Noname manuscript No.
(will be inserted by the editor)

Adaptive Energy Efficient Sensor Scheduling for Wireless Sensor Networks

Yinying Yang · Mihaela Cardei

Received: date / Accepted: date

Abstract In this paper, we consider an adaptive energy efficient sensor scheduling mechanism. We consider a wireless sensor network where the sink sends queries form time to time, and sensors are equipped with one or more sensing components. Our goal is to design an adaptive sensor scheduling mechanism to choose sets of active sensors to work alternatively such that different types of queries are served, the global connectivity requirements can be met, and network lifetime is maximized. A connected dominating set (CDS) based localized mechanism is proposed. Initially, a basic backbone is constructed, then when a query is issued, new sensors are activated locally such that to meet the requirements of the query and global connectivity. When a query expires, some sensors return to sleep and the CDS is restored. Our simulation results show that the solution is effective and it improved network lifetime.

Keywords Wireless sensor networks · composite event detection · coverage · connectivity · connected dominating set

1 Introduction and related works

A Wireless Sensor Network (WSN) [1] can detect single (or *atomic*) events or *composite* events [8]. Taking the sensor productions of Crossbow Technology Inc. as an example, a sensor equipped with MTS400 multi sensor board [10] can sense temperature, humidity, barometric pressure, and ambient light.

Let us consider a single sensing component, for example, the temperature. If the sensed temperature value exceeds a predefined threshold, we say that an *atomic* event occurred. A *composite* event is a combination of several *atomic* events. For example,

This work was supported in part by the NSF grants CCF 0545488.

Yinying Yang and Mihaela Cardei
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431
Tel.: 5619278583
Fax: 5612972800
E-mail: {yyang4@, mihaela@cse.}fau.edu

the composite event fire may be defined as the combination of the temperature and light. The *composite* event fire occurs only when both the temperature and the light exceed some predefined thresholds.

In this paper, we consider a publish-subscribe scenario as discussed in [4]. The sink sends queries as interests towards sensors in the monitored area. When sensors receive a query, if they can satisfy the interests and serve the query, then they are activated to perform the sensing task. The intermediate sensors help route sensed data toward the sink. The query includes {*type*, *interval*, *rect*, *timestamp*, *expire*} fields, where *type* shows what kind of sensing components are required, *interval* the frequency the sensor reports, *rect* indicates what rectangle area of interest to users, *timestamp* shows when the query is sent and *expire* shows when the query is expired.

Sensors are battery powered and in general, it is hard to recharge them. Energy management is an important issue in WSNs. We focus on a sensor scheduling mechanism that serves different queries. To prolong network lifetime, some sensors can go to sleep. A connected dominating set (CDS) based localized mechanism is proposed. Initially, a basic backbone is constructed so that a query can be propagated along the backbone into the whole network. When a new query is issued, sensors adaptively awake sleeping sensors in the area of interest such that to meet the requirements of the query. When a query expires, some sensors return to sleep, while the CDS backbone remains active.

In [8], Vu *et al.* propose an algorithm to construct a set of tree-structured detection sets to achieve energy efficient and reliable surveillance. To achieve reliable surveillance, each atomic event part of the composite event must be watched by at least k sensors. Their algorithm works in a greedy manner. At each step, the sensor node with the greatest contribution is added into the tree. The algorithm is repeated to find as many detection sets as possible, based on the sensors' energy constraint. Different detection sets work alternatively to achieve energy efficiency and to maximize network lifetime.

[9] proposes a localized solution for building a CDS in ad hoc wireless networks. The basic idea is a mark and prune process where each node decides locally, based on its neighborhood information, if it can go to sleep without breaking the overall connectivity requirement. Using a CDS based backbone provides an efficient mechanism to achieve global connectivity using a localized mechanism. In our paper, the objective is to use the CDS based backbone for global connectivity, while being able to satisfy different queries' sensing requirements.

Our work combines the main features of [4,8,9] to obtain a energy efficient data gathering mechanism that uses sensor scheduling to increase network lifetime. Different than [4], we use a sensor scheduling mechanism and apply the attribute-based data gathering mechanism on top of the active sensor backbone.

