A strategy for fault management in LDC wireless sensor network

Jian Yang[1, 2, 3]*

[1]*College of Computer, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu 210003, China*

[2]*Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks, Nanjing, Jiangsu 210003, China*

[3]*Key Lab of Broadband Wireless Communication and Sensor Network Technology (Nanjing University of Posts and Telecommunications), Ministry of Education Jiangsu Province, Nanjing, Jiangsu 210003, China*

Received 1 July 2014, www.tsi.lv

Abstract

With rapid development of hardware, wireless sensor networks (WSN) have been applied in a wide range of fields. However, energy cost constrains putting WSN into use. To reduce energy cost, extending life time, WSN in low-duty-cycle (LDC) draws researchers' attention. In general, work time of a node only occupies 0.1%-10% in a cycle. This model certainly reduces the energy for idle listening. On the other hand, it makes the probability of congestion very high due to a node that can only receive packets when waking up. This paper proposes a new LDCWSN model to solve the congestion from duty schedule. With the model, we show a strategy for WSN fault averting, diagnosing and recovery based on congestion in nodes. We include some attributes of LDC WSN in our strategy, i.e. probability of congestion, scheduler, and link quality. By improving the selection of nodes on every level, we get a low rate for network's fault appearance, low E2E delay and long lifetime. The simulation's result shows that our strategy has a better performance in packet loss, energy cost and time delay than proposed WSN fault management.

Keywords: low-duty-cycle, congestion control, congestion recovery, fault management, wireless sensor network

1 Introduction

Wireless sensor network (WSN) is an integrated system consisting of embedded systems, new sensor material, low-power signal processing and wireless networks [1]. In most WSN, battery is the main power source, which makes the whole net have a limited life time. Usually, it is difficult to supply energy with constriction of scale, deploying environment and cost. Thus, many researches focus on energy efficiency in many applications. Researchers found that most cost of energy comes from idle listening [2]. This ennobles the importance of reducing unnecessary communication and sensing duty cycle. For the aim, low-duty-cycle WSN with short active time and long sleep time of nodes are a good choice [3].

Low-duty-cycle WSN use energy management protocol to schedule the active cycle and communicating time [4-6]. This ensures the node will remain dormant in the most time of one cycle and only 0.1%-10% active time [3]. The different schedules of each node cause sleep latency. Once getting a packet from low-level neighbours, a node will store it in a cache and deliver it later.

Therefore, while a node turns into active, neighbouring nodes in low levels will try to send messages to it at the same time, which aggravate the probability of congestion on the whole net.

During the convergence of messages to the sink node, how to find a sequence of nodes in different levels to build a delivery path in order to improve performance of the whole net (like packet delay, packet loss and life time) is a key issue. This paper based on the specialization of low-duty-cycle WSN, constructs a new research model. According to the model, we designed a strategy for congestion fault management in low-duty-cycle wireless sensor networks. We included the dynamic property, like congestion probability, schedule and link quality, into the strategy. By optimizing the choosing of nodes in different levels, the aim of improving the network is realized. The simulation shows that this strategy has an enhancement in packet loss, energy cost and packet delay.

This article is organized by this: Section 2 introduces the background and related work of low-duty-cycle WSN; Section 3 demonstrates the particularity of congestion in low-duty-cycle WSN; Section 4, we describe our model and strategy; in last section, we give details about our stimulation and future work.

2 Related Work

2.1 LOW-DUTY-CYCLE WIRELESS SENSOR NETWORK

There are only two status of one node in low-duty-cycle WSN: active or dormant [2]. A whole cycle (T) consists of one active time and one dormant time [7, 8]. The node will be active if and only if these two reasons are met:

* ***Corresponding author*** e-mail: yangj@njupt.edu.cn

1) The schedule wakes up the node to receive packets and do sensing jobs;
2) The node has packets to send to its neighbouring nodes.

