A Novel Tree based Data Gathering Protocol for Wireless Sensor Networks

Yu Niu, Brian J. d'Auriol, Sungyoung Lee
Dept. of Computer Engineering
Kyung Hee University
Suwon, South Korea
{niuyu, dauriol, sylee}@oslab.khu.ac.kr

***Abstract*—In this paper, we proposed a novel Backbone tree topology for the data gathering in Wireless Sensor Networks. By integrate the ideas of both linear chain and tree topology, the new Backbone tree topology obtains quicker network responding speed and longer lifetime results.**

I. Introduction

With the introduction of low-cost processor, memory, and radio technologies, it becomes possible to build inexpensive wireless micro-sensor nodes. These networks can be used to collect useful information from an area of interest, especially where the physical environment is so harsh that the macro-sensor counterparts cannot be deployed.

One generic type of application for sensor networks is the data gathering, in which sensor nodes periodically sense their nearby environment and send the information to a sink (Base Station) which is not energy or computing power limited. At the sink the collected information can be further processed for end-user queries. The key challenge in such data gathering is conserving the sensor energies, so as to maximize their lifetime. The lifetime can be expressed in terms of rounds where a round is the time period between two sensing activities of sensor nodes.

To collect data from each individual sensor node periodically, there have three classes of methods. One is direct transmission which is not preferred in most of situations because of high energy cost by the long distance between sink node and network field. The second method is clustering [1] [2] [3] [4] [5], the cluster head in charge of collecting members' data and transmit them to the Base Station (BS). The reduced number of long transmission and data aggregation used in head helps increase the energy efficiency. The third one is routing, all nodes transmit their data locally along the given routing plan. The routing method has good scalability and load balancing. The routing method also can be used whin the clustered network situation. The algorithm proposed in this paper belongs to the routing method.

Based on the periodical and low bit rate characteristics of data gathering application in wireless sensor networks, this paper proposes a novel Backbone tree algorithm. The algorithm construct a chain-like backbone in the spanning tree. With the backbone, the head duty of long distance transmission can be rotated in each round without reconstructing the whole topology. Meanwhile, the higher connection degree of bone node makes the data can be parallel transmitted, which greatly decrease the network procession latency.

The paper is organized as follows. Section **??** gives out the network working mode and the energy consumption model. Section III presents the idea of Backbone tree and its construction algorithm. Besides the time and message exchange complexity of the algorithm also analyzed in this section. In Section IV, Backbone tree demonstration and performance are investigated. Section V summaries the work and discusses the future work.

II. Network Working Model

A. Application Scenario

In the application scenario considered for this network model, sensor nodes periodically sense the environment and generate data in each round of communication. Here also assumes that the time period of sensing the environment is much bigger than the time required for transmitting all the data to the base station. Data fusion or aggregation is used to reduce the number of messages in the network. Following the constructed routing plan, each sensor node receives the data from its neighbors, aggregates or fuses them into one single packet, and sends the packet to the node on its way to the base station.

The aim is efficient transmission of all the data to the base station so that the lifetime of the network is maximized in terms of rounds, where a round is defined as the process of gathering all the data from sensor nodes to the base station, regardless of how much time it takes.

B. Energy Consumption Model

The energy consumption in wireless sensor networks usually use the first order radio model [6] [7]. Each senor node will consume the following E_{Tx} amount of energy to transmit a l-bits message over distance d:

$$\begin{aligned} E_{Tx}(l,d) &= E_{Tx-elec}(l) + E_{Tx-amp}(l,d) \\ &= \begin{cases} lE_{elec} + l\epsilon_{fs}d^2, & d < d_0 \\ lE_{elec} + l\epsilon_{mp}d^4, & d > d_0 \end{cases} \end{aligned}$$

E_{Rx} amount of energy to receive this message:

$$E_{Rx}(l) = E_{Rx-elec}(l) = lE_{elec}$$

In the model, the radio dissipates E_{elec} to run the transmitter or receiver circuitry and E_{amp} for the transmit amplifier. If the distance is less than a threshold d_0, the free space model (ϵ_{fs} and d^2 power loss) is used; otherwise the multipath (ϵ_{mp} and d^4 power loss) is used.

