An Energy-Efficient Data-Dissemination Protocol in Wireless Sensor Networks

Zehua Zhou Xiaojing Xiang
State University of New York at Buffalo
Buffalo, NY, USA
{zzhou5, xxiang}@cse.buffalo.edu

Xin Wang
State University of New York at Stony Brook
Stony Brook, NY, USA
xwang@ece.sunysb.edu

Jianping Pan
University of Waterloo
Waterloo, Ontario, Canada
jpan@bbcr.uwaterloo.ca

Abstract

A typical data-dissemination sensor network consists of a large number of sensor nodes, which are normally energy-constrained and unlikely to be recharged in field. Redundant sensor nodes are often deployed to increase network robustness, and to extend network lifetime. Target and inquirer mobilities further bring more challenges to large-scale sensor networks. Frequent location updates for multiple inquirers and targets can drain the limited on-board energy excessively. We present EEDD, an Energy-Efficient Data-Dissemination *protocol to address both the target and inquirer mobility problem and the energy conservation problem. We extend network lifetime by adopting a virtual-grid-based two-level architecture to schedule the activities of sensor nodes. Furthermore, we propose an adaptive scheduling scheme and a data dissemination scheme to reduce sensing delay and packet forwarding delay. The simulation results show that our protocol saves up to twelve times energy without much increased delay when compared with protocol not considering energy efficiency.*

1 Introduction

Recent technology advances in micro-sensors enable the deployment of large-scale sensor networks. A typical sensor network consists of a large number of sensor nodes [3, 7] deployed in the environment being sensed and controlled. The on-board power and computation capacities of sensor nodes are normally limited. Nodes are prone to failures, and hence are often densely deployed to increase network reliability. The topology of sensor networks can change dynamically, especially when nodes fail in operation (e.g., due to running out of energy). Since the sensor network consists of a large number of nodes, recharging them is often infeasible. Energy optimization in sensor networks is difficult, since it involves not only reducing the energy consumption of a single sensor node, but also maximizing the lifetime of the entire network. How to save energy and extend the lifetime of wireless sensor networks impose a great research challenge.

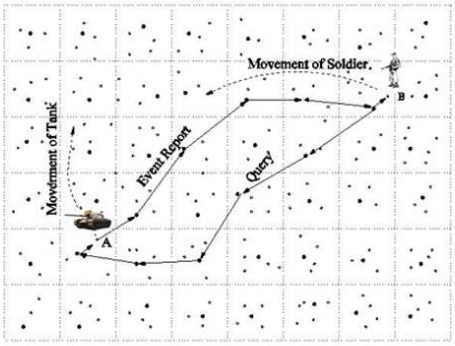

Figure 1. An enemy tank is detected and an event is sent to the soldier.

Target and inquirer mobilities (also known as source and sink mobilities in some contexts) bring more challenges to large-scale sensor networks. In this paper, a *source* refers to a sensor node that generates data reports for a target or an event of interest. A *sink* is a sensor agent of an inquirer that collects data reports from the sensor network. In Fig. 1, a soldier selects a nearby sensor node B as its sink node. A tank is detected by the sensor node A, which becomes a source node and generates data reports to be sent to the sink B. Both soldier (inquirer) and tank (target) can move quickly. In some recently proposed data dissemination protocols [4, 1, 12], the mobile inquirers continuously update their location information throughout the sensor field to inform the potential sources about their latest locations, which

causes increased transmission collisions and higher power consumption. TTDD [10] proposed using source-based grid structure to reduce overhead during inquirer mobility. However, when a target moves, the source nodes around the target will also change, which makes the maintenance of source-based grid structure very difficult. Also, TTDD did not consider energy efficiency in its design. None of the above approaches provides a scalable and efficient solution to deal with both target and inquirer mobility.

In this paper, we propose EEDD, an *Energy-Efficient Data-Dissemination* protocol, to address both the target/inquirer mobility and the energy conservation issues. We extend the lifetime of a sensor network by adopting a virtual-grid-based two-level architecture. Our grid structure is simple and generic (instead of source-based), and has very low overhead to build and maintain. Specifically, we introduce a two-level node activity schedule to match the features of the grid structure and the requirements of target tracking. At the coarse level, only a necessary set of working nodes are kept awake, and other nodes are turned into long-term sleep. At the fine level, each grid is divided into several sub-grids and working nodes in each sub-grid will alternatively stay active according to a schedule. This scheduling scheme further saves energy while efficiently controlling the maximum sensing delay. Reducing event delivery latency is important in most real-time sensor networks for monitoring or tracking purpose. Turning node into sleep can save energy, however, it may introduce additional delay. In our design, the network remains connected and an event can be sent out immediately. We further propose an efficient data dissemination scheme that considers the target location awareness scenarios to avoid the extra delay and overhead during query and data forwarding. In addition, an adaptive node scheduling scheme is introduced to reduce the detection delay for continuous events. The contributions of this paper are highlighted as follows.

- Develop schemes to reduce the query flooding cost while maintaining the network connectivity for timely data delivery.
- Design an efficient data dissemination scheme that can transmit packet quickly without incurring big overhead for route searching and maintenance.
- Propose a two-level node activity scheduling scheme to extend network lifetime, while keeping enough nodes awake to meet the sensing coverage requirement.
- Propose mechanisms to solve the target and inquirer mobility problems.
- Propose an adaptive scheduling scheme to reduce the initial sensing delay.

The rest of the paper is organized as follows. Section 2 compares EEDD with related work. We present the design of EEDD in Section 3, analyze delay and make improvement in Section 4, and evaluate EEDD performance in Section 5. Section 6 summarizes this paper.

