

Multi-Channel Polling in Multi-Hop Clusters of Hybrid Sensor Networks

Ming Ma, Zhenghao Zhang and Yuanyuan Yang

Department of Electrical & Computer Engineering
State University of New York, Stony Brook, NY 11794, USA

Abstract—In this paper we propose a multi-channel polling algorithm in multi-hop clusters of hybrid sensor networks. The hybrid sensor network consists of two types of nodes: basic sensor nodes and cluster head nodes. Basic sensor nodes have limited communication capacity and mainly focus on sensing the environment, while cluster head nodes are equipped with more powerful transceivers but simpler sensing modules. The cluster head node organizes basic sensor nodes around it into a cluster, and collects sensing data from sensors and forwards data to the outsider observer. This type of network has better energy-efficiency since traffic within cluster is scheduled by cluster head and packet collisions can be avoided. In addition, idle listening time can be shortened by turning off the transceivers of sensor nodes after data has been forwarded to cluster head. We will focus on finding energy efficient and collision-free polling schedules in the multi-hop cluster with multiple frequency channels. Due to its energy efficiency and scalability, the proposed algorithm will be very suitable for applications such as large scale environment monitoring. We also implement the proposed algorithm on NS-2. Simulation results show that multi-channel polling algorithm shortens the active time of sensor nodes by a significant amount compared to single channel polling. In the case that the total frequency bandwidth allocated to the cluster is fixed, the optimal number of channels can be obtained.

I. INTRODUCTION AND BACKGROUND

In recent years, wireless sensor networks (WSN) are playing an increasingly important role in a wide-range of applications, such as medical treatment, outer-space exploration, battlefield surveillance, emergency response, etc. [1], [2]. A wireless sensor network is generally composed of hundreds or thousands of sensor nodes, with each sensor capable of sensing the environment and also sending data to outside observers via wireless channels. Each sensor consists of five basic components: a Sensing module, a Processor/Memory module, a Location Finding module, a Transceiver module and a Power module. Although special attention has been paid to low power consumption when designing these modules, a sensor node can survive only very limited lifetime with current technologies [3], [4]. Compared to other components, transceiver unit is the most energy-consuming part of the entire sensor, which may have four states: sleeping, idle listening, receiving and sending. The power consumption ratio for above four states is about 4.2:7.3:7.5:8.2 [11]. Note that the power consumption of idle listening is close to that of sending/receiving, since sensor nodes may have to decode packets not destined to them.

The key challenges in establishing multi-hop sensor networks from a bunch of small, low-powered sensor nodes are energy-efficient mechanisms and scalability. [16] addressed these problems by introducing a hybrid sensor network which has two types of nodes: basic sensor nodes and cluster-head nodes. Basic sensor nodes have limited communication capacity and mainly focus on sensing the environment, while cluster head nodes are equipped with more powerful transceivers but simpler

sensing modules. The cluster head node organizes basic sensor nodes around it into a cluster, collects sensing data from sensors and forwards data to the outside observer. It was shown that by letting sensors enter the sleeping mode as much as possible, the two-layer hybrid sensor network achieves much longer life time than the homogeneous sensor networks.

However, in [16], it was assumed that there is only one data channel in a cluster. We note that usually sensors are equipped with multi-channel transceivers since multi-channel transceivers can be implemented without much extra hardware complexity and cost. For example, MICA2 series sensor nodes from Crossbow [12] support 433, 868/916, and 310 MHz multiple frequency channels. IEEE 802.11b has 14 channels on physical layer, where three of them can be used concurrently. Multi-channel protocols allow several nodes in a small area to transmit at the same time on different channels without colliding with each other, while at any time a transceiver can send/receive data via only one of the channels. In this paper, we study the problem of data gathering in a hybrid sensor network with multiple channels and show that higher throughput and less active time can be achieved by exploiting multiple channels.

The rest of the paper is organized as follows. Section II overviews existing work in this area. Section III introduces the assumptions used in this paper. Section IV presents the multi-channel polling algorithm for multi-hop sensor clusters. Section V describes the simulation results for the proposed algorithm. Section VI concludes the paper.

