Machine Learning in Disruption-Tolerant MANETs

BO XU
University of Illinois at Chicago
OURI WOLFSON
Pirouette Software Consulting and University of Illinois at Chicago
and
CHANNAH NAIMAN
Pirouette Software Consulting

In this article we study the data dissemination problem in which data items are flooded to all the moving objects in a mobile ad hoc network by peer-to-peer transfer. We show that if memory and bandwidth are bounded at moving objects, then the problem of determining whether a set of data items can be disseminated to all the moving objects is NP-complete. For a heuristic solution we postulate that a moving object should save and transmit the data items that are most likely to be new (i.e., previously unknown) to future encountered moving objects. We propose a method to be used by each moving object to prioritize data items based on their probabilities of being new to future receivers. The method employs a machine learning system for estimation of the novelty probability and the machine learning system is progressively trained by received data items. Through simulations based on real mobility traces, we show the superiority of the method against some natural alternatives.

Categories and Subject Descriptors: H.2.4 [**Database Management**]: Systems—*Distributed databases*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed databases; distributed applications*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Mobile data management, mobile ad hoc networks, publish/subscribe, resource discovery, mobile peer-to-peer networks

This research was supported by NSF grants DGE-0549489, 0513736, 0326284, OII-0611017.
Authors' addresses: B. Xu, Department of Computer Science, College of Engineering, University of Illinois at Chicago, IL; email: boxu@cs.uic.edu; O. Wolfson, Pirouette Software Consulting, Department of Computer Science, College of Engineering, University of Illinois at Chicago, IL; email: wolfson@ cs.uic.edu; C. Naiman, Pirouette Software Consulting, Chicago, IL; email: naiman@ pirouette-software.com.

DOI 10.1145/1636665.1636669 http://doi.acm.org/10.1145/1636665.1636669

ACM Reference Format:
Xu, B., Wolfson, O., and Naiman, C. 2009. Machine learning in disruption-tolerant MANETs. ACM Trans. Autonom. Adapt. Syst. 4, 4, Article 23 (November 2009), 36 pages.
DOI = 10.1145/1636665.1636669 http://doi.acm.org/10.1145/1636665.1636669

1. INTRODUCTION

A mobile ad hoc network (MANET) is a network of autonomous moving objects connected by short-range wireless links such as 802.11 or Bluetooth. A MANET that is subject to frequent disconnections is sometimes called a Disruption-Tolerant Network (DTN)[1] (see, e.g., Burns et al. [2005]). In this article we consider two application scenarios in a MANET or a DTN, namely routing and dissemination. In the routing scenario, a message is delivered from a source object to a specific single destination object. In the dissemination scenario, a message is delivered from a source object to all the objects in the system. An application example for the dissemination scenario is as follows. The moving objects (PDAs and/or vehicles) carry sensors which monitor the environment, for example, take pictures to detect intruders, sense available parking slots, or concentrations of various substances in the air. Assume that the set of values simultaneously sensed by a moving object O constitute a data item generated by O, and for reliability and functionality (so that each user has a local database representing the neighborhood) each data item has to be disseminated to all the moving objects. Since the ranges of 802.11 or Bluetooth are not sufficient to reach all moving objects, the dissemination is done by transitive multihop transmission.

There are two paradigms to conduct this dissemination, namely structured and structureless. In structured dissemination, a relay structure such as a tree is imposed and maintained among the moving objects (e.g., Huang and Garcia-Molina [2004]). Structured dissemination may be very ineffective in a highly mobile MANET, in which the routing structure quickly becomes obsolete. In structureless dissemination, the intermediate objects save data items and later (as new neighbors are discovered) transfer these data items. In the literature this paradigm is also called structureless gossiping [Jelasity et al. 2004; Friedman et al. 2007; Birman et al. 1999], epidemic [Vahdat and Becker 2000], or store-and-forward dissemination [Sormani et al. 2006].

Since there is no central control point, each moving object needs to decide when a data item has already been communicated to all the other moving objects in the system and therefore can be dropped. To do so, each moving object needs to collect information, analyze it, and decide whether to drop a data item from communication or from memory. This leads to the autonomic control loop, as introduced in Dobson et al. [2006].

We first formalize the data dissemination problem, and consider it in an ideal environment where all the trajectories of moving objects are known a priori at a centralized location. We show that if memory and bandwidth are bounded at

[1]DTNs is a newer term, favored by DARPA (Defense Advanced Research Projects Agency) over delay-tolerant networks.

moving objects, then the problem of determining whether a set of data items can be disseminated to all the moving objects is NP-complete (although we show that the problem can be solved efficiently if memory and bandwidth are unbounded). Thus in a distributed/mobile environment, where information such as the trajectory of a moving object is usually unknown to other objects, we resort to heuristics. More specifically, in our scheme each moving object prioritizes the data items in order to accommodate them in limited bandwidth and memory.

In particular, we postulate that an important factor that determines the priority of a data item is its current novelty probability, that is, its probability of being new (i.e., previously unknown) to an arbitrary moving object encountered in the future. The reason is that the transmission of data items that are already known wastes bandwidth, memory, and power. Thus, in this article data items are ranked based on their current novelty probability.

Under uniformity assumptions, the novelty probability of a data item R at a particular time is simply the ratio between the total number moving objects that have received R, and the total number of moving objects in the system. Unfortunately, a parametric formula giving the novelty probability is beyond the state-of-the-art. Results from epidemiology and random graphs may be useful in deriving such a formula, but they can't be applied directly. Even if such a formula existed it would depend on global parameters of the environment such as the turnover rate (i.e., the rate at which moving objects enter and exit the system) which are normally unknown to individual objects. To introduce the approach proposed in this article observe that the novelty probability of a data item R depends on attributes of R (e.g., its age), as well as global system parameters such as the turnover rate. The attributes of R that can affect its novelty probability are called *novelty indicators*.

In this article we show that, unfortunately, no single indicator is a good predictor of novelty in all environments. For example, in some environments the intuition that the age of the data item is a good predictor of novelty is correct. In other words, the older the data item, the more likely it is that an arbitrarily encountered object already knows it. However, consider an environment in which the turnover rate of moving objects is high; namely, many new moving objects are entering the system. In this case, even though the data item is old, it may not be known to the numerous objects that are new in the system. In this case, we show that the number of times a data item is received by a moving object is a better indicator of novelty to future-encountered objects than age. This variability is a problem since a moving object normally does not know the global parameters of the environment in which it operates. For example, it does not know if the moving object's turnover is high or low. Currently, a complete characterization of environments and novelty-indicators is beyond the state-of-the-art.

Previous work on dissemination in mobile ad hoc networks has studied various heuristics for reducing redundant transmissions, but each using a single individual novelty indicator. For example, Ni et al. [1999] considers the number of receptions and distance separately; Hayashi et al. [2003] and Burges et al. [2006] consider the number of hops.

In this article we propose a method that combines various novelty indicators in order to estimate the novelty probability. The combination uses machine

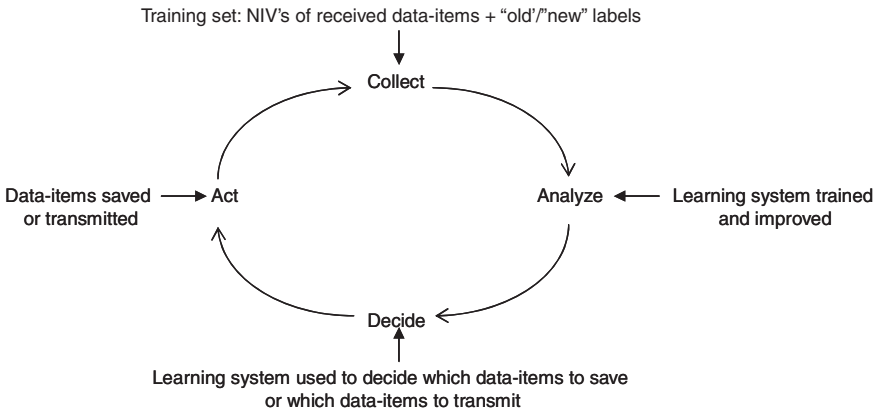


Fig. 1. The autonomic control loop formed by the MALENA method.

learning in a method called MALENA (MAchine LEarning-based Novelty rAnking). The method infers from previously received data items what the indicators of a new data item "look like." In other words, it learns the novelty probability based on the novelty indicators. For this purpose, the set of novelty indicators of a data item R, called the *Novelty Indicator Vector* (NIV) of R, are transmitted by each moving object together with the data item. This will increase the size of the data item by only a few bytes and it is negligible considering that data items containing, say, the picture of a conventioneer, may have a length of tens of Kbytes. The receiver of R determines whether or not R is new, and the respective NIV becomes a training example. In other words, the receiver treats itself as an arbitrary moving object that will be encountered in the future. Thus, a moving object progressively collects a training set which improves its learning system. When the object ranks its database to decide which data items to keep in memory or which data items to transmit, the learning system is used to calculate the novelty probability. Furthermore, using a sliding window of examples MALENA can adapt to new environments. In this sense MALENA provides a method that allows building Autonomic Communication (AC) algorithms (see Vasilakos et al. [2009]) that dynamically adapt to changing conditions or context. In summary, the execution of MALENA forms an autonomic control loop (see Dobson et al. [2006]) as shown in Figure 1.

Observe that it is possible to design some specific data dissemination algorithms that consider multiple parameters simultaneously, estimate parameters that cannot be measured directly, and adapt in sophisticated ways. However, MALENA provides a generic method in the sense that environments optimized by new novelty indicators can be considered simply by adding the new indicator to the machine learning system. Also, different machine learning algorithms can be "plugged-in" the method. (In this article we instantiate the method with the two novelty indicators *aro* and f_{in} discussed shortly, and with Bayesian machine learning). Furthermore, as we discuss at the end of the Introduction, the method can augment different store-and-forward algorithms.

We compare MALENA with three approaches to ranking of data items. In the first two approaches, ranking is based on a single individual novelty indicator. Specifically, the *aro*-ranking approach ranks data items based on the order in which they arrive at a moving object, and the f_{in}-ranking approach ranks a data item based on the number of times it is received by the moving object. We choose these two indicators because each of them is found to be an optimal indicator in an environment that is not favorable to the other. The third approach is PeopleNet [Motani et al. 2005]. We compare MALENA with these methods in both the dissemination scenario (i.e., all destinations) and the routing scenario (i.e., single destination).

Orthogonal to the application dimension, we compare the methods under two gossiping protocols, push gossiping and push-pull gossiping. In the push gossiping, two encountering moving objects directly exchange the data items in their local databases. In push-pull gossiping, the gossiping protocol follows two phases. In the first phase, the encountering moving objects exchange the ids of the data items in their local databases. In the second phase, they exchange the data items that are new (unknown) to the peer.

The performance measure that we propose is also novel. It integrates two existing evaluation metrics for data dissemination, namely throughput and delivery time, with the purpose of providing a single criterion for comparison. Thus the performance measure reflects both how many distinct data items are received by each moving object during its trip, and how soon they are received. The comparison of the methods is done by simulation, where we use traces of real Bluetooth sightings at a conference. We also evaluated MALENA using synthetic 802.11 traces.

The performance study indicates that, in either the dissemination or the routing scenarios, and with either push gossiping or push-pull gossiping, there are environments for which *age* is a better novelty indicator than f_{in} (in the sense that it has better performance), and there are other environments for which the opposite is true. However, MALENA always approaches or outperforms the best indicator in each individual environment, and therefore performs well in both. This demonstrates the power of combining the novelty factors and the adaptability of machine learning to the environment.

