their behavior to achieve the global goal.

Nullspace composition has been successfully used for a variety of tasks [3, 12, 16], including Sweeney et al. [22] who used it to maintain network connectivity for distributed agents. In the work, agents maintained line of site (necessary for infrared communication) while pursuing the exploration of an unknown environment.

III. MV FORWARDING ALGORITHM

In this section, we propose a new protocol for efficient bundle delivery in disruption tolerant networks. Our protocol, called *MV*, learns the frequency of *meetings* between nodes and which cells in a geographical grid are frequently *visited* by each node. The past frequencies are used to rank each bundle in a node's buffer according to the likelihood of delivering a bundle through a path of meetings and locations. In the next section, we evaluate the performance of *MV* and determine its efficiency and robustness.

We make three major assumptions about the type of network that *MV* supports.

1. Nodes have an infinite buffer for the bundles they originate, and only a fixed buffer size, n, for the bundles of others. We assume all nodes have the same fixed buffer size. This is the most realistic assumption regarding buffers, as people add storage for as much as their own needs require, but usually limit how much they donate to others.

2. When peers have an opportunity for transfer, they do so with a fully reliable, infinite bandwidth link layer. We are trying to isolate the better routing algorithm indepen-

Jain, K. Fall, R. Patra, "Routing in a Delay Tolerant Networking", to appear SIGCOMM 2004.

dent of the limits of the "link" layer. In fact, transfer opportunities would be limited in duration and bandwidth.

3. Bundles are delivered to stationary destinations located on a grid. This last assumption is a design choice. Our previous work [6] considered only mobile-to-mobile deliveries, however, we prefer mobile-to-stationary transfers here for a number of reasons. First, because the network is disconnected, the provision of naming and addressing is a fundamentally difficult problem (i.e., you cannot deploy DNS in a DTN). Geographic destinations are not only universal and easily looked-up, but can be made hierarchical for compressed routing tables. Second, destinations can easily be a desktop or office that is always on; in fact the address can be resolved as the bundle travels the network. For example, `john_doe@amherst.ma.usa` might be expanded to `john_doe@office346.-csblding.umass.amherst.ma.usa` as it arrives in town or on campus (just as in Milgram's famous experiment [15]). Secondly, destinations for all bundles might be a public access point leading to the Internet, and those may be static locations. Finally, a mobile-IP-like solution can be employed based on what we propose. A bundle may be delivered to the geographic location a person is expected to visit often (e.g., an office) and from there a meeting-only strategy (e.g., [6]) could be used to locate the user.

A Bundle Delivery Probability

Now we define the *MV* algorithm. When a node A meets another nodes B, they perform a bundle exchange through a number of steps. First, A gives to B a list of the bundles she carries with their destinations. Each bundle is also denoted by A with a likelihood of delivery according to the formula we derive below. A receives the same list from B and calculates the likelihood of delivering B's bundles. A now sorts the unioned lists by the likelihood of delivery, removes her own bundles (because she never deletes them), and also deletes from her buffer bundles that B has a higher likelihood of delivering. She then selects the top n bundles remaining, and requests from B all the bundles that she does not already store.

This is the same algorithm for exchange that we created for our previous work [6]. What we have replaced is the method for determining what bundles are most likely to be delivered. Specifically, *MV* determines a probability, $P_n^k(i)$, that the current node, k, can successfully deliver a bundle to a destination i within n transfers. Because it is more efficient, *MV* calculates an estimation of delivery likelihood assuming an infinite buffer at each node and limits the number of hops that are required in practice; in the next section we show empirically that this assumption provides good performance.

A.1 Derivation

We first derive $P_0^k(i)$, the probability that passing a bundle to some node k will result in the bundle being delivered with no more transfers (except the final delivery). In this case, the probability the bundle will be delivered is precisely the peer's probability of visiting the destination region. We assume that the probability of visiting a region in the future is strongly correlated with the node's history of visiting a region.

Accordingly, for each node k, we have a vector P_0^k with one entry for each region. Each entry i of $P_0^k(i)$ is based on the recorded movement of the node during the last t *rounds*, where a round is a fixed length of time (e.g., 1 hour or 1 day, depending on the movement speed of a typical node): $P_0^k(i) = t_i^k/t$, where t_i^k is the number of rounds node k visited region cell i during the previous t rounds. (This average is likely too simple for many contexts and movement patters; clearly, it could be substituted for a more sophisticated statistic, including an exponentially weighted moving average or Markovian process, which is what we plan for future work.)

Second, we assume bundles can be forwarded to at most one other node before being delivered to the destination. Both the current node k and the intermediate node j have a copy of the bundle and either (or both) can delivery it.