III. The Backbone Tree

A. Motivation and Tree Description

The spanning tree for constructing the whole network routing plan is a proper choice. Because the spanning tree can not only connect all member nodes but self grow along the physical shape of the field. However, the root node in a tree is prone to resource overburden because of the long distance transmission to the base station (Figure 1(a)). Once the root node fails, the whole topology will collapse. To avoid the root node die early, the system has to change the root node and reconstruct the tree frequently. This will increase the constructing cost. Based on this, [8] proposes the "Power-Efficient Gathering in Sensor Information Systems" (PEGASIS), in which all nodes are constructed into a linear chain. Data flow along the chain to arrive the head which is one of the node on chain and changed in each turn (Figure 1(c)). System do not need to reconstruct the whole topology to rotate the head duty, thus reducing the amount of energy spent per round. However, the data collecting latency of it is significant. In the chain topology, all node only can transmit their data serially. When the node number is not small, the time for waiting the data being transmitted from the ends of chain to the head may be very long.

To solve the above problems of the both methods, this paper propose a Backbone Tree for the data gathering application in sensor networks. The Backbone tree is a special kind of spanning tree. There are two kinds of nodes in the tree: bone nodes, which connect to each other to form a chain-like backbone, and child nodes which connect themselves to some bone node (Figure 1(b)). The bone nodes in charge of collecting the raw data from all its children and transporting the packed message along the backbone chain to the head which is one of the bone nodes. The head duty shifts on the backbone by turns.

The advantage of constructing a backbone in a tree is that the network can rotate the head duty without reconstructing the whole tree every round. The backbone tree shorten the length of 'chain' by allowing bone nodes to recruit children, on which the nodes can parallel transmit data so as ease the long latency problem.

Because the bone nodes take more responsibility than the child nodes, there are some energy threshold requirement for the bone node candidate during the backbone construction. During the backbone growing, the bone nodes take the greedy rule to recruit as much as possible children in its searching area. When the backbone stop growing and if there still left

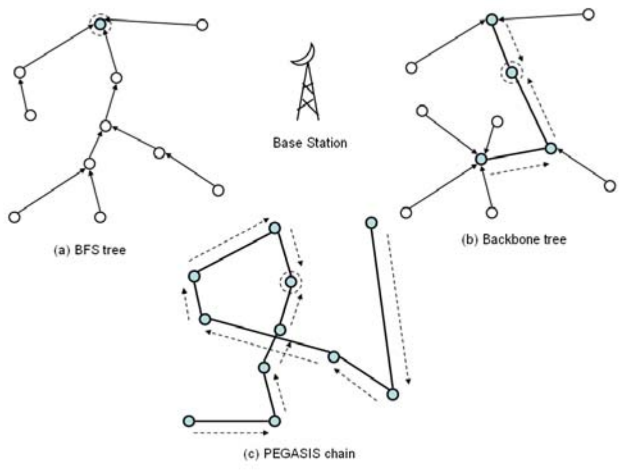


Fig. 1. Demonstration of three topologies. (Dashed circled node means the current head duty location).

some nodes that are not connected to the tree, these nodes try to connect themselves to the nearest node which is already connected to the backbone. Each bone node is only allowed to have at most two layers of child nodes, which is to guarantee the dominating role of backbone and shorten the network latency. In each turn, after the head collects all information and sends them out, the head duty will shift to the next node on the backbone chain. After all the bone nodes have taken the head duty once, the system can choose a new root and reconstruct a new backbone tree.

B. Construction of Backbone Tree

Before doing the construction, here gives out some node preconditions:

1) In network, each node has an unique identify number ID.

2) The sensor nodes can adjust different power levels to get different transmission ranges.

3) Each node keeps a neighbor table T_{nbr} to store the IDs of all its neighboring nodes located within the transmission range r. The content of this table is got from the BS message or constructed according to the nodes mutually informing messages. Once the T_{nbr} of each node constructed, unless some nodes die, the content of it won't change.

