Optimal Encounter-Based Routing of Queries for Mobile Objects

Zhiyu Wang, Mario A. Nascimento and Mike H. MacGregor
Department of Computing Sciences
University of Alberta, Canada
{zhiyu1, mn, macg}@cs.ualberta.ca

***Abstract*—Mobile objects can be equipped with sensors enabling them to collect data, as well as answer queries remotely and in real-time. For that, one needs to be able to effectively route queries from a base station to the queried object in an efficient way, i.e., with minimum energy-cost and/or minimum delay. Complicating factors in many domains are that the objects in the network are mobile and that they may not form a single connected component at all times. In this paper we take advantage of periodically repeated encounters, where an encounter is defined as a time-period long enough so that sensors can communicate with each other. Possessing such encounter patterns we show how to model routing as an optimization problem in a graph with domain-oriented constraints. Further, we propose polynomial time algorithms to find the optimal and near-optimal routes to deliver the query from a source to one or more query object(s). We also show that the proposed model is flexible enough to support different assumptions about the nature of the encounters. Experimental results, using both real and synthetic datasets, show that our proposed approaches always find better routes with regard to delivery delay and energy cost comparing to other state-of-art protocols.**

***Index Terms*—Mobility, Delay Tolerant Network, Regularity, Periodicity, Routing**

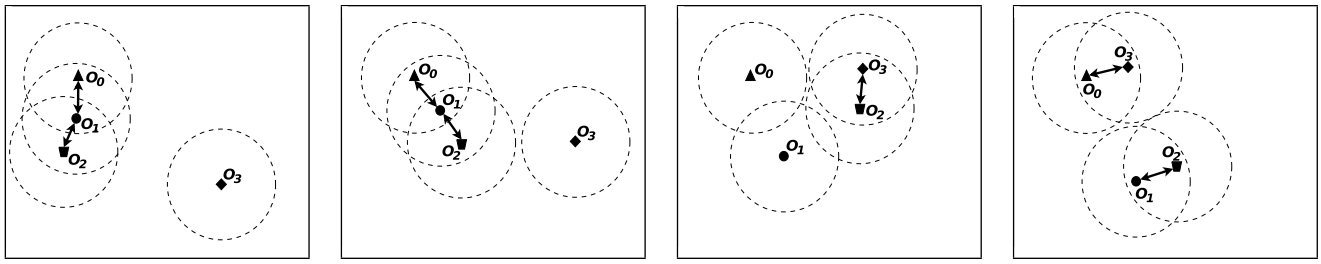
I. Introduction

New applications for wireless sensor networks have created demand for novel techniques to effectively and efficiently disseminate queries and collect data from sensors. Conventional wireless sensor networks are usually assumed to be connected as a single component at all times. In those networks, logical structures such as trees and paths, built on top of the physical topology, can be used to find routes between data sources and sinks. However, in some application domains the physical topology may change often and objects (or entire sub-networks) may not always be connected. For instance, mobile robots may circulate over wide areas and given typical short wireless ranges, may often be disconnected. In another example consider monitoring animals in a farm or even in a determined area in the wilderness. The monitored animals (thus objects) often roam around separately in physical clusters that are disconnected from a central backbone infrastructure.

Because constructing logical structures on top of unstable topologies is not efficient, routing protocols from mobile ad-hoc networks are adopted to handle communications in mobile wireless sensor networks under infrastructure-less environments. In mobile ad-hoc networks, routing protocols such as AODV [33] and OLSR [19] can be used to periodically discover new routes in order to establish communication. These route-discovery protocols lead to overhead and add delay to the communication task. A key constraint for sensor networks is energy. Because each sensor has limited battery capacity [2], the communication overhead of route discovery is a severe drawback. The other constraint is query delay. While some queries may be time-tolerant, others may have strict deadlines. Finding routes with the minimum delay or with the minimum energy consumption are thus both critical in mobile wireless sensor networks. Directly applying protocols from ad-hoc networks does not meet the specific requirements of mobile wireless sensor networks. Thus, new techniques are needed for the case of mobile wireless sensor networks where disconnected components are the norm rather than the exception.

In many domains, mobile objects often have relatively periodic patterns of movement. They tend to have a repeated, stable sequence of activities because of their social and/or controlled behaviours. For example, mobile robots in an automated factory have relatively invariant mobility traces because they have predefined schedules and areas to work. Farm animals and animals in the wild also exhibit mobility behaviours in terms of paths followed, places to rest, feed, etc. One outcome from these repeated activities is that each mobile object may encounter a set of other objects periodically, for example at approximately the same time every day. We call these sequences of periodic encounters *encounter patterns*. Hence, after some learning/training time each sensor can obtain knowledge about which objects will be encountered in the future and one can use this information to compute routes for network packets that minimize energy and/or delay.

In order to illustrate the benefits of encounter patterns, an example is presented in Figure 1 where we have four objects. In this example, object o_0 is a stationary base station, while the other three are mobile objects. Encounters, represented by directed lines, occur during four consecutive time intervals. In the example, at the beginning of interval I_0, object o_0 wants to send a query to object o_3. If routing protocols from mobile ad-hoc networks start route exploration immediately at interval I_0, no route will be found until interval I_3. Until then the routing discovery process is simply a waste of energy. In contrast to those protocols, and assuming one knows the encounter pattern for this scenario, two routes can be found to disseminate the query from o_0 to o_3. The first route is that object o_0 holds



(a) Snapshot at interval I_0 (b) Snapshot at interval I_1 (c) Snapshot at interval I_2 (d) Snapshot at interval I_3

Fig. 1. Encounters during four consecutive intervals

the query and sends it to destination o_3 during interval I_3. The second route is that object o_0 transmits the query to object o_1, and object o_1 carries the query and disseminates it to o_2. The query is eventually delivered to object o_3 by object o_2 during interval I_2. The first route saves energy, but is not the most efficient in terms of delay, whereas the second route has less delay but consumes more energy.

In this paper, we address the problem of guaranteed query delivery in a likely-disconnected mobile network. Towards this goal, we introduce a graph model which integrates all encounter patterns under the assumption that they have a periodic behaviour. Using the graph model, both the minimum delay and minimum energy consumption problems can be modelled as optimization problems. The only change between the graphs used to solve either problem is in the assignment of weights for the edges. The minimum delay problem can be modelled as the shortest path tree problem whereas the minimum energy consumption problem can be modelled as the minimum weighted Steiner Tree problem. With a modification, we can use Dijkstra's shortest path algorithm [9] to find an optimal solution for the minimum delay problem. However, because the minimum weighted Steiner Tree problem is NP-hard, we propose a polynomial time approximation algorithm to find a multicast tree for routing. The experiments show that our algorithm can quickly find sub-optimal routes whose energy cost is typically 2-3 orders of magnitude smaller than the cost of state-of-art solutions in delay tolerant networks [12].

The remainder of this paper is organized as follows. An overview of existing techniques to handle queries in mobile wireless sensor networks is discussed in Section II. We introduce some preliminary terminology and background for our graph models in Section III. In Section IV, we describe our graph model and its domain constraints. Mathematical optimization and optimal algorithms that capture the semantics of the discovered encounter patterns are presented in Section V. Experimental results shown in Section VI present the performance of our algorithms regarding both unicast and multicast routing. Finally, conclusions and suggestions for future work are given in Section VII.

II. Related work

Research in mobile ad-hoc networks has been very active in the past several years. Many routing protocols have been developed to provide end-to-end, *i.e.,* unicasting, communication under the assumption that there exists at least one route connecting any pair of objects in the network [19, 21, 33]. However, this assumption is not always true: object movements do impact the connectivity of the network [24], As a result, such protocols can no longer guarantee that a route will be found, even though one may exist. For instance, considering Figure 1, if the query originator was object o_1 instead of o_0 and the destination was o_3, AODV would simply not find a route. This is because during no given interval there is a complete route between o_1 and o_3. Nonetheless one could clearly route the query from o_1 to o_2 at interval I_1 and then from o_2 to o_3 at interval I_2 by taking advantage of the encounters between the objects.

Recently, delay tolerant networks have been introduced to address the challenge of sometimes-disconnected networks. As noted earlier, end-to-end communication can be achieved by storing data and then forwarding it later. To fully utilize opportunistic encounters without prior knowledge, epidemic routing protocol was first introduced to provide promising delivery by flooding the message into the network [43]. Because of a large number of replications and retransmissions in the network, this protocol is inefficient regarding network resources such as bandwidth and storage. In contrast to epidemic routing, in direct delivery [38] the source delivers the message if and only if it encounters the destination where minimal storage and energy usages are guaranteed. However, a low delivery ratio is always expected because the probability of directly meeting the destination could be low. Epidemic and direct delivery routing protocols set upper and lower bounds on resource usage and delivery ratio in delay tolerant networks.

With respect to inter-contact times and reachability of nodes, previous work has been shown that nodes mobility and pair-wise encounters can benefit communications in disconnected networks [4, 40, 46]. To strike a balance between epidemic and direct delivery protocols, nodes mobility and pair-wise encounters have been used to improve routing in delay tolerant networks [3, 11, 27, 35, 41]. Researchers focused on developing or improving routing protocols to include mobility statistics to achieve better performances. For example, in [8] the authors improved epidemic routing by exploiting object relative positions through their movements. A Mobile Relay Protocol was proposed in [31] to handle end-to-end communication through object mobility. In addition, [26] presented a novel approach for communication in disconnected

networks by modifying object trajectories to deliver messages. Beside utilizing nodes mobility, there are other protocols that also take advantage of object's social relationships to divert communications [6, 7, 15, 16]. New architectures are also proposed to model communications in disconnected networks. For example, [20] presented a three-tier architecture by using MULEs to disseminate queries and collect results between sensors and access points. Instead of using MULEs, special mobile objects called message ferries with scheduled movements were introduced to improve data delivery in [47].

The previous work shows that object movements can be exploited to create routing protocols for disconnected networks. However, they fall short in two ways: (1) they do not present a formal model that is flexible enough to reflect different assumptions and goals, and more importantly, (2) they do not provide an algorithmic approach to solve the problem optimally. This paper fills both gaps. We propose a flexible graph-based model to solve the minimum delay and minimum energy problems, and we present algorithms to solve both problems in polynomial time not only for the cases of unicasting, but also for multicasting.

III. Preliminary definitions

Given a set of mobile objects $O = \{o_0, o_1, o_2, ...o_{n-1}\}$, we assume that a set of *periodic encounter patterns* with period length m is given. We have developed off-line techniques to find these patterns, which are reported elsewhere [44]. The encounter patterns are considered given information for the purposes of this paper.