II. RELATED WORK

There have been many adaptive protocols on MAC layer [8], [9], [10] proposed to reduce energy consumption in wireless networks. PAMAS [9] uses separate signaling channel to transmit RTS/CTS messages, by which sensors can be informed when and how long they can be power-off. S-MAC [8] periodically puts sensors to sleep to avoid idle listening. Both PARAS and S-MAC are contention-based protocols, in which sensors have to listen to the channel for a while before trying to transmit. This leads to a lot of energy consumption in idle listening. Furthermore, when several sensors want to transmit at the same time, energy could be wasted in collisions. In SMACS protocol [10], after a link between a pair of nodes is established, a fixed time slot is assigned to the link. Since a node knows when to turn its transceiver on ahead of the slot assigned to it and when to turn it off after the slot, idle listening and collision can be minimized. However, once a slot has been assigned to a link between two nodes, other nearby nodes cannot use it, even when the link is idle.

In order to make the protocols scalable in large scale networks, it is more realistic to have hierarchical networks than flat

The research work was supported in part by NSF grant numbers CCR-0207999 and ECS-0427345 and ARO grant number W911NF-04-1-0439.

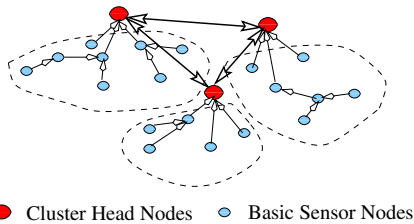


Fig. 1. A two-layered heterogeneous sensor network. The large nodes are the cluster heads. The small nodes are the basic sensor nodes.

ones. Traditional single-hop polling protocols, such as 802.11 PCF [7], require all nodes to be located within the transmission range of the central controller such that all nodes can be polled by the central controller directly without relay. Since one-hop polling algorithms are designed for the short-range communication, such as WLANs and bluetooth networks, it is difficult to be migrated to large sensor networks. [13] [14] gave protocols for randomized cluster forming and cluster head selection. However, they assume all sensor nodes in the network are homogeneous and have the same computing and communication capacity, and every sensor could be elected as the cluster head. Thus, every sensor node has to be powerful enough to communicate with other nodes within the cluster. Furthermore, every node has to be equipped with large memory and complex computing module to handle incoming and outgoing traffic.

III. ASSUMPTIONS

We note that for some applications, for example, environment monitoring, the sensing data is generally collected at a low rate. In addition, sensing data is not so delay-sensitive that it can be accumulated into one or several large data packets and uploaded once a while. For such applications, sensor nodes could turn their transceivers off most of the time to save energy and wake up periodically to upload data to cluster head. In order to improve the performance and energy efficiency for this type of applications, we have the following assumptions.

First, two different types of nodes are deployed in the field: basic sensor nodes and cluster-head nodes. As shown in Fig. 1, cluster heads focus on collecting sensing data from basic sensors and forwarding it to outside observer. For cluster head nodes, the sensing module could be optional. Thus, more energy can be used in forwarding packets. Cluster-head nodes are equipped with more powerful transceiver modules, which allow them to send control messages directly to any sensor node within the cluster. On the other hand, basic sensor nodes are mainly used to monitor environment. Compared to cluster head nodes, their communication modules are relatively simple and cheap. Thus, data packets from one sensor need to be relayed by other sensors to reach the cluster head. The cluster head receives the data by *polling* the sensors. One of the advantages of such a heterogeneous network is that majority of sensor nodes can be made very simple and very cheap, thus the overall cost of the network can be greatly reduced. Second, all sensor nodes and cluster-head nodes are equipped with multi-channel transceivers, which allow them to tune the carrier frequency to different bands. Unlike SMACS protocol, the number of bands is not necessarily large. Third, sensor nodes are densely deployed and can be connected to at least one of the cluster heads. We also as-

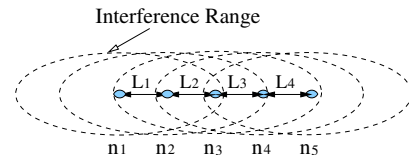


Fig. 2. Connection patterns of nodes n_1, n_2, \dots, n_5 .

sume that the network has been divided into clusters and focus on the in-cluster scheduling.

