Wireless Networks 11, 619–635, 2005
© 2005 Springer Science + Business Media, Inc. Manufactured in The Netherlands.

A Performance Evaluation of a Novel Energy-Aware Data-Centric Routing Algorithm in Wireless Sensor Networks*

AZZEDINE BOUKERCHE†
SITE, University of Ottawa, Canada

XUZHEN CHENG
The George Washington University, USA

JOSEPH LINUS
University of North Texas, USA

Abstract. In this paper, we present a novel Energy-Aware Data-Centric Routing algorithm for wireless sensor networks, which we refer to as EAD. We discuss the algorithm and its implementation, and report on the performance results of several workloads using the network simulator ns-2. EAD represents an efficient energy-aware distributed protocol to build a rooted broadcast tree with many leaves, and facilitate the data-centric routing in wireless micro sensor networks. The idea is to turn off the radios of all leaf nodes and let the non-leaf nodes be in charge of data aggregation and relaying tasks. The main contribution of this protocol is the introduction of a novel approach based on a low cost backbone provisioning within a wireless sensor network in order to turn off the non backbone nodes and save energy without compromising the connectivity of the network, and thereby extending the network lifetime. EAD makes no assumption on the network topology, and it is based on a residual power. We present an extensive simulation experiments to evaluate the performance of our EAD forwarding-to-parent routing scheme over a tree created by a single EAD execution, and compare it with the routing scheme over a regular Ad hoc On-Demand Distance Vector (AODV) Protocol. Last but not least, we evaluate the performance of our proposed EAD algorithm and compare it to the Low-Energy Adaptive Clustering Hierarchy (LEACH) protocol, a cluster-based, energy-aware routing protocol specifically designed for sensor networks. Our results indicate clearly that EAD outperforms AODV and LEACH in energy conservation, throughput, and network lifetime extension.

Keywords: wireless sensor network, spanning tree with maximum leaves, data-centric routing, in-network processing

1. Introduction

Recent advances in wireless communication, micro sensor technology, on-board processing, radio and actuators have been a driving force to expand our ability to remotely monitor and interact with the physical world. Wireless sensor networks provide a global view of the monitored area based on local observations measured by each sensor. These sensor networks [16] usually contains thousands or millions of sensors, which are randomly and densely deployed (10 to 20 sensors per m^2). A network with tens of neighbors per sensor have been studied in [5,6,16]. Sensor networks have short transmission range (up to 10 meters) and low data rate (several bytes). Each sensor has a light weight and a low cost (projected to be $< US\$1$ by the year 2004), and they may not have a globally unique Id. Sensors are powered by battery, which is impossible to get recharged after deployment. Note that sensor networks are designed to have long operation time (i.e., several years). However, sensors do not have unlimited resource supplies, e.g., power, CPU, memory, etc. Consequently, they are prone to failure. Thereby making routing schemes based on unique addresses that are originated and applied in IP networks a challenging problem for the design of reliable wireless sensor networks.

There are two kinds of dominant message traffic in a sensor network: queries from the user to the network and the actual data collected from the sensor node back to the user. Each sensor acts as a power aware sensor to sense the environment and as a router to relay traffic to other sensors. In this study, the sensor nodes are assumed to be using μAMPS sensors. The radio characteristics and power utilization of these sensors were simulated in experiments using the μAMPS extensions [10] to the Network Simulator ns-2 [19,24]. The amount of data generated by one sensor can be large enough to block the whole network, and a large part of these data may be useless to the end user. Thus, data is pre-processed before they are transmitted. This is referred to as *in-network processing* [13,15] during which redundant, useless and spurious data are deleted, and partial observations from different sensors are combined and aggregated. In-network data processing is a *must* to decrease the large volume of raw observations per sensor and to reduce

*Dr. A. Boukerche was partially supported by NSERC, Canada Research Program, Canada Foundation for Innovation, and Ontario Innovation Funds/Ontario Distinguished Research Award.
†Corresponding author.
E-mail: boukerch@site.uottawa.ca

the number of broadcasts. This can conserve the sensors' battery power, which to a great degree, determines the network lifetime.

Within a sensor, the dominant energy consumer is the radio transceiver [7]. For a sensor network with a short transmission range, the radio consumes almost the same amount of energy in transmit, receive and idle mode [1]. Therefore, the only way to save energy is to completely turn off the radio. However, a sleeping sensor can't function as a relay even though it can continue sensing and it can wake up when some events are detected. Thus, we can't turn off all sensors at the same time in a sensor network. Some sensors, indeed, must be active for traffic relaying purpose.

In this paper, we propose to construct a virtual backbone, which contains all active sensors, to assist the energy-aware routing scheme. All sensors not in the virtual backbone turn off their radios in order to conserve their power supply. Backbone sensors are in charge of in-network data processing and traffic relaying. This virtual backbone can be easily reconfigured when its topology changes.

Our work is motivated by SPAN [4], GAF [25] and LEACH [10]. SPAN and GAF are elaborated in the context of MANET (multihop ad hoc wireless networks), where traffic-flow may originate from any node to any other nodes. SPAN, a topology-based protocol, assumes that each node knows its 2-hop neighbors. GAF, a location-based protocol, requires that each node to be aware of its own position and the grid where it resides. LEACH, a clustering based hierarchical protocol used to save battery life, features the concept of rounds to counter the cluster-head's fatigue. Both SPAN and GAF models propose algorithms to compute a subset of nodes whose radios can be turned off with little influence on data dissemination while network lifetime can be greatly extended. Even though both SPAN and GAF are promising methods, they may not be suitable for dense micro sensor networks, due to their assumptions and the unique features of sensor networks presented above. A sensor with tens neighbors may not afford to store its 2-hop neighborhoods' information as is done in the SPAN model, and it may not have information such as the position of the nodes as in the GAF model.

In this paper, we propose an algorithm to compute a broadcast tree rooted at the gateway, and introduce a novel approach based on a low cost backbone provisioning within a sensor network in order to turn off the remaining nodes and save energy without compromising the connectivity. This broadcast tree spans all sensors and it has large number of leaves. All the leaf sensors turn off their radios to save power while all active sensors stay alert for traffic relaying. We map our problem to the construction of the *spanning tree with maximum number of leaves*, which is known as a NP-hard problem, since it is equivalent to minimum connected dominating set [8]. The reduced topology by all non-leaf nodes forms the virtual backbone.

We propose a novel Energy-Aware Data-centric routing heuristic, which we refer to as EAD, that exhibits a low message overhead, and computes a broadcast tree approximating the optimal spanning tree with a maximum number of leaves. The novel concepts involved in EAD include the *neighboring broadcast scheduling* and the *distributed competition among neighbors*, based on residual energy. These two characteristics ensure that the resultant tree has many leaves, and sensors with a higher residual power have higher chance to be non-leaf nodes. EAD follows this energy-aware paradigm and results in a special rooted broadcast tree, which is designed intentionally for data-centric routing. Each sensor needs to broadcast messages at most twice, which will result in a large message overhead in a large-scale sensor network containing thousands or millions of sensors. To address this problem, we propose to let a subset of sensors turn off their radios before the execution of EAD. We present a heuristic, which we refer to as a topology-based scheme, in order to determine which sensors need to sleep ahead of time.

The remainder of the paper is organized as follows. In Section 2, we present the network model. Section 3 outlines the basic idea of our data-centric routing scheme. In Section 4, we present our Energy-Aware Data-centric routing heuristic in full details. Section 5 reports on the simulation experiments we have carried out to evaluate the performance of EAD. Section 6 concludes the paper.

2. Network model

In this paper, we consider wireless micro sensor networks for monitoring abnormal events. Example applications include the *habitat monitoring* [13,15], the *contamination transport monitoring* [7], and the *forest fire pre-warning* [26], just to mention a few. We assume that the network contains hundreds or thousands of smart sensors deployed randomly in the target area. There exists one gateway that connects the micro sensor network to the outside distributed system such as the Internet. The gateway is located at the boundary of the monitored area, where it is reachable by at least some sensors. We refer to each micro sensor as a *data source* or an *event source* since data in a sensor network are generated by sensors, and the gateway as a *data sink* or an *event sink*.

The architecture of a micro sensor [16] contains four components: sensing circuitry, digital processing, power supply, and radio transceiver. Among these four components, radio transceiver is the dominant power consumer [1,6,16,21,23]. The energy spent for sensing and data processing is negligible. For example, the power consumed by a Berkeley mote [15] to transmit 1 bit data is equivalent to 800 instructions [11,12]. For sensors with short transmission range like mote, the energy consumed for different mode (transmit, receive and idle) are comparable [22], while a sleeping sensor (radio is off) consumes little energy. Thus, in order to save energy the sensor needs to completely turn off its radio. Figure 1 provides a good illustration on the energy consumption in a typical sensor.

3. Data-centric routing in micro sensor networks

In this section, we highlight the basic ideas of the data-centric routing in a micro sensor network. First, let us recall that the in-network processing can significantly improve the

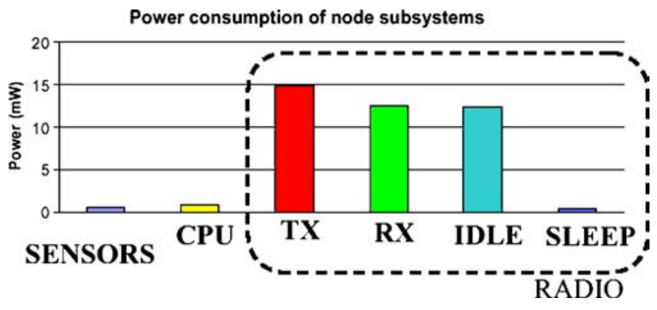

Figure 1. Energy consumption for a typical sensor reported in [7].

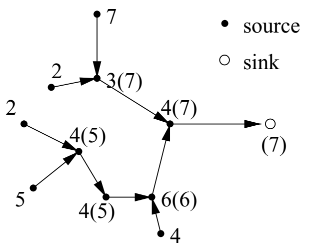

Figure 2. An example to demonstrate data-centric routing. Label $x(y)$ at each node means the local temperature measurement is x while the aggregated value so far is y. The aggregation function is max.

scalability and the lifetime of micro sensor networks. Thus, at each sensor, the local raw data is, first, combined with partially processed data delivered from sensors that are farther away from the sink. Then, the aggregated result is transmitted to the sensor that is closer to the sink or the sink itself for further processing. Intuitively, data is routed along a *reversed multicast tree* with the sink as the root. Data aggregation happens at each non-leaf node, which summarizes the output based on the aggregation functions (e.g., SUM, AVG, MEAN, MAX, etc.) used by the sensor nodes in the subtree rooted at itself and transmits the aggregated data to its parent. This process is referred to as a*data-centric routing* [9,13–15]. Figure 2 gives an example of data-centric routing where the highest temperature needs to be reported to the user.

Traditional network-wide routing is referred as *address-centric routing* [14]. A packet is routed based on the unique IP destination address and its data payload remains unchanged during the delivery from the source to its destination. This routing scheme does not work with micro sensor networks, mainly due to the lack of a globally unique address, and the sensor node's energy constraints. In a micro sensor network, the data are processed before their transmission. Redundant and useless data are discarded. Local data are aggregated to provide globally-effective result. It is possible that an information packet contains different values from hop to hop during the course of its transmission from a leaf to the intermediate sensors then to the sink in the tree, because each intermediate sensor may aggregate multiple packets. In this aspect, micro sensor network is similar to a pure *peer-to-peer* network.