The approach enables us to argue about other novelty indicators as follows. In the existing literature on DTNs and MANETs, various assumptions are used, each typical to an environment. For example, some papers assume that the moving objects have GPS and thus know their locations [Lebrun et al. 2005; Burns et al. 2005], that the future trajectories of all the moving objects and the future communication traffic demand are known a priori [Jain et al. 2004], that moving objects have fixed periodic trajectories such as buses [Zhao et al. 2005], that the communication network is always connected [Karp and Kung 2000], and that the mobility of some peers can be controlled by the dissemination algorithm [Li and Rus 2000]. Subsets of assumptions may produce environments for which different indicators are superior, and moreover, a moving object may not know the parameters of the environment in which it is operating. In this case MALENA can be used to provide a performance that is close to optimal regardless of the environmental parameters.

Finally, an important comment is that MALENA is a method of improving routing and dissemination, rather than an algorithm, in the sense that it can be combined with other routing and dissemination algorithms. We explained before that the method enables the incorporation of other novelty indicators. Furthermore, the novelty probability computed by MALENA is orthogonal to and can be combined with other measures of data item priority such as size. Specifically, if data items have different sizes, then ranking can be done based on the novelty probability per byte, that is, the novelty probability divided by the size. Similarly, if data items have different reliability or popularity factors, the novelty probability can be multiplied by such a factor to compute the rank. Another orthogonal issue is bandwidth allocation, namely how many data items should a moving object transmit during a peer-to-peer interaction. This is important in a dense environment where collisions are a major concern and/or in a highly mobile environment where the communication time between encountering objects is limited. Again MALENA can be combined with any bandwidth allocation scheme in the sense that once the bandwidth allocation is determined, MALENA can be used to select the data items to transmit in such allocation. For an example of a bandwidth allocation scheme, see Wolfson et al. [2006b]. Yet another orthogonal issue is which moving objects to exchange data items with. For the sake of scalability, a moving object may not necessarily exchange with every encountered object. Gossiping protocols have developed randomized mechanisms for a moving object to select a subset of the encountered objects to exchange with (see, e.g., Jelasity et al. [2004], Friedman et al. [2007], and Birman et al. [1999]). Once this subset is determined, MALENA can be used to select the data items to transmit. In general, MALENA provides a method for estimating novelty probabilities, and it can be incorporated into and augment any algorithm that addresses many other aspects of data dissemination in mobile ad hoc networks, including bandwidth allocation, collision management, gossiping protocols, and data item prioritization.

The rest of the article is organized as follows. Section 2 formalizes the problem and analyzes it theoretically. Section 3 describes the MALENA method, and Section 4 evaluates MALENA by simulations, under the push gossiping. Section 5 describes and evaluates MALENA under push-pull gossiping. Section 6 discusses related work, and Section 7 concludes the article and discusses future work.

2. THE MOVING OBJECTS DATA DISSEMINATION (MODD) PROBLEM

In this section we define and analyze the MODD problem in the centralized case. We show that the problem can be solved efficiently when there are no resource constraints, but is intractable otherwise. In Section 2.1 we introduce the model, in Section 2.2 we discuss P2P communication, and in Section 2.3 we analyze the two MODD variants.

2.1 The Model

Our system consists of a finite set of point (i.e., without an extent) *moving objects*. Each object O has a memory allocation and a bandwidth/energy allocation

that are dedicated to data item dissemination. Bandwidth and energy impose constraints on how much a moving object can transmit during a peer-to-peer interaction and therefore we consider them collectively. The motion of moving object O is represented by a *trajectory* that defines the location of O as a function of time. For example, the motion can be approximated arbitrarily close by a piecewise linear function defined as follows.

Definition 1. A *trajectory* is a polyline in three-dimensional space (two-dimensional geography, plus time), represented as a sequence of points (x_1,y_1,t_1), (x_2,y_2,t_2), $\ldots$, (x_n,y_n,t_n) $(t_1 < t_2 < \ldots < t_n)$. The object is at (x_i,y_i) at time t_i, and during each time interval $[t_i,t_{i+1}]$, the object moves along a straight line from (x_i,y_i) to (x_{i+1},y_{i+1}), and at a constant speed given by $v_i = \frac{\sqrt{(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2}}{t_{i+1}-t_i}$. Thus, given a trajectory Tr, the *location* of the object at a point in time t between t_i and t_{i+1} $(1 \leq i \leq n)$ is obtained by a linear interpolation between (x_i,y_i) and (x_{i+1},y_{i+1}). We denote by $Tr(t)$ the location of Tr at time t.

Any other (i.e., nonlinear) function that gives the (x,y) location of a moving object as a function of time is satisfactory for the purpose of this article.

Occasionally, a moving object O produces a *data item* R having some unique *item-id,* and a size $s(R)$. For example, the data item is an image taken by the moving object. The uniqueness of the item-id can be achieved by, for example, the combination of the MAC address of O (which is globally unique) and a sequence number that monotonically increases for each data item produced by O. For simplicity we assume that all data items have the same size. A data item can also be called a tuple. O is called the *producer* of R. With each data item is associated a timestamp called the *produce-time,* indicating the time at which R is produced. In our model time is local to each moving object, and is represented as a sequence number, starting from 1.

Each moving object O maintains a *database* which can hold M_O data items. Each O uses the data items that it needs when they are received from other moving objects (see next subsection), or when O produces them. The database is maintained solely for the purpose of disseminating the data items to other moving objects.

In addition, O has a memory area called the *buffer* that is common to all the applications running on the moving object. We assume that the size of the buffer is at least M_O.

2.2 Peer-to-Peer Data Dissemination

Moving objects communicate via short-range wireless communication networks (such as Bluetooth or 802.11), having a *transmission range* d. Thus, pairwise communication can occur between two moving objects while the distance between them is at most d. When two moving objects A and B *encounter* each other (i.e., when the distance between them becomes smaller than d), they engage in a *data items exchange*. In Bluetooth, encounter detection is a built-in feature of the protocol. In 802.11, an encounter can be detected by periodical hello messages. The sets of data items exchanged depend on the bandwidth/energy

of each moving object, the memory allocation of each, and the period of time for which they stay within transmission range. For brevity we omit a formal definition of a pairwise data items exchange; but it has to respect the bandwidth and memory allocations of each moving object.

We first consider the centralized version of the *Moving Objects Data Dissemination* (MODD) problem which can be defined as follows.

Definition 2 (*The MODD Problem*). Given:

—a set of moving objects MO, each having a trajectory, a bandwidth/energy, and a memory allocation; and
—a *data items production schedule S,* that is, a set of data items, each of which has a size, a producing moving object, and a production time.

Can each data item in the schedule S be received by all the moving objects in MO by a sequence of data items exchanges?

Observe that due to resource (memory and bandwidth/energy) limitations, it is possible that each one of two data items in the schedule can reach a moving object O, but both cannot do so. For example, suppose that in order to reach O both data items have to be carried simultaneously by another object P, but P has enough storage for only one data item. More specifically assume that the only object that comes within transmission range of O is P, and the only time interval when it does so is 7:00 to 7:02. Furthermore, during this time interval P does not come within transmission range of any other object, except for O. Then, only one data item can reach O.

2.3 Variants of the MODD Problem

In this subsection we demonstrate that the MODD problem can be solved efficiently if bandwidth/energy and memory are unlimited, but becomes NP-complete when such limitations are introduced.

2.3.1 *Unbounded Resources.* First we consider the variant of the problem in which memory and bandwidth/energy are unlimited, and we show that in this case the MODD problem can be solved in polynomial time. Specifically, we show that there is an efficient algorithm to determine whether a given data item can reach a given moving object; and since resources are unlimited, reachability of one data item is independent of another, and thus a set of data items can reach all the moving objects if each data item can reach some moving object in the set.

If bandwidth is unlimited the transmission time of a data item is zero. Note that even when resources are unbounded, a data item produced by a moving object cannot reach another moving object. For example, this is the case when there are only two moving objects which do not encounter each other.

We start with the following definition of the contact graph, which is a tool that will enable us to determine the set of moving objects that a data item can reach.

Definition 3. Let $dis(p,q)$ denote the distance between location p and location q. A *contact interval* between two trajectories Tr_i and Tr_j (of moving

objects i and j), denoted $L_{ij}(s,e)$, is a time interval $[s,e]$ such that:

(1) for any time point $t \in [s,e]$, $dis(Tr_i(t),Tr_j(t)) \leq d$. (Recall that $Tr_i(t)$ and $Tr_j(t)$ are the locations of i and j at t interpolated by Tr_i and Tr_j respectively; and d is the transmission range), and

(2) if there exists a time point $t < s$ (or $t > e$) such that $dis(Tr_i(t),Tr_j(t)) \leq d$, then there exists a point t' such that $t < t' < s$ (or $e < t' < t$) and $dis(Tr_i(t'),Tr_j(t')) > d$. This indicates that the time interval is maximal in the sense that it cannot be extended.

Intuitively, a contact interval is a continuous time interval where the moving objects represented by the two trajectories are within the transmission range of each other.

Given a pair of trajectories, it can be determined in quadratic time using standard computational geometry techniques what are their contact intervals.

Definition 4. The *contact graph* is an undirected labeled multigraph, where each node i represents a moving object i, and each link $L_{ij}(s,e)$ represents a contact interval between i and j.

Note that there can be multiple links between i and j, each corresponding to one contact interval.

Example 1. Figure 2 gives an example contact graph of four objects A, B, C, D. There are two contact intervals between A and B, indicating that they encounter each other from 9:30 to 9:35 and from 13:15 to 13:20. Similarly, the contact graph gives the contact intervals between A and C, C and D, and B and D.

Given a contact graph and a data item produced by a moving object O at a time t, the reachability set of the data item is defined using the concept of the *earliest arrival time* at a moving object. Intuitively, the earliest arrival time at a moving object N is earliest time at which the data item can reach N. Formally, the earliest arrival time of nodes in the contact graph is defined recursively by the following algorithm.

Initially the earliest arrival time at each node except O is set to infinity. The earliest arrival time at O is t. The earliest arrival time is updated for a neighbor N of O if there is an edge L_{NO}(s,e) such that e is higher than t. In this case, let b be the minimum e among all such edges between N and O, and denote that edge by L_{NO}(a,b). (Intuitively, (a,b) is the earliest time interval during which the data item can be transmitted from O to N.) Let w be the maximum between a and t. The new earliest arrival time at N is the minimum between the existing earliest arrival time at N and w. The process is repeated for every neighbor N of O, and then O is marked "examined". Then the previous procedure is repeated starting from the unexamined node with the smallest earliest arrival time, until there are no unexamined nodes.

For some nodes of the graph the earliest arrival time may remain infinity at the end of the preceding algorithm. The *reachability set* of a data item is the set of moving objects that have a finite earliest arrival time.

Example 1 (*continued*). From Figure 2 we can see that D is reachable by a data item produced by A at any time before 9:30. One possible routing scheme

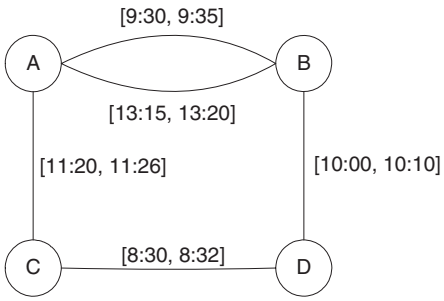


Fig. 2. An example contact graph.

is that A forwards the data item to B when A encounters B at 9:30, and then B forwards to D when B encounters D at 10:00. However, a data item produced by A at any time after 9:35 cannot reach D, via either B or C. It cannot reach via B because after 9:35 A encounters B after B encounters D. Similarly, the data item cannot reach via C because A encounters C after C encounters D.

