A Fault-recovery Routing Approach for Loop-based Clustering WSN*

Ming Xu[1], Shengdong Zhang[1], Jiannong Cao[2], Xiaoxing Guo[3]
([1]School of Computer, National Univ. of Defense Technology, Changsha, China)
([2]Dept of Computing, Hong Kong Polytechnic University, Hong Kong, China)
([3]School of Information Technology and Electrical Engineering, University College, University of New South Wales, Australia)

Abstract

Clustering has been used as an effective approach to saving energy and providing better scalability for Wireless Sensor Networks (WSN). Most of the existing methods for clustering use a star-based topology while the newly proposed loop-based topology brings some unusual advantages. In this paper, we present a loop-based clustering method for WSN with an improved mechanism. Based on the data flow characteristics of WSN, we design an algorithm to release the hop information from the sink node to all nodes in a fully distributed way, and design the routing protocol based on the information. Considering failure recovery, the paper proposes a recovery algorithm to overcome node failure and communication failure. The simulation results show that both algorithms have low communication and storage overhead and meet constraints on low energy consumption and low bandwidth of WSN while improving the fault-tolerance capability.

1. Introduction

Continuous advance in processor technology and in development of wireless communication and digitized electronics are pushing the research on Wireless Sensor Networks (WSN) with low-cost, low-power multi-purpose sensor nodes capable of communicating in short distance. A number of the micro-nodes constitute a sensor network by wireless communication that is called sensor nodes in WSN work in harness with each other to monitor the disposed area in real-time manner. Meanwhile, the monitoring data to be collected is determined by real application problems.

A WSN node is not of high mobility, so the introduction of stable clustering structure can save the energy cost and improve the high expansibility in most cases. The classical clustering model LEACH was proposed by W. R. Heinzelman *et al.* [2]. The model selects a cluster header randomly and shares the relaying communication services, while the cluster header is responsible for data fusion and routing. It also defines the concept of *round* that is composed of two stages: initiation stage and working stage. A cluster head is voted in every new round to achieve the balance of nodes. Many works such as TEEN [3], PAGASIS [4], HEED [5] have been proposed based on LEACH. However, almost all existing clustering methods for WSN are star-based, which manifest themselves with the following problems:

(1) The definition of round is vague;
(2) Cluster reconstruction has a high cost and the change of topology is hard to forecast;
(3) After cluster reconstruction, the local topology information of former nodes is useless;
(4) The failure of the cluster head would be a disaster;

Recently, a novel clustering model, which employs a loop-based topology for a cluster, is proposed [6]. The loop topology has the following advantages:

(1) There is no critical cluster head defined in a loop, so the loop-based topology never suffers from chain reactions caused by the changes of the cluster heads;
(2) Within a loop, every node is necessary to have knowledge of the other nodes on the loop. So. if the information of the local loop recorded in a node is corrupted, by querying the neighboring nodes, the loop knowledge can be recovered, which provides the network with better robustness;
(3) There exists two paths between every two nodes on the same loop, providing a backup route for connection loss during message transmission.

How to employ the loop-based WSN topology for routing and for failure recovery is the focus of this paper. The remainder of the paper is organized as follows. Section 2 introduces the related works on clustering based on loop topology and K-ary connected (KC) topology. Section 3 introduces our proposed algorithm of loop-based cluster construction. Section 4 proposes the algorithm to distribute the hop information of nodes and loops in WSN. Section 5 proposes a routing algorithm while section 6 proposes a loop-recovery algorithm.

* The research is supported by Doctoral Foundation Program of China Education Department, No. 20059998022

Section 7 describes the result of our simulation study for performance evaluation. Finally section 8 concludes the paper.

2. Related Works

The data flow characteristic of WSN is different from traditional ad hoc networks, and the communication within a WSN is asymmetrical. In many research works, the control center must send requests to set up a data routing path which is to be recorded in the nodes along the path temporarily. An example is the model of directed diffusion for WSN proposed by D. Estrin *et al.* [7].