The reversed multicast tree construction for data-centric routing is determined based upon the applications of the micro sensor networks. Data traffic can take several forms: *periodic, event-driven* and *query-based*. A sensor network may support all of these three kinds of data traffic. For periodic traffic, all of the sensors report their measurements back to the user once every time interval, which is fixed and could be preprogrammed before the sensors' deployment. This kind of networks require that all of the sensors to be synchronized (i.e., when to turn on their radio at the same time) such that the in-network processing can be done at each intermediate sensor in order to guarantee one broadcast per sensor and per time interval. For this kind of applications, all of the broadcast trees have the same effects with respect to radio transceiver energy consumption since each sensor broadcasts exactly once in a designated time interval. But latency and power consumption for data processing may be significantly different. In an event-driven model, no traffic flows within the network unless some special events are detected. These events must be reported to the user immediately after the detection. The multicast tree for data aggregation and dissemination is identical to a Steiner tree containing the sink and all of the sensors detecting the events, plus the relay sensors used to forward the data traffic to their destination. The number of relay sensors needs to be minimized to decrease the total power consumption.[1] In the query model, routes need to be computed for the query and data transmissions between sink and the queried sensor node. The problem related to this model is identical to that of the event-driven model, with the exception that the query model includes the query message which propagates from the sink to the source.

Among all of these applications, sensors remain in sleep mode most of the time in order to save their energy. If all of these sensors need to report their readings periodically, then they must turn on their radios at roughly the same time. In a large embedded sensor network, synchronization is do-able but quite expensive. If only part of the network is involved at a time, as in event-driven and query models, the sensors have no idea when an event will happen and when a query will be ever submitted through the network. Simply turning on all of the sensors is a waste of resources while turning off all of them put the network down or malfunctioning since a sleep sensor can not receive any message. An intuitive idea to overcome these problems is to activate a small subset of sensors at any instant of time such that they can collaborate and quickly respond to spontaneous events and queries. But one may raise the following question: how many sensors need to be on? Too few active sensors causes network partition and packet loss while too many causes unnecessary energy expenditure and higher interference. We propose to use *a spanning tree with maximum number of leaves* rooted at the sink as a *virtual backbone* to facilitate the data-centric routing task.

Each sensor is either a leaf or an inner node in the tree. All leaf nodes turn off their radios to save their energy. They periodically wake up to replace neighboring sensors with depleted power. Building a spanning tree with maximum number of leaves is equivalent to constructing a *minimum connected dominating set*, which is an NP-Complete problem [8].

[1] This problem is NP-hard.

Thus, there has been a continued interest in developing efficient heuristic solutions to solve this problem. In this paper, we propose an message-efficient distributed heuristic to build an energy-aware rooted spanning tree with many leaves which will be described later in detail.

Let us review briefly, some of the existing data dissemination schemes based upon the spanning tree rooted at the sink model that have appeared in literature. In [13], the authors proposed a *directed diffusion* scheme, where a sink broadcast an *interest*, and each intermediate sensor receiving the interest must broadcast it at least once to setup the reverse path to the sink. The target sensor (specified by the interest) sends back the data along several paths. The sink may reinforce the preferred path after the initial exploratory stage. Without the location information, the interest must be broadcasted globally. This consumes energy and wireless bandwidth. If all of the active sensors form a spanning tree rooted at the sink, then the dissemination of the interest can be restricted to the non-leaf tree node. If the queried sensor is sleeping, an active neighbor node can either activate it directly, or store the query until the targeted sensor wakes up. Another interesting attempt for data-centric routing in event-driven sensor networks is described in [14]. All of the sensors sensing the same event (within the event radius) first aggregate the data, then transmit the result to the sink. The computation of the transmission path problem can be mapped to the *network Steiner tree problem*, which is known to be an NP-hard problem. It is obvious that our virtual backbone can be used to relay the aggregated result to the sink. For applications with frequent occurrence of queries and events, our proposed approach is extremely helpful. Actually for a dense sensor network in which each sensor has a large number of neighbors, only a few number of sensors need to be active at any time. These sensors form a virtual backbone rooted at the sink, and they are ready for a query and/or an event dissemination. In this paper, we propose a heuristic protocol to build a rooted broadcast tree with many leaves, which we refer to as EAD, and will be described in the next section.

4. EAD: A heuristic algorithm to construct a rooted broadcast tree with many leaves

In this section, we discuss the details of EAD, the heuristic algorithm we use to construct a rooted broadcast tree with many leaves, as well as the broadcast tree maintenance scheme we are proposing to use within the EAD protocol.

4.1. EAD description

We assume that each sensor has its own radio transceiver on and is sensing the common channel when the network is initially deployed. We also assume that all of the sensors have the same transmission range. In other words, we only consider symmetric links. The control message contains four fields: *type, level, parent, power*. Let v be the sender of the message, and $type_v$ its status (0: undefined; 1: leaf node; 2: non-leaf node). $level_v$ refers to the number of hops from v to the sink; $parent_v$ is the next hop of v in the path to the sink; $power_v$ is the residual power E_v. If E_v is unavailable, we can use the difference between the expected lifetime of the battery and the total time with the radio transceiver already on. The basic outline of the heuristic is portrayed in figure 3.

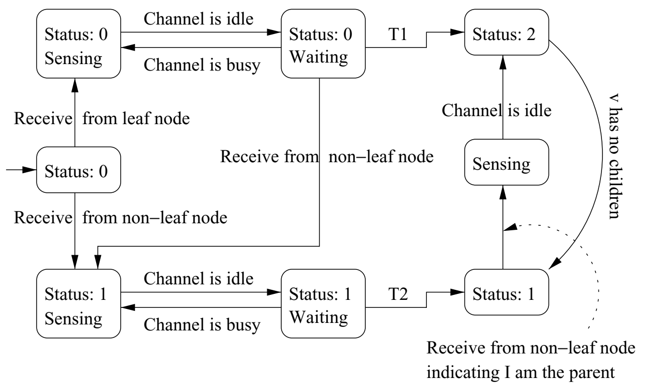

Figure 3. State diagram for the proposed heuristic run by a node v other than a sink.

Initially each sensor v has status 0. The sink first broadcasts $msg(2, 0, NULL, \infty)$, where ∞ indicates that the sender is the sink. When a node v receives $msg(2, level_u, parent_u, E_u)$ from node u, it becomes a leaf node, senses the channel until it is idle, then waits for T_2^v time. If the channel is still idle, v broadcasts $msg(1, level_u + 1, u, E_v)$. If v receives $msg(1, level_u, parent_u, E_u)$ from u, it senses the channel until it is idle, waits for time T_1^v. If the channel is still idle, v broadcasts $msg(2, level_u + 1, u, E_v)$. In other words, it becomes a non-leaf node. Note that a waiting sensor goes back to sensing (see figure 3) if the common channel is occupied by other sensors before it times out. If a node v with status 1 receives $msg(2, level_u, v, E_w)$ from w indicating that v is its parent, v broadcasts $msg(2, level_v, parent_v, E_v)$ immediately after the channel is idle (No waiting!). This process continues until every sensor is either a leaf node, or a non-leaf node. A sensor with status 2 will become a leaf node if it detects that it has no children.

Note that we use T_1^v and T_2^v to ensure that no two neighboring broadcasts are scheduled at the same time. T_1^v and T_2^v can be computed locally. Let N_v be the set of 1-hop neighbors of v. We require $T_1^v > max_{u \in N_v}\{T_2^u\}$ to ensure that a sensor becomes a non-leaf node in the tree only when necessary. We also require that T_1^v and T_2^v to be monotonically decreasing functions of E_v, the residual power of v. The basic idea is to force the neighboring sensors with a higher energy to broadcast earlier than those nodes with a lower residual power. For example, we can choose $T_1^v = 2 \cdot t_0 + \frac{c}{E_v}$ and $T_2^v = t_0 + \frac{c}{E_v}$, where t_0 is the upper bound of the propagation time between any pair of neighboring sensors, and $c > 0$ is an adjusting constant. Note that with properly selected functions for T_1^v and T_2^v, local broadcasting among neighboring sensors can be scheduled without conflict.

The main features of EAD include the *scheduling of local broadcasts* by T_1 and T_2, and the *distributed competition among neighboring nodes in order to become a non-leaf tree*

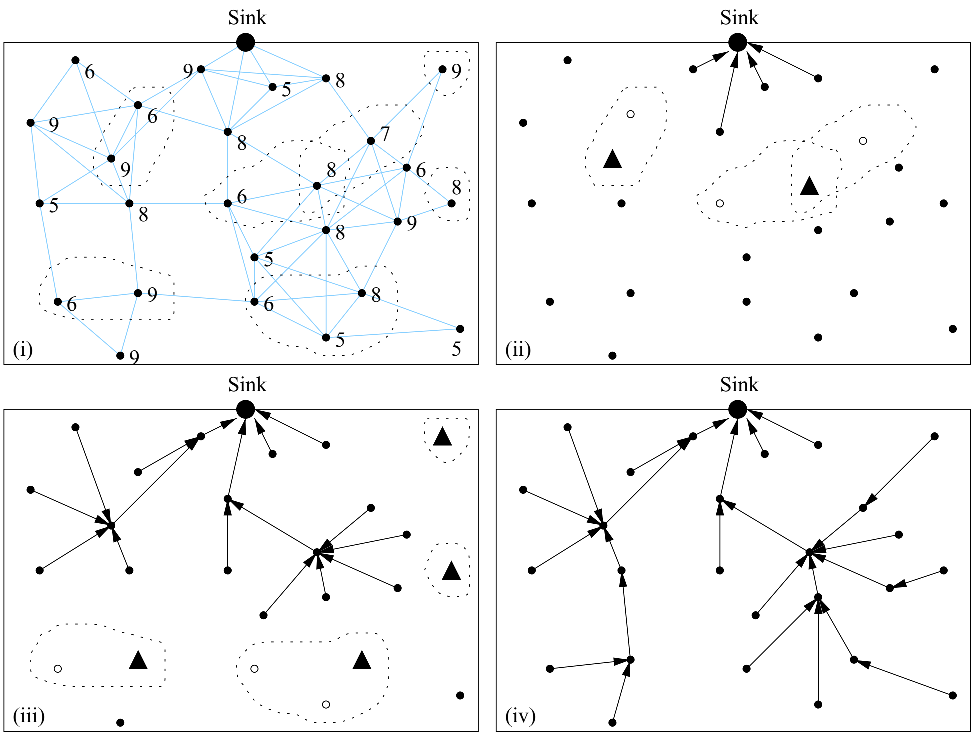

Figure 4. An illustrative example.

node by T_1. The intuition behind the algorithm is stated as follows: once a sensor u announces its status 2 (i.e., non-leaf node) through a broadcast, all of its 1-hop neighbors with status 0 become leaf nodes. They announce their status in the reverse order of their residual power, with higher energy node in the neighborhood broadcasts earlier (for example, $T_2^v = t_0 + \frac{c}{E_v}$). When the 2—hop neighbors of u with status 0 hear these broadcasting messages, they start to compete with each other. The winners are those with a highest residual energy among all of its neighboring competitors (thus, with a smallest T_1 among all of its neighboring competitors). Figure 4 gives an illustrative example. In figure 4(i) the original sensor network topology is given. Each sensor is labelled with its residual power. The islands indicate the competing neighboring groups in (ii) and (iii). In figure 4(ii), sink broadcasts to its 4 neighbors. The 2-hop neighbors form 3 neighboring groups. The sensors with highest energy in each group (replaced by triangles) win the local competition. In figure 4(iii), winners become non-leaf nodes. Each node specifies its own parent, the neighbor with the highest energy in the partial tree. Each designated parent becomes a non-leaf node, with its neighbors not in the tree joining the tree immediately after the parent announces its new status. Later, the neighbors of the winners (not in the tree) join the tree as children of its corresponding winner. The winners's 2hop neighbors (not in the tree) form four neighboring competing groups. Figure 4(iv) repeats (iii) to get the final broadcast tree with many leaves. Note that the two winners (triangles) in (iii) become leaf nodes in the final tree even though they were the winners, since they have no children.