Definition 1: A periodic encounter pattern with period length m for a pair of objects $\{x,y\}$ is a binary time series $P^m_{x,y}$ where:

$$P^m_{x,y} = \{p_t\}, t = 0, 1, \cdots, (m-1) \text{ and}$$

$$p_t = \begin{cases} 1 & \text{iff there is a periodic encounter during phase } t \\ 0 & \text{otherwise} \end{cases}$$

The duration of each phase, and thus of each entry p_t in the time series, is τ.

The period length m of the encounter pattern indicates the total time over which the pattern repeats itself. For example, if the duration of each phase is one day and $m = 21$, the encounter pattern represents the regular interactions between objects x and y over a three-week period. The phase t indicates the interval within the period where regular encounters occur. For example, continuing the previous example, if $p_6 = 1$ then objects x and y encounter each other regularly on the first Sunday in the three-week period, assuming p_0 occurs on a Monday.

During an encounter between two objects, and at no other time, messages can be exchanged between the two encountering objects. For instance, in Figure 1(a), o_0 encounters o_1 during phase I_0. Furthermore, mobile objects can relay packets among themselves. For instance, as illustrated in Figure 1, a message can be routed from o_0 to o_2 through o_1 at the cost of two transmissions.

Using encounter patterns, we can solve a variety of problems in routing a message from a source to one destination (unicast), or from a source to a set of destinations (multicast), such that either the energy cost or the time delay for message delivery is minimized.

Definition 2: The *encounter matrix* summarizes all periodic encounters for a given object. We define the encounter matrix $M^k = [m^k_{jt}]$ for mobile object o_k as:

$$m^k_{jt} = \begin{cases} 1 & \text{iff } p_t = 1 \in P^m_{k,j} \\ 0 & \text{otherwise} \end{cases}$$

where $0 \le k \le (n-1),\ k \ne j$ and $0 \le t \le (m-1)$.
We note that the number of objects encountered by any given object, and thus the number of rows in M^k, is expected to be much smaller than n. The encounter matrices for the four objects in Figure 1 are given in Tables I-IV.

TABLE I
Encounter matrix for object O_0.

	p_0	p_1	p_2	p_3
O_1	1	1	0	0
O_2	0	0	0	0
O_3	0	0	0	1

TABLE II
Encounter matrix for object O_1.

	p_0	p_1	p_2	p_3
O_0	1	1	0	0
O_2	1	1	0	1
O_3	0	0	0	0

TABLE III
Encounter matrix for object O_2.

	p_0	p_1	p_2	p_3
O_0	0	0	0	0
O_1	1	1	0	1
O_3	0	0	1	0

TABLE IV
Encounter matrix for object O_3.

	p_0	p_1	p_2	p_3
O_0	0	0	0	1
O_1	0	0	0	0
O_2	0	0	1	0

IV. The encounter graph

If we combine the encounter matrices for all the mobile objects in the network, we can create an *encounter graph* that represents all the observed periodic patterns for a given set of objects. For example, Figure 2 gives the encounter graph corresponding to the behaviours shown in Figure 1.

There are four objects whose encounters are represented in the graph. The objects are each represented by a different shape (triangle, circle, trapezoid and diamond). There is an implicit time axis in this graph, from left to right. The four different solid-line instances of each shape represent the same object during four successive time intervals. That is, the left-hand solid triangle represents object o_0 during period $t = 0$,

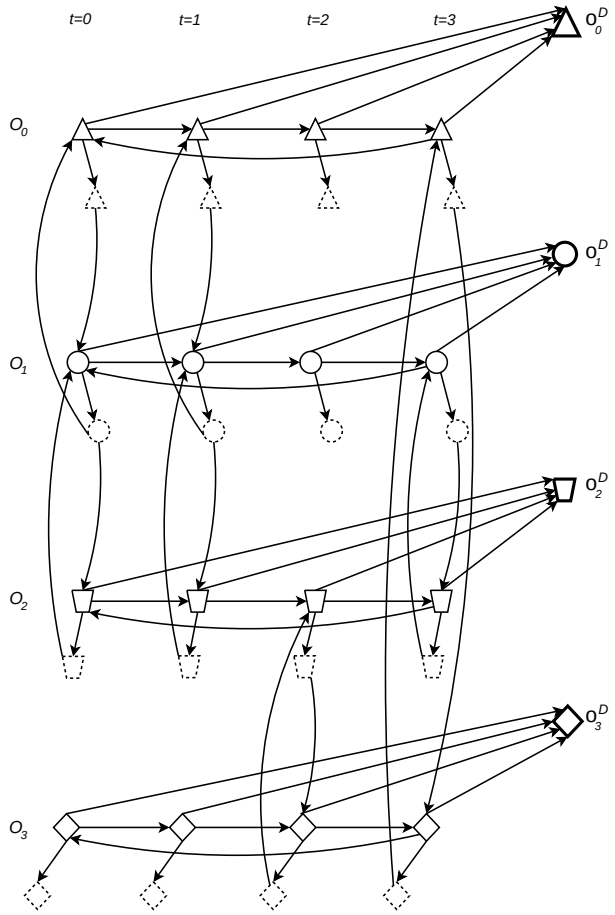


Fig. 2. A graph model reflecting the encounter patterns in Figure 1

and the right-hand solid triangle represents o_0 during period $t = 3$.

The horizontal left-to-right directed edge from one solid-line shape to the next instance of the same solid-line shape to the right represents the possibility for an object to buffer a message and carry it forward to successive instants in time. The final edge, from the furthest right-hand instance of a solid-line shape back to the furthest left version of the same solid-line shape represents the possibility for a message to be carried forward in time to the next repetition of the observed behaviours.

In this example, all encounter patterns have a period length of $m = 4$, so the graph has four instances of each solid-line shape. If we had patterns with different period lengths, the graph would have a width equal to the least-common multiple of the periods.

There are four solid-line instances of each shape, and four dotted-line instances. They enable us to differentiate between the object in receiving mode (solid), and the same object in transmitting mode (dotted). For example, the possibility of a transmission from o_0 to o_1 during $t = 1$ is represented by the directed vertical arc from the dotted triangle in interval $t = 1$ to the solid circle in the same interval. Messages can only be transmitted from one object to another during the same interval. Thus, transmission arcs cannot cross intervals. They may, however, be directed vertically either up or down, depending simply on the relative positions where two objects have been positioned in the graph. The vertical arcs represent the encounters in the binary time series; it is these encounters that make message transmission possible.

Lastly, there is a single "darker solid" lined version of each shape. This is a convenient representation of each object as a potential destination vertex in the graph model. For example, if our goal is to find the minimum delay path for a message from o_0 to o_2, one possibility is to send it from o_0 to o_1 during $t = 1$, and then from o_1 to o_2 during $t = 2$. This route could be found algorithmically by asking for the shortest path (under a few constraints) from o_0 at $t = 0$ to o_2^D, *i.e.,* to the destination version of o_2.

In summary, we use three types of vertices in our graph model: receiving vertices V^R, transmitting vertices V^T and destination vertices V^D. The edges used to connect these vertices can be grouped into four categories:

- Horizontal edges connect vertices with the same shape to represent the possibility for an object to buffer and carry a message from one phase to the next. We denote these edges as $e^H_{t,o}$. The first subscript gives the phase number and the second subscript gives the object number.
- Internal edges connect an object's receiver to its transmitter. The weight on an internal edge can be used, for example, to represent the energy cost of transmission. These edges are denoted $e^I_{t,o}$.
- Vertical edges link two objects in the same phase, and represent an encounter between these two objects. They are denoted $e^V_{t,o1,o2}$, where $o1$ and $o2$ are the indices of the two objects.
- Destination edges link receivers to their destination vertex to capture the fact the message has been delivered to its destination. They are denoted $e^D_{t,o}$.

We now use this terminology to describe the situation where an object can deliver a message to a set of destinations. Here, we assume that each object at a particular phase either broadcasts the message to all encountered neighbours or does not transmit at all. Using the encounter matrices $M^k, 0 \le k \le n-1$, the encounter graph is the weighted directed graph $G = (V, E, W)$ where:

- $V^R = \{v^R_{t,i} \mid 0 \le i \le n-1,\ 0 \le t \le m-1\} \in V$ denotes the receiving modules for a set of n objects during a period of length m phases,
- $V^T = \{v^T_{t,i} \mid 0 \le i \le n-1,\ 0 \le t \le m-1\} \in V$ denotes the transmitting modules for a set of n objects during a period of length m phases,
- $V^D = \{v^D_i \mid 0 \le i \le n-1\} \in V$ denotes destination vertices for a set of n objects,
- $E^H = \{e^H_{t,i} = (v^R_{t,i}, v^R_{t+1,i})\ \forall o_i \in O,\ 0 \le t \le m-1\} \in E$ denotes the horizontal edges,
- $E^V = \{e^V_{t,i,j} = (v^T_{t,i}, v^R_{t,j})\ \forall o_i, o_j \in O,\ 0 \le t \le m-1\} \in E$ if and only if $m^i_{jt} = 1\}$ denotes the vertical edges,
- $E^I = \{e^I_{t,i} = (v^R_{t,i}, v^T_{t,i})\ \forall o_i \in O,\ 0 \le t \le m-1\} \in E$ denotes

the internal edges,
- $E^D = \{e^D_{t,i} = (v^R_{t,i}, v^D_i) \ \forall o_i \in O, \ 0 \leq t \leq m-1\} \in E$ denotes the destination edges and
- W represents a function on an edge returning the weight assigned to it.

This graph is generic enough to be used to represent both energy and delay costs simply by adjusting the weights W assigned to each edge in G. We discuss this next.

A. Communication delay

The encounter graph can be used to solve the minimum delay problem as follows. If an object decides to store the

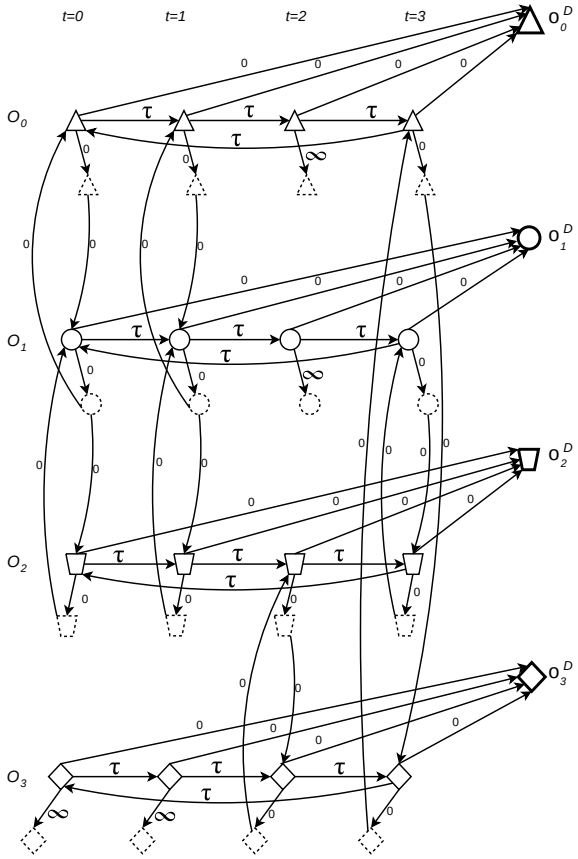


Fig. 3. Graph model for the minimum delay problem

message and carry it on to the next phase to disseminate it in the future, this yields a communication delay equal to τ, i.e., the duration of the current phase p_t. Transmission preparation (the weight for internal edges) yields a small amount of delay between the receiving module and the transmitting module, and signal propagation (the weight for vertical edges) also introduces another delay. However, we can treat both delays as zero because both are embedded within the weight of the horizontal edges τ, which is the duration of each phase. In addition, the weight of destination edges is always zero because they are artificial edges which do not add any delay in the physical environment. Therefore, the weights of the edges for minimum delay routing are (see Figure 3):

- $\forall e^H \in E^H, \ W(e^H) = \tau$.
- $\forall e^V \in E^V, \ W(e^V) = 0$.
- $\forall e^I \in E^I, \ W(e^I) = 0$.
- $\forall e^D \in E^D, \ W(e^D) = 0$.

