J. Parallel Distrib. Comput. 72 (2012) 778–790

Contents lists available at SciVerse ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Direction-based adaptive data propagation for heterogeneous sensor mobility

Azzedine Boukerche [a], Dionysios Efstathiou [b,c,*], Sotiris Nikoletseas [b,c]

[a] PARADISE Research Laboratory, University of Ottawa, Canada
[b] University of Patras, Greece
[c] Computer Technology Institute (CTI), Greece

ARTICLE INFO

Article history:
Received 22 March 2011
Received in revised form
22 January 2012
Accepted 14 February 2012
Available online 5 March 2012

Keywords:
Adaptation
Data propagation
Mobile sensors
Performance evaluation
Wireless sensor networks

ABSTRACT

We consider sensor networks where the sensor nodes are attached on entities that move in a highly dynamic, heterogeneous manner. To capture this mobility diversity we introduce a new network parameter, the direction-aware mobility level, which measures how fast and close each mobile node is expected to get to the data destination (the sink). We then provide local, distributed data dissemination protocols that adaptively exploit the node mobility to improve performance. In particular, "high" mobility is used as a low cost replacement for data dissemination (due to the ferrying of data), while in the case of "low" mobility either (a) data propagation redundancy is increased (when highly mobile neighbors exist) or (b) long-distance data transmissions are used (when the entire neighborhood is of low mobility) to accelerate data dissemination toward the sink. An extensive performance comparison to relevant methods from the state of the art demonstrates significant improvements, i.e. latency is reduced by even four times while keeping energy dissipation and delivery success at very satisfactory levels.

1. Introduction

Until the recent past, wireless sensor networks were usually viewed as static collections of very large numbers of smart sensor nodes. In the prevailing scenario, sensors provide fine grained monitoring of a region of interest by being largely deployed in the region and remaining static. The collected data is disseminated to a static data sink in the network area, using redundant multi-hop data propagation, [11]. Most research in the relevant state of the art that involves mobility, restricts mobility to the data sink(s), i.e., the sink(s) may move but the sensors themselves are assumed static.

However, recent technology advances will make wireless sensors ubiquitously present in the ambient environment, enabling new types of important applications, where motion becomes a fundamental, inherent characteristic. In such applications sensor devices will be attached to moving entities (vehicles, animals, rivers, people moving around large regions etc.), while robotic elements may also move in the region. Sensory data exchange between sensors and control nodes will drive important applications such as traffic and ecosystem monitoring, smart homes and pollution control.

Another application scenario that node mobility is crucial for is VANETs. A Vehicular Ad Hoc Network [26] is a distributed, self-organizing communication network which is formed by moving vehicles, and is characterized by high node mobility and limited degrees of freedom in mobility patterns. These networks exhibit dynamic topology changes and potentially frequent disconnections. In particular, the mobile nodes (vehicles) follow a certain mobility pattern that is a function of the underlying roads, the traffic lights, the speed limit, traffic condition, and drivers' driving behaviors. Examples of VANETs' scenarios are safety applications and applications for intelligent transportation systems (ITS).

In such scenarios mobility is a crucial characteristic of the system. Usually, such applications aim at gathering and processing of large amounts of sensed data; however, immediate delivery of recorded data is difficult and has high energy cost. Instead, measurements can be cached at the sensors (and also ferried to their neighbors) so as to be delivered to the sink whenever the nodes move within its range. Also, the data acquisition process must not interfere with the network e.g. people and other moving entities should not be asked explicitly to move close to the sink to deliver the collected data. Thus, the mobility is uncontrollable and should be left as such. Furthermore, the mobile sensors may follow several heterogeneous mobility patterns which also change a lot with time. The above remarks demonstrate the opportunities created by mobility, which basically stem from its ability to serve as a low cost replacement for connectivity (due to sensor movement to sparse, disconnected areas) and data propagation redundancy (data ferrying by sensors). Still, the mobility management must judiciously cope with complications arising, such as increased data delivery times (high latency), the lack of permanent reference points and the difficulty of maintaining system integrity.

* Corresponding author at: University of Patras, Greece.
E-mail addresses: boukerch@site.uottawa.ca (A. Boukerche), eustathi@ceid.upatras.gr (D. Efstathiou), nikole@cti.gr (S. Nikoletseas).

0743-7315/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2012.02.010

Related work and comparison. Under highly mobile conditions, the protocols and research recommendations for static wireless sensors networks cannot be directly applied (if at all). Data propagation protocols based on multi-hop data forwarding paths or clustering algorithms do not apply, since under mobility the network topology changes frequently. Also, coverage and localization problems become more difficult under mobility. Even very efficient and robust algorithms need to be redesigned, such as in [5] where the authors provide a special leader election algorithm suitable for mobile networks in particular. Furthermore, new problems arise due to the high network dynamics: preserving system integrity [15] and security [7] becomes more complicated. Probabilistic propagation protocols may be suitable in such settings since they are local and require a minimum amount of energy dissipation knowledge only. Also, randomness better balances the load in the network [30].

In [32] the authors introduce Epidemic Routing where random pair-wise exchanges of messages among mobile hosts ensure eventual message delivery and develop techniques to deliver messages in the case where there is never a connected path from source to destination or when a network partition exists at the time a message is originated.

Mobility in sensor networks has been studied mainly assuming that the (one or more) sinks are mobile and collect data traversing the network region and getting within the range of individual sensors. Different data collection and movement strategies have been proposed, as well as routing methods based on the sink stopping at certain positions to collect data [10]. Also, adaptive scheduling algorithms for the motion of mobile sinks to visit nodes according to the data traffic they produce have been suggested in [18]. Optimal trajectories of a mobile sink that minimize energy consumption have been shown in [28]. Repositioning of multiple mobile sinks has been proposed to maximize the network lifetime, as well as randomized distributed coordination mechanisms between mobile sinks [22].

Considering *sensor mobility*, [20] presents a case study of mobile sensor networks designed for wildlife position tracking. The authors assume varying mobility and propagate data to the node most likely to meet the sink; this likelihood is however based on previous history and not the current dynamics. In the Shared Wireless Infostation Model (SWIM) [31], a network of intermittently connected nodes is used to gather oceanographic data from sensors attached to whales through an infrastructure of buoys to which the whales can upload their data. The authors in [34] propose a data dissemination mechanism that also chooses the fittest nodes. Furthermore, they assume limited queues on each node and propose a mechanism to drop messages from the queues based on the likelihood of delivery of each message. We also try to select the best candidates for delivering messages but we assume that node behavior changes, thus instead of using history we choose the best nodes based on their, dynamically calculated, mobility level. Also, we propose and examine more elaborate variations of mobility patterns, while we adaptively select the amount of redundancy (i.e., the number of message ferrying nodes), in terms of the mobility levels in the network (to benefit from high mobility by reducing redundancy).

In [27] the authors propose a geometric data dissemination mechanism for delivering data to a mobile sink. Assuming bounded motion of the sink in a specific but arbitrary area, they characterize the motion using geometric criteria. In our work we propose more elaborate methods for characterizing mobility that capture more subtle variations both in speed and trajectory. Also, in [33] the authors investigate the trade-off between mobility of nodes and coverage of the network area. Our approach in fact exploits such trade-offs in the sense that we handle high mobility as a "replacement" for connectivity and coverage.

The authors in [23] proposed a related but difference mobility level notion as well as adaptive dissemination schemes for solving the problem of data propagation in wireless sensor networks with diverse and dynamic mobility. Our mobility level here does not take into account (in contrast to [23]) dislocation from the origin but is instead direction-aware, in the sense that it captures how close to the sink a mobile sensor tends to become. In addition, although our protocols are also adaptive on the mobility level as in [23], we here extend the data propagation protocol by introducing long distance transmissions to accelerate data forwarding in the case of low mobility. Our approach has been inspired by the research in [13] which, although referring to a different network type, emphasizes the importance and impact of high network dynamics. Also, by Clementi et al. [14] which formally shows that high mobility can strongly accelerate communication while saving energy.

Other studies assume the mobile agent approach where one or more mobile agents may visit a number of sensors and progressively aggregate retrieved sensory data, prior to returning back to the processing element (e.g. sink) to deliver the data. In [24,25] the authors presented an algorithm for finding near-optimal routes for mobile agents that incrementally aggregate data as they visit sensor nodes in a wireless sensor network. This algorithm gradually builds a number of trees whose traversal will eventually determine the itineraries followed by the mobile agents. In [17] the authors introduced the design decisions and implementation aspects of a complete mobile agent platform (MAP) research prototype. Pantziou et al. [29] presents a protocol that proposes the use of urban buses to carry mobile agents that retrieve information from isolated parts of wireless sensor networks. This protocol aims at minimizing the overall network overhead and energy expenditure associated with the multi-hop data retrieval process while also ensuring balanced energy consumption among network nodes and prolonged network lifetime.

Chen et al. [12] presents a model based routing that takes advantage of the predictable node moments along a highway. The authors have verified the hypothesis that the motion of vehicles on a highway can contribute to successful message delivery, provided that messages can be relayed and stored temporarily at moving nodes while waiting for opportunities to be forwarded further. Our method does not assume predictions, while the network type (sensors) is different.

A first discussion on how to incorporate controllable mobile relays into the network infrastructure has been presented in [21]. The authors describe an implementation of a sensor network with an autonomous mobile relay (a robot) that visits the static sensor network in order to collect their data, and deliver the collected data to the sink. The robot traverses networks with different densities following a straight trail, collects the data and ferries them to the sink. In contrast, our method does not control the nodes' motion but adaptively exploits this motion.

In [35] the authors proposed the Message Ferrying approach, which is a mobility-assisted approach that utilizes a set of special mobile nodes called message ferries to provide communication service for nodes in the deployment area. The main idea behind this approach is to introduce non-randomness in the movement of nodes and exploit such non-randomness to help deliver data. In our method, we do not assign ferrying roles to a specific of nodes but all nodes can ferry messages.

Some works based on creation of dynamic clustering of nodes to speedup and support data dissemination in the vehicular sensor networks scenarios can be found in [2,3,1,16].