While the node is dormant, all the hardware is not working except a clock. Nodes can send packets in any time but can only receive packets when it is active. Usually, to reduce redundancy of data, neighbouring nodes often have a different schedule [9, 10]. Because of the different active time of each node, if node A has a packet to node C, it have to wait until node B is active, which will cause a delay called sleep latency.

The congestion in wireless sensor networks can be classified into two types [11]:

One is node congestion. That is to say, the packets that need to be delivered exceed the node's capacity. The cache overflowing causes the packet loss and delay.

Another one is link congestion. Wireless transmission shares a channel. One channel can be used by a node at one time. While many neighbouring nodes compete for the channel, link congestion arises. This also leads to packet loss and delay, and lower throughput of whole net.

Paper [12] proposed a model of multi-channel. In this article, the wireless channel is divided into different paths by rate. The node will use the optimized channel to deliver the packet, avoiding the link congestion. In paper [2], nodes will trim the delivery time based on the link quality. The better the link quality is, the earlier the delivery time is. The two strategies avoid link congestion efficiently, however, in low-duty-cycle wireless sensor networks, the node congestion occurs more frequently due to the short interval of active time and large-amount packets to be received.

In paper [13], every node has multiple paths to choose. Once congestion occurs in one path, the node will try another path to deliver the packet. The frequent update of delivery paths will put a press on network overload, and the retransmit will cost more energy, and aggravate delay as well. In paper [14], sink node plays a role of controlling the nodes' delivery rate to avoid node congestion. In fact, most nodes will be dormant in low-duty-cycle WSN. In this situation the strategy performs poorly. In low-duty-cycle WSN, while taking the sleep latency into consideration, it is also important to lower the probability of congestion. Thus, a strategy that can detect the congestion and recovery from it is necessary.

2.2 CONGESTION IN LOW-DUTY-CYCLE

Congestion leads to the overload of the whole network, higher packet loss and delay. In low-duty-cycle, the results become worse. For example, once a packet is lost, the node has to resend it and consume more energy, which shortens life time. The active-dormant cycle succours the node congestion and effects the whole net. A node has a shot active time in a cycle. Neighbouring nodes will transmit packets at the same time, which cause the link congestion. In addition, the receiver node's parent may be still in dormant. That means receiver node has to store these packets in a cache and wait until its parent is active. Meanwhile, one delivery failure means that the node has to wait one duty cycle and send the same packet at the next active time. This consoles packet delay and enfeebles the network. Figures 2 and 3 shows congestion with the number of nodes and active time. Figures 4 and 5 show delay with the number of nodes and active time. From Figures 1 and 2, it is easy to conclude that congestion occurs more frequently with more nodes and less active time. Figures 3 and 4 indicates that delay becomes longer due to the congestion. Therefore, we need an algorithm, which can perform well with large-scale and long dormant time.