IV. MULTI-CHANNEL POLING ALGORITHM IN MULTI-HOP CLUSTERS

As mentioned above, sensor nodes wake up periodically to save energy. The time between two consecutive wake-ups is referred to as a *cycle*. The time when sensors are active is referred to as a *duty cycle*. During each duty cycle, all sensed data are uploaded to cluster head. In order to achieve collision-free polling in multi-hop clusters, no more than one conflicting transmissions should occur in the same time slot and with the same frequency channel. The proposed algorithm combines the time slot scheduling and frequency channel assignment to minimize the active time of sensors. The algorithm consists of three steps, and we will explain them in detail next.

A. Discovering Connection and Conflict Patterns

In order to find relay path for sensing data, cluster head needs to know the connectivity or the *connection pattern* of the cluster. Since cluster head can be heard by all nodes within the cluster, it can poll every sensor. The polled sensor can broadcast a Hello message. Then, all other sensors record the received signal level. After all nodes are polled, each sensor reports the received signal levels from all other nodes to cluster head. By comparing the received signal level with decoding threshold of sensor nodes, cluster head can determine if the message from one node can be decoded correctly by the other. If so, the cluster head considers two nodes are connected. Otherwise they are not connected. To avoid packet collision, cluster head also needs to acquire the *conflict pattern* of the cluster, or whether a group of one-hop transmissions are collision free. This can also be done by checking the received signal levels.

For instance, in Fig. 2 nodes n_1, n_2, \dots, n_5 are located along a straight line, which are connected by links L_1, L_2, \dots, L_4 , respectively. Since n_1 is located too far away from n_3 to decode the message from n_3 correctly, n_1 regards the signal sent by n_3 as noise. If the noise from n_3 collides with other packets to n_1 , we say n_1 is located within the interference range of n_3 , and the transmissions through L_1 and L_2 conflict with each other, so do the transmissions through L_1 and L_3 . Note that the conflict between L_1 and L_3 is *co-channel interference*, which is caused by the frequency channel being reused by multiple one-hop transmissions. Co-channel interference can be avoided by assigning different frequency channels to conflicting transmissions. However, the conflict between transmissions through L_1 and L_2 cannot be avoided by introducing new frequency channels, because they are both directed to n_2 and it is impossible for n_2 to communicate with n_1 and n_3 through L_1 and L_2 at the same time. In this paper, we refer to the second type of conflict as *node conflict*.

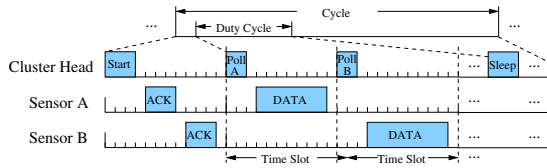


Fig. 3. Timing graph of the polling scheme.

The process of discovering connection and conflict patterns takes $O(N)$ times of polling, where N is the number of sensor nodes in the cluster. Note that due to the lack of an organized infrastructure, during the phase of the neighbor and link discovery, contention-based MAC protocols and simple routing algorithms have to be used. In addition, sensor nodes and cluster head need to tune their radios to the same frequency channel to overhear the shared channel. Fortunately, such simple but inefficient operations only need to be executed in the initialization phase.

B. Finding Relay Paths

Before each duty cycle starts, sensors turn on their transceivers and wait for the message from cluster head. Then, cluster head broadcasts a “start” message to inform sensors the beginning of the duty cycle. After receiving the start message, each sensor replies a short ACK message to cluster head to report how many packets it expects to send during the current duty cycle. ACK messages are relayed to cluster head via the paths indicated by the start message. Given the offered load information and the connection graph of nodes, the cluster head can find an optimal or sub-optimal relay path for each packet according to a given optimization objective. For example, shortest path algorithm is used to minimize the energy consumed per packet [17]. The problem of maximizing the lifetime of sensors in the cluster can be formalized and solved as the maximum flow problem [18]. In the paper, we assume that the relay path from each sensor to cluster head has been determined and focus on scheduling time slots and assigning frequency channels to one-hop transmissions.