Our research is actually related to similar work in the context of delay tolerant networks, see [19,34]. However, we note that in our case the node mobility is uncontrollable, more random and unpredictable than in the case of a classic delay tolerant network.

Our contribution. Until recently, most papers on sensor networks mobility have studied the case where only the sink(s) are mobile. However, the research investigating networks where the sensors themselves move receives growing attention. Motivated by important relevant applications where sensory mobility itself is a dominating aspect, we focus on heterogeneous, highly changing mobility profiles. In particular, (a) we propose a new (locally computable) network parameter, the *direction-aware mobility level*, which quite accurately captures how fast and how close a mobile sensor is expected to get to the data destination (sink), taking into account not only its speed but also the direction of its motion. (b) we exploit sensory mobility as a low energy replacement for connectivity and data propagation redundancy: we propose adaptive protocols, that propagate less data in the presence of high mobility with "good" direction toward the sink and favor relay-sensors with "nice" mobility levels. To cope with low mobility neighborhoods, we also introduce (either deterministically or probabilistically) the alternative of long distance transmissions ("jumps") which, although expensive, propagate data very fast toward the sink (c) we also propose a progress-sensitive message flooding inhibition scheme that further reduces communication cost by purging obsolete messages. (d) we implement our protocols and relevant state of the art methods under heterogeneous, dynamic mobility scenarios and perform extensive simulations examining realistic cases where sensors have limited queues for buffering messages. (e) the simulation findings demonstrate the efficiency of our mobility-sensitive adaptation schemes, since they manage to reduce latency a lot (even by 4 times) while also reducing energy dissipation, compared to non-adaptive protocols and even adaptive ones which however are not direction-aware. One major goal in the design of routing protocols for the constrained sensor networks is energy efficiency in order to reduce power consumption to levels suitable for devices powered by small batteries or energy harvesting supplies.

A preliminary version of this significantly extended work has appeared in [6].

2. Model

We assume that the network area $\mathcal{A}$ is a flat square region of size $D \times D$; this assumption can be easily relaxed to include general network areas of arbitrary shapes. The initial positions of sensor nodes within the network area are random and in the general case follow a uniform distribution. *Let n be the number of sensors* spread in the network area and *let d be the density of sensors in that area* (usually measured in numbers of *sensors*/m^2). There is a special node within the network area, which we call the sink $\mathcal{S}$, that represents a control center where data should be collected. $\mathcal{S}$ is immobile and passively awaits nodes to pass by it and transmit their data. In order to be detected by the nodes the sink transmits beacon messages at a rate of $\lambda_{Beacon}\frac{messages}{s}$.

Each sensor device has a *broadcast* (digital radio) *beacon mode* of fixed wireless transmission range R, and is powered by a battery. Also a sensor is equipped with *a general purpose storage memory (e.g., FLASH) of size* $\mathcal{C}$. This storage is used to cache messages that need delivery or forwarding.

Let $\mathcal{E}_i$ *be the available energy supplies of sensor i at a given time instance.* At any given time, each sensor can be in one of three different modes, regarding the energy consumption: (a) transmission of a message, (b) reception of a message and (c) sensing of events. For transmitting and receiving a message, we assume that the radio module dissipates an amount of energy proportional to the message's size. To transmit a `k-bit` message, the radio expends $E_T(k) = \epsilon_{trans} \cdot k$ and to receive a `k-bit` message, the radio expends $E_R(k) = \epsilon_{recv} \cdot k$ where $\epsilon_{trans}, \epsilon_{recv}$ are constants that depend on the radio module hardware, while ϵ_{trans} is also depended on the square of the transmission range R of the sensors. When the radio is idle, the energy consumed is constant and equals E_{elec}. Overall, there are three different types of energy dissipation: (a) E_T, the energy dissipation for transmission, (b) E_R, the energy dissipation for receiving and (c) E_{idle}, the energy dissipation for idle state. We note that in our simulations we *explicitly measure the above energy costs*.

We differentiate from most standard models by assuming *mobility* of the sensors. Sensor nodes can calculate their position in some common coordinate system (e.g., by using navigational equipment or running a virtual coordinates algorithms) and are aware of the dimensions of the network area. Sensors are attached to mobile objects; we model their movement through a high level mobility function which we symbolize by $\mathcal{M}$. Note that nodes generally follow different mobility functions and in fact a single node may follow different mobility functions from one time to another. We consider several types of random motion with respect to speed (low, medium, high) and "locality" (local motion within a limited area or global motion covering a large part of the network). We discuss several aspects of mobility modeling in Section 5. The movement of each sensor node i at time t is characterized by a mobility level $M\ell_i(t)$, which is dependent on the current time.

The mobility function returns a position p_t that the node should move to and the speed of the node to reach p_t. Note that the mobility function can be invoked at anytime even before reaching the designated point. The actual mechanism that moves the mobile entity from position p_{t-1} to position p_t is beyond the scope of this paper. However, in order to simplify our model we assume that all changes in speed and direction can be done instantly. More information about the calculation of $M\ell_i(t)$ and the movement of nodes is given in the following sections.

Finally, we assume that a specific, high-level, application is executed by the sensors that form the network. We model the application by the message generation rate per second λ_i in each sensor i.

3. Computing the direction-aware mobility levels

The mobility parameters studied, in previous approaches such as [23]. like distance traveled, current speed or area covered are not enough to fully capture the ability of a node to arrive close to the sink quite fast. Nodes with the same speed will travel the same distance independently of the trajectory followed, thus overall distance traveled is not characteristic. Furthermore, depending on the type of the mobility pattern, the area covered may vary significantly. Thus, the speed or the area covered by a node, gives us only partial information about the time needed to approach the sink. For example, compare a node with high speed which covers a large area that moves in opposite direction from the sink against a node that moves with lower speed and covers an smaller area than the first node, however with direction toward to the sink. Clearly, based on the above information, the latter's progress toward the sink is higher despite the fact that it moves at slower speed and covers an smaller area. We define a new parameter for characterizing mobility, that captures the differences between the speed and the direction in the trajectory followed.

The computation of the mobility level can be done easily with information locally obtainable by each node. Each sensor node i is responsible for computing its own mobility level. Consider a time interval t_l; every t_l seconds node i records its speed and the angle $d_i(t)$ between its direction of movement and the line connecting the current position and the sink (see Fig. 1).

Consider an integer $K \geq 1$. Let $\upsilon_i(t)$ be the exponential weighted moving average speed of the last K samples, i.e. a sample recorded i time intervals previously, so $t_l = i$ will be multiplied by a weight $w_i = \frac{e^{-i}}{\sum_{n=1}^{K} e^{-n}}$. Let $d_i(t)$ be the exponential weighted moving average direction of the last K samples. Measurements

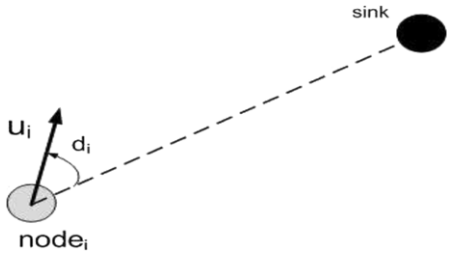


Fig. 1. Example of estimating direction d_i.

of speed and direction are smoothed by applying an exponential weighted moving average to avoid violent fluctuations of speed or/and direction which can be misleading about the real mobility level of a node and exploit recent history information (e.g. a node that moves toward sink for the latest K samples, is more possible to keep its direction). The mobility level of node i at time t is calculated as:

$$Ml_i(t) = \upsilon_i(t) * \left(1 - \frac{d_i(t)}{\pi}\right)$$

which is actually equivalent to the following:

$$Ml_i(t) = \upsilon_i(t) * \cos(d_i).$$

4. Adaptive data propagation

To accelerate data propagation while keeping energy dissipation and delivery success at satisfactory levels, each node decides based on some criteria, between three options: ferrying the message, transmitting the message to several direct neighbors or to one distant neighbor closer to the sink by doing a jump. Ferrying spends no energy at all and is fast when the nodes have high mobility level, while transmission to many neighbors incurs redundancy and long transmissions accelerate data propagation toward the sink at a high energy cost, however; so, the trade-off between these options must be judiciously handled.

Each node can ferry or disseminate the information it records to a number β of its neighbors or to a distant neighbor at the given time. Thus, a node can carry copies of data recorded from itself or from other nodes and deliver them along with their own as soon as they encounter the sink. When a node receives a beacon from the sink it immediately starts to unicast to the sink and the contents of its data cache. After a data message is successfully delivered, it is removed from the cache and no further attempts to disseminate this message are done from the node that delivered this message to the sink. Other nodes that have a copy of this message continue to disseminate the message until they reach the sink or they need to discard it because of cache overflow.

However, in the case of transmitting to direct neighbors, the selection of β and which particular β neighbors get a copy of a datum requires careful design. By setting $\beta = \infty$ the protocol degenerates to flooding, thus expending a lot of energy of the nodes because of the many redundant packets, however delay should be minimal since the data will follow all possible paths to the sink. On the other hand, setting β to a small number will decrease communication cost a little but at the same time the data will take a long time to reach the sink. Moreover, in the case of transmitting to a distant neighbor, the selection of which distant neighbor gets a copy and the length TR_i of the wireless jump done by the node is very important in terms of energy consumption. By setting $TR_i = \infty$ every node in the network has direct access to the sink, thus expending a lot of energy of the nodes because of the long transmissions, however delay should be minimal since there is only one-hop transmissions. On the other hand, setting TR_i to a small number will decrease communication cost a little but at the same time the data will take a long time to reach the sink.

Intuitively, for redundant transmission, slow nodes that make no or small progress toward the sink, nodes that go away from the sink with high speed or nodes that are far away from the sink (i.e., nodes i with small mobility level Ml_i) should choose a larger β to speed up the deliver of data. For jump transmission, nodes that are far away from the sink, have a very low mobility level and they are in a "bad" neighborhood should choose a larger TR_i to overcome the "trap" consisting of nodes with low mobility level. This is because such nodes will take a long time to move close enough to the sink to deliver the data on their own. By spreading out the information to other nodes the probability of meeting the sink increases and consequently the delay of delivery decreases. On the other hand, faster nodes that move toward the sink, can benefit of their high mobility and decide to ferry data or to disseminate data while using a smaller value for β. Such nodes are more capable at getting close to the sink, thus their messages have a high probability of being delivered rapidly. Similarly, when selecting to which nodes to disseminate a message we face again a dilemma. Intuitively, the faster nodes that move toward the sink are more appropriate for message ferrying because they can traverse the network faster and thus may approach the sink more frequently.

