Adaptive, Direction-Aware Data Dissemination for Diverse Sensor Mobility

Azzedine Boukerche
PARADISE Research Laboratory,
University of Ottawa, Canada
boukerch@site.uottawa.ca

Dionysios Efstathiou
University of Patras, Greece
eustathi@ceid.upatras.gr

Sotiris Nikoletseas
Research Academic Computer Technology Institute (CTI) and
University of Patras, Greece
nikole@cti.gr

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 towards 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 4 times while keeping energy dissipation and delivery success at very satisfactory levels.

Categories and Subject Descriptors: C.2.1 Computer-Communication Networks Network Protocols [Routing protocols]

General Terms: Algorithms, Design, Performance

Keywords: Adaptation, Data Propagation, Mobile Sensors, Performance Evaluation, Wireless Sensor Networks

1. INTRODUCTION

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. Sensory data exchange between sensors and control nodes will drive important applications such as traffic and eco-system monitoring, smart homes and pollution control.

In such scenaria 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 neighbours) 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.

Related Work and Comparison. 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. Considering *sensor mobility*, [4] 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. The authors in [8] 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 theirmobility 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 [6] 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

MobiWac'09, October 26–27, 2009, Tenerife, Canary Islands, Spain.

mobility that capture more subtle variations both in speed and trajectory. Also, in [7] the authors investigate the trade-off between mobility of nodes and coverage of the network area. We exploit such trade-offs and handle high mobility as a "replacement" for connectivity and coverage.

In [5] a related but different mobility level notion as well as adaptive dissemination schemes have been proposed. Our mobility level here does not take into account (in contrast to [5]) dislocation from the origin but is instead direction-aware, in the sense that is captures how close to the sink a mobile sensor tends to become. Also, although our protocols are also adaptive on the mobility level, we here introduce long distance transmissions to accelerate data propagation in the case of low mobility.

Our Contribution. We investigate networks where the sensors themselves move. 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 towards 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 towards 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 perform extensive simulations demonstrating 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.

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 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 λ_{Beacon} $\frac{messages}{sec}$.

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* $\mathcal{C}$, able to maintain 64 data messages of size 36 bytes each. The protocol message overhead is much less. 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. 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 for each second and equals E_{elec}.

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 algorithm) and are aware of the dimensions of the network area; we note that relative position information suffices and no absolute values are needed. We model sensor movement through a high level mobility function which we symbolize by $\mathcal{M}$. Nodes generally follow different mobility functions and in fact a single node may follow different mobility functions from one time to another. The movement of each sensor node i at time t is characterized by a mobility level $M\ell_i(t)$.

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

3. DIRECTION-AWARE MOBILITY

The mobility parameters studied, in previous approaches such as [5], like distance travelled, 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 speed will travel the same distance independently of the trajectory followed, thus overall distance travelled 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 towards to the sink. Clearly, based on the above information, the latter's progress towards 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 local information 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).

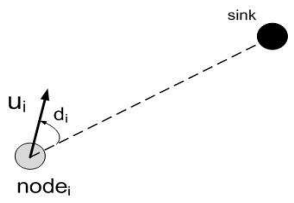

Figure 1: Example of estimating direction d_i.

Consider an integer K $\geq$ 1. Let v_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 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 towards 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 :

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

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 be fast when the nodes have high mobility level, while transmission to many neighbors incurs redundancy and long transmissions accelerate data propagation towards the sink at a high energy cost, however; so, the trade-off between these option must be judiciously handled.

Each node can ferry or disseminate the data 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 towards 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 tha 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 towards 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 towards 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 data to a number β of its neighbors at the given time (towards 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.

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.

