Open access • Proceedings Article • DOI:10.1145/940991.941000

Energy-aware data-centric routing in microsensor networks — Source link

Azzedine Boukerche, Xiuzhen Cheng, Joseph Linus

Institutions: University of Ottawa, George Washington University, University of North Texas

Published on: 19 Sep 2003 - Modeling Analysis and Simulation of Wireless and Mobile Systems

Topics: Wireless sensor network, Wireless Routing Protocol, Key distribution in wireless sensor networks, Mobile wireless sensor network and Wireless network

Energy-Aware Data-Centric Routing in Microsensor Networks

Azzedine Boukerche *
SITE, University of Ottawa, Canada
boukerche@site.uottawa.ca

Xiuzhen Cheng
Dept. of Computer Science, George Washington University, USA
cheng@gwu.edu

Joseph Linus
Dept. of Computer Sciences, University of North Texas, USA
jlinus@cs.unt.edu

ABSTRACT

Large-scale wireless sensors are expected to play an increasingly important role in future civilian and military settings where collaborative microsensors could be very effective in monitoring their operations. However, *low power* and *in-network data processing* make data-centric routing in wireless sensor networks a challenging problem.

In this paper, we propose a novel and efficient energy-aware distributed heuristic, which we refer to as EAD, to build a special rooted broadcast tree with many leaves that is used to facilitate data-centric routing in wireless microsensor networks. Our EAD algorithm makes no assumption on local network topology, and is based on residual power. It makes use of a *neighboring broadcast scheduling* and *distributed competition among neighboring nodess*. We discuss the implementation of our scheme, and present an extensive simulation experiments to study the its performance. Our experimental results indicate clearly that our EAD scheme outperforms previous schemes.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Distributed Systems; C.4 [**Performanceof Systems**]: Modeling Techniques;

General Terms

Algorithms, Performance, Experimentation

Keywords

Energy, Routing, Sensors, Network Simulator NS-2, Wireless Neyworks

*Dr. Boukerche's work was supported by the CRC Canada Research Program.

MSWiM'03, September 19, 2003, San Diego, California, USA.

1. INTRODUCTION

Research on wireless microsensor networks has received a great deal of interest in recent years mainly due to its wide applications, such as monitoring (habitat, medical, seismic, contamination transport), surveillance, and pre-warning purposes.

Wireless microsensor networks [18] usually contains thousands or millions of sensors, which are randomly and densely deployed (10 to 20 sensors per m^2). A sensor network provides a global view of the monitored area based on local observations measured by each sensor. Sensors are powered by battery, which is impossible to get recharged after deployement. 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 in wireless sensor networks.

Within a sensor the dominant energy consumer is the radio transceiver [11]. For a sensor network with 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 wakeup when some events are detected. Thus, we can't turn off all sensors at the same time in a sensor network. Some sensors must be active for traffic relaying.

In this paper, we propose to construct a virtual backbone, which contains all active sensors, to assist energy-aware routing. All sensors not in the virtual backbone turn off their radios in order to conserve thier 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 [5], GAF [22] and LEACH [14]. While SPAN and GAF are promising methods, they may not be suitable for dense microsensor networks, mainly 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. 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 an NP-hard problem, since it is equivalent to minimum connected dominating set [12]. The reduced topology by all non-leaf nodes forms the virtual backbone. We present a novel Energy-Aware Data-centric routing heuristic, which we refer to as EAD, that exhibits a low message overhead. EAD computes a broadcast tree approximating optimal spanning tree with 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 higher residual power have higher chance to be non-leaf nodes. EAD follows the energy-aware paradigm and results in a special rooted broadcast tree, which is designed intentionally for data-centric routing.

2. NETWORK MODEL

In this paper, we consider wireless microsensor networks for monitoring abnormal events. Example applications include *habitat monitoring* [15, 17], *contamination transport monitoring* [11], *forest fire prewarning* [23], etc. 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 microsensor network to the outside distributed system such as 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 microsensor as *data source* or *event source* since data in a sensor network is generated by sensors, and the gateway as *data sink* or *event sink*.

The architecture of a microsensor [18] contains 4 components: sensing circuitry, digital processing, power supply, and radio transceiver. Among these 4 components, radio transceiver is the dominant power consumer [1, 10, 18]. The energy spent for sensing and data processing is negligible. For example, the power consumed by a Berkeley mote [17] to transmit 1 bit data is equivalent to 800 instructions. For sensors with short transmission range like mote, the energy consumed for different mode (transmit, receive and idle) are comparable, while a sleeping sensor (radio is off) consumes little energy. Figure 1 gives more concrete idea on radio consumption in a typical sensor. Thus to save energy the sensor needs to completely turn off its radio.