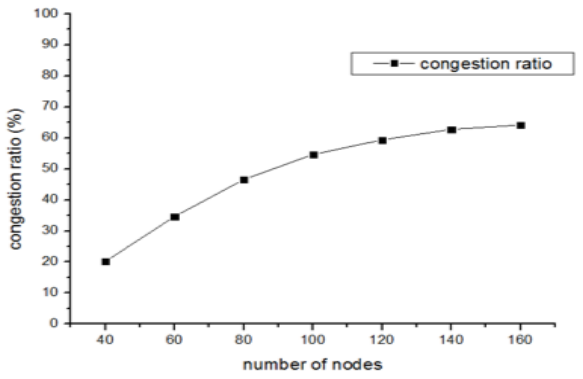


FIGURE 1 Number of nodes VS congestion ratio

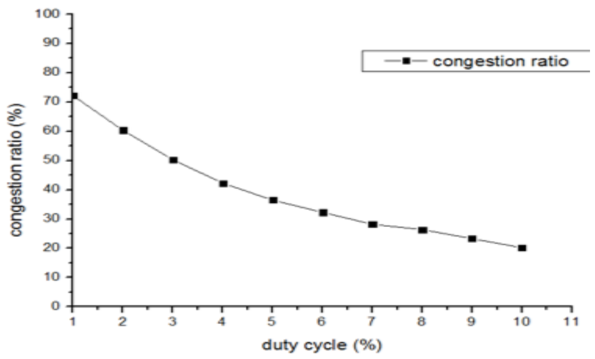


FIGURE 2 Duty cycle VS congestion ratio

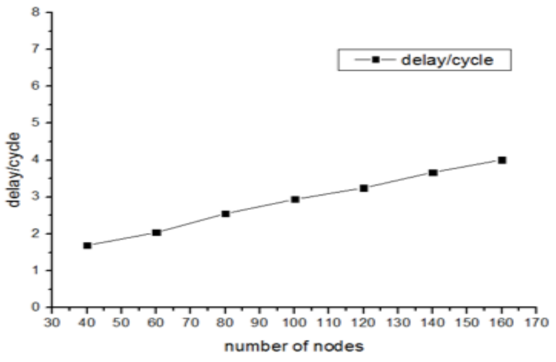


FIGURE 3 Number of nodes VS delay/cycle

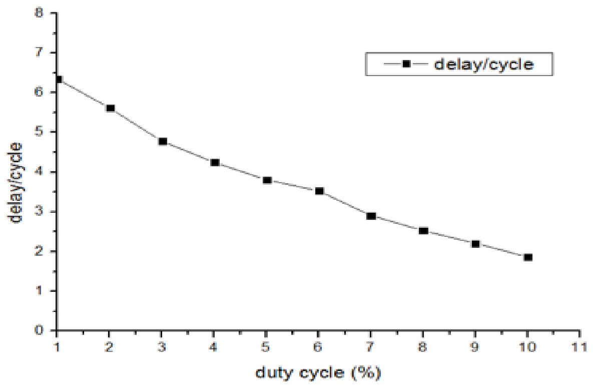


FIGURE 4 Duty cycle VS delay/cycle

3 A strategy for low-duty-cycle WSN

3.1 COMPETITION

Once the network is established, the sink node broadcasts a message. A node decides its level by the smallest hop of the message it received. If a node *i* receives three messages with 1 hop, 2 hops and 3hops, then node *i* belongs to level 1.

The node only communicates to the nodes in different levels. Paper [15] proposed FTSP which can make the whole network simultaneous with 2.24 μs error (usually, a duty cycle is at least 1ms). The node broadcast its schedule and neighbouring nodes to record just that. In every duty cycle, nodes in lowest level send a test packet to its parent according to the recorded schedule. If the parent received, then it will respond the ACK packet, and store the packet in a buffer, else, the send node marks this packet failure. If the parent received but the buffer is full, the node congestion occurs, and then it will respond the CN packet. While the lowest-level node has all finished, the second-low nodes repeat the process. One level by one level repeat this process until convergence to sink node. Assume this process repeats turns, because of the random link quality and congestion occurrence. The probability of link quality and congestion is described as follows:

Definition 1: represents that for a node a in level *K* and its neighbouring node *A* in level *K*–1, the link quality between them is calculated by $P_a^A(LQ) = m/n$, $m \le n$, where stands for the number of ACK packets received in turns.

Definition 2: represents that for a node *a* in level *K* and its neighbouring node *A* in level *K*–1, the congestion between them is calculated by $P_a^A(NCN) = 1 - l/m,\ (l \le m)$ where represents the number of CN packets during successful times.

Definition 3: represents a difference of the wake up time for a node an in level *K* and its neighbouring node *A*, which is calculated by:

$$\Delta T_a^A = \begin{cases} t_A - t_a, & t_A \ge t_a \\ t_A - t_a + T, & t_A < t_a \end{cases}, \tag{1}$$

where t_a is the wake up time of node *a* and *T* is the time of one cycle. We define that $\Delta t_a^A = 1 - \Delta T_a^A / T$. The value of Δt_a^A is meaningless so we just use it to reflect to the latency of two nodes.