Let $P_1^k(i)$ be the probability of successfully delivering a bundle to region i starting with node k with the help of at most one intermediate node. This is given by:

$$P_1^k(i) = 1 - \prod_{j=1}^{N} (1 - m_{jk} P_0^j(i)), \tag{1}$$

where N is number of nodes in the system and m_{jk} represents the probability of node k and node j visiting the same region simultaneously. As with the movement probability, we define meeting probability based on meetings during the last t rounds: $m_{jk} = t_{j,k}/t$ where $t_{j,k}$ is the number of times nodes j and k are in the same region. Note, $m_{jj} = 1$. Thus, Eq. 1 represents the probability that neither node k nor any other node k visits the destination directly. Finally, we assume that bundles can be forwarded to no more than

n other nodes:

$$P_n^k(i) = 1 - \prod_{j=1}^{N} (1 - m_{jk} P_{n-1}^j(i))] \qquad (2)$$

Unfortunately, Eq. 2 does not scale with the number of hops or peers in the system. To calculate the probability, the meeting maps of all other nodes must be known. Fortunately, we found in our evaluations that $P_1^k(i)$ is a close enough approximation to $P_n^k(i)$ to serve.

IV. *MV* EVALUATION

To evaluate our *MV* algorithm, we ran a series of ns2 [17] simulations. We are interested in several metrics of the algorithm: bundle delivery rate, latency, and duplication of delivered bundles at the destination. These metrics are measured over over the offered load and the number of nodes in the network.

A Methodology

The success of DTN forwarding algorithms is wholly tied to the movement pattern of nodes. Traditionally, researchers have used the random waypoint model in lieu of empirical models. Such a movement model cannot be used for DTNs: if nodes move randomly, then no node is any better at delivering a bundle than any other. A successful routing algorithm exploits periodic, distinguishable movements.

We believe movements of humans and vehicles (e.g., buses and planes) are periodic. In separate work,[2] we have attempted to collect human meeting and mobility models; however, this proves to be a challenging task and we are not yet able to generate models appropriate for the algorithms we propose here.

To generate the periodic movements which MV could exploit, nodes follow a triangular movement pattern. Each node has a home location and two remote locations. At each timestep, nodes move among the three points, with the home location being chosen 50% of the time, and the remote points visited 25% each.

Nodes move at a uniform speed of 30m/sec in a world that is 2000m-by-2000m. Moving node's radios reached 250m. There are also twenty-five stationary *sinks* in a five-by-five grid. The sinks have no storage, do not generate bundles, existing solely to accept bundles sent to them.

[2]Currently under review: A. Clayton, M. Corner, D. Jensen, B.N. Levine, "Empirical Mobility Modeling" May 2004.

Peers generate bundles with exponential inter-arrival times according to a specified mean, varied in the experiments. Buffers at nodes are stated in terms of the number of bundles they could store. Each point on the graph represents 10 simulations with 10 different random seeds. Error bars shows standard deviation, which is small in all cases. All simulations ran for 1000 seconds; *MV* is not given time to warm up.

B Evaluation

We compare the performance of *MV* against three other algorithms. First, *no buffer*, where nodes can only deliver bundles by visiting the destination directly. This shows a lower bound on the connectivity of the network. As an upper bound on connectivity, we use *unlimited buffers* with flooding; if a route ever exists during the simulation, then the bundle is delivered. The final comparison is with a first-in-first-out (fifo) buffer control strategy: nodes take previously unseen bundles and when necessary pushing out the oldest bundles in their buffer to make space. Previous work [6] found that delivering bundles based on node meeting probability alone (not considering node location) is not significantly better than fifo. As a result this algorithm is not tested.

Figure 3 shows the effect of offered load on packet delivery rates for various algorithms. *MV* can deliver at 83% of the maximum achievable delivery rate; fifo can deliver 69%. As the offered load increased from an average of 0.1 bundles/second to 0.8 bundles/second, *MV* maintains a significant advantage. *MV* falls to 56% of the achievable delivery rate, while fifo falls farther to 40%.

Figure 4 compares the latency of delivered bundles amongst the algorithms. That *MV* has higher latency is to be expected since it is delivering bundles which other algorithms fail to deliver. The average distance traveled by a delivered bundle is lengthened. None of the other algorithms approach the latency the shortest possible path which is obtained by the unlimited buffer flooding algorithm.

In Figure 5, we see the cost of unlimited buffer: duplicate copies of bundles are delivered at the destination. While *MV* delivers more duplicates on average than fifo, this is offset by the better delivery rate. This indicates that the buffers could be used more efficiently by a more sophisticated version of *MV*.

We evaluate the effect of the number of peers moving in the system on performance. As we add peers, each peer adds load by sourcing more bundles, but provides bundle

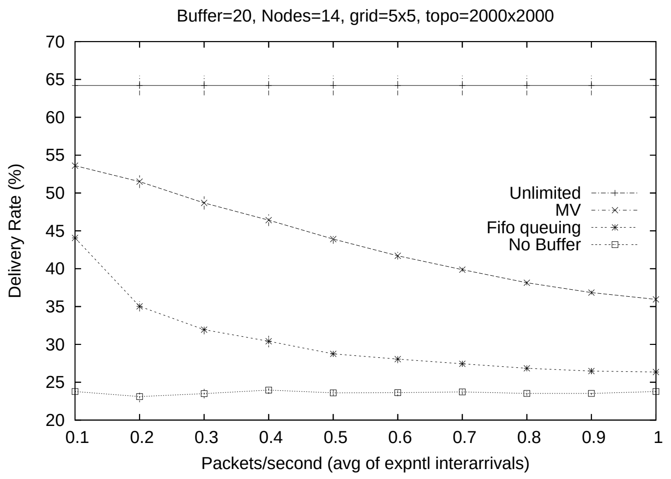


Figure 3: Delivery rate versus offered load.

