DOI: 10.3745/KIPSTB.2009.16C.5.621

무선 센서망의 병목 노드 탐색을 위한 분산 알고리즘

Haosong Gou[†] · 김 진 환[†] · 유 영 환[††]

요 약

무선 센서망은 위험이 존재하는 환경에서 공공 기관이나 군사적 목적의 신뢰성 있는 모니터링을 위한 수단으로 사용되어 왔다. 위험 상황이나 위험 물질이 존재하는 환경에서 사용된다는 특성 때문에, 망의 수명이 기존 무선망에서보다 더욱 중요한 성능 요소로 간주된다. 망 전체의 수명을 결정짓는 주요 요소 중 하나는 수명을 다하는 최초의 노드가 얼마나 이른 시간에 나타나느냐 하는 것이고, 이 최초의 노드는 망에 존재하는 병목 노드일 확률이 높다. 무선망의 병목 노드를 찾아내기 위한 방법으로 MINCUT 알고리즘이 대표적이나, 이는 중앙 집중형 방법이어서 많은 수의 노드로 이루어진 무선 센서망에는 적합하지 않다. 본 논문에서는 알고리즘의 계산 복잡도를 낮추어 매우 짧은 시간 내에 병목 노드를 탐지해 내는 분산형 방법을 제안한다. 수학적 분석과 실험을 통해 제안된 알고리즘이 기존 방법보다 훨신 향상된 성능을 보임을 확인할 수 있다.

키워드 : 무선센서, 병목탐지, 분산알고리즘, 에너지효율

A Distributed Method for Bottleneck Node Detection in Wireless Sensor Network

Haosong Gou[†] · Jinhwan Kim[†] · Younghwan Yoo[††]

ABSTRACT

Wireless sensor networks (WSNs) have been considered as a promising method for reliably monitoring both civil and military environments under hazardous or dangerous conditions. Due to the special property and difference from the traditional wireless network, the lifetime of the whole network is the most important aspect. The bottleneck nodes widely exist in WSNs and lead to decrease the lifetime of the whole network. In order to find out the bottleneck nodes, the traditional centralized bottleneck detection method MINCUT has been proposed as a solution for WSNs. However they are impractical for the networks that have a huge number of nodes. This paper first proposes a distributed algorithm called DBND (Distributed Bottleneck Node detection) that can reduce the time for location information collection, lower the algorithm complexity and find out the bottleneck nodes quickly. We also give two simple suggestions of how to solve the bottleneck problem. The simulation results and analysis show that our algorithm achieves much better performance and our solutions can relax the bottleneck problem, resulting in the prolonging of the network lifetime.

Keywords : Bottleneck Detection, Distributed Algorithm, Energy Efficiency, Wireless Sensor

1. Introduction

The wireless sensor network (WSN) lately attracts considerable attention and is widely used for sensing a variety of environment conditions such as temperature, humidity, and density of air pollutant. The main reasons for its popularity are the low price and the ease to form the network. Particularly, such networks are useful in hostile or dangerous environment such as the battlefield and a logistics port [1-2], as the convenience of deployment is the most important feature to be considered.

Unlike the traditional WSN, in most applications of wireless sensor networks, WSNs have resource constraints like limited energy, low storage capacity, and weak computing capability. Furthermore, due to the hazardous working environment, sensor resources, especially the battery, may not be replaced or recharged. Therefore, the lifetime of the WSN highly depends on the energy

※ This work was supported by the Grant of the Korean Ministry of Education, Science and Technology (The Regional Core Research Program/Institute of Logistics Information Technology).
† 준 회 원 : 부산대학교 컴퓨터공학과 석사과정
†† 종신회원 : 부산대학교 정보컴퓨터공학부 교수(교신저자)
논문접수 : 2009년 8월 6일
수 정 일 : 1차 2009년 9월 9일
심사완료 : 2009년 9월 9일

consumption of the sensors.

Like the other wireless networks, the position of sensor nodes has great impact on the performance of WSNs in terms of detection coverage, communication cost, lifetime, and the resource management. In a normal situation, sensors can be placed manually at their optimal positions using various algorithms from the literature [3-4]. These algorithms, however, cannot be used in a special area where sensor distribution cannot be controlled.