4.1. The dissemination scheme

The general dissemination algorithm followed by the nodes in our approach is the following:

1. New messages describing events that need reporting or messages that were forwarded by other nodes are stored in a FIFO queue, called the forward queue. The queue has limited size depending on the cache memory of the node. Each node prioritizes its own data over collected from others. In case the queue becomes full, older messages belonging to another node are discarded to make room for new messages. If messages belonging to another node are not available, it will delete its own oldest data.
2. The node pops the next message from the front of the forward queue and decides to act suitably according to the following scenarios:
   (a) *Data ferrying:* If the node has high mobility level, it decides to ferry/carry the data instead of transmitting to other nodes (to save energy while propagating data fast).
   (b) *Data transmission:* If the node is not ideal to ferry/carry the data, it transmits data using one of the following choices:
      i. *Redundancy:* If at least one direct neighbor of the node has a high mobility level ("good" neighborhood), the node disseminates the information to a number β of its neighbors at the given time (toward a higher success ratio).
      ii. *Jump:* If all of the node's direct neighbors have low mobility level ("bad" neighborhood), the node transmits data not to one-hop neighbors, but to TR_i-hop closer to sink neighbor in order to avoid the "trap" ("bad" neighborhood) (Trade-off between latency and energy dissipation).
3. Forwarded messages and messages that for some reason have not been sent successfully (e.g. due to collisions, absence of neighbors etc.) are then stored in a delivery queue that has the same characteristics (size, FIFO etc.) as the forward queue. The detection of an unsuccessful transmission and the retransmission of the message is assured by the underlying lightweight MAC protocol.
4. For the redundancy transmission scenario, if a beacon from the sink is received, then the node switches to a *connected* operational state and begins the delivery of the messages. For the jump transmission scenario, if the sink is in a node's new

transmission range, then the node switches to a *connected* operational state and begins the delivery of the messages. First it selects messages from the delivery queue to transmit directly to the sink. If the delivery queue is empty it selects messages from the forward queue. Note that in this case messages are not sent to β neighbors. After successful delivery messages are erased from the memory of the nodes thus freeing resources.

5. If the delivery of a message to the sink fails, the node reverts to the *disconnected* state and operates according to steps 1,2,3.

Note that messages received from other nodes are discarded, in case they already exist in the forward queue.

4.2. Decision criterion

Firstly, node i decides whether to ferry data or to forward data based on a hard threshold choice criterion. The two different scenarios are explained below.

We define γ for node i and for its neighborhood respectively, in terms of the current and maximum mobility levels as follows:

$$\gamma_i = \frac{Ml_i(t)}{Ml_{\max}(t)}$$

$$\gamma_{neigh\ i} = \frac{Ml_{\max\ neighbor}(t)}{Ml_{\max}(t)}$$

where $Ml_{\max}(t)$ is the maximum possible mobility level of a node, which is moving with a maximum velocity $\upsilon_{\max}(t)$ and with direction $d_i(t) = 0$ between its direction of movement and the line connecting the current position and the sink and $Ml_{\max\ neighbor}(t)$ is the maximum mobility node of a neighboring node.

1. *Ferrying:* Node i decides to ferry data based on probability γ_i.

   $\gamma_i \geq 1 - \alpha$

   where $\alpha \in [0, 0.7]$ a constant value. Clearly, ferrying is suitable when γ_i is relatively high. After many simulations tries for various a in this domain ([0, 0.7]), it is demonstrated that the particular value of $\alpha = \frac{1}{2}$ ($\gamma = \frac{1}{2}$) optimizes performance. In fact, larger γ_i values lead to many redundant transmissions (see below), while smaller γ_i values result in ferrying by low mobility nodes, thus increasing latency.
2. *Transmission:* If $\gamma_i < 1 - \alpha$ then no ferrying is done. Interestingly, the decision on whether to do ferrying or transmission exhibits a certain threshold behavior around a constant probability (which is optimized for $\gamma_i = 0.5$). Below we propose two methods for deciding whether the node will redundantly propagate data or forward to TR_i-hop remote neighbors.
   (a) *Hard threshold:* Node i selects to forward data to β_i neighbors or to TR_i-hop neighbors based on hard thresholds γ_i and $\gamma_{neigh\ i}$. Assuming that each node has a maximum mobility capacity, that is bounded by its maximum speed $\upsilon_{\max}$ when its speed's direction is $d_i = 0$, so it is moving exactly on the line between i and the sink, then the maximum mobility level a node i can reach is:

   $$ML_{\max} = \upsilon_{\max} \cdot 1 = \upsilon_{\max}.$$

   The same assumption can be applied for each neighbor's maximum mobility capacity, then the maximum mobility level a neighbor of node i can reach is:

   $$ML_{\max\ i\ neighbor} = \max_{j \in neigh\ i} ML_j.$$

   According to γ_i and $\gamma_{neigh\ i}$ values, there are two scenarios:
   i. *Redundancy:* The condition that has to be satisfied for transmitting to β_i neighbors is the following:

      $$\gamma_{neigh\ i} \geq \frac{1}{k}$$

      where $k \geq 5$ a large constant. Intuitively, node i decides to transmit to β_i neighbors, when node i and at least one of its neighbors have a relative high mobility level, in order to provide limited redundancy only when it can be useful. To be more specific, "bad" neighbors make no or very small progress toward the sink, so it is a waste of resources to transmit data if the node i has higher mobility level than them.
      The particular $\gamma_{neigh\ i}$ value of 0.1 ($k = 10$) has been selected after extensive simulations, which suggested that this value leads to best possible results. In fact, larger values result to more long distance transmissions (thus increasing energy dissipation too much), while smaller values lead to too few jumps, thus latency is high.
   ii. *Jump:* The condition that has to be satisfied for transmitting to TR_i-hop neighbors is the following:

      $$\gamma_{neigh\ i} < \frac{1}{k}$$

      where $k \geq 5$ a large constant. Intuitively, node i decides to transmit to a TR_i-hop neighbor, when node i and all of its neighbors have a relatively low mobility level, in order to avoid a "bad" neighborhood trap, and make the maximum possible progress to the sink (trading-off with energy cost).

   (b) *Probabilistic:* Node i examines the neighborhood information and performs a probabilistic choice using p_i in order to forward data to β_i neighbors or to TR_i-hop neighbors. We define p_i for node i in terms of the current and maximum mobility levels of node i and its neighborhood as follows:

   $$p_i = \left(1 - \frac{Ml_i(t)}{Ml_{\max}}\right) \cdot \left(1 - \frac{Ml_{neigh\ i}(t)}{Ml_{\max}(t)}\right).$$

   Let p_i the probability of transmitting to TR_i-hop neighbors, and $1 - p_i$ the probability of transmitting to β_i direct neighbors; the following choice is perform in node i:

   $$Transmission = \begin{cases} \textbf{Redundancy}, & 1 - p_i \\ \textbf{Jump}, & p_i. \end{cases}$$

The hard threshold choice is deterministic and each time selects the best possible option; but this can be myopic in contrast to the randomized decision which has "balancing" properties and can perform better in the long run avoiding anomalies.

4.3. Calculation of data redundancy β

Below we propose two methods for selecting the number of neighbors β to disseminate a message. The first one is completely local and low cost while the second collects additional information to improve the decision. The cornerstone of our methods is the use of the mobility level and the distance from the sink of the nodes involved in the process to estimate the requirement for redundancy of message transmissions.

Completely local protocol: A node that moves at maximum mobility level is considered capable of delivering messages, practically without disseminating them to the rest of the nodes. Furthermore, it is more crucial to have larger redundance β to regions far away from the sink, rather in regions close to the sink. We define β for node i in terms of the current and maximum mobility levels and current and maximum distance from the sink:

$$\beta_i = \left\lceil \left(1 - \frac{Ml_i(t)}{Ml_{\max}}\right) \cdot \left(\frac{D_i}{D}\right) \cdot \delta_1 \right\rceil$$

where D is the dimension of the $D \times D$ network area, D_i is the current distance from sensor i to the sink, and δ_1 represents the maximum possible redundancy as given by $\delta_1 = \left\lfloor \frac{\text{dist}_{sink}(i)}{R} \right\rfloor$. R is the transmission range of a mobile node and $\text{dist}_{sink}(i)$ is the Euclidean distance of node i from the sink. E.g. assuming that the transmission range of both nodes and sink is set to $R = 70$ m, for a flat square region of size 1000×1000 m^2 and that the node i is placed in the borders of the network ($\text{dist}_{sink}(i) = \frac{D}{2}$) the maximum possible redundancy δ_1 can be calculated by setting $\text{dist}_{sink}(i) = \frac{D}{2}$, then $\delta_1 = \frac{\frac{D}{2}}{R} = 7$. The first term $1 - \frac{Ml_i(t)}{Ml_{\max}}$ estimates how close the node's mobility level is to the maximum mobility level. The fraction $\frac{D_i}{D}$ estimates how close node i is to the data sink. Finally, the product of the two previously mentioned terms takes values between 0 and 1. This product is multiplied by $\delta_1 = 7$, which represents the maximum possible redundancy, respectively. The rationale of this function is to calculate large values of β for "slow", moving in "bad" direction and distant from the sink nodes. The opposite happens for "fast", moving in "good" direction and close to the sink nodes: as $Ml_i(t)$ approaches $Ml_{\max}$ and/or D_i is small relatively to D the value of β_i approaches zero, meaning that the node will not redundantly disseminate the message to other nodes but instead transmit directly to the sink as soon as it is within range. β_i is dependent on Ml_i and D_i, its value also changes over time to reflect the changes in these two metrics, thus the behavior of the node is adapting to its mobility, direction of movement and distance from the sink.