Note that EAD grows a broadcast tree from the sink. Once the algorithm terminates, all the leaf nodes can turn off their radios to save energy. These nodes may switch to "power-on" periodically, or when some abnormal events are detected. Using this simple scheme, each node will broadcasts at most twice. The induced graph by all of the non-leaf nodes forms the virtual backbone. Due to the broadcasting nature in a wireless network setting, the virtual backbone may have a mesh structure. But each sensor records its parent leading to the sink. The sink can restrict the broadcast of queries to nodes within the virtual backbone and the sources can send back data to the sink along the backbone.

EAD makes use of the following efficient broadcasting scheme. The sink first broadcasts a message containing level 0. After receiving a message with level k the first time, v senses the channel until it is idle, waits for time T_2^v. If the channel is still idle, v broadcasts a message with level $k + 1$. If the channel is occupied by other sensors before v times out, v senses the channel again. This process continues until v's broadcast succeeds. Each node only broadcasts once.

In figure 5, we present the pseudo-code of the EAD protocol.

4.2. Maintaining the broadcast tree

Our strategy extensively explores *the dense connectivity* of sensor networks. The maintenance of the tree is done by a strategy similar to the one described in [10]. The maintenance of the tree becomes important for two reasons. First, the non-leaf nodes may die, thus, orphaning all of the child nodes

EAD Algorithm

Node variables:
*EAD_type_, EAD_previous_type_, EAD_level_, EAD_parent_, EAD_has_child_, EAD_is_sink_*

EAD control message variables:
ead_type, ead_level, ead_parent, ead_power

Along with its specific variables, EAD also uses system variables like unique node id: *nodeId*, and source address of a received packet *source_add*. Each node start with *EAD_type_* 0. *EAD_is_sink* is used to start the execution at startup. Pseudocode executed at a node after receiving an EAD packet is as follows:

```
function receiveEADPacket {
    EAD_previous_type = EAD_type

    if (EAD_type_ == 0) {
        if (ead_type == 2) {
            EAD_type_ = 1
            EAD_level_ = ead_level+1
            EAD_parent_ = source_addr
        } else if (ead_type == 1){
            EAD_type_ = 2
            EAD_level_ = ead_level+1
            EAD_parent_ = source_addr
        }
        finalEADStatusUpdate
        sendEadControlMessage
        return
    } else if (EAD_type_ == 1) {
        if (ead_parent == nodeId){
            EAD_type_ = 2
            finalEADStatusUpdate
            sendEadControlMessage
            return
        }
    } else if (EAD_type_ == 2) {
        if ((ead_type == 2) && (EAD_parent_ == source_addr)) {
            EAD_type_ = 1
            finalEADStatusUpdate
            sendEadControlMessage
            return
        }
    }

    if (ead_parent == nodeId) {
        EAD_has_child_ = true
        EAD_type_ = 2
    }
}
function finalEADStatusUpdate {
    if (ead_parent == nodeId) {
        EAD_has_child_ = true
    }

    if (EAD_has_child_ == true) {
        EAD_type_ = 2

    }else if ((EAD_previous_type_ != 0)&&(EAD_has_child_ == false)) {
        EAD_type_ = 1
    }
}
```

Figure 5. Pseudo-code of the EAD algorithm.

which are transmitting data to the non-leaf nodes. Secondly, the non-leaf nodes that form the backbone tree have to stay awake all the time. This approach induces a huge energy drain on them when compared to the leaf nodes that are awake only occasionally. This leads to the so-called fatigue of the non-leaf nodes. To ensure a fairly similar energy demand from all sensor nodes and maintain the workload balance among them, the algorithm is run in rounds. This approach has been used in [10] and has been proven to be able to efficiently deal with the fatigue and orphaned node problem. Interested readers may wish to consult [10] for further details. In the initial phase of the algorithm, also known as the "initialization" phase, sensor nodes execute the EAD with the objective to identify the non-leaf nodes and set up the backbone. Once that is over, the nodes proceed to the "data-transmit" phase, where the nodes transmit the data to the sink. The initialization and the data-transmit phases constitute a single round. When one round terminates, the initialization phase for the next round begins, and dead nodes and orphaned nodes are identified. As outlined in [10], the initialization phase is smaller when compared to the data-transmission phase. As soon as the data-transmit state for current round expires, the sink will initiate a new round initialization by re-constructing the broadcast tree. This process helps identifying the non-leaf nodes' fatigue problem [10].

4.3. Topology-based scheme

When applied to large-scale sensor networks, EAD may take too much time since the execution process is propagated from the sink to the whole network. In our implementation, we have chosen to "pre-process the network topology" using a topology-based scheme. The idea is to turn off the radio of some sensors such that only a subset of sensors participate to run EAD. Our topology-based scheme guarantees a rooted broadcast tree spanning all sensors even though only a subset of them are participating in the execution of EAD.

Let us now describe how to determine which sensors should be active if no position information is available. Let's consider the number of active neighboring nodes in each direction. Assume that each sensor has k directions. Note that if $\alpha = 120°$, then $k = 3$. Let n be the number of active neighbors. Suppose that n_i neighbors are in direction i. Then $n_1, n_2, \ldots, n_k$ follow the multinomial distribution:

$$p_{n_1,n_2,\ldots,n_k} = \frac{n!}{n_1! \cdot n_2! \cdots n_k!} \cdot p_1^{n_1} \cdot p_2^{n_2} \cdots p_k^{n_k} \qquad (1)$$

where $p_{n_1,n_2,\ldots,n_k}$ is the probability that n_i neighbors are in directions i, p_i is the probability that a neighbor is in direction i, and $n_1 + n_2 + \cdots + n_k = n$. If all neighbors have the same probability to be in any direction i, that is, $p_1 = p_2 = \cdots = p_k = \frac{1}{k}$, then

$$p_{n_1,n_2,\ldots,n_k} = \frac{n!}{n_1! \cdot n_2! \cdots n_k!} \cdots \frac{1}{k^n} \qquad (2)$$

The probability P that at least one neighbor appears in each direction is $\sum_{n_1 \geq 1, n_2 \geq 1, \ldots, n_k \geq 1} p_{n_1,n_2,\ldots,n_k}$. Typical values of P are listed in Table 1.

Note that if each sensor has 4 or 5 active neighboring sensor nodes, then with a probability around 50%, it has one neighbor in each direction if $k = 3$. Based on this observation and assuming that initially all sensors have their power supplies off, we propose the following algorithm.

A sensor u randomly (once every T_0 time units) wakes up and broadcasts a hello message. An active neighbor v replies a message with a binary *INVI* bit. If v has less than 4 neighbors, then *INIT* $= 1$; Otherwise, *INIT* $= 0$. If u receives a message with *INIT* bit on, or u detects that it has less than 4 active neighbors, it will remain wake-up; otherwise, it goes back to sleep. After T_0 time, apply EAD to build a broadcast tree rooted at the sink. Note that with this approach, we can not guarantee a tree spanning all of the active sensors. But since sleeping sensors wake up periodically in order to determine its parent in the tree, they can be invited to join the tree as non-leaf nodes by active neighbors who need help to connect to the tree.

Table 1
The probability that at least one neighbor appears in each direction.

$k\backslash n$	3	4	5	6	7	8	9	10	11	12	13	14	15
3	0.22	0.44	0.62	0.74	0.83	0.88	0.92	0.95	0.97	0.98	0.98	0.99	0.99
4		0.09	0.23	0.38	0.51	0.62	0.71	0.78	0.83	0.87	0.91	0.93	0.95
5			0.04	0.12	0.22	0.32	0.43	0.52	0.61	0.68	0.74	0.79	0.83
6				0.01	0.05	0.11	0.19	0.27	0.36	0.44	0.51	0.58	0.64

5. Simulation environment

We have implemented our proposed protocol EAD and the optimized topology based scheme. The experiments were conducted using the Network Simulator ns-2 [24], a discrete event simulator widely used by the research community for wireless and wired network simulations. Currently ns does not have an adequate framework required to conduct sensor network experiments such as EAD. Furthermore, it doesn't have any support for the radio electronics for advanced power aware sensors such as the μ Adaptive Multi-domain Power Aware Sensors (μAMPS [10]) developed at MIT. As a result we have used the μAMPS extensions developed by MIT for ns2.1b5. The μAMPS extensions were developed for the simulation and performance evaluation of wireless sensor networks based on the μAMPS sensors. The simulation reported in this section includes a comparative study of the EAD forwarding-to-parent routing scheme over a tree created by a single (initial) EAD execution to a regular Ad hoc On-Demand Distance Vector (AODV) routing protocol [20], and a comparative study of EAD and LEACH (Low Energy Adaptive Clustering Hierarchy) as described in [10].

Our simulation experiments were carried out assuming μAMPS sensor nodes and implemented under the μAMPS framework. The statistics collection used by μAMPS was modified to suit the metrics we have used in our experiments. All of the experimental results presented in this paper were obtained by averaging multiple trial runs with a 95% confidence interval.

5.1. Performance metrics

In this section, we discuss the performance metrics we have selected to evaluate the performance of the EAD algorithm. The metrics were chosen to make an effective evaluation of the performance characteristics of EAD as an efficient and scalable routing algorithm for sensor networks. The parameters used in the simulation are reported in Table 2.

Table 2
Simulation parameters.

Simulation time	4500 seconds
Starting Energy for each node	2 J
Threshold for Error-free packet, RXThresh	$6e^{-}9$ W
Threshold for detection, CSThresh	$1e^{-}9$ W
Radio Electronics Energy, Excvr	$0e^{-}9$ J/Bit
Transmit Amplifier energy, ϵfriss_amp	$9.6741659015025702e^{-}12$ J/m^2
Amplifier energy, ϵtworay_amp	$1.303703703703703e^{-}15$ J/m^4
Beam forming energy, ϵbf	$5e^{-}9$ J/bit/Signal
Energy dissipation during sleep, PSleep	0

- **Total_number_of_active nodes**: indicates the number of alive nodes and thus the failed nodes due to low energy during the execution of our simulation models. The failure of a node may be characterized by its inability to generate packets that meet or exceed a certain threshold value (CSThresh). It is very important for any efficient routing algorithm to have enough alive nodes throughout its execution in order to be able to send data to/through the base station.
- **UDP Packets Throughput**: is the total number UDP data packets delivered to the sink. The primary task of the algorithm is to deliver data to the sink from the leaf nodes as efficiently as possible. This metric is used to evaluate the packet delivery achieved by EAD.
- **Energy_expended**: measures the total energy expended by the network. This metric is an important parameter in evaluating the effectiveness of the EAD algorithm and its power saving capability.

5.2. Simulation results

In this section, we report on the experiments which were carried out to assess EAD's performance on the ns-2. In our EAD's implementation, we have used the *topology-based approach* as described in Section 4. We have conducted an extensive set of simulation experiments to evaluate the performance of EAD protocol. We divide our discussions of the experimental results into three parts. In the first part, we assess the effect of the EAD refresh interval and the value of the time interval used to wake up the sensor nodes, which we refer to as T_0, as discussed earlier in Section 4. Then, we investigate its optimal value and evaluate the overhead of the EAD protocol. In the course of our discussions, we point out significant factors affecting EAD's performance. In the second part, we compare the performance of the EAD forwarding-to-parent routing scheme over a tree created by a single (initial) EAD execution with a regular Ad hoc On-Demand Distance Vector (AODV) routing protocol [20]. In the third part, we compare the performance of our EAD scheme to LEACH protocol [10].

5.2.1. Performance evaluation of EAD protocol

To evaluate the performance of EAD protocol, we used 100 and 200 sensor nodes in a 500×500 m^2 area with a grid topology (figure 6), where the nodes are either 50 or $50\sqrt{2}$ meters away from their neighbors. We assume that at the physical layer a broadcast (either EAD backbone formation, or UDP data packet) can only reach the one hop neighboring nodes. There are two physical layer channels: Channel-1 is used for data transmission while channel-2 is used for phenomena spread (phenomenon channel).