In our research, the sensor nodes are deployed in port randomly, and the nodes move periodically. Due to the randomness of deployment, there always exist some nodes connecting two or more partitions without any backup nodes. If these nodes die out, the whole network will be partitioned, and the application will fail. These nodes are called *bottleneck nodes*, and it is important to seek bottleneck nodes in order to prevent network partitioning in the future.

The MINCUT algorithm can be used for bottleneck node detection [5]. In this method, a normal node or the sink node knows the global topology information and it can determine the bottleneck point. It is very useful for a small network but it may not be useful for a huge network due to the high complexity. So we first propose a distribute method called DBND(Distributed Bottleneck Node Detection) for bottleneck nodes detection, and our simulation substantiates the efficiency of the proposed method. Meanwhile, after the bottleneck nodes detection, we also give two solutions to relax the problem caused by bottleneck node based on our previous work [11-12].

The rest of the paper is organized as follows: Section II reviews related work about the bottleneck detection. Section III describes the proposed method that how to find out the bottleneck nodes and solutions. Section IV shows the performance evaluation of the proposed protocol and compares with the previous network. Section V gives the conclusion.

2. Related Work

2.1 Study in non wireless sensor network

Bottleneck nodes have been concerned and studied extensively in Internet and traditional wireless networks fields [7-8] except WSNs. For Internet and traditional wireless networks, the goal of concern is to avoid the congestion and improve the capacity of data flow, resulting in the enhancement of network throughput. However, in WSNs, due to the different network properties, the most important objective is how to save and balance the energy consumption to prolong the lifetime of the whole network. So the related work for other networks may not be suited for WSNs.

2.2 Study in wireless sensor network

For WSNs, an algorithm called MINCUT proposed by Mechthild Stoer [5] is usually used for bottleneck nodes detection in traditional wired and wireless networks [6], meanwhile it also can be used for bottleneck nodes detection in WSNs [6] as a centralized algorithm. The basic idea is that a sink node gathers the location information of all nodes in the network and uses the MINCUT method to compute and find out the bottleneck nodes. According to the principle, it seems to be an optimal method for WSNs. However, the algorithm complexity is $O(N^3)$ [5], where N is the number of the nodes in the network. If the network is huge, it takes much time to collect the global information and compute the bottleneck nodes, thus it may be impractical.

In MINCUT algorithm, the course of execution can be divided into two phases: (a) Global information collection. All the nodes in the network have to broadcast their position information to the sink node, this course can cause message flooding. The cost is very high. (b) After getting the global information, the sink node will compute and find the bottleneck node using MINCUT algorithm.

2.3 Importance of bottleneck nodes detection

In WSNs, due to the limited energy of each sensor node, the prolonging of the whole network lifetime has to be considered as the primary issue. So many protocols[9 13] have been proposed for handling this problem. As mentioned in Section I, the bottleneck node plays the role of a “bridge” to connect two or more partitions. The goal of WSN is information collection. If a bottleneck node dies out, some partitions or nodes will be separated from the network and the separated partitions and nodes can be considered as *dead nodes*. Therefore, we can say that the bottleneck node can decide the fate of many other nodes.

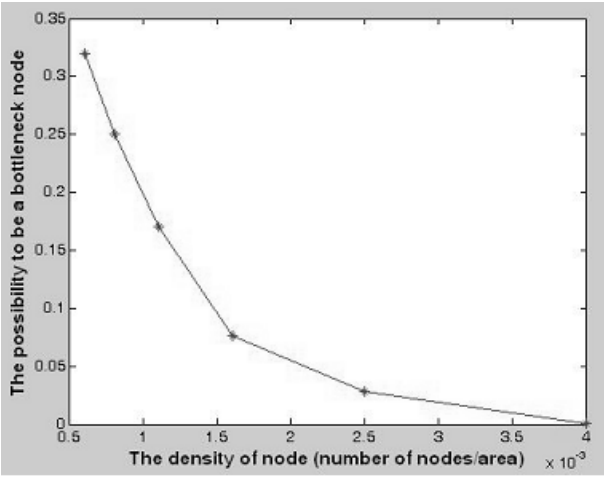
3. Proposed Method

3.1 Statistics of bottleneck nodes in real application

In our research, many sensor nodes are randomly deployed in a port to gain environmental information. Due to the random deployment, sometimes the network topology may be inefficient in the respect of throughput. The bottleneck node problem is common in usual

〈Table 1〉 The number of bottleneck nodes

Total nodes \ Area	100 x 100 m2	200 x 200 m2
100 nodes	7	13
200 nodes	5	8
300 nodes	2	6



(Fig. 1) Possibility to be a bottleneck node in different density

application. In order to prove the existence of bottleneck nodes in WSNs, we deployed the different number of sensor nodes in a different size area. <Table 1> shows the statistics of bottleneck nodes in our application with many experiments: (The communication radius of each node is 50m.)