Zhang *et al.* proposed a KC topology algorithm which can guarantee K-ary connection of every two nodes in most situations [1]. The algorithm is based on the distribution of the communication power. Its motivation is to delete the links of long distances so as to let a node save the consumption of communication power. Moreover, every node only needs to send out the message with the max power less than three times. It is proved that each node at most holds 6**k* links. The KC topology control algorithm could adjust the communication power of nodes, thus save energy and reduce conflict.

Also, in the loop-based clustering model [6], every node has the complete information about the loop it belongs to. The node that belongs to several loops is called the gateway. Because there are two links between every two nodes in a loop, providing the backup routing path, the topology can provide better route recovery. The process of loop construction is that every node broadcasts its message of request for loop (LREQ) locally; if node u receives two messages of LREQ that both passed node v, u discovers a loop, and generates the message of reply for

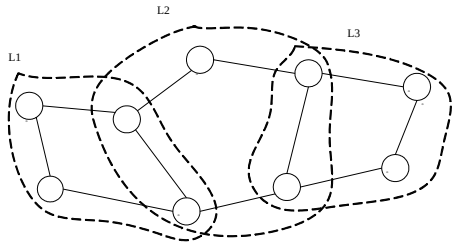

Fig. 1 A loop-based cluster

loop, and sends it to the nodes on the path. Fig. 1 shows the basic structure of a loop-based cluster, and Fig. 2 illustrates the construction of a loop.

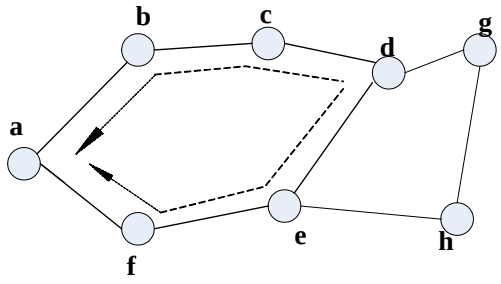

Fig. 2 An example for the basic algorithm

3. A Modified Loop-based Clustering Model

This section describes our modified loop-based clustering model in detail. We introduce a KC topology control algorithm before describing loop-based cluster construction.

3.1 KC topology control algorithm

We add some operations when KC algorithm is executing for getting the nodes that near the *sink* node. In our approach, the *sink* node receives the hello message from the nodes with the max communication power. And the sink node would be aware of the distance to the nodes that send the message according the signal intensity. Let's suppose the max communication radius is p_{max}, the *sink* node would choose the nodes within the distance of p_{max}/t (here t is a parameter, and its value is consistent with the nodes density and the application character), and transmit the reply message to them. The node that receives the reply message be aware of that it is the neighbor of the sink node, and it will initiate the dissemination of the hop information.

In addition, the neighbor nodes table would be reserved for the loop recovery algorithm.

3.2 Loop-based cluster and its refinement

The formation of a loop is based on the KC topology control algorithm. The process is similar to the algorithm in [6]. And the differences are listed as follows.

(1) With KC topology control algorithm, the nodes broadcast the LREQ with their necessary power gathered before, not with the max communication power;

(2) For loop building, we choose a range for the number of nodes in one loop instead of a predefined number. Any nested loops must be deleted. For example, as shown in Fig. 3, node *a, b, c, d, e, f* construct a loop and node *b, c, d, e, f* construct another loop, and these two loops are nested, so the smaller one (contain less nodes) should be deleted. If node *d* firstly discovers the case, it generates a

message *c-loop* (i.e., cancel of the loop), and sends it to the nodes that belonged to the smaller loop.