2 Network model and problem definition

In this paper, we consider that sensors can have single or multiple sensing components. Taking MTS400 multi sensor board of Crossbow Technology Inc. [10] as an example, it can sense temperature, humidity, barometric pressure, and ambient light. When we consider a single sensing component, for example the temperature, if it rises above some predefined threshold then an *atomic event* is detected. A *composite event* is a combination of several *atomic events*. For example, consider a fire-detection application. There can be two atomic events temperature $> th_1$ and smoke $> th_2$, where "th"

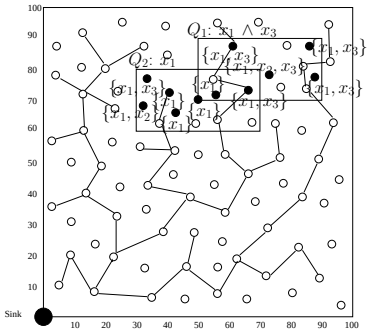

Fig. 1 Sensors deployed in a square area.

denotes the threshold for the corresponding attribute. A composite event fire might be defined as $fire = (temperature > th_1) \wedge (smoke > th_2)$. It is more accurate to report the fire when both atomic events occur, instead of the case when only one attribute is above the threshold.

Let us consider that M atomic events $x_1, x_2, ..., x_M$ form a composite event. Fig. 1 shows an example. For example, x_1, x_2, and x_3 are temperature, light, and smoke respectively. For a sensor which has only the temperature and light sensing components, we use the set $\{x_1, x_2\}$ to denote its sensing ability. We assume a sensor can be equipped with at most one sensing component of each type. All of a sensor's sensing components turn on or off simultaneously.

Sensor nodes may be equipped with different numbers and types of sensing components due to the following reasons [7]: they might be manufactured with different sensing capabilities, a sensor node might be unable to use some of its sensing components due to the lack of memory for storing data, or some sensor components might fail over time.

We consider a publish-subscribe scenario as discussed in [4]. The sink sends a query as interest towards sensors in the monitored area. When sensors receive a query, if they can satisfy the interests and meet the query, then they activate to perform the sensing task. The intermediate sensors help route sensed data towards the sink and the results are finally reported to the sink. For example, assume that we are interested in whether a fire will happen in the next 2 hours, using the interest query:

$type = temperature \wedge smoke$
$interval = 1min$
$rect = [(50, 70), (90, 90)]$
$timestamp = 01:30:00$
$expiresAt = 03:30:00$

The sensors that can sense the temperature and/or smoke and are within the rectangle with the lower left end (50, 70) and the upper right end (90, 90) are activated and report data every 1 minute from time 01: 30: 00 to 03: 30: 00. In general, the *type* field in the interest query is the combination of one or more sensing components, i.e., $x_1 \wedge x_2 \wedge ... \wedge x_l$.

More than one query can be sent to the monitored area. For example, another query can be issued as follows:

$type = temperature$
$interval = 2min$
$rect = [(30, 60), (70, 80)]$
$timestamp = 03 : 00 : 00$
$expiresAt = 04 : 00 : 00$

The areas of interest for the two queries intersect, thus some sensors might report data for both queries. The active sensors have to satisfy both the coverage and the global connectivity requirements.

An important issue in WSNs is energy management. Sensor nodes are battery powered and in general, it is hard to recharge them. It takes a limited time before they deplete their energy and become nonfunctional. One of the major components that consume energy is the radio, which can be in one of the following modes: transmit, receive, idle, and sleep. A radio is in the idle mode when the host is not transmitting or receiving data, and usually the power consumption is as high as in the receive mode. A radio is in the sleep mode when both the transmitter and the receiver are turned off, which is the most energy efficient state.

The objective of this paper is to design a sensor scheduling algorithm that allows sensors not actively participating in sensing or data relaying to go to sleep in order to conserve energy and to prolong the network lifetime. The sensor scheduling mechanism adaptively decides the set of active sensor nodes such that both the coverage and the connectivity requirements are met.