3. DATA-CENTRIC ROUTING IN MICROSENSOR NETWORKS

In-networking processing can significantly improve the scalability and lifetime of microsensor networks. At each sensor, the local raw data is first combined with partially processed data delivered from sensors farther away from the sink, and then the aggregated result is transmitted to the sensor 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 outputs based on the aggregation function (SUM, AVG, MEAN, MAX, etc.) from all sensors in the subtree rooted at itself and transmits the aggregated data to its parent. This process is termed *data-centric routing* [13, 15, 16, 17]. Figure 2 gives an example of data-centric routing where the highest temperature needs to be reported to the user.

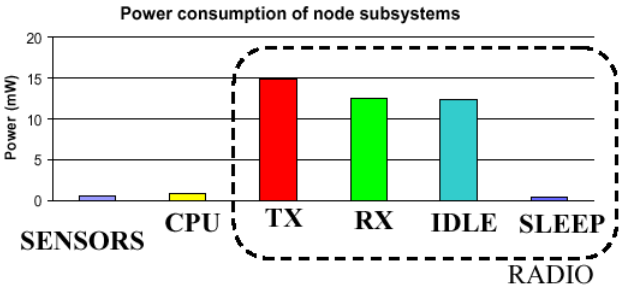

Figure 1: Energy consumption for a typical sensor reported in [11].

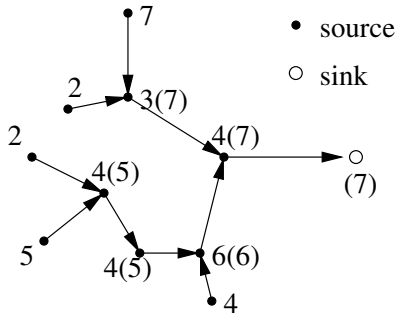

Figure 2: An example to demonstrate data-centric routing. Label $x(y)$ at each node means the local temperature measurement is x while the aggregated value so far is y. The aggregation function is max.

Traditional network-wide routing is termed *address-centric routing* [16]. A packet is routed based on the unique destination IP address and its data payload remains unchanged during the delivery from source to destination. This routing scheme does not work with microsensor networks because of the lackness of globally unique address and the extreme energy constraints for the large volume of raw data. In a microsensor network, data is processed before transmission. Redundant and useless data is discarded. Local data is aggregated to provide globally-effective result. It is possible that an information packet contains different values from hop to hop during the transmission from a leaf to intermediate sensors then to the sink in the tree, because each intermediate sensor may aggregate multiple packets. In this aspect, microsensor network is the pure *peer-to-peer* network.

The reversed multicast tree construction for data-centric routing is determined by the following application scenarios of microsensor networks: *periodic, event-driven* and *query-based*. Actually a sensor network may support all these 3 kinds of data traffic. For periodic traffic, all sensors report their measurements back to the user once every fixed time interval, as preprogrammed before deployment. This kinds of networks require all sensors to be synchronized (when to turn on their radios) such that in-network processing can be

done at each intermediate sensor to guarantee one broadcast per sensor per time interval. For this kind of application, all 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 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 a Steiner tree containing the sink and all sensors detecting the events, plus relay sensors to bridge the traffic. The number of relay sensors needs to be minimized to decrease the total power consumption. This problem is NP-hard. In query model, routes need to be computed for query and data transmission between sink and the queried sensor. This problem is similar to that in event-driven model, except that query model includes the query, which propagates from the sink to the sources.

In all of these application scenarios, sensors remain in sleep mode most of the time in order to save energy. If all 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 expensive. If only part of the network is involved at a time, as in event-driven and query model, sensors have no idea when an event will happen and when a query will be submitted. Simply turning on all the sensors is a big waste while turning off all of them make the network malfunction 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 collaboratively and quickly respond to spontaneous events and queries. But 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 data-centric routing.

Each sensor is either a leaf or an inner node in the tree. All leaf nodes turn off their radios to save 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 NP-Complete [12]. There exist no efficient heuristics thus we have to seek good approximation algorithms. In this paper, we propose an message-efficient distributed heuristic to build an energy-aware rooted spanning tree with many leaves.