4) Besides T_{nbr}, each node also keeps a dynamic child table T_{d_cld}. Each time, before constructing the new Backbone tree, node loads the content of neighbor table T_{nbr} to the child table T_{d_cld}. During the tree constructing procedure, the content of table T_{d_cld} are updated from time to time, until the node is connected to the tree. Figure 2 shows the structures of two tables and their loading relationship.

5) Every node knows the energy threshold of being a bone node, which is refreshed termly by the BS broadcasting.

Following gives out the constructing steps of the Backbone tree.

Initialization: All node load new child table T_{d_cld} from its neighboring table T_{nbr}, and reset their 'Status' field as $Unconnected$, 'Role' as 1 which means normal child node.

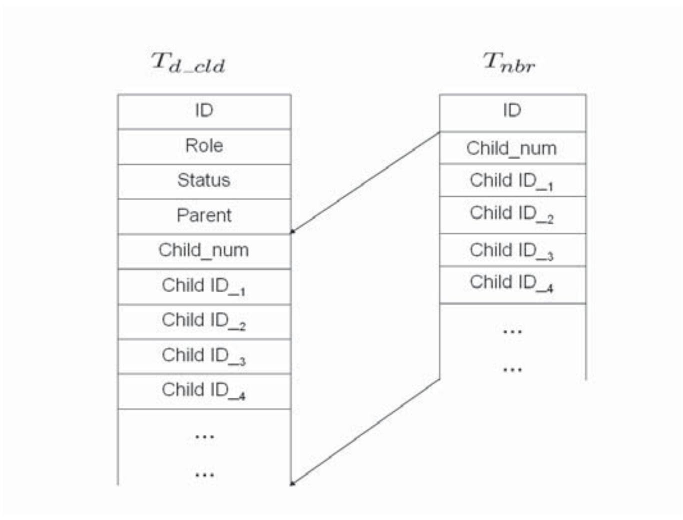


Fig. 2. Structure of neighbor table T_{nbr} and child table T_{d_cld}.

Step 1 Randomly select one energy qualified and unconnected node as the initial root and change the 'Role' field as -1 which means the root node. It also works as the first bone node of the backbone B_1.

Step 2 For each new determined bone node, check if there are entries in their child table T_{d_cld}. If it has, then go to step 3; if hasn't, wait until time out then go to step 4.

Step 3 For each bone node B_i who has child, determine the next bone node B_{i+1}. If current bone node is root node, then determine two bone nodes from B_0. Return to step 2.

Step 4 For all unconnected nodes, broadcast connecting request to all nodes in its neighboring table T_{nbr}. If the first acknowledgement message comes from node j, change its 'Status' field as $Connected$, 'Parent' filed as node j's ID and 'Role' field as 2 which means it is the second level child node. For any node who receives this connecting request, if its residual energy higher than the current threshold value, send acknowledgement message to the request node.

Step 5 If the unconnected node in step 4 didn't receive any acknowledgement message, then enlarge its transmission radius meanwhile decrease the threshold requirement and rebroadcast the request message. This time, only the node whose 'Role' field value is less than 2 react to this request. The receiving node who meets the both new threshold and 'Role' field requirement sends back the acknowledgement message. Repeat step 5, until all nodes's status become 'Connected'. Then terminate the algorithm.

To show How the current bone node B_i decides the next bone node B_{i+1} in step 3. Here lists the detail actions of the bone nodes and the related normal nodes respectively.

For the current bone node B_i

1) B_i receives the appointment message and change 'Role' field as 0 which means normal bone node, 'Parent' field as the sending node ID;

2) Assuming B_i has m_i entries in its current T_{d_cld}. B_i sends the grandchildren number inquiry request to all these m_i children.

3) B_i waits for all the m_i children's reply messages within the valid waiting time (VWT) and picks up the one who has the most grandchildren number.

4) B_i sends the appointment message to the picked up child and affirms it as the next bone node B_{i+1}.