THEOREM 1. *Assume that we are given a contact graph, and a data item R produced by a moving object i at time t. The reachability set of R can be determined in time which is quadratic in the size of the contact graph.*

Since in an unbounded resource instance of the MODD problem, the reachability set of one data item is independent of the set of other data items in the data items production schedule, we have the following.

COROLLARY 1. *Given a set of trajectories of size n, and a data items production schedule of m data items, the MODD problem can be solved in time $O(mn^2)$.*

2.3.2 *Bounded Resources.*

THEOREM 2. *Given bounds on memory and bandwidth for processors, the MODD problem is NP-complete.*

PROOF (SKETCH). The reduction is from the Multiprocessor Scheduling (MS) problem (see Garey [1979]). The complete proof is provided in Appendix A. □

Note that the bounded resource MODD problem is NP-complete even in the centralized case in which all the parameters are given a priori (i.e., all the trajectories and data item producers are known). In the distributed case a moving object usually does not know these parameters. Therefore, in the mobile case we will resort to heuristics that use machine learning.

Although there are many results for related problems in scheduling and network flow, we do not believe that the MODD problem has been solved.

3. NOVELTY PROBABILITY

In this section we discuss how to compute the probability for a data item to be new to future encountered objects. In Section 3.1 we discuss the indicators that will be used in computing the novelty probability, in Section 3.2 we discuss the Machine Learning (ML) framework, in Section 3.3 we discuss how to manage

the space overhead of the ML framework, and in Section 3.4 we discuss the use of the Bayesian machine learning algorithm in the framework.

3.1 Novelty Indicators

Consider the time when a moving object O assigns a rank to a data item R. We postulate that the probability that R will be new to the moving objects that will be encountered in the future by O depends on several elements called *novelty indicators*. The following are two possible novelty indicators.

(1) The relative order in which R arrives at O. This indicator is called the *arrival order*, and is denoted by *aro*. Specifically, if R is the kth data item that arrived at O (among all the data items in the current database), then the *arrival order* of R is k. Clearly $1 \leq aro \leq M_O$ (recall that M_O is the number of data items in O's database). A small arrival order suggests that R has been in the database for a relatively long time and thus has probably been in the system longer, and also has been transmitted by O more times than other data items. Therefore a small arrival order would indicate a low probability of future novelty.

(2) The number of times R has been received by O from other moving objects, denoted by f_{in}. The higher f_{in}, the less likely that R is new to O's future encountered objects, since this means that R has already been widely disseminated by other moving objects.

This set is by no means exhaustive. One can easily come up with other novelty indicators, such as the number of hops R has traveled before it reaches O, the number of times R has been transmitted by O, and the age of R. However, the method that we propose in this article is able to integrate these and other indicators. Moreover, we considered other indicators and found that among them *aro* or f_{in} are superior for the environments examined in this article.

Given a data item R at a moving object at a particular time, the pair (aro, f_{in}) is called the *Novelty Indicator Vector* (NIV) of R.

3.2 Computation of Novelty Probability Using Machine Learning

In this subsection we introduce a framework for using machine learning techniques to predict the novelty probability based on a novelty indicator vector. This is a general framework in the sense that different ML systems can be plugged into it.

ML intuitive framework. Suppose that we are given a multiset ES of examples, where each example is a pair ($\mathbf{X}$, *label*) and the same example may appear multiple times in the set. $\mathbf{X}$ is a NIV and *label* is either "new" or "old." The "new" indicates that the data item associated with the NIV $\mathbf{X}$ was new at the receiving moving object (i.e., the object has never received the data item before). And similarly, "old" indicates that the associated data item was not new.

A *machine learning system* Q is a function of the examples set ES and a NIV $\mathbf{X}$. Particularly, $Q(ES, \mathbf{X})$ returns the probability that a data item with NIV $\mathbf{X}$ will be new to encountered objects in the future, given the examples set ES.

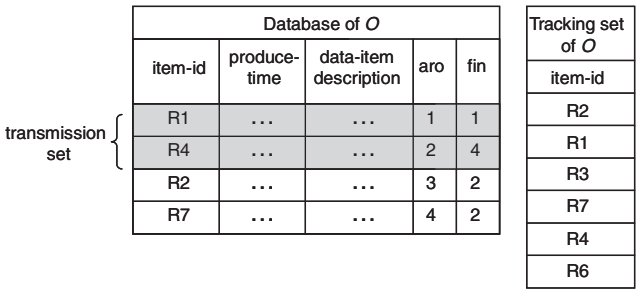


Fig. 3. Three sets of tuples (Database, Tracking set, Transmission set) maintained in a moving object O.

The examples set ES is collected as follows. When a data item R is transmitted, the sender attaches to R the NIV of R that is maintained by the sender. For each received data item, the receiver determines whether it is new to the receiver, and the respective NIV, along with the label "old" or "new", becomes an example in the receiver's ES set.

Old/new labeling and the tracking set. Now we elaborate on the old/new labeling of the examples collected by the previous framework. Observe that a data item may be received, then purged from the database, then received again. It would be false to label the data item "new" in the second receipt. But this is exactly what O would do if the label is determined by simply considering the database. Thus, O keeps a *tracking set*, in which each entry is the *item-id* (i.e., the unique identification) of a data item that has been received at O. An entry in the tracking set survives even when the corresponding data item is purged from the database. And when a data item is received, its item-id is searched in the tracking set for labeling, and thus "false" labeling is avoided.

Observe also that the size of each entry in the tracking set is only a few bytes, thus the tracking set can contain many more tuples than the database. Furthermore, as we discuss later in this subsection, the size of the tracking set can be bounded.

In summary, the *MAchine-LEarning-based Novelty rAnking* (MALENA) system distinguishes among four sets of tuples pertaining to data items. The tracking set described earlier pertains to all the data items ever received by a moving object; the database contains the data items that are currently stored by the moving object, which in turn is a subset of a tracking set; the transmission set is the subset of the database which is transmitted in an encounter. Object O also keeps the set ES of all the examples O has received. (As we will see later when we plug in the Bayesian machine learning system, O actually only needs to remember a limited amount of aggregate data about ES (e.g., the number of "new-data item" examples that have been received and so on), without remembering any actual example in ES.) The first three sets are demonstrated in Figure 3, and the examples set is demonstrated in Figure 4.

Formally, the pseudocode of the MALENA method is as follows.

Algorithm 1. MALENA, executed at a moving object O, when O encounters another moving object A

Input: DB_O – *the database at O*
TS_O – the tracking set at O
Q – the machine learning system at O
k – the size of the transmission set to be sent by O. // The value of k is determined by the bandwidth/energy allocation and the data item size.
M_O – the size of the database at O
G – the transmission set received from A
Output: F – transmission set sent from O
DB_O – updated database at O

1. **for each** R in DB_O, compute the novelty probability of R using Q
2. $F \leftarrow$ topK(DB_O, k)
   // Sort the data items in DB_O in decreasing order based on their novelty probabilities.
   // Select the top k data items (i.e., k items with highest probabilities).
3. Transmit the data items in F and their NIV's to A
4. Receive G the transmission set from A in exchange
5. **for each** R in G, **do** $aro \leftarrow arom$, where $arom = 1+$(the current maximum aro in DB_O).
6. **for each** data item R and its NIV **X** received from A, **do**
   Create an example (**X**, label) where label is "new" if the item-id of R does not exist in TS_O, and "old" otherwise.
   INSERT_EXAMPLE((**X**,label))
   // Add the example (**X**, label) to the examples set. INSERT_EXAMPLE is implemented by the machine learning system Q and it is where Q is actually trained. After the INSERT_EXAMPLE is finished, (**X**, label) is discarded. The INSERT_EXAMPLE procedure for Bayesian learning is described in the next subsection.
   if R is new to O, **then**
      Create an entry (R's-item-id, **Y**) in *TSO,* where **Y** is the NIV: ($arom$, $f_{in} = 1$)
   else // R is not new to O
      Update the NIV of R in TS_O by increasing its f_{in} value by 1**.**
7. $DB_O \leftarrow$ topK($DB_O \cup G$, M_O)
   // Sort the data items in G together with the data items in DB_O, in decreasing order of their novelty probabilities (computed by the machine learning system Q; see the intuitive framework at the beginning of this subsection); save the top M_O data items in DB_O. Reports in G that are labeled as "old" in step 6 are discarded directly, without participation in sorting.
8. The aro values of the data items in DB_O are adjusted to start from 1 and to eliminate the gaps created by the data items that did not fit in DB_O.

As we will see in the next subsection, the time complexity of the INSERT_EXAMPLE procedure is a constant. Assuming that the tracking set is accessed by using a hash table, step 6 can also be executed in constant time. Thus the complexity of the MALENA method is dominated by the sorts in steps 2 and 7, and is O(MlogM), where M is the number of data items in the database.

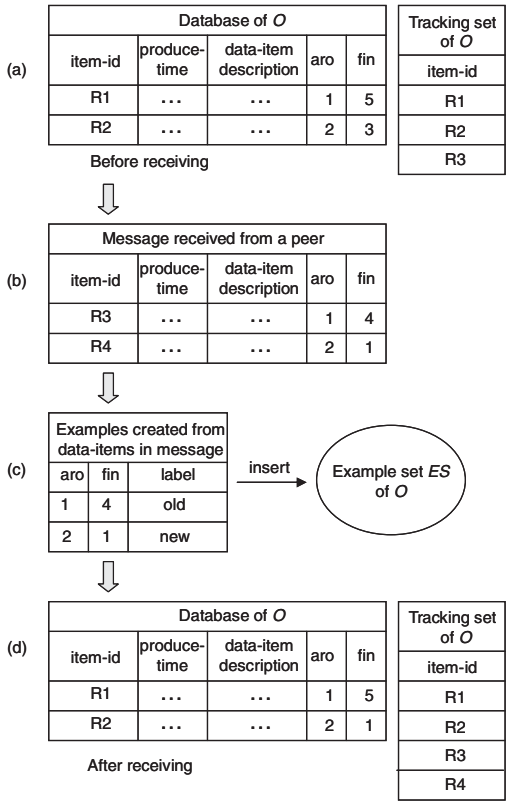


Fig. 4. An example for the MALENA method executed at a moving object O.

Example. Figure 4 illustrates the execution of the MALENA method at a moving object O. Figure 4(a) shows the database with a capacity of two data items, and the tracking set of O before it receives a transmission. At Figure 4(b) O receives a transmission from a neighbor which includes two data items R3 and R4. At Figure 4(c) O creates two examples from the received transmission and inserts them to O's example set. The NIV of R3 is labeled as "old" because O has received R3 before, as indicated by its tracking set at Figure 4(a). The NIV of R4 is labeled as "new" because O has never received R4 before. Observe that the example ((1,4), old) is created from R3 in the message, not from the R3 stored in O's database. For Bayesian machine learning, the examples set is simply a set of counters as will be explained in Section 3.4. Figure 4(d) shows the database and tracking set of O after the MALENA method ends. Notice that the f_{in} value R4 is set to be 1. The *aro* value of R4 is 2 because there are only two data items in O's database, and R4 arrives later than R1.