Neighbor discovery protocol: Node i transmits a beacon message announcing its mobility level and its id. Nodes that receive the beacon of i respond with a message containing their id and mobility level.

Node i then calculates the average mobility level in the neighborhood; assuming $neigh_i(t)$ is the set of all neighbors of node i (at circular disk of radius R, where R is the transmission range) at time t we have:

$$Ml_i^{avg}(t) = \frac{\sum_{j \in neigh_i(t)} Ml_j(t)}{|neigh_i(t)|}.$$

In essence, $Ml_i^{avg}(t)$ captures the available mobility at the neighborhood of i at time t. Using $Ml_i^{avg}(t)$ node i can calculate its β as follows:

$$\beta_i = \left\lceil \left(1 - \frac{Ml_i^{avg}(t)}{Ml_{\max}}\right) \cdot \left(\frac{D_i}{D}\right) \cdot \delta_1 \right\rceil.$$

Note that $Ml_i^{avg}(t)$ encapsulates only the mobility level of the neighbors, but not the mobility level of the node i itself. There is no need to include the average distance D_i^{avg} from the sink, because nodes in a neighbor have approximately the same distance from the sink. As before, the product of the two first terms takes values between 0 and 1, and approaches 0 when the average mobility approaches $Ml_{\max}$.

4.4. Calculation of length of jump TR_i

Below we propose two methods for selecting the length of the jump to transmit a message. As in calculation of redundancy β, the cornerstone of the first method is the use of the mobility level and the distance from the sink of the nodes involved in the process and the core idea of the second method is an expanding ring search. The first method is local and low cost, while the second is more detailed but can become energy-expensive. Note that a node after transmitting a message to a long neighbor, it discards the message from its cache.

Neighbor discovery protocol: Node i transmits a beacon message announcing its mobility level and its id. Nodes that receive the beacon of i respond with a message containing their id and mobility level. Using $Ml_i(t)$ and $Ml_i^{avg}(t)$ node i calculates its TR_i as follows:

$$TR_i = \left\lceil \left(1 - \frac{Ml_i(t) + Ml_i^{avg}(t)}{Ml_{\max}}\right) \cdot \left(\frac{D_i}{D}\right) \cdot \frac{\delta_1}{2} \right\rceil.$$

As in β_i calculation the product of the first two terms is multiplied by $\frac{\delta_1}{2} = 3$, where 3 represents the maximum possible jump range in the particular network setting. The rationale of this function is to calculate large values of TR_i for "slow", moving in "bad" direction, distant from the sink nodes which are in relatively "bad" neighborhood. In this case, node i makes a jump of TR_i length toward the sink by transmitting directly to TR_i-hop neighbors. The opposite happens for "fast", moving in "good" direction, close to the sink nodes which are in relatively "good" neighborhood: as the sum of $Ml_i(t)$ and $Ml_i^{avg}(t)$ approaches $Ml_{\max}$ and/or D_i is small relatively to D the value of TR_i approaches zero, meaning that the node will not transmit in long range the message to other nodes but instead transmit directly to the sink as soon as it is within range. TR_i is dependent on Ml_i and D_i, its value also changes over time to reflect the changes in these two metrics, thus the behavior of the node is adapting to its mobility, direction of movement and distance from the sink.

Expanding Ring Search (ERS): The problem we want to solve is to find the TR_i-hop neighbor that has relatively high mobility level and will be a "good" candidate in order to transmit data, so as to avoid the bad neighborhood. ERS successively searches larger areas until a node with mobility level higher than a hard threshold is located. The complexity of this algorithm can be easily bounded by putting an upper threshold on the number of search iterations; but it can be high anyway, and this is the weakness of this method.

Node i, which is in a "bad" neighborhood, begins the search with a time to live (TTL) taken as 2 for the query. Each time the query is forwarded by a node, the TTL value i is decremented by 1. When TTL reaches zero, the query packet is not forwarded any further. Thus, by setting the appropriate TTL value in the query packet, the source node can control the search radius. After sending a query with a given TTL, the source node waits for a time-out period to receive a reply. If there is no reply within the time-out period, the source increments the TTL by 1 and re-initiates the query. The repetitive search process continues until the TTL reaches a threshold value L. If no reply is received after L successive searches, the query is broadcasted through the L-hop neighborhood.

In order to decrease communication complexity, the above algorithm can search for "good" neighbors only in a sector of the circular disk of radius $k \cdot R$, where k is the current round of the algorithm. To be more specific, node i will pay the transmission cost of broadcasting the query, but only the nodes in the sector will pay the cost of receiving it. Furthermore, only the nodes in the sector will have to answer back to node i and node i will receive answer from a small partition of nodes on the circular disk of radius $k \cdot R$.

4.5. Neighbor selection

Our protocol has to do neighbor selection in two cases, when selecting direct neighbors in order to do redundancy and when jumping to a long neighbor so as to avoid bad neighborhood.

4.5.1. Direct neighbor selection

After calculating β_i, node i needs to select the particular β_i neighbors to deliver the message to. As mentioned earlier this selection can influence the overall performance of the protocol; intuitively "fast" moving in "good" direction nodes at high mobility level should be preferred. However, always selecting the same

"fast" nodes will result in uneven workload and strain their resources. Below we present three different strategies for selecting the nodes to disseminate a message to.

Completely random selection. Node i selects β_i of its neighbors randomly. In order to do so the node uses the neighborhood information gathered by the neighbor discovery protocol. This simple method probabilistically guarantees that the load distribution will be equally shared by the nodes. It is also particularly relevant in cases of limited network knowledge.

Fittest candidate selection. Node i selects β_i of its neighbors such that $Ml_i(t) < Ml_j(t)$ where j a neighboring node to i. In this way the neighbors with the highest mobility level are selected, hoping to reduce latency. In the case where no neighbors with higher mobility level can be found, node i waits for a short period of time and repeats the neighbor discovery process in the hope that it either reaches a new neighborhood or new neighbors approach it.

Probabilistic candidate selection. To avoid long delays until finding suitable nodes with higher mobility level and also to reduce the strain imposed on these nodes, we compromise our selection criterion. Again node i selects β_i of its neighbors such that $Ml_i(t) < Ml_j(t)$, however if no such neighbors are found the rest of the nodes are examined probabilistically, in a way that favors nodes with high mobility. In detail, if node i has higher mobility level from all of its neighbors, it probabilistically favors the neighbors that have the highest mobility level using the probability $\frac{Ml_j(t)}{Ml_i(t)}$. Let p_j the probability of sending a message to node j; p_j is calculated as follows:

$$p_j = \begin{cases} \dfrac{Ml_j(t)}{Ml_i(t)} & Ml_j(t) \leq Ml_i(t) \\ 1 & Ml_j(t) > Ml_i(t). \end{cases}$$

Thus, the node examines the neighborhood information and for each neighboring node it performs a probabilistic choice using p_j until the message is sent to β_i neighbors.

4.5.2. Long neighbor selection

After calculating TR_i, node i needs to select a particular long neighbor to deliver the message to. Node i, queries nodes that are at distance between $TR_i - 1$ and TR_i from node i, and have smaller euclidean distance from the sink from node i. As mentioned earlier this selection can influence the overall performance of the protocol, intuitively "fast" moving in "good" direction nodes at high mobility level should be preferred. However, always selecting the same "fast" nodes will result in uneven workload and strain their resources. Below we present two different strategies for selecting the nodes to disseminate a message to.

Completely random selection. See 4.5.1 for details.

Fittest candidate selection. See 4.5.1 for details.

4.6. Inhibition of obsolete messages

Note that although selecting only β neighbors at a time will have the effect of reducing the rate a message spreads throughout the network, the propagation of a message is arbitrary and eventually it may be transmitted to every single node. Even when a node k delivers the message to the sink, the rest of the nodes that have a copy of the message will propagate the message to about β neighbors each. Nodes that already store a copy of the message will discard it, however the message may still be flooded through the network at a slow pace. Here we present a mechanism to reduce the spread of a message.

We introduce a hop counter h_c contained in each message transmitted; a node i before transmitting a message increases its h_c. Each node j that receives a message performs a deterministic check to decide whether to further propagate the message to its neighbors or to simply store the message in its delivery queue until a sink is located. The decision to whether to propagate or just store the message is done as below:

$$Decision = \begin{cases} Propagate\ message & h_c < h_{opt} \\ Store\ message & h_c \geq h_{opt} \end{cases}$$

where h_{opt} is the optimal number of hops between node j and the sink as given by $h_{opt} = \left\lceil \frac{\text{dist}_{sink}(j)}{R} \right\rceil$. $\text{dist}_{sink}(j)$ is the Euclidean distance of node j from the sink and R the wireless transmission range of nodes. Since, the location of the sink (hence also $\text{dist}_{sink}(j)$) may not always be known, h_{opt} can be calculated by using another distance, for example by setting $\text{dist}_{sink}(j) = \frac{D}{2}$. In this way the inhibition decision depends on the distance the message has traveled (as given by h_c) with respect to the overall required distance (as given by h_{opt}). Thus a message will not be propagated further than h_{opt} hops. On the other hand, messages that performed few hops are more likely to enter the forward queue. We note that this inhibition mechanism is executed every time a node tries to store a message to its forward queue in both direct and long transmissions.

5. Modeling diverse mobility

In most real world scenarios most nodes will move in many different and diverse ways. For example, a sensor attached on a vehicle will move fast on a trajectory that consists of a consecutive set of line segments. On the other hand, a pedestrian will tend to move slower over local trajectories with more curves. During these types of movement small variations of speed are usual. Also, a node will most likely change the type of movement it follows after some time varying not only the average speed but also the type of trajectory it follows. Consider a person riding a bicycle to work, then spending several hours working (low mobility), then riding back home. For example, consider a person working in a university campus; for long periods of time she moves slowly in a confined space (e.g., 10 by 10 m) as she goes about her work in the office. At some point, the person may start walking faster toward a specific direction as she goes to the next building where she continues her work reverting to the previous type of movement. These examples demonstrate the diversity and variability that may arise in networks of mobile sensors. Modeling real life movement patterns is a subject of active research. Simplistic mobility patterns, such as random walk or random waypoint alone, cannot accurately capture the heterogeneous mobility characteristics we described. Here we try to mimic several main types of movements inspired from the above observations. Using well defined mobility models, we define a few characteristic mobility roles that are used to construct more complex mobility behaviors.