For each node in the bone node B_i's T_{d_cld} table:

1) After receiving the grandchildren number inquiry from B_i, it mark its 'Parent' field as B_i.

2) Broadcast updating messages to all nodes in its T_{d_cld} table, which claims it has been recruited and asks all the receivers to delete its entry in their T_{d_cld} tables.

3) Meanwhile, waiting for the updating messages from other child nodes and delete corresponding entries in its T_{d_cld} until the valid waiting time (VWT) ends.

4) After the waiting time end, node cleans up its T_{d_cld} and counts the new child number, then sends the reply message to its parent node.

5) Change the 'Status' field as $Connected$. Unless receiving the bone node appointment message or latter connecting request in step 4 and 5, it no longer receives any updating or inquiry requests.

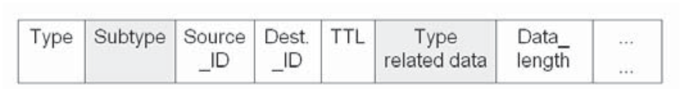
C. Time and Message exchange Complexity

Although in step 4 and 5, to connect the residual nodes to the tree, node may request all the nodes in network, the unconnected node number is constant. So processing time for connect residual node is at worst $O(n)$. In the algorithm, step 3 is the most time consuming part, in which each node will be inquired at most once. To reply the inquiry, it has to communicate with all its neighbors. So the processing time for arbitrate the bone nodes is at most $O(n)$ for each node and $O(n^2)$ for the whole network.

Next, the energy cost and message numbers used in step 3 are analyzed. For each child of B_i, there cost one receiving from its parent node; one reply to its parent; one broadcast to all nodes in its T_{d_cld} table; and at most m_i times receiving for the updating message to make them connected. Among them, for the chosen new bone node, besides the above mentioned cost it also needs one receiving of the appointment message; one new appointment message sending; one inquiry request broadcast to all nodes in T_{d_cld}; and m_i times reply message receiving from each child. For the nodes who are not the children of B_i but receive the updating messages from the children of B_i, they just delete the corresponding entries in their T_{d_cld} tables, but do not reply anything.

During this procedure, the system uses four types of control messages: 1) Bone node appointment message, 2) Grandchildren number inquiry message, 3) Table updating message, 4) Child number reply message. Among them, 1) and 4) is uni-cast that has clear destination ID; 2) and 3) can be whole transmission area broadcast or multi-cast with multiple destination IDs. Here gives out a general network message formate (Figure 3).

The 'Type' designates the property of message which is either control message or data message. Only if it is control message, the following 'Subtype' and 'Type related data' field



Type	Subtype	Source _ID	Dest. _ID	TTL	Type related data	Data_ length	... ...

Fig. 3. The format of system general control messages.

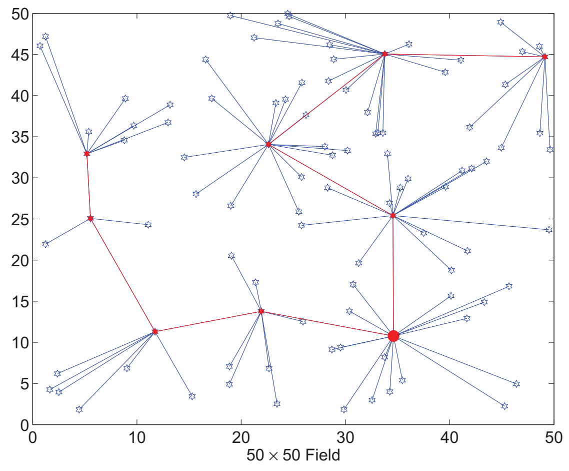


Fig. 4. Backbone tree in a 50×50 network field ($n = 100$, $r = 16$).