C. Time Slot Scheduling and Frequency Bands Assignment

A relay path may consist of several one-hop transmissions. Since sensing data is encapsulated into fixed-length data packets, every one-hop transmission can be done in a fixed-length time interval, or a time slot. All sensors in the cluster are synchronized and operate in a time-slotted manner. Fig. 3 illustrates the timing graph of the polling scheme. At the beginning of a time slot, the cluster head broadcasts a polling message, indicating which sensors should send or receive packets, which sensors should forward packets received at previous time slots, and which sensors should keep idle. After receiving the polling messages, polled sensors will perform their tasks according to the polling message, and unpolled sensors keep idle until the beginning of the next time slot. If a packet is lost and cannot be received by the cluster head at the expected time slot, the cluster head will poll the sensor again. After all packets are received, cluster head will broadcast a “sleep” message to turn off transceivers of all sensors.

In order to achieve collision-free scheduling, at any time slot two conflicting transmissions cannot share the same frequen-

cy channel. However, if two transmissions do not conflict with each other, it is collision-free to assign two transmissions with the same frequency band. The goal of the scheduling algorithm is to minimize the total transmission time, or the total number of time slots. However, the problem of finding an optimal schedule in multi-hop networks has been proved to be NP-hard [16]. We will give a suboptimal algorithm in the following. Considering the limited storage capacity of each sensor node, we assume that each packet must be forwarded to cluster head without delay at any intermediate node. Thus, cluster head only needs to decide when each packet should be sent from its source node to cluster head. Note that the major difference between our algorithm proposed in this paper and the algorithm in [16] is that the former considers both the time domain and the frequency domain, while the latter considers only the time domain.

The cluster head keeps a schedule table and an unscheduled-relay path pool, which are used to keep scheduled and unscheduled relay paths, respectively. At the beginning, a default frequency channel is assigned to all one-hop transmissions of relay paths. Before a time slot, the algorithm finds the schedule only for this time slot. Unscheduled relay paths are picked up from unscheduled-path pool one by one in an arbitrary order. If a relay path does not conflict with existing paths or the conflict can be solved by assigning the conflicting one-hop transmissions with other frequency channels, the relay path will be added to the scheduling table successfully. Otherwise, the relay path is returned to the unscheduled path pool and waits for the next time slot. At the time slot when the packet should have been received by the cluster head, if it is not received, its corresponding relay path will be put into unscheduled path pool again for rescheduling.

For example, Fig. 4(a) gives a network topology of sensors, where $1, 2, \dots, 7$ denote one-hop transmissions. For the sake of simplicity, we assume seven one-hop transmissions pairwise conflict with each other. Thus, the conflict pattern can be expressed by the conflict graph in Fig. 4(b). In Fig. 4(b), any two connected neighbor links cannot be in use concurrently. Assume that each sensor has exactly one packet to send at the beginning of the current duty cycle, and their routing paths to cluster head are $7 \rightarrow 4 \rightarrow 2$, $6 \rightarrow 2$, $5 \rightarrow 1$, $3 \rightarrow 1$, $1, 2$, as illustrated in Fig. 4(c). When all sensors share only one available frequency channel f_1 for transmission, any two conflicting transmission cannot occur concurrently. Fig. 4(d) gives an optimal time slot scheduling for single frequency channel f_1 . Since link $1, 2, 3, 4, 5$ and 6 conflict pairwise, only 7 and 1 can share time slot T_0 . Thus, it takes 10 time slots to upload one packet from every sensor in the case of single frequency channel. Without frequency reuse, the total length of six routing paths is 11 time slots. In Fig. 4(e), two frequency channels f_1 and f_2 are available for transmission. We can see that $7 \rightarrow 4 \rightarrow 2$, 1 and $5 \rightarrow 1$ all begin from T_0 , where 7 and 5 share f_1 and 1 uses f_2 . The total time for forwarding all traffic to cluster head is now 6 time slots, which is much less than that of the single frequency channel case. Fig. 4(f) shows the case that the algorithm reschedule the lost packet on line. The packet relayed by $3 \rightarrow 1$ is expected to arrive to cluster head at the end of T_5 . If the packet cannot be received at the expected slot, cluster head issues a new scheduling for the lost packet again. We can see in