(Fig. 1) indicates the high probability of bottleneck node existence in actual WSNs. So this problem is critical for the lifetime of the whole network and it is necessary to have much more concern in the bottleneck nodes.

3.2 Network scenario

Assume that N sensor nodes are randomly distributed in an $A*A$ square area and collect data periodically. The sink node is at one corner in the area. According to the actual design of our work, this WSN has the following features:

(a) Each node has a unique ID;
(b) All nodes move periodically;
(c) The accurate position is known.
(d) Cluster architecture is adopted for the network.

3.3 Problem description

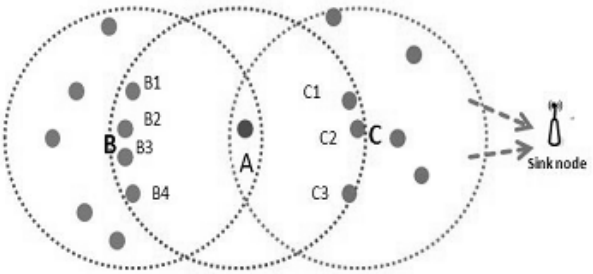
Definition1. Assume the transmission radius of node a is R. The neighbors of node a are the nodes whose Euclidean distance from a is less than R. Given two nodes a and d, if d is a's neighbor, then:

$$D(\alpha,\delta) < R$$

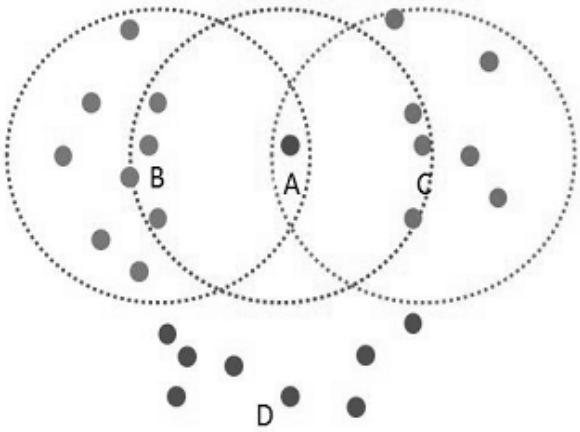
Here, $D(a,d)$ is the distance between a and d.

(Fig. 2) is an abstract model from real scenarios of WSNs. In this model, all the nodes have to transmit data to sink node. Both {B1, B2, B3, B4} in area B and {C1, C2, C3} in area C are A's neighbor nodes, but each neighbor node in area B cannot communicate with each neighbor node in area C directly. Instead, it can transmit the data through node A. Because node A is the only connection between area B and C, it can be regarded as a bottleneck node.

Meanwhile, another situation may happen in the network, as shown in (Fig. 3) There is another connection between area B and area C, like the nodes in



(Fig. 2) Bottleneck node model



(Fig. 3) Non-bottleneck node model

area D. Although they are not A's neighbor nodes, they can connect area B and area C, so node A is not a bottleneck node.

According to the problem description above, we can implement the bottleneck node detection mechanism with the following two steps:

(a) Bottleneck node candidate selection: The nodes acting as relay nodes to connect different partitions have to be detected.

(b) Bottleneck node confirmation: For each candidate, if it can be validated that it is the only relay between different partitions, it is a bottleneck node.

3.4 Proposed method

The implementation of DBDN algorithm can be divided into two phases: bottleneck node candidate selection and bottleneck node confirmation.

3.4.1 Bottleneck node candidate selection

First, every node in a WSN is requested to collect the location information of all its neighbor nodes. After the information collection, each node forms a neighborhood set.