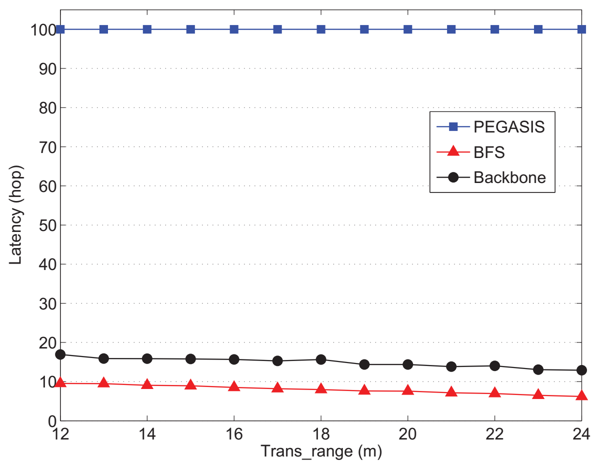


Fig. 5. Latency comparison of three algorithms. (n=100 in 100×100 m^2 field).

has meanings. The 'Subtype' specifies the kind of control message which could be one of the above four or the connecting request or acknowledgement message used in step 4 and 5. And the 'Type related data' field content is related to the 'Subtype' field value. Whatever the control message type is, the packet length is short and small amount of energy is needed. If it is data message with long data length, more energy is necessary and the data information is attached in the latter part of message structure.

IV. Simulation Results

In this section, we check the performance of the Backbone tree from the transmission latency (system react speed) and the network lifetime aspects respectively.

Before discussing the algorithms performance, here first shows an example of Backbone tree topology in a general sensor network in which 100 nodes are randomly deployed in a 50×50 sized field, and the initial transmission radius uses $r = 16$ (Figure 4).

A. Transmission Latency

During the data gathering, data information are transmitted in a multi-hop way. Besides the propagation delays in air, each relay node also need time to receive, repack and transmit the data package. All these cause the latency in the whole gathering procedure. So here use the network hops (including final node to the BS) to measure the transmission latency.

The transmission latency should be measured by the worst case that could happened during the data gathering. In PEGASIS, all member nodes are constructed into one chain. For chain-liked topology, all the receiving and transmitting is serial on the chain. The best case is when the head duty flows to the midpoint of chain that the latency is around the half length of chain; the worst case is when the head duty flows to one end of the chain that the latency would be the whole length of the chain. Also, the latency on a linear chain is not related with the transmission radius. So the transmission latency of PEGASIS is the number of node in the network and will not change until some node die early. In backbone tree, the bone nodes at most have two layers of underling nodes, so the longest latency of the backbone topology is the length of backbone chain plus one. For Breadth First Spanning (BFS) tree, the latency is the layer number of the deepest leaf nodes in the tree. For both backbone and BFS tree, the final tree topology is influenced by the transmission radius.

Figure 5 shows the transmission latency comparison of three algorithms under different transmission radiuses. All the data in this figure are averaged from 100 times network distribution. As we can see, the latency value of PEGASIS is very high compare to the other two, which always equals to the node number. The BFS has the shortest latency time because of its breadth first constructing rule. The backbone tree's latency is a little longer than BFS tree but still much shorter than PEGASIS.

B. Network Lifetime

In this part, we compare the lifetime of the three algorithms' topology. Here use the working round number to measure the network lifetime. Energy consumption of sensor nodes are calculated by the Equation 1 and II-B in the first order radio model. And parameters and network settings are listed in Table I.

Figure 6 shows the working round number of a network from its first to the last node dies, which demonstrates the lifetime performance and the energy consumption trend of three algorithms (Figure 6). In the figure, the later the first node dies, the better load balancing the algorithm has. And, the later the last node dies, the better 'survival ability' the algorithm has. The Backbone tree shows good load and energy consuming balancing. From the first to the earlier 30% died node, their lifetimes in Backbone tree are much longer than BFS tree. Also the earlier 16.7% died nodes' lifetimes in

TABLE I
SIMULATION ENVIRONMENT

Type	Parameter	Value
Network	Network grid	From (0,0) to (100,100) $meter$
	Base station	(200,200)
	Initial energy	$1J$/battery
	Head length	$25byte$
	Message length	$2000bits$
Radio model	E_{elec}	$50nJ/bit$
	ϵ_{fs}	$10pJ/bit/m^2$
	ϵ_{mp}	$0.0013pJ/bit/m^4$
	Threshold distance (d_0)	$\sqrt{\frac{\epsilon_{fs}}{\epsilon_{mp}}} \approx 87.7m$
	Data fusion cost	$5nJ/bit/message$