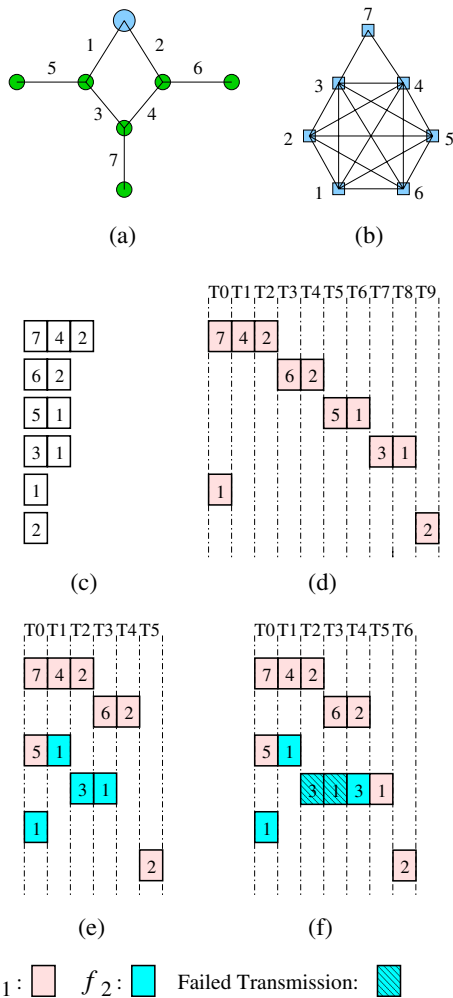


Fig. 4. Time slot scheduling and frequency channel assignment at link layer. (a) Cluster topology; (b) Link conflict graph; (c) Relay paths; (d) Time slot scheduling for one channel f_1 ; (e) Time slot scheduling for two separate channels f_1 and f_2 ; (f) Rescheduling the relay path $3 \rightarrow 1$.

Fig. 4(f), the lost packet is re-sent through $3 \rightarrow 1$ from T_4 and received by cluster head at T_5 .

The procedures for both cluster head and sensor nodes are described in Table 1.

V. SIMULATION RESULTS

We have implemented the proposed algorithm on the NS-2 simulator and in this section we will evaluate the performance of the multi-channel polling algorithm. We assume that all sensor nodes are uniformly deployed within a $200 \times 200m^2$ two-dimensional square and the cluster head is located at the center of the square. Two-ray propagation model is used to describe the feature of the physical layer. With the maximum transmission power $0.858mw$, each node can communicate with other nodes as far as $40m$ away. CBR traffic on the top of UDP is generated to measure the throughput. including header and payload. The simulation runs for 2000 seconds which contains 100 seconds warming-up period. All simulation data are collected from 100 seconds to 2000 seconds.

TABLE 1

MULTI-CHANNEL POLLING ALGORITHM IN MULTI-HOP CLUSTERS

Procedure at cluster head:

```

Poll sensors;
Receive connection and conflict pattern;
while cluster head is not terminated by upper level controller
Broadcast START message;
Receive ACK messages from sensors;
Calculate relay paths;
while not all expected packets are received
    Add collision-free relay paths into scheduling table;
    Broadcast the polling message;
    if the expected packet arrives
        Receive packets;
    else
        Reschedule the relay path for lost packet ;
    end if
    Wait for the next time slot;
end while
Broadcast SLEEP message;
Forward data to outside observer;
end while
    
```

Procedure at a sensor node:

```

Collect and report connection and conflict patterns to cluster head;
while the sensor is not terminated by cluster head
    Receive START message;
    Reply with ACK;
    Receive the message from cluster head;
    while received message is not SLEEP
        if the sensor is polled by cluster head
            Perform the task contained in polling message;
        else
            Keep idle;
        end if
        Keep idle until the next time slot;
        Receive the message from cluster head;
    end while
    Sleep;
end while
    
```