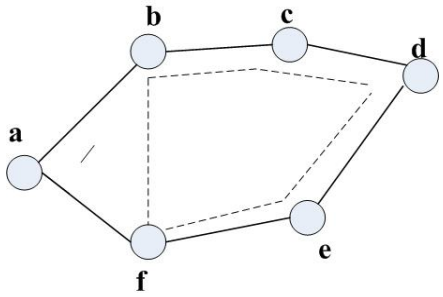

Fig. 3 The deletion of the nested loop

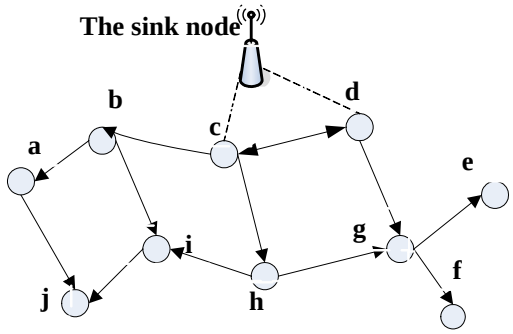

Fig. 4 The distribution of hop information

4. Build an advanced structure for data routing

A simple loop-based cluster in WSN is not enough for data routing. In this section, we propose two algorithms to get the hops from sensor nodes and loops to the *sink* node. With such information gathered, the routing algorithm would be guided, which means every node knows the correct path to transmit data to the *sink* node.

4.1 The distribution of hop information of nodes

According to the KC topology control algorithm, the nodes which find the *sink* node are those nodes whose distance is within p_{max}/t. These nodes will distribute the hop information over the whole network. In fact, the hop of them will be 1, and the neighbors of them within the same loop will be 2, and so on. Table 1 shows the data structure designed for the algorithm.

As shown in Fig. 4, node *c* and *d* are neighbors of the *sink* node, and suppose they initiate the algorithm. When node *b* and *h* receive the hop message from node *c*, they get their states and transmit the message to the nodes. The process would continue, as far as all nodes get their states.

Table 1 Data structure of hop information

Data structure	Explanation
v.hop	The hop of node *v*
v.hloop.head	The node with the smallest hop in the loop contains node *v*
v.hloop.hop	The hop of *v.hloop.head*
hinf.head	The begin node of a loop in transmitting the message
hinf.head_hop	The hop of *hinf.head*
hinf.hop	The current accumulative number of the message according to which the receive node get its own hop

The detail of the whole algorithm can be found in [8]. After the algorithm, every node (e. g. node *v*) knows its hop from the *sink* node, also knows the node with the smallest hop in the loop contains node *v*. They are the data structure *v.hop* and *v.hloop.head*. It was proved that *v.hop* is the smallest hop from node *v* to the *sink* node [8].

4.2 To build the hop of loops

After every node gets to know its hop from the *sink* node and other related information, it shares the information in its loops. Then the hops of loops can be acquired. There is no dependent among loops, and each loop can handle the information in parallel. Table 2 shows the explanation of the data structure used in the algorithm.

Generally, let p_1, p_2,…, p_k be nodes which located deasil around the loop L. For every i ($1 \leqslant i \leqslant k$), node p_i carries out the operations as follows:

$L.hop \leftarrow p_i.hop$,

If $p_i.hop=1$, $p_i.hloop.hop \leftarrow 0$.

And transmit the message hloopinf to p_{i+1}(p_k to p_1), the data structure like:

$\{p_i, p_i.hop, p_i.hloop.hop\}$.

When node p_j receives the message *hloopinf* from node p_i, it follows the steps below:

(1) If $p_i=p_j$ (which means that the message has transmitted around the whole loop and with no use), just discard the message, and then go to step 6.

