Distributed and Self-adaptive Cluster-head Selection Algorithm for Hierarchical Wireless Sensor Networks

Sai Ji [1,2,*], Liping Huang [1], Chang Tan [1], Jin Wang [1]

[1] Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science and Technology, 219# Ningliu Road, Nanjing, China, 210044
[2] The Aeronautic Key Laboratory for Smart Materials and Structures, Nanjing University of Aeronautics and Astronautics, 29# Yu Dao Street, Nanjing, China, 210016
jisai@nuist.edu.cn, {hlpwhy, passerby.tan}@gmail.com, wangjin@oslab.khu.ac.kr

Abstract. In the hierarchical wireless sensor network (WSN), selecting cluster head (CH) is important issue to increase the network energy efficiency, scalability and lifetime. For the sake of balancing energy expenditure of sensor nodes and improving the performance of routing, We propose a distributed and self-adaptive cluster-head selection algorithm. Based on the hierarchical agglomerative clustering (HAC) method, the algorithm uses the qualitative connectivity data as input data, and tailor simple numerical methods to generate a cluster tree. From such clustering sequence, the CH and the backup CH can be quickly selected without extra message exchanges. Simulation results demonstrate that this method is effective and self-adaptive, which can enhance network self-control capability and resource efficiency, and prolong the whole network lifetime.

Keywords: Wireless sensor networks; cluster head selection; hierarchical agglomerative clustering; backup cluster head

1 Introduction

In the last few years there has been a growing interest in small, low-power hardware platforms that integrate sensing, processing data and wireless communication capabilities. These devices are called sensor nodes and are grouped to form a Wireless Sensor Network (WSN). A WSN has application in environmental monitoring, infrastructure management, transportation and many others [1, 2]. The hierarchical network architecture of WSN shows its advantages on sharing limited wireless channel bandwidth, balancing node energy consumption, enhancing management, and so on[3]. The hierarchical routing protocols can be classified into two categories: random-selected-CH protocol and well-selected-CH protocol. The representative random-selected-CH protocols are: LEACH [4] and HEED[5]. LEACH-C [6] and AHP [7] are well-known well-selected-CH protocols.

The random-selected-CH protocols have two main disadvantages. Firstly, the randomly picked CH may have a higher communication overhead. Secondly, the periodic CH rotation or election which needs extra energy to rebuild clusters. To avoid

the problem of random CH selection, the approach of well-selected-CH has considered three factors: energy, mobility, and the better cluster quality. However, they usually have a more complex scheme and higher overhead to optimize the CH selection and cluster formation.

In this paper, we propose a distributed HAC (DHAC) routing algorithm for wireless sensor networks. One of the most commonly accepted method, UPGMA, is used to make clustering decision in this paper. The qualitative one-hop connectivity information is adopted as input data, which can be easily obtained through message transmission with low or no extra communication cost. Simulations have validated its effectiveness.

2 Distributed hierarchical agglomerative clustering

2.1 DHAC Inrtroduction and Notations Definitions

Hierarchical agglomerative clustering (HAC)[8] is a conceptually and mathematically simple clustering approach which uses four clustering methods, CLINK, SLINK, UPGMA, and WPGMA. Recently, the most research does focus on the clustering technique analysis and comparison. All of these methods comprise three common key steps: obtain the data set, build the similarity matrix, and execute the clustering algorithm. Based on the concept of HAC, we propose a DHAC method for distributed environments by improving the HAC algorithms. The main idea behind DHAC is that a node only needs one-hop neighbor knowledge to build clusters. To apply the DHAC algorithm in WSNs, we present a bottom-up clustering approach by simple six steps. Firstly, the qualitative connectivity data is obtained as input data set for DHAC. Secondly, the similarity matrix is built. Thirdly, the similar nodes are grouped together by executing the distributed clustering algorithm. The last three steps are cutting the cluster tree with the threshold, merging the smaller cluster, and electing the CHs. The process of all steps is illustrated in the following sections. Figure 2 and Figure 3 illustrate the pseudo code of the DHAC implementation for WSNs. Table 1 summarizes the notations we will use in our discussion.

