A Distributed and Energy-efficient Clustering Method for Hierarchical Wireless Sensor Networks

Sai Ji[1,2,3], Liping Huang[1,2] and Jin Wang[1,2]

[1]*Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science & Technology, Nanjing, 210044, China*
[2]*College of Computer & Software, Nanjing University of Information Science & Technology, Nanjing, 210044,China*
[3]*The Aeronautic Key Laboratory for Smart Materials and Structures, Nanjing University of Aeronautics and Astronautics, 29# Yu Dao Street, Nanjing, 210016,China*
jisai@nuist.edu.cn, hlpwhy@gmail.com, wangjin@nuist.edu.cn

Abstract

In the hierarchical wireless sensor network (WSN), cluster-based network architecture can enhance network self-control capability and resource efficiency, and prolong the whole network lifetime. Thus, clustering has also been a topic of interest in many different disciplines. Finding an energy-effective and efficient way to generate cluster is very important in WSN. We propose a distributed, energy-efficient and flexible clustering approach. Based on the hierarchical agglomerative clustering (HAC) method, the algorithm uses the qualitative connectivity data as input data, and tailor simple numerical method to generate a cluster tree. Simulation results demonstrate that this method is energy-effective and flexible, 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

Wireless sensor networks (WSNs) are composed of large number of low-cost, low-power, and multifunctional sensor nodes. It is a distributed self-organizing and data-centric network. Those tiny sensor nodes are deployed randomly in inaccessible terrains or disaster relief environments. Sensor nodes will use their limited abilities to locally carry out simple computations and transmit the monitored data to the sink node (SN) in an autonomous and unattended manner. WSNs have many broad applications such as military surveillance and tracking, environment monitoring and forecasting, healthcare as well as smart home *etc.*, [1, 2]. Some of these applications require a large number of devices in the order of tens of thousands nodes. Due to the small dimensions, sensor nodes have strong hardware and software restrictions in terms of processing power, memory capability, power supply, and communication throughput. For this reason the hierarchical network architecture shows its advantages on sharing limited wireless channel bandwidth, balancing node energy consumption, enhancing management, and so on.

Hierarchical or cluster-based routing, are well-known techniques with special advantages related to scalability and efficient communication [3-6]. As such, the concept of hierarchical routing is also utilized to perform energy-efficient routing in WSNs. In a hierarchical network,

similar nodes aggregate into clusters. In each cluster, one node acts as a CH (Cluster Head), and the rest of the nodes are grouped into the cluster members. The CH node takes charge of communicating with its cluster members and other CHs. The cluster members just need to transmit message to their CH. By the method of CH selection, the hierarchical routing protocols can be classified into two categories: random-selected-CH protocol and well-selected-CH protocol. The former randomly selects CHs and then rotates the CH task among all nodes, while the latter carefully selects appropriate CHs and then gathers nodes under the CHs based on the network status. The representative random-selected-CH protocols are: LEACH [7], PEGASIS [8] and HEED [9]. LEACH-C [10] and AHP [11] are well-known well-selected-CH protocols.

Although random-selected-CH protocols can bring more flexibility and toleration, these approaches have two main disadvantages. Firstly, the randomly picked CH may have a higher communication overhead because it has no knowledge of intra-cluster or inter-cluster communication. 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. This paper is organized as follows. In Section 2, the distributed hierarchical agglomerative clustering is introduced. To apply the DHAC algorithm in WSNs, we proposed six-step clustering to generate appropriate clusters, obtaining the data set, computing the resemblance coefficients, executing the clustering method, cutting the cluster tree with the threshold, merging the smaller cluster, and electing the CHs. The performance of the DHAC for WSNs is evaluated by simulation and experiment in Section 3. Concluding remarks and future works are given in Section 4.

2. Distributed Hierarchical Agglomerative Clustering

2.1 DHAC Introduction and Notations Definitions

Hierarchical agglomerative clustering (HAC)[12, 13] 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.