3.3 Bounding the Size of the Tracking Set

Recall that the purpose of the tracking set is to prevent incorrect labeling of received data items. We propose a method called *data item lifespan* to bound the storage allocated to the tracking set of a moving object O. The main idea

is that O removes a data item R from the tracking set when the lifespan of R ends, that is, when R has been purged by all the moving objects from their database. Intuitively, if R has already been purged by all the moving objects from their database, then R will not be received again, so there is no risk of incorrect labeling. Thus, in this case there is no reason to keep the tracking information for R. Obviously the lifespan of R is not known by an individual moving object O, but intuitively, O assumes that the lifespan of R ended when O has not received R from other moving objects for a sufficiently long time. More precisely, the lifespan of R is estimated based on the history of R in O's own database plus an extension period. Specifically, each entry R of the tracking set contains an element called the *expiration time*. The expiration time is O's estimate of R's lifespan. When the expiration time of R arrives, R is removed from the tracking set. The expiration time is updated as follows. When an entry R is added to the tracking set, its expiration time is initialized to be infinite. When R is purged from O's database, say at time *now*, the expiration time of R is updated to be R's-*produce-time* + (*now*-R's-*produce-time*) * 2. Recall that *produce_time* is the time at which R is produced (see Section 2.1). In other words, the lifespan of R is initially estimated to be: (the period of time starting when R is produced and ending when R is purged from O's database) * 2. That is to say, the global lifespan of R is estimated to be twice the lifespan of R observed locally at O. Each time R is received again, if R is still in the tracking set, then the expiration time of R is updated in the same fashion. Namely the expiration time of R is updated to be R's-*produce-time* + (*now*−R's-*produce-time*) * 2, where *now* is the time at which R is received again. (Observe that R is not going to be saved by O in the database according to step 7 of the MALENA method.) In other words, the lifetime of R is estimated to be twice the period of time starting when R is produced, and ending when R is last received by O.

There is one issue with the data item lifespan method: the clocks among the moving objects may be asynchronous, and the clock differences affect the calculation of the lifespan. This issue can be solved as follows. When two moving objects A and B exchange data items, they exchange their clocks as well. For each new report R received at A from B, A adjusts the produce-time of R using the difference between A's clock and B's clock. Thus the produce-time of R is given in terms of the local clock, and the clock difference does not affect the calculation of the lifespan of R at A.

The size of the tracking set can also be bounded by another method, called *global-DB-size*. With global-DB-size, a moving object keeps the tracking information for only the $N \cdot M$ most recently received data items, where N is the number of moving objects in the system, and M is the average database size among these objects. We postulate that having the size of the tracking set bounded by $N \cdot M$ should work almost as well as the infinite tracking set, because $N \cdot M$ gives the maximum number of distinct data items that can currently exist in the system. This postulate has been verified by our preliminary experiments. Experiments show that the global-DB-size method and the data item lifetime method provide identical performance but the latter bounds the tracking set to a smaller size.

3.4 Bayesian Learning System

In this subsection we introduce the Bayesian system as an instantiation of the machine learning system Q used by the MALENA method. The system can be plugged into the MALENA method for training (step 6) and data items ranking (steps 2 and 7).

At a high level, the Bayesian learning system maintains a set of counters (e.g., the number of "new-data item" examples with a particular $(aro,\ f_{in})$ pair). When an example is added, these counters are updated. When invoked for ranking, the system uses these counters to compute the probability that a data item will be new to a moving object encountered in the future.

Now we describe the Bayesian learning system in further detail. The description focuses on the case where the NIV consists of only two indicators, $(aro,\ f_{in})$, because our experiments have shown that adding more indicators will not change the performance significantly; and on the other hand it increases resource consumption, and complicates learning significantly.

The probability that a data item is new given its NIV (aro, f_{in}) is

$$p(new|(aro,\ f_{in})) = \frac{C_{new}(aro,\ f_{in})}{C(aro,\ f_{in})}, \tag{1}$$

where $C(aro,\ f_{in})$ is the number of examples for which the NIV equals to (age, f_{in}) and $C_{new}(aro,\ f_{in})$ is the number of "new-data item" examples for which the NIV equals to (aro, f_{in}).

The novelty probability of a data item with NIV (age, f_{in}) is then taken to be $p(new|age,\ f_{in})$ which is computed according to Eq. (1).

Given an example $((aro,\ f_{in})$, label), the INSERT_EXAMPLE procedure increases $C(aro,\ f_{in})$ by 1; and if the label of the example is "new", then $C_{new}(aro,\ f_{in})$ is also increased by 1. Thus, assuming that the counters of an $(aro,\ f_{in})$ pair are accessed using a hash table, the time complexity of the procedure is constant.

4. PERFORMANCE OF MALENA

In this section we report on the comparison of MALENA with three other data dissemination methods. In Section 4.1 we introduce the other methods, in Section 4.2 we describe the simulation method and data used; we also discuss the overhead of MALENA compared to the alternatives. In Section 4.3 we define and discuss the performance measure by which the methods are compared, and in Section 4.4 we report and discuss the comparison results.

4.1 The Methods Compared

In this subsection we compare the MALENA method introduced in Section 3, and three other naïve methods. One naïve method determines the novelty probability based on *aro* alone, and one determines the novelty probability based on f_{in} alone. They are called *aro*-ranking and f_{in}-ranking, and are similar to MALENA, except that no machine learning is used, and each one is based on a single indicator.

In *aro*-ranking, in an encounter, the data items are sorted in decreasing order of arrival, and the top-k, that is, the k items that arrived latest are exchanged. When saving the received data items in the database, if there is not enough space available, the bottom-k data items are discarded.

In f_{in}-ranking the data items are sorted in increasing order of f_{in} values, and the top-k, that is, the k items that have been received the least times are exchanged. When saving the received data items in the database, if there is not enough space available, the bottom-k data items are discarded.

The third naïve method is the random-spread mode of PeopleNet [Motani et al. 2005], in which data items are randomly selected for saving and transmission.

We compare the methods in two application scenarios. One is dissemination, in which a data item is to be delivered to as many moving objects as possible. Another scenario is routing, in which a data item is to be delivered to a single destination.

4.2 Simulation Method

In this section we discuss various components of the simulation system. In Section 4.2.1 we discuss the mobility and communication model, as well as the input data used for the simulation. In Section 4.2.2 we discuss the database size. Since we gave the other methods extra storage to account for the overhead of MALENA, this overhead is discussed in Section 4.2.2. Furthermore, the computational and energy overhead are also discussed there. In Sections 4.2.3 and 4.2.4 we discuss moving objects turnover and data items generation, respectively.

4.2.1 *Mobility and Communication.* We built a simulation system using Java. For mobility and communication, we used the trace collected by the Computer Laboratory at Cambridge University (see Hui et al. [2005]) at the IEEE InfoCom 2005 conference in Miami. This trace records the Bluetooth encounters by small devices, called iMotes, which were carried by 41 attendees (moving objects) in the conference for 3 days. The record for each encounter contains the IDs of the pair of moving objects involved in the encounter and the time interval during which they stay in contact (i.e., the *contact interval*). Within each contact interval the two objects are able to communicate using Bluetooth. We used the records of the second day for the time period from 8am to 4pm. During this time, on average each iMote experienced an encounter every 130 seconds, and the average contact interval lasts for 160 seconds; each iMote has an average of 1.2 neighbors at a point in time.

We superimposed data items exchanges over the iMotes trace as follows. At each second of a contact interval, the pair of encountering moving objects exchange their database (the reason for multiple exchanges during a contact interval is to disseminate to the neighbor new data items received or produced since the last exchange). For each exchange, a moving object A transmits a fraction q of its database to its peer B and vice versa. The communication is assumed to be reliable in the sense that there are no errors and retransmissions, and the simulation results were obtained in this context. In Section 4.4.5 we

study the case in which the communication is not reliable. The parameter q is used to model the constraints imposed by the bandwidth/energy allocation. A small q simulates a situation in which the power allocated by the user to P2P dissemination is low. So, the user assigns to mobile P2P a memory allocation (which may vary from one user to the next, that is, from one moving object to the next) and a fraction q of this allocation which bounds the transmission in each encounter. In our simulations q is the same for all the moving objects in one experiment but it may vary for different experiments.

One limitation of the iMotes trace is that it does not record the physical locations (x-y coordinates in a geospace) of the moving objects. This prevents our simulation system from using a network simulator such as ns-2 [ISI 2009] to incorporate the intricate details of the Bluetooth protocol. In such a simulator, physical locations have to be known for determining the operation of the network. As a result, our simulation system does not capture the communication factors such as radio attenuation, wireless collisions, and the resulting packet losses. However, remember that the focus of the article is data item prioritization (that can be combined with any store-and-forward data dissemination algorithm, and in particular any collision management mechanism), thus ignoring collisions should not affect the comparison among various ranking methods.

We also conducted simulations using a synthetic mobility model called Random Way-Point (RWP). The purpose is to evaluate the methods in a 802.11-like environment where the transmission range, 100 meters (see e.g., GigaFast [2009]), is much higher than in a Bluetooth environment. In the RWP simulations on average each moving object has 18 neighbors. The results are better than those obtained from the iMotes trace, but due to space considerations we need to omit most of these. Unless specified, the simulation setup and results presented in this section pertain to the iMotes trace.

4.2.2 *Database Size and the Overhead of MALENA*. The size of each data item is 3000 bytes. The number of data items that fit in the database of each moving object is randomly chosen from $[.5\times M,\ 1.5\times M]$, where M is fixed to be 100. In order for the comparison to be fair, in the simulations of PeopleNet, *aro*-ranking, and f_{in}-ranking methods, we expanded the database size of each moving object to match the space overhead of MALENA.

The space overhead of MALENA consists of three components. The first component is the NIV's attached to each data item in the database. The size of each NIV is 2 bytes (1 byte *aro* value and 1 byte f_{in} value). Thus the space overhead of the NIV's is $2{\cdot}M = 200$ bytes. The second component is the tracking set. We examined both the global-DB-size method and the data item lifespan method for bounding the size of the tracking set (see Section 3.3). It turned out that data item lifespan is strictly superior to global-DB-size in the sense that the former provides as good flooding performance as the latter but results in a smaller tracking set size. Using the global-DB-size method, we limit the size of the tracking set to be $N{\cdot}M$, where N is the number of moving objects in the system, that is, 41. Each entry of the tracking set is a 4 byte item-id. Thus the space overhead of the tracking set is $4{\cdot}N{\cdot}M = 16{,}400$ bytes. We use the size of the global-DB-size (i.e., 16,400 bytes) to count the space overhead

Table I. System Parameters and Their Values

Parameter	Symbol	Unit	Value
Mean of database size	M	data item	100 (In section 4.4.4 the database size varies. See details there)
Data item size		byte	3000 (In section 4.4.4 the data item size varies. See details there)
Data item production rate	f	items/second	0.5, 2
Delivery-time bound	c	minute	0, 1, 2, . . . , 60
Transmission fraction	q		0.02, 0.1
Turn-over rate	Low: no turn-over		
	High: normal distribution (300, 300) seconds for lifespan		
Injection percentage	High: normal distribution (80%, 0%)		
	Low: injected to a single random object		

of MALENA even when the data item lifespan method is used. This clearly favors the other compared methods. The third component is the storage of the Bayesian counters (i.e., $C(aro,\ f_{in})$'s and $C_{new}(aro,\ f_{in})$'s). The size of this storage straightly depends on the number of distinct $(aro,\ f_{in})$ pairs that are received by a moving object. Specifically, let M_{max} be $1.5 \times M = 150$, that is, the maximum size aro in the system. Let W_{max} be the maximum f_{in} value that may be received by an object. In our simulations W_{max} is 15. Then the maximum number of the $C(aro,\ f_{in})$ counters (or $C_{new}(aro,\ f_{in})$) is $M_{max} \cdot W_{max}$. The size of each counter is 2 bytes. Thus the space overhead of the Bayesian counters is $4 \cdot M_{max} \cdot W_{max} = 9,000$ bytes. Thus we increased the database size by 9 data items for PeopleNet, aro-ranking, and f_{in}-ranking.