A. Percentage of Active Time

The major goal of our pooling scheme is to reduce the active time of sensors. Fig. 5 shows the percentage of active time needed to ensure all packets are received by the cluster head, where the number of sensors in a cluster ranges from 10 to 100 and data generating rate is $50 Bps$. The percentage of active time when there are 1, 2, 3 and 20 frequency channels are shown in the figure, where the bandwidth of each channel is $200kbps$. Note that the percentage of active time is reduced by half when the number of channel increases from 1 to 2. However, when the number of channels keeps increasing to 3, and finally to 20, the percentage of active time has relatively little improvement. This is because that when the number of channel increases to 2, most co-channel interferences have been resolved and the node conflict becomes dominant, which cannot be resolved by adding more channels.

B. Optimal Number of Frequency Channels with Limited Frequency Resource

From above discussions we know that if the number of frequency channels keeps increasing, the percentage of active time could always be reduced. However, the total frequency resource that can be used by sensors may be limited in some situations. In other words, the product of the number of channels and the bandwidth of each channel should be fixed. In this scenario, we

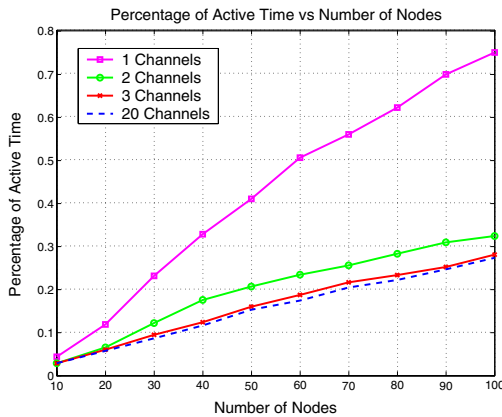


Fig. 5. Percentage of active time vs. number of nodes.

assume that the entire available frequency resource is equally divided into C channels and for simplicity we ignore the bandwidth of guard bands between frequency channels. Apparently, as the number of channels increases, the active time of sensors will likely decrease due to the reduction of co-channel interference. However, if the the number of channels keeps increasing, the benefit of having more channels will decrease and moreover, the time needed to send a packet will increase since the bandwidth of a channel is now small, which will cause the active time of sensors to increase. Therefore, there should be an optimal number of frequency channels with which the active time of sensors is minimized. Through our simulations, we want to reveal the relations between sensor active time and the number of frequency channels to give a guideline to hardware designers for dividing limited frequency resource into the optimal number of channels to minimize the percentage of active time.

We assume that the total available frequency bandwidth for a cluster is 1Mbps , which will be divided equally into a number of channels. 30, 60 and 90 sensors are deployed in the field to see the effect of the density of the nodes. From Fig. 6, we can observe that the percentage of active time for 30 nodes increases monotonously as the number of frequency channel increases, while the curves for 60 and 90 nodes decrease a little at the beginning and achieves the minimum value when the number of frequency channels is 2. The reason for this is that when sensors are deployed sparsely in the field, co-channel interference is not the dominant factor and if the limited frequency resource is divided into C channels, at any time only $\frac{1}{C}$ of the total bandwidth can be used by a one-hop transmission. Thus, the transmission is slowed down, and the percentage of active time goes up. However, when sensors are densely deployed, co-channel interference becomes dominant thus there is some gain by dividing frequency bandwidth into two channels. After that, node conflict becomes the major conflict in the cluster and increasing the number of channel cannot solve the conflict and will only decrease the utilization of frequency resource.