Working movement. We parameterize a version of random walk [8] to achieve slow movement with small variations away from the center of the movement. We define the function $\mathcal{M}_{work}$ with parameters [0.5, 1.5] m/s for choosing speed and by setting the movement distance toward a direction to be small, [1, 5] m. Nodes move slowly, the movement is mostly centered in the area around their initial position $\mathcal{M}_{work}$. Such movement can be approximated by a random walk [8] mobility function. We define the function to choose a direction and a distance to move toward that direction. Good parameters for obtaining a local movement are [0.5, 1.5] m/s for choosing speed and by setting the movement distance toward a direction to be small, e.g., [1, 5] m. $M_{avg} \simeq 7.5$

Walking movement. Nodes move more quickly than the working mobility and travel in smoother trajectories $\mathcal{M}_{walk}$. Such behavior can be obtained by using a variation of the Boundless Area [8] mobility model to define $\mathcal{M}_{walk}$, which is more rapid and less local than $\mathcal{M}_{work}$. When a node reaches the boundaries of the network

area we force it to reflect, i.e., take a left turn of 45°. We bound the speed to vary between [1, 2] m/s, we set the time step $\Delta t = 2$ s; at each time step we allow the speed to vary by $\Delta v = 0.25$ m/s and the direction to vary by $\Delta\alpha = 30°$.

Bicycle ride. This type of movement is similar to the walking movement except that the speed is usually greater and there are less direction changes $\mathcal{M}_{bic}$. Again we use our variation of the Boundless Area [8] mobility model; we bound the speed between [3, 6] m/s (10.8–21.6 km/h), we set $\Delta t = 3$ s, $\Delta v = 0.5$ m/s and $\Delta\alpha = 30°$.

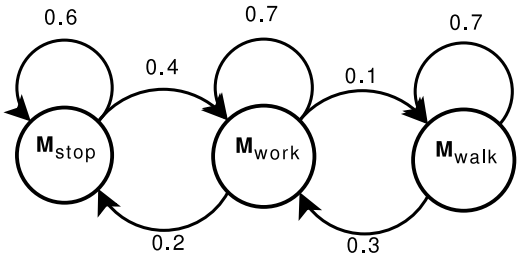
Vehicular movement. Vehicular movement $\mathcal{M}_{veh}$ is the faster of all, we use the Probabilistic Random Walk [8]. In this mobility model, nodes move only toward predefined directions north, north east, east etc. We vary the speed between [5.55, 10] m/s (20–36 km/h).

Mobility transitions. Assigning a mobility role is enough to diversify the mobility levels of the nodes. However, in realistic scenarios nodes will change mobility roles. To model such dynamic mobility, we use a state transition diagram to change between mobility models. Each state of the diagram corresponds to a mobility role as defined above. From each state a set of outgoing edges to one or more of the other states exist; each edge is associated with a probability of transition. Also, there is an outgoing edge that returns to the same state. The sum of all outgoing edges from a state is equal to 1. While on a state the node follows the mobility defined by the corresponding mobility model. As soon as a new position needs to be selected a probabilistic experiment is performed to choose a new state according to the state transition diagram, then the corresponding mobility function is invoked to select the position and speed of the node. We also define a special state called the stop state in which the node remains still for a small period of time. The following diagrams define characteristic mobility transitions.

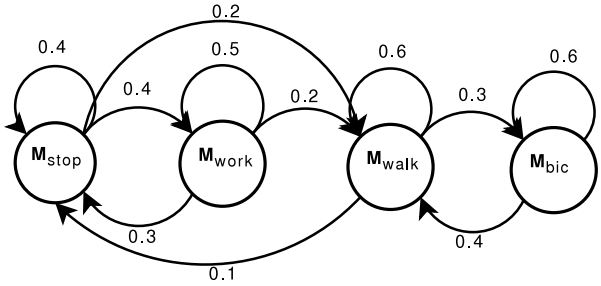
6. Experimental evaluation

We implement our protocols on the **ns-2** simulation platform version 2.33, using the TRAILS toolkit [9] which simplifies the implementation and evaluation of complex and dynamic mobility scenarios. Our configuration uses 802.11 for the MAC layer and our experimental results are affected by the collisions occurred in the wireless network, message re-transmissions etc. We set the network area to be 1000 × 1000 m^2 and we always position $\mathcal{S}$ at (500, 500) the center of the network. We deploy in a random uniform manner, a number of 50, 100, 150, 200, 250, 300 nodes in the network area and repeat each experiment 70 times, with 95% confidence intervals calculated. The nodes and sink have significant energy resources (100 J) to prevent failures due to energy depletion. Also, we do not consider the possibility of other types of node failures. The sink $\mathcal{S}$ transmits beacon messages at a steady rate $\lambda_{Beacon} = 1$, that is a beacon message per second. We assume that all sensor nodes record an instance of the environmental conditions producing 20 messages during the simulation. Messages are produced at random intervals and on the average new messages are produced at rate $\lambda_i = 0.025$ messages/s. Thus, the data generation phase lasts for about 800 s, we simulate the network for 3600 s in order to collect delayed data. The data is generated in packets of 36 bytes while the size of a beacon message is 24 bytes. Each node uses fixed sized caches for the forward and delivery queues, each cache can accommodate 64 messages, thus there is a possibility of message drops due to caches exceeding their maximum size., i.e., we avoid the ideal case of infinite buffers. The transmission range of both nodes and sink is set to $R = 70$ m which corresponds to the transmission power value of 0.00263 (−26 dBm) on ns-2. The characteristics of the radio module, i.e., the values of ϵ_{trans}, ϵ_{recv} and E_{idle}, were set to match as close as possible the specifications of commercially available sensors such as TelosB motes that uses the Chipcon CC2420 [4] transceiver.

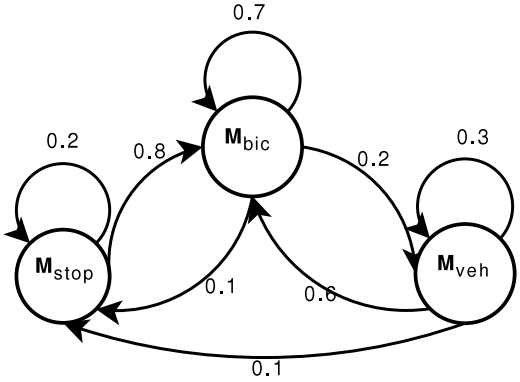
Node movement. We assign different mobility roles to the nodes of the network. In a first experiment, we examined mixed mode scenarios where 25% of the nodes follow $\mathcal{M}_{work}$, 25% $\mathcal{M}_{walk}$, 25% $\mathcal{M}_{bic}$ and 25% $\mathcal{M}_{veh}$. The assigned mobility functions remain the same for a particular node during the simulation. In the second experimental setup, the mobility of the nodes changes during the simulation using the mobility transitions defined earlier see Fig. 2, we assign **C1** (slow mobility) to 25% of the nodes, **C2** (medium mobility) to another 25%, **C3** (medium mobility with fast bursts) to another 25% and **C4** (fast mobility) to the remaining 25% of the nodes.



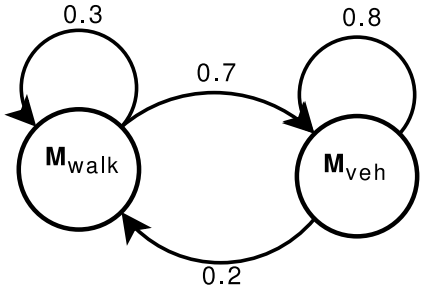
C1: Transitions between slow mobility roles



C2: Transitions for medium mobility level



C3: Transitions for medium mobility with fast bursts



C4: Transitions for fast mobility

Protocol comparison. We implemented and evaluated in these settings three known protocols to use as a point of reference in our evaluation. We obtain the *simple flooding protocol* simply by setting $\beta = \infty$ (i.e., the node will send the message once to all its neighbors) without any adaptation or the message to inhibition mechanism. Also, this protocol does not execute the neighbor discovery protocol, instead it broadcasts a message simultaneously to all neighboring nodes. The second test case protocol is the *gossiping protocol* in which a message is sent randomly to one neighbor and after the transmission the message is discarded from the cache. So, only the initial message is traveling through the network and no copies are created. The third test case protocol is the *Adaptive Mobility Protocol (AMP)*, [23] which is an adaptive redundancy protocol for data propagation. We compare these three protocols to our *fixed TR_i protocol* for both hard threshold and probabilistic decision criterion, which is a non-adaptive version of our main protocol; we set $TR_i \in \{2, 4, 6, 8\}$. Also, we comparatively study characteristic variations of our method corresponding to selected different design alternatives.

Metrics. Conducting these experiments, we measure several metrics that depict the behavior of our protocols. We call *success rate* the percentage of data messages that were received by all sinks over the total number of generated messages. We measure the *energy* consumed at the sensor network due to communication, as the average number of Joules consumed at each node. Note that we consider the motion of the nodes to be initiated by the objects/persons/vehicles they are attached onto, so the nodes themselves do not consume energy for movement. We also measure the *delivery delay*, which is defined as the average time interval between the creation of a message and the time when it is delivered to the sink.

6.1. Performance findings

First experiment. In the first set of experiments we present here, the sensor nodes are divided in four groups, where the nodes of each group follow a specific mobility role. Firstly, we compare the best variation of our protocol with flooding, gossiping and AMP. In Fig. 2 we can see that the highest success rate is achieved by our adaptive *Direction Sensitive Mobility Protocol (DSMP)*, in which the hard-threshold decision is selected and the fittest neighbor either direct or long, is being selected (95%). The AMP protocol is almost in a tie with the fixed DSMP protocol. Flooding achieves a very low success rate due to the many packets dropped by the limited sized queues and the large number of collisions during transmissions. Gossiping achieves the lowest success rate due to the lack of replicas of the message being transmitted.

