Event-Driven Power Management for Wireless Sensor Networks

Sang Hoon Lee[1], Byong-Ha Cho[1], Lynn Choi[1], Sun-Joong Kim[2]

[1] School of Electrical Engineering
Korea University, Anam-Dong, Sungbuk-Ku, Seoul, Korea
{smile, sntblue, lchoi}@korea.ac.kr

[2] RFID/USN Research Group, Telematics• USN Research Division
ETRI, Daejeon, Korea
kimsj@etri.re.kr

Abstract. In this paper we propose event-driven power management techniques for wireless sensor networks. To accomplish this we model a sensor network application as a set of application-specific events that the application may contain. Events are first classified into scheduled and non-scheduled events. These events are further classified according to the size and the locality of the data, and the real-time characteristics of the event. For scheduled events we propose schedule-driven power control and global coordination. For non-scheduled events we propose source-driven and sink-driven power control for both lower energy consumption and higher performance. Experimentation results confirm that the event driven power management can substantially save energy compared to existing low energy sensor network protocols while it can meet the performance required by the application.

1 Introduction

Energy efficiency has been one of the key issues in implementing wireless sensor networks. Although a wide variety of sensor network protocols have been proposed [4, 5, 7, 10], the existing low-energy protocols are protocol-specific in a sense that they do not collaborate with the power management functions of upper or lower layers, limiting their scope. For example, when a source node reports an event to a sink, not only the nodes on the communication path but also all the other idle nodes repeatedly wake up unnecessarily. Furthermore, the busy nodes on the communication path still employ the periodic wake up and sleep during the event processing, which would substantially degrade the network performance. This can be attributed to the fact that each node decides its power management action without knowing its context, i.e. the state of the application or the characteristics of the on-going event.

This work was supported by the research commissioned by the Electronics and Telecommunications Research Institute.

In this paper, we investigate ways of exploiting the application-level information to further improve the energy efficiency of networking protocols. To accomplish this, we characterize the sensor network events by using the following classification parameters: the event timing, the size and the locality of the report data, and the real-time characteristics of the event. A sensor network application is then modeled as a set of application-specific events that the application may contain.

Sensor network events are largely classified into scheduled and non-scheduled events. For scheduled events we propose *schedule-driven power control* and *global coordination*. For non-scheduled events we propose *source-driven* and *sink-driven power control*. With these event-driven power management (EPM) techniques, all the nodes on the communication path fully wake up during the event processing while all the non-participating nodes may not need to wakeup at all. This is controlled by each event source, i.e. a source node in the case of source-driven event, a sink node in the case of sink-driven event, and a report timer in the case of a scheduled event. The full duty-cycle operation during the event processing can not only reduce the message delay but also can reduce the time spent on idle listening by increasing the interval of the periodic wakeup during an idle state. Our detailed simulation results show that the energy savings achieved by EPM range from 29% to 94% depending on the application scenarios. Furthermore, EPM can also reduce the average message delay by up to 98% by employing the full duty-cycle operation on demand.

2 Application Model

2.1 Event Classification

In this work we define an event as an incident where a report needs to be sent to the sink. An event may occur due to a sensing activity by a sensor node, a query generated by a sink, or a local report timer at a sensor node since all of these activities may generate a report to the sink. Thus, an event is always associated with the generation of a report. However, a sensing activity can be performed regardless of the report. In this sense a sensing is regarded as a means to recognize an event.

We can classify events based on the following parameters: the timing of the event, the data characteristics of the report, and the real-time characteristics of the event. When an event is scheduled at a specific time, the event is called a *scheduled event*. Scheduled events are further classified into periodic and non-periodic events. *Periodic events* generate reports at every constant interval, such as hourly, daily, weekly, or monthly. *Non-periodic events* occur at predetermined times but not periodically. When an event occurs non-deterministically, the event is called a *non-scheduled event*. Non-scheduled events are classified into source-driven and sink-driven events. *Source-driven events* are asynchronous events that are triggered by a sensing activity at a sensor node, i.e. a source. *Sink-driven events* are triggered by a

query sent by a sink and is considered as another type of non-scheduled asynchronous events.

Events can be further classified according to the size and the redundancy characteristics of the data that are reported. Depending on the size of the data events can be classified into *single data events* and *burst data events*. A single data event requires a report of a small data item such as the temperature or humidity, leading to the generation of a few data packets. On the contrary, a burst data event requires a report of a large data such as images or videos, leading to the generation of a packet stream. The redundancy characteristics of the data are closely related with the locality characteristics of the event. When an event can be detected by multiple sensor nodes nearby, the event is classified as a *spatial locality event*. For this type of event, only a single source needs to report the event. If a node detects multiple consecutive events but there exist a significant redundancy among the data reports, the events can be classified as a *temporal locality event*. In this case the node can summarize or aggregate the data before sending out a report. This locality characteristic of an event determines the type of aggregation that can be performed for the event.