B. Energy consumption

The encounter graph can also be used to find the minimum energy consumption for routing a message. If an object decides

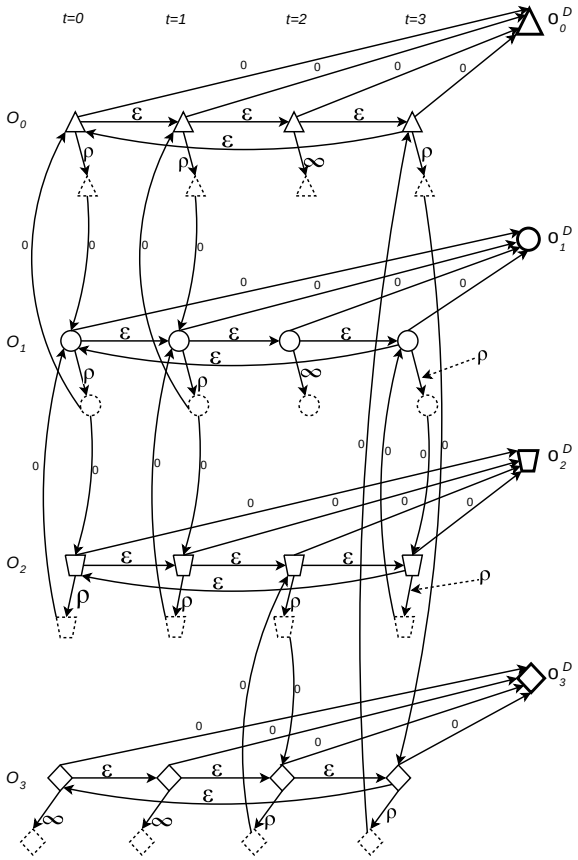


Fig. 4. Graph model for the minimum energy consumption problem

to carry the message on to the next phase and disseminate it in the future, this requires a very small amount of energy for data storage during the current phase p_t. In this paper, we let ε be the energy cost for an object to store data during a phase with duration τ. If an object carries the message on to the next phase, this results in an energy cost, ε, to store the message. In contrast, if the object decides to transmit the message immediately to a set of encountered objects during the current phase, a certain amount of energy, depending on the application, is required. In this paper, we assume homogeneous networks where each transmission costs the same amount of energy, ρ. However, our model can be used to model the case where the energy required to transmit a message between two objects depend on when and where they meet. For instance, in some encounters the objects can be closer or further apart. Formally, to model energy costs for routing, the weights of

horizontal and vertical edges can be defined as follows (see Figure 4):

- $\forall e^H \in E^H,\ W(e^H) = \varepsilon$.
- $\forall e^V \in E^V,\ W(e^v) = 0$.
- $\forall e^I \in E^I,\ W(e^I) = \rho$.
- $\forall e^D \in E^D,\ W(e^D) = 0$.

C. Domain Constraints

In some scenarios, an intermediate object within two consecutive encounters during the same phase may participate in the optimal route. For example, if object o_0 has a query for object o_2 during phase p_0, the route with the minimum delay in Figure 2(a) is $v^R_{0,0} \rightarrow v^T_{0,0} \rightarrow v^R_{1,0} \rightarrow v^T_{1,0} \rightarrow v^R_{2,0} \rightarrow v^D_2$ with zero delay if the propagation delay and processing delay are negligible. In this route, the intermediate object o_1 acts as a bridge for two consecutive transmissions during the same phase. However, because the encounters between object o_1 and objects o_0 and o_2 both take place during the same phase and the order of encounters is unknown in the graph model, we forbid such type of retransmissions during the same phase. Therefore, we propose a domain-oriented constraint called NOT (NO reTransmissions during the same phase in the graph). However, if a retransmission is required for an object during a particular phase, *e.g.,* object o_1 at phase p_0 with a transmission from object o_0 during the same phase, it has to wait for the same phase during the next repetition of the periodic encounters. As a result, each retransmission costs an additional delay that equals the total period of time of such a periodic pattern. Because of this domain-oriented constraint, conventional shortest path algorithms cannot be used to find optimal routes satisfying the constraint. In the next section, we discuss mathematical optimization and modifications of Dijkstra's algorithm [9] that enforce the NOT constraint in their results for both the minimum delay and minimum energy consumption problems.

V. Mathematical optimization and Algorithms

One of basic functions of a network is to distribute information to various groups of destinations with the objective to minimize the usage of resources including overall energy consumption or/and communications delay. Given a set of destinations, our objective is to distribute the query to them using the minimum delay or the minimum energy consumption. We need to find a tree structure rooted at the source that connects desired destinations in the graph. The tree in our graph model is a *Steiner Tree*. Our objective is to find the minimum weighted Steiner Tree that connects the source to all destinations [28, 29].

In this section, we introduce two approaches, binary integer programming and optimal/approximate algorithms, to find routes for communications in our graph model. We discuss the multicast version of the problem as the unicast is a special case of multicast.

A. Binary integer programming

Mathematical programming is a powerful tool that can be used to solve optimization problems. Targeting both the minimum energy consumption and minimum delay problems by assigning different weights on our graph model, the optimal route is a set of sequential edges where the summation of weights on those edges is the minimum. If we treat all edges as variables, an edge either does participate in an optimal route (its value is 1) or does not (its value is 0). This general approach is called the binary integer programming. In this section, we define the formulation to find a minimum weighted Steiner Tree that satisfies our constraints in the graph model.

1) The minimum delay problem: In the minimum energy consumption problem, our goal is to find the Steiner Tree whose total weight is the minimum. This is because the energy cost on the path from the source to a destination is isolated from other paths. The total weight of a tree is the cumulative cost of all these paths. However, this strategy is incorrect for the minimum delay problem where there are multiple destinations in a query. The reason is that the time to deliver a query to one destination may partially overlap the time to deliver the query to other destinations. For example, as shown in Figure 2 a), if object o_2 has a query for objects o_0 and o_3 at the beginning of phase p_0, the minimum delay will be 2τ instead of 3τ because the delay (τ) to deliver the query from o_2 to o_0 is embedded within the delay (2τ) of path from o_2 to o_3. Therefore, when we count the minimum delay, it is incorrect to simply add delays from all paths. This special phenomenon cannot be modelled by linear programming, but we can solve this problem algorithmically; this is discussed in detail in the next subsection.

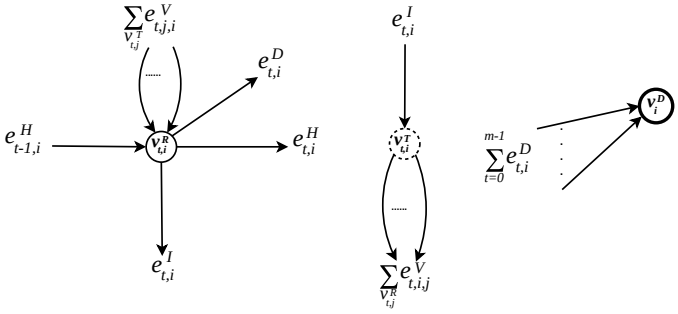


Fig. 5. Illustrations for edge notations

2) The minimum energy consumption problem: In this binary integer program, each edge in the graph has a weight. Each edge either participates in the minimum weighted Steiner Tree or it does not. Therefore, if we set each edge as a variable, this variable has value either one or zero. Because each edge also has a weight, our goal is to minimize the total weight of the Steiner tree being used in order to connect all desired destinations. To define the formulation of all edges, we first have a look at different types of vertices. As we described before, we have three types of vertices in our graph model. Receiving and processing vertices have five

$$y = \sum_{e \in E} w(e) \cdot e$$

subject to

$$e = \begin{cases} 1 & \text{if used in the minimum Steiner tree} \\ 0 & \text{otherwise} \end{cases} \quad (C_1)$$

$$\sum_{t=0}^{m-1} e^D_{t,i} = \begin{cases} 1 & \text{if } v^D_i \in D \\ 0 & \text{otherwise} \end{cases} \quad (C_2)$$

$$\sum_{v^T_{t,j}} e^V_{t,j,i} + e^H_{t-1,i} + e^D_{t,i} = 0 \text{ where } v^R_{t,i} = v^R_{st,s} \quad (C_3)$$

$$e^I_{t,i} + e^H_{t,i} \geq 1 \text{ where } v^R_{t,i} = v^R_{st,s} \quad (C_4)$$

$$\sum_{v^T_{t,j}} e^V_{t,j,i} \leq 1 \text{ where } v^R_{t,i} \neq v^R_{st,s} \text{ and } \forall o_i, o_j \in O \text{ and } 0 \leq t \leq m-1 \quad (C_5)$$

$$\sum_{v^T_{t,j}} e^V_{t,j,i} + e^H_{t-1,i} \leq e^I_{t,i} + e^H_{t,i} + e^D_{t,i} \text{ where } v^R_{t,i} \neq v^R_{st,s} \text{ and } \forall o_i, o_j \in O \text{ and } 0 \leq t \leq m-1 \quad (C_6)$$

$$e^I_{t,i} \leq e^H_{t-1,i} \text{ where } v^R_{t,i} \neq v^R_{st,s} \text{ and } \forall o_i, o_j \in O \text{ and } 0 \leq t \leq m-1 \quad (C_7)$$

$$\sum_{v^T_{t,j}} e^V_{t,j,i} + e^H_{t-1,i} \geq e^H_{t,i} \text{ where } v^R_{t,i} \neq v^R_{st,s} \text{ and } \forall o_i, o_j \in O \text{ and } 0 \leq t \leq m-1 \quad (C_8)$$

$$\sum_{v^T_{t,j}} e^V_{t,j,i} \geq e^D_{t,i} \text{ where } v^R_{t,i} \neq v^R_{st,s} \text{ and } \forall o_i, o_j \in O \text{ and } 0 \leq t \leq m-1 \quad (C_9)$$

$$\sum_{v^T_{t,j}} e^V_{t,j,i} + e^H_{t-1,i} \leq e^I_{t,i} + 1 \text{ where } v^R_{t,i} \neq v^R_{st,s} \text{ and } \forall o_i, o_j \in O \text{ and } 0 \leq t \leq m-1 \quad (C_{10})$$

$$\sum_{v^T_{t,j}} e^V_{t,j,i} + e^H_{t-1,i} \leq e^H_{t,i} + 1 \text{ where } v^R_{t,i} \neq v^R_{st,s} \text{ and } \forall o_i, o_j \in O \text{ and } 0 \leq t \leq m-1 \quad (C_{11})$$

$$e^I_{t,i} \leq \sum_{v^R_{t,j}} e^V_{t,i,j} \text{ where } \forall o_i, o_j \in O \text{ and } 0 \leq t \leq m-1 \quad (C_{12})$$

$$e^I_{t,i} \geq e^V_{t,i,j} \text{ where } \forall o_i, o_j \in O \text{ and } 0 \leq t \leq m-1 \text{ if } m^i_{jt} = 1 \quad (C_{13})$$

Fig. 6. Binary Integer program for routing in our graph model

categories of edges in either incoming and outgoing directions. Transmitting vertices have only incoming internal edges and outgoing vertical edges. Lastly, destination vertices have only incoming destination edges.