The communication overhead of MALENA results from the NIV's attached to the transmitted data items. The size of each NIV is 2 bytes, and therefore the communication overhead of MALENA is $2 \cdot M \cdot q$ bytes (see Table I for symbol interpretation) per transmission, or at most $\frac{20}{3000}$ data items for a 3000-byte data item. We consider this overhead negligible for this data item size.

Finally, let us discuss the computational overhead of the MALENA method. It consists of two components. The first component is ranking in steps 2 and 7 of the method (see Section 3.2). Ranking of 100 keys takes at most 15,000 instructions, thus this component takes 30K instructions. The second component is the computational cost for maintaining the tracking set (which takes about 10 instructions per data item when the tracking set is accessed using a hash-table), and the cost of the INSERT_EXAMPLE procedure (which again takes about 10 instructions per data item when a hash-table is used to access a counter). Thus, the overhead of the second component is at most 2000 instructions, and overall the computational overhead of MALENA is less than 32K instructions per data items exchange.

Now we estimate the time and the energy consumed by the MALENA computation. We consider a typical PDA system, namely Dell X50 with a 624 MHz Intel PXA270 processor and a 1100 mAh, 3.7V Li-ion battery. Assuming that the CPU processes one instruction per clock cycle, each instruction takes $1/(624 \times 10^6)$ second. Thus 32K instructions take less than 50 μs. In other words, each execution of the MALENA computation takes less than 50 μs.

Now consider the total energy consumed by the MALENA computation at a moving object throughout an 8-hour period. Since each moving object has one exchange with each one of its neighbors per second, and on average each moving object has 1.2 neighbors (see Section 4.2.1), each moving object has 1.2×28800 exchanges during the 8 hours. The CPU consumption is at most 5.3Watt [Dell 2009]. Thus the total energy consumed by the MALENA computation is at most $50 \times 10^{-6} \times 1.2 \times 28800 \times 5.3 = 9$ Joules, that is, 0.06% of the total battery capacity ($3.7 \times 1.1 \times 3600 = 14652$ Joules). It is noted that the preceding calculation considers only the energy overhead introduced by the MALENA method (for machine learning and ranking). It does not include the overhead of the gossiping protocol itself, such as the communication energy, the device and service discovery energy, and the extra energy costs of waking up a sleeping kernel. Presumably, these energy costs are incurred by any gossiping protocol.

4.2.3 *Moving Objects Turnover.* Each moving object has a lifespan determined by the simulation system. When the lifespan of an object O expires, its database, tracking set, and Bayesian counters are reinitialized. This simulates O exiting the system and a new object entering it. Thus the number of *live* moving objects is fixed during a simulation run.

We consider two cases in terms of the lifespan. In the first case, the lifespan of each object equals to the length of the time period of the whole simulation run. This case is referred to as *low turnover* as it represents an environment in which the set of moving objects is fixed, that is, no new objects join in the system and no existing objects leave. Places such as conference halls or sports stadiums are practical examples of low turnover environments. In the second case, the lifespan of each object follows a normal distribution with the mean of 300 seconds and the standard deviation of 300 seconds. This case is referred to as *high turnover* as it represents an environment in which objects frequently join in and exit. Places such as train stations and airport terminals are examples of high turnover environments.

4.2.4 *Data Items Injection.* In each simulation run f data items are produced every second. The number f is referred to as the *data item production rate*. In the routing scenario, a destination for the data item is generated together with the item. The destination of a data item is randomly chosen from the objects that have ever entered the system and the objects that will enter the system in the future. When produced, the data item is injected instantaneously to a percentage of moving objects which become its producers. We consider two cases. In the first case, the data item is injected to a single moving object which is randomly selected. This is referred to as the *low injection* case. It simulates, for example, a matchmaking profile being generated by a single user. In the second case, the percentage of moving objects that learn the data item follows a normal distribution with the mean of 80% and the standard deviation of 20%. This is referred to as *high injection* case. An example of a high injection case would be where some users subscribe for stock quotes notifications or news

alerts via the cellular network and then they share the information with other users via peer-to-peer data dissemination.

All the system parameters and their values are listed in Table I. Each parameter configuration is run once to get the curve for that configuration.

Now let us justify that our communication setup is reasonable in the sense that the bandwidth consumed by pairwise exchanges fits within the capacity of Bluetooth. In our experiments, each moving object exchanges with each of its neighbors at each second, each exchange is finished within 1 second, and on average each moving object has 1.2 neighbors at any point in time. Since the maximum database size is 150 and the transmission fraction is at most 0.1, each moving object communicates (transmitting and receiving) at most $3000\text{bytes/item} \times 150\text{items} \times 0.1 \times 1.2 \times 2 = 108\,\text{Kbytes} = 864\text{Kbits}$ per second, well within Bluetooth's theoretical bandwidth limits. In other words, the whole database can be transmitted within one second. And indeed all the iMotes contact intervals are at least one second. In addition, in the iMotes traces, each contact interval must have excluded the device discovery time and the service discovery time because the contact interval started after the device and the service are discovered.

4.3 Performance Measure

As the performance measure for comparison of the four methods we take a combination of throughput and delivery-time; it averages the number of distinct data items received by a moving object within a certain time limit c. This is similar to the way an academic department is evaluated according to the percentage of its students that graduate within 4 years.

More specifically, the measure takes the average number of data items with delivery-times smaller than c that are successfully delivered to a moving object. In the routing scenario, only those data items those are addressed to and received by the destination object are counted. This measure is called the *delivery-time bounded throughput*, or *throughput*, and c is called the *delivery-time bound*. By varying the value of c, we evaluate the throughput of a method for different delivery-time constraints.

Finally we define the delivery-time of a data item R received at a moving object O. Intuitively, the delivery-time interval starts at the time at which O is introduced in the system or R is introduced, whichever is later, since both must be present for O to receive R. The interval ends when O receives R. Formally, let O receive a data item R for the first time at time t. If R is produced after O is introduced in the system, then the *delivery-time* is the length of the time period starting at the production-time of R and ending at time t; otherwise it is the length of the time period starting at the time O is introduced and ending at time t. This concept is illustrated by Figure 5.

Observe that it is possible to evaluate performance by other measures, such as the probability that a data item is disseminated to all the moving objects in the system, but we chose the more traditional throughput and delivery-time measures.

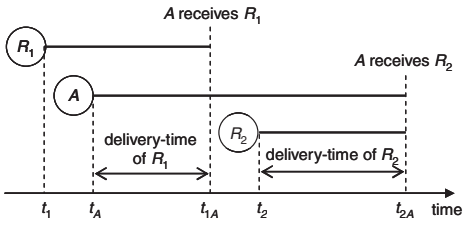


Fig. 5. Illustration of delivery-time calculation. Data items R_1 and R_2 are produced at times t_1 and t_2 respectively. Object A is introduced at time t_A. The delivery-time of R_1 is $t_{1A} - t_A$. The delivery-time of R_2 is $t_{2A}-t_2$.

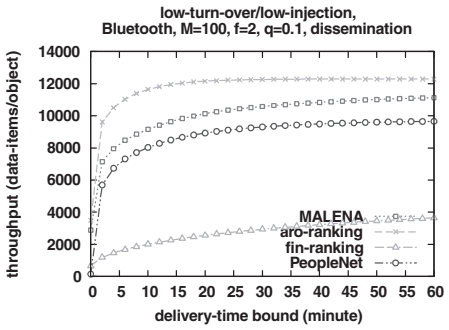


Fig. 6. Experiment 1: L/L environment for the Bluetooth scenario.

4.4 Simulation Results

In this section we discuss the results in the dissemination scenario. In Sections 4.4.1 and 4.4.2 we compare the four ranking methods in a *low-turnover/low-injection* environment and a *high-turnover/high-injection* environment, respectively. In Sections 4.4.3 and 4.4.4 we study the impact of transmission fraction and data item size, respectively. We evaluate the impact of the communication reliability in Appendix B. The results for the routing scenario are similar, thus they are omitted. The results presented in Figures 6, 7, and 8 were obtained using the data item lifespan tracking set bounding method. The results in all the other figures were obtained using the global-DB-size method.

4.4.1 *Low-Turnover/Low-Injection (L/L) Environment.* Experiment 1 (Figure 6) shows the throughput as a function of the delivery-time bound for different methods in the L/L environment for the Bluetooth scenario. Observe that *aro* is a good indicator in the L/L case, as *aro*-ranking performs at least three times better than f_{in}-ranking, for the entire range of the delivery-time bound. MALENA closely follows *aro*-ranking. Intuitively, in the L/L case, the set of the moving objects in the system is fixed, and therefore a data item that has stayed at a moving object for a long time is likely to be known by many objects.

Observe that with MALENA, each moving object receives about 11,000 new data items with delivery-times 60 minutes or less. On the other hand, two data items are generated per second in the whole system. Thus, the performance of

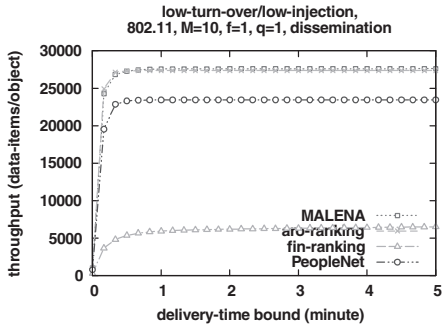


Fig. 7. Experiment 2: L/L environment for the 802.11 scenario.

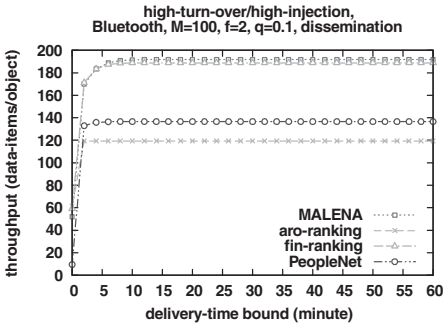


Fig. 8. Experiment 3: H/H environment for the Bluetooth scenario.

MALENA is 20% of the ideal case (i.e., when a central server is employed, and it instantaneously broadcasts each new data item to all the moving objects; in this case a moving object would have received 57600 data items). In addition, 75% of the data items received by MALENA are younger than 5 minutes, well within reasonable limits for the applications considered.

Experiment 2 (Figure 7) shows the results with the Random Way-Point (RWP) mobility model with 802.11 in the L/L case (Recall that with 802.11 each moving object has 18 neighbors on average, in contrast to 1.2 for the Bluetooth case). The comparison among the methods is similar to that with the iMotes trace (Bluetooth) except that MALENA performs even better.

4.4.2 *High-Turnover/High-Injection (H/H) Environment.* Experiment 3 (Figure 8) shows the throughput as a function of the delivery-time bound for the different methods in the H/H environment. Observe that f_{in} is a good indicator in the H/H case, as it performs much better than *aro*-ranking (by up to 30%). MALENA is almost as good as f_{in}-ranking. The fact that the *aro* indicator performs poorly can be explained as follows. Since the turnover rate is high, even the old data items are "new" to the moving objects that have newly joined in. On the other hand, since the injection percentage is high, "young" data items are known to the moving objects that have been injected with them. Thus, *aro* becomes a bad indicator of novelty.

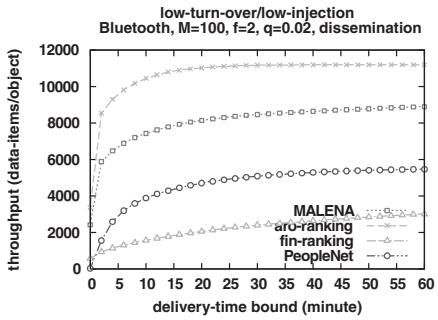


Fig. 9. Experiment 4: Low transmission fraction in the L/L environment for the Bluetooth scenario.