Let's look at how a spanning tree rooted at the sink (a reversed broadcast tree) may help with the existing data dissemination models in literature. The first one we study is *directed diffusion* [15]. An *interest* is broadcasted by the sink first. 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 location information, the interest must be broadcasted globally. This consumes energy and wireless bandwidth. If all the active sensors form a spanning tree rooted at the sink, the dissemination of the interest can be restricted to the non-leaf tree node. If the queried sensor is sleeping, an active neighbor can either activate it directly or store the query until the target sensor wakes up. Another interesting attempt for data-centric routing is described in [16]. This reference describes an event-driven sensor network. All the sensors sensing the same event (within the same event radius) first aggregate the data then transmit the result to the sink. The computation of the transmission path is formed to a *network Steiner tree problem*, which is NP-hard. 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 tens of neighbors, only a few percent of total sensors need to be active at any time and these sensors form a virtual backbone rooted at the sink, ready for query and event dissemination. We are going to propose EAD, a heuristic to build a rooted broadcast tree with many leaves.

4. EAD: THE HEURISTIC TO CONSTRUCT A ROOTED BROADCAST TREE WITH MANY LEAVES

In this section, we give the details of EAD plus a brief description of broadcast tree maintenance.

4.1 EAD description

We assume each sensor has its radio transceiver on and is sensing the common channel when the network is initially deployed. We also assume that all the sensors have the same transmission range. In other words, we only consider symmetric links. The control message contains 4 fields: $type, level, parent, power$. Let v be the sender of the message. Then $type_v$ indicates the status of v: 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 radio transceiver already on. The heuristic is sketched in Figure 3.

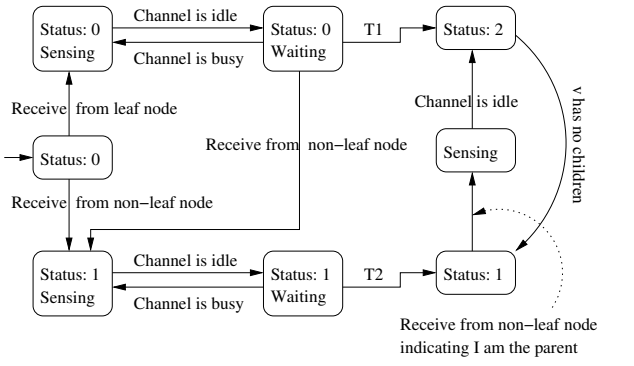

Figure 3: State diagram for the proposed heuristic run by any node v other than sink.

Initially each sensor v has status 0. Sink first broadcasts $msg(2, 0, NULL, \infty)$, where ∞ indicates that the sender is the sink. When any 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 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 T_1^v and T_2^v be monotonically decreasing functions of E_v, the residual power of v. The basic idea is to force the neighboring sensors with higher energy broadcast earlier than those with 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 node* by T_1. The intuition behind the algorithm is stated below. After a sensor u announces its status 2 (non-leaf node) through broadcast, all 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 broadcastings, They start to compete with each other. The winner are those with highest residual energy among its neighboring competitors (thus with smallest T_1 among its neighboring competitors). Figure 4 gives an illustrative example. In Figure 4(i) The original sensor network topology is given. Each sensor is labeled 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 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. Latter, the neighbors of the winners (not in the tree) join the tree as children of its corresponding winner. Their (the winners) 2-hop neighbors (not in the tree) form 4 neighboring competing groups. Figure 4(iv) Repeats (iii) to get the final broadcast tree with many leaves. Note that 2 winners (triangles) in (iii) become leaf nodes in the final tree event hough they the winner in (iii), since they have no children.

Note that EAD grows a broadcast tree from the sink. After the algorithm is done, all the leaf nodes can turn off their radios to save energy. They switch to "power-on" periodically or when some events are detected. This heuristic takes linear number of messages. Actually each node broadcasts at most twice the induced graph by all non-leaf nodes form the virtual backbone. Due to the broadcast nature of wireless communication, 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 also implies an algorithm for robust and efficient broadcast. It is stated below: 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.

