CDAP: Congestion Detection and Avoidance Protocol for Dynamic Multilevel Priority Packet Scheduling in Sensor Networks

C.Vijayakumaran, K. Janaky

Abstract—The packet collision is one of the major sources of energy consumption in WSN. Due to packet collision, deadlock will occur in packet processing and scheduling schemes. The existing WSN uses various packets scheduling scheme and if a real time task holds the resources for a longer period of time, other tasks need to wait for longer period time, causing the occurrence of a deadlock. The occurrence of deadlock situation degrades the performance of task and packet scheduling schemes in terms of end to-end delay. In this paper, energy efficient Congestion Detection and Avoidance Protocol (CDAP) is proposed for dynamic multilevel priority packet scheduling (DMP) scheme which includes of three mechanisms. 1) First Use buffer and weighted buffer difference for congestion detection; 2) Propose a bottleneck-node-based source data sending rate control scheme; and 3)Use Flexible Queue Scheduler for packets transfer. The result of simulation of the proposed work shows that CDAP achieves efficient congestion control and flexible weighted fairness for dynamic multilevel priority packet scheduling. Therefore it significantly enhances packet delivery ratio and reduce end-to-end delay without increasing total energy consumption. It leads to higher energy efficiency and better QoS in terms of throughput, fairness compared to the tested baseline approach.

Index Terms—CDAP, DMP, congestion detection, packet scheduling, congestion avoidance, bottleneck node, WSN.

1 INTRODUCTION

In recent years, Wireless Sensor Networks (WSNs) have achieved a widespread applicability in many areas such as environmental monitoring, health care, agriculture, security and battlefield. WSN consists of a large number of tiny sensor nodes that can be deployed randomly, configured and controlled automatically without any human intervention.

WSN nodes are battery operated in nature. The major sources of energy consumption in WSN are prolonged wakeup schedule of a node, delay in packet arrival, frequent network partitioning due to node mobility and congestion in the network. The network congestion, which is quite common in wireless networks, occurs when offered load exceeds available capacity or the link bandwidth. Network congestion causes channel quality degradation and raises packet dropping rates in the channel. It leads to increase in packets waiting time at the buffers, increased delays, wasted energy, and hence retransmission of missing packets is required. Congestion is the main source of increased buffer occupancy which results in unfair handling of traffic flows which traverse either through congested area or through a significant number of hops thus reducing the performance and lifetime of the network [1].

In this paper, a deadlock free dynamic multilevel priority packet scheduling scheme is proposed for WSN. The existing similar approach has some congestion when scheduling the packets [2]. The proposed CDAP method is used to detect congestion in the packet scheduling scheme and avoid congestion for dynamic multilevel priority (DMP) packet scheduling, which has a distributed mechanism operating at network and MAC layer. The contributions of this work towards CDAP are:

- CDAP enables cross-layer optimization. Specifically, a technique to measure congestion is proposed, which measures buffer and weighted buffer difference. To differentiate traffic priority, static priority and dynamic priority are introduced.
- Transient congestion and persistent congestion are differentiated and are dealt separately. For transient congestion, hop-by-hop implicit backpressure method is used. For persistent congestion, bottleneck node based data sending rate control and multi-path loading balancing are proposed. Unlike [3], this method does not need explicit ACK from sink. Using the method, bottleneck nodes can be identified and source data sending rate can be dynamically adjusted.
- A flexible queue scheduler can dynamically select the next packet to send according to packet priority.

- *C.Vijayakumaran is currently pursuing Ph.D. program in AISECT University, Bhopal, MP, PH-9444328196. E-mail: c_vijayakumaran@yahoo.com*
- *K. Janaky is currently pursuing masters degree program in computer science and engineering in Valliammai Engineering College in Anna University, Chennai, India, PH-9597482161. E-mail: janaky.sarav16@gmail.com*

2 RELATED WORK

In WSN, many network design issues, such as routing protocols, and packet scheduling that reduce sensor's energy consumptions and data transmission delay, packet scheduling is very important since it determines the transmission order of a number of data packets based on different criteria such as transmission deadline and data priority. For instance, real-time data should have the higher priority to be transmitted to the base station (BS) than non-real-time data.

Though extensive research for scheduling the sleep-wake times of sensor nodes has been conducted[5]–[6], only a few studies exist in the literature on the packet scheduling of sensor nodes [9]–[11] that schedule the processing of data packets available at a sensor node and also reduces energy wastage. The most existing Wireless Sensor Network (WSN) operating systems use First Come First Serve (FCFS)[12] schedulers that process data packets in the order of their arrival time and it takes a lot of time to be delivered to a relevant base station (BS) which introduces congestion in the network.

