Aggregation Tree Construction in Sensor Networks

Min Ding Xiuzhen Cheng Guoliang Xue

Abstract— **Large-scale wireless sensor networks are expected to play an increasingly important role in future civilian and military settings. 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 heuristics to construct and maintain an aggregation tree in sensor networks. This aggregation tree can be used to facilitate data-centric routing. The main idea is to turn off the radio of all leaf nodes to save power, and thereby extending the network lifetime. Therefore, in order to save the number of broadcasting messages, only the non-leaf nodes in the tree are in charge of data aggregation and traffic relaying.**

In this paper, we propose an efficient energy-aware distributed heuristic to generate the aggregation tree, which we refer to as EADAT. Our EADAT algorithm makes no assumption on local network topology, and is based on residual power. It makes use of *neighboring broadcast scheduling* and *distributed competition among neighbors*. These novel concepts make EADAT very efficient and effective, as demonstrated by our simulation experiments with *NS2*.

I. Introduction

The technologies of sensing, on-board processing, and wireless communication have made smart sensors in very small scale available and the research on wireless sensor networks has received a great deal of interest in recent years [2], [10]. Distributed networks of thousands of collaborative sensors promise long-lived and unattended systems for monitoring (habitat, medical, seismic, contamination transport, etc), surveillance, and pre-warning purposes. A sensor network provides a global view of the monitored area based on local observations measured by each sensor.

Wireless microsensor networks [3], [2], [11] usually contains thousands or millions of sensors, which are randomly and densely deployed. Sensor networks have short transmission range and low data rate. Each sensor has a light weight, a low cost, but very limited energy. Nevertheless, sensor networks are designed to have long operation time.

Within a sensor, the dominant energy consumer is the radio transceiver [11]. For a sensor network of short transmission range, the radio consumes almost the same amount of energy in transmit, receive and idle mode [11]. Therefore the only way to save energy is to completely turn off the radio, if possible. However, a sleeping sensor cannot function as a relay, even though it can continue sensing and it can wake up when some events are detected. Thus, we cannot turn off all sensors at the same time in a sensor network. There must exist some active sensors in the network at any time for traffic relaying. In this paper, we are going to consider a tree structure, termed *aggregation tree*, such that all non-leaf sensors in the tree will be active and all leaf sensors will turn off their radios. Let p be the percentage of active nodes, then roughly speaking, the network lifetime can be $\frac{1-p}{p}$ times longer if only active nodes have power on.

Sensor networks are featured by multihop routing and strict resource limitation, as the transmission range of a sensor is finite and the sensors are powered by battery, which is almost impossible to be recharged or replaced after deployment. The multihop nature of a sensor network exacerbates the energy shortage problem. Extending network lifetime by conserving energy is a very challenging problem [10], [11]. In this paper, we are going to tackle this problem by considering *energy-aware data-centric routing*. Next we first provide some background information.

II. Network model

In this paper, we consider wireless microsensor networks for monitoring abnormal events. Example applications include *habitat monitoring* [8], [10], *contamination transport monitoring* [4], *forest fire pre-warning* [15], 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 [11] contains 4 components: sensing circuitry, digital processing, power supply, and radio transceiver. Among these 4 components, radio transceiver is the dominant power consumer [1], [3], [11], [12], [14]. The energy spent for sensing and data processing is negligible. For example, the power consumed by a Berkeley mote [10] to transmit 1 bit data is equivalent to 800 instructions [6], [7]. For sensors with short transmission range like mote, the energy consumed for different mode (transmit, receive and idle) are comparable [13], 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.

III. Data-centric routing in microsensor networks

There are two kinds of dominant traffics in a sensor network: queries from the user to the network and data from sensors to the user. Each sensor acts as a "plain sensor" to sense the environment and a router to relay traffic for others. The amount of data generated by one sensor can

Department of Computer Science, the George Washington University, Washington, DC 20052, USA. Email: {minding,cheng}@gwu.edu.

Department of Computer Science and Engineering, Arizona State University, Tempe, AZ 85287, USA. Email: xue@asu.edu.