The *coverage requirement* requires that as queries are propagated to the monitored area, related sensors in the area of interest are activated to perform the sensing tasks. This is an adaptive mechanism: as new query requests arrive, new sensors might be activated, and as queries expire, sensors in the area of interest might go to sleep.

The *connectivity requirement* requires that the set of active sensors to be connected all the time. This condition is necessary for the communication between sensors and the sink in operations such as data reporting, query propagation, and forwarding of control messages.

In this paper we address the following problem: given a WSN where sensors are equipped with one or more sensing components (attributes) from the set $\{x_1, x_2, \ldots, x_M\}$, design an adaptive sensor scheduling mechanism such that the set of active sensors change over time such that to satisfy the *coverage* and *connectivity* requirements, while WSN lifetime is maximized.

3 Adaptive Sensor Scheduling in WSNs

We propose to use a localized Connected Dominating Set (CDS)-based solution, called Adaptive Sensor Scheduling in WSNs (ASW). A dominating set is a subset of sensors with the property that every sensor is either in the subset or has a neighbor in the subset. A connected dominating set requires that the sensors in the dominating set be connected. In our solution, at each time the active nodes form a CDS such that to satisfy the current *coverage requirements*. We design an adaptive CDS that change over time as new requests arrive or expire.

At the beginning, a CDS algorithm is run among all the sensors in the monitored area to choose a starting backbone in the network. CDS nodes participate in forwarding data or any other control messages. When a new request regarding sensing a particular

rectangular area is received/expires, sensors adaptively update the active nodes in the required area to satisfy both coverage requirement and global connectivity. The main operations are described below.

3.1 CDS backbone selection

At the beginning, an initial CDS backbone is selected so that the requests can be propagated to the monitored area along the backbone. The sensors that are not included in the backbone go to sleep to save energy.

Each sensor u has associated the following fields: time $t(u)$, a priority $p(u)$, $status(u)$, $CDS(u)$ and role label $r(u)$. The time data structure stores information regarding the amount of time that u has to be active. This is based on whether u is part of the CDS or not and whether u is currently serving any query. When the CDS role ends or a query expires, then the time that the sensor u has to remain active is updated accordingly. The sensors will also store additional information, such as the reporting interval for each query that it is serving.

Sensor priority $p(u)$ is defined as a 2-tuple $p(u) = (E(u), ID(u))$, where $E(u)$ is the node u's residual energy and $ID(u)$ is the node u's identifier. A node with higher residual energy has a higher priority. If two nodes have the same residual energy, then the nodes' IDs are used to break the tie. We assume that the CDS is updated periodically after each time interval T. Before re-running the CDS, sensors update their priority based on the remaining energy. Then for a time T, priority stays unchanged.

The *status* field can be active or sleeping. For active, $t(u)$ can be used to determine how long the sensor has to stay active. The field $CDS(u)$ can be TRUE or FALSE, depending on whether node u is currently in the backbone CDS or not. The field r keeps information about the role of a sensor: if the node is in the CDS and the set of queries that it is currently serving.

Sensors collect h-hop neighborhood information by exchanging $Hello$ messages, where h is a tunable parameter. Each sensor relies only on local information from its h-hop neighborhood to decide whether it will be a CDS node during the next period T. All other nodes which are not in the CDS and are not currently serving any query go to sleep. We assume that an active node can awake its 1-hop sleeping neighbors.

Node u's h-hop neighborhood is defined as $N_h(u) = \{v | dist(u, v) \leq h \ hops\}$, where $dist(u, v)$ is the distance between sensors u and v. The set $N'_h(u)$ is the set of sensors within the h-hop neighborhood which have higher priority than u, defined as $N'_h(u) = \{v \in N_h(u) | p(v) > p(u)\}$.

Every sensor u decides whether it will be a CDS node or not during the next period T as follows. Initially the node u sets $CDS(u) = TRUE$. Then sensor u sets $CDS(u) = FALSE$ if the following condition holds:

CDS-LocalRule: For any two of u's neighbors in $N_1(u)$, w and v, w and v are connected by a path with all intermediate nodes in $N'_h(u)$.