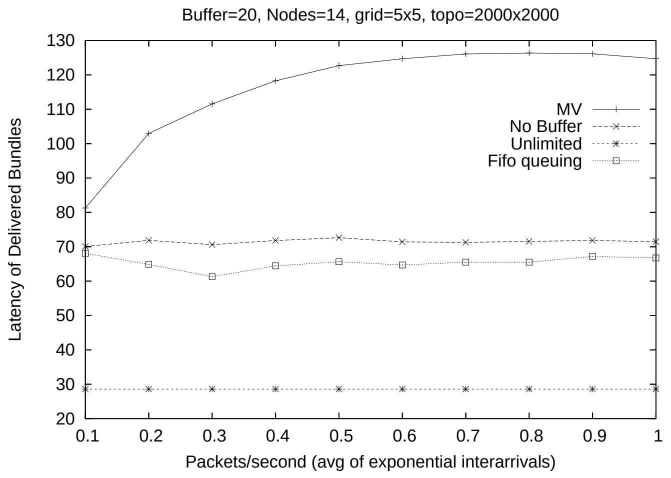


Figure 4: Latency versus offered load.

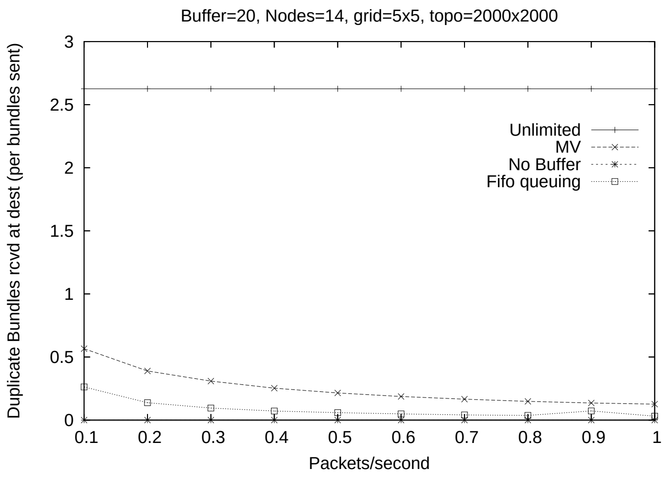


Figure 5: Duplicates received at destinations versus offered load. (normalized by the number of bundles sent).

carrying capacity. Since this is a peer-to-peer system, it is important that the network improve in performance as peers are added. In fact, we see in Figure 6 that in terms of delivery rate this is the case for *MV*, but not fifo. More extensive simulations are needed before we can say conclusively that *MV* is stable, but the results are encouraging. Figure 7 shows that *MV*does not scale well in terms of the number of duplicates delivered as the number of peers in the system increases. This tradeoff appears worth the higher delivery rate. We can see from the figure that *MV* is using its buffer more efficiently than the flooding strategy.

V. Control of Autonomous Agents in DTNs

In this section, we begin our design of a controller for autonomous agents that roam the DTN with the purpose of increasing network performance. We first show that the problem is NP-hard, which is the justification for our control-based approach. Then, we define the multiple individual controllers that optimize particular network metrics. In the next section, we define a multi-objective controller which coordinates the individual controllers. In Section B we evaluate the performance of our approach; in fact, we compare two methods of balancing the multiple objective the controllers: *nullspaces* and *subsumption*.

A Complexity of Scheduling Agent Movement

The overall complexity of the decision process for a set of agents servicing a DTN is NP hard, as we show here; a similar problem has been shown to be NP hard by Zhao et al. [25].

The problem of agents in a DTN can be stated as a reduced form of the *dial-a-ride problem* [20], which 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 [5], and is known to be NP-hard.

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

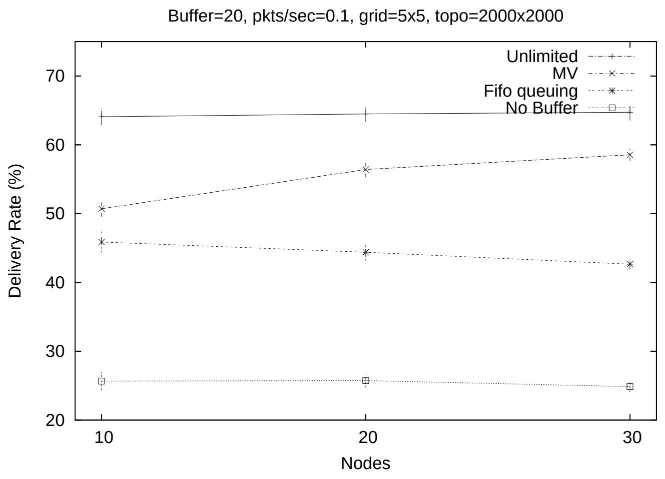


Figure 6: Delivery rate versus number of nodes.