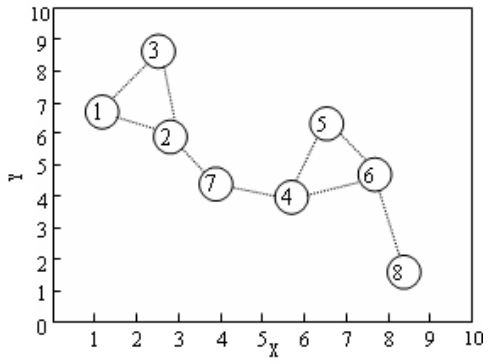

Figure 1. A simple 8-node network

Figure 1 shows a simple randomly generated 8-node network. Each node in this network merely knows its neighbors list without any other information. We use a 1 value to represent 1-hop connection and a 0 value to represent no direct connection. To collect input data, in Figure 2, lines 1-9, each node sends *HELLO* messages to its neighbors, and keeps listening until the node has received the neighbors' feedback information. The returning information is saved as input data in local table. For instance, the 1-hop neighbor nodes of {1} are {2}, {3} and itself. Hence, the input data for 8-node could be illustrated as Table 2.

Table 2. One-hop network connectivity input data set

	{1}	{2}	{3}	{4}	{5}	{6}	{7}	{8}
{1}	1	1	1	0	0	0	0	0
{2}	1	1	1	0	0	0	1	0
{3}	1	1	1	0	0	0	0	0
{4}	0	0	0	1	1	1	1	0
{5}	0	0	0	1	1	1	0	0
{6}	0	0	0	1	1	1	0	1
{7}	0	1	0	1	0	0	1	0
{8}	0	0	0	0	0	1	0	1

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 [13] 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 $S_{a,b}$:

$$S_{a,b} = 1 - \frac{2C}{N_a + N_b} \quad (1)$$

Where C is the 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.

In Table 3, the similarity matrix is the Dice coefficient which is based on the connectivity information from one-hop neighbors. For example, node {1} has two neighbors, {2} and {3}. The Dice coefficient between node {1} and {2} is $S_{1,2}$ =1-(2*3)/(3+4)=0.14, and another directly connected neighbor's coefficient is $S_{1,3} = 0.00$.

Table 3. Initial local similarity matrix with one-hop qualitative data using Dice coefficient

	{1}	{2}	{3}	{4}	{5}	{6}	{7}	{8}
{1}		0.14	0.00					
{2}	0.14		0.14				0.43	
{3}	0.00	0.14						
{4}					0.14	0.25	0.43	
{5}				0.14		0.14		
{6}				0.25	0.14			0.33
{7}		0.43		0.43				
{8}						0.33		

```
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
```

Figure 2. Pseudo code of distributed HAC (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. For instance, in Table 3, cluster{1} sends AsktoMerge message to cluster{3} while cluster{8} just keeps waiting. In Figure 3, lines 3-11 show the process of sending message.

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. For example, node {3} is the initial CH in cluster{3}, as shown in Table 3, its cluster members include node {1} and node {2}, its *Min_Coeff_Id* is {1}. When it receive the *ASKtoMerge* from cluster{1}, cluster{3} sends a CONFIRM messeage back to cluster{1}.

```
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
```

Figure 3. Pseudo code of distributed HAC (b)

Next, after cluster{1} receiving the CONFIRM message from cluster{3}, it goes to merge the two clusters into a new cluter with node id {1} as its new *CH_ID*(Figure 3, lines 31-36). At the same time, local similarity matrix and neighbor 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 clusters.

$$C_{UPGMA} = \frac{1}{mn} \sum_{i=1,j=1}^{m,n} C_{(i,j)} \qquad (2)$$

When 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 updated.

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). Table 4 shows the updated similarity matrices of new clusters using UPGMA with qualitative data after the first round of the DHAC algorithm execution.