In our model, there are three different nodes. The sink node (black node) has infinite power. It is placed at an extreme edge of the grid to test the effects of hop count and bottleneck nodes more clearly. Sensor nodes are assigned energy of 2 Joules initially, and they are stationary. Non-leaf sensor nodes are colored with brown while leafs are gray. Sensor nodes

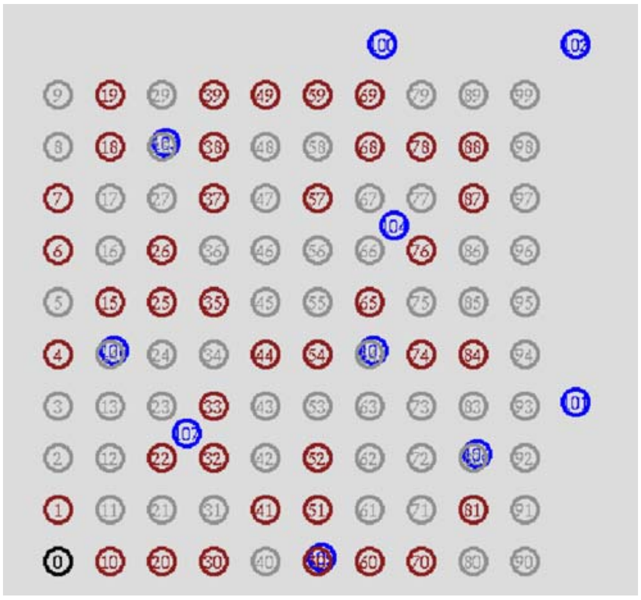

Figure 6. Simulation layout.

use channel-1. Event generators (blue nodes) are specifically designed to emanate phenomenon packets at variable rate. In the simulation, we used 10 medium rate (10 packets/sec) event creators in which two of them were mobile (to simulate a moving cloud). Phenomenon nodes use channel-2.

Sensor nodes have an interface at channel-2. Therefore, if these nodes detect a packet in that channel, they create a UDP data packet and send it to sink. A UDP packet was selected mainly due to its connectionless nature which eases the acknowledgment burden in the network. Each UDP packet has the same length of 100 bytes, in which 28 bytes form the header and the rest is data. Each EAD control packet is 48 bytes long. However, in our total packets throughput measurement, we use *packets* instead of *bytes*. The aggregation logic is our choice. For the correlated data assumption, a non-leaf node waits until it gets a certain number of packets (number of leafs), and then sends a single packet as an aggregated data. The data coming from a non-leaf node (already aggregated) is sent as it is, without any aggregation. If the assumption is uncorrelated data, then all nodes send whatever they got to the next node in the hierarchy, without waiting for a certain number of packets.

Let us now turn to our results. In what follows, we wish to evaluate the performance of our EAD protocol under different settings of EAD refresh interval and different time interval T_0 used to wake up the sensors node. We also used the topology based approach as described in Section 5.

In order to observe the system performance for different values of T_0, we have used a network with a 100 nodes layout, and fixed the value of the EAD refresh interval to 20 seconds. Figures 7 and 8 display both the total number of UDP packet reaching (or UDP packet throughput) the sink and the number of alive nodes during the execution of the simulation. Our results show that a small value of T_0's induces a slightly high UDP packets throughput and a high energy consumption. We

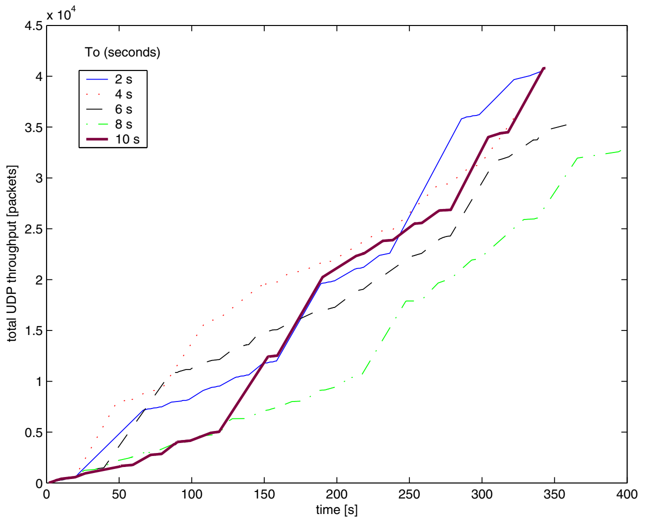

Figure 7. Total throughput with different T_0's, 100 Nodes, Energy Settings 2.

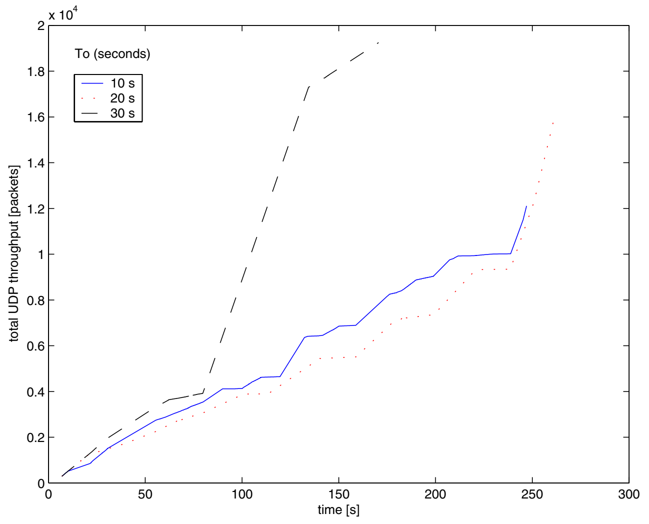

Figure 9. Total throughput with different T_0's, 400 Nodes, Energy Settings 1.

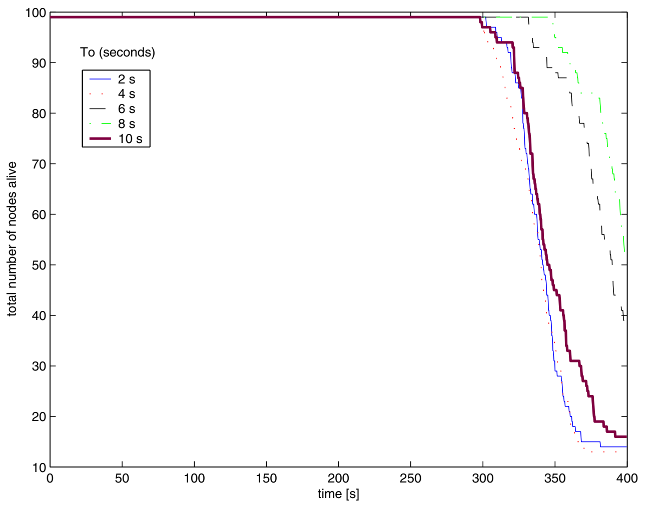

Figure 8. Number of nodes alive with different T_0's, 100 Nodes, Energy Settings 2.

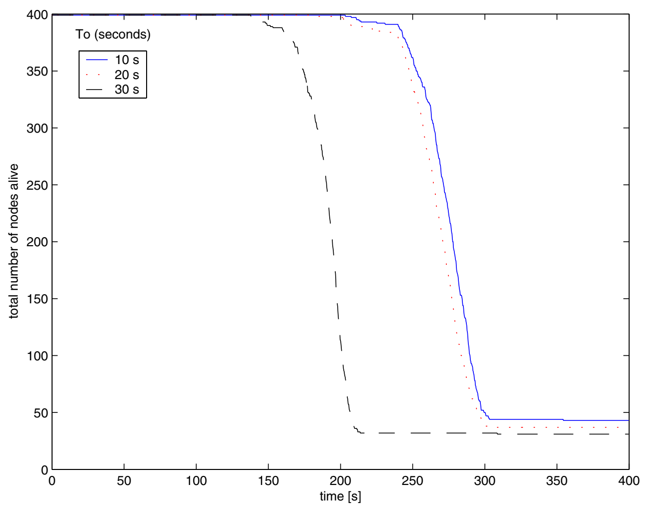

Figure 10. Number of nodes alive with different T_0's, 400 Nodes, Energy Settings 1.

have also investigated the case where we have network size of 400 sensor nodes and the EAD refresh interval is fixed at 40 seconds. Figures 9 and 10 portray the results we have obtained while varying the values of T_0. Our results indicate that with smaller T_0 values the system achieves lower packet throughput but operates longer when compared to a 100 nodes population. These results suggest that the size of the network and the choice of the value of T_0 has a great impact in the performance of EAD.

In our next set of experiments, we wish to investigate the effect of the choice of EAD refresh interval (or round time) on the EAD performance. Recall that the best features of EAD can be summarized as follows: the sleeping nodes are at most one hop away from a backbone node. An event occurring at close proximity of a sleeping node has a high likelihood of being detected by a backbone node. Therefore, this tree-like coverage area has inherent power saving advantages if the sleeping and awake nodes are carefully chosen, which is the case in EAD. This coverage area also decreases the probability of an event going undetected for a long period of time, and the optimum value of EAD refresh interval depends on three main factors: (i) The event characteristics: traffic generated in the network; (ii) Initial node energy; and (iii) Energy spent by packet transmission and their receptions (data/event packets). Furthermore, in order to determine the optimal value of the EAD refresh interval, the following three energy settings must be defined: (i) *series_initial_node_energy*: represents the initial energy assigned to a node; (ii) *series_txPower*: represents the energy spent for a single packet transmission; (iii) *series_xPower*: represents the energy spent for a single packet reception; and (iv) *series_sensePower*: represents the energy spent for a single event packet reception.

Table 3 presents the first set of energy settings used in our experiments to study the effect of varying the EAD refresh interval.

Table 3
Energy Settings 1.

Initial node energy	txPower	rxPower	sensePower
2 Joules	0.02 mW	0.02 mW	0.000002 mW

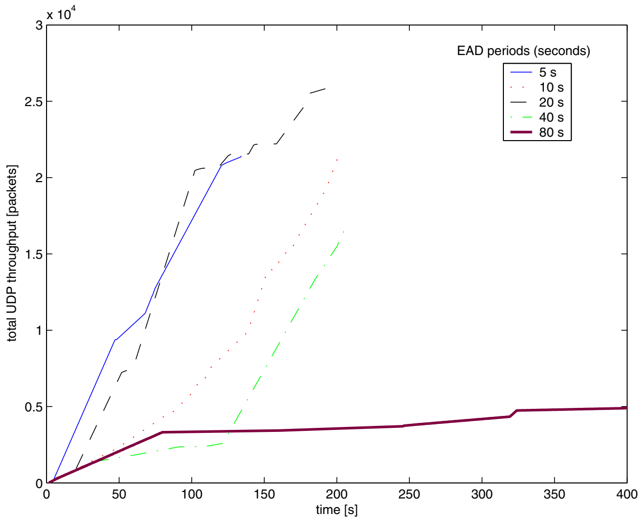

Figure 11. Total throughput with different EAD refresh intervals, Energy Settings 1.

Figure 11 portrays the values of the total UDP throughput, i.e., total number of packets delivered to the sink we have obtained in our simulation experiments, as we vary the EAD refresh interval. Our results indicate that shorter EAD refresh interval increases the system throughput, while longer EAD refresh interval increases the system lifetime. In the course of our simulation experiments, we first created a power limited subnet to test the burden imposed by EAD. Using an initial energy of 2 Joules. When the energy spent for each packet transmission and reception is selected as 0.02 mW, our results have shown that a higher number of EAD executions decreases the system lifetime. For this setting (Setting 1), we have observed, see figure 11, that EAD with an EAD refresh interval of 20 seconds performs best as far as the throughput performance metric is concerned. We also observe that with an EAD refresh interval of 60 seconds, the system operates quite well with a lower throughput, though making a better use of node's available energy. Hence, we believe that the selection of EAD refresh interval has a great effect on the performance of EAD, and may be closely tied to the type of applications that is used for. Sensing jobs, requiring a high throughput, can use small refresh interval as long as the energy supplies are robust.