Second, every node performs the algorithm described in (Fig. 4) and decides whether it can be a candidate of bottleneck nodes.

```
Create two set S1, S2;
Move all A's neighbour nodes into S2;
Select a node randomly from S2;
Move the selected node into S1;
for (node I in S1)
{
  for (node J in S2)
  {
    if (D(I,J)<R)
    {
      Move node J from S2 to S1;
    }
  }
}

if set S2 is not empty
{
  A is a candidate;
}
else
{
  A is not a candidate;
}
```

(Fig. 4) Process of candidate selection

3.4.2 Bottleneck node confirmation

After the candidate selection, some nodes out of the candidates are confirmed as the bottleneck nodes. As shown in (Fig. 2), if node A is the only relay connecting the different neighbor partitions, A is a bottleneck node. On the other hand, if there is another area D, as shown in (Fig. 3), which can also connect the two different partitions, node A is not a bottleneck node.

In this phase, the actual bottleneck nodes can be selected out of the candidate nodes with further confirmation. The confirmation process can be divided into two steps:

(a) First, after the candidate node selection, each candidate will notify its neighbors. As shown in (Fig. 2), A is selected as a candidate, and it notifies all its neighbors in area B {B1, B2, B3, B4} and area C {C1, C2, C3}.

(b) Second, the neighbor nodes will search the other possible ways to connect the different partitions, such as path B D C. In this step, one node in area B will be selected randomly to send a confirm message to a random node in area C. Within a certain time threshold T, if the given node in area C receives the message, node A cannot be a bottleneck node. Otherwise, node A is a bottleneck node.

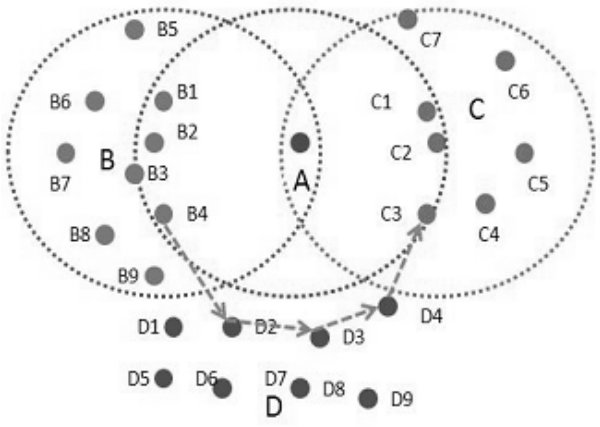
In order to reduce the hops and time of message transmission, the following rule will be used for relay node selection:

(a) For the message transmission, the source node will form a set of its neighbor nodes, such as {N1, N2, N3, …}, and it computes the distance between every neighbor node and the given destination node according to the known position, forming a distance set {d1, d2, d3, …}.

(b) The source node will choose the closest node to the destination as the next hop node.

(c) After the next hop node receives the message, it repeats the steps (a), (b) till the message arrives at the destination node.

For example, as shown in (Fig. 5), if node B4 is selected to transmit the message to a random node C3, node B4 will form a set of its neighbor nodes, {B1, B2, B3, B6, B7, B8, B9, D1, D2, D5}, and then it computes the distance from each neighbor node in the set to the destination node C3. Assuming node D2 is the closest to node C3, node D4 will choose node D2 as the next hop



(Fig. 5) Process of bottleneck node confirmation

node for transmission.

Next, node D2 will repeat the above steps, choosing D3 as the next hop till node C3 receives the message.

3.5 Algorithm analysis

3.5.1 Algorithm complexity analysis

In the centralized MINCUT algorithm, the complexity is $O(N^3)$ [5]. On the other hand, the complexity of the DBND algorithm is $O(n^2)$, where n is the number of neighbor nodes.

Assume that N nodes are deployed in an $A*A$ (meter) square area uniformly and the communication range of each node is R. The density of the node distribution is:

$$\lambda = \frac{N}{A^2} \tag{1}$$

For each node, the average number of neighbor nodes is:

$$n = \lambda \times \pi R^2 - 1 \tag{2}$$

The rate of the both algorithm complexity is

$$\frac{n^2}{N^3} = \frac{(\lambda \times \pi R^2 - 1)^2}{N^3} \tag{3}$$

If N=1000, R=100m and A=1000m,

$$\frac{n^2}{N^3} = 9 \times 10^{-9} \tag{4}$$

According to Equations (3) and (4), the algorithm complexity of DBND is much less than MINCUT.

3.5.2 The time for location information collection

For DBND in the location information collection phase, each node has to broadcast a message to its neighbors to gain all its neighbors' location information.

