Detour-Based Mobility Coordination in DTNs

HANY MORCOS AZER BESTAVROS IBRAHIM MATTA
hmorcos@cs.bu.edu best@cs.bu.edu matta@cs.bu.edu
Computer Science Department
Boston University

Abstract

Commonly, research work in routing for delay tolerant networks (DTN) assumes that node encounters are predestined, in the sense that they are the result of unknown, exogenous processes that control the mobility of these nodes. In this paper, we argue that for many applications such an assumption is too restrictive: while the spatio-temporal coordinates of the start and end points of a node's journey are determined by exogenous processes, the specific path that a node may take in space-time, and hence the set of nodes it may encounter could be controlled in such a way so as to improve the performance of DTN routing. To that end, we consider a setting in which each mobile node is governed by a *schedule* consisting of a list of locations that the node must visit at particular times. Typically, such schedules exhibit some level of slack, which could be leveraged for DTN message delivery purposes. We define the Mobility Coordination Problem (MCP) for DTNs as follows: Given a set of nodes, each with its own schedule, and a set of messages to be exchanged between these nodes, devise a set of node encounters that minimize message delivery delays while satisfying all node schedules. The MCP for DTNs is general enough that it allows us to model and evaluate some of the existing DTN schemes, including data mules and message ferries. In this paper, we show that MCP for DTNs is NP-hard and propose two detour-based approaches to solve the problem. The first (DMD) is a centralized heuristic that leverages knowledge of the message workload to suggest specific detours to optimize message delivery. The second (DNE) is a distributed heuristic that is oblivious to the message workload, and which selects detours so as to maximize node encounters. We evaluate the performance of these detour-based approaches using extensive simulations based on synthetic workloads as well as real schedules obtained from taxi logs in a major metropolitan area. Our evaluation shows that our centralized, workload-aware DMD approach yields the best performance, in terms of message delay and delivery success ratio, and that our distributed, workload-oblivious DNE approach yields favorable performance when compared to approaches that require the use of data mules and message ferries.

[§] This work was supported partially by a number of National Science Foundation grants, including CISE/CSR Award #0720604, ENG/EFRI Award #0735974, CISE/CNS Award #0524477, CNS/NeTS Award #0520166, CNS/ITR Award #0205294, and CISE/EIA RI Award #0202067.

1 Introduction

Motivation: In a Delay-Tolerant Network (DTN), it is generally assumed that, on the one hand, there is no end-to-end path between a message's source and its destination, but that on the other hand, messaging between mobile nodes does not require immediate delivery. Email delivery is a canonical example of a DTN application over traditional IP networks. Over the last few years, interest in DTN applications over ad-hoc and infrastructure-less networks has mushroomed, fueled by envisioned applications that range from amorphous environmental sensing to social networking applications. There are many motivations for assuming an infrastructure-less networking environment. In some cases, such an assumption is necessary as is the case with networking applications envisioned for rural, under-developed, or impoverished milieus. In other cases, such an assumption may be motivated by cost considerations, given the amount of real-time traffic that needs to be carried as is the case for amorphous sensing applications, for example.

Even in settings – *e.g.*, large metropolitan areas – where networking (cellular or wireless 802.11) infrastructures may exist, issues of trust, privacy, and anonymity may make the use of infrastructure-less networks quite desirable. Example DTN applications along these lines include anonymous tipping for crime prevention and law-enforcement purposes, communication between covert agents in hostile countries for homeland security purposes, or the exchange of news or political messages between individuals when infrastructure networks are under the control of repressive regimes. Given the proliferation of wireless communication devices, it is conceivable that individuals will allow their mobile personal or vehicular communication devices to be used as part of an infrastructure-less overlay network to facilitate such applications.[1] For the purposes of this paper (and given the nature of the real traces we used in our experimental evaluation), we will restrict our attention to infrastructure-less DTN overlays established through the use of vehicular communication devices, noting that except for our trace-driven performance evaluation, our entire framework (model, algorithms, and results) is applicable to general mobile ad-hoc networks.

[1] The incentive to contribute one's storage and communication resources to establish such an overlay is not too different from the incentives for setting up Thor overlays (onion-routing) for anonymous file sharing, for example.

While one may assume that the infrastructure-less overlays we envision could be quite dense – *e.g.*, if all vehicles in a metropolitan area join it – thus enabling the formation of an ad-hoc mesh network, it is more likely that for the set of motivating applications we presented, such overlays will be sparse – *e.g.*, if vehicles belonging to a specific organization (a taxi company, cars owned by members of a university, vehicles that secret agents bugged to establish a covert communication overlay, etc.) join it. By sparse, we mean that for most of the time, nodes in such an overlay are not within communication range of one another and hence the existence of an end-to-end path between the source and destination of a message in such a network is highly unlikely, rendering useless conventional ad hoc routing techniques. Instead, in such sparse overlays, node mobility is exploited to circumvent the lack of an end-to-end path. A store-carry-and-forward model is adopted to deliver messages to their destinations minimizing the total delay of each message.

Towards Realistic Mobility Processes in DTNs: Commonly, research work in DTN routing assumes that node encounters are predestined, in the sense that they are the result of unknown, exogenous processes that control the mobility of these nodes. For example, a taxi hired for a trip between points A and B would use (say) the shortest path between points A and B, making that taxi's encounters with other vehicles that belong to the DTN overlay "predestined". While the above assumption (that node encounters are predestined) may make sense in some settings, it its too restrictive in general. For instance, the hired taxi may have multiple paths (of almost equal quality) to choose from when travelling from point A to point B. The taxi may even opt to take a detour that takes it away from such shortest paths as long as it can make it to its destination by a given deadline. For example, consider a free taxi at some location A and which needs to pick up a customer at location B in 15 minutes. Now assume that using the shortest path would get the taxi to location B in 5 minutes. Clearly, the taxi has some "slack" in its schedule which it could use to cover very many different paths between locations A and B (including travelling at slower speeds on alternate routes). The same observations could be made about mobility of nodes in general (individuals, vehicles, etc.): namely, that in most settings, *the exogenous processes that drive the mobility of nodes do not predetermine paths, but rather they establish constraints on the spatio-temporal coordinates of the start and end points of the node's journey.* Thus, in this paper, we argue that the specific path that the node may take in space-time, and hence the set of nodes it may encounter could be controlled in such a way so as to improve message delivery in DTNs.

To illustrate this idea, assume (for now) that nodes move in one dimension, and consider a node q whose schedule is given in the table in Figure 1. Each entry in this schedule gives the location and a corresponding time. For node q to meet this schedule, it has to be present at the given location at the specified time for all entries in the schedule. We refer to each two consecutive entries in such a schedule as a *waypoint*. Without loss of generality, assuming a maximum speed of unity, the schedule given in the table in Figure 1 allows q slacks of 9, 5, and 10 in the first, second and third waypoints, respectively. Figure 1 illustrates this schedule (and the slack it allows) by showing the location coordinate of an entry in the schedule on the x-axis, and the time coordinate of that entry on the y-axis. The rectangles shown in Figure 1 enclose the set of feasible (legitimate) paths that q could take during any waypoint. The more slack that q has, the wider the rectangles, and vice versa (if there is no slack whatsoever, then the rectangle will be reduced to a straight line, *i.e.*, the shortest

Time	Location
01	05
30	25
45	35
60	30

Schedule

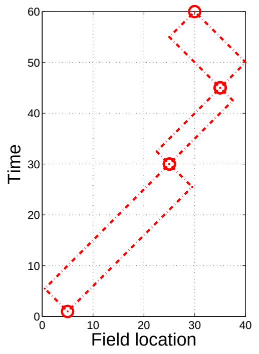

Figure 1. Visualization of a node schedule: Schedule entries are marked with circles. Rectangles mark the legitimate paths a node could take during each waypoint.

path). Since some of the legitimate paths could lead to useful encounters with neighbors while other paths could miss such encounters, it becomes evident that judicious mobility coordination by leveraging slack in node schedules could potentially improve the performance of a message delivery protocol (*e.g.*, improve message delivery rate, or decrease latency of message delivery). This is the main thesis of this work.