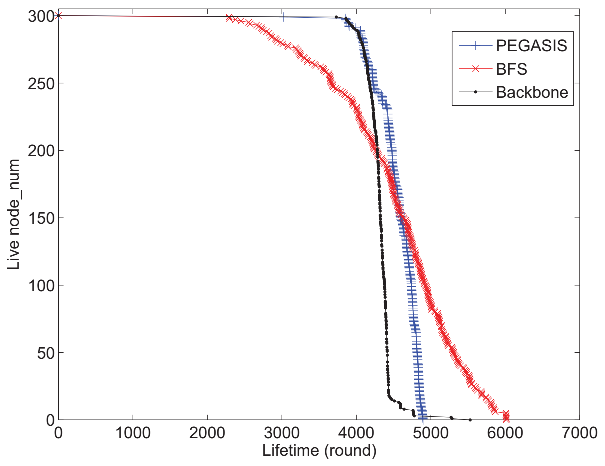


Fig. 6. Lifetime of all nodes in three algorithms. ($n = 300$, $r = 16$).

Backbone tree are longer or similar with in PEGASIS. In Backbone tree, after the first node die, the whole network also dies rather quickly. This property is preferred by some applications that require the whole network work cooperatively. On the other hand, although the first node die early, the BFS tree shows longer whole network lifetime than the other two. After a portion of node die, the remainder nodes can continue work for a rather long time with relatively high energy levels. Choosing what kind of property depends on the application requirement. From the above two criteria, PEGASIS achieve a relative best performance among the tree. It has later died first node than BFS tree and longer whole network lifetime than Backbone tree. However, its long latency is a fatal and restrict its applicability.

After knowing the energy depletion trends of the three algorithms, we discuss their lifetime performance under different node density (Figure 7) and transmission radius (Figure 8). Here mainly focus on the situation before the first node die. All the data in these two figures are averaged from 100 times network distribution.

In Figure 7, PEGASIS and backbone tree's lifetime is

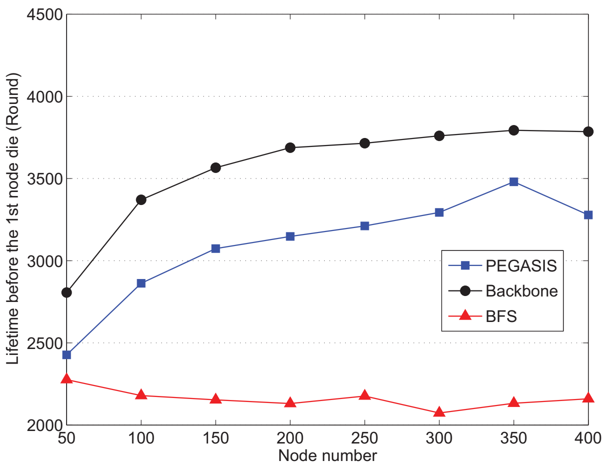


Fig. 7. Lifetime comparison of three algorithms under different node density before the 1st node die ($r = 17$).

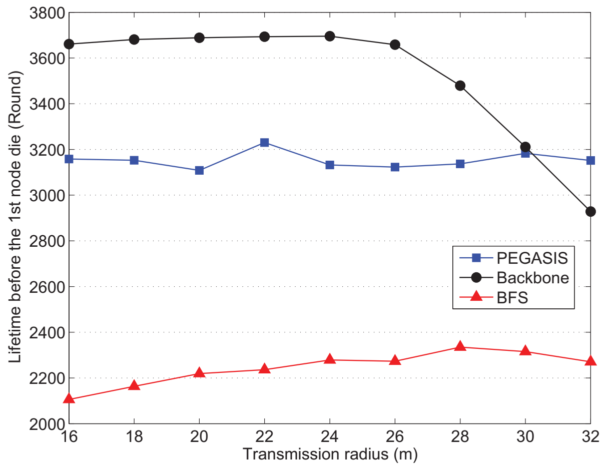


Fig. 8. Lifetime comparison of three algorithms under different transmission radius before the 1st node die ($n = 200$).