(2) If $L.hop>p_i.hop$, $L.hop \leftarrow p_i.hop$

(3) If $p_i.hloop.hop<L.hop$, add $\{p_i.hloop.hop, p_i\}$ to *L.preloop*

(4) For every member $\{v, v.hloop.hop\}$ in *L.preloop*, if $L.hop \leqq v.hloop.hop$, delete $\{v, v.hloop.hop\}$ from *L.preloop*

(5) Transmit the message *hloopinf* to p_{j+1} without any modification

(6) End

Table 2 Data structure for loop-hop building

Data structure	Explanation
u.hop	the hop from node *u* to the *sink* node
L.hop	the hop of loop *L*, is defined by the smallest hop of the nodes belong to it Viz $L.hop=\min\{p_1.hop, p_2.hop, \ldots, p_k.hop\}$
L.preloop	The set of nodes and its hops of loop *L* that belong to some loops with smaller hop than *L* Viz.$L.preloop=\{p_i.hloop.hop, p_i \mid p_i.hloop.hop<L.hop\}$

After every node acquires *preloop* data structure of all the loops, the node randomly deletes the member of some loop if it discovers that there are the same two members in *preloop* of two different loops. The reason is that these two members will construct the same path in the routing algorithm.

Fig. 5 shows the typical execution of the algorithm, node p_1, p_2, p_3, p_4, p_5 belong to a loop, and they would transmit their message clockwise independently. For example, node p_1 would transmit the message to the node p_2, and the message pass through p_2, p_3, p_4, p_5, and these nodes would update their information according to the message. When the message finally arrives at node p_1, it is discarded.

If there are *n* nodes in loop L, each node in loop L would communicate *2n* times (including transmitting and receiving). The algorithm could be explained by:

(1) If node p_j receives the message from node p_i, and discovers that $L.hop \leqq p_i.hop$, p_j it knows that the message isn't useful for the hop of the loop, and thus will be discarded. Why p_j transmits the message as usual? The reason is that the $p_i.hloop.hop$ information is within the message, which would be useful for the *L.preloop*.

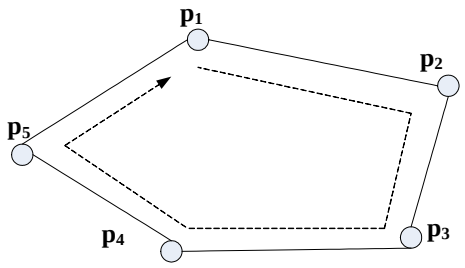

Fig. 5 An example of the algorithm

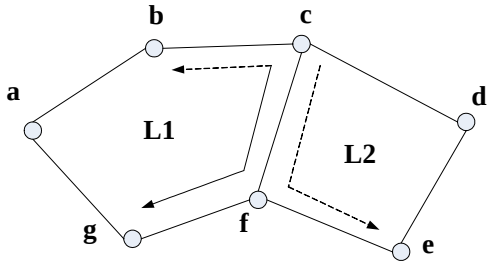

Fig. 6 An example of the routing algorithm

(2) The data $p_i.hloop.hop$ isn't within the loop L, but it is gathered by the nodes in the loop L in the above algorithm, so this algorithm can execute in loop.

(3) At the beginning of the algorithm, when node *u* find that *u.hop*=1, then $u.hloop.hop \leftarrow 0$. Why? If *u.hop*=1, *u* is the neighbor node of the *sink* node. Suppose loop L contains node *u*, we can get *L.hop*=1, and if *u.hloop.hop*=1, *L.preloop* would be empty! We can't process the next routing algorithm! So *u.hloop.hop* must be 0.

If we consider that the node *u* and the *sink* node construct a loop (a loop with two nodes!), it is easy to understand: the smallest hop node which belongs to the same loop with node *u* is the *sink* node, and the hop of the *sink* node is 0.

5. Routing

Considering the shortage of mobility, a novel mechanism is designed to make every node in the network know the correct routing to the *sink* node. The data routing is based on the distribution of the hop information, stored in the *preloop* data structure owned by the nodes that attempt to transfer data. The detail of the algorithm is explained by three specific cases below:

Case 1:

Node *u* itself gathers the data and attempts to transmit to the *sink* node; (let *cont* be the content of the data.)