2 Related work

There have been active research efforts on sensor networks in recent years. Recent work [4, 1, 12] generally assumes that inquirers are stationary or have low-mobility and hence can hardly be applied to deliver data to an inquirer with high mobility. TTDD [10] attempts to solve the inquirer mobility problem by using a two-tier data dissemination scheme. When a source node detects an event, it creates a virtual grid structure throughout the sensor field. A query from a sink node traverses two tiers to reach the source node. However, target mobility cannot be easily supported, as a source can no longer maintain a static grid upon target mobility. Our system can efficiently support both target mobility and inquirer mobility.

There are many clustering protocols proposed for energy saving, for example, LEACH [2] and PACT [6]. LEACH uses cluster heads to aggregate data information. However, it assumes that a cluster head can communicate with base stations directly. PACT uses passive clustering which allows nodes to become communication backbone nodes in turn. Different from these clustering protocols, our virtual grid structure is simple and there is no need to exchange role information among neighbors and to maintain dynamic routing tables. This further reduces the energy consumption and storage requirements at sensor nodes.

GAF [9] proposes to turn off unnecessary nodes within transmission range for energy conservation. However, in sensor networks, the detection range of most sensor nodes is smaller than their transmission range. To detect events in a timely fashion, it is necessary to keep a certain sensing coverage. PEAS [11] considers the detection range when forming a connected network. Our coarse level scheduling assumes a similar strategy, but is more general in the number of nodes that can be kept active. S-MAC [13] tries to reduce the additional delay caused by the node sleep, but there is still a considerable time lag at each hop. While no additional delay is introduced in our system at intermediate nodes. The power saving (PS) mode in IEEE 802.11 DCF is designed for single-hop networks and requires global synchronization. Tseng et al. [8] propose three sleep schemes to improve the PS mode in IEEE 802.11 multi-hop networks. But the overhead and delay can be large, as a sender has to wait for its neighbors along the data forwarding path to wake up before sending out the actual packet.

3 EEDD design

In this section, we describe the EEDD protocol in detail. We make the following assumptions in EEDD. Each sensor

node is aware of its location after deployment (e.g., using some localization techniques). Sensor nodes are stationary, while targets and inquirers can move.

3.1 Network initialization

We will present how to initialize the sensor network including working nodes selection, virtual grid formation and grid head election in this section.

3.1.1 Working nodes selection

Initially, all nodes are in *working* mode. To decide if a node should stay active, the node needs to go through a detection process by changing its mode to *detecting* after waiting for a random period. The node then broadcasts a detecting message to its neighbors to detect the number of active nodes within the detection range (not the transmission range). If a *working* node in the detection range of the message sender (based on sender's location) has energy higher than E_{gridhead}, it sends back a response message. If the number of response messages received exceeds N_{node}, the detecting node will be considered redundant and go to long-term sleep. We call this kind of sleep nodes *redundant sleep nodes*. Otherwise, the node will enter the working mode. The value of N_{node} depends on the node redundancy and sensing coverage requirements. After this initial stage, a topologically-connected network is established and a working node won't go back to long-term sleep until it is dead. After a time period T_{sleep}, a redundant sleep node will wake up and enter the detecting mode to determine if it should stay active by using the procedure described above. Before changing the mode from detecting to working, the node needs to announce the change to the grid head.

3.1.2 Virtual grid formation and grid head election

The whole sensor field is divided into small virtual grids (see Fig. 2). Every grid has a grid ID. A sensor node can calculate its grid ID (a, b) from its location (x, y) as: $a = [\frac{x-x_0}{grid\_size}]$ and $b = [\frac{y-y_0}{grid\_size}]$, where (x_0, y_0) is the location of the virtual origin, which is set as a system parameter at the network initialization stage. For simplicity, we assume that all the grid IDs are positive. To ensure that all the nodes in adjacent grids can communicate with each other directly, the grid size is set to less than $1/2\sqrt{2} * R_{\text{trans}}$, where R_{trans} is the transmission range.

Initially, each active sensor node s will compete for being a grid head by broadcasting an announcement (carrying its position and address) after waiting for a random time period $f(s)$, which is determined as follows:

$$f(s) = Random(T_{\text{grid}})/energy(s). \quad (1)$$

Here, $energy(s)$ is the remaining energy of node s and $Random(T_{\text{grid}})$ is a random value between 0 and constant

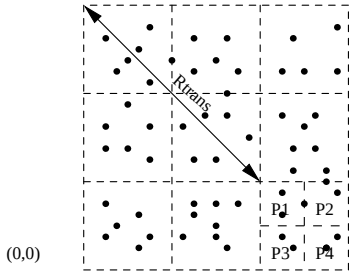

Figure 2. Virtual grids of sensor nodes.

T_{grid}. A node with more remaining energy has a higher chance to become a grid head. Once a node receives a grid head announcement from its own grid, it will give up its own attempt. When there are multiple announcements, the one with larger address wins the competition. A grid head will re-announce its leadership when it detects the existence of another leader with a smaller address (e.g., by overhearing a message sent to or from another leader). After election, there is one grid head in each grid.

3.2 Scheduling scheme

There are two levels of node activity scheduling. At the coarse level, each node wakes up periodically after long-term sleep to contend for becoming a working node as described in Section 3.1.1. At the fine level, working nodes can wake up after short-term sleep in turn.