The intuition behind the rule is that if sensor u's neighbors can be connected without the help of u, then u does not have to be in the CDS. When running the *CDS-LocalRule*, sensor priorities $p(u) = (E(u), ID(u))$ are totally ordered, thus the CDS formation is guaranteed. Sensors with a higher priority have higher chances to be active. The *CDS-LocalRule* is a localized algorithm which was first proposed in [9].

3.2 CDS update mechanisms for serving queries

Let us consider that a query is issued for an area of interest, which we assume for simplicity as being a rectangle. A query Q is sent by the base station and has the format *Query(type, interval, area R, Tstart, Tend)*. The field *type* contains the attributes of interest that have to be reported, for example $type = x_1 \wedge x_3$. The field *interval* specifies how often data from the area of interest have to be reported. Since all the time the active nodes form a connected backbone, data will be forwarded to the sink along the active nodes. *Area* represents the query's area of interest, specified using rectangular coordinates. The attributes of interest have to be monitored between $Tstart$ and $Tend$. The sink sends the query Q towards the area R using controlled flooding or geographical flooding.

CDS nodes awake the 1-hop sleeping nodes in area R and forward the query information. One way to accomplish this step is if each node keeps a list with the location of its 1-hop neighbors. Then, upon receiving the *Query* message, the active nodes in the CDS awake their sleeping nodes in area R. An awaken node checks if it has at least one of the sensing components from the query type field, then it remains active for the duration of the query and will report data according to the reporting interval in the query. The sensor updates its fields: time, status, and role.

When a query ends, the reporting sensors check their role field. If a sensor is part of the CDS or is serving another query, then it remains active. Otherwise it returns to sleep. In addition, the node updates its fields time, status, and role to remove the expired query.

One case that can occur regards overlapping queries. A sensor may serve multiple queries if the reporting areas intersect. The frequency of reporting is done according to the requesting queries and this field will update as queries expire or new queries arrive.

After a period of time, some sensors in the CDS may lower their energy or even run out of energy. To better balance the energy consumption, the CDS is updated every time interval T. All the sleeping nodes in the monitoring area awake and update their priority based on the remaining energy, $p(u) = (E(u), ID(u))$ for a node u. Then they update their h-hop neighborhood by exchanging *Hello* messages. The *CDS-LocalRule* is executed to decide the new CDS for the next period T. All the nodes participate in the new CDS selection and update their fields accordingly depending on whether they will be part of the CDS or not. The nodes serving queries will continue to be active and to perform their sensing tasks. Nodes which are not in the CDS or serving queries go to sleep.

Implementing the scheduling mechanism using a CDS has several advantages. It provides an energy-efficient mechanism which constructs a connected backbone on which queries, data messages, and control messages can be exchanged between sensor nodes and the sink. The CDS has the property that each sleeping node is one hop away from a node in the CDS. Thus, when queries arrive for an area of interest, the CDS nodes can easily awake their sleeping neighbors. Rotating the nodes in the CDS is also important in balancing the energy consumption. Using a localized mechanism to decide the nodes in the CDS is an important property since it scales well with a large number of sensor nodes.

4 Simulations

In this section, we present the simulation results. We study the average number of active sensors, the overhead, the average dissipated energy, the delay, and the remaining energy level of sensors. We evaluate the performance of the ASW algorithm and compare it with the directed diffusion (DD) [4].

4.1 Simulation environment

Sensors are randomly deployed in a square area of 300 × 300 area units. The transmission range of sensors is 35 units. The queries' inter-arrival time uses exponential distribution. The query's duration is a random number between 1 and 2.5 hours. The queries' interest area is 150 × 150 area units and the locations of the queries are randomly chosen. A total of 20 queries are served by the network and after 10 queries the ASW updates the CDS-based backbone.