Currently, there are extensive studies to address congestion problems in WSNs [13-15]. The Congestion Detection and Avoidance (CODA) [16] scheme is jointly sampling the channel loading during every epoch and monitoring buffer length of being filled to judge if congestion happens or not. For transient congestion, the node sends explicit backpressure messages to its neighbors that further propagate the message to upstream source nodes depending their local buffer occupancy or channel loading. For persist congestion, CODA needs explicit ACK from sink. If insufficient ACK reaches the source, the source will lower its sending rate. However the explicit ACK wastes much energy and the loss of ACK due to link quality will give a false congestion signal to the source and affect the network throughput. CODA can't differentiate bottleneck link either. In this paper, the CDAP is used to detect and avoid congestion and propose a bottleneck-node-based source data sending rate control scheme for DMP.

3 PROPOSED DMP CONGESTION AVOIDANCE SCHEME

The proposed DMP scheduling scheme assumes that the nodes are at the same hop distance from the base station (BS) are considered to be located at the same level. Timeslots are allocated to nodes in WSN at different levels using TDMA scheme. For instance, nodes that are located at the lowest level and the second lowest level can be allocated timeslots 1 and 2, respectively. Three-level of queues are considered, that is, the maximum number of levels in the ready queue of a node is three: priority 1 (*pr*1), priority 2 and priority 3 queues. Real-time data packets are assigned to the highest priority queue, and are processed using FCFS. Non-real-time data packets that arrive from sensor nodes at lower levels are given to the second level highest priority queue. Finally, non-real time data packets are put to the lowest priority queue.

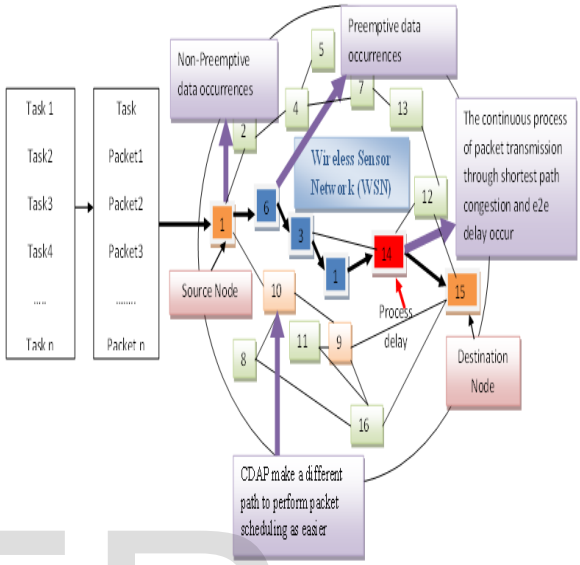

Fig 1. Architecture of DMP congestion avoidance scheme

In wsn, congestion occurs due to the continuous process of packet transmission through shortest path. The proposed DMP detects congestion by using CDAP protocol and make a different path to perform packet scheduling as easier in Fig 1.