If *u.hop*=1 (which means that *u* is the neighbor of the *sink* node), *u* generates message:{*u, cont*}, and transmits to the *sink* node directly.

Otherwise, *u* lookups its *preloop* data structure.

Let $p_1, p_2, \ldots, p_k$ be the members in it, and $q_1, q_2, \ldots, q_k$ be the corresponding hops. Then *u* chooses p_i with the probability of:

$$\frac{1/(q_i+1)}{1/(q_1+1)+1/(q_2+1)+\ldots+1/(q_k+1)}$$

for the transition to the *sink* node (here we add 1 to the hop because some hops are equal to zero). If node p_i is chosen, *u* generates message: {*u, cont,* p_i}. Let L be the loop that contains node *u* and p_i. Then *u* chooses a direction (clockwise or anticlockwise) of L to transmit the data.

Case 2:

Node *v* receives a message:{*u, cont, v*}, which means that *v* is the stopover of the message. Then *v* transmits the message in the way of case 1, but it doesn't transmit the message to the node *u*, even if *u* is in the *preloop* data structure of node *v*.

Case 3:

Node *t* receives a message:{*u, cont, v*}, also $t \neq v$. Then *t* simply transmits the message to the next node nearer to the node *v* in the loop L.

Fig. 6 shows the typical execution of the algorithm. Node *c* belongs to the loop L1 and L2. When it attempt to transfer data to the *sink* node, it lookups its *preloop* data structure, and finds that nodes *b, g, e* are the stopover. It chooses one node of them with a predefined probability. For example, node *g* is chosen, and then *c* transmits the message to the node *g* clockwise or anticlockwise.

6. Loop recovery

In WSN, node failure is quite common because of the limited energy. So our loop recovery algorithm is considered mainly to overcome it. We focus on topology management. As the energy of every node in the network is limited, and the batteries can't be replaced timely, especially in countryside. When the batteries are exhausted, the node is dead, so we need to guarantee that the network can still behave normally even though some nodes are dead. The survivability relies on the data structure of neighbor nodes table gathered in the KC topology control algorithm and the loop-based structure. Our algorithm is described as follows.

When node *u* discovers that its neighbor in loop L is invalidated, *u* lookups the nodes list of loop L, and find that node *v* is the another neighbor of the node *x*. Then the algorithm goes according to the conditions below:

(1) *L.preloop*={*x, x.hloop.hop*}, which means that the nodes in the loop L must transmit the message to the node *x* if they attempt to transfer data to the sink node. So it executes in the situations below:

(1-a) *v* is the neighbor of *u*, then *u* transmit the neighbor nodes information to node *v* directly (except the nodes that belong to the loop L). When node *v* receives this message, it discovers the nodes that are also the neighbor of itself in the message. Suppose they are $p_1, p_2, \ldots, p_k$, for each p_i, add the distances that from p_i to *u* and from p_i to *v*, and sort them with the sums by decrease order, transmit this sorted information to the node *u*. Then *u* and *v* transmit messages to these nodes in turn. For example, node *u* to node *t*, the message is like: {*u, L.hop, t*}. When node *t* receives the message, if *t.hloop.hop*<*L.hop*, *t* transmits reply message to the node *u*, the message is like: {*t, t.hloop.hop, t.hop, u*}; otherwise it does nothing. If both *u* and *v* receive the reply message from node t, they add the node *t* to the loop L, and inform other nodes in the loop L to update their loop information and hop information. All the nodes in the loop L transmit the updated information to their neighbor nodes in the other loops, to update the information of the whole networks. If *u* and *v* can't receive the reply message from the same node, for every two neighbors in the loop L, carry out the operation just like the *u* and v did. And if neither neighbor receives the reply message from the same node, the loop is invalidated.

(1-b) *v* is not the neighbor of the node *u*, which communicates to node *v* through another path in the loop, and does the same operations like (1-a). However if *u* and *v* can't find a neighbor both, the loop is invalidated.