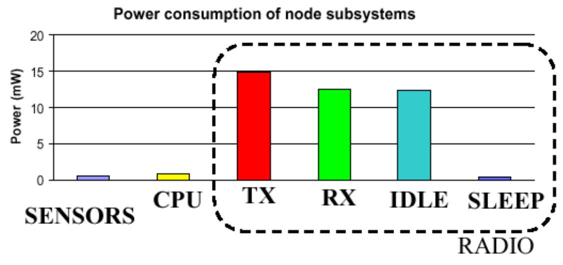

Fig. 1. Energy consumption for a typical sensor reported in [4].

be large enough to block the whole network. And a large part of these data is useless to the end user. Thus data is pre-processed before they are transmitted. This is referred to as *in-network processing* [10], during which redundant, useless and spurious data are deleted, and partial observations from different sensors are combined and aggregated.

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* [5], [8], [9], [10]. Figure 2 gives an example of data-centric routing where the highest temperature needs to be reported to the user.

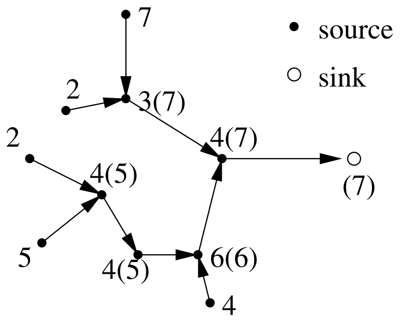

Fig. 2. An example to demonstrate data-centric routing. Label $x(y)$ at each node means the local temperature measurement is x while the aggregated value so far is y. The aggregation function is *max*.

IV. Aggregation tree construction

In this section, we propose heuristics to construct and maintain an aggregation tree. We also show how this aggregation tree will provide help to existing data dissemination models in literature.

A. Constructing aggregation tree

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. Note that we do not require globally unique ID for each sensor, but we do assume that neighboring sensors have different IDs. The control message, denoted by msg, contains 5 fields: $ID, parent, power, status, hopCnt$, indicating the sensor's ID, its parent in the aggregation tree, its residual power, it's status in the tree (undefined state, leaf node, non-leaf node, or danger state when non-leaf node does not have enough power to serve as the active node), and the path length (number of hops from the sink). For easier elaboration, we attach subscript $_v$ to each field, if the sender of the message is sensor v. Note that if the residual power is unavailable, we can use the difference between the expected lifetime of the battery and the total time with radio transceiver already on.

There is a timer T_v associated with each sensor v. The initial value of T_v, denoted by T_v^0, is a monotonically decreasing function of residual power p_v. In other words, the bigger the residual power, the smaller the value of the timer, the shorter the waiting time. We define $T_v^0 = \frac{1}{power_v}$. Ties are broken by ID.

The algorithm is initiated by the sink s, which broadcasts $msg(ID_s, -, \infty, status_s, 0)$. Here, we assume sink has infinite power supply and it is the root of the aggregation tree. After receiving an msg the first time, a sensor v who is listening to the channel will set its timer to T_v^0. T_v will count down only if the channel is idle, and it will be reset to T_v^0 if v receives any msg before T_v times out. During this process, v records the parent with higher residual power and shorter path to the sink. When T_v times out, v broadcasts $msg(ID_v, parent_v, power_v, status_v, hopCnt_v)$, where $hopCnt_v = 1 + hopCnt_{parent_v}$. If a node u receives a message from v indicating that $parent_v = u$, u will mark itself a non-leaf tree node; otherwise, u is a leaf node. This process continues until each sensor broadcasts once. The result is an aggregation tree or a reversed multicast tree rooted at the sink.

This heuristic has several nice features. First, sensors with higher residual power have higher chance to be non-leaf tree node. Second, residual power is used to distributively schedule the local broadcasts among neighboring sensors. Node with higher residual power will broadcast earlier, based on T_v^0. Third, neighboring sensors compete with each other in a distributed manner. The winner will grasp the channel and broadcast the control packets. Forth, if we let each sensor selects two parents (all are based on residual powers) whenever possible, with no change to the original protocol we can construct an aggregation tree (actually a DAG structure) with higher survivability. Lastly, since msg is quite short, we may transmit multiple copies in one burst to overcome the unreliable wireless links with little extra power consumption. Note that during the aggregation tree construction, EADAT only requires each sensor to broadcast once.