VI. CONCLUSIONS

In this paper we have studied inner-cluster scheduling in hybrid multi-channel sensor networks, where the network is partitioned into clusters and two types of nodes are deployed in the fields, the powerful cluster head and resource-limited basic sen-

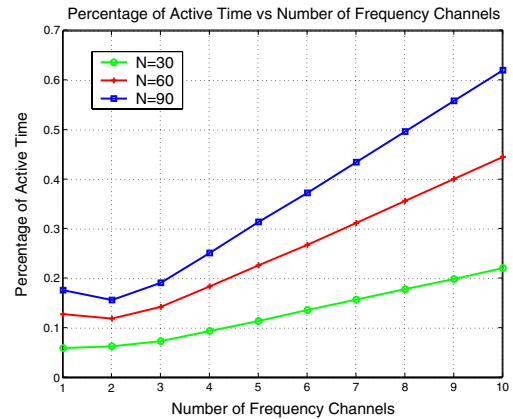


Fig. 6. Percentage of active time vs. number of frequency channels.

sor nodes that are controlled by cluster head. We used polling to obtain data from sensors instead of letting sensors send data randomly, so that less energy is consumed. We gave an algorithm for scheduling time slot and assigning frequency channels to prolong network life. We also conducted simulations on NS-2 simulator, and the results reveal that (1) when the frequency bandwidth is unbounded, increasing the number of frequency channels can reduce the percentage of active time; (2) when the frequency resource is limited, an optimal number of frequency channels can be obtained to minimize the percentage of active time.

REFERENCES

- [1] S. Chessa and P. Santi, "Crash faults identification in wireless sensor networks", *Computer Communications*, vol. 25, no. 14, pp. 1273-1282, 2002
- [2] L. Schwiebert, S. K. S. Gupta, and J. Weinmann, "Research challenges in wireless networks of biomedical sensors," *ACM MobiCom 2001*.
- [3] G. Asada, T. Dong, F. Lin, G. Pottie, W. Kaiser, and H. Marcy, "Wireless integrated network sensors: low power systems on a chip," *European Solid State Circuits Conference*, The Hague, Netherlands, 1998.
- [4] The Ultra Low Power Wireless Sensor Project, http://www-mtl.mit.edu/jimg/project_top.html, 2004.
- [5] A. Cerpa and D. Estrin, "ASCENT: Adaptive self-configuring sensor network topologies," *IEEE INFOCOM 2002*.
- [6] C. Ma, M. Ma, and Y. Yang, "Data-centric energy efficient scheduling for densely deployed sensor networks", *IEEE ICC 2004*.
- [7] IEEE standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, *IEEE Standard 802.11*, 1999.
- [8] W. Ye, J. Heidemann and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," *IEEE INFOCOM 2002*.
- [9] S. Singh and C. Raghavendra, "PAMAS: Power aware multi-access protocol with signalling for ad hoc networks," *ACM Computer Communication Review*, vol. 28, no. 3, pp. 5-26, 1998.
- [10] K. Sohrawi, J. Gao, V. Ailawadhi, and G. J. Pottie, "Protocols for self-organization of a wireless sensor network," *IEEE Personal Communications*, vol. 7, no. 5, pp. 16-27, 2000.
- [11] V. Raghunathan, et. al, "Energy-aware wireless microsensor networks," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 40-50, 2002.
- [12] Crossbow technology, inc. <http://www.xbow.com>, 2005.
- [13] W. Rabiner, et. al, "Energy efficient communication protocols for wireless microsensor networks," *Hawaii International Conference on System Sciences*, 2000.
- [14] O. Younis and S. Fahmy, "Distributed clustering in ad-hoc sensor networks: a hybrid, energy-efficient approach," *IEEE INFOCOM 2004*.
- [15] Alan Amis, et. al, "Max-min D-cluster formation in wireless ad hoc networks," *IEEE INFOCOM 2000*.
- [16] Z. Zhang, M. Ma and Y. Yang, "Energy efficient multi-hop polling in clusters of two-layered heterogeneous sensor networks," in *19th IEEE International Parallel and Distributed Processing Symposium*, Denver, 2005.
- [17] S. Singh, M. Woo, and C. Raghavendra, "Power-aware routing in mobile ad hoc networks," *ACM MobiCom 1998*.
- [18] J.H. Chang and L. Tassioulas, "Energy conserving routing in wireless ad-hoc networks," *IEEE INFOCOM 2000*.