shorter in sparse network. That's because sparse network usually requires longer transmission distance between two nodes on chain. While the node density has not much influence on BFS tree. In Figure 8, for Backbone tree, when the transmission radius $r \leq 24$, the lifetime keeps almost similar level. And after $r > 24$, the lifetime decrease with the transmission radius increase. That's because in backbone tree, with the increase of transmission radius, the neighboring nodes also increased. Increased neighboring node in sparse network (like in Figure 7) helps rotate and distributed the bone node duty to more nodes so as to prolong the whole network lifetime. However, continue adding neighboring nodes (like enlarging transmission radius in Figure 8) will cause the bone nodes overburden in one reconstruction period. Because the backbone tree is not reconstructed every round like BFS tree, the bone node may die before the next reconstruction happened. So the large transmission radius does not influence BFS tree as much as Backbone tree. Combine the above discussion and the results in Figure 7 and Figure 8, we conjecture there exists a local extremum for Backbone tree's first node die lifetime under some certain neighboring number nodes (in Figure 7 and Figure 8's network setting, we calculate the maximum neighboring node is around 50). And this extremum gives a clue to the network's node saturation for obtaining Backbone tree's best performance.

V. Conclusion

In this paper, a novel Backbone tree idea and its constructing algorithm was proposed. The Backbone tree topology shows good performance in data gathering application. It shortened the linear chain length by allowing chain node to have child nodes, which greatly decreases the processing latency and helps to increase the network responding speed. Meanwhile the head duty rotating on the chain like Backbone decrease the reconstruction frequency, which decrease the energy consumption per round. Integrate with the bone node energy threshold control scheme, Backbone tree achieved good load balancing property.

However, the greedy bone node searching mode increase the algorithm's time and message exchange complexity. And this will be further improved in our future works.

Acknowledgment

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National Industry Promotion Agency) (NIPA-2010-(C1090-1021-0003)).This research also was partially supported by Kyung Hee University under the Young Researcher's Program (20090704).

References

[1] O. Younis and S. Fahmy, "Heed: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transcations on Mobile Computing*, vol. 3, no. 4, pp. 366–379, Oct 2004.

[2] F. Kuhn, T. Moscibroda, and R. Wattenhofer, "Initializing newly deployed ad hoc and sensor networks," in *Proceedings of the 10th annual international conference on Mobile computing and networking(MOBICOM)*. Philadelphia, PA, USA: Springer Gerlin, Sept 2004, pp. 260–274.

[3] A. Amis, R.Prakash, T. Vuong, and D. Huynh, "Max-min d-cluster formation in wireless ad hoc networks," in *In Proceedings IEEE Conference on Computer Communications INFOCOM 2000*, Mar 2000.

[4] H. Chan and A. Perrig, "Ace: An emergent algorithm for highly uniform cluster formation," in *Proc. First European Workshop Sensor Networks (EWSN)*. Springer, Jan 2004, pp. 154–171.

[5] M. Chatterjee, S. K. Das, and D. Turgut1, "Wca: A weighted clustering algorithm for mobile ad hoc networks," *Cluster Computing.*, vol. 5, no. 2, pp. 193–204, Apr 2002.

[6] W. Heinzelman, A.Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless sensor networks," in *In Proceedings of the 33th Hawaii International Conference on System Sciences*, 2000.

[7] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transcations on wireless communications*, vol. 1, no. 4, pp. 660–670, Octorber 2002.

[8] S. Lindesey and C. Raghavendra, "Pegasis: Power-efficient gathering in sensor information systems," in *Aerospace Conference Proceedings, 2002. IEEE.* Big sky, Montana, USA: IEEE, Mar 2002, pp. 3–1125–3–1130.