The whole field is divided into virtual grids as described in the previous section. Additionally, if the event is continuous, each grid is further divided into four sub-grids, $P1$, $P2$, $P3$ and $P4$ (shown in Fig. 2). Also, a time interval T_{interval} is divided into four time slots, $T1$, $T2$, $T3$ and $T4$. Each sub-grid will be associated with one time slot, $T1$ for $P1$, $T2$ for $P2$, $T3$ for $P3$ and $T4$ for $P4$. During any time slot, only the working nodes in the corresponding sub-grid need to stay active. For example, nodes in sub-grid $P1$ can be active in $T1$ and turned to sleep during the other time slots. There is a tradeoff in the selection of T_{interval}. Too large a T_{interval} will increase the initial event detection delay as described in Section 4.2. If T_{interval} is too small, there is not enough time for a sensor node to get a chance to send out a packet. Reducing the number of working nodes at a time will not only reduce energy consumption but also reduce communication collisions.

With the coordination of the grid head, the synchronization within a grid is not difficult. The global synchronization is not required. Different grids can have different time schedules. The grid heads will stay active to reduce the transmission delay. When an event is generated and needs to be sent to the sink, the packet is forwarded by the working node to the neighboring grid head closer to the sink node until reaching the sink.

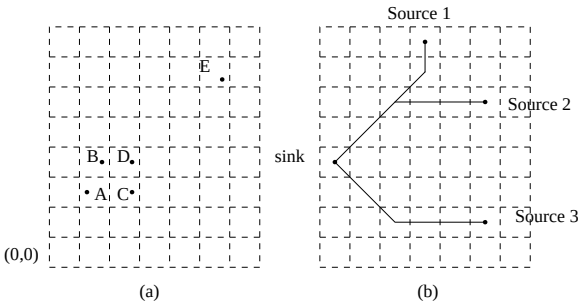

Figure 3. Three routing candidates and diagonal-first routing path.

3.3 Data dissemination

For more efficient data dissemination, we ascribe inquirers into three different types and the data dissemination strategies are different for these types. The three types are:

1. Target location aware: the inquirer knows the current location of the target.
2. Target area aware: the inquirer knows the area within which the target is currently located, but does not know the exact location of the target.
3. Target location unaware: the inquirer does not have any location information of the target.

3.3.1 Target location aware data dissemination

For the first type, the inquirer wants to monitor some target at a specified location. The sink node, representing the inquirer, first registers the query with the head of its grid, which will forward the query to the head of the neighboring grid that is closer to the source node. The query will be progressively propagated until it reaches the head of the source grid, which will then notify all the working nodes in its grid about the query.

For a forwarding node, normally, there are three candidate grid heads that are closer to the source. For example, in Fig. 3(a), if the sink node is located in grid $A(1,2)$ and the source node is located in grid $E(5,6)$, three candidate grid heads are in $B(1,3)$, $C(2,2)$ and $D(2,3)$. In order to minimize the number of traveled grids on the path to the source, the diagonal candidate node is preferred until the query reaches the grid head whose grid ID has the same vertical or horizon coordinate value as that of the source grid. The grid head will then forward the query vertically or horizontally to the source grid as shown in Fig. 3(b). Our *diagonal-first* routing path has the same number of grid hops as the straight-line path between the source and the sink node, which is the shortest path. The routing scheme is simple and there is no need to actively maintain a routing path, which is very important for the resource constrained sensor networks. If multiple sources exist and some segment of the path is shared by several sources, for example, the path segment from sink to the grid (2, 5) is shared by Source 1 and 2 in Fig. 3(b), only one copy of the query is sent along the shared path and then it is duplicated and sent to different sources at the branching grid (grid (2, 5)). The data forwarding from the source to the sink also follows diagonal-first routing strategy, but the path is normally different from the one from the sink node to the source node. This routing strategy not only reduces collisions in two directions, but also leads to more balanced transmission paths.

When a sensor node wants to send a data packet, it first calculates the destination's grid ID (x_2, y_2) and its own grid ID (x_1, y_1), and then calculates next hop's grid ID ($x_{nexthop}$, $y_{nexthop}$) using the following equations:

$$\begin{cases} x_{nexthop} = x_1 + \frac{x_2 - x_1}{|x_2 - x_1|}, \text{when } \mathrm{x_2} \neq \mathrm{x_1} \\ x_{nexthop} = x_2, \text{otherwise} \end{cases} \quad (2)$$

$$\begin{cases} y_{nexthop} = y_1 + \frac{y_2 - y_1}{|y_2 - y_1|}, \text{when } \mathrm{y_2} \neq \mathrm{y_1} \\ y_{nexthop} = y_2, \text{otherwise.} \end{cases} \quad (3)$$

The forwarding node then broadcasts the data packet with the next-hop grid ID inserted in the packet header. When a neighboring grid head receives this packet, if its grid is the designated next-hop, it forwards the packet following the same procedure; otherwise, it drops the packet. To improve the data delivery ratio, the receiving grid head will send an acknowledgment to the sender. If the sender cannot get the acknowledgment for a certain time period T_{resend}, it will resend the packet. After trying several times, it will try the other two candidate grid heads (Fig. 3). If all neighboring grid heads couldn't forward the packet, the packet is dropped. When the packet header contains multiple next hops, a receiving grid head waits for a random time before sending back the acknowledgment to avoid collision.

When a data packet reaches the destination grid, the grid head will check its forwarding entry. If there is a forwarding request from a moving away inquirer, the grid head will further forward the data packet to the new grid where the inquirer is currently located; otherwise, the grid head will broadcast the data packet in its grid so that the sink node can receive it. We will discuss the rule for creating forwarding entry in detail in Section 3.5.