(2) *L.preloop* $\neq$ {*x, x.hloop.hop*}, the algorithm goes according to the situations as follows:

(2-a) *v* is the neighbor of *u*, then simply delete node *x* from loop L, and update the information of nodes of the loop L.

(2-b) *v* is not the neighbor of *u*. Nodes *u* and *v* search their neighbors like (1-a). The result is that the node *t* with the smallest sum of distances from *t* to *u* and *v* could be discovered no matter what the hop information of node *t* will be and how the network updated. Also if there is no neighbor, the loop is invalidated.

With the neighbor nodes information gathered in the KC topology control algorithm, loop recovery algorithm can recover the loop even though some nodes are dead. Because the algorithm uses the original structure of the loops, it shed small effect on the other parts of the network. The update message is disseminated only if

needed. When node *u* and *v* search their neighbors, the ones they selected is the one with the lowest cost.

There are two situations that the algorithm is invalidated:

(1) Node *u* and *v* are not neighbors, and they can't find their common neighbor with the max communication power. Due to this reason, the probability is very small with the loop topology control.

(2) There is no node with a smaller hop than the loop can discover by every two neighbors. We have proved that *u.hop* is the smallest number of hops from node *u* to the *sink* node, if this happens, it is considered that there are too many nodes died and the network no longer works.

7. Simulation

Our algorithm is simulated with MATLAB. The monitoring area is a square with the size of 100*100, inside the area there are 100 nodes supposed distributed randomly.

Table 3 The comparison of 4 protocols

The energy of a node （J/node）	Protocols	First node dies	Last node dies
0．25	Direct	55	117
	Directed Diffusion	47	457
	LEACH	394	665
	Our approach	231	673
0．5	Direct	109	234
	Directed Diffusion	238	573
	LEACH	932	1312
	Our approach	641	1231
1	Direct	217	468
	Directed Diffusion	457	852
	LEACH	1848	2608
	Our approach	1324	2314

The radio model here is similar to LEACH [2]. The max communication radius is 30. The sink node is located in the border of the area. Table 3 shows the result of our algorithm compared with some existing algorithms. It could be seen that our algorithm is near to LEACH, better than the other two.

The traditional criteria of WSN are classified as: the first node dies; the last node dies and the half of the nodes die. In fact, there are some problems as shown in Fig. 7. We can see that there are still many nodes working in the network, but in fact the network can't monitor the majority of the area. Thus new criteria are needed. Here, we propose a new criteria, key area, which is the most important place for the monitoring tasks. If all key areas are out of the monitoring, the network is invalidated. Fig. 8 shows the key areas defined in our simulation. Table 4 shows the simulation result under the new criteria. It is seen that our algorithm is better than other algorithms. Also, we show the energy consumption on data gathering simulation results in Fig. 9. It is obvious that the curve increases much slower than other protocols while gathering data.