A vertex $v^R_{t,i} \in V^R$ has five types of incoming and outgoing edges as shown in Figure 5: a set of vertical incoming edges, a set of horizontal incoming edges, a set of horizontal outgoing edge, a set of vertical outgoing edge to the transmitting module and an edge to v^D_i. The combinations of their corresponding values are also presented in Table V. The total weight of all incoming edges must be zero or one. Otherwise, it violates the definition of a tree structure where there is only one path from the root to every node on the tree. In addition, a vertex $v^R_{t,i} \in V^R$ has four different roles in routing. First, it can be the source vertex. As shown in the seventh column of Table V where the cross sign represents the invalid combination of values, it has no incoming edge, and the edge to its destination vertex must not be enabled. Second, this type of vertex can be the vertex on the tree which directly connects to its destination vertex. The third role for this type of vertex is to be an intermediate vertex, which does not directly connect to the destination vertex. The last one is a vertex that does not participate to form the tree. Similar to the last two roles, intermediate vertices and vertices which are not on the tree also have some invalid combinations. In Table V, the last column summarizes all invalid scenarios for the last three roles of receiving and transmitting vertices.

A vertex $v^T_{t,i}$ has the following properties shown in Figure 5:

1) if $v^T_{t,i} \in V^T$ is an intermediate vertex on the tree, $e^I_{t,i} = 1$ while $\sum_{v^R_{t,k}} e^V_{t,i,k} \geq 1$ for all $o_k \in O$ if $m^i_{kt} = 1$.
2) if $v^T_{t,i} \in V^T$ is not on the tree, $e^I_{t,i} = 0$ while $\sum_{v^R_{t,k}} e^V_{t,i,k} = 0$ for all $o_k \in O$ if $m^i_{kt} = 1$.

A vertex c^D_i has the following properties shown in Figure 5:

1) $\sum_{t=0}^{m-1} e^D_{t,i} = 1$ if v^D_i is one of the destinations
2) $\sum_{t=0}^{m-1} e^D_{t,i} = 0$ if v^D_i is not one of the destinations

TABLE V
COMBINATIONS OF EDGE VALUES TO IDENTIFY $i \in V^R$

Constraint	$\sum_{v^T_{t,j}} e^T_{t,j,i}$	$e^H_{t-1,i}$	$e^I_{t,i}$	$e^H_{t,i}$	$e^D_{t,i}$	$v^R_{st,s}$	$v^R_{t,i} \in V^R$
	0	0	0	0	0	×	√
C_9	0	0	0	0	1	×	×
C_8	0	0	0	1	0	√	×
C_8, C_9	0	0	0	1	1	×	×
C_7	0	0	1	0	0	√	×
C_7, C_9	0	0	1	0	1	×	×
C_7, C_8	0	0	1	1	0	√	×
C_7, C_8, C_9	0	0	1	1	1	×	×
C_6	0	1	0	0	0	×	×
C_9	0	1	0	0	1	×	×
	0	1	0	1	0	×	√
C_9	0	1	0	1	1	×	×
	0	1	1	0	0	×	√
C_9	0	1	1	0	1	×	×
	0	1	1	1	0	×	√
C_9	0	1	1	1	1	×	×
C_6	1	0	0	0	0	×	×
	1	0	0	0	1	×	√
	1	0	0	1	0	×	√
	1	0	0	1	1	×	√
C_7	1	0	1	0	0	×	×
C_7	1	0	1	0	1	×	×
C_7	1	0	1	1	0	×	×
C_7	1	0	1	1	1	×	×
C_6	1	1	0	0	0	×	×
C_6	1	1	0	0	1	×	×
C_6	1	1	0	1	0	×	×
C_{10}	1	1	0	1	1	×	×
C_6	1	1	1	0	0	×	×
C_{11}	1	1	1	0	1	×	×
	1	1	1	1	0	×	√
	1	1	1	1	1	×	√

To enforce all properties regarding different types of vertices, we have the formulation shown in Figure 6. Given a directed graph G, let $w(e)$ be the cost of an edge $e \in E$. Given a source vertex $v^R_{st,s}$ and a set of destination vertices D, we present our BIP formulation for the minimum edge-weighted directed Steiner tree problem.

There is a total of 13 constraints in our formulation.[1]

- Constraint $C1$ treats each edge as a variable to determine whether that edge is on the optimal Steiner tree.
- Constraint $C2$ controls destination vertices. All desired destination vertices on the tree must have one incoming edge. Other destination vertices that do not belong to the destination set D must not have any incoming edge at all.
- Constraints $C3$ and $C4$ are used to constrain the source. A source has no incoming edge (see $C3$). In addition, it must have at least one outgoing edge except the destination edge (see $C4$).
- Constraints $C5$ to $C11$ are used to regulate receiving and processing vertices. For example, constraint $C5$ requires that every receiving and processing vertex can only have one incoming vertical edge.
- Constraints $C12$ and $C13$ are used to control each transmitting vertex. If its internal edge is on the tree, at least one of the vertical outgoing edges must also be on the tree. Otherwise, both incoming and outgoing edges must not be included on the tree.

Even though we have discussed the domain-oriented constraint in the previous section, no constraint in our formulation is dedicated to it. However, if we have a close look at Table V, our domain-oriented constraint is enforced by combining different constraints in the formulation. We know that in a logical tree, each vertex has only one incoming edge and at least one outgoing edge. However, if we look at two valid scenarios presented by the last two rows in Table V, both incoming vertical and horizontal edges can be included on the tree. It seems that they are contradictions to the definition of a logical tree, but this is valid because of our domain constraint and the repetition of periodic encounters. For example, if an object receives a packet from another object (one of the incoming vertical edges is enabled), this object can store the packet and transmit it in the future. If the best time to transmit this packet is at the same phase during the next period, as a result, both incoming horizontal and vertical edges will be included on the tree.

[1]Table V provides details about the functionalities of each constraint in this group. In that table, cross signs represent violations; check signs represent valid scenarios.

B. Optimal and approximate algorithms

We have to treat the minimum energy consumption and minimum delay problems slightly differently. This is because the nature of time in the minimum delay problem is quite different from that of energy usage in the minimum energy problem. In this section, we present approaches that address these two problems separately.

1) The minimum delay problem:

The minimum delay problem in routing is relatively simple. The main result can be found in the following theorem.

Theorem 1: The minimum delay in routing for a set of destinations equals the maximum of all pair-wise minimum delays between the source and destinations.

Proof: We prove this theorem by induction. Let us use d to represent the number of destination objects in the graph.

If $d = 1$, then a path with minimum delay from the source to that object must also yield the the minimum delay for unicast routing.

If $d = 2$, we can pick one object first and find a path with the minimum delay from the source to it. Then, we find a path with the minimum delay from the source to the second object. If the delay to reach the second object is less than the delay to reach the first object, the delay to reach the second object must be embedded in the delay to reach the first object. Otherwise, the minimum delay to reach the second object must include the delay to reach the first object. Therefore, in either case, the maximum of the minimum pair-wise path delays among two objects is the minimum delay in multicast routing.

Let us assume our theorem is true when $d = n-1$.

If d equals n, for every combination of $n-1$ destination objects, the minimum delay to reach those $n-1$ destinations is the maximum of all minimum path delays between the source and $n-1$ destinations. By introducing one extra destination object, we need to show that the theorem statement holds as well. We can now use the same argument for the case where $d = 2$ to complete the induction argument. ■

From our theorem, many classical shortest path algorithms can be used to solve the minimum delay problem with small modifications. The algorithm used in this paper for the minimum delay problem is a modification of Dijkstra's algorithm [5]. Its pseudo-code is presented as Algorithm 1. There is one difference with respect to the original algorithm in [5]: the if-statement at lines 18 to 21 is inserted to enforce the NOT constraint. This if-statement does not re-direct the optimal route; it only adjusts the weight of related vertices to avoid transmissions that violate the NOT constraint. For example, in line 19, if the predecessor and successor of the current vertex on the route are from the same phase, it is a violation of the NOT constraint. If the retransmission must be part of the optimal route, that means the edge in the graph for the retransmission represents the only encounter between both intermediate objects. In other words, there is only one path between the source and the destination. Because of our domain-oriented constraint, the retransmission to its successor has to wait, or the packet has to be held until the same phase in the next period. As shown in line 20, data storage requires an additional energy cost, $\varepsilon \times m$ units of energy, to store the message for a whole period with m phases. The principle of the modification in Algorithm 1 is: whenever there is a violating retransmission, we eliminate the retransmission by having the object to store the query until the same encounter at the next period in time. Because the cost of storing the message has already been included in the final cost in Algorithm 1, the route returned is an optimal route that does not violate the NOT constraint. Note that if we remove lines 18 to 21 from the algorithm, this part of the algorithm is exactly the same as Dijkstra's algorithm. In other words, the path to each destination generated by the algorithm which satisfies our constraint is guaranteed to be the optimal route, and furthermore, it can be found in polynomial time (c.f., Theorem 2). In summary, our modified Dijkstra's algorithm finds a shortest-path tree for a given set of destinations. Based on Theorem 1, the minimum delay to reach all destinations equals the weight of the longest shortest path on the tree.