4.2 Maintaining the broadcast tree

Our strategy extensively explores *the dense connectivity* of sensor networks.The maintainance of the tree is taken care of using a strategy similer to the one described in [14].The maintainance of the tree becomes important for two reasons. First,the non-leaf nodes may die thus orphaning all the child nodes which are transmitting data to that non-leaf node.Secondly,the non-leaf nodes which form the backbone tree have to stay awake all the time.This induces a huge energy drain on them as compared to the leaf nodes which are awake only occassionaly.This leads to fatigue in the non-leaf nodes.To impose an fairly similar energy demand on all nodes and to distribute to the work on the nodes the algorithm is executed in rounds .The technique effectively used in [14] takes care of node fatigue and orphaned nodes.The initial pahse of the algorithm is the "initialization" phase where the nodes execute the EAD to elect the non-leaf nodes and set up the backbone. Once that is over the nodes proceed to "data-transmit" phase.In this phase the nodes transmit the data to the sink.Together the initialization phase and the data-transmit phase constitute a round.The round ends at the end of a data-transmit phase. When one round ends, the initialization for the next round begins and dead nodes and orphaned nodes are thus taken care of. The algorithm is effective since the initialization phase is significantly small compared to the data-transmission phase. When a non-leaf node effectively dies,its leaf nodes will have to resort to direct transmission to the sink which is costly.But this condition is only till the next leaf rotation round.As soon as the data-transmit state for current round expires,the sink will initiate a new round initialization by re-construction of the tree.This takes care of fatigue in the non-leaf nodes which have to stay awake all the time.

5. OPTIMIZATION TO LARGE SCALE DENSE SENSOR NETWORKS

EAD is efficient when the network size is small. For large-scale sensor networks, EAD may take too much time since the execution process is propagated from the sink to the whole network. We propose in this section two approaches to

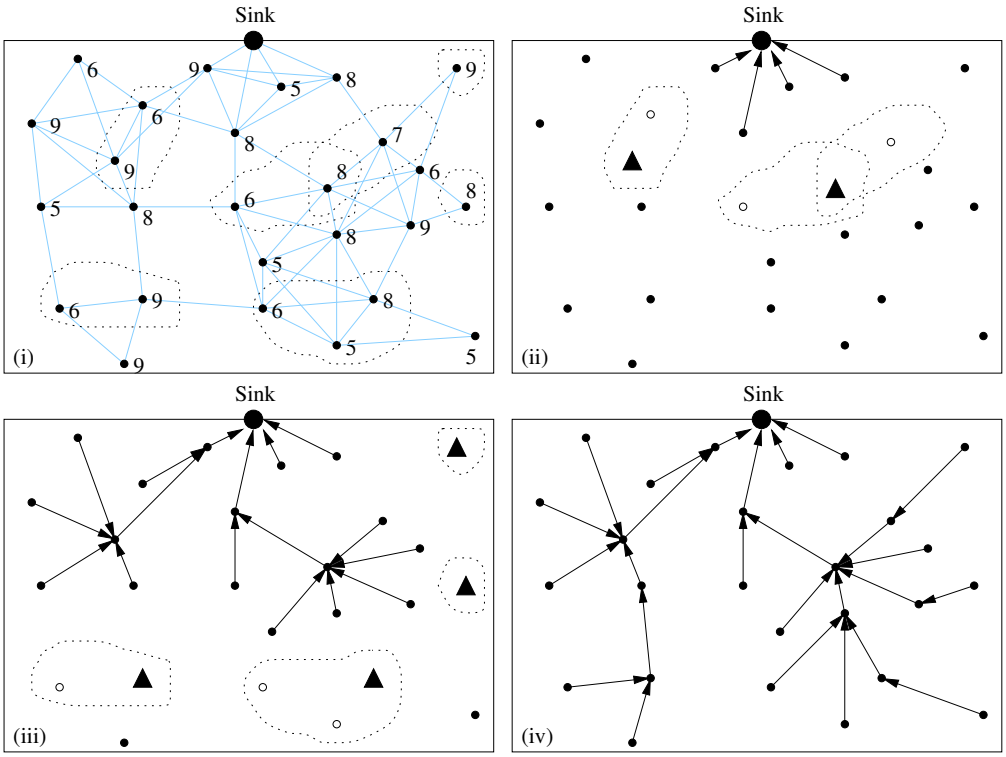

Figure 4: An illustrative example

"pre-process the network topology". The idea is to turn off the radios of some sensors such that only a subset of sensors attend the execution of EAD. Our approaches guarantee a rooted broadcast tree spanning all sensors even though only a subset of them execute EAD.