In terms of the delivery-time, observe that 95% of all the data items received by MALENA are younger than 5 minutes.

4.4.3 *Impact of the Transmission Fraction.* Experiment 4 (Figure 9) differs from Experiment 1 (Figure 6) in the value of the transmission fraction (q). Specifically, in Experiment 1 on average ten data items are exchanged per encounter, whereas in Experiment 4 two data items are exchanged per encounter. It can be seen that the advantage of MALENA over PeopleNet is higher when the transmission fraction is lower. This suggests that MALENA is particularly useful in an environment where the available bandwidth and power are low. Intuitively, when the transmission fraction is low, the prioritization becomes more critical and random ranking suffers. The same can be observed in the H/H environment.

4.4.4 *Impact of the Data Item Size.* In all the previous simulations, the size of each data item is fixed to be 3000 bytes. In this subsection we vary the data item size and study its impact to the performance of the ranking methods. First let us discuss how we take the space and communication overhead of MALENA into account in the simulations conducted in this subsection. We fix the size of the database of MALENA to be 300K bytes. The space overhead of MALENA is taken into account as described in Section 4.2.2, with M fixed to be $300\text{K}/L$; L is the data item size (excluding NIV).

The communication overhead of MALENA is taken into account as follows. Denote the cardinality of a transmission set by k. For the MALENA method, $k = 300\text{K} \cdot q/L$. Since each transmitted data item is attached with a 2 byte NIV, the communication overhead of MALENA is $2 \cdot k$ bytes per transmission, or $2 \cdot k/L$ data items. For *aro*-ranking, f_{in}-ranking, and PeopleNet, k is increased to compensate the communication overhead of MALENA. The increase is as follows. When $2 \cdot k/L$ is greater than 1, k is increased by $\lfloor 2 \cdot k/L \rfloor$ data items. When $2 \cdot k/L$ is smaller than 1, for each transmission, k is increased by 1 with probability $2 \cdot k/L$.

Experiment 5 (Figure 10) corresponds to Experiment 1 in terms of parameters-configuration, and shows the throughput as a function of the data item size for the four methods in the Bluetooth L/L environment. The delivery-time bound is fixed to be 900 seconds since the throughput of each method

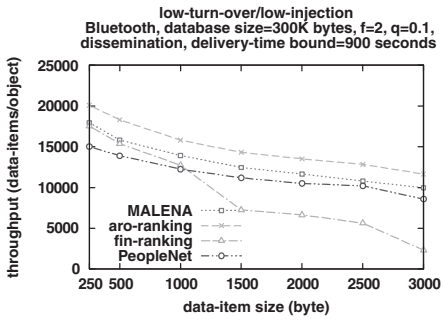


Fig. 10. Experiment 5: Impact of the data item size. L/L environment for the Bluetooth scenario.

except f_{in} is stable beyond this bound. From Experiment 5 it can be seen that, as the data item size decreases, the throughput increases for all the methods. Furthermore, the advantage of MALENA over f_{in} decreases as the data item size decreases. This suggests that in a L/L environment, MALENA is particularly useful when the data item size is big.

However, the difference between MALENA and *aro* or PeopleNet does not change much with data item size. This is because the communication overhead of MALENA remains very small even when the data item size is small. In fact, MALENA only adds two bytes to each transmitted data item. Thus even when the data item is 250 bytes (the smallest size tested), the communication overhead of MALENA is only $2/250 = 0.8\%$.

Summary. In each environment MALENA comes close or exceeds the best indicator for that environment. This is important since a moving object usually does not know the parameters of its current environment, and furthermore, they change over time.

5. NOVELTY RANKING IN PUSH-PULL GOSSIPING

In this section we study novelty ranking in another gossiping protocol, called *push-pull gossiping*. Let W be the maximum number of bytes that can be transmitted by a moving object O at each exchange, according to the bandwidth/energy allocation. W is called the *communication bound*, and it is computed based on the database size at O, the data item size, and the transmission fraction, q. When O encounters a moving object A, a push-pull gossiping is executed as follows.

(1) O informs A what are the ids of the data items that O has ever seen. This is done by O sending its tracking set (denoted TS_O) to A.
(2) Symmetrically O receives TS_A the tracking set of A.
(3) O transmits to A the data items that are new to A, that is, those data items whose ids do not appear in TS_A. If all the data items that are new to A do not fit the remaining transmission size (the remaining transmission size is $W - \mathrm{size}(\mathrm{TS}_O)$, where $\mathrm{size}(\mathrm{TS}_O)$ is the size of TS_O in number of bytes), then

the data items are ranked according to their novelty probabilities and the top ones that fit in $W - \text{size}(\text{TS}_O)$ bytes are transmitted to A.

(4) Symmetrically O receives data items from A. O ranks the data items received from A together with the data items in the database, and saves the top M_O data items in the database.

The rest of this section is organized as follows. In Section 5.1 we introduce MALENA+2P, the push-pull version of MALENA. In Section 5.2 we compare MALENA+2P and the push-pull versions of *aro*-ranking, f_{in}-ranking, and PeopleNet. We compare MALENA+2P with the push version of MALENA in Appendix C.

5.1 MALENA+2P: The Push-Pull Version of MALENA

First, observe that MALENA does not attempt to estimate the probability of novelty to the peer participating in the current encounter, but to a *future* encountered peer. Thus, the fact that the push-pull gossiping transmits only new data items does not obviate the need for estimating novelty probability.

Second, observe that MALENA could be applied to the push-pull gossiping without any changes. In other words, a moving object O creates examples and trains the machine learning system Q (see Section 3.2) when O receives data items from a peer A (after O informs A of its tracking set). However, there is a drawback to this straight forward carry-over. Since A only transmits new data items to O, all the examples created by O will be labeled as "new." Thus Q will be trained with only positive examples, instead of with both positive and negative examples as in the push case. This may compromise the learning capability of Q. In some cases it simply fails the machine learning system. For example, the Bayesian method used in this article does not work when there are only positive examples.

In this subsection we adjust MALENA such that it learns from both positive and negative examples in the push-pull gossiping. The idea is that, instead of O learning upon receiving data items from A, it learns upon receiving the tracking set of A. Specifically, for each data item R in O's database, one example is created using the NIV of R: the example is labeled as "new" if R is new to A (i.e., the id of R does not appear in A's tracking set); and "old" otherwise. We call this method MALENA+2P and formally describe it next.

Similar to the MALENA method, the complexity of MALENA+2P is O(MlogM), where M is the number of data items in the database.

Let us explain step 3, where the method increases the f_{in} value by 1 for each data item in O's database that is known by A. Observe that in the push-pull gossiping, since a peer only transmits new data items to O, O will not receive a data item R more than once. Thus the original definition of f_{in} (i.e., the number of times R is received) will give at most 1 for the value of f_{in}, regardless of how many peers encountered by O have known R before the encounters. This clearly deviates from the purpose of the f_{in} as a novelty indicator. In the MALENA+2P we address this issue by redefining f_{in} to be the number of times O has *sighted* R's item-id in the tracking sets of the peers it has encountered. In step 3, the method increases the f_{in} value of R by one if it sights R in the tracking set of A.

Algorithm 2. The MALENA+2P method, executed at a moving object O, when O encounters another moving object A.

1. Receive from A the tracking set TS_A of A.
2. For each data item R in O's database and its NIV **X**,
   a. Create an example (**X**, label) where label is "new" if the item-id of R appears in TS_A, and "old" otherwise.
   b. Invoke INSERT_EXAMPLE((**X**, label)) to add the example (**X**, label) to the examples set.
   c. If the item-id of R does not appear in TS_A, mark R as "candidates for transmission".
3. For each item-id i in TS_A, if i appears in the database of O, then update the NIV of i (in O's database) by increasing its f_{in} value by 1.
4. For each data item R marked as "candidates for transmission", compute the novelty probability using the machine learning system denoted Q. Then sort these data-items in decreasing order based on their novelty probabilities, and select the top k data items (i.e., k items with highest probabilities); the value of k is determined by the data-item size and the remaining transmission size (recall that the remaining transmission size is the communication bound W minus the size of O's tracking set).
5. Transmit the selected data-items to A
6. Transmit the tracking set of O to A and receive a set of data items from A. All the received data items have the same *aro*, denoted *arom*, where $arom = 1+$ (the current maximum *aro* in O's database). All the received data-items have $f_{in} = 1$.
7. For each data-item R received from A, insert R's-item-id to the tracking set.
8. Sort the data items received from A in step 6 together with the data items in the database, in decreasing order of their novelty probabilities (computed by the machine learning system Q); save the top M_O data items in the database.
9. The aro values of the remaining data items are adjusted to start from 1 and to eliminate the gaps created by the data-items that did not fit in the database.

5.2 Comparison of MALENA+2P and the Push-Pull Versions of *aro*-ranking, f_{in}-ranking, and PeopleNet.

We conducted simulations to compare MALENA+2P, f_{in}-ranking+2P, *aro*-ranking+2P, and PeopleNet+2P where f_{in}-ranking+2P, *aro*-ranking+2P, and PeopleNet+2P are the push-pull versions of f_{in}-ranking, *aro*-ranking, and PeopleNet, respectively. For f_{in}-ranking+2P, f_{in} is defined to be the number of times a moving object has sighted the item-id in the received tracking sets (see Section 5.1). The simulation setup is similar to that for the push gossiping (see Section 4.2). In Figures 11 and 12 we present the results.

Experiment 12 (Figure 11) shows the results in the low-turnover/low-injection (L/L) case for one parameter configuration (the results with other parameter configurations point to similar conclusions and are omitted due to space considerations). Experiment 13 (Figure 12) corresponds to Experiment 3 in the push case in terms of parameters configuration, and shows the results in the high-turnover/high-injection (H/H) case. From these experiments it can

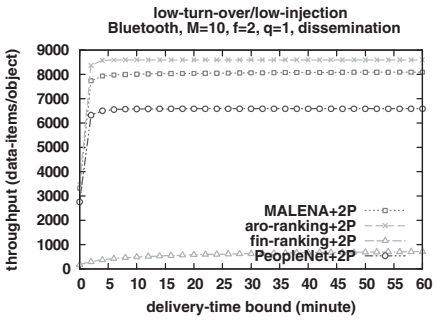


Fig. 11. Experiment 12: L/L environment for the Bluetooth scenario.

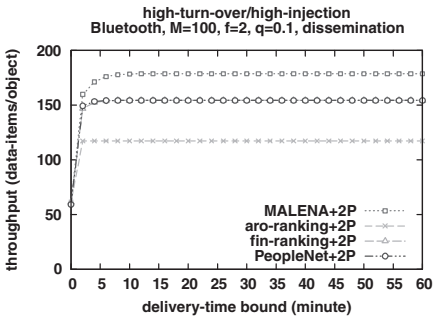


Fig. 12. Experiment 13: H/H environment for the Bluetooth scenario. (Note that the curve of fin-ranking+2P and that of PeopleNet+2P coincide).

be seen that, again, in each individual environment MALENA+2P approaches or outperforms the best one.

6. RELATED WORK

Static-sensor databases. A database approach has been applied to static sensor networks (see e.g., Yao and Gehrke [2003], Hellerstein et al. [2003], Muller and Alonso [2006], Lian et al. [2005], Coman et al. [2005], Deligiannakis [2004], Xu et al. [2005], and Kotidis [2005]). All these methods require that a certain routing structure, for example, a tree, be established in the network. Each node aggregates the results returned by its downstream neighbors and its own result, and forwards the aggregation result to its upstream neighbor. However, in our environment, due to the dynamic/partitionable network topology, such a structure is hard to maintain. MALENA relies on opportunistic interactions between encountering mobile nodes and thus is disruption tolerant.