3.2 STATIC CONGESTION AVOIDANCE STRATEGY (SCAS)

In the period while, the whole network is established but not in low-duty-cycle yet, according to the value of already known parameters, nodes can optimize convergence path to sink node to reduce the probability of congestion occurrence. We call this Static Congestion Avoidance Strategy (SCAS).

For every node in level *K*, after *n* turns, they can calculate their own competition C_a^A in the neighbouring node, which is calculated by $C_a^A = P_a^A(LQ) + P_a^A(NCN) + \alpha\Delta t_a^A \quad (\alpha > 1)$, where α is the weight of *D*-value of wake up time. In our algorithm, the competition reflects the priority of node *a* in the upper node *A*. The bigger the competition is, the more possible that node a should be in the receive sequence of the node *A*. Because there is sleep latency in low-duty-cycle WSN, the most significant value is sleep latency. So we give a weight for Δt_a^A to ensure that the D-value of wake up time will affect the competition most effectively.

Definition 4: $E_a^A(PS)$ represents the exception packet size of node a in its neighbouring node *A*. We can calculate it by $E_a^A(PS) = P_a^A(LQ) \times PS_a$, where PS_a is the packet size.

When nodes in level *K* have calculated their competition, they sort the upper nodes by competition to construct a competitive sequence.

Definition 5: sequence $C_a = \{C_a^1, C_a^2, ..., C_a^n\}$, $(C_a^1 \ge C_a^2 \ge ... \ge C_a^n)$ represents node *a* in a lower level, sorting all its upper neighbouring nodes by competition.

Nodes in level *K* then compete with each other for getting parents in the upper level. First, nodes in level *K* sends a packet, which includes application for joining the receive sequence and competition, to the first node in its competitive sequence. The node in level *K*–1 received these packets, and then sorts their competition. The node in level *K*–1 repeats the process:

1) put the node with biggest competition to the receive sequence and delete its packet;

2) calculate residual size of buffer $B_A = B_A - E_a^A(PS)$, where B_A is the current buffer size;

3) if B_A is larger than the threshold ϕ, then back to step 1.

When the node *A* in level *K*–1 has finished it's receive sequence, it will broadcast the nodes which are in the sequence to all of its neighbours. The nodes in the sequence will update their competition value. For example, if node *a* is selected by node *A*, the next node, which constructs receive sequence is node *B* and is neighbouring node of *a* as well. Then, the link quality between node *a* and node *B* will be $P_a^B(LQ)=(1-P_a^A(LQ))P_a^B(LQ)$ and congestion will be $P_a^B(NCN)=(1-P_a^A(NCN))P_a^B(NCN)$, the competition also updates. By doing the update, for event node a send a packet to node *B* has become a conditional probability, which can be represented as *P* (send packet to *B* | fail to send packet to *A*). Thus, the competition goes down with probability.

In the broadcast message from node *A*, there is a message if the buffer size is full. If not, other nodes still have to compete to join *A*'s receive sequence.
Pseudo code is described as follows:

```
calculate the competitiveness for every neighbouring node in level K–1
sort nodes by the competitiveness $C_a=\{C_a^1,C_a^2,...,C_a^n\}$, $(C_a^1\ge C_a^2\ge ...\ge C_a^n)$
for $(i=1;i\le n;i++)$
        if isfull (node i) is false
          send competitive packet to node i
          if (node i) admit
          for $(j=i;j<n;j++)$
$C_a^j=(1-P_a^i(LQ))\times P_a^j(LQ)+(1-P_a^i(NCN))\times P_a^j(NCN)+\alpha\Delta t_a^j$
          else continue

sort nodes by the competitiveness from level K $S^A=\{C_1^A,C_2^A,...,C_e^A\}$, $(C_1^A\ge C_2^A\ge ...\ge C_e^A)$
for $(i=1;i\le e;i++)$
        $B_A=B_A-E_i^A(PS)=B_A-P_i^A(LQ)\times PS_i$
        if $(B_A\le\phi)$ break;
        else put node I to receive sequence;
```

Additionally, according to the receive sequence and the order of lower nodes in the sequence, every lower-level node calculate its send time TX_t by $TX_t=\frac{t_A}{e}\times e_a+\tau_A$, where *e* is the number of nodes in the receive sequence, e_a is the order of node a in the sequence and τ_A is the wake up time of node *A*.