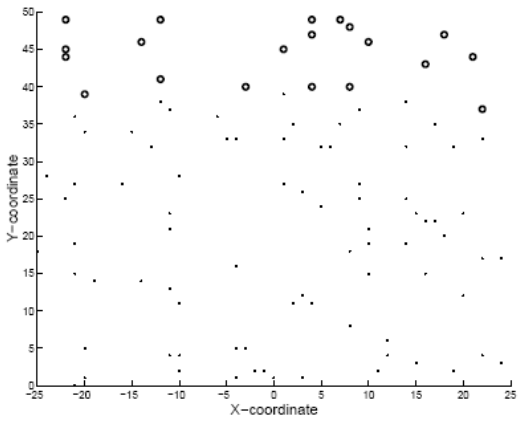

Fig. 7 Some nodes work while the network is invalidated

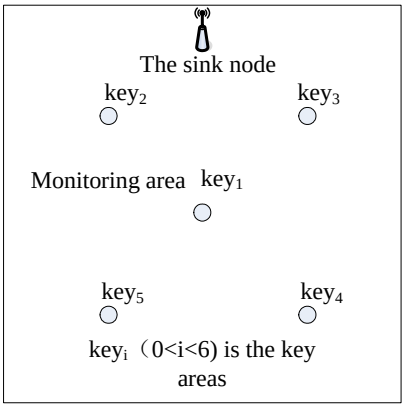

Fig. 8 The explanation of key area

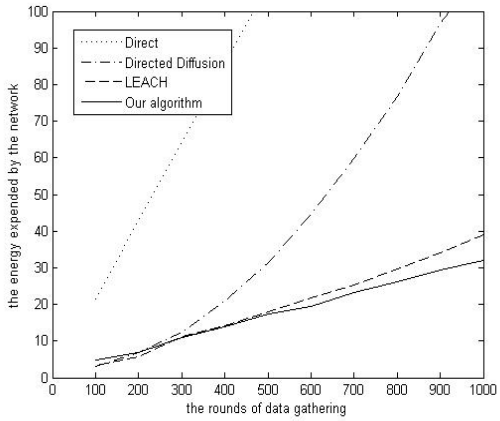

Fig. 9 Energy consumption of different protocols

Table 4 Data gathering under different protocols (new criteria, radius is 10)

The energy of a node （J/node）	Protocols	Half of the key areas are out of monitoring
0．25	Direct	95
	Directed Diffusion	243
	LEACH	268
	Our approach	431
0．5	Direct	176
	Directed Diffusion	379
	LEACH	1065
	Our approach	1143
1	Direct	321
	Directed Diffusion	642
	LEACH	2034
	Our approach	2117

8. Conclusion

Till now, few works on clustering in WSN take into consideration of the loop-based topology in the design of network protocols, and almost no work addressed the evaluation of the architecture. In this paper, we introduced a novel loop-based clustering method, including algorithms for routing and loop-recovery. The simulation result shows that our approach performs better, in terms of energy consumption, than some existing protocols. Our future research will be focused on two aspects:

(1) Acquiring more precise evaluation on the analytic model.

(2) Improving the proposed algorithms to fit some specific WSN applications.

References

[1] L. Zhang, X Wang, W. H. Dou, “A K-connected energy-saving topology control algorithm for WSN”, *Proc. of the 2nd International Symposium on Parallel Distributed Processing and Applications (ISPA’04)*, Hong Kong, China, Dec. 2004, pp.178-187.

[2] W. R. Heinzelman, A. Chandrakasan, *et al.*, “Energy-efficient communication protocol for wireless micro-sensor networks”, *Proc. of the 33rd Hawaii International Conference on System Sciences*, Jan. 2000, pp. 4-7.

[3] A. Manjeshwar, D. P. Agrawal, “TEEN: A routing protocol for enhanced efficiency in wireless sensor networks”, *Proc. of the 2nd International Workshop on Parallel & Distributed Computing Issues in Wireless Networks and Mobile Computing*, 2002, pp. 195b.

[4] L. S. Raghavendra, “PEGASIS: Power efficient gathering in sensor information systems”, http://www.cs.wayne.edu/~loren/csc8220-info

[5] O. Younis, S. Fahmy, “Distributed Clustering for Scalable, Long-Lived Sensor Networks”, *Purdue University, Technical Report CSD TR-03-026*, June 2003.

[6] Y. P. Li, X. Wang, F. Baueregger, *et al.*, “Loop-based Topology Control in Wireless Sensor Networks”, *Proc. of ICCNMC’05, LNCS 3619*, Aug. 2005, pp.1160-1169.

[7] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, “Directed Diffusion for WSN”, *IEEE Transaction on Networking*, Vol. 11, No. 1, Feb. 2003, pp. 2-16.

[8] Shengdong Zhang, “The Research of Energy-Saving Routing and its Fault-Tolerance capability for Loop-based WSN”, MSc Dissertation, National University of Defense Technology, Dec. 2005.