1 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, in terms of the current and maximum mobility levels as follows:

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

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

1. **Ferrying :** Node i decides to ferry data based on hard threshold γ_i. If:

$$\gamma_i \geq \frac{1}{2}$$

then ferrying is chosen. We note that $0 \leq \gamma_i \leq 1$. Clearly, ferrying is suitable when γ_i is relatively high. The particular $\frac{1}{2}$ threshold value has been chosen after many simulations tries suggesting that this value 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 $\gamma_i < \frac{1}{2}$ then no ferrying is done. 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 υ_{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:

$$\gamma_{neigh\ i} \geq \frac{1}{10}$$

Intuitively, node i decides to transmit to β_i neighbors, when node i and at least one of its 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 towards 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 has been selected after extensive simulations, showing that this value leads to the 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:

$$\gamma_{neigh\ i} < \frac{1}{10}$$

Intuitively, node i decide 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×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 $\delta_1 = \left\lfloor \frac{dist_{sink}(j)}{R} \right\rfloor$. R is the transmission range of a mobile node and $dist_{sink}(j)$ is the Euclidean distance of node j from the sink. E.g. assuming that the transmission range of both nodes and sink is set to R = 70m, for a flat square region of size $1000 \times 1000m^2$ and if the location of the sink (hence also $dist_{sink}(j)$) may not be known, δ_1 can be calculated by using another distance, for example by setting $dist_{sink}(j) = \frac{D}{2}$, then $\delta_1 = 7$. 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. The cornerstone of the first method is the use of the mobility level and the distance from the sink and the core idea of the second method is an expanding ring search. The first 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 towards 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. For details, see full paper [1].

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 neighborhod.

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, 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. We present three different strategies for selecting the nodes to disseminate data 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 method probabilistically guarantees that the load distribution will be equally shared by the nodes. It is also relevant in cases of limited network knowledge.

Fittest Candidate Selection. Node i selects β_i of its neighbors such that $Ml_i(\mathrm{t}) < Ml_j(\mathrm{t})$ 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 $Ml_i(\mathrm{t}) < Ml_j(\mathrm{t})$, however if no such neighbors are found the rest of the nodes are examined probabilistically, in a way that favors nodes with high mobility. Let p_j the probability of sending a message to node j; p_j is calculated as follows:

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

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. As in direct neighbor selection in 4.5.1

Fittest Candidate Selection. As in direct neighbor selection in 4.5.1

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 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 $h_{opt} = \left\lceil \frac{dist_{sink}(j)}{R} \right\rceil$. $\mathrm{dist}_{sink}(\mathrm{j})$ 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 $\mathrm{dist}_{sink}(\mathrm{j})$) may not always be known, h_{opt} can be calculated by using another distance, for example by setting $\mathrm{dist}_{sink}(\mathrm{j}) = \frac{D}{2}$. 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}) and the mobility of the node compared to the maximum mobility. 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. 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. We define the following four characteristic mobility roles: Working movement $\mathcal{M}_{work}$, Walking movement $\mathcal{M}_{walk}$, Bicycle ride $\mathcal{M}_{bic}$ and Vehicular movement $\mathcal{M}_{veh}$.

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.

More information can be found in the full version of our paper [1].

6. EXPERIMENTAL EVALUATION

We implement our protocols on the **ns-2** simulation platform version 2.33, using the TRAILS toolkit [3] 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 $1000 \times 1000m^2$ and we always position $\mathcal{S}$ at $(500, 500)$. 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. The nodes and sink have significant energy resources $(100J)$ 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 at rate $\lambda_i = 0.025$ messages/second. Thus, the data generation phase lasts for about $800sec$, we simulate the network for $3600sec$ 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. Also, we note that since latency is low, the message probability is in fact very low as well; in addition, a relatively small memory $\mathcal{C}$ suffices, since few data messages are kept in memory each time. The transmission range of both nodes and sink is set to $R = 70m$. 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.

Node movement. We assign different mobility roles to the nodes of the network. In a first eperiment, 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, we assign **C1** to 25% of the nodes, **C2** to another 25%, **C3** to another 25% and **C4** to the remaining 25% of the nodes.