Table 1. Summary of notations

Symbol	Definition
Simi_Matrix	Similarity Matrix
Node_Id	Node Id
Ch_Id	Cluster Head(CH) Id
Min_Coeff	The minimum coefficient in the Similarity Matrix
Min_Coeff_Id	the cluster(CH_Min) Id corresponding to Min_Coeff
T	The threshold of *Min_Coeff*
C_{size}	The number of cluster member in a cluster
Min_Cluster_Size	The threshold of minimum cluster size

2.2 Input Data Set

In this paper, DHAC can use simple qualitative connectivity information of a network or quantitative data through received signal strength or GPS. The quantitative could be the location of each node, the nodes' residual energy, or other features. Either qualitative data or quantitative data is the properties of the sensor node, and the nodes with similar properties can be crusted together. For simplicity and without loss of generality, we use the qualitative connectivity information as the input data set for DHAC.

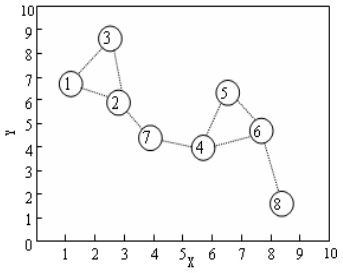

Fig. 1. A simple 8-node network.

```
1.   procedure obain_local_input_data ()
2.     Send HELLO, Node_Id to 1-hop neighbors;
3.     if (isHELLOReceived==false)
4.       Keep listening to neighbors;
5.     else
7.       Build local data with (sender's Node_id, 1)
8.     endif
9.   end procedure
10.  procedure simi_matrix ()
11.  Ch_Id=Node_id;
12.    send AskLocalData and its local data to
         direct connected neighbors;
13.  if (isAskLocalDataReceived==false)
14.      Keep listening to neighbors;
15.  else
16.     Obtain sender's data;
17.    Establish Simi_Matrix via Dice coefficient;
18.  endif
19. end procedure
```

Fig. 2. Pseudo code of distributed HAC (a).

2.3 Build the Similarity Matrix

To set up the local similarity matrix, in figure 2, lines (11-16), each node elects itself as a CH(cluster head) and send *AskLocalData* message to its direct connected neighbors. Then node keeps listening until accepted the senders' local qualitative connectivity data. After obtained the qualitative input data, the Similarity matrix could be built (Figure 2, line 17), and there are three typical methods [9] to calculate the similarity coefficient for qualitative data: Dice (Sorenson's) coefficient, Jaccard coefficient and Simple Matching coefficient. The Dice coefficient between node {a} and {b} can be formulated as $S_{a,b} = 1 - \frac{2C}{N_a + N_b}$, Where C is number of positive matches between nodes {a} and {b} in input data. N_a is total number of "1" value filled in the node {a}'s local table that are directly connected. The calculation principle of N_b is similar to that used in N_a.