Paper Contributions and Overview: Given a set of nodes, each with its own schedule, and a set of messages to be exchanged between these nodes, the DTN Mobility Coordination Problem (MCP) is to find a set of node encounters that minimize message delivery delays while satisfying all node schedules. In Section 3 we give a concrete definition of MCP for DTN routing. Our formulation of the MCP for DTN routing is general enough that it allows us to model and evaluate some of the existing DTN schemes, including data mules and message ferries. We, then, show that MCP for DTNs is NP-hard. Next, in Sections 4, and 5 we propose two approaches to solve the problem: the first, called Detour for optimized Message Delivery (DMD), is a centralized heuristic that assumes (and leverages) knowledge of the message workload to suggest specific detours to optimize message delivery, whereas the latter, called Detour for maximizing Node Encounters (DNE) is a distributed heuristic that is oblivious to the message workload, and which selects detours so as to simply maximize node encounters. In Section 6, we quantify the performance of our detour-based mobility coordination approaches using extensive simulations, including trace-driven simulations using real schedules obtained from taxi logs in a major metropolitan area. Our evaluation shows that our workload-aware DMD approach yields the best performance, in terms of message delay and delivery success ratio, and that our workload-oblivious DNE approach yields favorable performance when compared to approaches that require the use of data mules and message ferries.

2 Related Work

Our work is relevant to a number of research communities, including: delay-tolerant networks, vehicular networks, and robot mobility planning.

Research in DTNs [2] assumes lack of end-to-end connectivity between communicating nodes, and leverages node mobility to transport messages that are, otherwise, wirelessly communicated in presence of end-to-end network connectivity. Research efforts in DTNs concentrate on finding an optimized algorithm to forward messages between nodes upon an encounter. The result is a routing protocol that outlines what messages to forward to which neighbor when an encounter takes place. The simplest solution is epidemic routing [3, 18], whereby all messages are replicated upon an encounter. Gossip routing and probabilistic routing [6, 10] are more efficient by being judicious in terms of utilizing available bandwidth and storage. Various solutions (*e.g.*, [14, 12]) have been proposed with different assumptions about the requisite knowledge of the node encounter pattern and messages workload.

The detour-based approaches we advocate in this paper differ from these efforts in that a node's motion (path) is viewed as a controllable *variable* as opposed to a fixed, uncontrollable input. While constrained by node schedules (among possibly other constraints), the mobility of a node could be manipulated to improve the performance of the entire system.

There have been some recent proposals for controlling/planning the mobility of a group of nodes in an ad-hoc network. The first group of such proposals [8, 7, 9] focused on actively mobilizing some nodes to bridge unconnected islands of nodes, hence improving the instantaneous end-to-end connectivity of the network. The second group of proposals [15, 21, 16, 19] suggested the use of special nodes as ferries/mules. These special nodes which are unconstrained in terms of their communication, computation or power resources act as a "postmen"; collecting messages from sources and delivering them to destinations, improving temporal connectivity.

In our work we demonstrate that judicious mobility coordination of nodes spares the need for external helper nodes (*e.g.,* ferries) while meeting all functional requirements of the nodes (*e.g.,* spending a given percentage of the time monitoring the environment in a sensor network, or satisfying a given node schedule of locations and deadlines in an ad-hoc network). Notice that the use of "helper nodes" (ferries or mules) implies the use of an infrastructure of sorts. As we alluded earlier, a major motivation for the use of DTN overlays is to avoid the use (and hence the need to trust) any infrastructure, making these approaches less attractive for such applications.

Information dissemination in vehicular networks [13, 20, 5] is another example of DTN applications. The main difference here is that mobility takes place on mostly one-and-half dimension (*i.e.* mobility on a network of roads) at higher speeds. Vehicular networks are less concerned with energy, storage, and communication constraints as it is conceivable that vehicles can easily host powerful computing platforms (compared to *e.g.,* hand-held devices). Also, the DNE technique we devise in this paper differs from these efforts in that, it attempts to "guide" node mobility (as opposed to react to it) in order to increase the number of node encounters leading to an improved message delay.

Sensor and robot mobility coordination (see [17, 11] for example works) is an active field of research. Here the main target is to plan mobility so as to achieve some global goal (*e.g.* uniform field coverage, or distributed target tracking). Unlike our work, mobility planning in such settings is not constrained by other exogenous requirements – namely, the need to satisfy the constraints imposed by individual node schedules.

3 Mobility Coordination for DTNs

In this section, we define the Mobility Coordination Problem (MCP) for DTNs, show that this problem is NP-hard, and formulate it as a constrained optimization problem.

Definitions and Notation: We consider a DTN overlay consisting of n mobile nodes. We assume that any two nodes within distance less than or equal to a fixed communication range r can communicate. We also assume that the maximum speed of motion for a node i is v_i, and without loss of generality assume that $v_i = v_{\max}$. We define a message (or communication) workload G in a DTN to be a vector of m of messages in the system. Any message $g \in G$ is a tuple $g = (t, o, d)$, where t is the time at which message g originates (*i.e.,* arrives), o and d are the identifiers of the source and destination for message g, respectively. Each node i in the DTN has a schedule s_i that consists of a list of $L(s_i)$ tuples of the form $u_{ij} = (\tau_{ij}, l_{ij})$, where $1 \leq j \leq L(s_i)$. To satisfy a schedule entry u_{ij}, node i has to be at location l_{ij} at time τ_{ij}. For i to satisfy its schedule, it has to satisfy u_{ij} for all $1 \leq j \leq L(s_i)$.

The MCP problem: Given a set of n nodes, each with its own schedule, and given a message workload G, the MCP problem is to find a set of node encounters that minimize message delivery delays while satisfying all node schedules. Solving the MCP problem for DTNs amounts to synthesizing the mobility profile for each node. The mobility profile for node i gives the location of node i at time t for $1 \leq t \leq T$, where T is the evaluation epoch. Any feasible solution to the MCP problem must satisfy the maximum speed requirement, *i.e.,* no node is allowed to move with a speed higher than $v_{\max}$. Message delivery is through node encounters induced by node mobility profiles. Node encounters must satisfy the communication range requirement, *i.e.,* nodes can only communicate if the distance between them is less or equal to r. We show that MCP for DTNs is NP-hard by reduction to the Minimum Latency Tour (MLT) problem [4], which we define next.

The Minimum Latency Tour (MLT) Problem: Given a set of locations $P = \{p_1, p_2, \ldots, p_n\}$ in a metric space where a symmetric distance function $d_{i,j}$ is defined between each pair of locations p_i and p_j, the MLT problem amounts to finding a tour on the set P minimizing $\sum_{i=1}^{n} \ell(i)$, where $\ell(i)$ is the latency to visit location p_i for a mobile element starting at some given location p_{init}. The MLT problem is known to be NP-hard for general metric spaces [4].