3.3.2 Target area aware data dissemination

For the second type, the inquirer wants to detect some events in a subarea. In this case, the query is forwarded toward the source area. When the request reaches the source area, it will be flooded to all grid heads in this sub-area. For example, in Fig. 4, the soldier in $(2,3)$ wants to know the information of enemy tanks in area $[6,6,8,8]$. The query will be forwarded by a diagonal path along grid $(3,4)$, $(4,5)$ and

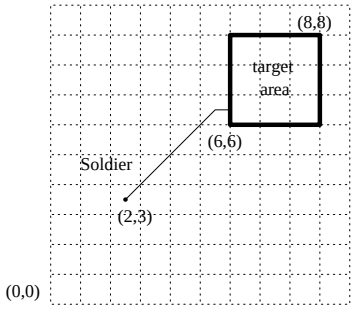

Figure 4. Routing path for source area aware data dissemination.

$(5, 6)$. Then it will pass grid $(6, 6)$ by a horizontal path and arrive at the target area $[6, 6, 8, 8]$. The query will then be flooded in this sub-area. Instead of involving all the nodes in flooding, only grid heads participate in the forwarding process, which can reduce energy consumption significantly. The working nodes can get the query by listening to the forwarding from local grid heads. A sensor node that is in sleep can later solicit the ongoing queries from the grid head. Hence, all the sensor nodes in this sub-area will be aware what kind of events they need to detect.

3.3.3 Target location unaware data dissemination

For the third type, the inquirer wants to detect some kind of events in the whole area. The query will be flooded throughout the field and reach all grid heads in the field. The working nodes will overhear the bypassing messages, and later all sensor nodes will know the events to detect. Even in this case, since queries are forwarded only by the grid heads, it can still save a lot of energy and lessen the broadcast storm problem.

3.4 Grid head maintenance

Since a grid head will play important role as to be discussed, it is important to maintain an active grid head within each grid. When the energy of a grid head is below a threshold E_{gridhead}, it will broadcast a request to reelect a new grid head. All the active sensor nodes in the grid with energy above E_{gridhead} will participate in the competition using the election procedure described above. The old grid head will send all the grid information to the new grid head. If no grid head is elected, either due to the loss of the control messages or because the energy of all the active sensor nodes in a grid is lower than E_{gridhead}, the old grid head will continue its role and broadcast the reelection request periodically. If there is no new grid head elected for several cycles, the grid head assumes that the energy of all active nodes is lower than the threshold. For reliability purpose, the grid head will broadcast the grid information so that each active node keeps a copy of the information. The grid head will send reelection request again when detecting that a new node becomes active. If a grid head dies suddenly, there is no grace period for grid head reelection. When a sensor node detects that the grid head is dead, i.e., it fails to send packets to the leader node for several times, it will send a head reelection request to the grid. The nodes with energy higher than E_{gridhead} will compete for the election, and the new leader needs to recollect the information about the grid. If a reelection message is sent from a node other than the grid head (e.g., the grid head dies) and all the active nodes have energy lower than E_{gridhead}, if no leader is announced within a certain period, all the active nodes will compete for being a grid head but with a flag set indicating low energy. If the grid head has sent each (low energy) node a copy of the grid information before it died, the reelected leader will already have the information and there is no delay for the leader to accumulate the information again.

3.5 Target/inquirer mobility

Although sensor nodes are generally stationary, inquirers and the targets can move quickly. We will discuss how to handle the inquirer and target mobility in this section.

3.5.1 Inquirer mobility

When moving within the same grid, an inquirer does not need any location update. The query results will be forwarded by its grid head. When the inquirer moves from one grid to another, it will pick another nearby sensor node as its sink node, and register with the head of the new grid by providing both the query information and the original grid ID. Sometimes, an inquirer moves out of a grid and then moves back shortly. To avoid forwarding loop, the head in the new grid will first check whether it already has a forwarding entry for the inquirer. If it has, the head will delete the old forwarding entry; otherwise, the grid head will send a forwarding request to the original grid, which will add an entry into its forwarding table and forward the data packets for the inquirer to the new grid later. The forwarding path may be long if the inquirer keeps moving from one grid to another. To avoid this case, before the query expires, the inquirer can send a location update that is similar to a new query.

3.5.2 Target mobility

To meet different requirements and to track target mobility more efficiently, we introduce the concept of *intelligence* in this work. Whether the data packets will be generated or forwarded to the sink node depends on the intelligence level of the source node. A sensor node is assigned an intelligence type, NORMAL or SMART. For a SMART node,

a number between 0 and $MaxSmartness$ is used to represent the SMART level of the node. The intelligence assignment is controlled by the inquirer and flooded during the network initialization phase. A NORMAL sensor node will only generate data packets if detected event has been inquired, while a SMART sensor node will search for the relevant query with the effort corresponding to its intelligence level if it is not inquired for the event.

During the target location aware data dissemination, the sink node will send the query to the source node whose original location is known. But the event may be detected by a node other than the queried source node. If the detecting node is SMART, in the case it knows the query (e.g., by overhearing), it will just send back the data directly towards the inquirer; otherwise, it will send the data with its SMART level to the local grid head. Upon receiving the event, the grid head will search for the queries within the maximum *ring* of surrounding grids, where the maximum *ring* number is equal to the SMART level of the event source. When the SMART level is 0, the grid head will only check its query table. If a query for this event is found, the data packets will be sent to the sink; otherwise, the data packets are dropped. When the target moves out of its detection range, both the NORMAL and SMART nodes keep silent.