B. Maintaining the aggregation tree

The tree can be re-constructed periodically from the sink. Here we very briefly propose the heuristic to maintain the tree. When its residual power is below some threshold P_{th}, an active sensor periodically broadcasts help messages for T_d time units and then shuts down its radio. After receiving the first help message from its parent, an active node switches to a new parent in the original tree, if it exists; Otherwise, it turns into a danger state. A sleeping sensor periodically wakes up and broadcasts hello message, which contains its path length to the sink. A danger node receiving a hello message from a neighbor u with shorter distance to the sink in the tree invites u to join the tree.

C. Applying the Aggregation Tree.

Let's take a look at how an aggregation 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* [8]. In this model, 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 large amount of energy and wireless bandwidth. With the aggregation tree, 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 [9]. 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 an aggregation tree can be used to relay the aggregated result to the sink.

V. Performance evaluation of EADAT

The following simulations based on *NS2* platform are used to study the effects of EADAT. Compared with the sensor network routing method without aggregation tree, we show that EADAT extends the network lifetime and conserves more energy. Then we briefly analyze the reason why EADAT increases the network performance. Package delivery ratio along the network lifetime is also illustrated. Further more, the network lifetimes with different sensor densities are compared. At last, we show the EADAT protocol overheads under different environment conditions.

A. Methodology for simulation

EADAT is implemented on the platform of ns-2.26. The sensor working field is a square of 160m by 160m. In order to examine the network performance vs. sensor density, we vary the number of sensors from 100 to 200 and fix the size of the working area. For all simulation scenarios, nine traffic nodes are randomly selected as event sources. The sink is located in the boundary of the area. Traffic nodes randomly send packages with constant bit rate (CBR) to the sink. Packet size is 64 bytes. Package rate is either 1 pkt/s or 2 pkts/s. The sensor's energy settings are similar to those used in direct diffusion [8]. To be specific, we choose $14J$ as the sensor initial energy value, $0.66W$ as the transmit power, $0.395W$ as the receive power, and $0.035W$ as the idle power. We assume a sensor consumes no energy when in sleep mode. Each sensor has a radio range of 40m. We use AODV as the default routing protocol.

B. Extending network lifetime by EADAT

We first study how EADAT affects the network lifetime. Here, we simply count the number of alive sensors when simulation time flies. We generate 100 sensors which are randomly located in the square, and the packet rate is 1 pkt/s. we compare the case when aggregation tree is applied and the case when it is not. In Figure 3, the solid line is the number of surviving sensors at each time step using our EADAT algorithm, and the dotted line is the number of alive sensors without aggregation tree. When no data aggregation tree is introduced, most of the sensors die suddenly at around 370 second. This is because sensors are busy in transmitting and receiving all the time when 9 event sources are located randomly in the monitored area. With an aggregation tree, more senors can survive much longer time. Figure 3 shows that about 65 sensors are still alive after 750 seconds.

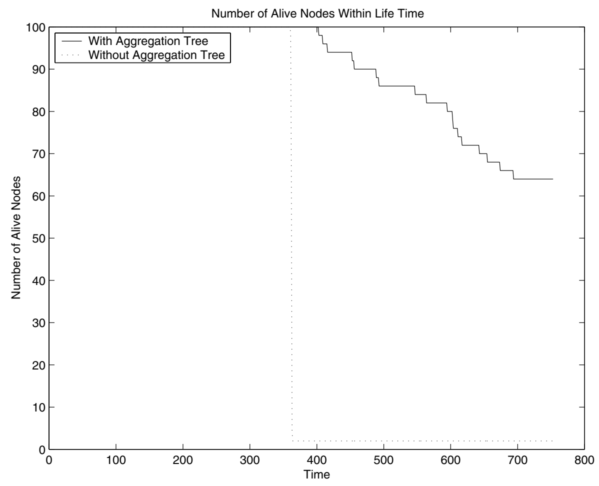

Fig. 3. An example to demonstrate the number of alive nodes in network lifetime: with EADAT vs. without EADAT construction.

C. Conserving energy by EADAT

To keep more sensors alive, EADAT protocol conserves more energy than without aggregation tree under the same network condition. Energy saving actually enlarges the network lifetime. Our algorithm only needs to activate the non-leaf nodes in the aggregation tree to maintain network traffic. All leaf nodes are turned off to save energy. Figure 4 demonstrates clearly that the average residue energy of all alive sensors when EADAT is applied decreases much slowly than that of case when no aggregation tree is introduced.