In Fig. 3 we observe that the flooding protocol is the most energy consuming among the four compared protocols. The adaptive DSMP protocol consumes about 25% more energy than the gossiping and AMP protocols. This is explained by the fact that the DSMP protocol is the only one that uses the expensive, but fast, long transmissions.

The delivery delay is shown in Fig. 4. Our adaptive DSMP protocol is the fastest and improves about 360% the delay compared to the adaptive AMP protocol. In DSMP protocol every message is transmitted/ferried exclusively toward sink thus incurring the lowest possible delay. A very interesting result that can be explained from the above is, that DSMP achieves better delivery delay than the fast one-hop flooding protocol. We can see that flooding has very low delay, but this is expected since messages farther away from the sink, that exhibit long delays, are most likely to be dropped and thus are not considered in the calculation of the delay. As expected, gossiping has the highest delay because of the absence of adaptation and replication in data dissemination.

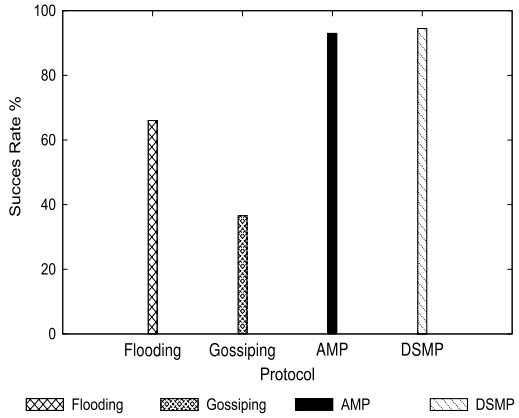


Fig. 2. Success rate of the protocols when nodes are assigned a static mobility role.

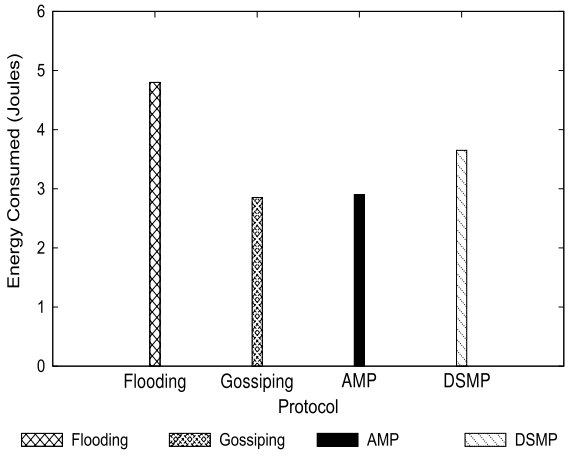


Fig. 3. Energy dissipation of the protocols when nodes are assigned a static mobility role.

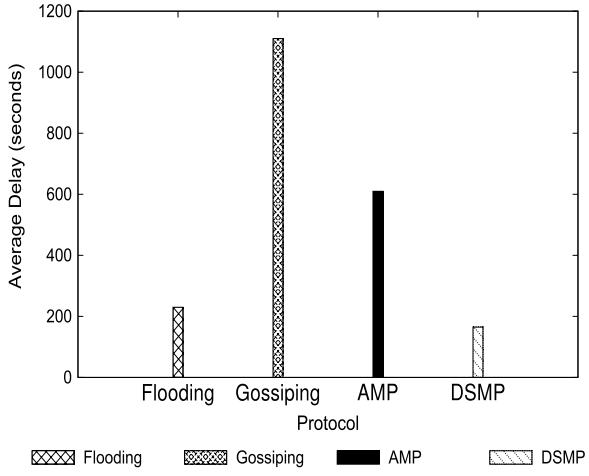


Fig. 4. Message delivery delay of the protocols when nodes are assigned a static mobility role.

Variations of our protocol. Due to lack of space it was impossible to present experimental results for every possible variation of our protocol. As a result, we decide to calculate data redundancy β and length of jump TR_i using the neighbor discovery protocol. In addition, for neighbor selection we decide to use the fittest candidate selection for both direct and long neighbors.

In Fig. 5 we can see that the adaptive protocol with the hard-threshold decision criterion achieves the highest success rate. The hard-adaptive protocol is almost in a tie with

the fixed $TR_i = 2$ protocol. The probabilistic adaptive protocol achieves a satisfactory success rate (89%). For both the hard-threshold and probabilistic case, as TR_i increases the delivery ratio decreases rapidly, because of the increasing number of unsuccessful transmissions due to the collisions in the MAC layer.

The energy dissipation of the protocols is shown in Fig. 6. We observe for both cases of decision criterion (probabilistic and hard-threshold) that, as the fixed TR_i increases, the number of collisions increases and the number of successful data transmissions decreases. As a result, the nodes cannot discover many neighbors because the control messages for neighbor discovery fail to reach their destination and thus our protocol is no longer operational as it relied on local information. Comparing the two different decision criteria, the hard-threshold criterion is the most energy conservative in any case.

The delivery delay is shown in Fig. 7. The lowest delivery delay is achieved by the hard-adaptive protocol which is about 200% lower than the prob-adaptive protocol. For the fixed TR_i protocols, the best performance is achieved for $TR_i = 4$ in both hard-threshold and probabilistic decision criterion. As TR_i increases, the delivery delay deteriorates since the number of successful data transmissions decreases because of the vast number of collisions.

Second experiment. The results for this setup (varying mobility profiles) that are depicted in Figs. 11–16 are similar in the sense that our protocol and its variations perform very good in this case as well.

To be more specific, Fig. 14 shows that the AMP and DSMP adaptive protocols have the highest success rate (92%). All the four studied protocols have a little bit lower success rate than the case where the mobility roles are assigned statically.

Fig. 15 depicts the findings about energy consumption. We note that the AMP adaptive protocol has again the smallest energy consumption among all of the protocols. Interestingly, the DSMP adaptive protocol has better performance in this case than the static mobility role scenario which proves that our protocol is efficient and adapts wells in more complex and changing mobility scenarios.

Fig. 16 depicts the delivery delay where all the studied protocols have the same performance as in the case of the statically assigned mobility roles.

Variations of our protocol—Second experiment. In Fig. 14 we note that the adaptive protocol with the hard-threshold decision criterion achieves again the highest success rate (92%), while the other variations (<85%) for the varying mobility profiles scenario have a bit lower success rate than the case of statically assigned mobility profiles. The fixed $TR_i = 2$ variation for both the hard-threshold and probabilistic case and the probabilistic adaptive protocol achieve a satisfactory success rate (85%). As TR_i increases, the number of collisions increases and the success ratio decreases vastly.

The energy dissipation of the protocols is depicted in Fig. 15. We note that the adaptive protocol with the hard-threshold decision criterion is the most energy efficient protocol by combining the information for the success rate metric and has better performance for the case of the dynamically changing mobility roles.

Fig. 16 the data delivery latency. The lowest delivery delay is achieved by the hard-adaptive protocol which has better performance for the dynamic mobility profiles and it has over 250% better performance than the prob-adaptive protocol. Again, for the fixed TR_i protocols, the best performance is achieved for $TR_i = 4$ in both hard-threshold and probabilistic decision criterion. As TR_i increases, the impact of the collisions and the re-transmissions to the average delivery delay is dominant.

The impact of density. We note that the figures above concern the case of 300 nodes. However, we have conducted experiments for various densities corresponding to 50, 100, 150, 200, 250, 300 nodes. In Figs. 8–10 we present the comparison of the two adaptive protocols (AMP, DSMP). We interestingly remark that our DSMP

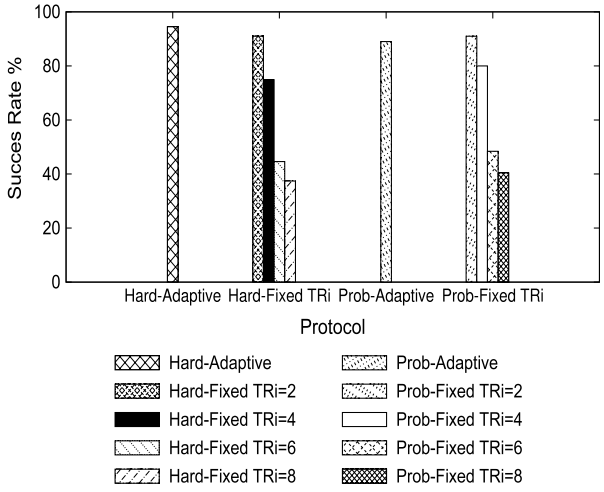


Fig. 5. Success rate of our protocol's variations when nodes are assigned a static mobility role.

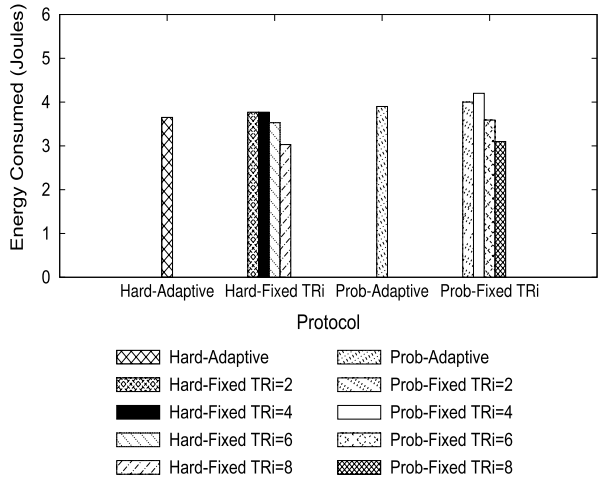


Fig. 6. Energy dissipation of our protocol's variations when nodes are assigned a static mobility role.