Similarly, for the case that a subarea has been queried, if the target is detected by a node outside the queried area, the node will keep silent if it is NORMAL and in the case it is a SMART node, it will generate data packets and search for queries within the range according to its SMART level. When target location is unaware, the sink queries some events in the whole area. Since every node is queried, when a node detects the target, it will generate and send out data packets no matter it is NORMAL or SMART.

4 Delay analysis and improvement

4.1 Average delivery delay

This section analyzes the average delivery delay of the proposed scheme. Compared with 802.11-like protocols, an extra sleep delay is introduced in EEDD due to sensor nodes going sleep periodically.

If a node is in sleep mode, when an event happens, the node cannot detect the event until it wakes up. This extra delay is common for a protocol which adopts periodic wake-up and go-sleep scheme. In our scheme, the time delay is between 0 and $3/4$ of T_{interval}, i.e., the node goes to sleep just before the event happens and detects the event after $3/4$ of T_{interval} when it wakes up. There is no extra delay if the sensor node is awake. The average delay for the first hop is

$$D_1 = 3/8 * T_{\text{interval}} + T_{\text{other}},$$

where T_{other} is other delays except for the sleep delay. T_{other} can include *carrier sensing delay*, *transmission delay*, *propagation delay*, *processing delay*, etc.. These delays are inherent to a multi-hop network using contention-based MAC protocols, and have the same impact on both EEDD and 802.11-like protocols.

Different from other periodic wake-up and go-sleep schemes, EEDD does not have extra delay in the forwarding path, since grid heads are always awake. Suppose there are N hops from the source to the sink. The total average delivery delay is

$$D(N) = 3/8 * T_{\text{interval}} + N * T_{\text{other}}.$$

The overhead ratio that the sleep delay introduces is

$$P_{\text{sleep}} = \frac{3/8 * T_{\text{interval}}}{3/8 * T_{\text{interval}} + N * T_{\text{other}}}.$$

Apparently, the longer the forwarding path is, the smaller overhead ratio due to the extra sleep delay.

All grid heads are awake in the basic EEDD scheme. To save more energy, some grid heads can go sleep periodically as well. This strategy trades off more energy saving with a longer delivery delay. In this case, the extra sleep delay for event detection is the same as the one when every grid head is awake. But extra delay will also be introduced at intermediate nodes in the forwarding path as they need to wait for the wake up of neighboring grid heads. The average delay for each hop is

$$D_N = 3/8 * T_{\text{interval}} + T_{\text{other}}.$$

The total average delivery delay is

$$D(N) = 3/8 * N * T_{\text{interval}} + N * T_{\text{other}}.$$

The overhead ratio introduced by the sleep delay is

$$P_{\text{sleep}} = \frac{3/8 * T_{\text{interval}}}{3/8 * T_{\text{interval}} + T_{\text{other}}}.$$

4.2 Adaptive node scheduling

Although only the first hop has the extra sleep delay, the total average delay can still be large when T_{interval} is large. To further reduce the total average delay, adaptive scheduling can be adopted. When a source sends out the first few data packets, nearby awake working nodes can overhear these packets. If the distance between an awake working node and the source is less than a certain threshold, for example, the grid size, the working node can decide to stay awake to be ready for detecting the target. When the distance is greater than the threshold or the working node has not heard packets from the source for a while, it will follow the normal activity schedule. By doing so, we only have the

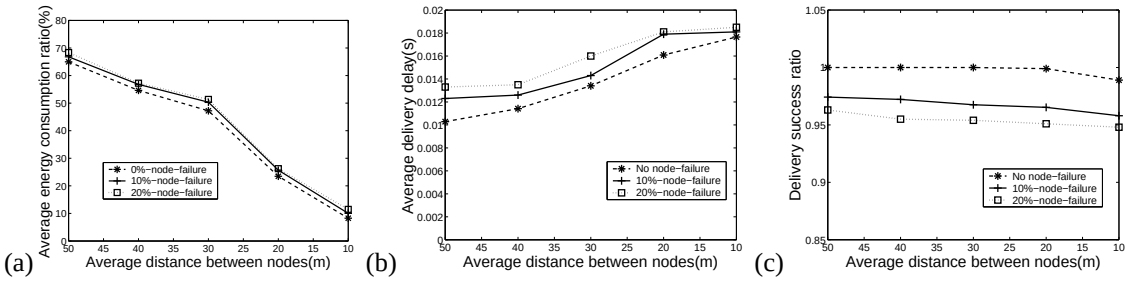

Figure 5. Performance with different node densities and reliabilities: (a) average energy consumption ratio; (b) average delivery delay; (c) delivery success ratio.

extra sleep delay for the first few packets at the first hop. Compared with the entire field, the area around the source is small. Even though these awake working nodes cannot enter sleep mode for a while, overall the system can still benefit from our two-level node activity scheduling scheme, while maintaining a low average delay. This scheme is efficient when the event is continuous, and most sensor networks are deployed to track the continuous events.

5 Performance evaluation

In this section, we evaluate the performance of EEDD through simulation. We first describe our performance metrics and simulation scenarios. We then evaluate the system performance with given scenarios and parameters. Finally, we show the comparisons between our scheme and TTDD. The results confirm that EEDD can deliver data efficiently and handle both target and inquirer mobilities well.