Figure 12 shows the number of alive sensor nodes with different EAD refresh intervals. Here again, we see that with high refresh frequencies more nodes die sooner when compared to other nodes. We also observe a sharp end in total throughput. This is mainly due to the fact that, in general, the nodes at one hop neighborhood of the sink are mostly non-leaf nodes which are awake most of the time, and when they die, no packets can reach the sink. This kind of sharp end of the simulation is a general behavior where the destination (sink in our case) can only be reachable through a limited number of highly utilized but energy limited nodes which constitute a bottleneck.

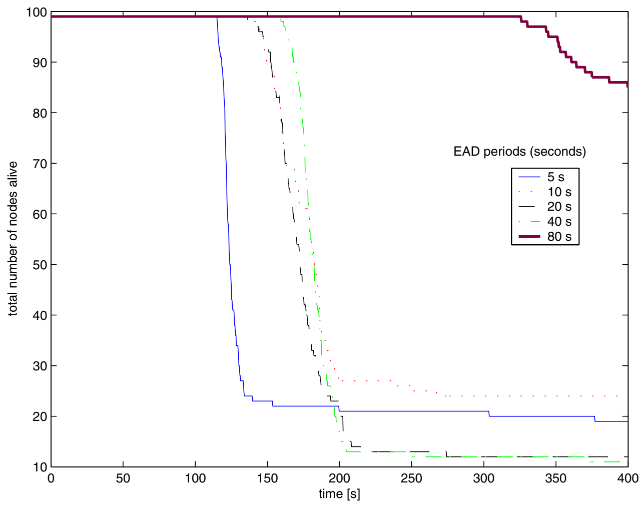

Figure 12. Number of nodes alive with different EAD refresh intervals, Energy Settings 1.

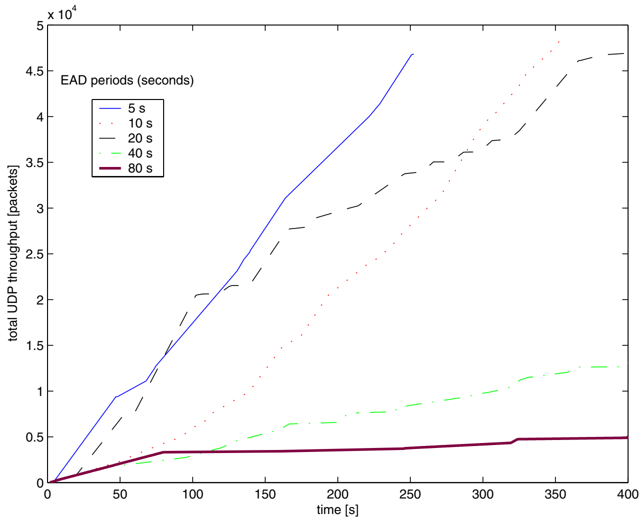

Figure 13. Total throughput with different EAD refresh intervals, Energy Settings 2.

Let us now, consider a more relaxed model where energy consumption for transmission and reception routines are halved. Using the parameters, as described in Table 4, the throughput of different EAD intervals is displayed in figure 13. With these settings (Energy Setting 2), a refresh interval of 10 seconds achieves slightly higher throughput when compared to a situation when we use a refresh interval of 20 seconds.

Table 4
Energy settings 2.

Initial node energy	txPower	rxPower	sensePower
2 Joules	0.01 mW	0.01 mW	0.000001 mW

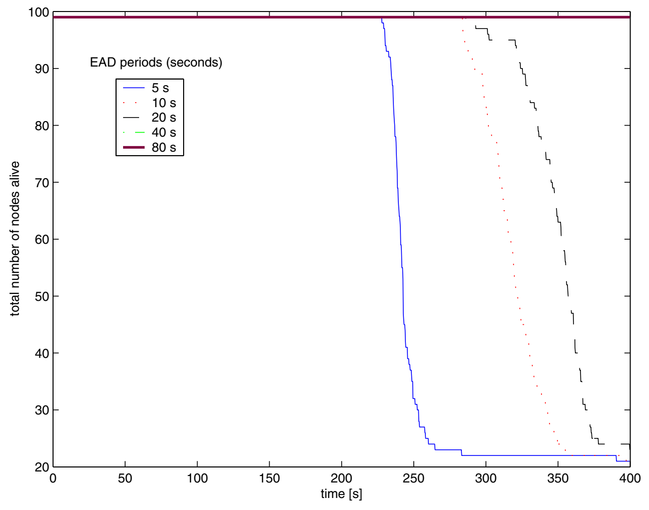

Figure 14. Number of nodes alive with different EAD refresh intervals, Energy Settings 2.

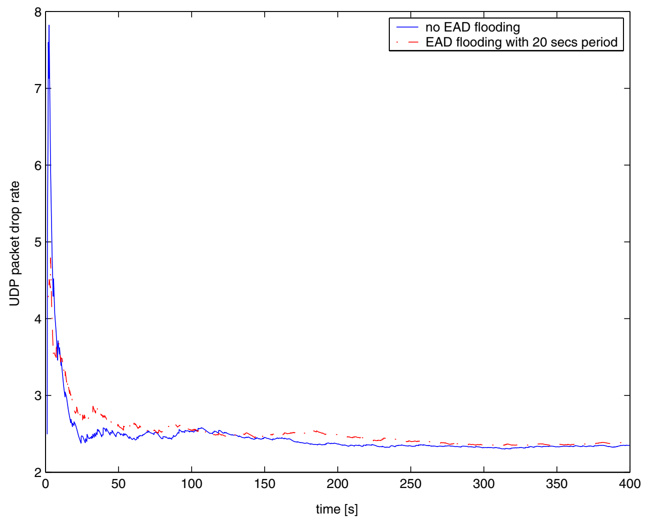

Figure 16. UDP packet drop rate and without EAD flooding.

However, with a 20 seconds interval, the system continues to deliver packets to the *sink* for significantly a longer time. Figure 14 illustrates the number of alive nodes throughout the execution of the simulation. Since the nodes spend less energy for transmission and reception, they live longer. In our simulation experiments, we have observed that no node dies during the entire simulation for at least the refresh interval of 40 seconds.

Let us now, evaluate the overhead incurred by the EAD protocol. Since EAD operates at the same channel with the data packets, we tested the contention effects of EAD flooding over the data flow toward the sink. Figure 15 shows the average delay of data packets with and without EAD flooding without constraining the energy of the nodes. We conclude that EAD overhead does not cause contention. Figure 16 gives the packet drop rates of the cases.

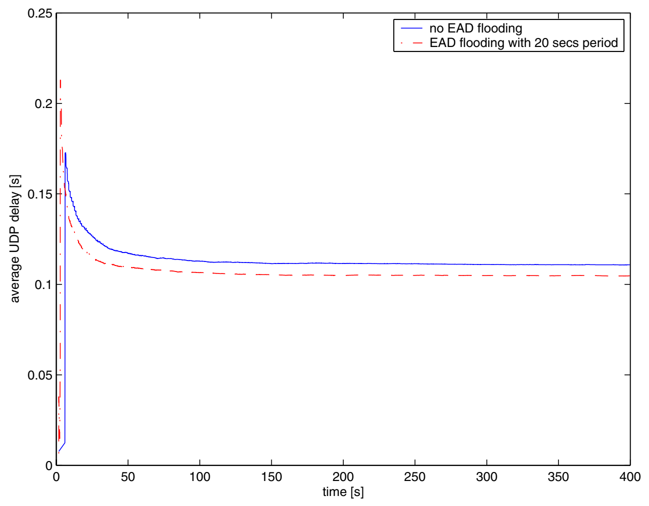

Figure 15. Average UDP delay with and without EAD flooding.

Last, but not least, during the course of our simulation experiments, we have investigated the trade-off between the duration of the "initialization" phase and the "data transmit" phase during the maintenance of the tree, that is, how the performance of the network varies with different values of the update frequency. While it is very hard to control the duration of the execution phase, we have designed a state machine and provided its rules. We believe that the execution length of the EAD protocol is closely dependent on the network diameter. Our simulation experiments with 500×500 and 100 nodes have revealed that the execution of EAD protocol lasts less than 0.05 seconds. We have also noticed that there were no tradeoff between the initialization and data transmit phases.

5.2.2. EAD vs. a simplified-AODV routing protocol: A comparison

While designing sensor networks, it is important that the communication protocol used is energy-efficient with a minimum communication overhead and a minimum memory usage. After the backbone formation phase, EAD uses a single rule for packets' transfer which is identified as a *forward-to-parent* scheme. This scheme shall help decrease the communication overhead while keep a small amount of memory space usage. In this section, we wish to evaluate the efficiency of the *EAD forwarding-to-parent* routing protocol over a tree created by a single (initial) EAD execution without being concerned about EAD roles and its power on-offs in energy saving mode.

A basic comparison then would be to compare EAD forwarding-to-parent routing protocol with a modified ad hoc routing algorithm, a scheme that does not consider mobility and therefore minimizes the routing overhead by curtailing the number of control packets to a minimum. In our simulation experiments, we have chosen AODV and modified it to its bare bones [20].

Let us now turn to our results. Figure 17 portrays the results we have obtained to compare the total UDP packet throughput

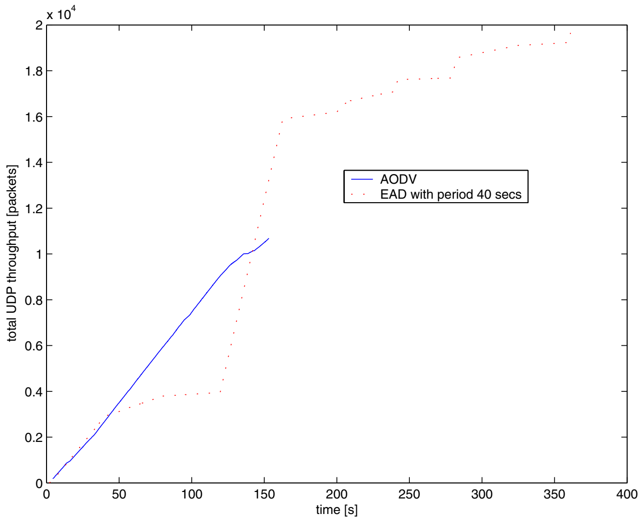

Figure 17. Total throughput with EAD and simplified-AODV, pulse-rate of 1 packets/sec.

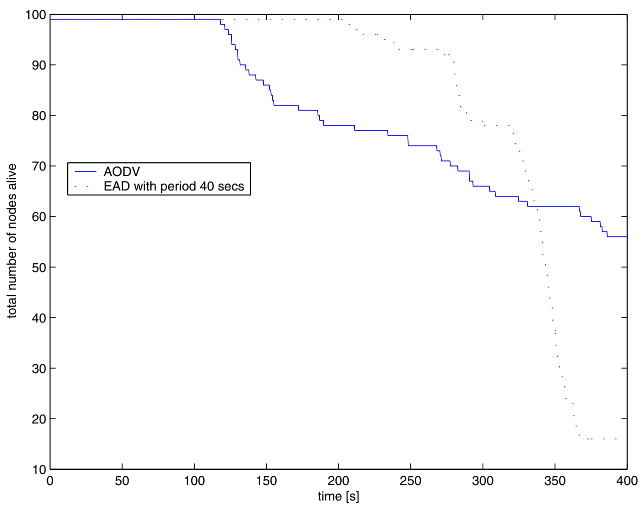

Figure 18. Number of nodes alive with EAD and simplified-AODV, pulse-rate of 1 packets/sec.