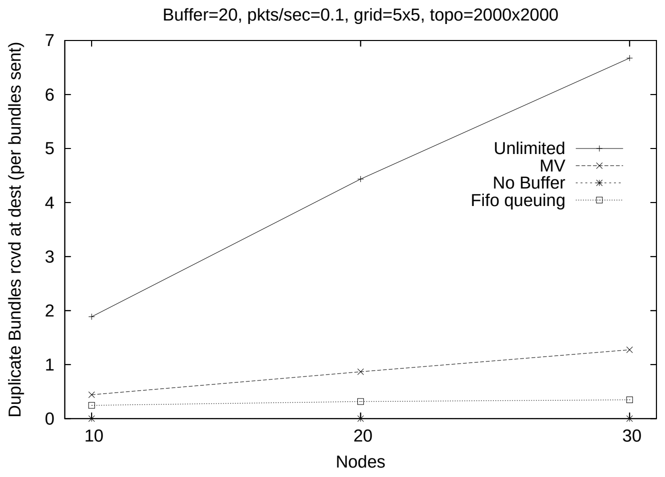


Figure 7: Duplicates received at destinations versus number of nodes (normalized by the number of bundles sent).

work is static.

Every request made to the dial-a-ride system for transport from a location A to a location B is exactly a bundle in the DTN sent from a node statically located at location A to a node statically located at location B. Since all of the participants in the network are static and incapable of communicating, the transport of the bundle from A to B must be accomplished by the agent. By optimizing the routing of bundles 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 must be NP-hard as well. As a result, we choose to use a approximating control strategy derived from the related works described in Section B.

B Metrics for the Controller

Our aim in designing a successful agent controller is to improve a variety of network performance metrics. Here we define them carefully since the control algorithm uses the metrics as a guide for ranking and choosing between various actions.

- *Bandwidth:* The total number of bundles which are currently active in the network.
- *Unique Bandwidth:* The total number of *unique* bundles which are currently active in the network (multiple copies of a bundle may exist in the network).
- *Bundle Latency:* The average amount of time it takes for a bundle to be delivered.
- *Node Latency:* The average time since each node was last visited by an agent. Since our perception of the network is maintained in a distributed manner, it is important that the all of the participants in the network be visited intermittently.

Each metric provides a specific optimization for the network. The total bandwidth metric measures use of the network; maximizing it ensures that every possible space available for the transport of a bundle is in use. The unique bandwidth metric measures the usefulness of the bandwidth usage, maximizing it ensures that bundles not already in transit are more likely to be selected. Minimizing the bundle latency metric prioritizes nodes which are sending or receiving bundles. Minimizing the node latency metric attempts to prevent starvation of nodes whether they are sending or receiving bundles.

Accordingly, each metric provides the basis for an individual atomic *base* 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 describe our implementation of these atomic controllers in Section VI.

Ideally, each metric could be optimized independently, but in practice the metrics are dependent upon each other. In traditional wired networks, this balance is achieved through the specification of fixed network parameter settings by a network administrator. In the case of agent-augmented DTN routing, the network's performance must instead be optimized by the control algorithm. By specifying where agents move the controller can ensure that the network performs adequately for the needs of its participants. Even while performing this optimization, the control algorithm must remain adjustable so that the balance of service metrics it achieves can be adjusted to suit the needs of a particular network. One example of such and adjustment would be favoring minimizing latency over maximizing absolute bandwidth. We describe the algorithm we used to achieve this balance in the forthcoming section.

VI. CONTROL FOR NETWORK ENHANCEMENT

In this section we specify the base controllers that make up the individual objectives that our multi-objective control algorithm balances.

A Metric Controllers

The goal of each controller is to specify a destination for the agent that optimizes a single specific network metric. This optimization is complicated by the fact that the gains for visiting some location in the network are amortized over the time it takes to reach the destination. Thus the selection of the location to visit next must take the distance to that location into account. As a result, a closer destination, which results in a smaller gain, maybe be preferable over a location with a larger gain which is more distant.

The details of the individual controllers are as follows:

ϕ_B : **Total Bandwidth Controller**
Traveling to any node N will increase the bandwidth of the network by the number of bundles which the agent can obtain from node N. The node chosen by this controller is the node which has the largest number of unseen bundles (amortized by travel time).

ϕ_U : **Unique Bandwidth Controller**
The unique bandwidth controller chooses the node that has the largest number of bundles not present anywhere in the network.

ϕ_D : **Delivery Latency Controller**
The delivery latency controller chooses the node whose average delivery time is the largest.

ϕ_N : **Node Latency Controller**
The node latency controller chooses the location n_i least recently visited by an agent, unless there exists a set of locations such that $\text{LastVisited}(n_i) + \text{TravelTime}(n_i)$ is less than $\sum_{n \in N}(\text{LastVisited}(n) + \text{TravelTime}(n_i))$. This insures that traveling time to visit the least recently visited location does not actually cause an increase in the node latency statistic.