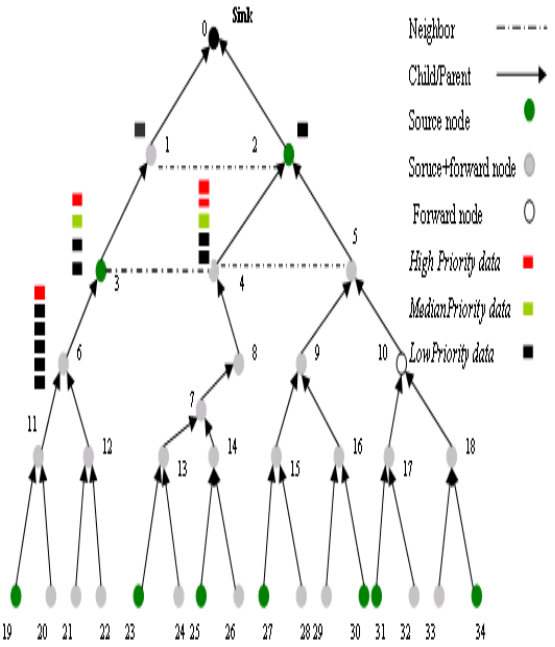

Fig 2. Network topology for discussion

3.1 Use Buffer and Weighted Buffer Difference for Congestion Detection.

Queue management is often used in traditional networks for congestion detection. It is also used in many protocols for sensor networks, such as [7-8]. But in some occasions, buffer alone is insufficient to detect congestion. According to buffer based congestion detection scheme, node3, node4, node6, and node8 experience congestion, as shown in Figure 1. These nodes send backpressure message to their neighbors in explicit or implicit manner. After node6 receives node3's suppression message and node4 receivesnode8's suppression message, node6 and node4 lower down their data sending rate to let node8 and node3 empty their queues. But this will exacerbate congestion because the buffer queues of node4 and node6 quickly become full. Furthermore, node4 and node6 have many high priority packets in their queues. These packets are overwritten by subsequent low priority packets, which will degrade systems performance drastically.

The solution to the problem is to adopt buffer and weighted buffer difference to detect congestion. According to Fig. 2, if we assume that the weight of high, medium, low priority data to be P_h=3, P_m=2, P_l=1. The weighted buffer length of node6 isWBL6 = 2*P_h+1*P_m+3*P_l= 2*3+1*2+3*1 = 11. If we denote the weighted buffer difference as WBD, then WBD (node6, node3) = 6, WBD (node4, node3) = 5,WBD (node4, node8) = 7, WBD (node4, node2) = 9, WBD (node8, node4) = -7, WBD (node3, node4) = -5. A node with higher WBD has more important data to send. If the buffer lengths of the four sensor nodes exceed the utilization threshold (that is measured from buffers length) simultaneously, congestion is resolved in the order of node4→node6→node3→node8.

3.2 Bottleneck-node-Based Source Data sending Rate Control and Multipath Loading Balancing.

Although the backpressure message of distributed hop by hop congestion control can finally reach the source node and source data sending rate is adjusted to mitigate congestion which happens in the downstream (sink side), it cannot accurately adjust source data sending rate. A method is proposed, in which every node can determine routing path (from itself to sink) status. The forwarder can find better path to forward data and the source data sending rate can be adjusted more accurately and efficiently.

Every packet has two kinds of priorities: static priority and dynamic priority. Static priority is represented as an integer and the lowest static priority is SP (packet) =0.

The Packet dynamic priority is defined as:

$$DP(packet) = \frac{\alpha * hop + SP(packet)}{1+\beta * delay} \quad \text{--Eq(1)}$$

Where α and β are two parameters for tuning system performance; *hop* is the number of hops to sink; *SP* is the packet static priority; *delay* is the time from the packet generation to current node. When congestion happens, some packets are dropped or retransmitted. Dynamic priority scheme is used to determine which packets are dropped or retransmitted.

4 PROTOCOL DESIGN DETAILS

4.1 CONGESTION DETECTION

In order to precisely measure local congestion level at each node, buffer and weighted buffer difference is used for congestion detection. Every node which has data to send monitors its buffer and piggybacks its *WR* and *WQ* in its outgoing packets. If a node's buffer occupancy exceeds a threshold and its data has higher priority, the congestion bit in the outgoing packet header is set.

The pseudo code of congestion detection algorithm is as follows:

```
If (WQDnodei(t)>=0) &&
   (buffer length>=Qmax)CN=2;
else if ((WQDnodei(t)>=0) &&
   (bufferlength>=Qmin)||
   buffer length>=Qmax)CN=1;
else
   CN=0;
```

Two thresholds Q_{min} and Q_{max} are used to differentiate congestion level. In CDAP, Q_{min}= (1.0/3.0)*20 and Q_{max}=(2.0/3.0)*20, where 20 is the maximum queue length.

4.2 Bottleneck-node Based Source Rate Control

A Source S maintains data sending interval g(i), i=1, 2,3... It periodically updates its data generation interval according to its neighbor's congestion level. Owing to many to one communications, when source nearby nodes send data smoothly, congestion could still happen in the downstream (sink side). To solve the problem, a method is proposed which is called as bottleneck node based source data sending rate control. It includes: 1) Determine routing path

status from a certain node to sink; 2) Bottleneck node detection and source data sending rate control. Using this scheme, source data sending rate can be regulated more accurately.

4.2.1 Determination of Routing Path Status from a Certain Node to Sink

For a node whose next hop is the sink, the data forwarding delay is piggybacked in the data packets. Its child node overhears this information and compares its own data forwarding $delay_i$ with its parent's data forwarding $delay_{parent}$, and does the following calculation:

$$delay_{max}=MAX\{delay_{parents}, delay_i\} \quad \text{– Eq (2)}$$

Next time, when it has data to send, delaymax will be piggybacked in the packet header, delaymax is the path status from current node to sink. This process is recursively computed up to the final source node.