Table 4. The first round of DHAC: updated similarity matrix using UPGMA method with qualitative data

	{1,3}	{2}	{4,5}	{6}	{7}	{8}
{1,3}		0.14				
{2}	0.14				0.43	
{4,5}				0.20	0.43	
{6}			0.20			0.33
{7}		0.43	0.43			
{8}				0.33		

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 chose 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.

This clustering sequence of the node in each cluster also can be used to handle the dynamic network conditions. For example, if a CH in the corresponding clustering sequence fails, the next node in the cluster chain will be the new default CH without extra message exchanges due to the fact that each cluster member has the CH backup chain information. With this mechanism, there is no need to re-execute clustering once we established cluster by using DHAC.

3. Simulation and Performance Evaluation

In this section, we evaluate the performance of DHAC algorithm implemented in Ns-2 simulator [14]. For increasing comparability, most parameters are similar to [15]. 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 network.

Next, we will present the performance comparison among the proposed DHAC and LEACH protocols. LEACH and is a typical random selected-CH protocol. 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%.

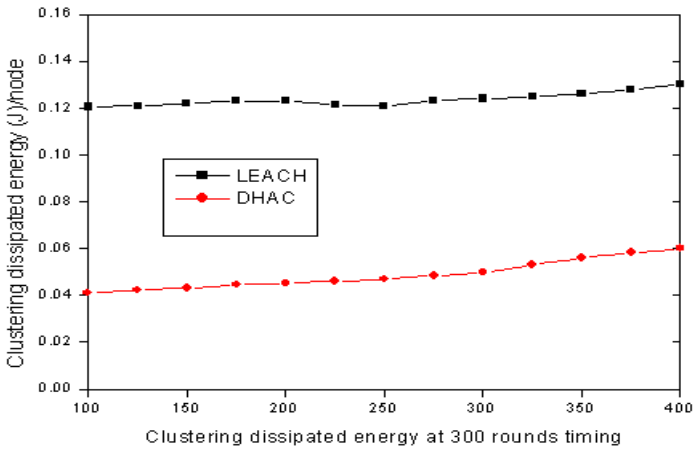

Figure 4. Clustering dissipated energy at 300 rounds timing

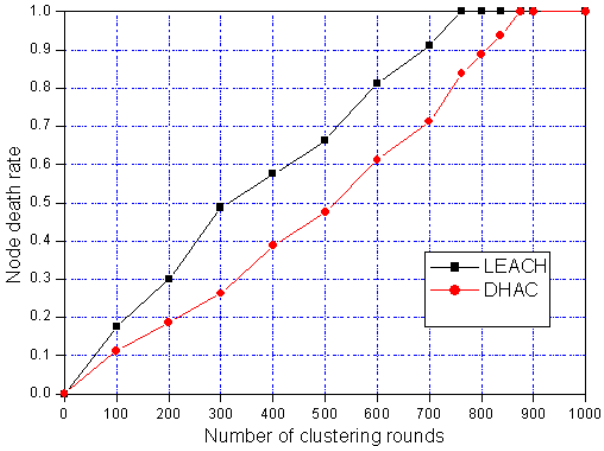

Figure 5. Node death rate versus the rounds of clustering

4. Conclusions

Clustering has generated a lot of discussion in WSNs. Clustering has also been a topic of interest in many different disciplines for a long time. Many clustering methods have been successfully used in other application areas. In this paper, we have proposed a distributed approach, DHAC, to classify sensor nodes into appropriate groups instead 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.

Acknowledgments

This work was a project supported by the Natural Science Foundation of the Jiangsu Higher Education Institutions of China (Grant No. 11KJB520011) and a Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions. It was also supported by the Industrial Strategic Technology Development Program (10041740) funded by the Ministry of Knowledge Economy (MKE) Korea, and by the Natural Science Foundation of Jiangsu Province (No. BK2012461).

References

[1] B. R. Badrinath, M. Srivastava, K. Mills, J. Scholtz and K. Sollins, "Special issue on smart spaces and environments", IEEE Personal Communications, **(2000)** October.