A.1 Distributed control in a DTN

The atomic controllers described above assume perfect information about the state of the environment. However, just as we did in the previous section on *MV*, we require peers and agents to learn the state of the network only through meetings with other peers and agents. Each member of the network maintains the last known statistics for each known node in the network, as well as a timestamp of when the statistics were observed. These statistics are a superset of the information in the movement map and include all information necessary to compute the network statistics described above.

If two nodes have information for the same node, then the information with the latest time-stamp replaces the older information. Additionally, nodes make worst case hypotheses about what may have happened in an attempt to improve the performance of the network. For example, a node will assume that immediately after it left an area, someone in that area wanted to send a bundle and has yet to do so successfully. Unlike the observed information, these estimates do not propagate out to other nodes since they are not ground truth, and if a newer observation from another node contradicts this worse case assumption, it is discarded in favor of the more recent observation. Such worst case estimates ensure that the agents do not assume that in the absence of other information, unvisited areas and nodes do not have anything to send.

A benefit of this maintenance of statistics is that each member of the network has an approximate picture of the networks performance. For example, a human user of the network may know that although they are likely to be unsuccessful sending bundles from the bus stop, if they move over toward the library, their chances improve. Zhao et al. [24] provides a similar benefit, but they cannot adjust to changes in the network. This also means that any network maintainer can obtain an approximate overview of the DTN from any of its participants. (In our evaluations we do not assume any non-agent peers adjusts their movements.)

B Methods of Multi-Objective Control

The task of composing the controllers to achieve the mix of network performance desired by the network administrator falls upon the multi-objective control algorithm. Below, we present both a modified *nullspace* approach and a *subsumption* approach to the multi-objective control of the agents in the network. Both are techniques from robotic research; nullspace controllers use mathematics of linear algebra to coordinate controllers, while subsumption requires direct engineering by the system designers. In Section B we compare their performance by simulation.

The ordering of the metric controllers is the same for both approaches. Therefore, we begin by stating that ordering.

B.1 The Metric Controller Ordering

Because the multi-objective controller is using nullspace composition, the ordering of the metric controllers determines the order in which the network metrics will be optimized. The specification of this ordering provides a network administrator with a simple way of designing a network. The techniques of nullspace composition insure that the individual metric controllers are composited together to achieve as close to the desired network performance as possible.

The ordering used we used for control is:

$$\phi_{\text{Node Latency}} \leftarrow \phi_{\text{Bundle Latency}} \leftarrow \phi_{\text{Unique Bandwidth}} \leftarrow \phi_{\text{Bandwidth}} \tag{3}$$

In other words, network bandwidth is the most important concern, followed by unique bandwidth, followed by delivery latency, and finally node latency. This ordering is based upon the observation that the nullspace of equal 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 lower in the hierarchy. It does not make sense for bandwidth to be subject to unique bandwidth, since the actions chosen by the bandwidth controller are necessarily a super-set of those chosen by the unique bandwidth controller.

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 adminstrators or learned automatically. Likewise, the choice of thresholds is up to the end user. A network administrator can manipulate the performance of their network to suit its demands. As an example the definition of minimally acceptable delivery latency depends greatly upon the users and uses of the network. Further, we anticipate that these network controllers will also be deployed as part of autonomous agents which are at work on other tasks. The strength of a nullspace multi-objective approach is that it is easy to introduce additional individual controllers (such as for lawn-mowing) and to organize different control hierarchies. All of the networking controllers may be subject to a lawn-mowing controller or a lawn-mowing controller may be subject to all or some subset of the networking controllers. For any such hierarchy, nullspace control will synthesize the individual objectives to achieve the global objective.

B.2 Method 1: Nullspace Composition

Nullspace composition [13] has been used successfully to coordinate collections of controllers. The controllers are ordered into a hierarchy such that subordinate controller is forced to operate in the *nullspace* of controllers above it in the hierarchy. The nullspace is the set of inputs to a function where the value of the function does not change. In general these functions may be any arbitrary function. In our case, the functions are the network performance metrics. Since each controller operates in the nullspace of its predecessors, a lower controller is said to be *subject-to* [3] the higher one.

In the case of our network metrics, each controller is concerned with maintaining the value of a performance metric with respect to a threshold (e.g., latency must not fall below some value). Nullspace controllers are not designed specifically to work with thresholds, but rather to optimize the value of some function toward a local minimum/maximum. Therefore, we redefine the nullspace of the function so that it encompasses the range of values above or below some threshold as appropriate. This type of control can be obtained using the existing nullspace mechanisms. To do this, it is necessary to define two new functions:

$$\begin{aligned}\text{ThresholdMax}(f(), T, x) &= f(x) < T : f(x)\\ & \quad\; f(x) \geq T : T\end{aligned}$$

$$\begin{aligned}\text{ThresholdMin}(f(), T, x) &= f(x) > T : f(x)\\ & \quad\; f(x) \leq T : T\end{aligned}$$

To construct an optimizing thresholded control, the appropriate function is substituted in place of the original function. A minimizing threshold uses ThresholdMin. A maximizing threshold uses ThresholdMax. This new relation is defined as *subject-to-threshold*.