For MINCUT in this phase, the nodes that are not within a single hop range of a sink node have to broadcast a message to its neighbors. After receiving the message, the neighbor node will broadcast to their neighbors until the message arrives at a sink node. Besides each node broadcasting message to its neighbors, in MINCUT, most nodes have to forward a location message to sink nodes using multi hops, resulting in more collision and consuming more time.

4. Solutions for bottleneck nodes

4.1 Weakening the action of bottleneck nodes in WSNs

For energy efficiency in WSNs, the most efficient way is using the clustering algorithm [9, 11]. In a clustering network, the cluster head will consume much more energy than a non cluster head, meanwhile, each node in the network has the same priority to be a cluster head. In this case, the bottleneck nodes should be kept from being cluster heads.

After the bottleneck nodes detection, bottleneck nodes cannot be a cluster head forever. It can save the energy consumption for prolonging the lifetime of bottleneck nodes. So the nodes just connecting with the bottleneck nodes can benefit from the prolonged lifetime of bottleneck node.

4.2 Add more new nodes or use mobile nodes to relax the bottleneck problem

After the bottleneck detection, we can deploy some new nodes or move some mobile nodes [12] to the bottleneck area in special application, such as port logistics, in order to achieve relaxation of bottleneck problem.

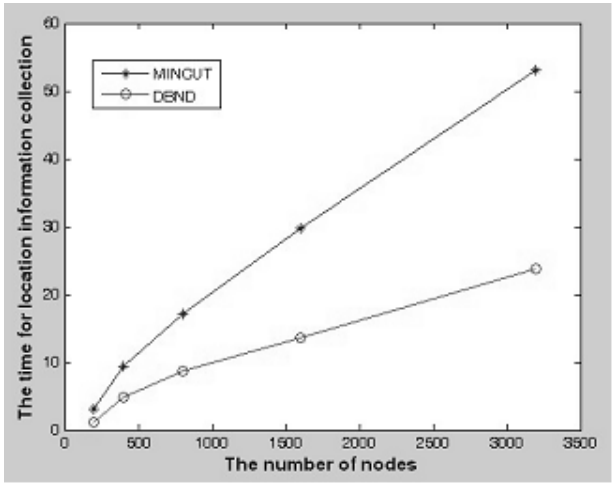
5. Algorithm Performance Evaluation

For both MINCUT and DBND algorithms, the execution time includes two parts: (a)

the time for location information collection. (b) The time for bottleneck node detection. We compare the execution time in each part of both algorithms.

5.1 The time for location information collection.

(Fig. 6) compares the time for location information

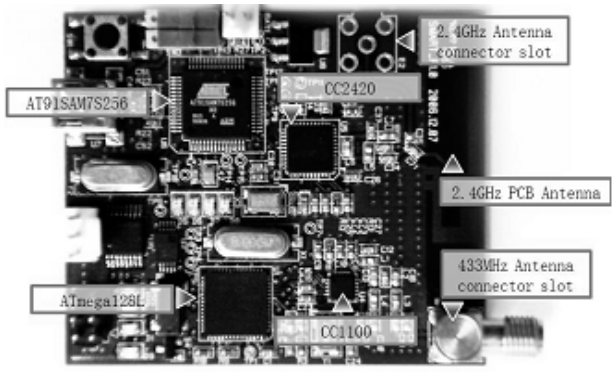


(Fig. 6) The time for location information collection

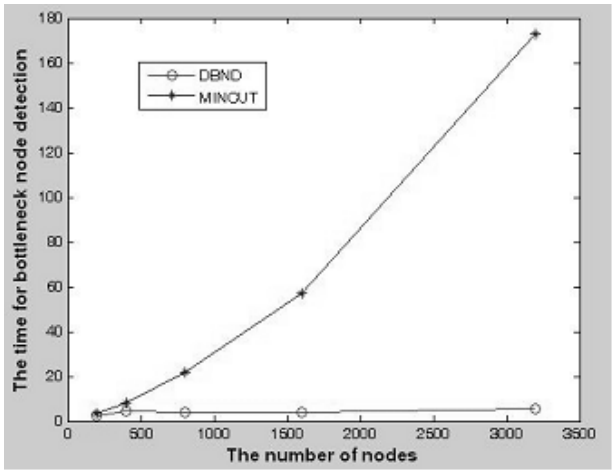
collection against different numbers of sensor nodes. As shown in the figure, the difference of the implementation time between DBND and MINCUT becomes larger with more sensor nodes deployed in the network. The reasons can be: In MINCUT, it has to collect the global information and the location messages of further nodes hs of oinfosent to the sink node by multi hops. This isormmessage flooding course with a high cost. In DBND, for each node, it just needs the location information of its neighbors and also it just sends its own location information to its single hop neighbors without multi hops. It can decrease the cost of the flooding course in MINCUT. Of course it can save the time for location information collection.