In the simulations, we compute the transmission and receiving energy consumption similar to LEACH [2]. The energy consumption for transmitting and receiving a message is $E_{Tx} = 50 \times 10^{-9} \times msg\_length + 100 \times 10^{-12} \times msg\_length \times r^2$ J and $E_{Rx} = 50 \times 10^{-9} \times msg\_length$ J, where r is the transmission range and msg_length is the length of the message. The initial energy of each sensor is 200J. Similar to [6], we assume that the packet size of a data message is 64 bytes and the size of a control message is 16 bytes. $Hello$ messages used to form the CDS backbone and messages used to propagate interests into the monitoring area are control messages.

The energy consumed to transmit and to receive a data message is 8.832×10^{-5}J and 2.56×10^{-5}J respectively. The energy consumed to transmit and to receive a control message is 2.208×10^{-5}J and 0.64×10^{-5}J respectively. We take the data rate of 250 kbps, similar to ZigBee [11]. Since the packet size is 64 bytes for a data message and 16 bytes for a control message, the duration to transmit (or receive) a data packet is $\frac{64*8}{250*10^3} = 2.048$ms and the duration to transmit (or receive) a control packet is 0.512ms. Any time a sensor is not transmitting or receiving, it is in the idle state. A sensor in idle state consumes 2mW [6]. We consider that the energy consumed for sensing an event is 5×10^{-5}J. This is for example the sensing energy used by an acceleration sensor to take a sample of $10ms$ [3].

Active sensors form a data delivery tree in order to deliver the sensed data to the sink which is located on the left bottom corner of the deployment area. The tree is formed using controlled flooding initiated by the sink . The sink broadcasts a message containing the number of hops. When an active sensor receives the message, it records the shortest path (the minimum number of hops) to the sink, keeps a reference to the parent from which the message was received, increases the number of hops by one, and forwards the message. A message is forwarded only if it has a shorter path to the sink.

We study the following metrics in the simulation:

- *The average number of active sensors* shows the average number of active sensors over time. It is computed as $\frac{|ActiveSet_1| \times t_1 + |ActiveSet_2| \times t_2 + ... |ActiveSet_n| \times t_n}{t}$, where $|ActiveSet_i|$ is the number of sensors active during the time t_i and t is the total duration from the time the first query begins to the time the last query ends.
- *Overhead* shows how many control messages are sent and received, which also implies the energy consumption in transmitting and receiving control messages.

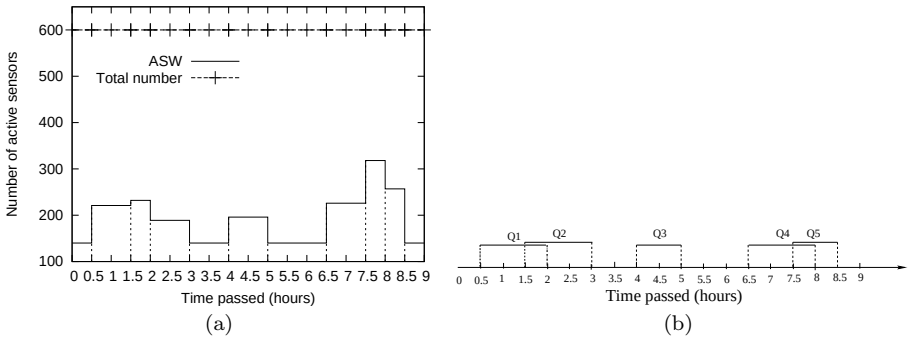

Fig. 2 (a) An example. (b) Queries' timing.

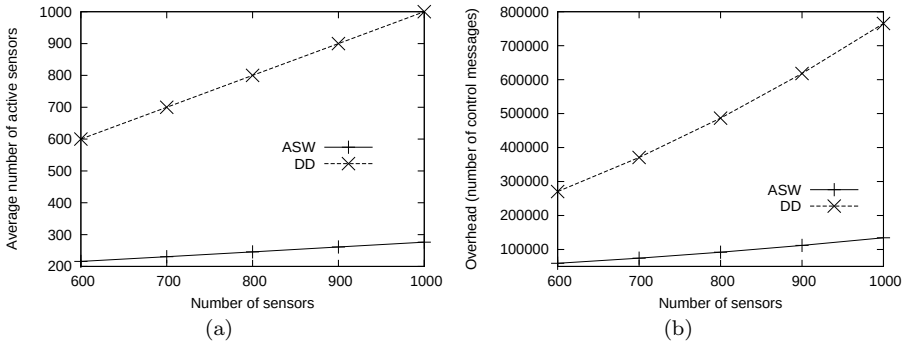

Fig. 3 (a) The average number of active sensors. (b) Overhead.