Some works in static sensor networks involve negotiations between neighboring sensors (see e.g., Medidi et al. [2005]). The objective is to ensure that the data items wanted (i.e., unknown data items in our context) by the receiver are transmitted. However, the negotiation considers only what items are new to the receiver, whereas we consider what are likely to be new data items to future encountered moving objects. In other words, we consider the utility to moving

objects functioning as brokers, not consumers. In this article we assume that the consumer function of devices is handled orthogonally and independently of the broker function.

Novelty ranking. Previous work on dissemination in mobile ad hoc networks and routing in disruption-tolerant networks has studied various heuristics for ranking packets based on their novelty probability. For example, Ni et al. [1999] propose two schemes, one using the number of times a packet has been received at a node (i.e., f_{in}), and another using the distance between the sender and the receiver. Hayashi et al. [2003] and Burges et al. [2006] consider the number of hops. Vahdat et al. [2000], Costa et al. [2004], and Chaintreau et al. [2006] use FIFO (equivalent to arrival order). Thus, in each of these schemes, a single individual novelty indicator is used. In contrast, our method combines various indicators and it adapts to the environment via machine learning.

In Burges et al. [2006], a node uses acknowledgements from destination nodes to purge from its buffer those packets that have already been received by their destinations. This method is designed for DTNs, that is, environments in which a message is sent to a specific destination; but it does not work for dissemination where the destination ids are not known a priori to the sender. Our method is applicable for both cases.

Prioritization in mobile peer-to-peer data dissemination. Ranking data items for memory (cache) management and bandwidth management in mobile peer-to-peer networks has been studied in a number of works. In Motani et al. [2005] and Wolfson et al. [2006a; 2006b] data items are ranked according to their popularity and the assessment of popularity is enabled via the dissemination of queries. In Wolfson et al. [2007] the rank of a data item is jointly determined by its popularity and size. In Sailhan and Issarny [2002] the rank of a data item is jointly determined by its popularity, reliability, and size. In Xu et al. [2004] data items are ranked based on their spatio-temporal relevance. The relevance indicates, for example, the probability that a parking slot reported by the data item will be still available when the user reaches it. In Perich et al. [2004], Hull et al. [2006], and Datta [2004] data items are ranked based on an abstract utility function which is to be defined by specific applications. Our present work does not compete with these methods. Instead, it adds an orthogonal consideration, the novelty probability which, as mentioned in the Introduction, can be combined with other priority metrics.

Resource discovery and publish/subscribe in mobile ad hoc networks. Resource discovery (e.g., Pucha et al. [2004] and Frank and Karl [2004]) and publish/subscribe (e.g., Huang and Garcia-Molina [2004] and Zhang and Hu [2005]) in mobile ad hoc networks are often implemented by building a routing structure for resource information dissemination. Consequently they can be inefficient, particularly in networks that are prone to frequent topology changes due to mobility and turnover. In such an environment, either a lot of communication has to be expended to keep the routing structure up to date, or the routing structure rapidly becomes obsolete and misses many matches. Our method does not rely on any routing structures and it adapts to the environment via machine learning.

Peer-to-peer networks. In current P2P networks (see e.g., Halevy et al. [2004] and Balazinska et al. [2002]), the requester sends a query which is propagated through a specific community (say Napster) and if a match is found, the response is routed back to the requester. The main difference between static and mobile P2P (sensor) networks is that in the mobile case nodes can communicate only when they are in physical proximity, whereas static P2P networks are not so constrained. This fundamental difference makes most static approaches inapplicable. For example, Michel et al. [2006] also use the novelty concept to reduce duplicate transmissions in a P2P environment. In their case, the ids of the data items stored at each peer O are summarized by a synopsis. All the synopses are published to a distributed directory. When another peer P issues a query, P looks up the synopses and sends the query to the peers that have minimum overlap in their synopses. This method clearly does not fit a mobile environment, due to the overhead of directory maintenance and access, and the constant change of synopses.

7. CONCLUSION AND FUTURE WORK

In this article we addressed the dissemination problem in which data items are flooded to all the moving objects in a mobile ad hoc network. Dissemination occurs via peer-to-peer communication using short-range wireless networks such as 802.11. We first considered the problem in an environment where all the trajectories of moving objects are known a priori at a centralized location. We showed that if memory and bandwidth are bounded at moving objects, then the problem of determining whether a set of data items can be disseminated to all the moving objects is NP-complete; otherwise it is tractable.

We then proposed a method for dissemination of data items based on machine learning. The proposed method, MALENA, is used by each moving object to prioritize data items in terms of their probabilities of being new to future recipients. The machine learning system is progressively trained by received data items.

We experimentally evaluated MALENA using Bayesian as the machine learning system, and compared it with three other methods. For the gossiping protocol we considered both push gossiping and push-pull gossiping. The simulation used two inputs, one real and the other synthetic. The real one was collected as Bluetooth sightings at a convention, and the synthetic one was generated using the random-way-point modeling an 802.11 network. The measure of comparison is new, combining the average number of data items received by a moving object and their average age when received. The experimental results show that in each individual environment MALENA approaches or outperforms the best method for that environment and outperforms the inferior one by up to five times. This is important since the best method depends on the global environment, and the global environmental parameters change and are usually not known to a moving object. The results hold for an application that has become popular recently, disruption tolerant networks (see Burns et al. [2005]), in which a data item is sent to a single destination rather than disseminated.

Much more research still needs to be carried out to determine the optimal learning method for different gossiping protocols. For example, instead of a pairwise exchange, the moving objects may communicate by periodic broadcasts (see, e.g., Wolfson et al. [2006b]). Does the same learning mechanism work well in these gossiping protocols? Do other (than Bayesian) machine learning techniques perform better?

In this article we studied how to transmit during a peer-to-peer interaction. Another important issue is *how much to transmit*. In other words, if the total bandwidths of the neighboring moving objects are not sufficient to exchange all data items, how to allocate bandwidth among these objects? In recent work [Wolfson et al. 2006b; Wolfson and Xu 2007] we addressed this issue. The general approach is that each moving object dynamically adjusts the transmission size based on the length of the period of time between subsequent peer-to-peer interactions, to deliver reliably (i.e., without collisions) the maximum number of data items.

In general, there can be many gossiping protocols and they can be compared in terms of the dissemination performance, the amount of communication, and the number of rounds of communication. However, developing all these gossiping protocols and finding which is the best is clearly out of the scope of this article. In this article we have shown that MALENA improves performance in two gossiping protocols.

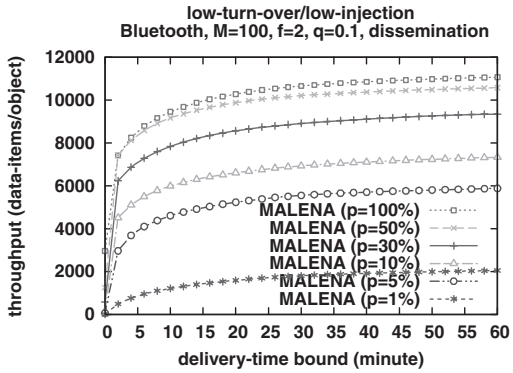
Another important future research issue is to augment MALENA such that it adapts to a changing environment, for example, when an object moves from an L/L environment to an H/H one. In machine learning, changing environments are often handled by sliding windows of fixed or adaptive size on the training data (see, e.g., Widmer and Kubat [1996] and Wang et al. [2003]) or by weighting data or parts of the hypothesis according to their age and/or utility for the prediction task (see, e.g., Taylor et al. [1997]).

APPENDIXES

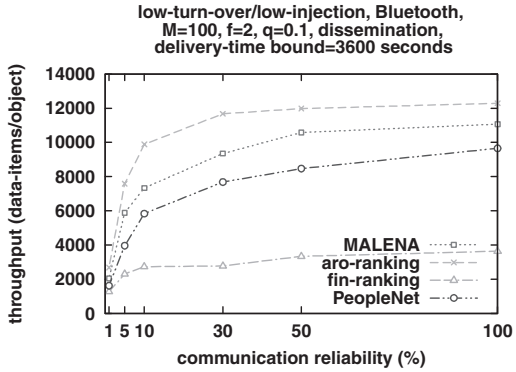
A. PROOF OF THEOREM 2

THEOREM 2. *Given bounds on memory and bandwidth for processors, the MODD problem is NP-complete.*

PROOF. The reduction is from the Multiprocessor Scheduling (MS) problem (see Deligiannakis et al. [2004]). Given an instance of MS we create an MODD instance as follows (the construction uses the Deligiannakis et al. [2004] notation for MS). We introduce a set S of objects, each corresponding to a processor in MS; and a number of data items that is equal to the number of tasks. The moving objects in S are always within transmission range of each other, and each has infinite bandwidth. Then we introduce a new moving object s that is not in S, it is static within the transmission range of all the moving objects in S for the period of time 0 to D (but not afterwards), and it receives all the data items at time 0. Due to bandwidth constraints, the time for s to send a data item to a member of S is equal to the length of the corresponding task. It is easy to show that the MS instance has a solution if and only if the MODD instance has a solution. □



(a) The throughput of MALENA as a function of the delivery-time bound for various communication reliabilities



(b) The performance of the four methods as a function of the communication reliability (delivery-time bound=3600 seconds)

Fig. 13. Experiment 11: Impact of the communication reliability. L/L environment for the Bluetooth scenario.

B. IMPACT OF THE COMMUNICATION RELIABILITY

For all the experiments conducted so far, we assumed that the communication is perfectly reliable and each exchange always succeeds. In this subsection we study the case in which the communication is not reliable. Particularly, we let each exchange succeed with probability p. If the exchange does not succeed, no data items are received by either of the two interacting moving objects. p is referred to as the communication reliability. p varies from 1% to 100%.

Experiment 11 (Figure 13) shows the impact of the communication reliability in the L/L environment for the Bluetooth scenario. Figure 13(a) shows the throughput of MALENA as a function of the delivery-time bound for various communication reliabilities. Figure 13(b) shows the performance of the four methods as a function of the communication reliability, with the delivery-time bound fixed to be 3600 seconds. From Figure 13(b) it can be seen that the four curves are quite flat when the communication reliability is higher 30%. This means that all the four methods are tolerant to communication failures.

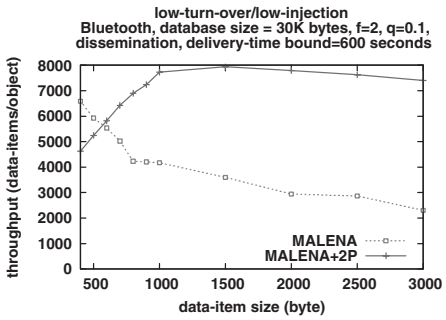


Fig. 14. Experiment 14: Comparison of MALENA+2P and MALENA in a L/L environment for the Bluetooth scenario.

Particularly, when 50% of the exchanges fail, the throughput of MALENA drops only 5%.

C. COMPARISON OF MALENA+2P AND MALENA

Observe that in MALENA+2P, two encountering moving objects exchange tracking sets so that only the new data items are exchanged. In other words, the communication cost of the data item exchange is reduced, but the tracking set exchange is added. Presumably, when the size of each data item is big, then the overhead of the tracking set communication is relatively small compared to the reduction of data item communication, and MALENA+2P is efficient. On the other hand, when the data item size is small, then MALENA+2P may not be worthwhile. Our simulations results validated this intuition and further identified the cut-off sizes (i.e., the data item sizes beyond which MALENA+2P outperforms MALENA) for various parameter configurations. Figure 14 shows throughput as a function of the data item size for MALENA+2P and MALENA in a Bluetooth L/L environment. The delivery-time bound is fixed to be 600 seconds since the throughput of either method is stable beyond this bound.