5.2 Time for bottleneck node detection

To evaluate the performance of the algorithm, we implemented the simulation using MATLAB. All sensor node parameters for simulation come from the hardware in (Fig. 7) The computing chip is ATmega128L (CPU:



(Fig. 7) Hardware architecture



(Fig. 8) The time for bottleneck node detection

8MHz and Memory: 4KB)

The other network parameters are as following:

Network area: 1000*1000 m^2

The number of sensor nodes: 200, 400, 800, 1600, 3200.

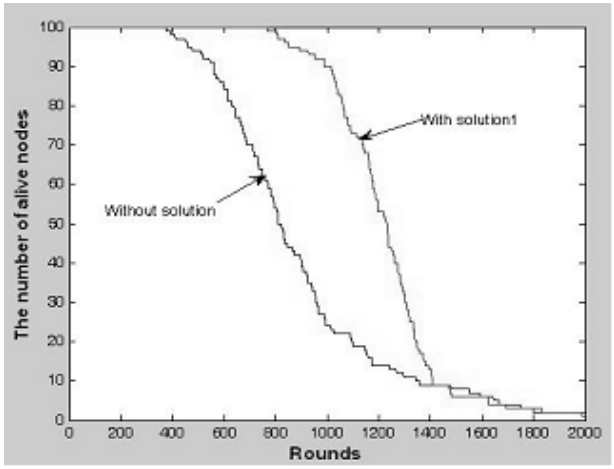
(Fig. 8) compares the time for bottleneck node detection against different numbers of sensor nodes. As shown in the figure, the proposed method, DBND, can detect bottleneck nodes within a mun ashorter time than MINCUT. The reason is: the algorithm complexity of DBND is mun aless than MINCUT. Especially, with the incis: d number of nodes in the network, the execueneck ime of DBND is chang d osed aalittle due to the tiny chang in the number of n ashort nodes. On the other hand, the execueneck ime of MINCUT incis: s very faed, becau the algorithm complexity is proportional to $N^{3.}$

6. The performance of proposed solutions for bottleneck nodes

In order to prove our proposed solution can be used for relaxing the bottleneck nodes problem, we deploy some bottleneck partitions in the network.

6.1 The performance of solution I

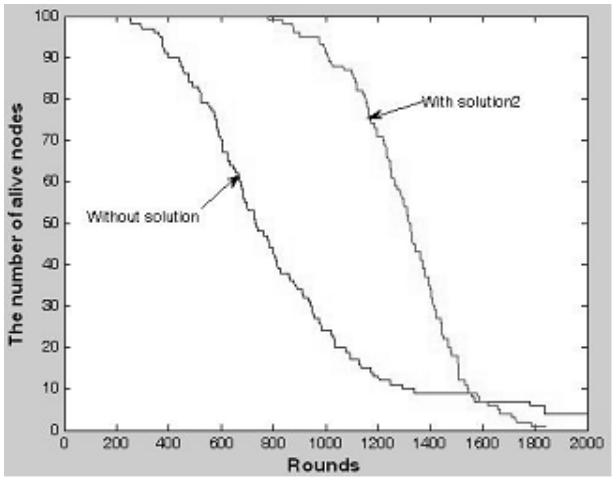
(Fig. 9) compares the number of live nodes with communication rounds. As shown in the figure, we can overcome the bottleneck node problem and prolong the lifetime of the whole network two times more than before. The reason is: in each round of communication, the bottleneck node just acts as a relay. It can save the energy to prolong the whole lifetime.



(Fig. 9) The performance of solution I

6.2 The performance of solution II

(Fig. 10) also compares the number of live nodes with communication rounds. As shown in the figure, we can prolong the lifetime 150% more than before. The reasons is : when we find out the bottleneck node areas, we place some sensor nodes to help the bottleneck nodes, it wil tlower the load of the bottleneck nodes, and balance the distn bution of al tsensor nodes. So it can prolong the lifetime of the whole network through even load distribution.