4.2.2 Bottleneck node Detection and Source Reporting Rate Control

When source node overhears data from its parent, it extracts the delay information piggybacked in the data packets and set its data sending rate as:

$$data\_sending\_interval=delay_{max}$$

A simple example is used to demonstrate the process. As shown in Fig. 2, if we assume the average data forwarding delay of node 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 as 1, 1, 1.5,2, 1, 1.6, 1.5, 1.7, 1.4, 1.2, 1.0, 1.3, respectively,then$delay_{max}(11)=2$,$delay_{max}(8)=2$,$delay_{max}(9)=1.4$,$delay_{max}(10)=1.6$, $delay_{max}(12)=2$.

Node 0 has three paths to the sink; these paths are denoted as path1, path2, path3, as shown in Fig. 3. The source (node0) has three data sending intervals namely v1, v2, and v3.

For path1: $v1=delay_{max}(10)=1.6$;
For path2: $v2=delay_{max}(11)=2$;
For path3: $v3=delay_{max}(12)=2$.

If an intermediate forwarding node has more than one paths to sink, load balancing is performed. Taking node12 as an example, it has two paths.

For path 3,1: $delay_{max}(8)=2$
For path 3,2: $delay_{max}(9)=1.4$

Path 3,2 is better than path3,1 and more data will be forwarded through path 3,2. Using bottleneck node based source data, easily detect the congestion node and avoid the node by using sending rate control intervals.

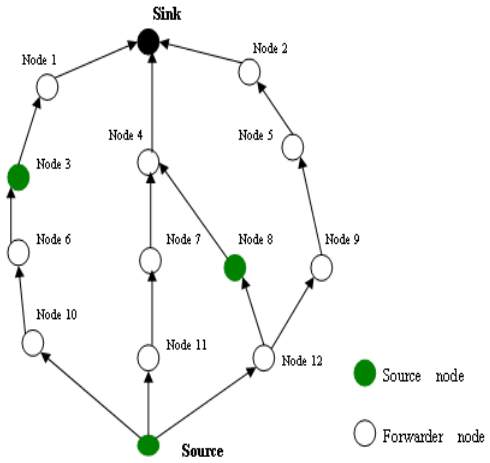

Fig 3. Bottleneck node based source data sending rate control

4.2.3 Rate Adaptation

When congestion occurs, packets are dropped to alleviate congestion. Most of the queue schedulers, both in wired or wireless networks, drop packets from tail rather than any position in the queue. We introduce a scheduler with two sub-queues, as shown in Fig. 4. When a packet is successfully transmitted, the scheduler gets the next packet from queue. We denote scheduling interval as R(t), the MAC forwarding interval as Rout(t), the minimum scheduling interval Min(R(t))=Rout(t). The two sub-queues are separately used for locally generated traffic and route-through traffic. For every source, packets are sorted by their dynamic priority from high to low. When a packet is sent, a round robin algorithm is executed. To ensure fairness, our algorithm scans the route-through traffic queue from head to tail and one packet from each source is sent, then a locally generated packet is sent.

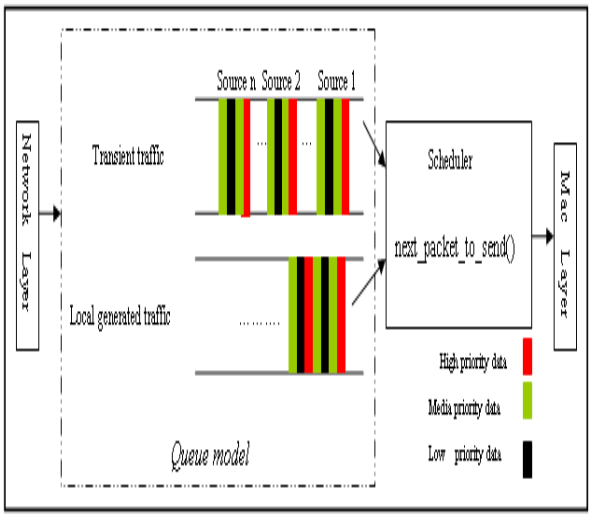

Fig 4. Queue model

Like existing protocols, upon receiving a backpressure message, the source node or forward node decreases its data sending rate or adjusts data sending rate for different paths if multiple paths exist. However, unlike existing protocols, if no backpressure message is received, the source node doesn't increase its data sending rate additively. In CDAP, the data sending interval can be exactly set as delaymax upon receiving a message from downstream. For a node which is a forwarder as well as a source, its data forwarding rate and data sending rate can be adjusted separately.