Level by level, every node constructs it is receive sequence for its neighbouring nodes in lower levels. If the nodes exist that are rejected by all of the upper nodes, the node chooses one of the nodes with the largest competition and sends the joining message. The upper ones make it to its receive sequence.

3.3 DYNAMIC CONGESTION AVOIDANCE STRATEGY

When all receive sequences are finished, the sink node broadcasts and the whole network goes into the low-duty-cycle model.

Due to the SCAS, we use exception packet size to judge if the buffer is full. If in one cycle, most packets are sent successfully, the real value must be bigger than the exception. Thus, it is still possible that congestion will occur. Adjustments based on the real situation, we call this Dynamic Congestion Avoidance Strategy (DCAS).

While one node's buffer overflows and a new packet arrives, it will respond the CN packet. The sender node then tries to deliver the packet to the next node in its send sequence. If one node in the lower level gets x CN packets in a continuous cycle (x is a threshold). If the cause is that packet size is too big for upper node's buffer size, the node should surrender the packet and send to sink node. If the buffer size is truly full, the node has to change its send sequence.

To change the send sequence, the node in the lower level sends is full packet. If the response is true, which means the buffer cannot get any packet; the lower level node deletes the upper node from its send sequence. If the response is false, the node will split its packet into small pieces based on the response and send these pieces into different upper nodes. Finally, the sink node will integrate them.

4 Simulation and future work

4.1 SIMULATION

To know the number of retransmissions, packet loss and packet delay in the different algorithms, different duty cycle and different scale; we use NS-3-3.15 to test. The simulation steps are described as follows:

Put *n* (*n* from 50 to 600 with the step of 50) nodes in 100*100 square randomly, every node has a level number based on the hop to sink node.

Initialize the character of node. Let the duty cycle be 100*s* and the node will be active in 1*s* during 0-99*s* randomly. Every node has a communication radius and some buffer size. Each node will generate a packet. According to the position of every node, they have their parents with a link quality, which is a random number from 0.4 to 1, a congestion probability which is a random number from 0.1 to 0.6 and a schedule.

Sending packets. We use three different algorithms to construct a different send sequence. Then nodes will deliver packets according to the send sequence. Change the duty cycle and repeat the simulation.

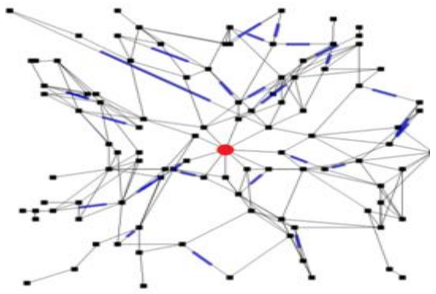
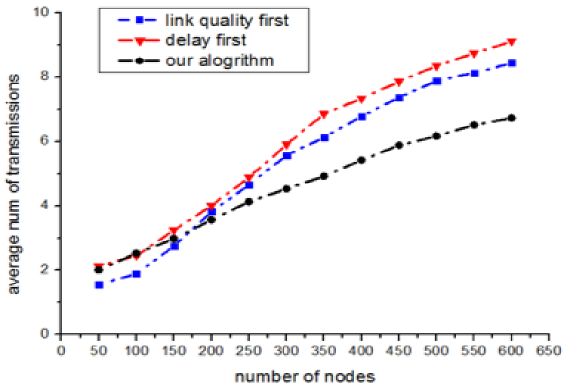
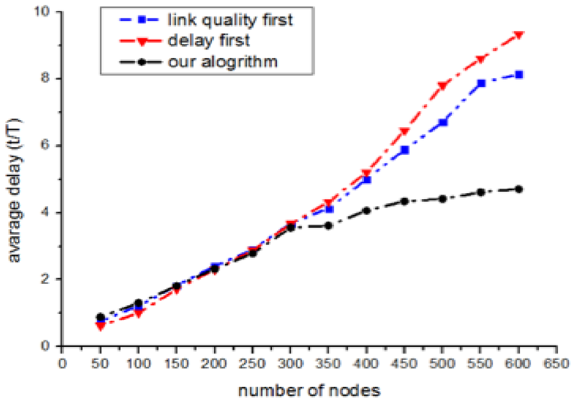


FIGURE 5 Topology of the simulation.