In order to prove the correctness of Algorithm 1, we prove

Algorithm 1 ModifiedDijkstra(G, D, w, s)

```
1:  for each vertex v ∈ V[G] do
2:      d[v] ← ∞            //initialize the distance to be infinite
3:      p[v] ← null         //set the predecessor to be empty
4:  end for
5:  d[s] ← 0                //initialize the source distance
6:  S ← ∅
7:  Q ← V[G]                //initialize the priority queue
8:  while Q ≠ ∅ do
9:      u = EXTRACR-MIN(Q)
10:     if u ∈ D then
11:         D ← D\{u}
12:     end if
13:     if D == ∅ then
14:         break
15:     end if
16:     for each vertex v ∈ Adj[u] do
17:         if d[v] > d[u] + w(u,v) then
18:             //check whether the predecessor/successor and the
                current vertex are from the same phase
19:             if phaseOf(v) == phaseOf(p[u])
                    && phaseOf(v) == phaseOf(u) then
20:                 d[v] ← d[u] + ε * m
21:             else
22:                 d[v] ← d[u] + w(u,v)
23:             end if
24:             p[v] ← u
25:         end if
26:     end for
27: end while
28: // retrieveTree function retrieves the tree by traversing the
    predecessor recorded at each destination back to the source
29: T = retrieveTree(G,D,s)
30: return T
```

that the tree returned from that modified algorithm is the shortest path tree.

Theorem 2: Given a source and a set of destinations, Algorithm 1 returns a shortest path tree satisfying the NOT constraint.

Proof: As we discussed before, the only difference between Algorithm 1 and Dijkstra's algorithm is that we enforce the NOT constraint from line 18 to 21. As a result, Algorithm 1 only examines paths that satisfy the NOT constraint, and does not change the logic of Dijkstra's algorithm. Therefore, the path to each one of the destinations returned by Algorithm 1 satisfies the NOT constraint, and must be the shortest based on the correctness of Dijkstra's algorithm. Because all optimal paths originate from the same source, if we combine those optimal paths, the tree returned must also be a shortest path tree satisfying the NOT constraint. ■

Using Theorems 1 and 2, the weight of the longest path between the source and any destination returned by Algorithm 1 is the minimum delay in order to deliver a query to multiple destinations.

It is obvious that the if-statement from line 18 to 21 only updates the weight of the corresponding vertices. It does not bring additional complexity into the algorithm. Therefore, based on the complexity of Dijkstra's algorithm, our algorithm has polynomial complexity.

2) The minimum energy consumption problem:

As we mentioned before, the minimum energy consumption problem can be solved by finding the minimum weighted Steiner Tree in the graph where the destination vertices are leaf nodes [32]. However, the minimum weighted Steiner Tree problem is a classical NP-hard problem [13] and has been well studied [17, 18, 45]. The minimum weighted Steiner tree problem can be further divided into two categories: node-weighted Steiner tree and edge-weighted Steiner problem. The minimum energy consumption problem in our graph model falls into the second category. In graph theory, the minimum weighted Steiner tree problem is NP-hard in general; however, studies have shown that the Steiner tree problem can be solved in polynomial time in many special graph types such as Series-Parallel graph, Halin graph, K-planer networks and strongly chordal graphs [39, 45]. Unfortunately, our graph model does not belong to any one of these graph types.

Previous works have presented both exact algorithms and approximation algorithms that are based on some heuristics. The exact algorithms that have been proposed are all exponential algorithms [10, 14, 25]. They are not practical for graph models with a large number of vertices. Therefore, we focus on approximation algorithms which can find the result faster, although without the guarantee of optimality. Many works have been conducted on heuristics [22, 34, 36, 42]. The most well-known heuristics are from Kou et al. [22] and Takahashi and Matsuyama [42]. In their paper, Kou et al., used the minimum spanning tree heuristic on source and destinations; Takahashi and Matsuyama used the minimum cost paths heuristic. Both algorithms can achieve a $O(kn\log n)$ complexity where k is the number of destinations; however, these algorithms do not work on directed graphs.

In order to find a heuristic for directed graphs, we decided to modify previous heuristic algorithms for undirected graphs. The minimum spanning tree heuristic cannot be applied to our graph model because the destination vertices has no outgoing edges. The algorithm cannot construct a complete graph among the source and all destination vertices. Therefore, we focus on the minimum cost path heuristic.

In the minimum path cost heuristic [42], the algorithm first finds a path from the source to one of the destinations that has the minimum cost among all destinations. Using the path as the base of a tree, the algorithm selects the next destination that is the closest to the base tree and merges the selected path to that tree. The algorithm keeps doing this until the last destination vertex has been included in the tree. This is why the algorithm has $O(kn\log n)$ complexity. There are k destinations in the tree, and each one of them is required to run Dijkstra's algorithm once. As we just mentioned, the destination vertices in our graph model have no outgoing edge. Thus, there is no path to any vertex in the graph by using the destination vertex as the source. To change that, we make a small modification from the original algorithm. Instead of running Dijkstra's algorithm from destination vertices, the next closest destination vertex can be used by running Dijkstra's algorithm from all intermediate vertices on the result tree. A detailed algorithm is described in Algorithm 2. In the

Algorithm 2 Shortest Path Heuristic to find minimum weighted directed Steiner tree(G, D, w, s)

```
1:  T ← {s}                    //initialize the result tree
2:  shortestPathTree = null
3:  while D ≠ ∅ do
4:      minCost ← ∞
5:      P ← ∅                  //the set used to hold the selected path
6:      v̂ ← null
7:      d̂ ← null
8:      for each vertex v ∈ T do
9:          if shortestPathTree[v] == null then
10:             shortestPathTree[v]←ModifiedDijkstra(G,D,w,v)
11:             for each destination d ∈ D do
12:                 if cost(path(v,d)∈shortestPathTree[v])<minCost
                    then
13:                     minCost = cost(path(v,d))
14:                     v̂ ←v
15:                     d̂ ←d
16:                     P←path(v,d)
17:                 end if
18:             end for
19:         end if
20:     end for
21:     T←T∪P\{v̂}
22:     D←D\{d̂}
23: end while
24: return T
```

algorithm, we introduce a storage space, *shortestPathTree*, to store the shortest-path tree rooted at each intermediate vertex that is returned by our modified Dijkstra algorithm in Algorithm 1. In addition, *path(v,d)* represents the shortest path from vertex v to destination d in the shortest-path tree rooted at v. In summary, Algorithm 2 incrementally builds the heuristic tree by inserting a path to one of the destinations at a time.

In the following analysis, we show that our algorithm is a k-approximation algorithm in the worst case.

Theorem 3: Given any directed graph $G = (V,E)$, a source vertex s and a set of destinations D where $D \in V$, Algorithm 2 always provides a solution to the minimum weighted directed Steiner tree problem that is at most k times the optimal solution.

Proof: According to Algorithm 2, there is an order among all destinations for the sequence that each one has been added into the tree. For simplicity, we name the destination vertices as $d_1, d_2, \cdots d_k$. A destination vertex with a smaller index is inserted into the result tree earlier. Correspondingly, the same sequence can be used to name the destination vertices in the optimal directed Steiner tree.

Given the optimal tree for the minimum weighted directed Steiner tree problem for k destinations, T^{OPT}, let $c^{OPT}(u,v)$ be the weight of the path between vertices u and v on the optimal tree. For the tree that is incrementally built by Algorithm 2, let $c(x,y)$ be the weight of the path between vertices x and y. Vertices without OPT notation are in the tree built by Algorithm 2, and the ones with OPT notation are from the optimal tree.

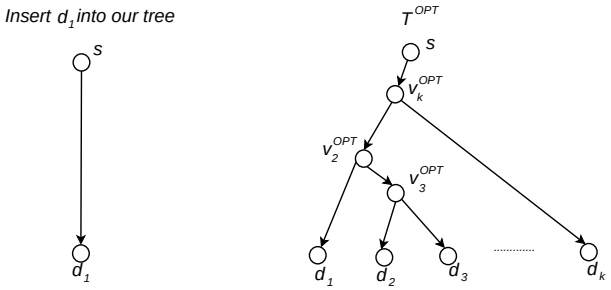


Fig. 7. Subtrees Partial tree T_1 and T^{OPT}

At the beginning, our algorithm creates a path with the smallest cost connecting the source and one of the destinations. As a result, our partial tree T_1 is a single path as shown in the left part of Figure 7 [2].

Because the path selected by Algorithm 2 is the shortest path between the source and any one of the desired destinations, the following inequality holds.

$$c(s,d_1) \leq c^{OPT}(s,d_1) \tag{2}$$

Next, the algorithm adds another destination d_2 into the tree. For illustration, please refer to Figure 8. When our algorithm adds the second destination d_2 into the tree, there is an intermediate vertex on the path (s,d_1) in the left part of Figure 7 that has the smallest cost to connect d_2. Let us call this vertex v_2. Both v_2 and d_2 in this example can be mapped to $\hat{v}$ and $\hat{d}$ in Algorithm 2 at each iteration. Similarly, there is a corresponding vertex on the optimal tree, T^{OPT}, that connects d_2 to the path between s and d_1. Let us call it v_2^{OPT}. Because the path between v_2 and d_2 has the smallest cost over all paths between s and d_2, we have the following inequality.

$$\begin{aligned} c(v_2,d_2) &\leq c(shortestPath(s,d_2)) \\ &\leq c^{OPT}(s,v_2^{OPT}) + c^{OPT}(v_2^{OPT},d_2) \end{aligned} \tag{3}$$

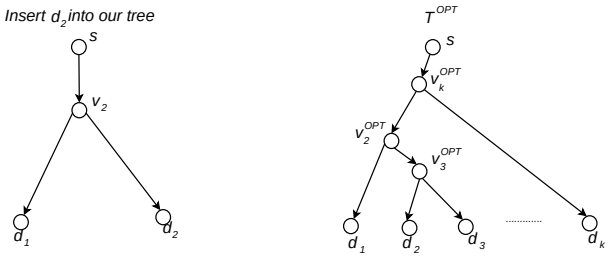


Fig. 8. Partial tree T_2 and T^{OPT}

When our algorithm adds the third destination d_3, with the same argument before, there are vertices v_3 and v_3^{OPT} in our tree and the optimal tree connecting d_3. Because of the shortest path heuristic in Algorithm 2, the cost between path v_3 and d_3 in our tree must be less than or equal to the any path between s and d_3 in the graph. Therefore,

$$\begin{aligned} c(v_3,d_3) &\leq c(shortestPath(s,d_3)) \\ &\leq c^{OPT}(s,v_3^{OPT}) + c^{OPT}(v_3^{OPT},d_3) \end{aligned} \tag{4}$$

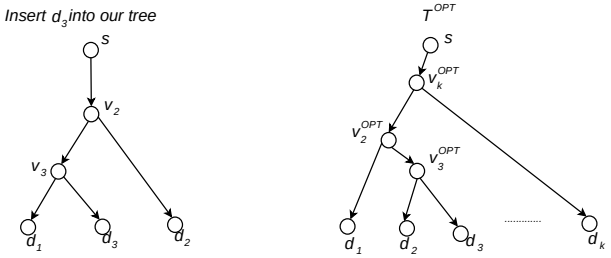


Fig. 9. Partial tree T_3 and T^{OPT}

The same analysis applies to each insertion of one destination. When the last destination d_k is inserted, we have the inequality shown in Equation 5.

$$\begin{aligned} c(v_k,d_k) &\leq c(shortestPath(s,d_k)) \\ &\leq c^{OPT}(s,v_k^{OPT}) + c^{OPT}(v_k^{OPT},d_k) \end{aligned} \tag{5}$$

After including all destinations, the cost of our tree, T, is:

$$c(T) = c(s,d_1) + c(v_2,d_2) + \cdots + c(v_k,d_k) \tag{6}$$

[2] All the figures in this proof are for demonstration only, vertices such as v_2, v_2^{OPT} and so on can be any vertex on the corresponding trees.