(i.e., the total number of packets delivered to the sink) achieved by the EAD forwarding-to-parent scheme and a simplified-AODV protocol while figure 18 displays the number of nodes alive during the execution of the simulation. As we can see, the throughput of EAD is significantly higher when compared to a simplified-AODV. Our belief is that once a spanning tree using the EAD protocol is constructed, each node knows where to send its packets since there is no mobility in the network. The dead nodes are eliminated from the backbone formation phase in the next round, therefore a dead node can only cause link to break at most for the duration of EAD refresh interval. As stated earlier, EAD is especially powerful in an event driven sensor network model. At first, it seems that the energy efficiency (in terms of alive nodes) is better when one uses a simplified-AODV protocol. However, due to its control message overhead and its energy-unaware paths, key nodes around

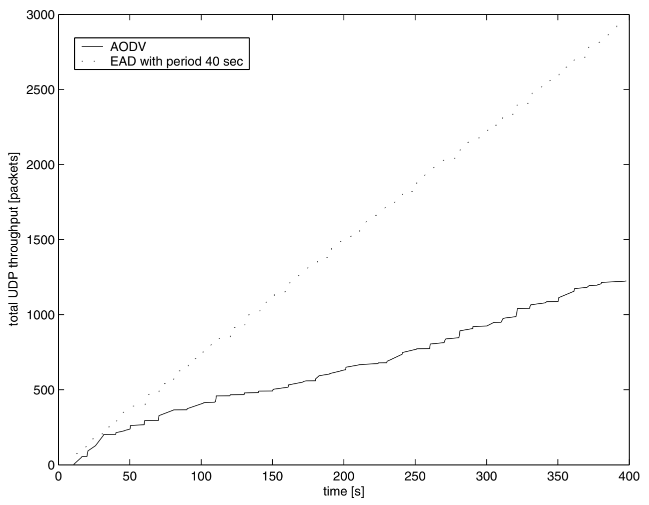

Figure 19. Total throughput with EAD and simplified-AODV, pulse-rate of 0.1 packets/sec.

the base station die much earlier than expected, thereby cutting off the base stations from the rest of the network. Figure 19 displays the throughput results for a reduced event creation rate of 0.1 packets/sec.

5.2.3. Performance evaluation of EAD vs LEACH: A comparison

In this section, we wish to evaluate EAD protocol when compared to a Low-Energy Adaptive Clustering Hierarchy (LEACH) protocol [10]. Before we proceed further, we shall describe briefly the LEACH scheme.

(a) *LEACH: A low-energy adaptive clustering hierarchy protocol*

LEACH is a cluster-based, energy-aware routing protocol specifically designed for sensor networks [10]. This protocol makes use of inherent properties of sensor networks. It assumes that the data to be transmitted in a sensor network is locally correlated. Therefore, if the data is processed and aggregated at local centers, or cluster heads, before being sent to the base station (BS), the energy spent in the whole network will be reduced, and, thereby increasing the system lifetime. The identification and the maintenance of the cluster head is an energy-consuming task, since data coming from other clusters are first aggregated (i.e., data processing), and then sent to the base stations via the cluster heads. At each round, the cluster head randomly rotates between the cluster members which guarantee the close-to-uniform distribution of energy. Cluster head selection and cluster formation can be done in a distributed or centralized manner.

In both the centralized (LEACH-C) and the decentralized LEACH model, the cluster heads remain cluster heads within the interval which is identified as *round*. The operation of LEACH is designed within these rounds. The main steps *in a LEACH round* can be summarized as follows:

- *Set-up phase* (*also known as a cluster head selection*): At the current round, each node generates a random number between 0 and 1, and compares it with a threshold that is a function of the expected percentage of cluster heads for the network and the total number of times the node has been a cluster head so far. If the random number is less than the threshold, the node becomes a cluster head.
- *Set-up phase*: *cluster formation*: Each cluster head broadcasts a message announcing itself as the cluster head for this round. A non-cluster node receiving multiple cluster head announcements chooses the cluster head that requires less energy to communicate with. Then, non-cluster nodes inform their cluster heads about their selection. The cluster head node then sets up a TDMA schedule and broadcasts this schedule to its members. After each node learns the TDMA schedule of their cluster, set-up phase ends.
- *Steady-state phase*: At the steady-state phase, each node transmits data to the cluster head during its allocated slot. At the end of each frame when the cluster head has received data from all associated sensor nodes, it aggregates the data and sends it to the base station. The cluster heads send this data to base station using CSMA. To save energy, non-cluster nodes can turn off their power until their allocated time slot in the TDMA cycle.

(b) *EAD vs. LEACH: A comparison*

In this section, we will report on the results we have obtained to evaluate the performance of EAD protocol when compared to the LEACH protocol. Both LEACH and EAD offer methods for selecting higher energy nodes for intense use, and methods of changing the overly used node when its energy level is lower than that of its neighbors. LEACH-like algorithms make use of the correlated nature of data (or a max-min kind of selection) and send less packets to the sink using, cooperatively chosen, higher-energy cluster-head nodes. If the data is correlated, it is obvious that, this scheme is advantageous over no aggregation algorithms in which nodes send, whatever they get, to the data collection node. EAD has an advantage, when compared to LEACH, independently from the correlated nature of data. If the data is correlated, then less data will be sent over the backbone nodes. LEACH offers a star-like subnet creation inside a cluster and introduces a power on and off scheme during a TDMA cycle. Basically, a non-cluster head node is allowed to go to sleep until it reaches its turn in the TDMA cycle. Implementing very short on-off cycles (in the scale of TDMA duration) may not be feasible. EAD uses longer (in the scale of EAD interval) on-off cycles to save energy. EAD implements a tree-like coverage structure and multihop transmission of aggregated data over this tree. In LEACH, aggregated data is transferred to the base station in a single-hop transmission which assumes that each node in the subnet is able to reach the base station. Furthermore, LEACH is suitable for continuous sensing jobs where every node has data to send at regular intervals. It is not suitable for event-driven models, since a node has to wake up and transmit in short intervals, therefore spend energy regardless of whether or not it has detected an event. EAD is more suitable in event driven environments. During an EAD round, EAD puts a large number of nodes into sleep, as long as they do not detect anything while maintaining connectivity with the non-leaf nodes. In EAD, nodes do not have to wake up and send if there is no detection. Therefore, if EAD is used in an event driven model, a contention based channel access (like CSMA) is likely to be more efficient because of the light traffic an event driven model produces. In this implementation, we have used 802.11 CSMA/CA MAC for data transfer as a result of sensing events.

Traffic characterization is an important issue. In order to create a *realistic phenomena* cloud, we used *mobile* and stationary event creating nodes that emanate phenomenon packets at a different channel. Sensor nodes have interface at both data and phenomenon channels. One can always raise objections to the way event sources are created. We decided that this kind of event creation logic are closer to the real life events for which sensor networks designed. Basically we are creating an event cloud instead of using only stationary event sources.

Let us now turn to our results. Figures 20–24 show the number of nodes alive plotted against simulation time and a network size of 50, 75, 100, 150 and 200 nodes respectively.

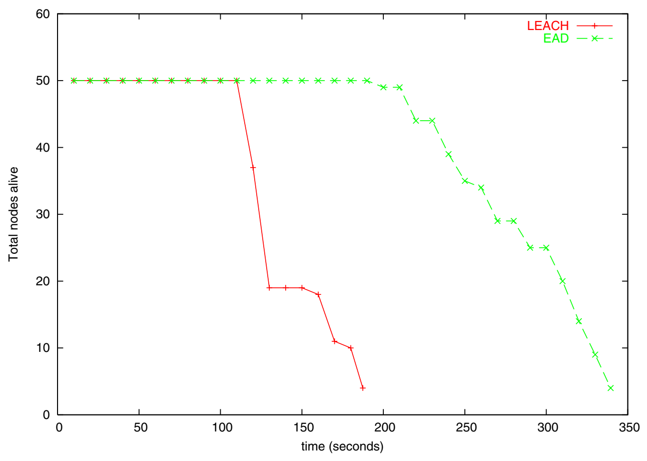

Figure 20. LEACH vs. EAD (50 Nodes).

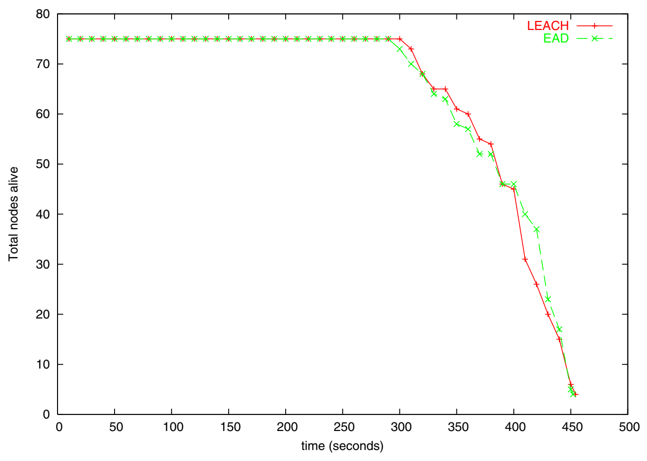

Figure 21. LEACH vs. EAD (75 Nodes).

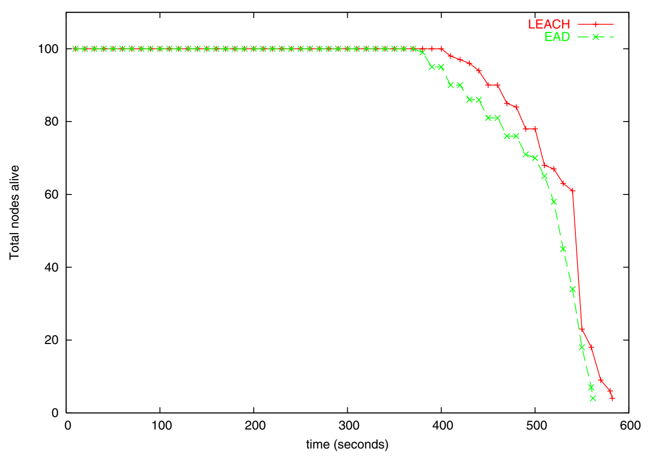

Figure 22. LEACH vs. EAD (100 Nodes).

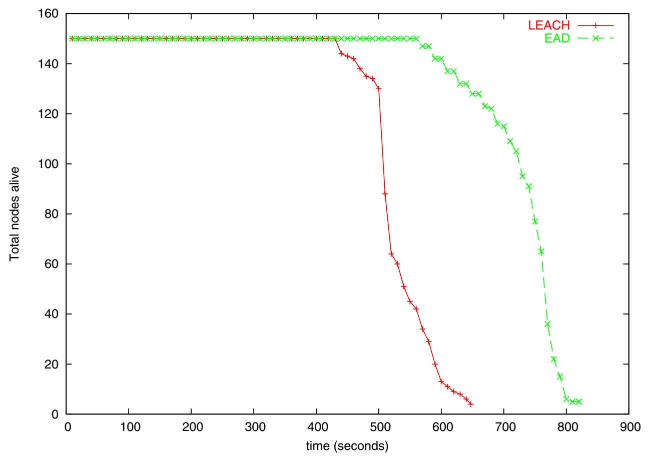

Figure 23. LEACH vs. EAD (150 Nodes).