5.1 Scenarios and metrics

We implemented EEDD in Network Simulator version 2 (NS2) [5] and selected IEEE 802.11 as our MAC protocol. The transmission, reception, idle and sleep power consumption of sensor nodes are 24.75 mW, 13.5 mW, 13.5 mW and 0.015 mW, respectively [13]. The transmission range is 250 m, and the detection range is 20 m. The grid size is set to 80 m to make sure it's less than $1/2\sqrt{2} * R_{\text{trans}}$. $N_{\text{node}} = 1$, since it is already enough to have a reliable connected network. T_{interval} depends on application requirements. We set T_{interval} to 0.6 s, and it only impacts the event detection delay for the first few packets at the first hop. T_{grid} for grid head election is set to $2 * T_{\text{interval}}$. T_{sleep} is set to $Random(0, 10)$ s. With a larger T_{sleep} and a correspondingly longer T_{interval}, more energy can be saved.

We use three metrics to evaluate the performance of EEDD. The *average energy consumption ratio* is defined as the ratio of the average energy consumption to the initial energy in the network. This metric also indicates the overall lifetime of sensor nodes. The *delivery success ratio* is the ratio of the number of successfully delivered data packets to the number of data packets generated by the source. This metric shows the data delivery efficiency. The *average delivery delay* is defined as the average time delay between the moment a source transmits a packet and the moment a sink receives the packet. This metric indicates how quick the sink can get reports from the source. The extra sleep delay is not included here since it largely depends on T_{interval} and only exists for the first few packets at the first hop.

In most scenarios, we use a field size of $400 * 800$ m^2 where $3,200$ nodes are randomly distributed. The average distance between nodes is 10 m. By default, one source and one sink without mobility are assumed except in the performance studies on the impacts of mobility, and the source location is known by the sink. The source generates one report packet per second. Each simulation lasted for 800 s.

5.2 Simulation results

5.2.1 Impact of node density and reliability

Higher node density can help extend network lifetime. In order to evaluate how node density impacts the performance of EEDD, the average distance between sensor nodes is varied by reducing the number of deployed nodes. To study how node failure affects EEDD, we allow randomly-chosen nodes to fail simultaneously at time 30 s.

Fig. 5(a) shows the energy consumption ratio when node density and failure ratio increase. For a fixed failure ratio, when the average distance between nodes is greater than 20 m, most energy saving is due to working nodes going sleep periodically. When the average distance between two nodes is 40 m, there are four sensor nodes in each grid and one in each sub-grid on average since the grid size is 80 m. The grid head is always awake. The working nodes in a sub-grid are awake periodically. If the active sub-grid is the one where the grid head is located, only the grid head is awake in the whole grid; otherwise, one sensor node in the active sub-grid and the grid head are both awake. Hence, nearly two of four sensor nodes are awake in this scenario. When node density becomes higher, only working nodes

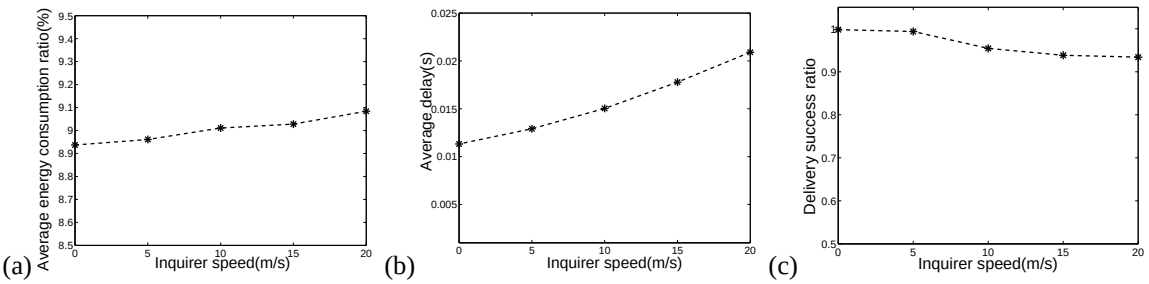

Figure 6. Performance with different inquirer speeds: (a) average energy consumption ratio; (b) average delivery delay; (c) delivery success ratio.

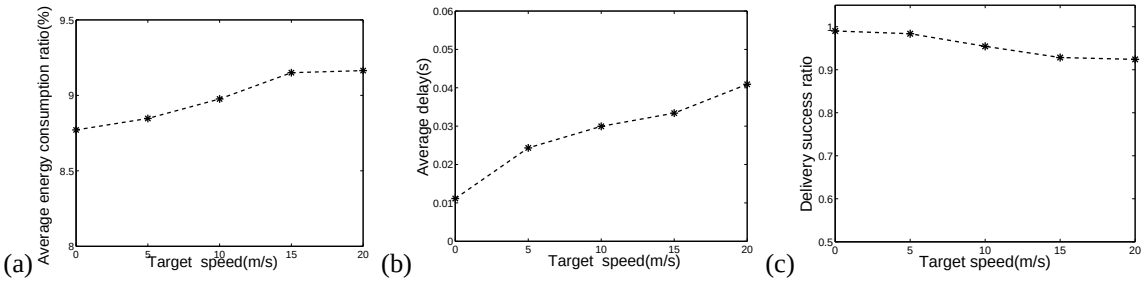

Figure 7. Performance with different target speeds: (a) average energy consumption ratio; (b) average delivery delay; (c) delivery success ratio.

in active sub-grid (about $1/4$ of working nodes in a grid) plus a grid head are awake, which reduces energy consumption. When the average distance between sensor nodes becomes smaller, more sensor nodes can go to long-term sleep at coarse level, which leads to further energy saving. Due to the memory limitation of the simulation tool, we cannot simulate the scenario having even more nodes. But we know from the trend that more energy will be saved when we have a higher node density. For a fixed node density, the energy consumption increases when node failure ratio increases. This is because that when more nodes fail, node density becomes lower. The energy saving due to a higher node density is in fact reduced.