From Experiment 14 (Figure 14) it can be seen that the cut-off size is about 600 bytes for the given parameter configuration. Observe that the throughput of MALENA increases as the data item size decreases. This is because with a smaller data item size, more data items can be stored in the database and transmitted during an exchange. The behavior of MALENA+2P is a little bit more intricate. The throughput of MALENA+2P increases as the data item size decreases from 3000 bytes to 1500 bytes, and then it decreases as the data item size decreases from 1500 bytes to 400 bytes. This is explained as follows. When the data item size increases, two effects, positive and negative, are generated. On one hand, more data items can be stored in the database and be potentially transmitted; on the other hand, the overhead of the tracking set exchange increases because there are more tuples in the tracking set (Recall Section 3.2; the number of entries in the tracking set is proportional to the number of tuples in the database). The increase of the tracking set leaves less room for the data item exchange. Experiment 14 shows that the data item size that optimally trades off these two effects is 1500 bytes.

The cut-off size may vary depending on the environmental parameters such as the transmission fraction and the data item production rate. For example, for the parameter configuration used in Experiment 14, when the data-time production rate (f) is changed from 2 to 10, the cut-off size is changed from 600 bytes to 300 bytes. Thus our simulation results enable the moving objects to decide whether to use MALENA or MALENA+2P in a specific environment. However, remember that whether to use push gossiping or push-pull gossiping is an issue in selecting the dissemination algorithm, whereas the focus of the article is the machine learning method which, as we have shown, can improve either.

REFERENCES

Balazinska, J., Balakrishnan, H., and Karger, D. 2002. INS/Twine: A scalable peer-to-peer architecture for intentional resource discovery. In *Proceedings of the International Conference on Pervasive Computing*. 195–210.

Birman, K., Hayden, M., Ozkasap, O., Xiao, Z., Budiu, M., and Minsky, Y. 1999. Bimodal multicast. *ACM Trans. Comput. Syst. 17*, 2, 41–88.

Burges, J., Gallagher, B., Jensen, D., and Levine, B. 2006. MaxProp: Routing for vehicle-based disruption-tolerant networking. In *Proceedings of the 25th IEEE Conference on Computer Communications*. 1–11.

Burns, B., Brock, O., and Levine, B. N. 2005. MV routing and capacity building in disruption tolerant networks. In *Proceedings of IEEE InfoCom*. 398–408.

Chaintreau, A., Hui, P., Crowcroft, J., Diot, C., Gass, R. and Scott, J. 2006. Impact of human mobility on the design of opportunistic forwarding algorithms. In *Proceedings of the 25th IEEE Conference on Computer Communications*. 1–13.

Costa, P., Migliavacca, M., Picco, G., and Cugola, G. 2004. Epidemic algorithms for reliable content-based publish-subscribe: An evaluation. In *Proceedings of the 24th IEEE International Conference on Distributed Computing Systems (ICDCS)*. 552–561.

Datta, A., Quarteroni, S., Aberer, K. 2004. Autonomous gossiping: A self-organizing epidemic algorithm for selective information dissemination in wireless mobile ad-hoc networks. In *Proceedings of the 1st International IFIP Conference on Semantics of a Networked World (ICSNW)*. 126–143.

Deligiannakis, A., Kotidis, Y., and Roussopoulos, N. 2004. Hierarchical in-network data aggregation with quality guarantees. In *Proceedings of the 9th International Conference on Extending Database Technology (EDBT)*. 658–675.

Dell. 2009. http://www.dell.com/downloads/global/corporate/environ/Axim_X50.pdf

Dobson, S., Denazis, S., Fernandez, A., Gaiti, D., Gelenbe, E., Massacci, F., Nixon, P., Saffre, F., Schmidt, N., and Zambonelli, F. 2006. A survey of autonomic communications. *ACM Trans. Auton. Adaptive Syst. 1*, 2, 223–259.

Frank, C. and Karl, H. 2004. Consistency challenges of service discovery in mobile ad hoc networks. In *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*. 105–114.

Friedman, R., Gavidia, D., Rodriques, L., Viana, A., Voulgaris, S. 2007. Gossiping on MANETs: The beauty and the beast. *ACM SIGOPS Oper. Syst. Rev. 41*, 5, 67–74.

Garey, M. and Johnson, D. 1979. *Computer & Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman.

GigaFast. 2009. http://www.gigafast.com/products/product_pdf/WF748-CUI.pdf

Halevy, A., Ives, Z., Madhavan, J., Mork, P., Suciu, D., and Tatarionov, I. 2004. The Piazza peer data management system. *IEEE Trans. Knowl. Data Engin. 16*, 7, 787–798.

Hayashi, H., Hara, T., and Nishio, S. 2003. Cache invalidation for updated data in ad hoc networks. In *Proceedings of the 11th International Conference on Cooperative Information Systems (CoopIS)*. 516–535.

Hellerstein, J., Hong, W., Madden, S., and Stanek, K. 2003. Beyond average: Toward sophisticated sensing with queries. In *Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks (IPSN)*. 553.

Huang, Y. and Garcia-Molina, H. 2004. Publish/subscribe in a mobile environment. *Wirel. Netw. 10*, 6, 643–652.

Hui, P., Chaintreau, A., Scott, J., Gass, R., Crowcroft, J., and Diot, C. 2005. Pocket switched networks and the consequence of human mobility in conference environments. In *Proceedings of the ACM SIGCOMM Workshop on Delay Tolerant Networking*. 244–251.

Hull, B., Bychkovsky, V., Zhang, Y., Chen, K., Goraczko, M., Miu, A., Shih, E., Balakrishnan, H., Madden, S. 2006. CarTel: A distributed mobile sensor computing system. In *Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems*. 125–138.

ISI. 2009. http://www.isi.edu/nsnam/ns/

Jain, S., Fall, K., and Patra, R. 2004. Routing in a delay tolerant network. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. 145–158.

Jelasity, M., Guerraoui, R., Kermarrec, A., Steen, M. 2004. The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. In *Proceedings of the 5th ACM/IFIP/USENIX International Middleware Conference*. 79–98.

Karp, B. and Kung, H. T. 2000. Greedy perimeter stateless routing (GPSR) for wireless networks. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*. 243–254.

Kotidis, Y. 2005. Snapshot queries: Towards data-centric sensor networks. In *Proceedings of the 21st IEEE International Conference on Data Engineering (ICDE)*. 131–142.

LeBrun, J., Chuah, C.-N., Ghosal, D., and Zhang, H. M. 2005. Knowledge-based opportunistic forwarding in vehicular wireless ad hoc networks. In *Proceedings of the IEEE Vehicular Technology Conference*. 2289–2293.

Li, Q. and Rus, D. 2000. Sending messages to mobile users in disconnected ad hoc wireless networks. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*. 44–55.

Lian, J., Chen, L., Naik, K., Ozsu, T., Agnew, G. 2005. Localized routing trees for query processing in sensor networks. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM)*. 259–260.

Medidi, M., Ding, J., and Medidi, S. 2005. Data dissemination using gossiping in wireless sensor networks. In *Proceedings of the SPIE*: *Digital Wireless Communications VII and Space Communication Technologies 5819*, 316–327.

Michel, S., Bender, M., Triantafillou, P., Weikum, G. 2006. IQN routing: Integrating quality and novelty in P2P querying and ranking. In In *Proceedings of the 10th International Conference on Extending Database Technology (EDBT)*. 149–166.

Coman, A., Nascimento, M., Sander, J. 2005. Exploiting redundancy in sensor networks for energy efficient processing of spatiotemporal region queries. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM)*. 187–194.

Motani, M., Srinivasan, V., and Nuggehalli, P. S. 2005. PeopleNet: Engineering a wireless virtual social network. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*. 243–257.

Muller, R. and Alonso, G. 2006. Efficient sharing of sensor networks. In *Proceedings of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems*. 109–118.

Ni, S., Tseng Y., Chen, Y., and Sheu, J. 1999. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*. 151–162.

Perich, F., Joshi, A., Finin, T., and Yesha, Y. 2004. On data management in pervasive computing environments. *IEEE Trans. Knowl. Data Engin. 16*, 5, 621–634.

Pucha, H., Das, S., and Hu, Y. 2004. Ekta: An efficient dht substrate for distributed applications in mobile ad hoc networks. In *Proceedings of the 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*. 163–173.

Sailhan, F. and Issarny, V. 2002. Energy-Aware Web caching for mobile terminals. In *Proceedings*

of the 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW). 820–825.

SORMANI, D., TURCONI, G., COSTA, P., FREY, D., MIGLIAVACCA, M., AND MOTTOLA, L. 2006. Towards lightweight information dissemination in inter-vehicular networks. In *Proceedings of the 3rd ACM International Workshop on Vehicular Ad Hoc Networks (VANET)*. 20–29.

TAYLOR, C., NAKHAEIZADEH, G., AND LANQUILLON, C. 1997. Structural change and classification. In *Workshop Notes on Dynamically Changing Domains: Theory Revision and Context Dependence Issues, 9th European Conference on Machine Learning (ECML'97)*. 67–78.

VAHDAT, A. AND BECKER, D. 2000. Epidemic routing for partially connected ad hoc networks. Tech. rep. CS-200006, Duke University.

VASILAKOS, A., PARASHAR, M., KARNOUSKOS, S., PEDRYCZ, W (EDS). 2009. *Autonomic Communication*. Springer.

WANG, H., FAN, W., YU, P., AND HAN, J. 2003. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the 9th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 226–235.

WIDMER, G. AND KUBAT, M. 1996. Learning in the presence of concept drift and hidden contexts. *Mach. Learn. 23*, 69–101.

WOLFSON, O. AND XU, B. 2007. Mobile peer-to-peer data dissemination with resource constraints. In *Proceedings of the 8th International Conference on Mobile Data Management*. 16–23.

WOLFSON, O., XU, B., YIN H., AND CAO, H. 2006a. Searching local information in mobile database. In *Proceedings of the 22nd IEEE International Conference on Data Engineering*. 136.

WOLFSON, O., XU, B., YIN, H., CAO, H. 2006b. Search-and-discover in mobile P2P network databases. In *Proceedings of the 26th International Conference on Distributed Computing Systems.* 1–9.

XU, B., OUKSEL, A., AND WOLFSON, O. 2004. Opportunistic resource exchange in inter-vehicle ad hoc networks. In *Proceedings of the IEEE International Conference on Mobile Data Management*. 4–12.

XU, J., TANG, X., AND LEE, W. C. 2005. EASE: An energy-efficient in-network storage scheme for object tracking in sensor networks. In *Proceedings of the 2nd IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'05)*. 396–405.

YAO, Y. AND GEHRKE, J. 2003. Query processing in sensor networks. In *Proceedings of the 1st Conference on Innovative Data Systems Research*.

ZHANG, R. AND HU, Y. C. 2005. HYPER: A hybrid approach to efficient content-based publish/subscribe. In *Proceedings of the 25th International Conference on Distributed Computing Systems (ICDCS)*. 427–436.

ZHAO, W., AMMAR, M., AND ZEGURA, E. 2005. Controlling the mobility of multiple data transport ferries in a delay-tolerant network. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (InfoCom)*. 1407–1418.

Received September 2007; revised August 2009; accepted August 2009