5.1 Position-based approach

Assume each sensor knows its position. Wattenhofer *et. al.* in [21] have proved that if every node u has at least one active neighbor in each direction α, where $\alpha \leq 120°$, then the topology is connected. Based on this claim, we propose the following heuristic.

We assume initially all sensors are in sleep mode. A sensor wakes up randomly (once every T_0 time units) and broadcasts a hello message containing its own position. An active sensor u hearing a hello message from v determines whether v resides in one of its expected directions or not. u replies with a message containing its position and an $INVI$ bit. If there is no active neighbor in the direction where v resides, $INVI = 1$; Otherwise, $INVI = 0$. If v receives a reply with $INVI$ bit on, or detects that it has no neighbor in at least one direction (based on received message), v remains on. Otherwise, v returns back to sleep. After some time, the network enters an equilibrium state. After T_0, we apply EAD over all active sensors to build a broadcast tree rooted at the sink. Those sleeping sensors wake up periodically (with period T_0) to determine its parent, the active neighbor with highest energy. If an active node in the tree detects no children after T_0 time units, it turns off its transceiver and goes back to sleep.

5.2 Topology-based approach

Note that the position-based approach described in Subsection 5.1 depends on the precise node position. However, location discovery in dense microsensor networks is very challenging [19, 20]. Existing algorithms achieve results with errors within several meters [19]. For dense sensor networks with transmission range less than 10 meters, this error range is not tolerable. In this subsection, we propose a topology-based approach.

How to determine which sensor should be active if no position information is available? Let's consider the number of active neighbors in each direction α again, as in Subsection 5.1. Assume 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 $n_1, n_2, \cdots, n_k$ follow the multinomial distribution:

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

where $p_{n_1,n_2,\cdots,n_k}$ 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

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

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

Note that if each sensor has 4 or 5 active neighbors, then with probability around 50%, it has one neighbor in each

$k \diagdown n$	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

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

Simulation time	4500 seconds.
Starting Energy for each node	2J
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

Table 2: Simulation Parameters

direction if $k = 3$. Based on this observation, we propose our algorithm as follows. Note that we assume initially all sensors have power off.

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. Not that with this approach, we can not guarantee a tree spanning all 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.

6. SIMULATION EXPERIMENTS

The experiments were carried out using the Network Simulator ns-2 [24]. In order to evaluate the performance of EAD, we choose the following metrics:

• **Total number of active nodes:** indicates the node failures due to low energy with passing time.The failure of a node is characterized by inability to generate packets that meet or exceed a thresh hold value (CSThresh). Efficient routing algorithms should have enough nodes alive throughout the simulation time to send data to base station.

• **Throughput:** shows the volume of data transmitted to the sink.The throughput of the algorithm shows the efficiency of the algorithm to deliver data to the sink.The primary task of the algorithm is to deliver data to the sink from the leaf nodes as efficiently as possible.We use this metric to evaluate the data throughput achieved by EAD.

• **Energy expended:** measures of the total energy expended by the network as a whole up to that point in time during simulation.Energy expended is an important parameter in evaluating the effectiveness of our algorithm. This metric shows the power saving capability of the algorithm.

Let us now turn to our reslts.

Figures 5-7 show the number of nodes alive plotted against simulation time when the network contains 50, 150 and 200 nodes respectively. The amount of energy per node is 2 J at the beginning. As we can see from the figures, the number of nodes alive decreases after some simulation time. As non-leaf nodes that fail the load 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 5 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 similer 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 routs the data packets to the sink by multihop routing as opposed to LEACH where the clusterheads have to tranmit 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 up one level from it.

Figures 8-10 portray the total energy dissipated vs simulation time for the same set of network topologies. This is a limited energy supply and the amount of energy per node is 2 J. 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 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 invloved 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 makes it more efficient than LEACH overall.