MCP for DTNs is NP-Hard: To show that the MLT problem is a special case of MCP for DTNs, consider a DTN with $n+1$ nodes. The initial locations of the first n nodes is set to $P = \{p_1, p_2, \ldots, p_n\}$, and the initial location of the $(n+1)^{\text{th}}$ node is set to p_{n+1}. The schedule of the first n nodes is set to their respective locations for the entire time (*i.e.,* schedule s_i of node i is given by $s_i = \{(t, p_i)\}, 1 \leq t \leq T$). The communication workload of the first n nodes is empty. This in effect "pins down" the first n nodes to their initial locations throughout the epoch T. The schedule s_{n+1} of node $n+1$ consists of two entries $s_{n+1} = \{(1, p_{n+1}), (T, p_{fin})\}$. For some random field location p_{fin} and some time T such that $T \gg 1$. This schedule gives node $n+1$ the freedom to roam around the field long enough to have visited all n fixed locations $\{p_1, p_2, \ldots, p_n\}$, and finally goes to some random location p_{fin}. The communication workload of node $n+1$ is set so as to deliver n messages (one to each of the static nodes), such that the origination time of all messages = 1. This reduction to the MLT problem proves that MCP for DTNs is NP-hard.

MCP as an Optimization Problem: In the remainder of this section, we cast the MCP for DTNs as an optimization problem.[2] The resulting formulation is an integer nonlinear problem requiring a search of the entire space, which is only feasible for the smallest of problem sizes. Nevertheless, while such formulation does not yield a practical solution for realistically-sized DTNs, it provides the reader with insights into the main optimization variables and the constraints that

[2] This formulation may be skipped on a first reading of the paper.

shape the solution of the problem.

As before, we consider a DTN over an epoch T, with n mobile nodes, each with a maximum speed v_i, a communication range r. Let G represent the message workload consisting of m messages. Furthermore, for any given message g, we use $\Theta(g)$, $O(g)$, and $D(j)$ to denote the origination time, source and destination of g. We also use $\text{Dist}(a,b)$ to denote the distance between two field locations a and b.

Let the *mobility matrix* X be a $T \times n$ real matrix, such that $X(t,i)$ denotes the *derived* location of node i at time t. Let the *message carrier matrix* Y be a $T \times n \times m$ binary matrix, such that $Y(t,i,g) = 1$ if and only if node i buffers message g at time t. Let the *neighborhood matrix* E be a $T \times n \times n$ binary matrix such that $E(t,i,k) = 1$ if and only if nodes i and k are neighbors at time t. Let the *message host matrix* H be a $T \times m$ integer matrix such that $H(t,g)$ is the id of the node that hosts message g at time t. Finally, let the *delivery time matrix* Δ be an integer vector of length m, such that $\Delta(g)$ is the time that message g reaches its destination. In the following equations we use: i,k as node indices ranging from $1 \ldots n$, g as a message index ranging from $1 \ldots m$, j as an index in the schedule of a given node i, j ranges between $1 \ldots L(s_i)$, and t as a time index ranging between $1 \ldots T$ (unless specified otherwise).

The MCP in DTNs could be formulated as an optimization problem – namely to minimize the objective function.

$$\sum_{j=1}^{m} \Delta(j) - \Theta(j) \tag{1}$$

subject to the following constraints:

$$X(\tau_{ij}, i) = l_{ij} \tag{2}$$

$$Dist(X(t,i), X(t-1,i)) \le v_i, 2 \le t \le T \tag{3}$$

$$H(t,g) = \sum_{i=1}^{n} i \cdot Y(t,i,g) \tag{4}$$

$$E(t,i,k) = \begin{array}{l} 1, \text{ if } Dist(X(t,i),X(t,k)) \le r \\ 0, \text{ otherwise} \end{array} \tag{5}$$

$$Y(t,i,g) = 0, 1 \le t < \Theta(g) \tag{6}$$

$$Y(t,i,g) = 1, i = O(g), t = \Theta(g) \tag{7}$$

$$\sum_{i=1}^{n} Y(t,i,g) = 1, \Theta(g) \le t \le \Delta(g) \tag{8}$$

$$Y(t,i,g) \le E(i, H(t-1,g), t), \ \ \Theta(j) < t \le \Delta(j) \tag{9}$$

$$\Delta(g) = \sum_{t=1}^{T} t \cdot Y(D(g), g, t) \tag{10}$$

The role of the above constraints can be explained as follows: Equation 2 constrains the mobility matrix in order to satisfy the schedule of each node. Equation 3 constrains the mobility of all nodes such that the travelled distance during any time unit does not exceed the maximum speed of mobility. Equation 4 defines the host of each message at all time units (this is set to zero before the message arrives at its origin and after it is delivered). Equation 5 constrains encounters to be between nodes within communication range of each other. Equation 6 ensures that no node would host a message before this message originates, and Equation 7 ensures that when a message g originates, it is only hosted at the node $O(g)$ that originated it. Equation 8 ensures that messages are not duplicated. Equation 9 ensures that messages are communicated between nodes only when nodes come into contact with one another. Equation 10 defines the time of delivery of all messages.

As we mentioned above, solving the above optimization problem entails solving an integer non-linear problem, which is not tractable for practical systems.[3] In the next section, we examine a restricted (serialized) version of this optimization as well as other distributed heuristics.

4 Detour for optimized Message Delivery

So far, our formulation of the MCP for DTNs, though complete, is impractical to solve. Therefore, in this section we propose to solve a serialized version of the problem. By serialized we mean that we optimize the delay for each message in the message workload G *in order of message origination time*. More specifically, we consider one message at a time and identify for that one message the node encounters that help minimize the delay of that one message, subject to the constraints of the current schedule of all nodes. Each such encounter is then *committed* by adding the spatio-temporal coordinates of the encounter in the schedule of the nodes involved in that encounter, forcing these nodes to take the necessary "detours" to synthesize these encounters. This process is then repeated for each subsequent message in order of origination time. Notice that decisions made to optimize delay for message g are considered as input when optimizing delay for message $g+1$ – hence the "serial" nature of this optimization as opposed to the optimization approach we presented in Section 3, which optimizes the detours that each node takes for the *entire message workload*. In the remainder of this paper, we refer to this solution of MCP for DTNs as the Detour for optimized Message Delivery (DMD) approach.

The output of the DMD approach is an *augmented schedule* for all nodes.[4] An augmented schedule is a copy of the original schedule plus more tuples of the form (τ_{ij}, l_{ij}), making an augmented schedule more restrictive (*i.e.*, featuring less slack) compared to the original schedule. Notice that by definition, an augmented schedule is always feasible, since detours are only added to a schedule if they are feasible for the node to satisfy (subject to maximum speed constraints, *etc.*)

The Potential Encounter Graph (PEG): In order to obtain the feasible detours as described above, we represent potential encounters between nodes as a directed graph that has two groups of vertices V_1, and V_2, and two groups of edges E_1, and E_2. V_1 represents actual nodes in the system, while, V_2 represent *potential encounters* between any two nodes. There are no edges between vertices $v \in V_1$. Two vertices $v_a, v_b \in V_2$ are connected if there exists a node n_c that can have both encounters represented by v_a and v_b simultaneously (i.e., taking part in both encounters is physically possible, based on the physical distance between the two encounter locations, their respective times, and the maximum speed). We call these edges *vertical* edges, E_2. If two vertices in V_2 (*i.e.*, encounters) are connected with a vertical edge, then the cost of this edge is the difference between the earliest times each of the two encounters could take place. There are also edges between vertices in V_1 and vertices in V_2 such that each node (represented by a vertex in V_1) is connected to all vertices representing encounters that this node takes part in. We call these edges *horizontal* edges, E_1. Horizontal edges have an associated cost of 0, ini-

[3] Non-linearity stems from the definition of a contact between two nodes (Equation 5).