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. This protocol does not execute the neighbor discovery protocol, but it broadcasts a message simultaneously to all neighbors.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 travelling through the network and no copies are created. The third test case protocol is the *Adaptive Mobility Protocol (AMP)*, [5] which is an adaptive redundancy protocol for data propagation. We compare these 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 $\mathrm{TR}_i \in \{2, 4, 6, 8\}$. Also, we study characteristic variations of our method corresponding to selected different design alternatives.

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. 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 transmited/ferried exclusively towards 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.

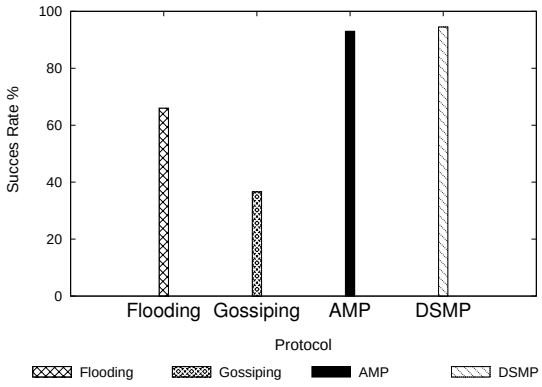

Figure 2: Success rate of the protocols when nodes are assigned a static mobility role.

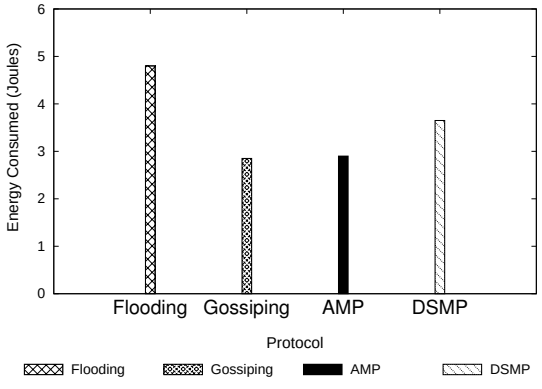

Figure 3: Energy dissipation of the protocols when nodes are assigned a static mobility role.

Variations of our protocol. Due to lack of space it was impossible to present experimental results for every possible variation of our protocol. Experimental results for some variations of our protocol can be found in our full paper [1].

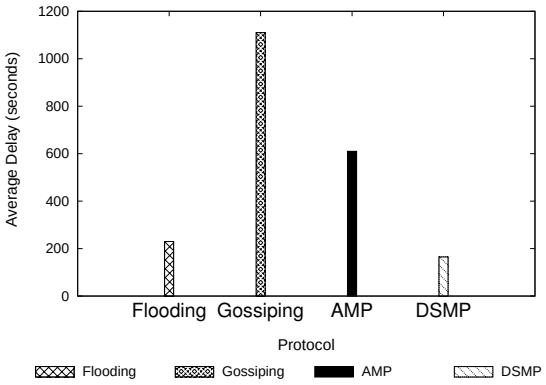

Figure 4: Message delivery delay of the protocols when nodes are assigned a static mobility role.

Second experiment. The results for this setup (varying mobility profiles) are similar in the sense that our protocol and its variations perform very good in this case as well. Due to lack of space these results are included in the full version of the paper [1].

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 Fig. 5, Fig. 6 and Fig. 7 we present the comparison of the two adaptive protocols (AMP, DSMP). We interestingly remark that our DSMP protocol behaves better in sparse netwroks as well; this is due to its ability to "jump" over sparse or even disconnected areas.