5 EVALUATION AND COMPARISON

In this section the performance of CDAP is evaluated and compared with other existing solutions like CODA. CDAP has three components, buffer and weighted buffer difference (WQD) for congestion detection, bottleneck node based source data sending rate control, and Flexible Queue Scheduler for sending next packet. Since CODA is widely accepted congestion control scheme, hence, CDAP is compared with CODA. The simulations were done in NS2 with the parameters given in Table 1. There are 35 nodes and three dynamic priorities are adopted in simulations, which are 0, 1, and 2, respectively. The parameters used in α=0.5, β=0.02, Q_{min}=(1.0/3.0)*20,Q_{max}=(2.0/3.0)*20. The maximum dynamic priority *MP* is set to 3. Three metrics, throughput, end-to-end packet delay, and weighted fairness, are selected to evaluate system performance.

TABLE 1
NS-2 SIMULATION PARAMETERS

Area of sensor field	100*1000 m2
Number of sensor nodes	35
Radio range of a sensor node	70 m
Packet length	36 bytes
IFQ length	20 packets
Transmit Power	0.660 W
Receive Power	0.395

5.1 Throughput and End-to-End Delay

Throughput and delay are simulated and shown in Figure4. From figure, one can see that CDAP has higher throughput than CODA. Since CODA adopts queue-length-based scheme for congestion detection, every time when buffer length exceeds a threshold, congestion is reported. When persistent congestion happens, CODA needs feedback from sink to maintain its sending rate, which has two drawbacks: 1) Too many ACKs cause extra energy consumption; 2) ACKs may be dropped due to some reasons (congestion, link variation) and cannot reach the source. CDAP solves this problem by introducing a bottleneck node based source reporting controls scheme which is in implicit manner. It costs no extra energy and is more robust than CODA's closed loop multi-source regulation scheme.

The average delays of CODA and CDAP are presented in Fig. 5. Because CODA resolves persistent congestion in efficiently and cannot deal with the situation that several buffers exceed threshold simultaneously, packets experience much longer delay in CODA than in CDAP.

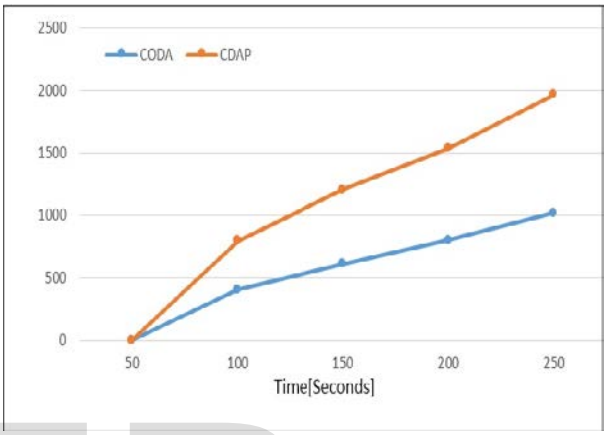

Fig 5. Throughput comparison

5.2 Weighted Fairness

The most important improvement of CDAP is that it provides fairness to DMP. The packet throughput and delay for different packet priorities are simulated and shown in Fig. 6. Since CDAP adopts weighted buffer difference for congestion detection, only the node which has higher priority data can send congestion information and packets are scheduled according to their priority. As indicated in Fig. 6, the packet with high priority has higher throughput and lower delay.

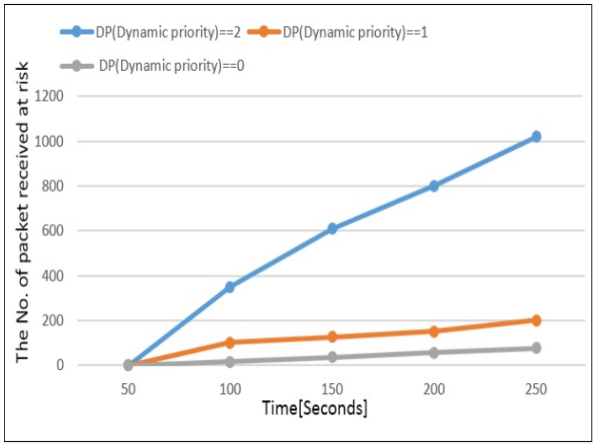

Fig 6. Throughput for DMP congestion avoidance scheme

CDAP provides weighted fairness for different class of traffic. It also provides fairness for packets with same static priority but its source is at different depth in the routing tree of a network. The dynamic priorities of the packets are partially determined by the number of hops to sink and the delay. With the number of hops increasing, packet dynamic priority increases and coverage fidelity is guaranteed.