Similarly, the cost of the optimal tree, T^{OPT}, is:

$$c^{OPT}(T) = c^{OPT}(s,d_1)+c^{OPT}(v_2^{OPT},d_2) + \cdots + c^{OPT}(v_k^{OPT},d_k) \quad (7)$$

By applying the inequalities in Equations 2, 3, 7 and so on, we have:

$$\begin{aligned} c(T) &\leq c^{OPT}(s,d_1)+c^{OPT}(s,v_2^{OPT})+c^{OPT}(v_2^{OPT},d_2) \\ &\quad + \cdots + c^{OPT}(s,v_k^{OPT})+c^{OPT}(v_k^{OPT},d_k) \\ &= c(T^{OPT})+\sum_{i=2}^{k} c^{OPT}(s,v_i^{OPT}) \end{aligned}$$

and

$$\frac{c(T)}{c(T^{OPT})} \leq 1+\frac{\sum_{i=2}^{k} c^{OPT}(s,v_i^{OPT})}{c(T^{OPT})} \quad (8)$$

Therefore, our algorithm is a $1+\frac{\sum_{i=2}^{k} c^{OPT}(s,v_i^{OPT})}{c(T^{OPT})}$ approximation where $c(s,v_i^{OPT})$ is the shared weight with the existing subtree in the optimal solution for a destination when it is added into the tree. There are two extreme cases for this shared weight. First, the newly added destination does not use any edge from existing subtree; second, the newly added destination fully utilize edges on existing subtree to be reachable by the source s. Therefore, it is easy to see that $0 \leq c^{OPT}(s,v_i^{OPT}) \leq c(T^{OPT})$. In the worst case scenarios where all $c^{OPT}(s,v_i^{OPT}) = c(T^{OPT})$, we have

$$\frac{c(T)}{c(T^{OPT})} \leq 1+\frac{(k-1)*c(T^{OPT})}{c(T^{OPT})} = k$$

As a result, $c(T) \leq k*c(T^{OPT})$ and Algorithm 2 is a k-approximation algorithm. ■

C. Unicast, Broadcast Versus Multicast

In the previous discussion, distributing queries aiming at a set of destinations is called *multicast* whereas *unicast* and *broadcast* are just special cases of multicast. When there is only one destination, finding the minimum Steiner tree is to find the shortest path between the source and the destination. Both the minimum delay and minimum energy consumption problems in unicast routing are relatively easy to solve by using Algorithm 1. To achieve broadcast in our model, our objective is to connect the source to all destination vertices with the minimum Steiner tree rather than finding the minimum spanning tree to connect all vertices in the graph. With some variations, our graph models can also solve unicast and broadcast routing by treating them as special cases of multicast routing.

VI. Performance evaluation

In order to evaluate our method, we propose two sets of experiments where one set uses synthetic traces, and the other one uses real traces.

A. Evaluation Metrics

The complexity of the network is controlled by two different parameters: the number of objects in the network and each object's radio range. Increasing the number of objects will increase the number of vertices and horizontal edges. In addition, increasing the radio range increases the coverage area of each sensor, which leads to more encounters. In this section, we first present experiments using synthetic traces in which we evaluate the effectiveness and efficiency of our algorithms with respect to two parameters: (1) the number of objects, n, and (2) the radio range, r. In addition, a third parameter that is the number of destinations in a query, d, is introduced for multicast routing. Next, experiments using real mobile traces are presented. We compare our algorithm to those results obtained via two straightforward delay tolerant network routing protocols, *epidemic* [43] and *direct delivery* [38] protocols, because both protocols exhibit optimal bounds on either delivery ratio or network resources. In order to make a fair comparison, we make the assumption that there are unlimited buffers and energy supply in each objects to guarantee the maximum delivery ratio.

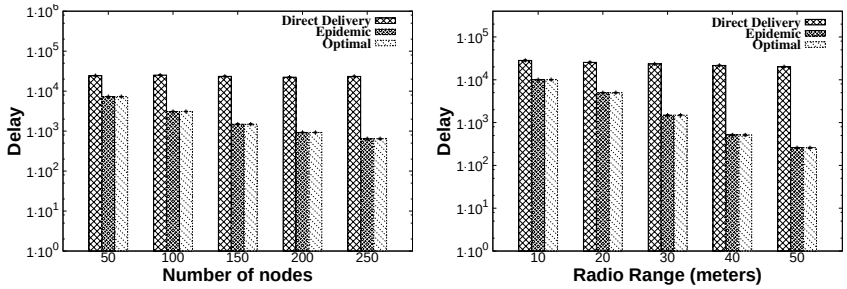
Two metrics are examined in our experiments: (1) Delivery delay and (2) energy cost. Regarding both metrics, the lower their values are, the better the performance is. In addition, in order to have accurate results, each measurement is calculated by taking the average of each metric from running 100 queries individually with different sources and destinations.

In our experiments, we evaluate both unicast and multicast communications. This is because in some application one type of communications may be preferred over the other.

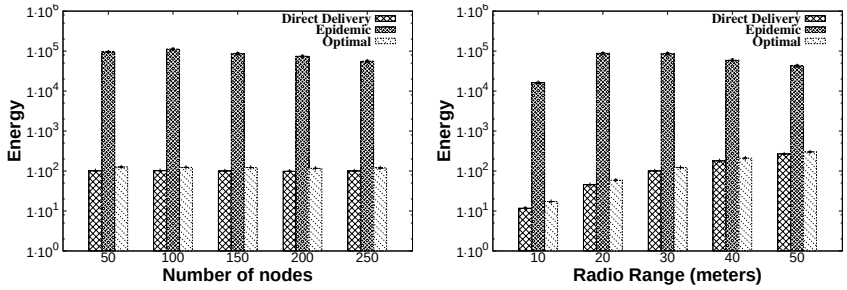
B. Experiments using synthetic traces

First, we examine the running time of our algorithm. Our experiments are implemented in Java with JDK 1.6. All mobility traces are generated using NS2 [1] Random Way-Point mobility model. Because this mobility model is not suitable to generate traces for a large number of objects, the maximum number of objects used in our experiments is 250. In addition, experiments are conducted within a 400×400 square meters area whereas artificial objects have the radio range up to 50 meters. Finally, we make the reasonable assumption that transmitting a message is much more expensive than simply holding on to a message during a phase. This is related to how one assigns cost to the vertical and horizontal edges in our graph model, respectively. We set the ratio of the energy cost to transmit a message over the energy cost to store-and-carry a message to be 1000:1. In our experiments, the energy cost grows quadratically with the radio range. In addition, the duration of each phase is set to be 60 seconds.

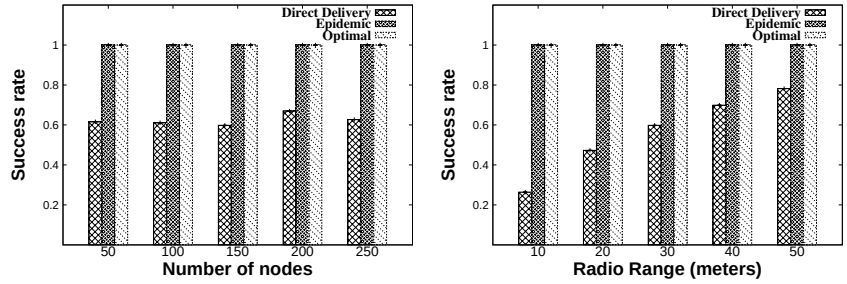
1) Unicast: In unicast routing, we focus on a special case of multicast where there is one destination per query. As we mentioned earlier, both the number of objects in the network, n, and object's radio range, r, are controlled in our experiments. By default, $n = 150$ and $r = 30$. The performance of our algorithm is presented in Figure 10 that contains two sets of experiments. The first column presents experimental



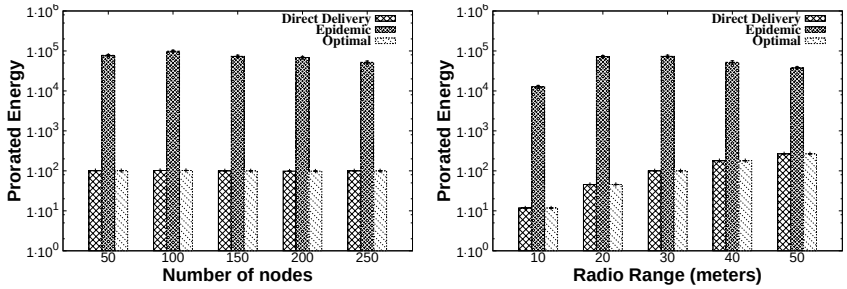
(a) Varying n while $r = 30$ (b) Varying r while $n = 150$