Fig. 5(b) shows the average delivery delay when node density increases and node failure ratio increases. For a fixed failure ratio, the average delay increases when node density increases. More working nodes will be awake at a higher density, which may create more collisions and hence increase the delivery delay. When the average node distance is below the detection range, i.e., 20 m, the increase of delay is much smaller. This is as expected, as redundant nodes within the same detection range are turned off without impacting the transmission. For a fixed node density, the average delay increases when node failure ratio increases. When a grid head fails to forward packets to its neighboring grid head, it will try another two grid head candidates that are also closer to the destination. It takes a longer time for data packets to arrive at the sinks when an alternative path is used.

Fig. 5(c) shows that the success ratio drops slightly when node density increases, as more collisions can happen. For a fixed node density, the success ratio drops when node failure ratio increases. But the delivery success ratio is still above 95% even when the failure ratio is 20%. This confirms that EEDD is resilient to node failure.

5.2.2 Impact of inquirer mobility

Now we evaluate the impact of inquirer mobility on EEDD. Fig. 6(a) shows that energy consumption ratio increases when the inquirer has a higher speed. This is because with the increased inquirer mobility, more location update messages are needed to inform the original grid head, and more grid heads are involved in routing packets. Fig. 6(b) and (c) show that the delay increases while success ratio decreases slightly when the inquirer has a higher speed. When an inquirer moves faster, the forwarding path will become longer. Also more location update messages would cause more collisions. Both factors contribute to the longer delay and lower delivery ratio. However, even when the inquirer moves at a speed of 20 m/s, the success ratio is still above 90%. This indicates that our forwarding mechanism works quite well when the inquirer changes its location quickly.

5.2.3 Impact of target mobility

To study how target mobility impacts the performance of EEDD, we vary the speed of the target. An inquirer queries a specific sub-area $[4, 2, 7, 4]$. The target will randomly move in the whole area. We set the sensor nodes to have

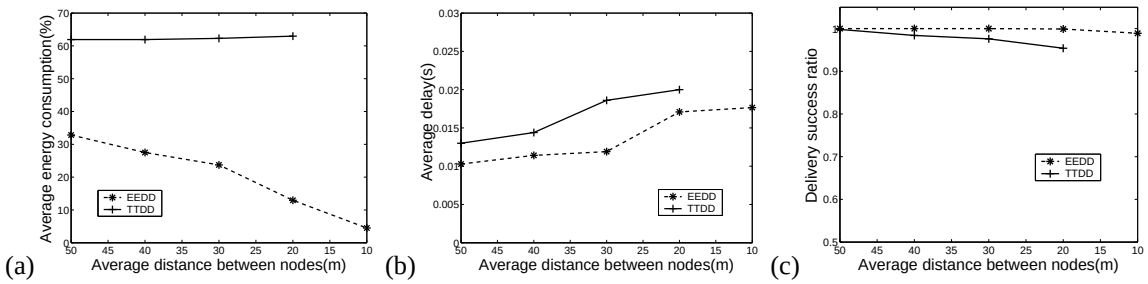

Figure 8. Performance comparison with different node densities: (a) average energy consumption ratio; (b) average delivery delay; (c) delivery success ratio.

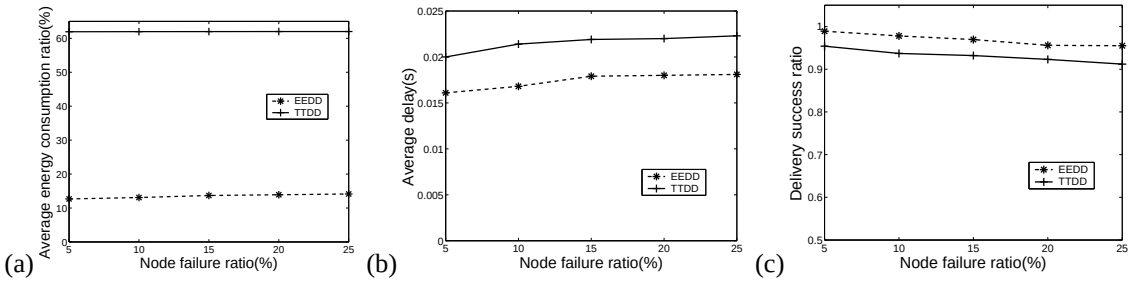

Figure 9. Performance comparison with different node failure ratios: (a) average energy consumption ratio; (b) average delivery delay; (c) delivery success ratio.

a SMART level of 4, which means that if the event is not queried by the inquirer in a grid, the grid head will query four rings of grids around the target region to search for a possible query.

Fig. 7(a) shows that the energy consumption increases when target mobility increases. When a target moves faster, there is a larger possibility that the target leaves the queried area. To search for the query, more grid heads will be involved in data routing, and hence consume more energy.

Fig. 7(b) shows that the average delay increases when target mobility increases, as it takes a longer time to arrive at the queried area. The delay for a moving target is much longer than that of a stationary target. When the source searches for the queried sensor nodes, it uses the flooding technique among grid heads, which can cause much higher delay.

Fig. 7(c) shows that the success ratio decreases slightly when target mobility increases. When a source node searches for the query using flooding, more collisions happen. This can reduce delivery success ratio. When the target moves at a maximal speed of 20 m/s, the success ratio is still quite high.