[2] S. Lindsey, C. Raghavendra and K. Sivalingam, "Data gathering in sensor networks using the energy delay metric", In International Workshop on Parallel and Distributed Computing: Issues in Wireless Networks and Mobile Computing, San Francisco, USA, **(2001)** April.

[3] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: a survey", IEEE Wireless Commun., vol. 11, no. 6, **(2004)**, pp. 6–28**.**

[4] R. Sheikhpour, S. Jabbehdari and A. khademzadeh, "A Cluster-Chain based Routing Protocol for Balancing Energy Consumption in Wireless Sensor Networks", International Journal of Multimedia and Ubiquitous Engineering(IJMUE), vol. 7, no. 2, **(2012)** April, pp. 1-16**.**

[5] C. S. Nam, K. S. Jang, G. S. Choi and D. R. Shin," Study on Use of a Clustering Technique with Zone-Based Multi-hop Communication in Wireless Sensor Networks", International Journal of Smart Home(IJSH), vol. 6, no. 1, **(2012)** January, pp. 65-70.

[6] J. Zhang, C. K. Jeong, G. Y. Lee and H. J. Kim, "Cluster-based Multi-path Routing Algorithm for Multi-hop Wireless Network", International Journal of Future Generation Communication and Networking(IJFGCN), vol. 1, no. 1, **(2008)** December.
[7] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks", in Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS'00), **(2000)** January.
[8] S. Lindsey and C. S. Raghavendra, "PEGASIS: power efficient gathering in sensor information systems", in Proceedings of the IEEE Aerospace Conference, BigSky, Montana, **(2002)** March.
[9] O. Younis and S. Fahmy, "Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach", Proc. of IEEE INFOCOM, **(2004)** March, pp. 629-640.
[10] W. B. Heinzelman, A. Chandrakasan and H. Balakrishnan, "An application specific protocol architecture for wireless microsensor networks", IEEE Transactions on Wireless Communications, vol. 1, no. 4, **(2002)**, pp. 660–670.
[11] Y. Yin, J. Shi, Y. Li and 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.
[12] M. R. Anderberg, "Cluster Analysis for Applications", Academic Press Inc., New York, **(1973)**.
[13] H. C. Romesburg, "Cluster Analysis for Researchers", Krieger Publishing Company, Malabar, Florida **(1990)**.
[14] http://www.isi.edu/nsnam/ns/.
[15] A. Akhtar, A. A. Minhas and S. Jabbar, "Energy Aware Intra Cluster Routing for Wireless Sensor Networks", International Journal of Hybrid Information Technology (IJHIT), vol. 3, no. 1, **(2010)** January, pp. 29-48.

Authors

Sai Ji Dr. Sai Ji is an associate professor in Computer Sciences at Nanjing University of Information Science & Technology, China. He received his Bachelor (NUIST, China, 1999), Master (NUAA, China, 2006). His research interests are in the areas of Data Mining, Computer Measurement and Control and Wireless sensor networks. He has published more than 20 journal/conference papers. He is principle investigator of three NSF projects.

Liping Huang Liping Huang is a graduate student in Nanjing University of Information Science & Technology currently. She received her bachelor's degree in Network Engineering from Nanjing University of Information Science & Technology in 2007. Her research interest includes network security, mobile computing, machine learning and WSNs.

Jin Wang Dr. Jin Wang received the B.S. and M.S. degree in the Electronical Engineering from Nanjing University of Posts and Telecommunications, China in 2002 and 2005, respectively. He received Ph.D. degree in the Ubiquitous Computing laboratory from the Computer Engineering Department of Kyung Hee University Korea in 2010. Now, he is a professor in the Computer and Software Institute, Nanjing University of Information Science and technology. His research interests mainly include routing protocol and algorithm design, performance evaluation and optimization for wireless ad hoc and sensor networks. He is a member of the IEEE and ACM.