Figures 11-13 illustrate total data throughput to base station plotted against simulation time. In this case we continue to consider the networks with 50, 150 and 200 number

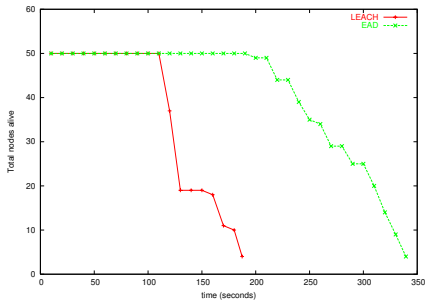

Figure 5: EAD- 50 Nodes

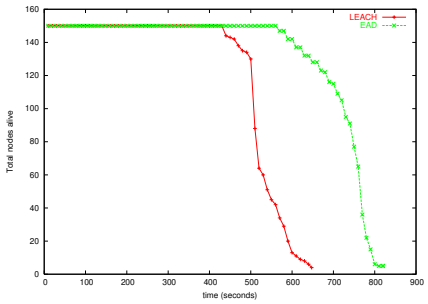

Figure 6: EAD- 150 Nodes

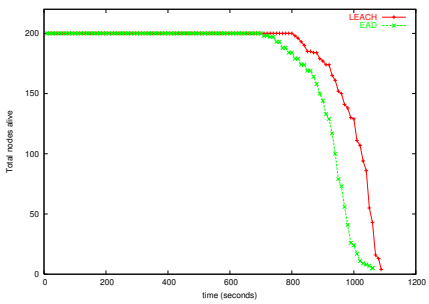

Figure 7: EAD- 200 Nodes

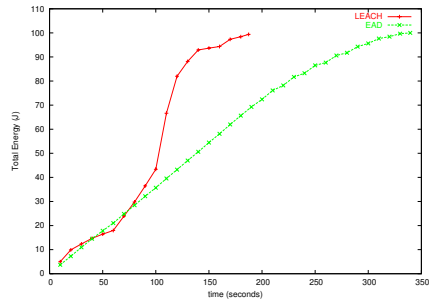

Figure 8: EAD- 50 Nodes

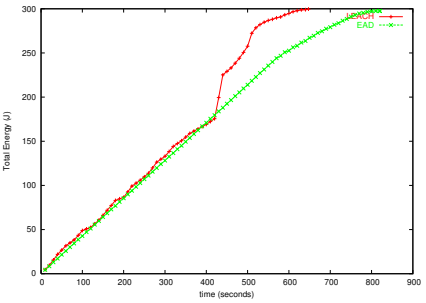

Figure 9: EAD- 150 Nodes

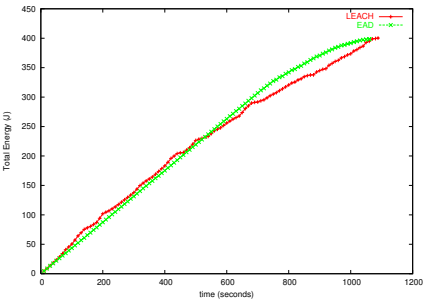

Figure 10: EAD- 200 Nodes

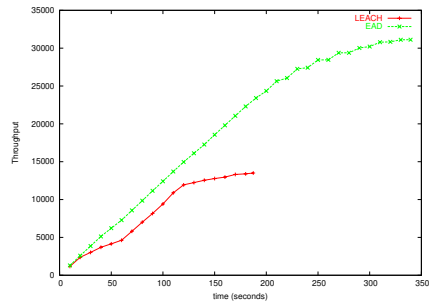

Figure 11: EAD- 50 Nodes

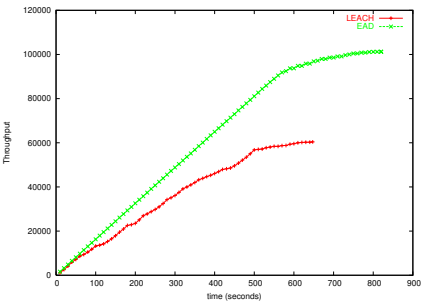

Figure 12: EAD- 150 Nodes

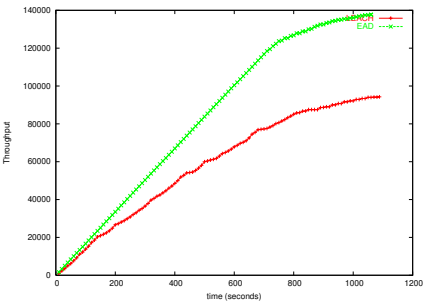

Figure 13: EAD- 200 Nodes

of nodes respectively. The amount of energy per node is still 2 J. The throughput to the sink is cumulative and steadily increases with the simulation time. The gradual flattening of the curve towards the end of simulation is due to the fact that nodes are failing with passing simulation time. From graph 12 we can see that the throughput rate slows down after 300 seconds. Similar behavior is seen in other graphs also. This is because of the fact that the number of live nodes is significantly low, and this lowers the throughput. The throughput growth from 30000 to 35000 data signals in figure 12 is significantly slow for this reason.