5.2.4 Comparison with TTDD

We compare the performance of EEDD and TTDD by varying node density, and by varying the node failure ratio when average inter-node distance is 20 m. We double the initial energy of sensor nodes so that TTDD will not run out of energy. We cannot simulate TTDD when the average distance between nodes is shorter than 20 m due to memory saturation in simulation (TTDD needs to maintain more per-node information). With too small average distance between nodes, more nodes need to be handled and TTDD simulations would fail because of the memory constraint.

Fig. 8(a) shows the energy consumption ratio comparison between EEDD and TTDD with different node densities. As described in previous section, two out of four sensor nodes are awake during most time periods when the average distance between nodes is 40 m. We can see that EEDD only consumes less than half of the energy consumed in TTDD when the average distance is 40 m. When node density becomes higher, the energy consumption of EEDD decreases as more working nodes can go sleep and save energy. When the distance is 10 m, EEDD saves energy significantly, with the energy consumption at distance 50 m six times larger.

Fig. 8(b) shows that average delay for both EEDD and TTDD increases when node density increases. This is because there are more working nodes at higher node density, which will cause more collisions. The delay for EEDD is lower than that for TTDD, as EEDD has fewer transmission collisions due to node sleep. Remember that the initial extra sleep delay is not counted in EEDD because it largely depends on T_{interval} and only appears when the source sending out the first few packets. The impact of the initial extra delay will be averaged out as time goes on when the source keeps tracking target.

Fig. 8(c) shows the delivery success ratio for EEDD and TTDD. The success ratio for EEDD decreases a little with the density, while decreases quicker for TTDD due to more working nodes and hence more collisions. Also, EEDD tries to find alternate path when the packet cannot be forwarded. It can improve the delivery success ratio at the cost of longer delay. Overall we can see that TTDD consumes energy six times larger than EEDD when average distance between nodes is 20 m and according to the trend it would save up to twelve times when the distance is 10 m. More energy can be saved when the node density becomes higher. The average transmission delay is similar since the extra sleep delay only impacts the first few packets at the first hop. With fewer working nodes awake and fewer collisions, EEDD has a lower average delay.

Fig. 9(a) shows the impact of node failure on EEDD and TTDD when the average distance of sensor nodes is 20 m. When more nodes fail, the energy consumption increases slightly for EEDD. This is because that node density decreases when node failure ratio increases, and the energy saving due to high node density decreases. Fig. 9(b) shows that average delay increases for both EEDD and TTDD when node failure ratio increases. TTDD takes time to recover from node failure. EEDD takes time to reelect grid head to replace the dead one, and simultaneously tries to use alternate paths for routing packets. Both approaches increase the delivery delay. Fig. 9(c) shows that the delivery success ratio for both EEDD and TTDD decreases when node failure ratio increases. EEDD gets higher delivery success ratio since it will try alternate path when transmission fails.

6 Conclusion

In this paper, we propose EEDD, an energy-efficient data-dissemination protocol to address both the target/inquirer mobility problem and the energy conservation problem. EEDD extends the lifetime of the sensor network by controlling the node activity at two levels. Furthermore, we propose an adaptive scheduling scheme and an efficient data dissemination mechanism to reduce the sensing delay and packet forwarding delay. Our grid-based data dissemination scheme also reduces the query flooding cost and routing overhead, which further improves the energy efficiency.

The simulation results show that EEDD saves more energy as node density increases. Up to twelve times energy can be saved when the average node distance is half that of the detection range compared to the protocol without periodic wake-up and go-sleep. At the same time, EEDD can maintain lower average delay due to fewer collisions with reduced number of working nodes as a result of the EEDD scheduling mechanism. The results also show that EEDD can efficiently handle target/inquirer mobility and node failure. As compared to TTDD, EEDD can handle target mobility and is more resilient to node failure. EEDD can reduce average energy consumption significantly, though EEDD may have higher event detection delay for the first few packets at the first hop.

References

[1] D. Coffin, D. V. Hook, S. McGarry, and S. Kolek. Declarative ad-hoc sensor networking. *SPIE Integrated Command Environments*, July 2000.

[2] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy efficient communication protocol for wireless microsensor networks. In *Proc. IEEE Hawaii Int. Conf. on System Sciences*, pages 4–7, January 2000.

[3] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX)*, November 2000.

[4] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *Networking, IEEE/ACM Transactions on*, 11(1):2–16, February 2003.

[5] NS2 network simulator. *http://www.isi.edu/nsnam/ns/*.

[6] G. Pei and C. Chien. Low power TDMA in large wireless sensor networks. *Military Communications Conference*, 1:28–31, October 2001.

[7] G. Pottie and W. Kaiser. Wireless integrated network sensors. In *Communications of the ACM*, May 2000.

[8] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh. Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks. In *Proceedings of the IEEE INFOCOM*, pages 200–209, New York, NY, June 2002.

[9] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc networks. In *Proceedings of the Seventh ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, July 2001.

[10] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *Proceedings of the Eighth International ACM Conference on Mobile Computing and Networking (MOBICOM)*, Atlanta, GA, 2002.

[11] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang. PEAS: A robust energy conserving protocol for long-lived sensor networks. In *Proceedings of 23rd International Conference on Distributed Computing Systems*, pages 28–37, May 2003.

[12] F. Ye, G. Zhong, S. Lu, and L. Zhang. A robust data delivery protocol for large scale sensor networks. In *Information Processing in Sensor Networks (IPSN '03)*, California, April 2003.

[13] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *21st Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, volume 3, pages 23–27, June 2002.