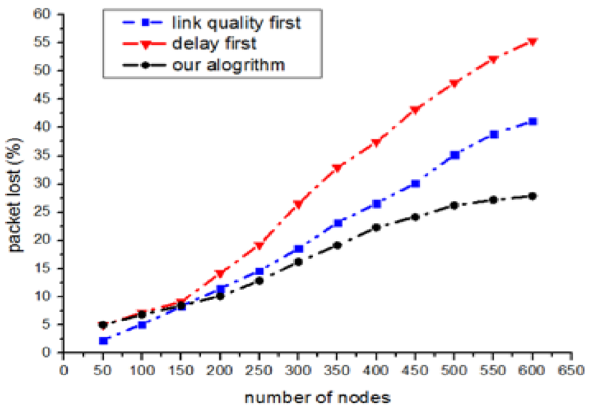
For different network scale and duty-cycle, we compare our algorithm to link quality first and delay first in number of transmissions, send delay and packet loss. The result shows in Figures 6 and 7.



a) Network size VS average num of transmissions

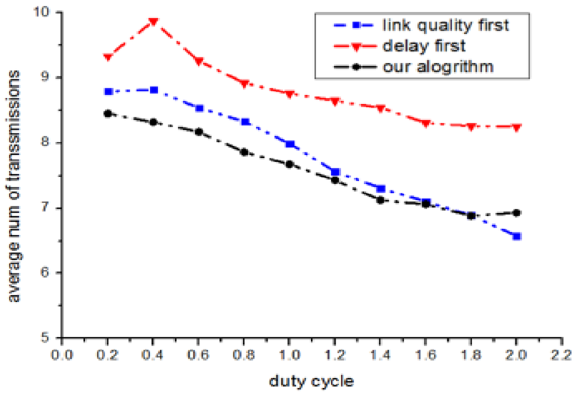


b) Network size VS average delay

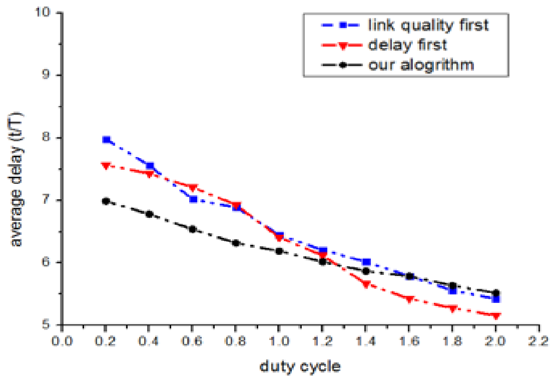


c) Network size VS packet lost

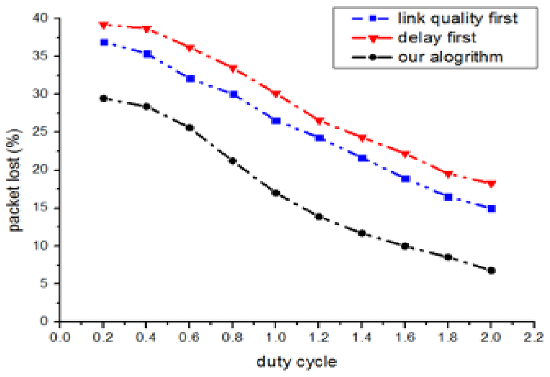
FIGURE 6 Performance comparison under different network scale



a) Duty cycle VS average num of transmissions



b) Duty cycle VS average dela



c) Duty cycle VS packet lost

FIGURE 7 Performance comparison under different working duration n

5 Conclusions

This paper proposed the low-duty-cycle congestion strategy. Although the performance is nice, the constriction is ideal, especially the schedule synchronization. In future work, we will discuss how to synchronize the whole network and when the node should adjust its wake up time.