[4] The final mobility of each node could then be determined by any basic technique (*e.g.*, move radomly between entries in the augmented schedule, move at maximum speed and wait at destination, *etc.*) as long as the augmented schedule is satisfied. Notice that the resulting mobility will satisfy the original schedule since the augmented schedule is more restricted than the original one.

n_1		n_2		n_3	
time	location	time	location	time	location
1	8	1	18	1	45
20	13	38	38	30	35
50	33	70	23	58	23

Table 1. Example of a node schedule

	1	2	3	4	5	6
1		30	45.5		24	32
2			4.5			
4		21.5	27		15.5	23.5
5		6	11.5			8
6			3.5			

Table 2. Costs of edges between vertices in V_2

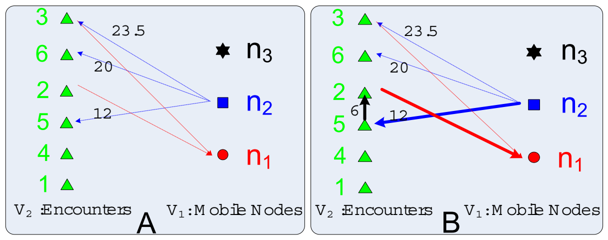

Figure 3. Using PEG in determining the route of a message g that originates at n_2 to n_1 at time = 18, (A) applying the five steps, (B) finding the shortest path

tially.

To handle a message g arriving at node n_1 and targeting node n_2 at time t_x we do the following: (1) Temporarily eliminate all horizontal edges between all vertices representing nodes other than n_1 and n_2; (2) Assign direction from V_1 to V_2 to all horizontal edges coming out of node n_1; (3) Assign direction from V_2 to V_1 to all horizontal edges going into node n_2; (4) Eliminate all edges incident to either n_1 or n_2 connecting these two nodes to encounters taking place earlier than t_x; and (5) Each horizontal edge coming out of $n_1 \in V_1$ to a node (encounter) $e \in V_2$ is assigned cost that is the difference between the time of message arrival, t_x, and the time of having encounter e. Finding the shortest path between n_1 and n_2 in the resulting graph amounts to finding the list of encounters, which if *committed*, would deliver the message from the source to the destination incurring the least possible latency.

To illustrate the above process, consider three nodes n_1, n_2, and n_3 in a one-dimensional field of size 60. The nodes' schedule is given in table 1. Figure 2 (left) gives a visual representation of this schedule. By inspecting Figure 2 (left), it is possible to locate the potential encounter points between the different waypoints of the three nodes, we number these encounters 1 through 6 (Figure 2-center). Encounters 1 and 3 take place between n_1 and n_2, encounter 2 takes place between n_1 and n_3, while encounters 4, 5, and 6 takes place between n_2 and n_3. From this graph we can construct the potential encounter graph (PEG), shown in Figure 2-right. Notice that for the sake of clarity, Figure 2-right shows only horizontal vertices, E_1, but does not show vertical edges, E_2. Table 2 gives the edges in E_2. In Table 2, the label of the row gives the source vertex of the edge while the label of the column is the destination vertex of this edge. A blank entry in Table 2 means that there is no node that can carry a message between two encounters. For example, encounter 1 takes place between n_1 and n_2. Encounter 4 takes place between n_2 and n_3. However, n_2 cannot simultaneously satisfy both encounters (given its original schedule), hence there is no vertical edge between their corresponding vertices.

A detailed description of the PEG construction process, along with an illustrative example is given in the Appendix.

Detour Synthesis using PEG: Once constructed, the PEG graph is used to find the set of encounters that minimize the delay for each message, in order. As we alluded before, DMD considers one message at a time.

For a message g originating from node n_1 to node n_2 at time t_x, we proceed as follows: (1) We temporarily eliminate all horizontal edges between all vertices representing nodes other than n_1 and n_2. This is done since we need to find the set of encounters to deliver the message (*i.e.*, vertices in V_2), hence there is no need to going back to V_1. (2) We assign direction from V_1 to V_2 to all horizontal edges coming out of node n_1. Since the only time we cross from V_1 to V_2 is when the message originates at n_1. (3) We assign direction from V_2 to V_1 to all horizontal edges going into node n_2. Since the only time we cross back to V_1 from V_2 is to deliver the message the message to n_2. (4) We eliminate all edges incident to either n_1 or n_2 connecting these two node to encounters taking place earlier than t_x. This is done to prevent past encounters from being used to deliver future messages. And, (5) to each of the remaining horizontal edges *going out of the message source*, we assign a cost that equals the difference between the time at which the message originates and the time the respective encounter takes place. This represents the amount of time a message waits in the source node until the first encounter. Similar wait times in intermediate destinations is represented by weights of the vertical edges E_2.

Finding the shortest path between the n_1 and n_2 in the resulting graph amounts to finding the list of encounters, which when *committed* would result in the delivery of the message from the source to the destination, while incurring the least possible latency.

Figure 3 shows this procedure for a message g that originated at node n_2 to node n_1 at time 18. Figure 3(A) shows applying the five steps on the PEG, while Figure 3(B) shows finding the shortest path on the resulting graph. The resulting path yields a minimum delay of 18 secs. In this path, the message waits at n_2 for 12 secs, then is transported to n_3 (encounter 5), where it waits for another 6 secs, and is finally delivered to n_1 (encounter 2). Notice that for message g to be delivered in this delay, encounters 2 and 5 must be confirmed. Thus for the following messages, we confirm encounters 2 and 5, partially rebuild the PEG graph, and re-apply this procedure on the new PEG.

5 Detour for maximizing Node Encounters

The DMD approach is centralized in nature, thus imposing limitations on its applicability in practical settings. In this section we propose a heuristic that introduces detours with so as to maximize the number of Node Encounters. Using this DNE heuristic, instead of trying to explicitly minimize the delay of every message in the system (as in DMD), we rely on increasing the number of encounters between nodes in the system. The motivation is that by using the slack in the schedules to create new encounters between the nodes, we are likely to increase the probability of having useful encounters, which could lead to maximizing the success ratio of message deliveries and minimizing message latencies.

In DNE, we assume that there is an ordered set Ω of suggested encounter locations along with a frequency parameter μ and start time t_0. The set Ω as well as μ and t_0 are known to all nodes in the system. Based on its schedule, a node n identifies the locations in Ω that could be visited at time t_0 without violating its own schedule. Let us denote these feasible lo-

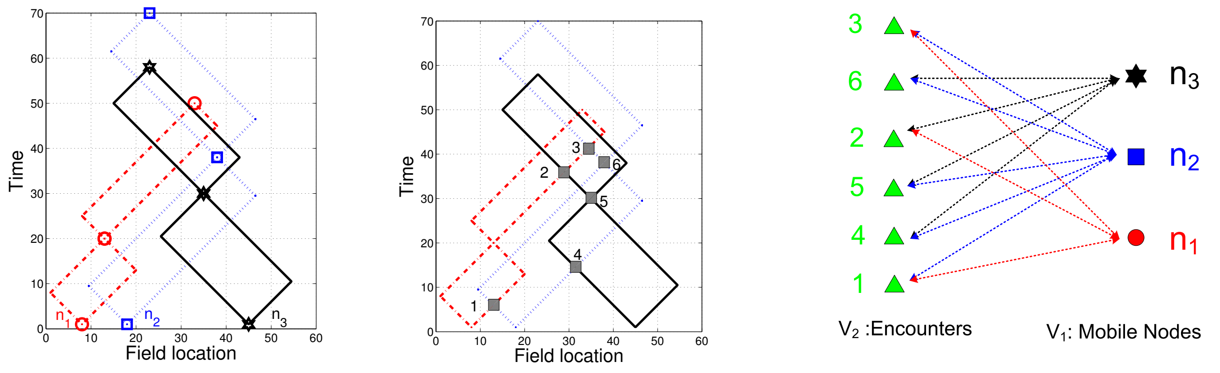

Figure 2. A three-node schedule (left) with potential encounters 1 through 6 (center) and PEG graph (right).

cations by $\omega \subseteq \Omega$. In case $\omega \neq \phi$, *i.e.*, n could make one or more of the proposed locations at time t_0, n selects its target location based on the order of the original set Ω. If $\ell_0 \in \omega$ is the highest-ranked location in Ω, then node n adds the tuple (t_0, ℓ_0) to its schedule. This has the effect of "committing" that node n will be at location ℓ_0 at time t_0. Node n then repeats the same process for all times $t_k = t_0 + k \times \mu$, for $k = 1, 2, \ldots$. The outcome of this procedure is an *augmented schedule*, with the locations of the added entries being all from the same set Ω. Hence there is higher chance of having the same meeting point added to the schedule of more than one node, which in effect creates new encounters.