C Method 2: Subsumption

The subsumption approach to multi-objective control of the DTN agents differs from the nullspace approach. We still use the four metric controllers described in Section B and ordered by the same hierarchy as in nullspace composition.

The difference is in how the controllers dominate one another. Each controller examines the current value of the metric it is associated with. If the value of its metric is incorrect with respect to a set threshold, the controller outputs a geographic position toward which the agent should move. The output of dominant controllers completely *subsumes* the output of the subordinate controllers. Accordingly, subsumption makes it impossible to optimize controllers at once. This is an important difference from nullspace composition, which balances the outputs of *all* controllers within the set order and precedence. Experimentally (Section B) it can be seen that this difference leads to a decrease in subsumption's performance when resources are limited.

VII. Evaluation of Agent Augmented DTN

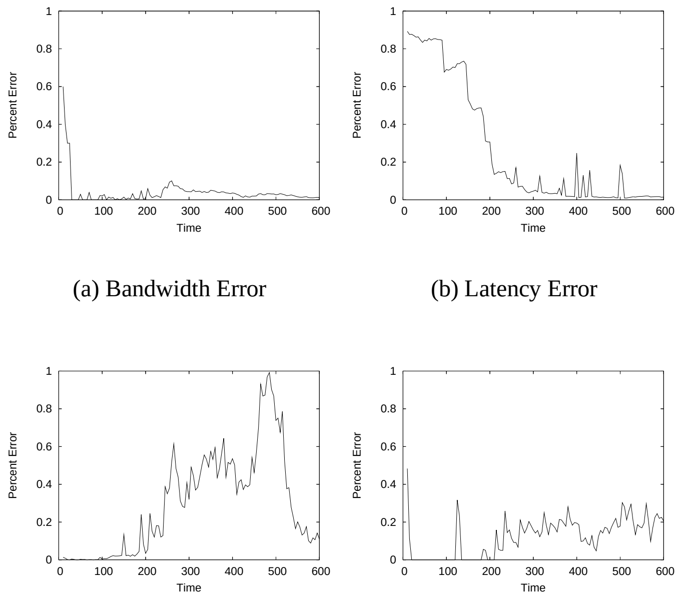
In order to explore and validate the ability of controlled agents to augment the performance of DTN routing we ran a number of experiments in ns2 [17]. The implementation of the DTN routing protocol is the same as in the previous experiment.

A Methodology

In the experiments, agents move at 30m/s and had a buffer capacity of 100 bundles. Traffic on the network is bundles sent to a particular sink. The interarrival time between bundles is chosen using a Gaussian distribution. Different choices for the mean of the distribution produced different traffic densities.

We use offered load rates between 0.1 and 0.5 bundles per second, which is in the lower half of those used in previous experiments with queuing strategies. This is appropriate since the traffic densities represent the difference between demand and capacity rather than the total capacity of the network. When non-agent participants appear in these experiments, they use the *MV* algorithm with a buffer size of twenty.

Since we are concerned with the ability of the agents to augment a network when there is a mismatch between demand and capacity, we run our initial experiments with a number of fixed sink locations with no network participants,



(a) Bandwidth Error (b) Latency Error

(c) Last Visited Error (d) Agent Location Error

Figure 8: Accuracy of the distributed status information over time.

with the exception of the agents, traveling between them. As a result all of the network capacity had to be borne by the agents. Later experiments to demonstrate other features of the network include non-agent participants moving in the network.

A.1 Accuracy and Error of distributed network statistics

The control strategies are dependent upon a high 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 nodes 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 the estimate of where each agent is going. The graphs of these results are shown in Figure 8. The results shown are the average error over time during a run with moderate traffic (0.3 bundles per second).

It can be seen that both bandwidth and latency error stabilize with low error as the experiment proceeds. Latency error is initially quite high due to pessimistic assumptions when no other information is available. The effect of this pessimistic assumption is also seen in the surge in error in last visited accuracy. The agent location error is relatively stable, although the difficulty in predicting agent location means that the resulting error is generally greater.

B Comparing Subsumption and Nullspace

To determine the appropriate multi-objective controller to use (nullspaces or subsumption), we compared the performance of the two algorithms by simulation. For each controller ten experiments were run with a moderate amount of random network traffic. The averaged results are show in Figure 9.

The nullspace approach outperforms subsumption when resources (the number of agents) are limited. As the number of agents increases, resources for delivering bundles become abundant, and both control algorithms converge on the same upper limit on accuracy. This indicates that the nullspace approach is using limited resources more effectively. When there are more than enough resources to provide effective bundle transport, the choice of control algorithm does not matter. However, when resources are limited (and thus their allocation more important) the nullspace approach balances the needs of the network while providing improvements in the most important metric, the percentage of bundles delivered.