- *Average dissipated energy* measures the ratio of total energy consumption per sensor in the network to the number of reports received by the sink. It is computed as $\frac{eConsume_1+eConsume_2+...+eConsume_n}{n\times NumReport}$, where $eConsume_i$ is the energy consumption of sensor i, n is the total number of sensors in the network, and $NumReport$ is the total number of reports received by the sink.
- *Delay* measures the number of hops from the source sensor to the sink along the path in the data delivery tree, which implies the one-way latency observed between transmitting a report and receiving it at the sink.
- *The number of sensors in different remaining energy level* measures the remaining energy for each sensor in the network as time passes. Energy level 1 means that the remaining energy is less than or equal to 100J. Energy level 2 means that the remaining energy is greater than 100J and less than or equal to 200J.

We conduct the simulation on a custom discrete event simulator. All the tests are repeated 50 times. The collected data is averaged and reported in the following figures.

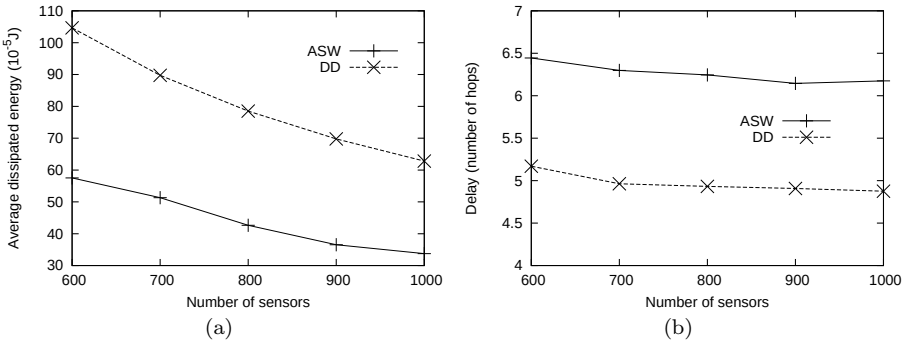

Fig. 4 (a) Average dissipated energy. (b) Delay.

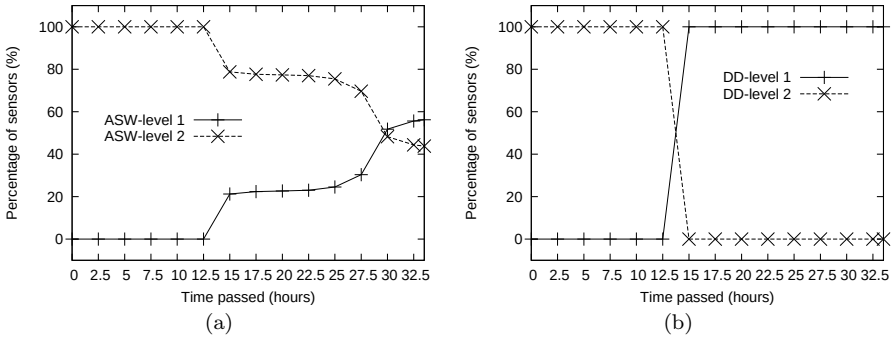

Fig. 5 (a) Energy level for the ASW. (b) Energy level for the directed diffusion.