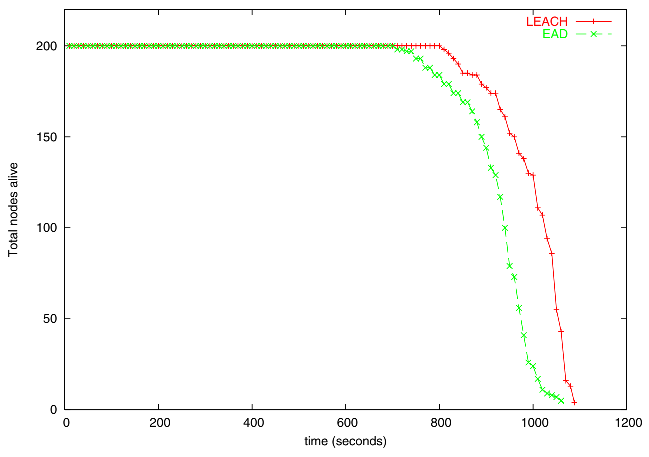

Figure 24. LEACH vs. EAD (200 Nodes).

The amount of energy per node is 2 Joules at the beginning. As we can see from the figures, the number of nodes alive decreases after some simulation time. As non-leaf nodes fail, the loads on the remaining nodes increase and more nodes are woken up and recruited in the tree. The failures increase rapidly after a critical point in simulation. In figure 24 the node failures increase rapidly for the EAD curve after 200 seconds. Similarly for the rest of the curves the node failures increase rapidly towards the end of the simulations. Both EAD and LEACH behave in a similar way in this respect. It can be seen that EAD performs better than LEACH in the figures in terms of the node failure rate. EAD routes the data packets to the sink by multihop routing as opposed to LEACH where the cluster-heads have to transmit the data directly to the base station. The energy dissipated is lower in the case of EAD because the backbone node transmits only to a neighboring node one level up from it.

Figures 25–29 portray the total energy dissipated vs simulation time for the same set of network topologies. There is a limited energy supply and the amount of energy per node is 2 Joules. For this particular simulation the sleep energy have been set to zero. In actual sensor networks PSleep is a negligible quantity which can be safely ignored in a simulated environment like this. Energy dissipated is a measure of the power awareness of our algorithm, which attempts to extend network lifetime by forming a routing tree rooted at the sink, and recruiting only a minimum number of non-leaf nodes. The non leaf nodes are the only nodes that have to stay awake

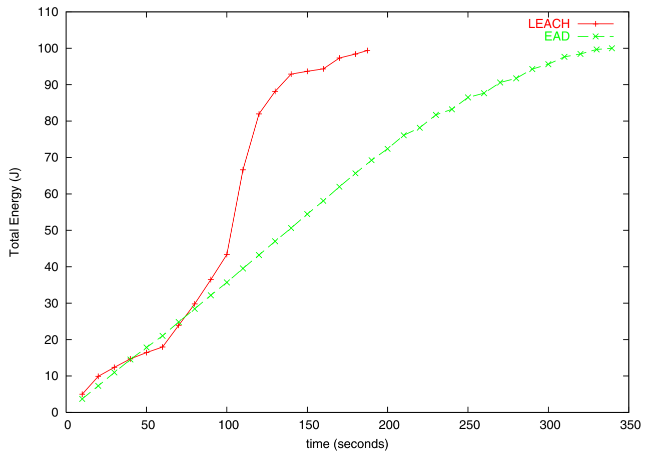

Figure 25. LEACH vs. EAD (50 Nodes).

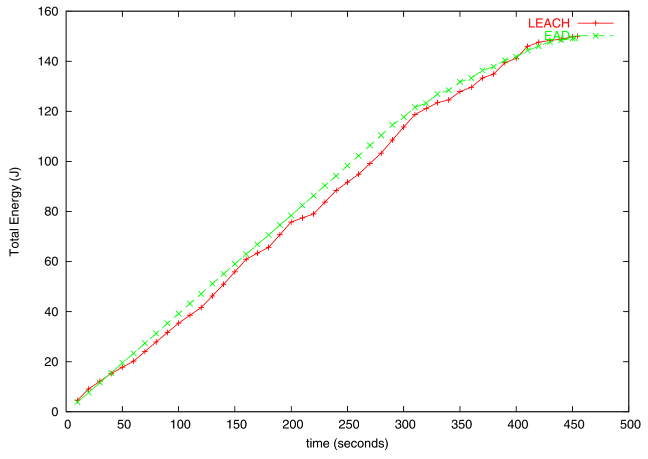

Figure 26. LEACH vs. EAD (75 Nodes).

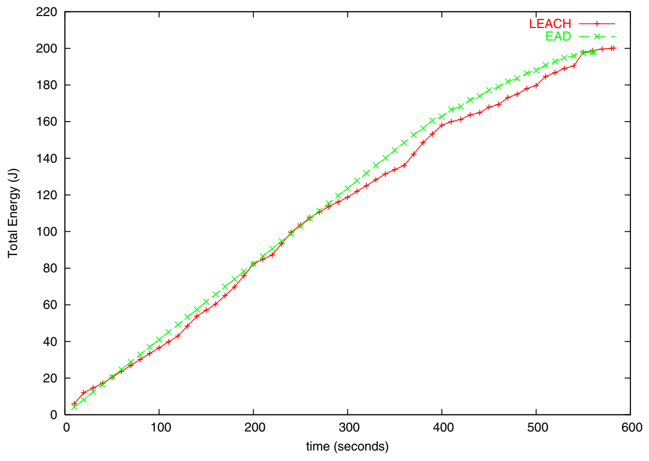

Figure 27. LEACH vs. EAD (100 Nodes).

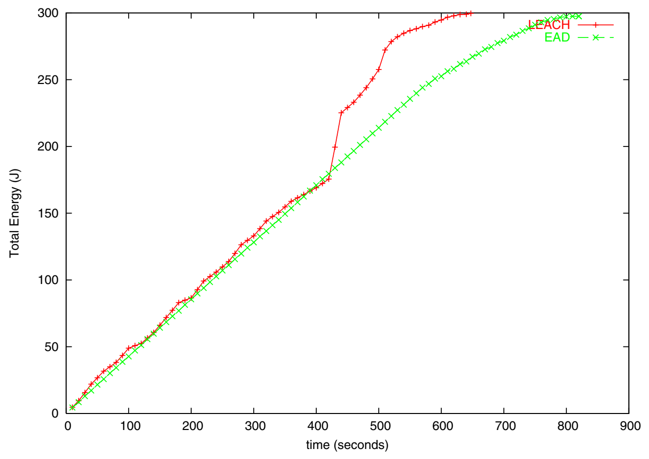

Figure 28. LEACH vs. EAD (150 Nodes).

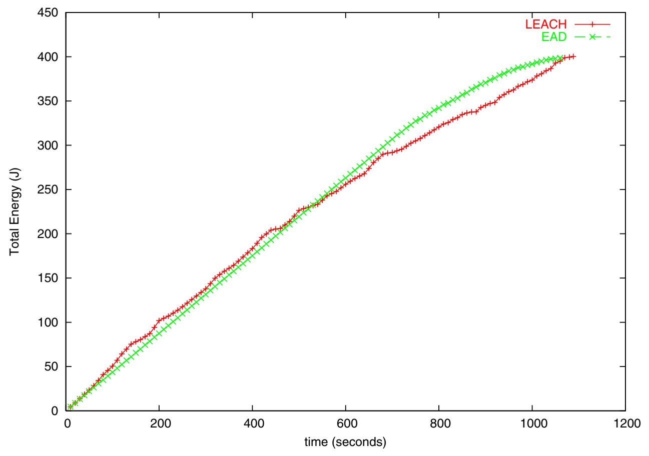

Figure 29. LEACH vs. EAD (200 Nodes).

throughout a single round to be able to receive from the leaf nodes and transmit to the base station. This is the reason why EAD performs better compared to LEACH. The amount of work involved in setting up the virtual back bone in the case of EAD makes it slightly costlier than LEACH during the set-up phase but this is not a disadvantage when looking at the overall performance. The energy savings in the steady or data-transmit phase of EAD make it more efficient than LEACH overall.

Figures 30–34 illustrate data throughput to base station plotted against simulation time. Here again, we consider networks with different node populations, i.e., 50, 75, 100, 150 and 200 number of nodes respectively, and the amount of energy per node fixed at 2 Joules. Our results indicate that the total number of packets delivered to the sink is cumulative and steadily increases as we increase the simulation time. The gradual flattening of the curve towards the end of simulation is due to the fact that the nodes are failing as the simulation progresses in time. Our results also indicate that the packet throughput increase slows down after 300 seconds. A Similar behavior is seen in the other figures as well. This is mainly due to fact that the number of alive nodes is significantly low, thereby lowering the packet throughput delivered to the sink. Our results show that the total number of packets increases from 30,000 to 35,000 and data signals are significantly slow for this reason.

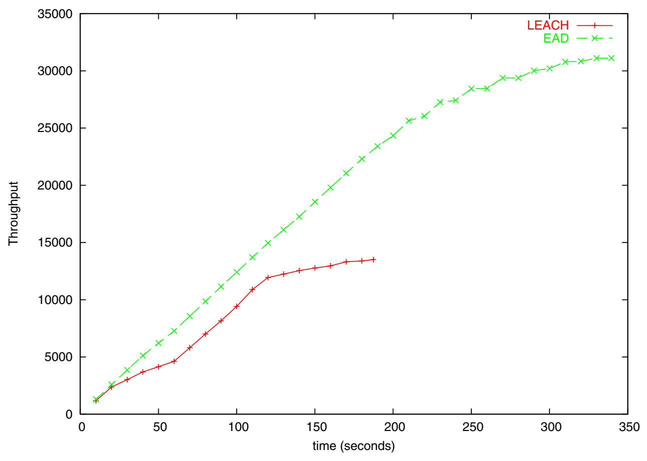

Figure 30. LEACH vs. EAD (50 Nodes).

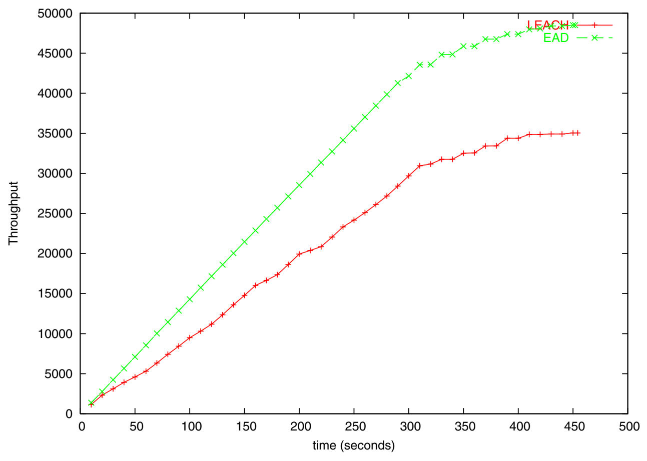

Figure 31. LEACH vs. EAD (75 Nodes).

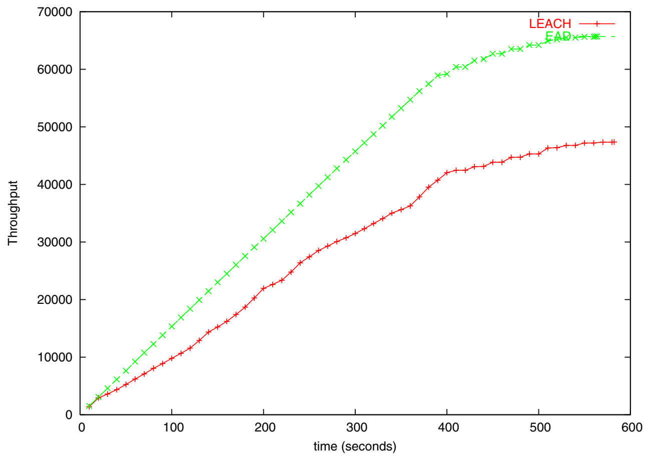

Figure 32. LEACH vs. EAD (100 Nodes).