As with DMD, the actual motion of the node is determined using any basic strategy as long as the augmented schedule is satisfied.

6 Performance Evaluation

In order to evaluate the efficacy of our detour-based approaches, we developed a mobility simulator. Our simulator models the mobility of the nodes by keeping track of the location of each node at each time unit. It also models messages origination and delivery. Since our goal is to evaluate the synthesized mobility of our detour-based techniques, we make simplifying assumptions about the communication model as we assume that nodes within certain communication range could successfully exchange data. We assume that the size of exchanged messages is small with respect to the bandwidth in a single contact between two nodes. We also, willingly, overlook the storage issue of the nodes. We do this motivated by current advances in storage technology that make memory chips of tens of gigabytes available off-the-shelf.

We compare the mobility resulting from the use of our approaches to three basic alternatives. The first two are wait-at-source (WAS), and wait-at-target (WAT) approaches. In WAT, given a schedule, nodes take the shortest path to the destination of the current waypoint and wait there, *i.e.*, spend all the slack time waiting at the target. In WAS, all the slack time is spent at the source of the waypoint, and then nodes take the shortest path to the destination of the waypoint. The third approach is random mobility (RND), in which nodes move randomly from the source to the destination of any waypoint provided that the schedule is satisfied. The point of these algorithms is to gauge the improvement in performance attained by our DMD and DNE detour-based approaches.

It should be clear that we are not trying to design a routing algorithm, nor a message forwarding technique. Rather our work focuses on the synthesis/coordination of node mobility subject to schedules and message workloads. Hence, after obtaining the node location across time (*i.e.*, the result of applying the various mobility synthesis/coordination approaches), we can easily infer the contact model induced by the synthesized node mobility. The resulting node encounters, along with the message workload can be fed to any message routing algorithm to decide which messages to forward to which neighbor upon an encounter. The details of the specific routing algorithm are orthogonal to our work. In this paper, we choose to use an optimum algorithm that calculates the optimum forwarding path for every message, given the current node contacts. This means that, results we report here are the best case performance for **all** mobility synthesis approaches. Notice that the exact performance of the optimization program could be attained in a distributed fashion by communicating all messages to other nodes upon contact, *i.e.*, using flooding. A more efficient algorithm is to use gossiping [6, 10] to avoid much of the problems associated with flooding while reaping some of its benefits. In short, we stress that the message forwarding technique is orthogonal to our work, and any technique could be used here.

The optimized algorithm we used to find the optimum path for every message to reach its destination is based on the formulation given by Jain *et. al* [14].

6.1 Evaluation Using Synthetic Workloads

Schedule Generation: Every node starts at time = t_{current} (initially, $t_{\text{current}} = 1$) at a random location in the field loc_1. The entry $(t_{current}, loc_1)$ is added to the schedule. Then we randomly select another location loc_2 in the field such that the minimum time to move from loc_1 to loc_2 is t. For the loc_2 we assign time t_s

$$t_s = t_{current} + t + (\kappa \times r) \tag{11}$$

where κ is the *maximum slack* we allow in any waypoint, and r is a uniform random variable such that $r \in [0, 1]$. The entry (t_s, loc_2) is appended to the schedule. We repeat this process until the end of the simulation time is reached.

Message Generation: Message sources and destinations are randomly generated such that the source and destination of any message are not the same. The message arrival process follows a Poisson process with mean 0.5 message/sec.

Performance Metrics The performance metrics we use are the *delivery ratio* and *average delay*. Delivery ratio is the ratio of successfully delivered messages to the total number of messages generated. Average delay is measured for *delivered* messages only.

Baseline Results: We simulated a field of 30x30 city blocks where nodes can communicate only when they are at the same

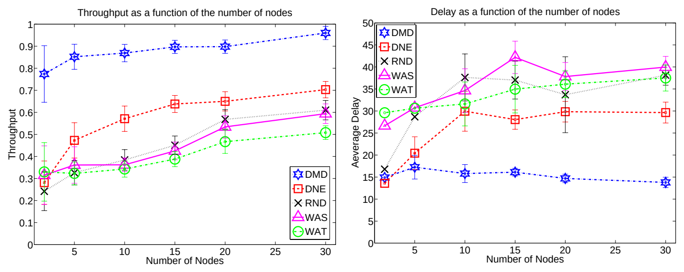

Figure 4. Performance of mobility synthesis approaches.

intersection. The simulation runs for 100 seconds. In the following graphs, each point is the average of 20 simulation runs, with the 95% confidence interval shown as well.

In the first set of experiments, we compare DMD and DNE to the basic WAT, WAS, and RND approaches. In these experiments, the maximum slack allowed κ was set to 15.

Figure 4 (left) shows the delivery ratio of the five approaches. As expected the delivery ratio of all approaches improve as we increase the number of nodes, which in turn increases the number of encounters, thus enabling more messages to get delivered. This effect is more evident for WAS, WAT, and RND. DMD is able to achieve from 80% to two times higher delivery ratios than the basic algorithms. This underscores the importance of our PEG-based approach, and the value of the encounters it chooses. DNE yields from 30% to 80% higher delivery ratios compared to the basic algorithms, confirming our intuition that a simple distributed mobility coordination algorithm that focuses only on increasing the number of encounters (while being oblivious to the message workload) is bound to improve the delivery ratio.

Figure 4(right) shows the average delay of delivered messages. The difference between DMD and the other mobility synthesis approaches is very clear; it has between 13% and 170% less delay compared to the basic RND, WAS, and WAT techniques. On the other hand, DNE achieves from 13%-40% lower delay than WAS, WAT, and Random. An interesting point is that increasing the number of nodes increases the average message delay for all approaches, except for DMD, *i.e.*, for all distriubted workload-incognizant approaches. The reason is that, DMD creates encounters between nodes that are certain to help minimize message delay. While, using the other approaches, increasing the number of nodes creates more encounters that help deliver more messages but not necessarily on the most optimum path, yielding higher average message delay.

To summarize, DNE improves the delivery ratio and the average message delay compared to the basic approaches. DNE's efficiency is more evident in networks with low node density (typical in DTN networks). DMD achieves the best message delivery ratio and average delay.

Effect of Partially Following Detours: The goal of this experiment is to measure the effectiveness of DNE in two cases – namely when (1) only a given percentage of the nodes follow detour hints provided by DNE, and (2) When all nodes follow DNE hints with some probability. Both of these scenarios are motivated from the observation that nodes in a DTN are autonomous and may opt not to follow routes suggested by DNE. Figure 5 (left) shows the performance as a function of the probabilities of following hints for different number of nodes. Clearly, the performance improves as nodes follow hints more consistently. As we hinted before, experiments with more nodes have higher success ratio and longer average message delays. It is interesting to see that the gain is almost linear (as a function of the percentage of nodes following hints). The rational is that as nodes try to make the meeting points more often, the higher the chance of actually having a useful encounter that could be used to deliver messages. Figure 5 (right) shows the effect of having different percentages of nodes completely follow the detours proposed by DNE, while the rest of the nodes completely disregard these proposals. Similar to Figure 5 (left), performance improves linearly as the percentage of complying nodes increases. This is expected from a distributed algorithms, where nodes have no way of knowing other nodes decisions. The performance of the entire system improves as more nodes comply with the distributed protocol.