6 CONCLUSION

In this paper, a Congestion Detection and Avoidance Protocol (CDAP) has been proposed. It detects congestion jointly using buffer and weighted buffer difference. CDAP deals with transient congestion and persistent congestion efficiently. For transient congestion, it adopts hop-by-hop congestion control scheme. For persistent congestion, it adopts bottleneck node based source data sending rate control. CDAP has a flexible queue scheduler and packets are scheduled according to their priority. It has been demonstrated through simulation that CDAP achieves efficient congestion control and flexible weighted fairness for dynamic multilevel priority packet scheduling. Therefore it significantly enhances packet delivery ratio and reduce end-to-end delay without increasing total energy consumption. It leads to higher energy efficiency and better QoS in terms of throughput and fairness.

REFERENCES

[1] C. Larsen, M.Zawodniok, " Route Aware Predictive Congestion Control Protocol for Wireless Sensor Networks "IEEE transactions on wireless communications, vol. 6, no. 11, november 2007

[2] L. Karim, N. Nasser, T. Taleb, and A. Alqallaf,"An Efficient Priority Packet Scheduling Algorithm for Wireless Sensor Network" IEEE ICC 2012 - Ad-hoc and Sensor Networking Symposium.

[3] B.Hull, K. Jamieson, and H. Balakrishnan, "Mitigating congestion in wireless sensor networks", in Proc, ACM Sensys, Nov. 2004.

[4] C.-Y. Wan, S.B. Eisenman, and A. T. Campbell, "CODA: Congestion detection and avoidance in sensor networks", in Proc. ACM SenSys, Nov.2003.

[5] E. Bulut and I. Korpeoglu, "DSSP: a dynamic sleep scheduling protocol for prolonging the lifetime of wireless sensor networks," in *Proc. 2007International Conf. Advanced Inf. Networking Appl.*, vol. 2, pp. 725–730.

[6] P. Guo, T. Jiang, Q. Zhang, and K. Zhang, "Sleep scheduling for critical event monitoring in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 2, pp. 345–352, Feb. 2012.

[7] C.-T. Ee and R. Bajcsy, "Congestion control and fairness for many-toonerouting in sensor networks", in Proc. ACM Sensys, Nov. 2004.

[8] B.Hull, K. Jamieson, and H. Balakrishnan, "Mitigating congestion in wireless sensor networks", in Proc, ACM Sensys, Nov. 2004.

[9] N. Edalat, W. Xiao, C. Tham, E. Keikha, and L. Ong, "A price-based adaptive task allocation for wireless sensor network," in *Proc. 2009IEEE International Conf. Mobile Adhoc Sensor Syst.*, pp. 888–893.

[10] H. Momeni, M. Sharifi, and S. Sedighian, "A new approach to task allocation in wireless sensor actor networks," in *Proc. 2009 International.*

[11] X. Yu, X. Xiaosong, and W. Wenyong, "Priority-based low-power task scheduling for wireless sensor network," in *Proc. 2009 International Symp. Autonomous Decentralized Syst.*, pp. 1–5.

[12] W. Stallings, *Operating Systems*, 2nd edition. Prentice Hall, 1995.

[13] C.-Y. Wan, S.B. Eisenman, A. T. Campbell, "CODA: Congestiondetection and avoidance in sensor networks", in Proc. *ACM SenSys*,Nov.2003.

[14] C.-T. Ee, R. Bajcsy, "Congestion control and fairness for many-to-one routing in sensor networks", in Proc. *ACM Sensys*, Nov. 2004.

[15] B. Hull, K. Jamieson, H. Balakrishnan, "Mitigating congestion in wireless sensor networks", in Proc, *ACM Sensys*, Nov. 2004.

[16] C.-Y. Wan, S.B. Eisenman, A. T. Campbell, "CODA: Congestiondetection and avoidance in sensor networks", in Proc. *ACM SenSys*,Nov.2003.