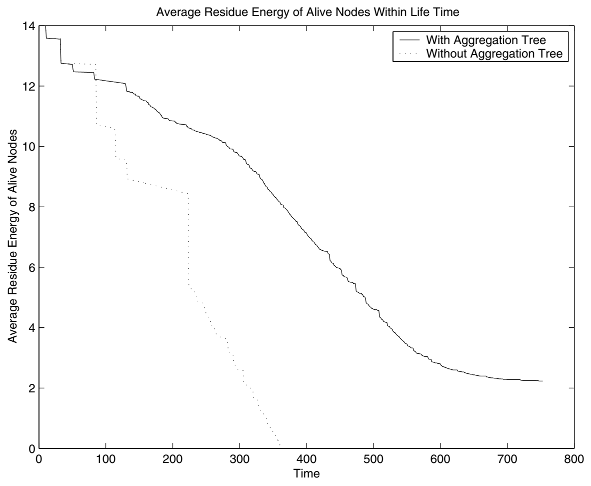

Fig. 4. An example to demonstrate the residue energy comparison in network lifetime: with EADAT vs. without aggregation tree construction.

D. Network efficiency effects by EADAT

Besides examining the network lifetime extension roughly via the number of survival sensors and energy saving, We also evaluate the network efficiency influenced by EADAT. Here, we measure the efficiency in term of data delivery ratio which is defined as the number of received packets divided by the number of sent packets for a certain time period. From our simulation results illustrated in Figure 5, we find that this ratio does not change much while the network is alive. It shows the stable performance of our protocol. When the network energy is running out, the data delivery ratio collapses rapidly. This phenomenon probably can be taken as a sign of the network death. In other words, in our simulation settings, the network dies after 850 seconds.

E. Network lifetime effects with network density

By considering the changes of network density, we also study the relationship between the network lifetime and the network density λ^1. In this experiment, the value of network lifetime is roughly measured as the time when network data delivery ratio collapses. For more convenient comparison, we take the network lifetime ratio as the benchmark. The network lifetime with 100 sensors is considered 1 unit in figure 6.

[1]The network density is computed via the equation:

$$\lambda = \frac{N\pi R^2}{A},$$

where N is the sensor number, R is the sensor range and A is the area of sensor field.

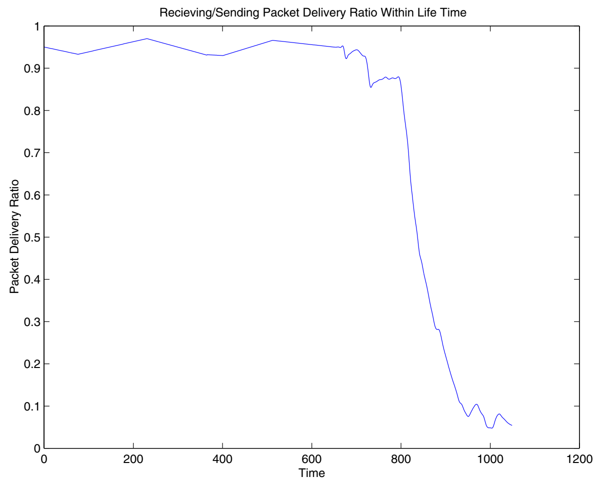

Fig. 5. The data delivery ratio changes along with the network lifetime when EADAT is employed as the routing algorithm. The death of the network is shown as the collapse of the ratio of received packets and sent packets.

Illustrated by figure 6, the higher the network density, the longer the network survived. Note that the network lifetime increases super-linearly as density increases. This is because all active nodes computed by EADAT reflect a "coverage" of the monitored area, which means that current active nodes are enough to cover the sensor area. Thus, with the increase of network density, more leaf nodes join the tree and more sensors have the chance to be turned off to save energy. In this simulation, we use 100, 120, 140, 160, 180, and 200 sensors respectively, to increase the sensor density from 21.8 to 43.6. Packet rate is set to 2 pkt/s.