Comparison With Ferries and Data Mules: As we discussed in Section 2, we propose better planning of slack time instead of relying on external helper elements: *i.e.*, message ferries and data mules to deliver messages. Of course, if there is very little slack, then there is no way to improve the system performance but to rely on the help of external nodes. However, as we show in the results in this section, if well-planned, enough slack might prove very useful, resulting in performance that is even better than relying on external helping elements. To that end, in this set of experiments, we compare DMD and DNE to data mules and message ferries.

Message ferries (referred to as NIMF in [21]) are external helper nodes that are not limited in power, computation nor communication capabilities. A ferry has a well-defined route in the field. When a node has enough slack, it approaches the route of the ferry, and upon encountering it, the node unloads messages to send unto the ferry, and gets messages destined to itself from the ferry. We simulated this scheme by having a node whose schedule is to go in a square route in the field. No messages are originated nor destined to the ferry. All other nodes, plan their mobility to approach the ferry whenever it is possible, assuming the ferry location is known at all times to all nodes at all times. Data mules [15] (also referred to as FIMF in [21]) are similar to message ferries, except that whenever a node needs to send messages to any other node, it "calls" the data mule. The data mule, having all the node requests, schedules its own mobility to server as many requests as possible. Shah *et al.* [15] propose that mules use a random walk on the field, while Zhao *et al.* [21] show that the mule scheduling problem is NP-hard and propose different heuristics to solve the problem. One of the heuristics they proposed is the nearest neighbor heuristic, in which the mule moves to meet the closest node with a request. We experiment with this heuristic, where data mules have double the speed of normal nodes. In this model, mobility of the normal nodes is not derived by the communication workload, *i.e., nodes do not approach the data mule to speed up the message exchange* as in the case of ferries. Hence, in our experiments, normal nodes follow WAS in order to remain static for most of the time allowing the mule a higher chance to reach them. Whenever any of the nodes change its location, it sends a `location-update` message to the mule so that it can update its mobility accordingly. After encountering a node, the mule selects the closest node with a standing request as its next target, an so on. In this evaluation, we did not limit message exchanges to be done only with a ferry (or a mule), rather encounters between normal nodes could also be used to deliver messages. It should be noted that the flexibility of our scheme enable modelling both ferries and mules by controlling the node schedule and message workload. In case of ferry, it is modelled as a node whose schedule is to go in a predetermined path in the field, with no message arriving or destined to the ferry. Data mule

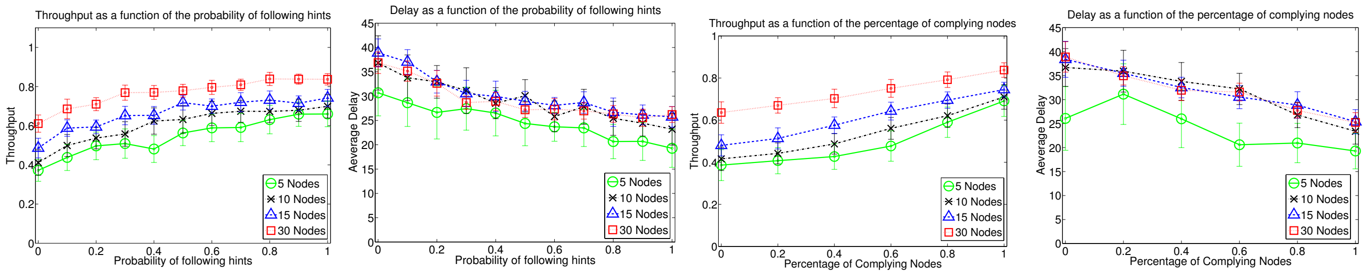

Figure 5. Effect of partially following hints on delivery ratio and delay.

shares with ferries not having any messages arriving or destined to it. However, it differs in that it has an empty schedule which enable it to move anywhere in the field.

We can categorize DMD, DNE, ferries and mules along three dimensions. The first dimension is the distributivity of the solution. While DNE and ferries are distributed approaches, DMD and data mules are both centralized approaches, in the sense that knoweldge about the message workloads and node locations must be aggregated and processed centrally.[5] The second dimension is whether the message workload is used in coordinating node mobility. DMD, data mules, and message ferries are workload-cognizant while DNE is workload-oblivious. In DMD, data mules and message ferries, node mobility is derived, at some point in time, by the knowledge that there is a message that needs to be communicated to some other node in the network. This is not he case in DNE, under which, nodes take mobility decisions motivated by the desire to increase their chances of encountering other nodes, irrespective of the message workload. The third dimension is the dependence on external helper nodes (*i.e.*, some form of "infrastructure"). It is clear that both DMD and DNE do not depend on external helper nodes, while message ferries and data mules do.

Since DMD and DNE work by leveraging the slack that exists in the schedules, in the set of experiemnts where we compare DMD and DNE to ferries and data mules, we vary the amount of slack available to nodes (κ in Equation 11).

Figure 6 shows the performance of the DMD, DNE, ferry, and data mules with 5 and 10 nodes. As we increase the slack in the schedules, the performance of DMD, DNE, and ferry improves. The reason is that both DMD, and DNE explicitly benefit from relaxed schedules since they depend on the available slack. As for ferries, more slack allows nodes more chances to encounter the ferry, since nodes try to move towards the current ferry location, if their slack permits. In the case of data mules, as we mentioned above, nodes do not approach the mule to speed up the exchange, hence, increasing their slack does not improve the performance. It should be noticed that the delivery ratio of data mules is always better than that of ferries (which is consistent with what Zhao *et al.* reported [21]). The reason is that, unlike ferries, data mules have more freedom in terms of their route in the field, as they can change their route based on the current workload. Moreover, data mules move with double the speed of normal nodes making them more effective than ferries in message delivery.

Figure 6 shows that data mules are more effective than DNE, but only for the tightest of schedules. Increasing the slack improves the performance of DNE until it surpasses that of mules. DMD has the best performance for all parametrizations.

[5] Zaho *et al.* [21] propose a distributed approach to implement data mules by allowing a node to use of long-range communication to inform the mule about the requests of service and location updates. We do not argue the practicality of this proposal and just note that the solution is centralized in nature.

6.2 Trace-Driven Evaluation

Following our motivating application, we used cab traces [1] for cabs in the San Francisco area as input to our models. The goal is to show that, with little coordination between cabs, they could function as an effective DTN system.

Methodology: For each cab, the traces show location updates of the cab. This is composed of latitude and longitude of the cab location, the time of the location update, along with the cab status: metered (occupied/hired) or empty (free). We gathered a little bit more than a full day's worth of data for more than 450 cabs. In the traces we collected, some cabs have as many as 400 location updates, while others have as few as 5 updates. We used all location updates for all cabs to construct a "map" of the San Francisco area. We represented the map as an undirected graph $G = (V, E)$. V is the set of all legitimate locations any cab can be in at any time, where a location is defined by its latitude and longitude coordinates. In the data we collected, the total number of locations is 40399, and the number of unique locations, $|V| = 39,103$ locations. To determine the relation between different locations (*i.e.*, the edges, E), we used a threshold-based neighborhood algorithm with a threshold value r_{th}. This means that, for any two locations a, b, such that the distance between them is $Dist(a,b)$, if $Dist(a,b) \leq r_{th}$, then we add an edge between a and b whose cost = $Dist(a,b)$. We used $r_{th} = 200$ meters (≈ 0.12 miles = 656 feet). This value of r_{th} partitioned the unique field locations into different partitions, with the largest partition consisting of 36,368 unique locations. We used this partition as a representative of the map. Finally, out of the 450 cabs, we selected the 50 cabs with highest number of location updates. We mapped the location updates of the cabs to the map we generated, and used the map to "fill" in the gap of the missing location updates for the first 150 minutes. This is done by mapping each two consecutive updates to the map, finding the shortest route between them. Next, we interpolate a number of locations along this route that is equal to the number of minutes between the location updates. This process allows us to infer the location of at one-minute granularities. The cab status for those interpolated locations is set to be its reported status during the last location update.