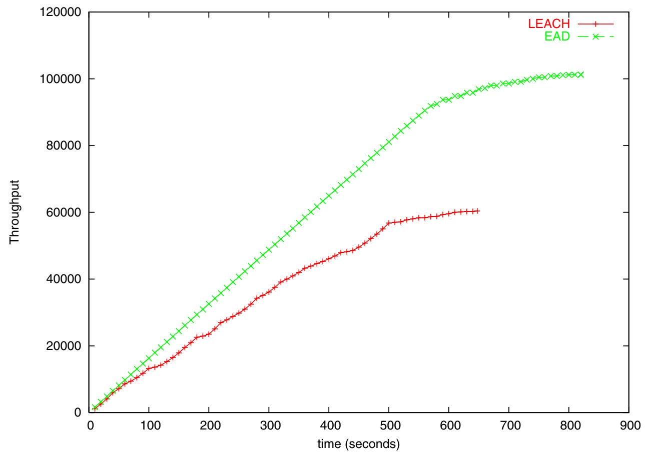

Figure 33. LEACH vs. EAD (150 Nodes).

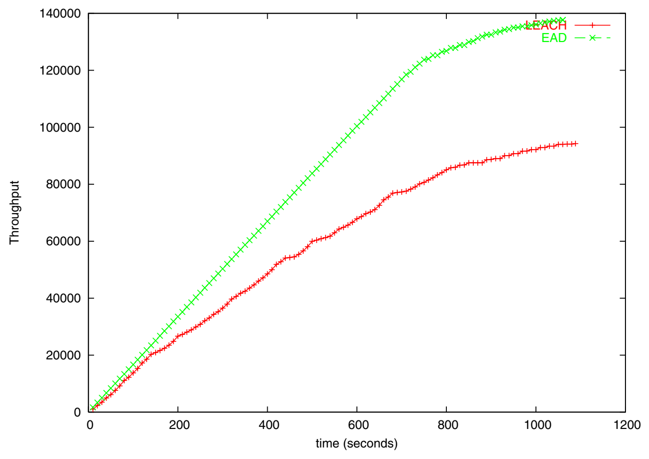

Figure 34. LEACH vs. EAD (200 Nodes).

As outlined in our set of simulation experiments, our results indicate some gain of EAD over LEACH, due to the effective coverage area and structure introduced in EAD with the innovative idea of creating a broadcast tree rooted at the sink. Though, not specific to EAD, this kind of *coverage* paradigm is more effective if there are overlapping sensor coverage areas. This means that with a selection algorithm, if we could decrease the number of powered-on nodes, we can save energy without compromising the well functioning of the nodes and their sensing task. However, as shown in our experimental simulations, EAD is quite effective even in the worst case conditions, where nodes do not have overlapping coverage. Note that even in the worst case settings, the probability of an event going undetected is very low since EAD covers the subnet area with a low number of nodes participating in it. As mentioned earlier, after the EAD execution phase, all sleeping nodes are, at the most, one hop away from a backbone node, and an event, occurring at a close proximity of a sleeping node, has a high likelihood of being detected by a backbone node.

6. Conclusion and future work

Recent innovative wireless technologies, and the evolution of smart devices and smart sensors have played a major factor in the development of future wireless sensor systems. However, before these systems become a common place, many challenging issues need to be resolved. In this paper, we focus upon the energy consumption related problems, and we propose a data-centric routing mechanism based on a broadcast tree routed at the sink node with a maximum number of leaves. We have presented our EAD protocol, and we have reported on its performance evaluation using an extensive set of simulation experiments. Our results indicate that EAD extends the overall network lifetime by turning off the transceivers of all leaf nodes in the broadcast tree, leaving only non-leaf nodes in charge of data aggregation and traffic relaying.

As a future work, we intend to study analytically the performance of our scheme, and define an analytical function that determines the optimal value of the EAD refresh interval. We also plan to investigate how our algorithm would behave in real world scenarios [2,3,18].

References

[1] *ASH Transceiver Designer's Guide*, http://www.rfm.com (2002).

[2] A. Boukerche and S. Nikoletseas, Protocols for Data Propagation in Wireless Sensor Networks, in: *Wireless Communications Systems and Networks*, ed. M. Guizani (Kluwer Academics, 2004) pp. 23–51.

[3] A. Boukerche, R.W.N. Pazzi and R. Araujo, A supporting protocol to periodic, event-driven and query-based application scenarios for critical condition surveillance, in: *ALGOSENS* (2004).

[4] B. Chen, K. Jamieson, H. Balakrishnan and R. Morris, Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks, in: *Proceedings ACM SIGMOBILE Annual International Conference on Mobile Computing and Networking* (July 2001) pp. 85–96.

[5] D. Estrin, D. Culler, K. Pister and G. Sukhatme, Connecting the physical world with pervasive networks, IEEE Pervasive Computing (2002) 59–69.

[6] D. Estrin and R. Govindan, Next century challenges: Scalable coordination in sensor networks, in: *Proceedings The International Society for Optical Engineering* (1999) pp. 229–237.

[7] D. Estrin, et al., http://nesl.ee.ucla.edu/tutorials/mobicom02.

[8] M.R. Garey and D.S. Johnson, *Computers and intractability: A guide to the Theory of NP-completeness* (Freeman, San Francisco, CA, 1978).

[9] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin and D. Ganesan, Building efficient wireless sensor networks with low-level naming, in: *Proceedings The Eighteenth ACM Symposium on Operating Systems Principles* (2001) pp. 146–159.

[10] W.R. Heinzelman, A. Chandrakasan and H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, in: *Proceedings Annual Hawaii International Conference on System Sciences* (2000).

[11] J. Hill, A software architecture to support network sensors (Master's Thesis, UC Berkeley, 2000).

[12] J. Hill, R. Szewczyk, A. Woo, S. Hollar and J. Heidemann, System architecture directions for networked sensors, in: *Proceedings International Conference on Architectural Support for Programming Languages and Operating Systems* (November 2000).

[13] C. Intanagonwiwat, R. Govindan and D. Estrin, Directed diffusion: A scalable and robust communication paradigm for sensor networks, in: *Proceedings The Annual international conference on Mobile computing and networking* (2002) pp. 56–67.

[14] B. Krishnamachari, D. Estrin and S. Wicker, Impact of data aggregation in wireless sensor networks, in: *Preceedings The International Conference on Distributed Computing Systems Workshops* (2002) pp. 575–578.

[15] S. Madden, M.J. Franklin and J.M. Hellerstein and W. Hong, TAG: A tiny aggregation service for ad-hoc sensor networks, in: *Proceedings Symposium on Operating Systems Design and Implementation* (2002).

[16] R. Min, M. Bhardwaj, S.-H. Choi, N. Ickes, E. Shih, A. Sinha, A. Wang and A. Chandrakasan, Energy-centric enabling technologies for wireless sensor networks, IEEE Wireless Communications (August 2002) 28–39.

[17] A. Nasipuri and K. Li, A directionality based location discovery scheme for wireless sensor networks, in: *Proceedings The First ACM International Workshop on Wireless Sensor Networks and Applications* (2002) pp. 105–111.

[18] S. Nikoletseas et al., A sleep-awake protocol for information propagation in smart dust networks, in: *Proceedings 3rd Workshop on Mobile and Ad-Hoc Networks* p. 225-.

[19] NRL's Sensor Network Extension to ns-2, http://nrlsensorsim.pf.itd.nrl.navy.mil/.

[20] C.E. Perkins, Ad Hoc On Demand Distance Vector (AODV) Routing, *IEFT Internet Draft*, available at: `http://www.ieft.org/internet-drafts/draft-ietf-manet-aodv-02.txt`.

[21] V. Raghunathan, C. Schurgers, S. Park and M. Srivastava, Energy-aware wireless sensor networks, IEEE Signal Processing 19 (2002) 40–50.

[22] C. Schurgers, V. Tsiatsis, S. Ganeriwal and M. Srivastava, Optimizing sensor networks in the energy-latency-density design space, IEEE Transactions on Mobile Computing 1(1) (2002) 70–80.

[23] K. Sohrabi, J. Gao, V. Ailawadhi and G. Pottie, Protocols for self-organization of a wireless sensor network, IEEE Personal Communications Magazine 7(5) (2000) 16–27.

[24] The Network Simulator ns-2: Documentation, http://www.isi.edu/nsnam/ns.

[25] Y. Xu, J. Heidemann and D. Estrin, Geography-informed energy conservation for ad hoc routing, in: *Proceedings ACM SIGMOBILE Annual International Conference on Mobile Computing and Networking* (Rome, Italy, July 2001) pp. 70–84.

[26] W. Ye, J. Heidemann and D. Estrin, An energy-efficient MAC protocol for wireless sensor networks, in: *Proceedings IEEE INFOCOM* (2002) pp. 1567–1576.

Azzedine Boukerche is a Full Professor and holds a Canada Research Chair Position at the University of Ottawa. He is also the Founding Director of PARADISE Research Laboratory at Ottawa U. Prior to this, he hold a faculty position at the University of North Texas, USA, and he was working as a Senior Scientist at the Simulation Sciences Division, Metron Corporation located in San Diego. He was also employed as a Faculty at the School of Computer Science McGill University, and taught at Polytechnic of Montreal. He spent a year at the JPL-California Institute of Technology where he contributed to a project centered about the specification and verification of the software used to control interplanetary spacecraft operated by JPL/NASA Laboratory.

His current research interests include wireless networks, mobile and pervasive computing, wireless multimedia, QoS service provisioning, wireless ad hoc and sensor networks, distributed systems, distributed computing, large-scale distributed interactive simulation, and performance modeling. Dr. Boukerche has published several research papers in these areas. He was the recipient of the best research paper award at PADS'97, and the recipient of the 3rd National Award for Telecommunication Software 1999 for his work on a distributed security systems on mobile phone operations, and has been nominated for the best paper award at the IEEE/ACM PADS'99, and at ACM MSWiM 2001. Dr. A. Boukerche serves as an Associate Editor and on the editorial board for ACM/Springer Wireless Networks, the Journal of Parallel and Distributed Computing, The Wiley Journal of Wireless Communication and Mobile Computing. He served as a Founding and General Chair of the first Int'l Conference on Quality of Service for Wireless/Wired Heterogeneous Networks (QShine 2004), ACM/IEEE MASCOST 1998, IEEE DS-RT 1999-2000, ACM MSWiM 2000; Program Chair for ACM/IFIPS Europar 2002, IEEE/SCS Annual Simulation Symposium ANNS 2002, ACM WWW'02, IEEE/ACM MASCOTS 2002, IEEE Wireless Local Networks WLN 03-04; IEEE WMAN 04-05, ACM MSWiM 98–99, and TPC member of numerous IEEE and ACM conferences. He served as a Guest Editor for JPDC, and ACM/kluwer Wireless Networks and ACM/Kluwer Mobile Networks Applications, and the Journal of Wireless Communication and Mobile Computing.

Dr. Boukerche serves as a Steering Committee Chair for ACM MSWiM, IEEE DS-RT, and ACM PE-WASUN Conferences.
E-mail: boukerch@site.uottawa.ca

Xiuzhen Cheng is an Assistant Professor in the Department of Computer Science at the George Washington University. She received her MS and Ph.D. degrees in Computer Science from University of Minnesota—Twin Cities in 2000 and 2002, respectively. Her current research interests include localization, data aggregation services, and data storage in sensor networks, routing in mobile ad hoc networks, and approximation algorithm design and analysis. She is a member of the ACM and IEEE.
E-mail: cheng@gwu.edu

Joseph Linus has recently graduated with a MSc Degree from the Department of Computer Sciences, University of North Texas. His current research interests include wireless sensors networks, and mobile ad hoc networks.