C Network Performance Experiments

We ran several experiments to evaluate how much improvement agents offered to DTN routing.

C.1 Bandwidth

We explored explore the performance of DTN routing under increasing bandwidth loads. For these experiments the motions of the non-agent participants in the network are limited so that they could never communicate directly. All of the bundle traffic is carried by the agents in the network. The number of agents and the level of traffic is varied. A graph of the delivery rate resulting from these variations is shown in Figure 11. The response to increased traffic seems to match that of earlier experiments (Section B).

C.2 Responsiveness

The above experiments show that agents can successfully provide transport for otherwise partitioned areas in the network. In the following experiments, we examine the ability of agents to detect and respond to changes in the network over time.

We change the movement generation algorithm to restrict individuals to choosing from a subset of locations. We varied the subset of locations over time. Bundle traffic is generated uniformly to all destinations. The effect of this scenario is that areas of good network coverage shifted over time and agents respond accordingly to maintain uniform coverage.

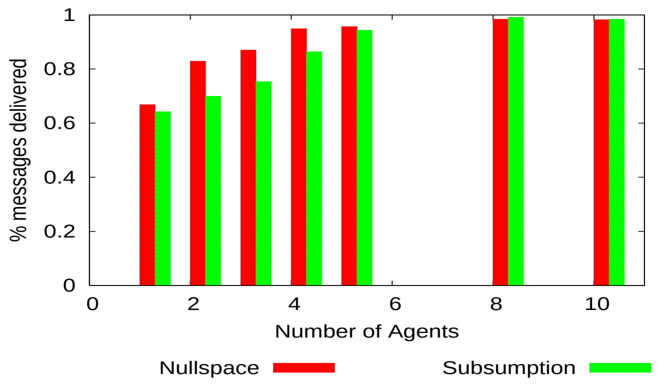


Figure 9: Delivery as a function of number of agents, 0.1msg/sec

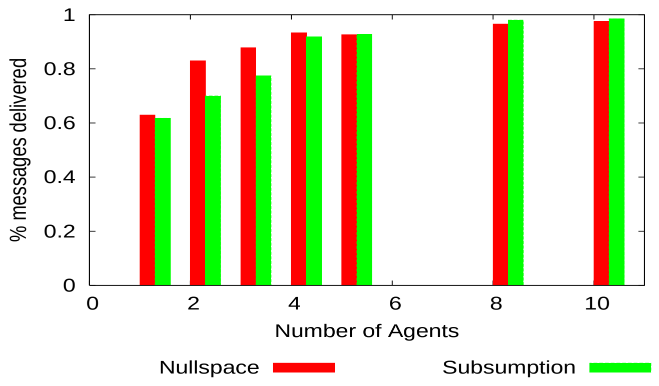


Figure 10: Delivery as a function of number of agents, 0.2msg/sec

For a period of time, peers in the network avoid half of the locations. As a result, only forty seven percent of bundles are delivered. When two agents are introduced, they detect the ignored locations and service them, resulting in seventy-six percent of the bundles being delivered, an increase of nearly fifty percent. With no agents present the average age of the oldest undelivered bundle is 478 seconds old (out of a 500 second experiment) indicating that at least some of the bundles sent would likely never get delivered, while with two agents present, the average age of the oldest bundle is 342 seconds, indicating that nearly all bundles will eventually be delivered.

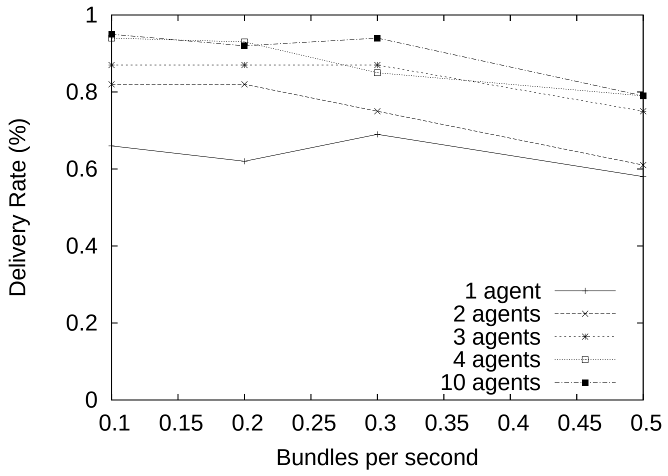


Figure 11: Percent of bundles delivered for different bundle rates and number of agents

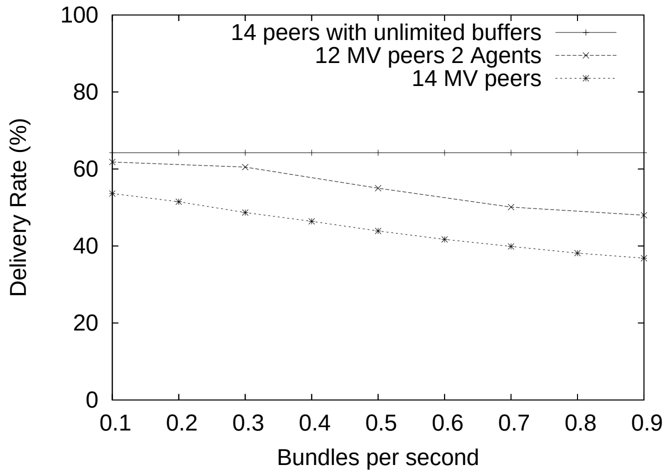


Figure 12: Performance of replacing two *MV* peers with two agents (to keep capacity the same).