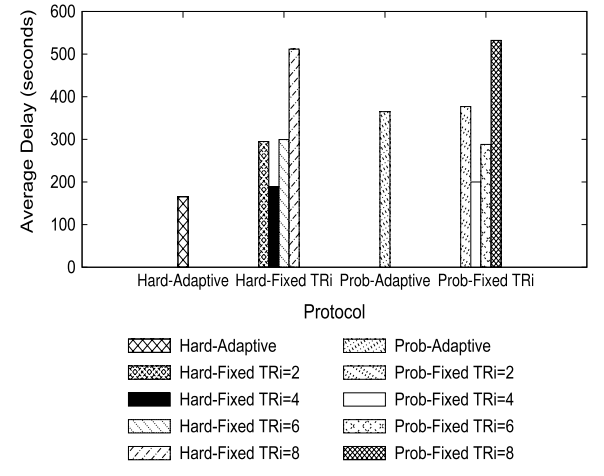


Fig. 7. Message delivery delay of our protocol's variations when nodes are assigned a static mobility role.

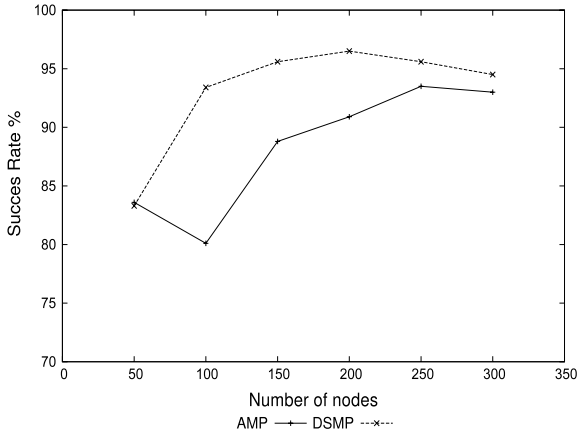


Fig. 8. Success rate of the protocols for various densities.

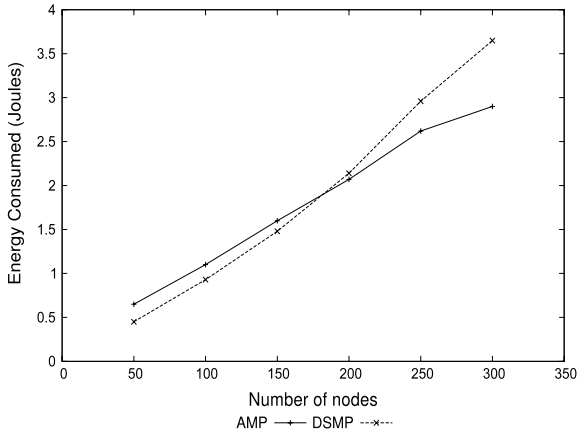


Fig. 9. Energy dissipation of the protocols for various densities.

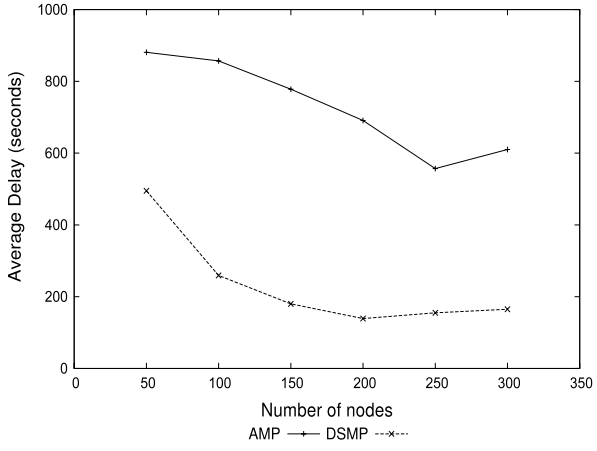


Fig. 10. Message delivery delay of the protocols for various densities.

protocol behaves better in sparse networks as well; this is due to its ability to jump over sparse or even disconnected areas.

7. Conclusions and future work

We investigate networks of mobile sensors with diverse, highly changing mobility profiles. To abstract the sensor mobility dynamics we propose a novel network parameter, the direction-aware mobility level, which captures how fast and close toward the sink each mobile node is moving. Exploiting the potential of mobility to serve as a low cost replacement for connectivity and data propagation redundancy, we provide distributed, adaptive data dissemination protocols that significantly improve performance (especially latency) when compared to known methods. In particular, data ferrying is used when mobility is high, while in the case of low

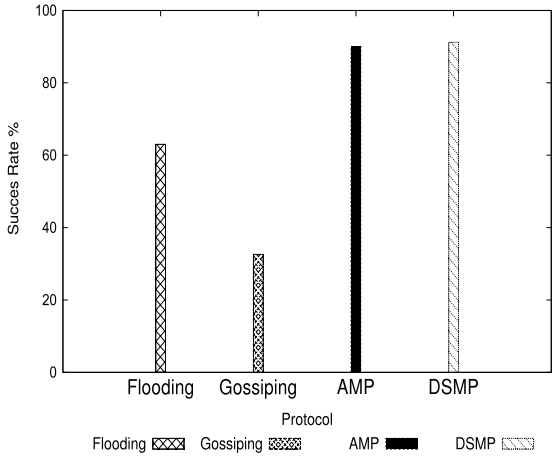


Fig. 11. Success rate of the protocols when nodes are assigned dynamically mobility roles.

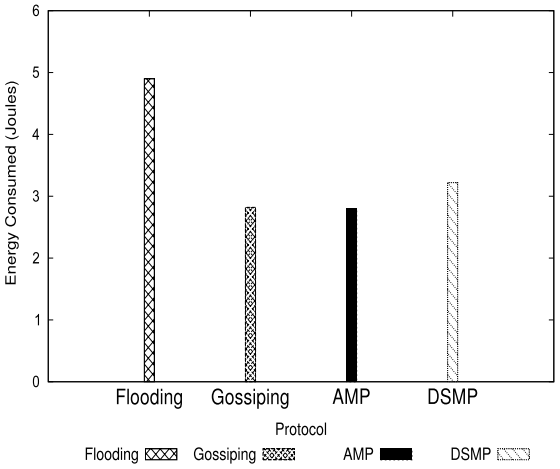


Fig. 12. Energy dissipation of the protocols when nodes are assigned dynamically mobility roles.

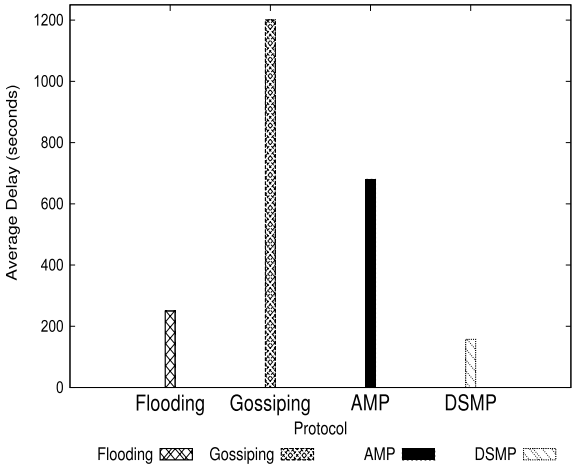


Fig. 13. Message delivery delay of the protocols when nodes are assigned dynamically mobility roles.

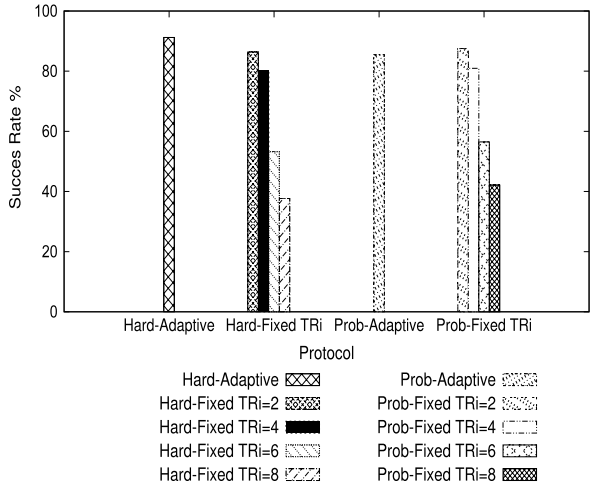


Fig. 14. Success rate of our protocol's variations when nodes are assigned dynamically mobility roles.

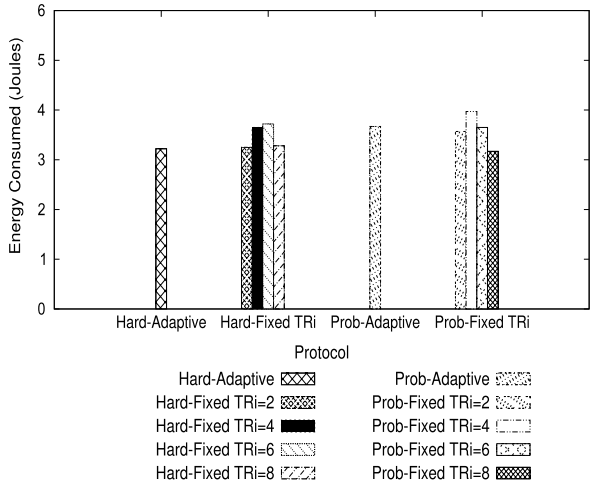


Fig. 15. Energy dissipation of our protocol's variations when nodes are assigned dynamically mobility roles.

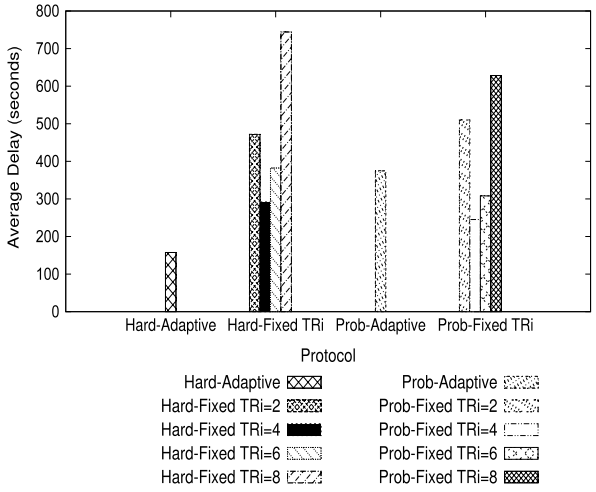


Fig. 16. Message delivery delay of our protocol's variations when nodes are assigned dynamically mobility roles.

mobility we either employ data redundancy or use long distance wireless transmissions to accelerate data propagation.