Based on each cab's mobility profile (obtained as described above), we defined the schedule of the cab as follows: every time the cab is metered, its location is added to the schedule of the cab. This means that, if the cab is occupied (according to the mobility profile), then it has to be in the indicated location at the indicated time. In other words, we can not change the location of an occupied cab. This leaves room for offering hints to the cab only when it is empty.

We compared two mobility synthesis approaches: WAT

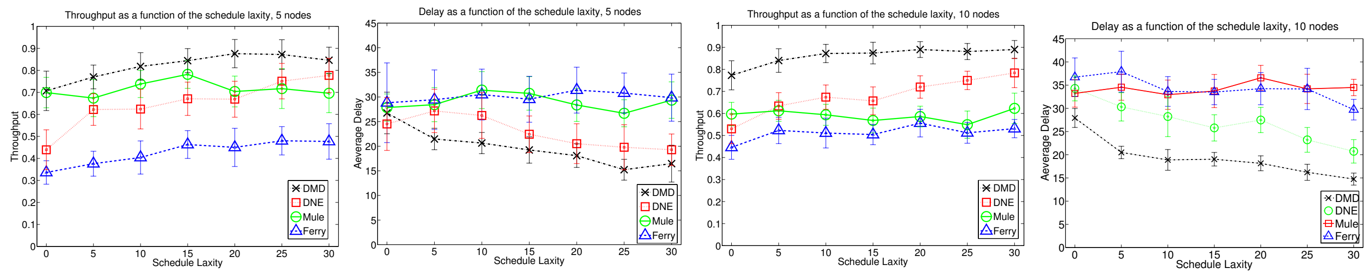

Figure 6. Comparison of DMD, DNE, data mules, and ferries. Number of nodes = 5, 10.

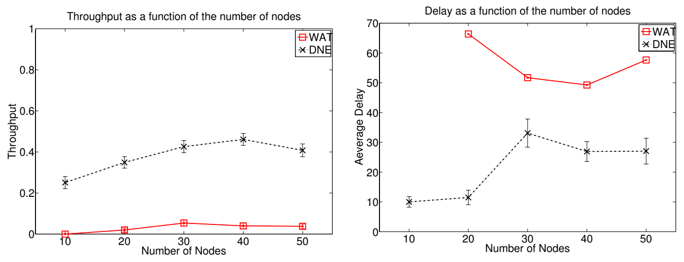

Figure 7. DNE vs WAT using SF cab traces.

and DNE. Under WAT, when empty, a cab moves to the next location where it picks up its next customer, as early as possible, and spends its slack time there waiting for the customer. However, under DNE, we divided the field into 11 big sections and selected some location in each section as the potential meeting location with other cabs. When a cab is empty, it calculates the distance between its current location and all suggested meeting points, and if there is enough time, it moves to the closest location to spend its slack time. When the slack time is over, it moves to the next location where it picks up its next customer. In doing this we assumed a maximum speed of 30 mph, which is quite conservative.

Message Workload Generation: We generated message workloads similar to those used in our synthetic simulations. Specifically, we used a Poisson arrival model with a mean of 0.75 message/minute. Messages sources and destination were randomly selected from the nodes.

Results: Figure 7 give the results of this experiment. DNE yields superior performance compared to WAT in terms of average message delay and delivery ratio. Figure 7 shows that, similar to the simulation-based evaluation, increasing the number of nodes increases both the delivery ratio and the average message latency.

It should be noted that assuming higher maximum speeds could improve the performance of DNE even more. Another important factor is the number of cabs we conducted the study on; increasing this number is bound to improve the delivery ratio.

7 Conclusion

In this paper we argued that many of the processes inducing encounters in DTNs exhibit some flexibility, in that they only define the starting and ending points in any node's waypoint. We then argued that this flexibility could be leveraged to improve message delivery and delays in DTN's. We showed that the resulting problem – the mobility coordination problem (MCP) in DTN – is NP-hard, and proposed two detour-based heuristics to solve it. The first heuristic (DMD) adopts a centralized workload-cognizant approach, whereas the latter heuristic (DNE) adopts a distributed workload-oblivious approach. We showed that our heuristics achieve better performance than basic mobility planning techniques (*e.g.*, WAT, WAS, and RND). We also showed that, when the nodes schedules exhibit enough slack, DMD and DNE are more effective than approaches that depend on external helper nodes (*e.g.*, data mules, and message ferries). Using taxi traces from a major metropolitan area, we confirmed the advantage of DNE over basic models.

In the future, we intend to explore the design space of our heuristics. For example, we will investigate the role of the order in which DMD considers messages – since committing to some encounters *early* in the planning process might lead to missing some later useful encounters, which could have helped deliver more messages. Similarily, in DNE, since the density and the order of proposed encounter points may affect overall performance, we intend to consider better approaches for generating such encounter points.

8 References

[1] Cabspotting. `http://www.cabspotting.org/api`.

[2] Delay Tolerant Networking Reserach Group. http://www.dtnrg.org/wiki/Docs.

[3] Epidemic Routing Bibliograph. http://roland.grc.nasa.gov/w̃eddy/ biblio/epidemic/.

[4] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan. The minimum latency problem. In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 163–171, New York, NY, USA, 1994. ACM Press.

[5] D. Choffnes and F. E. Bustamante. Exploiting emergent behavior for inter-vehicle communication. In *HotAC II: Hot Topics in Autonomic Computing on Hot Topics in Autonomic Computing*, Berkeley, CA, USA, 2007. USENIX Association.

[6] Z. J. Haas, J. Y. Halpern, and L. Li. Gossip-based ad hoc routing. *IEEE/ACM Trans. Netw.*, 14(3):479–491, 2006.

[7] A. Kansal, A. A. Somasundara, D. D. Jea, M. B. Srivastava, and D. Estrin. Intelligent fluid infrastructure for embedded networks. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 111–124, New York, NY, USA, 2004. ACM Press.

[8] J. Leguay, T. Friedman, and V. Conan. Dtn routing in a mobility pattern space. In *WDTN '05: Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 276–283, New York, NY, USA, 2005. ACM Press.

[9] 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.

[10] A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):19–20, 2003.

[11] B. Liu, P. Brass, O. Dousse, P. Nain, and D. Towsley. Mobility improves coverage of sensor networks. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 300–308, New York, NY, USA, 2005. ACM.

[12] C. Liu and J. Wu. Scalable routing in delay tolerant networks. In *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 51–60, New York, NY, USA, 2007. ACM.

[13] S.-D. N. P. M. G. M. Island hopping: Efficient mobility-assisted forwarding in partitioned networks. *Sensor and Ad Hoc Communications and Networks, 2006. SECON '06. 2006 3rd Annual IEEE Communications Society on*, 1:226–235, 28-28 Sept. 2006.

[14] R. P. S. Jain, K. Fall. Routing in delay tolerant networks. In *ACM SIGCOMM*, 2003.

[15] R. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: Modeling a three-tier architecture for sparse sensor networks. In *IEEE SNPA Workshop*, May 2003.