D Combining Agents and Non-Agents

To demonstrate that the nullspace multi-objective control provides improvement over less informed behavior, we performed an experiment where we replaced two (out of fourteen) non-agent participants in the earlier MV experiments with two controlled agents. In this manner, the capacity of the network isn't increased, but two of the members of the network are now moving in order to improve the performance of the network. Figure 12 shows the resulting improvements in performance. This experiment also demonstrates that the agents can provide improvements in a network with non-agent participants.

VIII. CONCLUSIONS

Contemporaneous routing in ad hoc networks are insufficient for many real world scenarios due to the presence of partitions in the network. Instead, many real world situations require disruption tolerant networks (DTNs). In these networks traffic may be delayed at a node until such time as an appropriate method of delivery is found.

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 be successful in such environments it is necessary for it to take this into account. We have addressed two problems of importance to the development and maintenance of DTNs; routing strategies for moving bundles through a DTN and adding capacity to a DTN when demand exceeds the network's present capabilities.

We introduce the *MV* routing protocol which maintains a movement model of the participants in a DTN and uses this information to perform informed routing of bundles on the network. Experimentally we show that this routing algorithm shows large improvement over other techniques in achieving delivery rates significantly closer to the true optimal rate. These improvements continue even as traffic on the network increases by an order of magnitude.

The second problem addressed are mismatches between available capacity and demand. When such a mismatch occurs in a DTN the only way to add capacity is to increase the number of participants carrying bundles in the network. To achieve this we suggest the addition of a limited number of autonomous agents to the network area. The addition of these agents requires a control algorithm which can coordinate agent movements in order to optimize the performance of the network according to quality of service metrics desired by the network administrator. This requirements of control for the DTN required the development of a control algorithm which balanced various metrics with respect to given thresholds.

Two approaches to multi-objective control, subsumption and nullspace, have been implemented and explored. The thresholded nullspace approach extends the nullspace approach to handle the networking situation which needed thresholded control. Experimentally the threshold nullspace approach out-performed subsumption when resources are limited.

Experiments with this controller and various numbers of agents in simulated networks with varying traffic patterns show that autonomous agents are capable of providing effective improvements in networking capabilities. Further experiments in which the physical motion of participants varied also showed that the agents are capable of detecting and responding to changes in the structure of the DTN.

ACKNOWLEDGEMENTS

We are very grateful to Ping Hung Lee for programming many of the MV simulation results presented in this paper.

REFERENCES

[1] D. Braginsky and D. Estrin. Rumor routing algorithms for sensor networks. In *First International Workshop on Sensor Networks and Applications*, pages 22–31, 2002.

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

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

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

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

[6] 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.

[7] Baa04-13: Disruption tolerant networking (dtn), may 2004. Technical POC: Preston Marshall, DARPA/ATO.

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

[9] M. Grossglauser and M. Vetterli. Locating nodes with ease: Mobility diffusion of last encounters in ad hoc networks. In *IEEE Infocom*, 2003.

[10] K. M. Hanna, B. N. Levine, and R. Manmatha. Mobile distributed information retrieval for highly-partitioned networks. In *IEEE ICNP*, Nov 2003.

[11] 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.

[12] O. Khatib and J. Burdick. Optimization of dynamics in manipulator design: The operational space formulation. *International Journal of Robotics Research*, 2(2):90–98, 1987.

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

[14] 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.

[15] S. Milgram. The small world problem. *Psychology Today*, (2):60–67, 1967.

[16] Y. Nakamura. *Advanced Robotics – Redundancy and Optimization*. Addison Wesley, 1991.

[17] The network simulator, ns-2. Available at: `http://www.isi.edu/nsnam/ns/`.

[18] 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.

[19] N. Sarafijanovic-Djukic and M. Grossglauser. Last encounter routing under random waypoint mobility. In *Networking 2004, The Third IFIP-TC6 Networking Conference*, May 2004.

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

[21] UMass Segway Mobile Robotic Platform. Available at: `http://bodega.cs.umass.edu/segway/`.

[22] J. Sweeney, H. Li, R. A. Grupen, and K. Ramamritham. Scalability and schedulability in large, coordinated, distributed robot systems. In *Proceedings of the International Conferance on Robotic Applications (ICRA)*, 2003.

[23] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Technical Report CS-2000-06, University of California San Diego, July 2000.

[24] W. Zhao and M. Ammar. Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks. In *IEEE Workshop on Futrure Trends in Distributed Computing Systems*, May 2003.

[25] 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.