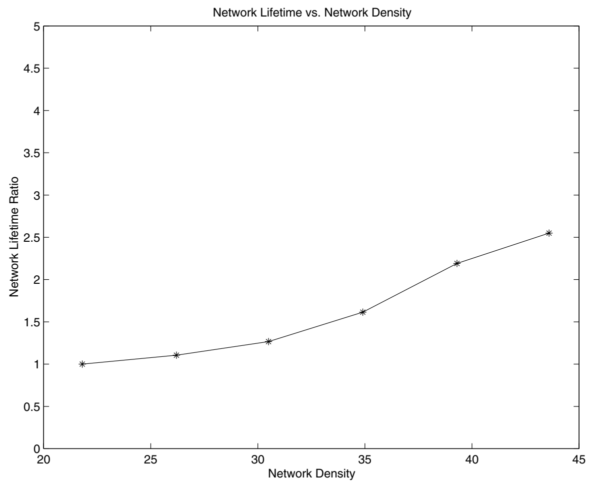

Fig. 6. The network lifetime ratio changes along with the increase of the network sensor density.

F. Overhead effects by EADAT

In EADAT, we maintain a tree structure routing to choose active non-leaf nodes. The aggregation tree construction and maintenance also cause some traffic load and consume energy. In order to study the EADAT protocol overhead, we measure the energy used by EADAT control messages and compare it with the total system energy usage. The ratio vs. density relation is reported in Figure 7. According to our simulation, the protocol overhead almost keeps as a small and stable value which is around 1% of the total system energy with the density range from 21.8 to 43.6. This means that EADAT has good scalability in terms of energy saving.

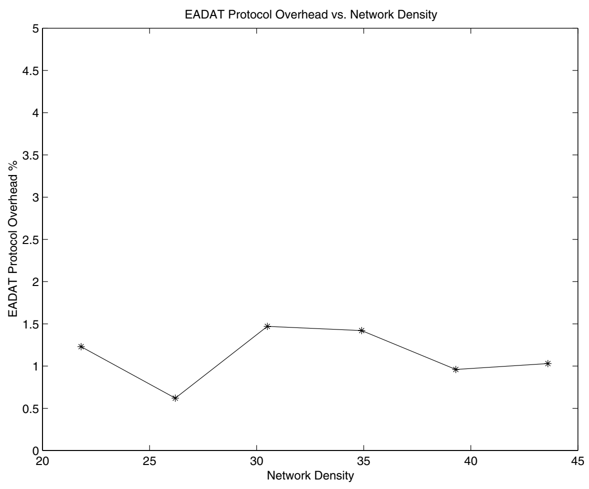

Fig. 7. The EADAT protocol overhead almost keeps constant around 1% along the increase of network density when EADAT algorithm is employed as the routing algorithm.

VI. Conclusion and discussion

In this paper, we have proposed heuristics to construct and maintain an energy-aware aggregation tree in sensor networks. Applying this tree structure, network lifetime can be extended by turning off the radios of all leaf nodes, and by restricting the post of query and the dissemination of data along this reversed broadcast tree. We also simulate and analyze the performance of EADAT algorithm in terms of network lifetime, energy saving, data delivery ratio and the protocol overhead. Simulation results show that EADAT performs very well.

References

[1] "ASH Transceiver Designer's guide", http://www.rfm.com, 2002
[2] D. Estrin, D. Culler, K. Pister, and G. Sukhatme, Connecting the physical world with pervasive networks, *IEEE Pervasive Computing*, pp. 59-69, 2002.
[3] D. Estrin and R. Govindan, Next century challenges: scalable coordination in sensor networks, *SPIE*, pp. 229-237, 1999.
[4] D. Estrin, *et. al.*, http://nesl.ee.ucla.edu/tutorials/mobicom02
[5] 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.
[6] J. Hill, A software architecture to support network sensors, Master's Thesis, UC Berkeley, 2000.
[7] J. Hill, R. Szewczyk, A. Woo, S. Hollar, and J. Heidemann, System architecture directions for networked sensors, *Proc. 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, November 2000.
[8] 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
[9] 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.
[10] 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.
[11] 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.
[12] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava, Energy-aware wireless sensor networks, *IEEE Signal Processing*, Vol. 19 (2), pp. 40-50, 2002.
[13] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava, Optimizing sensor networks in the energy-latency-density design space, *IEEE Transactions on Mobile Computing*, Vol. 1(1), pp. 70-80, 2002.
[14] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie, Protocols for self-organization of a wireless sensor network, *IEEE Personal Communications Magazine*, Vol. 7 (5), pp. 16-27, 2000.
[15] W. Ye, J. Heidemann and D. Estrin, An energy-efficient MAC protocol for wireless sensor networks, *INFOCOM 2002*, Vol. 3, pp. 1567-1576.