Acknowledgements

Our work supported by A Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions (yx002001); Graduate student research and innovation program Jiangsu (CXLX12_0480).

References

[1] Peng Y, Song J, Peng X 2009 Survey of fault management framework in wireless sensor networks *Journal of Electronic Measurement and Instrument* **23**(11) 1-10 *(in Chinese)*

[2] Guo S, Gu Y, Jiang B, He T 2009 Opportunistic Flooding in Low-Duty-Cycle Wireless Sensor Networks with Unreliable Links *Proceedings of MobiCom'09* September 2009 133-44

[3] Pak W, Bahk S 2012 Centralized route recovery based on multi-hop wakeup time estimation for wireless sensor networks with ultra-low duty cycles Computer Communications **35**(11) 1355-67

[4] Jeong G, Gu Y, He T, Du D 2009 VISA: Virtual Scanning Algorithm for Dynamic Protection of Road Networks *Proceedings of IEEE INFOCOM* 927-935

[5] Wang X, Xing G, Zhang Y, Lu C, Pless R, Gill C 2003 Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks *Proceedings of the First International Conference Embedded Networked Sensor Systems* 28-39

[6] Kasbekar G S, Bejerano Y, Sarkar S 2011 Lifetime and Coverage Guarantees through Distributed Coordinate-Free Sensor Activation *IEEE/ACM Transactions on Networking* **19**(2) 470-83

[7] He T, Krishnamurthy S, Luo L, Yan T, Gu L, Stoleru R, Zhou G, Cao Q, Vicaire P, Stankovic J A, Abdelzaher T F, Hui J, Krogh B 2006 VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance *ACM Transactions on Sensor Networks* **2**(1) 1-38

[8] Tolle G, Polastre J, Szewczyk R, Turner N, Tu K, Burgess S, Gay D, Buonadonna P, Hong W, Dawson T, Culler D 2005 A Macroscope in the Redwoods *Proceedings of SenSys'05* 51-63

[9] Gupta H, Navda V, Das S, Chowdhary V 2008 Efficient Gathering of Correlated Data in Sensor Networks *ACM Transactions on Sensor Networks* **4**(1) 402-13

[10] Wang J, Liu Y, Das S 2008 Asynchronous Sampling Benefits Wireless Sensor Networks *Proceedings of the 27th Conference on Computer Communications IEEE* 2207-15

[11] Li L, Li B, Zhou X 2008 A Survey of Congestion Control Technology for Wireless Sensor Networks *Journal of Computer Research and Development* **45**(1) 63-72 *(in Chinese)*

[12] Rathi M K 2010 Data Dissemination in Low Duty-cycle Multi-channel Multi-hop-Wireless-Sensor-Networks Proceedings of the Computer Engineering and Technology (ICCET) the 2nd International Conference April 2010 258-62

[13] Liu Y, Ma Z, Cao Z 2005 A mitigating stagnation based ant colony optization routing algorithm Communications and Information Technology *Proceedings of IEEE International Symposium on Communications and Information Technology* 36-9

[14] Akan Ö B, Akyildiz I F 2005 Event-to-Sink Reliable Transport in Wireless Sensor Networks *IEEE/ACM Transactions on Networking* **13**(5) 1003-16

[15] Maroti G S M, Kusy B, LedecziA 2004 The Flooding Time Synchronization Protocol *Proceedings of SenSys'04*

Author

Jian Yang, born in March, 1978, Jiangsu, China

Current position: Associate Professor, PhD candidate in the Nanjing University of Posts and Telecommunications, Department of Computer.
University studies: Nanjing University of Posts and Telecommunications.
Scientific interest: Computer networks, health care services, energy efficiency, reliability, resource allocation, wireless sensor networks
Publications: 4