7. CONCLUSIONS

In this paper we have proposed an efficient Energy-Aware Data-centric routing heuristic, to build a broadcast tree rooted at gateway to facilitate data-centric routing in dense wireless microsensor networks. EAD computes a tree with many leaves. With the transceivers of all leaf nodes being turned off, the network lifetime can be greatly extended. Simulation study shows that EAD performs very well.

As a future work, we intend to extend our simulation experiments to take the residual power of parent nodes into consideration, thereby improving the averaged power over all active neighbors.

8. REFERENCES

[1] "ASH Transceiver Designer's guide", http://www.rfm.com, 2002

[2] K.M. Alzoubi, P.-J. Wan and O. Frieder, New distributed algorithm for connected dominating set in wireless ad hoc networks, *Proc. 35th Hawaii International Conference on System Sciences*, pp. 3881-3887, 2002.

[3] A. Boukerche and S. Nikoletseas, Protocols For Data Propagation In Wireless Sensor Networks, Submitted, 2003.

[4] A. Cerpa and D. Estrin, ASCENT: adaptive self-configuring sensor networks topologies, *IEEE INFOCOM'02*, pp. 1278-1287, 2002.

[5] B. Chen, K. Jamieson, H. Balakrishnan and R. Morris, Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks, *MobiCom 2001*, pp. 85-96, July 2001.

[6] X. Cheng and D.-Z. Du, Virtual Backbone-Based Routing in Multihop Ad Hoc Wireless Networks, *submitted*, 2001.

[7] X. Cheng, X. Huang, D. Li and D.-Z. Du, Polynomial-time approximation scheme for minimum connected dominating set in ad hoc wireless networks, to appear in *Networks*.

[8] M. Cadei, X. Cheng and D.-Z. Du, Connected domination in ad hoc wireless networks, in *Proc. 6th International Conference on Computer Science and Informatics*.

[9] S. Butenko, X. Cheng, D.-Z. Du and P. Pardalos, On the construction of virtual backbone for ad hoc wireless networks, in *2nd Conference on Cooperative Control and Optimization*, 2001.

[10] D. Estrin and R. Govindan, Next century challenges: scalable coordination in sensor networks, *SPIE*, pp. 229-237, 1999.

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

[12] M.R. Garey and D.S. Johnson, Computers and intractability: a guide to the theory of NP-completeness, *Freeman, San Francisco*, 1978.

[13] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin and D. Ganesan, Building efficient wireless sensor networks with low-level naming, Proceedings of the eighteenth ACM Symposium on Operating Systems Principles (SOSP'01), pp. 146-159, 2001.

[14] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, *HICSS'00*, 2000.

[15] C. Intanagonwiwat, R. Govindan and D. Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, Proceedings of the *6th annual international conference on Mobile computing and networking (MobiCom'02)*, pp. 56-67, 2000

[16] B. Krishnamachari, D. Estrin and S. Wicker, Impact of Data Aggregation in Wireless Sensor Networks, Preceedings of the *22nd International Conference on Distributed Computing Systems Workshops (ICDCSW'02)*, pp. 575-578, 2002.

[17] S. Madden, M. J. Franklin and J.M. Hellerstein and W. Hong, TAG: a tiny aggregation service for ad-hoc sensor networks, to appear in *OSDI* 2002.

[18] 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*, pp. 28-39, August 2002.

[19] A. Nasipuri and K. Li, A directionality based location discovery scheme for wireless sensor networks, *WSNA'02*, pp. 105-111, 2002.

[20] A. Savvides, C.-C. Han and M.B. Strivastava, Dynamic fine-grained localization in ad-hoc networks of sensors, *ACM Sigmobile*, pp. 166-179, 2001.

[21] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang, Distributed topology control for power efficient operation in multihop wireless ad hoc networks. *INFOCOM 2002*, Vol. 3, pp. 1388-1397, 2001.

[22] Y. Xu, J. Heidemann and D. Estrin, Geography-informed energy conservation for ad hoc routing, *MobiCom 2001*, Rome, Italy, pp. 70-84, July 2001.

[23] W. Ye, J. Heidemann and D. Estrin, An energy-efficient MAC protocol for wireless sensor networks, *INFOCOM 2002*, Vol. 3, pp. 1567-1576.

[24] The Network Simulator ns-2 Documentation. www.isi.edu/nsman/ns