(c) Varying n while $r = 30$ (d) Varying r while $n = 150$



(e) Varying n while $r = 30$ (f) Varying r while $n = 150$

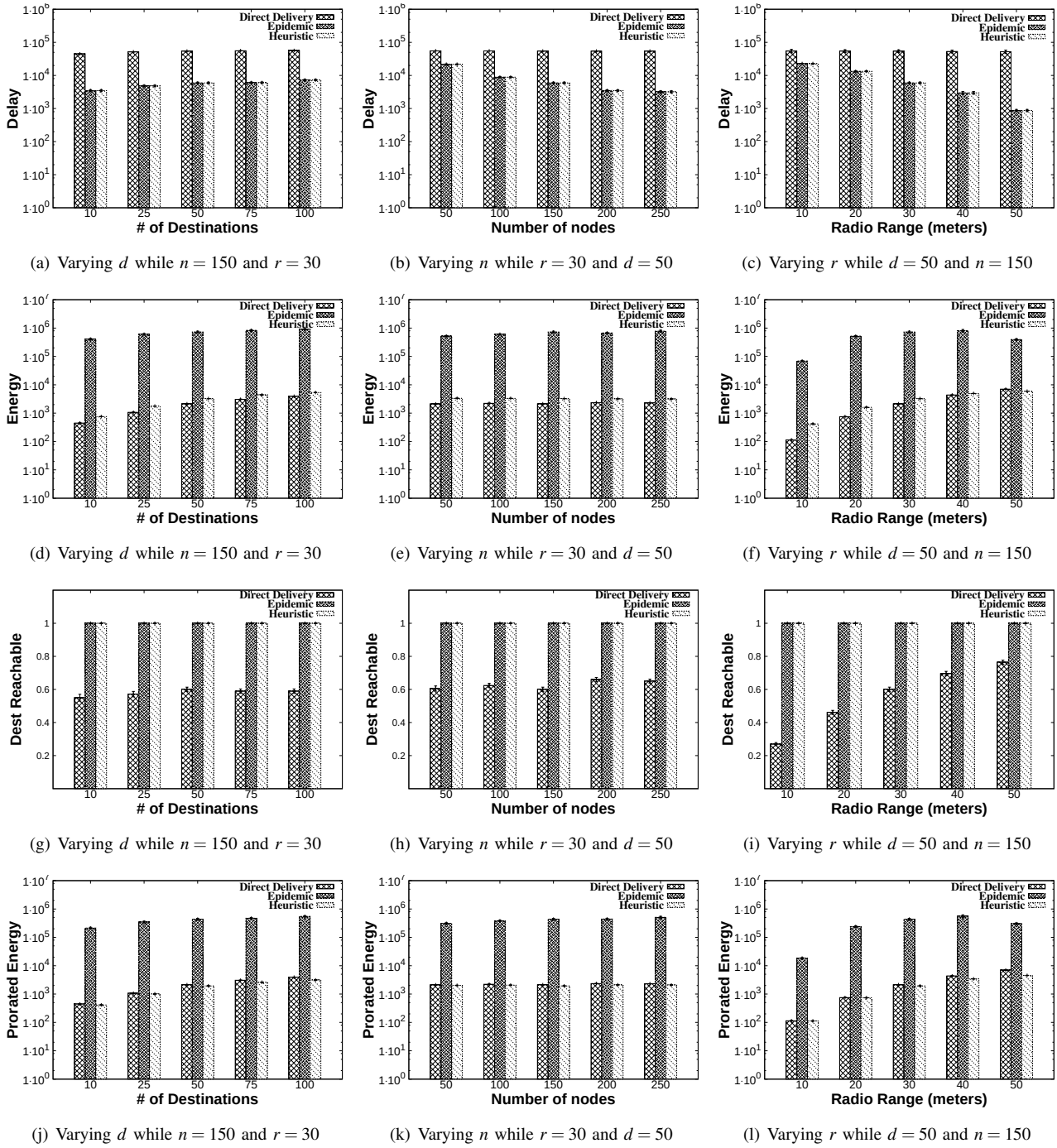


(g) Varying n while $r = 30$ (h) Varying r while $n = 150$

Fig. 10. Unicast performance regarding delay and energy cost

results with varying n while maintaining a fixed 30 meters radio range. The second column shows results by varying radio range while keeping 150 objects in the network. We compare our algorithm with direct delivery and epidemic routings. In order to be as fair as possible to epidemic routing we just accounted for the energy and delay spent until a route is found. This applies to the rest of our experiments.

Figures 10(a) and (b) show the end-to-end delay obtained by running our algorithm. Recall that both our approach and the epidemic routing provably yield the optimal delay, thus a comparison is not relevant in this case. As expected the more the objects (larger the radio range) the larger the number of encounters and the higher the probability that a shorter route is found. This causes descending trends in all three approaches in both experimental settings. However, the direct delivery method has the highest delay because the source has to wait until it meets the destination directly. In addition, this method does not have obvious declining trends because of the random



(a) Varying d while $n = 150$ and $r = 30$ (b) Varying n while $r = 30$ and $d = 50$ (c) Varying r while $d = 50$ and $n = 150$

(d) Varying d while $n = 150$ and $r = 30$ (e) Varying n while $r = 30$ and $d = 50$ (f) Varying r while $d = 50$ and $n = 150$

(g) Varying d while $n = 150$ and $r = 30$ (h) Varying n while $r = 30$ and $d = 50$ (i) Varying r while $d = 50$ and $n = 150$

(j) Varying d while $n = 150$ and $r = 30$ (k) Varying n while $r = 30$ and $d = 50$ (l) Varying r while $d = 50$ and $n = 150$

Fig. 11. Multicast performance regarding delay and energy cost

mobility model where the probabilities to meet other object do not vary significantly.

Figure 10(c) shows that the obtained energy cost is fairly insensitive to the the number of objects in the network. It turns out that for a given default radio range, all queries can be optimally delivered in a fairly small number of hops. We believe this is an artifact, rather than a characteristic, due to the size of the area of interest. In this experiment, direct delivery uses the least amount of energy because only one transmission is required to deliver the query whereas epidemic protocol costs the most because of flooding the query to every encountered object that does not have a copy of the query. Our algorithm has a performance very similar to that of direct delivery. When the radio range increases as shown in Figure 10(d), the energy cost increases for both direct delivery and our algorithm.

This may sound contradictory as larger radio ranges lead to more encounters and thus more chances of creating better

TABLE VI
REAL USER TRACES

Source	Users	Duration	Type	Analysed Users	Analysed Duration
Milano [30]	44	19 days	PMTR encounters	44	19 days
UMPC [23]	36	12 days	iMote encounters	36	12 days
Cambridge [37]	12	6 days	iMote encounters	12	6 days

Milano = University of Milano
PMTR = Pocket Mobility Trace Recorder
UMPC = University Pierre et Marie Curie

routes. However, as in the case of Figure 10(c), the actual number of hops in the optimal is fairly constant. The curves goes up due to the well known fact that the energy cost increases quadratically (at least) with the radio range. This increase in energy usage itself dominates any energy savings by having better routes. In contrast, the energy cost for epidemic algorithm increases at the first, then it drops after a certain point of time. It is because the increasing usage of energy first dominates the performance, then as the number of encounters increases, the saving from having better routes dominates the energy cost. If we keep increasing the radio range to a certain point of time where all objects encounter each other during the same phase, epidemic protocol will always find a route with only one transmission just like direct delivery.

In Figure 10(e), direct delivery has only approximately 60% of successful delivery ratio because of the random mobility model. This ratio increases to approximately 75% as the radio range increases in Figure 10(f) because more encounters in the network increases the probability to meet a certain object. The performance of direct delivery in the previous studies is calculated on successful deliveries whose total number is much less than that of our algorithm and epidemic protocol. Therefore, its performance could be overestimated. To cancel such effect, we compare three approaches only with the queries that are deliverable using direct delivery. The result in Figures 10(g) and (h) shows that our algorithm performs just as good as direct delivery.

2) Multicast: In addition to the number of objects in the network, n, and object's radio range, r, we introduced the number of destinations, d, in each query in our experiments for multicast routing. By default, $n = 150$, $r = 30$ and $d = 50$. The experimental results are presented in Figure 11 that contains three sets of experiments. The first column presents experimental results with varying the number of destinations in a query while maintaining 150 objects in the network and the radio range to be 30 meters. The second column shows results by varying the number of objects in the network while keeping 30 meters radio range and each query with 50 destinations. The third column gives experimental results by varying radio range where we use 150 objects in the network and queries with 50 destinations. In general, the experimental results are similar to the results for unicast routing except multicast routing costs more time and energy.

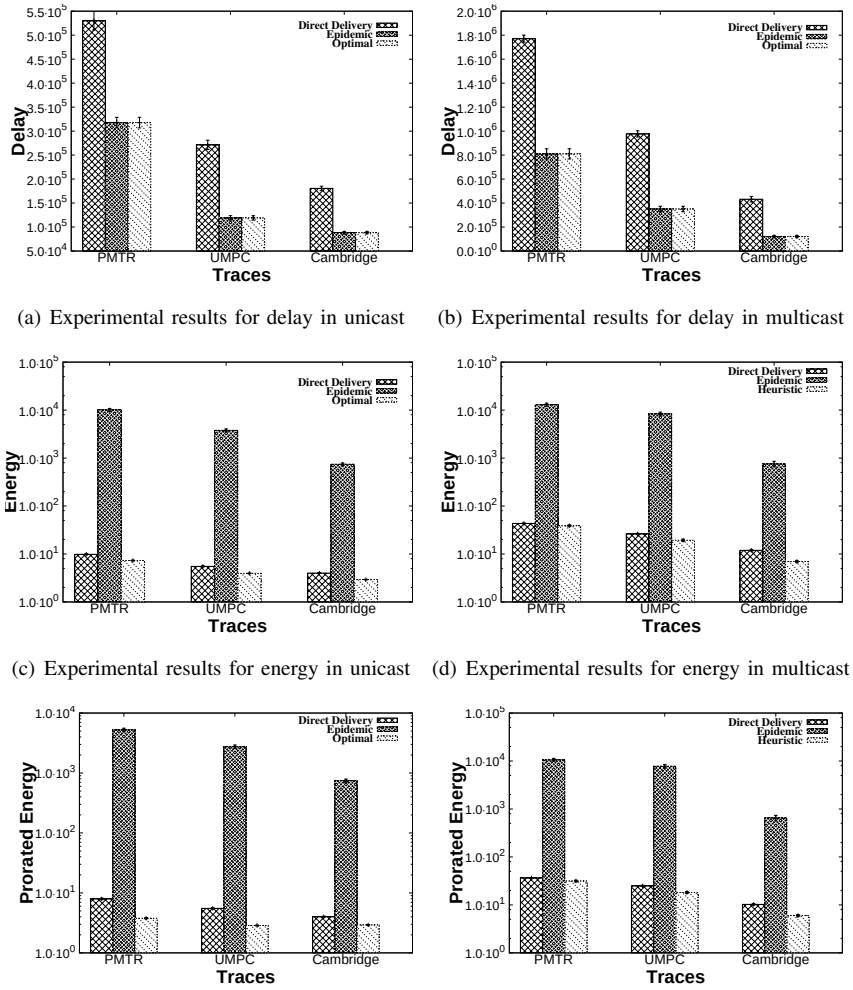
Figures 11(a), (b) and (c) show the query delay obtained by running different methods when varying three parameters. Results show that direct delivery has the highest delay in all circumstances because of the random mobility model where the probabilities to meet other object do not vary significantly. As the number of destinations increases, both our algorithm and epidemic protocol spend more time searching in the network to deliver the query to all destinations. For the other two parameters, as expected the more the objects and larger the radio range cause more encounters in the network leading to less time to reach all destinations.