(Fig. 10) The performance of solution II

7. Conclusion

In this paper, we addressed the effect of the bottleneck nodes in the wireless sensor network, and focused on how to find the bottleneck nodes quickly. The MINCUT algorithm has been widely used for bottleneck nodes detection, but its computational complexity is very high, being impractical for a huge network. On the other hand, our distributed algorithm, DBND, can improve the complexity as compared with the centralized MINCUT algorithm, working for bottleneck node detection is practical even for a huge network. We also give two simple solutions for the problem. It can help the network overcome the fault and prolong the lifetime of the whole network.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A Survey on Sensor Networks,” *IEEE Communications Magazine*, Vol.40, No.8, pp.102-114, Aug., 2002.

[2] Y. Yoo and D. P. Agrawal, “Mobile Sensor Relocation to Prolong the Lifetime of Wireless Sensor Networks,” in *Proc. IEEE VTC Spring* 2008, pp.193-197.

[3] X. Ji and H. Zha, “Sensor Positioning in Wireless Ad hoc Sensor Networks Using Multidimensional Scaling,” in *Proc. IEEE INFOCOM*, 2004, pp.2652 - 2661.

[4] F. Y. S. Lin and P. L. Chiu, “A Near Optimal Sensor Placement Algorithm to Achieve Complete Coverage/ Discrimination in Sensor Networks,” *IEEE Communications Letters*, Vol.9, No.1, Jan., 2005, pp.43-45.

[5] Mechthild Stoer, Frank Wagner, “A Simple Min Cut Algorithm,” Journal of the ACM, Vol.44, No.4, July, 1997, pp.585 - 591.

[6] Lizhi Xu et al., “Computer Mathematics of Modern Mathematics Handbook,” Huazhong University of Science and Technology Press, P539.

[7] Ningning Hu, Li (Erran) Li, Zhouqing Morley Mao, Peter Steenkiste and Jia Wang,“A Measurement Study of Internet Bottlenecks,” in *Proc. IEEE INFOCOM* 2005, pp.1689-1700.

[8] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance”. *IEEE/ACM Transactions on Networking*, Vol.1, No.4, pp.397-413, Aug., 1993.

[9] Heinzelman W. Application Specific protocol architectures for wireless networks [Ph.D. Thesis]. Boston: Massachusetts Institute of Technology, 2000.

[10] Handy MJ, Haase M, Timmermann D. Low energy adaptive clustering hierarchy with deterministic cluster head selection, In: Proc. of the 4th IEEE Conf, on Mobile and Wireless Communications Networks. Stockholm: IEEE Communications Society, 2002

[11] Haosong Gou, Gang Li and Younghwan Yoo, “A Partition Based Centralized LEACH Algorithm for Wireless Sensor Network Using solar Energy,” in Proceedings of the International conference on Convergence & Hybrid

Information Technology 2009 (ICHIT 2009).
[12] Al Karaki JN, Ul Mustafa R, Kamal AE. Data aggregation in wireless sensor networks—Exact and approximate algorithms. In: Proc. of the IEEE Workshop on High Performance Switching and Routing. Phoenix: IEEE Communications Society, pp.241-245, 2004.
[13] Younis O, Fahmy S. Heed: A hybrid, energy efficient, distributed clustering approach for ad hoc sensor networks. IEEE Trans. onMobile Computing, 3(4):660-669, 2004.

Haosong Gou
e-mail : gouhaosong@pusan.ac.kr
2007년 화중사범대학교 컴퓨터공학과(학사)
2008년～현 재 부산대학교 컴퓨터공학과 석사과정
관심분야 : WSN, NGN, Wireless Network 등

김 진 환
e-mail : compunix@pusan.ac.kr
2008년 동명대학교 컴퓨터공학과(학사)
2008년～현 재 부산대학교 컴퓨터공학과 석사과정
관심분야 : WSN, RFID/USN, NGN, Embedded System 등

유 영 환
e-mail : ymomo@pusan.ac.kr
1996년 서울대학교 컴퓨터공학과(학사)
1998년 서울대학교 컴퓨터공학과(공학석사)
2004년 서울대학교 컴퓨터공학과(공학박사)
2004년～2006년 신시내티대학교 전기컴퓨터공학부 연구원
2007년～현 재 부산대학교 정보컴퓨터공학부 교수
관심분야 : 센서네트워크, RFID/USN, 애드혹통신 등