4.2 Simulation results

Fig. 2 shows an example of running the ASW algorithm. Totally 600 sensors are randomly deployed in the monitored area to serve 5 queries. Fig. 2b shows the timing and duration of each query, for example, query 1 starts at time 0.5 and ends at time 2. Fig. 2a shows the number of active sensors, for example, at the very beginning from time 0 to time 0.5 there is no query in the network and only backbone sensors are active. At time 0.5, more sensors are active since query 1 starts and more sensors are activated for sensing purpose. Fig. 2a shows that using ASW only a small number of sensors has to be active for sensing and delivering reports to the sink.

Fig. 3 compares the average number of active sensors and overhead between ASW and directed diffusion. Fig. 3a shows that our algorithm has fewer number of active sensors. Fig. 3b shows that directed diffusion has more overhead. That is because every time when a new query comes, directed diffusion propagates the interest to every sensor in the network, while in ASW the query is only propagated over the active sensors including backbone nodes and a few nodes active for sensing purpose. The sleeping sensors will not receive and forward the control messages. Therefore, the overall overhead is reduced.

Fig. 4 compares the average dissipated energy and delay between ASW and direct diffusion. Fig. 4a shows that ASW consumes less energy for delivering a data message compared with directed diffusion. This is because in directed diffusion, more energy is consumed in transmitting and receiving control messages and more sensors are in the idle state, which consume considerable energy. On the other hand, in ASW less energy is wasted on control messages. Most sensors are put to sleep and and as a result they consume much less energy compared to the idle state. Fig. 4a shows that ASW is more energy efficient. Fig. 4b shows that ASW has a comparable but longer deliver path. This is because in the directed diffusion all sensors are active, while in ASW only part of the sensors are active, and as a result the directed diffusion may form a shorter data delivery path.

Fig. 5 compares sensors' remaining energy level. 600 sensors are deployed in the monitored area. The lines in the figures show the percentage of sensors with remaining energy in levels 1 or 2. With the time passing, the number of sensors with higher remaining energy (energy level 2) decreases, while more sensors have lower energy (energy level 1). Compared to ASW, directed diffusion consumes energy more quickly. At time 32.5 ASW still has sensors in energy level 2 while all sensors in the directed diffusion are in the level 1.

To summarize, compared with the directed diffusion, ASW is more energy efficient, it generates less overhead, however, it may have a longer delivery delay.

5 Conclusions

In this paper, we propose an adaptive energy efficient sensor scheduling mechanism to choose sets of active sensors to work alternatively such that different types of queries are served and global connectivity requirements are met. A connected dominating set (CDS) based localized mechanism is proposed. Our simulation results show that the solution is effective and energy efficient.

References

1. I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, Wireless sensor networks: a survey, *Computer Networks*, 38(4), pp. 393-422, 2002.
2. W. Heinzelman, A. Chandrakasan, and H.Balakrishnan, Energy-efficient communication protocols for wireless microsensor networks, *HICSS*, 2000.
3. J. L. Hill, System Architecture for Wireless Sensor Networks, Ph.D. Thesis, 2003.
4. C. Intanagonwiwat, R. Govindan, D. Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, *MOBICOM*, 56-67, 2000.
5. C. Intanagonwiwat, R. Govindan, D. Estrin, J. S. Heidemann, F. Silva, Directed diffusion for wireless sensor networking, *IEEE/ACM Trans. Netw*, 11(1): 2-16, 2003.
6. W. Liang, H. Yu, P. Zeng, and C. Che, BESM: A balancing energy-aware sensor management protocol for wireless sensor network, *International Journal of Information Technology*, Vol. 12 No.4 2006.
7. A. V. U. P. Kumar, A. M. Reddy V., and D. Janakiram, Distributed collaboration for event detection in wireless sensor networks, *MPAC'05*, 2005.
8. C. T. Vu, R. A. Beyah and Y. Li, Composite event detection in wireless sensor networks, 26^{th} *IEEE IPCCC*, 2007.
9. J. Wu and H. Li, An extended localized algorithm for connected dominating set formation in ad hoc wireless networks, *IEEE Transactions on Parallel and Distributed Systems*, Oct. 2004, pp. 908-920.
10. http://www.xbow.com/Home/HomePage.aspx
11. http://www.zigbee.org/