[16] M. M. B. Tariq, M. Ammar, and E. Zegura. Message ferry route design for sparse ad hoc networks with mobile nodes. In *MobiHoc '06: Proceedings of the seventh ACM international symposium on Mobile ad hoc networking and computing*, pages 37–48, New York, NY, USA, 2006. ACM Press.

[17] W. G. C. G. L. P. T.F. Movement-assisted sensor deployment. *Transactions on Mobile Computing*, 5(6):640–652, June 2006.

[18] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Technical Report CS-2000-06, Duke University, July 2000.

[19] W. Wang, V. Srinivasan, and K.-C. Chua. Using mobile relays to prolong the lifetime of wireless sensor networks. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 270–283, New York, NY, USA, 2005. ACM Press.

[20] H. Wu, R. Fujimoto, R. Guensler, and M. Hunter. Mddv: a mobility-centric data dissemination algorithm for vehicular networks. In *VANET '04: Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, pages 47–56, New York, NY, USA, 2004. ACM.

[21] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 187–198, New York, NY, USA, 2004. ACM Press.

Appendix: PEG Construction

We denote the Potential Encounter Graph, PEG = (V,E), where $V = V_1 \cup V_2$, and $E = E_1 \cup E_2$. For every vertex $v \in V_1$, $\gamma(v)$ gives the id of the node represented by this vertex. Any vertex $v \in V_2$ is a tuple of the form $(\nu_1, \nu_2, \lambda, \omega)$ such that ν_1 and ν_2 are the ids of the two nodes having the encounter, λ is the location of the encounter, and ω is the earliest time at which the encounter can take place. For any encounter $v \in V_2$, $\gamma_1(v)$ and $\gamma_2(v)$ give the ids of the two nodes in the encounter, respectively, $\theta(v)$ is the earliest time this encounter can take place, and $\ell(v)$ is the location of the encounter. For any edge $e \in E$, the functions $\gamma_1(e)$, $\gamma_2(e)$, and $C(e)$ give the source, destination, and cost of the edge, respectively.

```
ALGORITHM 1. constructPEG()
Input: Schedule $S = \bigcup_{i=1:n} s_i$
Output: PEG = (V , E)
1. Set $V_1 = \{1, 2, \dots, n\}$.
2. P = calculate-potential-encounter-points(S);
2. Initialize $V_2 = \phi$, $E_1 = \phi$. c = 0;
3. For every $p \in P$
    3.1 add $\{p\}$ to $V_2$ as follows: Set $c = c + 1$; Let $v_c = p$, $V_2 = V_2 \cup \{v_c\}$
    3.2 Add horizontal edges to $E_1$ linking $v_c$ to the 2 nodes in that encounter.
4. To put the vertical edges, initialize $E_2 = \phi$.
5. For encounters $v_1$ and $v_2$ with common node i, let $v_1$ be the earlier encounter.
    5.1 if (can-do-both-encounters($i, v_1, v_2$) == true), then
        5.1.1 Add a vertical edge e to $E_2$, such that e connects $v_1$ to $v_2$
        5.1.2 Let $C(e) = \theta(v_2) - \theta(v_1)$.
6. set $V = V_1 \cup V_2$, and $E = E_1 \cup E_2$.
```

In Algorithm 1, steps 1-3 construct V_1, V_2 and add the horizontal edges to E_1, while steps 4 and 5 add vertical edges to E_2. Function *calculate-potential-encounter-points*(S) returns a list of the points where nodes might encounter each other, along with the time of such potential encounters.

```
ALGORITHM 2. calculate-potential-encounter-points(S)
Input: Schedule $S = \bigcup_{i=1:n} s_i$
Output: List of encounter points $P = \{p_1, p_2, \dots\}$, $p_i = (\nu_1, \nu_2, \lambda, \omega)$
1. For every two nodes i, j such that $i \neq j$.
    1.1 For every waypoint a ($1 \leq a \leq L(s_i)$) in i's schedule, mark each field location (loc) with $er_i(loc)$ and $lat_i(loc)$: the earliest and latest times i could be at loc during this waypoint.
    1.2 Do the same for every waypoint b ($1 \leq b \leq L(s_j)$) in j's schedule to get $er_j(loc)$, and $lat_j(loc)$.
    1.3 Let $\zeta$ be all field locations at which nodes i and j could meet at waypoints a and b. Associate with each point $z \in \zeta$ the earliest time $\theta(z)$ at which an encounter can take place at this location.
    1.4 if $\zeta$ is not empty, let $p = \{v = (i, j, \lambda, \omega) : v \in \zeta; \omega \leq \theta(z), \forall z \in \zeta\}$. i.e., $\lambda$ is the meeting location at which the earliest encounter could take place (i.e., at time $\omega$) during waypoints a and b.
    1.5 Let $P = P \cup \{p\}$.
```

Function *calculate-potential-encounter-points* depends on marking field locations with the earliest and latest times a node could be at this location during any given waypoint. To see what this means, consider the first waypoint of n_1 (from Table 1). The source of this waypoint is location 8 at time 1, and its destination is location 13 at time 20. Notice that Figure 2 (left) reveals that node n_1 can only visit locations 1 through 20 in the field (the span of the first rectangle of n_1 on the x-axis), which means that locations 21 through 60 are unreachable during this waypoint, hence they will be marked as so. For locations 1 to 20, the earliest time n_1 could be at these locations is given by the lower two sides of the rectangle representing this waypoint. While the latest times to visit these locations corresponds to the upper two sides of the same rectangle. Figure 8 shows the marking of corresponding field locations during this waypoint. By intersecting different such tables of different waypoints (*i.e.,* rectangles in Figure 2 left) of different nodes, we are able to determine the locations and times at which different nodes could encounter each other.

Finally, function *can-do-both-encounters*(i, v_1, v_2) returns a boolean answer indicating whether or not a given node i, whose schedule is s_i, can have two meetings at the two given meeting points.

```
ALGORITHM 3. can-do-both-encounters
Input: Node i with schedule $s_i$, first and second encounters $v_1$ and $v_2$ such that $\theta(v_1) \leq \theta(v_2)$
Output: True if node can be at both meeting points, false otherwise.
1. Let $wp_1$ be the waypoint in $s_i$ such that $\theta(v_1) \geq$ starting time of $wp_1$ and $\theta(v_1) \leq$ end time of $wp_1$. Similarly let $wp_2$ be the waypoint where $\theta(v_2)$ falls.
2. If $wp_1 \neq wp_2$ return true, otherwise go to step 3.
3. Let dist = Dist($\ell(v_1)$,$\ell(v_2)$). and $\delta = \theta(v_2) - \theta(v_1)$.
4. If $\delta \times v_{max} \geq$ dist return true, else return false.
```

The reason Algorithm 3 returns `true` in line 2 is that, any two meeting points in different waypoints, could be trivially met by a node, given that each meeting point fits the original node schedule. In other words, two different meeting points laying in two different rectangles in Figure 2 could be met by the node irrespective of the actual distance and time interval between them (given that they actually **lay on or inside** two different rectangles). Otherwise, if the two meeting points fall within one rectangle, we need to check that distance between them is less than the maximum distance the node can cover during the time interval between the two meeting points. Notice that we call function *can-do-both-encounters* with two meeting points such that we already know that each meeting point fits the original schedule (Algorithm 1, line 5.1).

Field Location	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21-60
Earliest Time	8	7	6	5	4	3	2	1	2	3	4	5	6	7	8	9	10	11	12	13	UR
Latest Time	8	9	10	11	12	13	14	15	16	17	18	19	20	19	18	17	16	15	14	13	UR

Figure 8. Earliest and latest times that node n_1 could visit various locations in its first waypoint (start (τ_{11} = 1, l_{11} = 8), end (τ_{12} = 20, l_{12} = 13)). UR stands for "unreachable"