In future work, we plan to extend our schemes in the case of multiple (static) sinks. Also, to come up with (possibly new) methods for sensor networks where the sink(s) are also mobile (in that case lightweight distributed coordination techniques for sink mobility management may be needed as well). Actually, this research has a strong relation with geographic routing. We plan to indeed investigate the relation of our routing scheme which tries to maximize the speed toward the destination, whereas greedy geographic routing tries to maximize distance traveled to the destination as it forwards data. Also, to examine alternative probabilistic choices that could further improve performance.

Acknowledgments

This work has been partially supported by the EU Project HOBNET (ICT/FIRE STREP 257466). We wish to warmly thank the anonymous reviewers for insightful comments.

References

[1] L. Badia, E. Fasolo, A. Paganini, M. Zorzi, Data aggregation algorithms for sensor networks with geographic information awareness, in: Proceedings of WPMC, S.Diego, CA, US, 2006.
[2] L. Bononi, M. Di Felice, A cross layered MAC and clustering scheme for efficient broadcast in VANETs, in: Proceedings of 1-st IEEE International Workshop on Mobile Vehicular Networks (IEEE MoVeNet 2007), Pisa, Italy, October 8, 2007.
[3] L. Bononi, M. Di Felice, S. Pizzi, DBA-MAC: dynamic backbone-assisted medium access control protocol for efficient broadcast in VANETs, Journal of Interconnection Networks (JOIN) (MobiWac 2009) 10 (4) (2009) 321–344. ISSN: 0219-2659, 1793-6713.
[4] B. Bougard, F. Catthoor, D.C. Daly, A. Chandrakasan, W. Dehaene, Energy efficiency of the IEEE 802.15.4 standard in dense wireless microsensor networks: modeling and improvement perspectives, Design, Automation and Test in Europe (2005) 196–201.
[5] A. Boukerche, K. Abrougui, An efficient leader election protocol for mobile networks, in: International Conference on Wireless Communications and Mobile Computing (IWCMC), ACM, New York, NY, USA, 2006, pp. 1129–1134.
[6] A. Boukerche, D. Efstathiou, S. Nikoletseas, Adaptive, direction-aware data dissemination for diverse sensor mobility, in: 7th ACM/IEEE International Symposium on Mobility Management and Wireless Access Protocols (MobiWac 2009), ACM Press, MobiWac, Tenerife, Canary Islands, 2009, pp. 50–57.
[7] A. Boukerche, Y. Ren, Z. Zhang, Performance evaluation of an anonymous routing protocol using mobile agents for wireless ad hoc networks, in: 32nd IEEE LCN, pp. 893–900, 2007.
[8] T. Camp, J. Boleng, V. Davies, A survey of mobility models for ad hoc network research, Wireless Communications & Mobile Computing 2 (5) (2002) 483–502.
[9] I. Chatzigiannakis, A. Kinalis, G. Mylonas, S. Nikoletseas, G. Prasinos, C. Zaroliagis, TRAILS, a toolkit for efficient, realistic and evolving models of mobility, faults and obstacles in wireless networks, in: 41st ACM/IEEE ANSS, pp. 23–32, 2008.
[10] I. Chatzigiannakis, A. Kinalis, S. Nikoletseas, Sink mobility protocols for data collection in wireless sensor networks, in: 4th ACM MobiWac, pp. 52–59, 2006.
[11] I. Chatzigiannakis, S. Nikoletseas, P. Spirakis, Smart dust protocols for local detection and propagation, ACM Mobile Networks and Applications (MONET) Journal 10 (1) (2005) 9–16.
[12] Z. Chen, H. Kung, D. Vlah, Ad-hoc relay wireless networks over moving vehicles on highways, in: Proceedings of The 2001 ACM Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc'2001), 2001.
[13] A. Clementi, F. Pasquale, A. Monti, R. Silvestri, Communication in dynamic radio networks, in: 26th ACM PODC, pp. 205–214, 2007.
[14] A. Clementi, F. Pasquale, R. Silvestri, MANETS: high mobility can make up for low transmission power, in: 36th International Colloquium on Automata, Languages and Programming (ICALP), 2009.
[15] M. Elhadef, A. Boukerche, H. Elkadiki, Diagnosing mobile ad-hoc networks: two distributed comparison-based self-diagnosis protocols, in: 4th ACM MobiWac, pp. 18–27, 2006.
[16] E. Fasolo, C. Prehofer, M. Rossi, Q. Wei, J. Widmer, A. Zanella, M. Zorzi, Challenges and new approaches for efficient data gathering and dissemination in pervasive wireless networks, in: Proceedings of INTERSENSE, Nice, France, 2006.
[17] D. Gavalas, G. Tsekouras, C. Anagnostopoulos, A mobile agent platform for distributed network and systems management, Journal of Systems and Software 82 (2) (2009) 355–371.
[18] Y. Gu, D. Bozdag, R.W. Brewer, E. Ekici, Data harvesting with mobile elements in wireless sensor networks, Computer Networks 50 (17) (2006) 3449–3465.

[19] M. Ho, K. Fall, Poster: delay tolerant networking for sensor networks, in: IEEE Conference on Sensor and Ad Hoc Communication and Networks (SECON), 2004.
[20] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, D. Rubenstein, Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet, in: 10th ASPLOS, 2002.
[21] A. Kansal, A. Somasundara, D. Jea, M. Srivastava, D. Estrin, Intelligent fluid infrastructure for embedded networksm, in: 2nd ACM/USENIX International Conference on Mobile Systems, Applications, and Services (MobiSys04), 2004.
[22] A. Kinalis, S. Nikoletseas, Scalable data collection protocols for wireless sensor networks with multiple mobile sinks, in: 40th ACM/IEEE ANSS, Norfolk, VA, USA, 2007, pp. 60–69.
[23] A. Kinalis, S. Nikoletseas, Adaptive redundancy for data propagation exploiting dynamic sensory mobility, in: Proceedings of the 11th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), ACM Press, 2008, pp. 149–156. Also, in the Journal of Interconnection Networks (JOIN), 2009.
[24] C. Konstantopoulos, A. Mpitziopoulos, D. Gavalas, G. Pantziou, Effective determination of mobile agent itineraries for data aggregation on sensor networks, in: IEEE Transactions on Knowledge and Data Engineering, vol. 99, no. PrePrints, 2009.
[25] C. Konstantopoulos, A. Mpitziopoulos, D. Gavalas, G. Pantziou, Exploiting the cloning capability of mobile agents for cost-effective data fusion in wireless sensor networks, in: ISCC 2008, pp. 963–968.
[26] K. Lee, U. Lee, M. Gerla, Survey of routing protocols in vehicular Ad Hoc networks, 2010.
[27] C. Liu, J. Wu, Scalable routing in delay tolerant networks, in: Mobihoc, 2007.
[28] J. Luo, J.-P. Hubaux, Joint mobility and routing for lifetime elongation in wireless sensor networks, in: 24th IEEE INFOCOM, 2005.
[29] G. Pantziou, A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, B. Mamalis, Mobile sinks for information retrieval from cluster-based wsn islands, ADHOC-NOW (2009) 213–226.
[30] O. Powell, P. Leone, J. Rolim, Energy optimal data propagation in wireless sensor networks, Parallel Distributed Computing 67 (3) (2007) 302–317.
[31] T. Small, Z. Haas, The shared wireless infostation model: a new ad hoc networking paradigm, in: MobiHoc 03, 2003.
[32] A. Vahdat, D. Becker, Epidemic routing for partially-connected Ad Hoc networks, Duke Tech Report CS-2000-06, 2000.
[33] W. Wang, V. Srinivasan, K-C. Chua, Trade-offs between mobility and density for coverage in wireless sensor networks, in: Mobicom, 2007.
[34] Y. Wang, H. Wu, DFT-MSN: the delay/fault-tolerant mobile sensor network for pervasive information gathering, in: INFOCOM, 2006.
[35] W. Zhao, M. Ammar, E. Zegura, A message ferrying approach for data delivery in sparse mobile ad hoc networks, in: MobiHoc'04, 2004.

Azzedine Boukerche is a Full Professor and holds a Canada Research Chair position in distributed simulation and wireless and mobile networking at the University of Ottawa. He is the Founding Director of PARADISE Research Laboratory at Ottawa. His current research interests include sensor networks, mobile ad hoc networks, mobile and pervasive computing, wireless multimedia, QoS service provisioning, performance evaluation and modeling of large-scale distributed systems, distributed computing, large-scale distributed interactive simulation, and parallel discrete event simulation. He has published several research papers in these areas. He has also published a Book on "Algorithms and Protocols for Wireless and Mobile Systems" (Chapman & Hall/CRC 2005), a Book on "Algorithms and Protocols for Wireless and Mobile Ad Hoc Networks", Wiley&Sons, 2008, and a Book on "Algorithms and Protocols for Wireless Sensor Networks", Wiley&Sons 2008.

Dionysios Efstathiou has a Bachelor and Master degree in the Computer Engineering and Informatics Department of Patras University, Greece. He is a PhD Student and his research interests include Sensor Networks (communication protocols, routing protocols, data collection, mobility), Distributed and Mobile Computing with focus on mobile sensor networks and sensor network testbeds.

Sotiris Nikoletseas is a Professor at the Computer Engineering and Informatics Department of Patras University, Greece and Director of the Sensors Lab at CTI. His research interests include Algorithmic Techniques in Distributed Computing (focus on sensor and mobile networks), Probabilistic Techniques and Random Graphs, and Algorithmic Engineering. He has coauthored over 120 publications in Journals and refereed Conferences, several Book Chapters and two Books (one on the Probabilistic Method and another on Theoretical Aspects Sensor Networks, by Springer Verlag), while he has delivered several invited talks and tutorials. He has served as the Program Committee Chair of many Conferences, and as Editorial Board Member of major Journals. He has co-initiated international conferences on sensor networking. He has coordinated several externally funded European Union R&D Projects related to fundamental aspects of modern networks.