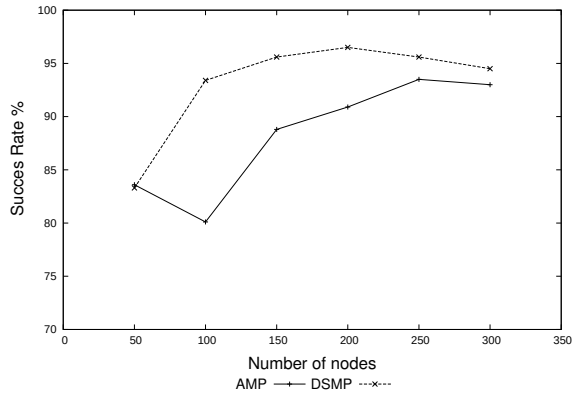

Figure 5: Success rate of the protocols for various densities.

7. FUTURE WORK

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). Finally, to compare with other ferrying and opportunistic routing mechanisms and more regular, human-related mobility patterns.

Acknowledgements This work has been partially supported by the IST Programme of the EU under contract number IST-2005-15964 (**AEOLUS**)

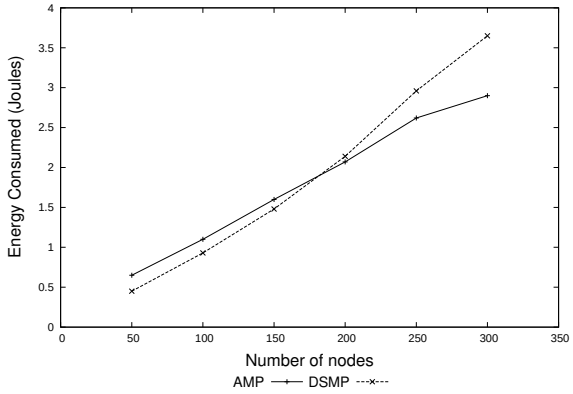

Figure 6: Energy dissipation of the protocols for various densities.

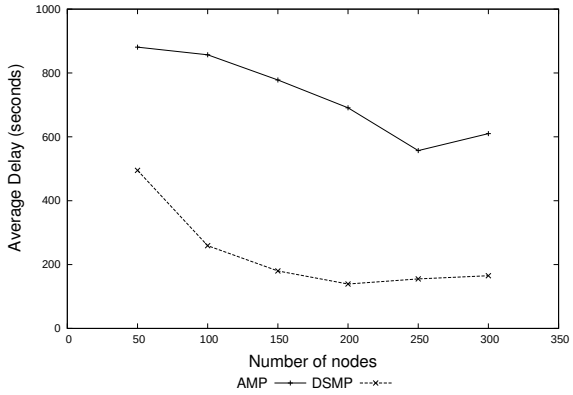

Figure 7: Message delivery delay of the protocols for various densities.

8. REFERENCES

[1] A. Boukerche, D. Efstathiou and S. Nikoletseas. Adaptive, Direction-Aware Data Dissemination for Diverse Sensor Mobility. `http://www.cti.gr/RD1/nikole/mobiwac09full.pdf`.

[2] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications & Mobile Computing*, 2(5):483–502, 2002.

[3] I. Chatzigiannakis, A. Kinalis, G. Mylonas, S. Nikoletseas, G. Prasinos, and C. Zaroliagis. TRAILS, a toolkit for efficient, realistic and evolving models of mobility, faults and obstacles in wireless networks. In *41st ACM/IEEE ANSS*, pages 23–32, 2008.

[4] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *10th ASPLOS*, 2002.

[5] A. Kinalis and S. Nikoletseas Adaptive Redundancy for Data Propagation Exploiting Dynamic Sensory Mobility. In the Proceedings of the *11th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, ACM Press, pages 149–156, 2008. Also, in *the Journal of Interconnection Networks (JOIN)*, 2009.

[6] C. Liu and J. Wu. Scalable routing in delay tolerant networks. In *Mobihoc*, 2007.

[7] W. Wang, V. Srinivasan, and K-C. Chua. Trade-offs between mobility and density for coverage in wireless sensor networks. In *Mobicom*, 2007.

[8] Y. Wang and H. Wu. DFT-MSN: The delay/fault-tolerant mobile sensor network for pervasive information gathering. In *INFOCOM*, 2006.