2.4 Executing the Distributed Clustering Algorithm

After building the similarity matrix. Each node takes itself as the cluster head(CH) and obtains its own local resemblance matrix, from which its minimum coefficient(*Min_Coeff*) can be easily found. In table 1, we name the ID of CH as *Ch_Id,* and define the cluster(CH_Min) Id corresponding to *Min_Coeff* as *Min_Coeff_Id.* If *Ch_Id* is smaller than *Min_Coeff_Id*, then CH sends *AsktoMerge* message to its CH_Min for merging themselves together, otherwise CH does nothing and just waiting. In figure 3, lines 3-11 show the process of sending message.

```
1.  procedure execute_DHAC ()
2.    do   {
    /*---- Distributed Sending Message----*/
3.     if (Ch_Id== Node_Id)
4.          Find minimum coefficient in
            simi_matrix   to   assign to
            variable Min_Coeff;
5.          Set Node_id with Min_Coeff   to
            CH_Min;
6.     if   (Ch_Id< Min_Coeff_Id)
7.          Send AsktoMerge message to
            CH_Min;
8.    else
9.          CH keeps waiting;
10.   endif
11.   endif
   /*---- Distributed Receiving Message----*/
12.   if (isAsktoMergeReceived==true)
13.   if (Min_Coeff_Id ==sender's CH_Id)
14.         Faceback CONFIRM message;
15.         CH_ID=sender's CH_Id;
16.   else
17.         Faceback DENY message;
18.   endif
```

```
19.   elseif (isDENYReceived==true )
20.         CH stops sending AsktoMerge
            message until Simi_Matrix
            refreshed;
21.   elseif (isCONFIRMReceived==true)
22.          do merge_clusters();
23.   elseif(isREFRESHReceived==true)
            Update the Simi_Matrix;
24.   endif }
25.   while(Min_Coeff<T)
    /*--- Control the minimum cluster size--*/
26.         caculate the cluster's member
            number to C_size;
27.   if (C_size<Min_Cluster_Size)
28.          do merge_clusters();
29.   endif
30.   end procedure

31. procedure merge_clusters()
32.    merge two clusters;
33.    blend local information of two
       clusters;
34.    update Simi_Matrix by using UPGMA
       method
35.    broadcast REFRESH message;
36. end procedure
```

Fig. 3. Pseudo code of distributed HAC (b).

In distributed clustering algorithm, another role action of node is receiving message(figure 3, lines 12-18). When a cluster head(CH) receives a *ASKtoMerge* message, it compares the sender's cluster head id with its *Min_Coeff_Id.* If they are just the same, then the CH facebacks a message to the source node to confirm the merging condition and elects the source to be the new CH. Otherwise, the CH sends back a DENY message. When a cluster receives the CONFIRM message from another cluster, it goes to merge the two clusters into a new cluter (figure 3, lines 31-36). At the same time, local similarity matrix and neigbor list are combined together, and the similarity matrices of two clusters are updated through the chosen HAC algorithm. CLINK, SLINK, UPGMA, and WPGMA are four main types of the HAC algorithm methods. Among them, un-weighted pair-Group Method (UPGMA) is the most commonly adopted clustering method. This defines the similarity measure between two clusters as the arithmetic average of resemblance coefficients among all

pair entities in the two cluster. After a new cluster is formed, the CH broadcasts a REFRESH message to notify its neighbors to update their similarity matrices. Clusters update their own similarity matrix after receiving this REFRESH message, which contains the new cluster information and the merged neighbor list. Once a CH receives a DENY message from its CH_Min, the CH stops sending the AsktoMerge Message until its similarity matrix has been uptated.

Since the qualitative connectivity data are very simple, and a few of *Min_Coeff* in initial local similarity matrix is usually very small. A do-while loop is used to ensure clustering process to be executed at least once. This process will repeat until the while condition (line 25).

2.5 The Last Three Steps of DHAC

The last three steps are cutting the cluster tree with the threshold, merging the smaller cluster, and electing the CHs. After generating a cluster tree, a pre-configured threshold *T* (figure 3, lines 2-25) is used in do-while loop to controls the upper bound size of clusters. The predefined threshold can be transmission radius, number of clusters, or cluster density. If the cluster size is less than a pre-defined threshold, *Min_Cluster_Size,* merge the cluster with its closest cluster (figure 3, lines 26-29). To select the appropriate CHs in clustering tree or clustering sequence, DHAC simple chosse the nodes which satisfy two conditions: (1) the node is one of two nodes which are merged into the cluster at the first step. (2) the node with the lower ID. Another node which has the higher ID becomes the backup CH.

3 Simulation and performance evaluation

In this section, we evaluate the performance of DHAC algorithm implemented in Ns-2 simulator . For increasing comparability, most parameters are similar to [10]. Each node is equipped with an omni-directional antenna. Computer simulation is carried out in a sensor network, where 400 sensor nodes are deployed randomly in a rectangular region of size 400×400 units. The sink node is located at the center of netwok. Next, we will present the performance comparison among the proposed DHAC and LEACH protocols. We use two metrics to analyze and compare our simulation results for clustering and energy saving: network lifetime and cluster energy dissipation. Here we use node death rate versus the rounds of clustering to represent network lifetime.