For energy cost, epidemic routing has the highest energy cost in all three settings because of its nature of flooding, and direct delivery always has the lowest cost. Figure 11(d) shows that more energy is required in order to reach more destinations whereas experimental results in Figures 11(e) and (f) are very similar to the results in unicast routing. As shown in Figure 11(e), energy cost is fairly insensitive to the the number of objects in the network because of the artifact from random mobility model due to the size of the area of interest. The energy cost increases as the radio range increases because the energy cost is quadratically related to the radio range while the actual number of hops in the result is fairly constant. In general, our algorithm has a close performance as the direct delivery.

Similar to what we have discussed in unicast routing, direct delivery is not able to reach all destinations, Figures 11(g), (h) and (i) present the percentage of reachable destinations for each of method. We can see that direct delivery can only reach approximately 60% of destinations in average if we vary the number of destinations and the number of objects in the network. This probability is determined by the random mobility model. However, the percentage of reachable destinations increases as the radio rang increases because of more encounters in the network for direct delivery. Because our results are calculated by taking an average of delivering queries to all reachable destinations, only delivering to a portion of destinations makes direct delivery to be the best approach. For fair comparisons, we compare the results of three approaches prorated to the percentage of reachable destinations in direct delivery. In Figures 11(j), (k) and (l) our algorithm has either an equal or a better performance comparing to direct delivery after prorating the result.

C. Experiments using real mobile trace

In this section, we use traces collected from real mobile objects. There are total three traces being used in our experiments. Table VI presents the details of these traces. We derived encounters from each trace and assume derived encounters are



(a) Experimental results for delay in unicast

(b) Experimental results for delay in multicast

(c) Experimental results for energy in unicast

(d) Experimental results for energy in multicast

(e) Prorated energy cost in unicast

(f) Prorated energy cost in multicast

Fig. 12. Unicast and multicast performance regarding delay and energy cost in real mobile traces

periodic encounters which repeat themselves at every analysed duration along the time.

In our experiments, we again compare our algorithm to epidemic and direct delivery routing protocols. Again, each measurement is calculated by taking the average of 100 unicast/multicast queries where queries are generated randomly. For each trace, we use the same set of queries across three routing approaches to guarantee the fair comparison.

1) Unicast: Regarding delay in the optimal results found by our algorithm are equal to the ones returned by epidemic routing protocol. This is suggested by experimental results shown in Figure 12(a). However, when we compare our method to direct delivery routing, the routes found by direct delivery always have longer delays than our method.

For the energy cost, an object has to transmit whenever there is an encounter in epidemic protocol. This approach causes much more redundant transmissions comparing to directly delivery and our optimal routing in the graph model. As shown in Figure 12(c), average energy cost from epidemic routing is an order of multiple magnitude higher from direct delivery and our optimal algorithm. In theory, our algorithm can find an optimal route which is less or equal to the energy cost from direct delivery protocol. The prorated energy cost is presented in Figure 12(e).

2) Multicast: For the delay problem, our algorithm finds the shortest path tree whose delay equals the delay of the longest branch. This delay is the same as the delay returned by the epidemic protocol. As shown in Figure 12(b), the result of our algorithm is the same as the epidemic algorithm. In the direct delivery protocol, it always has to wait for the directly encounter. In general, the source has to wait until all reachable destinations are encountered. Therefore, the delay for each query in direct delivery protocol is longer than the delay to deliver the message to destinations in our algorithm and epidemic algorithm.

For the energy cost in the multicast, our algorithm and direct delivery consume much less energy comparing to epidemic protocol as shown in Figure 12(d). Experimental results show that our heuristic multicast tree works better than direct delivery in all three traces. Again, prorated energy analysis

is presented in Figure 12(f).

VII. Conclusion

Assuming a scenario of a mobile wireless network where disconnections are more the norm than an exception, we explored the ways mobility patterns of objects can be used to improve routing in such networks. We modelled the problem with a graph that is flexible enough to accommodate several different assumptions. We developed algorithms to solve *optimally* the complementary problems of forwarding a query with minimum delay, and with minimum energy, both in polynomial time. Our experiments show the effectiveness and efficiency of our solutions.

In order to simplify our model, we assume that each mobile object has the same mobility behaviours during every consecutive period in time. However, mobility behaviours are often disturbed by unexpected events. How to handle these sudden incidents requires our further studies. Also, all calculations and computations are currently performed in a centralized approach. In applications where there are thousands of mobile objects, this centralized approach may be not practical. Like conventional routing protocols, mobile objects should be able to make local decisions based on their mobility patterns. How to distribute this process is another interesting direction for further research.

VIII. Acknowledgements

Research partially supported by NSERC's DIVA Strategic Network.

References

[1] Ns2. http://www.isi.edu/nsnam/ns/.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38: 393 – 422, 2002.

[3] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *Proceedings of the 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–11, 2006.

[4] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6:606–620, 2007.

[5] Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L., and Stein, Clifford. *Introduction to Algorithms*. McGraw-Hill Science/Engineering/Math, 2nd edition, 2003. ISBN 0-26-2032937.

[6] P. Costa, C. Mascolo, M. Musolesi, and G. Picco. Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 26(5):748 –760, 2008.

[7] E. M. Daly and M. Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 32–40, 2007.

[8] J. A. Davis, A. H. Fagg, and B. N. Levine. Wearable computers as packet transport mechanisms in highly-partitioned ad-hoc networks. In *Proc. of IEEE ISWC*, pages 141–152, 2001.

[9] Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[10] S. Dreyfus and R. A. Wagner. The steiner problem in graphs. *Networks*, 1:195–207, 1972.

[11] H. Dubois-Ferriere, M. Grossglauser, and M. Vetterli. Age matters: efficient route discovery in mobile ad hoc networks using encounter ages. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 257–266, 2003.

[12] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proc. of IEEE SIGCOMM*, pages 27–34, 2003.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1990. ISBN 0716710455.

[14] S. L. Hakimi. Steiner's problem in graphs and its implications. *Networks*, 1:113–133, 1971.

[15] P. Hui and J. Crowcroft. How small labels create big improvements. In *Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 65–70, 2007.

[16] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: Social-based forwarding in delay-tolerant networks. *IEEE Transactions on Mobile Computing*, 10(11):1576 –1589, 2011.

[17] R. D. Hwang F. Steiner tree problems. *Networks*, 22:55–89, 1992.

[18] W. P. Hwang F, Richards D. The steiner tree problem. *Annals of Discrete Mathematics*, 53, 1992.

[19] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. In *Proc.of INMIC*, pages 62 – 68, 2001.

[20] S. Jain, R. C. Shah, W. Brunette, G. Borriello, and S. Roy. Exploiting mobility for energy efficient data collection in wireless sensor networks. *Mobile Network. Applications*, 11:327–339, 2006.

[21] D. B. Johnson, D. A. Maltz, and J. Broch. *DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks*. Addison-Wesley, 2001.

[22] L. Kou, M. G, and B. L. A fast algorithm for steiner trees. *Acta Informatica*, 15:141–145, 1981.

[23] J. Leguay, A. Lindgren, J. Scott, T. Friedman, J. Crowcroft, and P. Hui. CRAWDAD data set upmc/content (v. 2006-11-17). Downloaded from http://crawdad.cs.dartmouth.edu/upmc/content, Nov. 2006.

[24] V. Lenders, J. Wagner, and M. May. Analyzing the impact of mobility in ad hoc networks. In *Proc. of REALMAN*, pages 39–46, 2006.

[25] A. J. Levin. Algorithm for the shortest connection of a group of graph vertices. *Soviet Math*, 12:1477–1481, 1971.

[26] Q. Li and D. Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. In *Proc. of ACM MobiCom*, pages 44–55, 2000.

[27] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. In *Proceedings of SIGMOBILE on Mobile Computing and Communication Review*, page 2003, 2004.

[28] N. Maculan. The steiner problem in graphs. *Annals of Discrete Mathematica*, 31:185–222, 1987.

[29] Z. Melzak. On the problem of steiner. *Canadian Mathematical Bulletin*, 4:143–148, 1961.

[30] P. Meroni, S. Gaito, E. Pagani, and G. P. Rossi. CRAWDAD data set unimi/pmtr (v. 2008-12-01). Downloaded from http://crawdad.cs.dartmouth.edu/unimi/pmtr, Dec. 2008.

[31] D. Nain, N. Petigara, and H. Balakrishnan. Integrated routing and storage for messaging applications in mobile ad hoc networks. *Mobile Network Applications*, 9:595–604, 2004.

[32] C. A. S. Oliveira and P. M. Pardalos. A survey of combinatorial optimization problems in multicast routing. *Computers & Operations Research*, 32:1953–1981, 2005.

[33] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (aodv) routing. In *Proc. of IEEE SIGCOMM*, pages 234–244, 1994.

[34] J. Plesnik. A bound for the steiner tree problem in graphs. *Mathematica Slovaca*, 31:155–163, 1981.

[35] R. Ramanathan, R. Hansen, P. Basu, R. Rosales-Hain, and R. Krishnan. Prioritized epidemic routing for opportunistic networks. In *Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking*, pages 62–66, 2007.

[36] V. J. Rayward-Smith. The computation of nearly minimal steiner trees in graphs. *International Journal of Mathematical Education in Science and Technology*, 14:15–23, 1983.

[37] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau. CRAWDAD data set cambridge/haggle (v. 2009-05-29). Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/haggle, May 2009.

[38] L. Song and D. F. Kotz. Evaluating opportunistic routing protocols with large realistic contact traces. In *Proceedings of the second ACM workshop on Challenged networks*, pages 35–42, 2007.

[39] J. P. Spinrad. *Efficient graph representations*. American Mathematical Society, 2003. ISBN 0821828150.

[40] J. Su, A. Chin, A. Popivanova, A. Goel, and E. de Lara. User mobility for opportunistic ad-hoc networking. In *Proc. of IEEE SIGCOMM*, pages 41–50, 2004.

[41] J. Su, A. Goel, and E. de Lara. An empirical evaluation of the student-net delay tolerant network. In *Proceedings of the 3rd Annual International Conference on Mobile and Ubiquitous Systems - Workshops*, pages 1 –10, 2006.

[42] H. Takahashi and A. Matsuyama. An approximate solution for the steiner problem in graphs. *Mathematica Japonica*, 24:573–588, 1980.

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

[44] Z. Wang, M. A. Nascimento, and M. H. MacGregor. On the analysis of periodic mobility behaviour. In *Mobile Data Challenge Workshop*, 2012.

[45] P. Winter. Steiner problem in networks: a survey. *Networks*, 17:129–167, 1987.

[46] X. Zhang, J. Kurose, B. N. Levine, D. Towsley, and H. Zhang. Study of a bus-based disruption-tolerant network: mobility modeling and impact on routing. In *Proc. of ACM MobiCom*, pages 195–206, 2007.

[47] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *Proc. of ACM MobiCom*, pages 187–198, 2004.