As figure 4 and figure 5 shown, the performance of DHAC is much better than LEACH. In figure 4, The clustering dissipated energy of DHAC is about 4 times less than that of LEACH which predicates that DHAC can achieve much high reliability and efficiency in energy consumption at 300 rounds timing. In figure 5, LEACH has the shortest network: the number of clustering rounds is 300 and only 50% of the nodes are alive. Compared to LEACH, DHAC prolongs it by 25%.

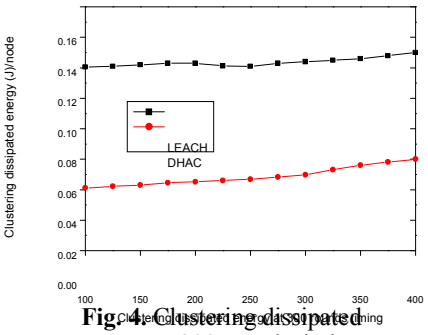

Fig. 4. Clustering dissipated energy at 300 rounds timing.

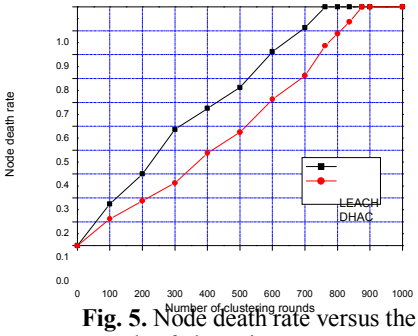

Fig. 5. Node death rate versus the rounds of clustering.

4 Conclusions

In this paper, we have proposed a distributed approach, DHAC, to classify sensor nodes into appropriate groups in stead of simply gathering nodes to some randomly selected CHs. We demonstrated the application and evaluation method, UPGMA, with qualitative data. Simulation results demonstrate that this method is effective and self-adaptive, which can enhance network self-control capability and resource efficiency, and prolong the whole network lifetime. In our future work, we will evaluate the cluster quality with different HAC methods, SLINK, CLINK, and WPGMA.

Acknowledgements. This work was supported by the Natural Science Foundation of the Jiangsu Higher Education Institutions of China(Grant No. 11KJB520011) and the PAPD.

References

1. T. Nagayama, M. Ushita, Y. Fujino, Suspension Bridge Vibration Measurement Using Multihop Wireless Sensor Networks. Procedia Engineering, 2011, 14:761-768
2. He Shibo, Chen Jiming, Sun Youxian. Coverage and Connectivity in Duty-Cycled Wireless Sensor Networks for Event Monitoring. IEEE Transactions on Parallel and Distributed Systems, 2012, 23(3):475-482
3. J.N. Al-Karaki, A.E. Kamal, Routing techniques in wireless sensor networks: a survey, IEEE Wireless Commun. 11 (6) (2004) 6–28.
4. W. Heinzelman, A. Chandrakasan,H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, in: Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS'00), January 2000.
5. O. Younis and S. Fahmy "Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach", Proc. of IEEE INFOCOM, March 2004 pp. 629-640.
6. W.B. Heinzelman, A. Chandrakasan, H. Balakrishnan, An application specific protocol architecture for wireless microsensor networks, IEEE Transactions on Wireless Communications 1 (4) (2002) 660–670.

7. Y. Yin, J. Shi, Y. Li, P. Zhang, Cluster head selection using analytical hierarchy process for wireless sensor networks,in:Proceedings of IEEE 17th International Symposium PIMRC,Helsinki,Finland,2006,pp.1–5.

8. M.R. Anderberg, Cluster Analysis for Applications, Academic Press Inc., New York, 1973.

9. O. Younis and Sonia Fahmy, "Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach", in Proceeding of ACM MobiCom 2003, San Diego, California USA, Sep.14-19, 2003.