The real-time characteristic of an event is related with the latency tolerance characteristics of an application for the event. The events with hard or soft deadlines are classified as a *real-time event* since the deadline must be met by the network. Events that can tolerate a considerable latency are classified as a *non-real time event*.

Table 1. Event classification factors.

Event class	Non-scheduled event	Sink-driven non-scheduled event	NS_{Sink}
		Source-driven non-scheduled event	NS_{Source}
	Scheduled event	Periodic event	$S_{Periodic}$
		Non-periodic event	$S_{Non\text{-}periodic}$
Data characteristic	Data size	Single data	D_{Single}
		Burst data	D_{Burst}
	Data redundancy	Spatial locality	$DR_{Spatial}$
		Temporal locality	$DR_{Temporal}$
		Spatial & temporal locality	$DR_{Locality}$
		No locality	$DR_{No\text{-}locality}$
Latency tolerance	Real time		L_{Real}
	Non-real time		$L_{Non\text{-}real}$

2.2 Application Model

Table 1 shows our event classification parameters and their corresponding notations. Using the notation an event can be classified as a tuple, {event type, data size, data redundancy, latency tolerance}. Table 2 classifies the major event types of several well-known sensor network applications according to our classification parameters. For example, the most common event of volcanic monitoring application is classified as a source-driven, non-scheduled, single-data, real time event since the volcanic alarm must be reported within a limited delay. However, this application may have a periodic report of regional temperature and its image on an hourly basis, suggesting

that it may include a periodic scheduled, burst-data, non-real time event. Thus, a sensor network application in general can be viewed as a set of different event classes rather than a single event class.

Table 2. Event classification of several well-known sensor network applications.

Applications	Event class
Great Duck Island Project [6]	$\{ S_{Periodic}, D_{Single}, DR_{Temporal}, L_{Non\text{-}real} \}$
James Reserve Extensible Sensing System [3]	Climate: $\{ S_{Periodic}, D_{Single}, DR_{Temporal}, L_{Non\text{-}real} \}$ Wildlife: $\{ S_{Non\text{-}periodic}, D_{Single}, DR_{Spatial}, L_{Non\text{-}real} \}$
Volcanic monitoring [8]	$\{ NS_{Source}, D_{Single}, DR_{No\text{-}locality}, L_{Real} \}$
CORIE [2]	$\{ S_{Periodic}, D_{Single}, DR_{Temporal}, L_{Non\text{-}real} \}$
FabApp [5]	$\{ S_{Periodic}, D_{Single}, DR_{No\text{-}locality}, L_{Non\text{-}real} \}$
CodeBlue [1]	Monitoring: $\{ S_{Periodic}, D_{Single}, DR_{No\text{-}locality}, L_{Real} \}$ Alert: $\{ NS_{Source}, D_{Single}, DR_{No\text{-}locality}, L_{Real} \}$
Traffic pulse technology [9]	$\{ S_{Periodic}, D_{Single}, DR_{Temporal}, L_{Real} \}$

2.3 Application State

An application state can be specified by the type of event that the application is currently processing. Figure 1 shows the state transition diagram of a general sensor network application that has all three different event classes, i.e. scheduled event, source-driven event, and sink-driven event. On a deployment, the application starts from the initial state. In the initial state the self-organizing nature of the network requires all the network setup functions to be completed such as the routing path setup and the global time synchronization. After the setup process is complete, the application goes to the idle state and is ready to process any event. During this idle state, a node may need to wake up to detect a non-scheduled event. As discussed, the sensing activity is not regarded as an event and is processed locally by each node. Depending on the event source, i.e. the report timer, query, or an asynchronous event triggered by a sensing activity, the application goes to the corresponding state that handles the particular event class. Sometimes, another event may occur during the processing of an event. We assume that each event is processed in order. Thus, after the first event is completed, the application goes back to the idle state, and immediately makes a transition to process the second event.

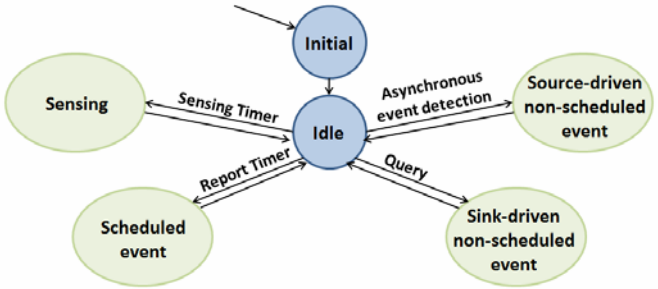

Fig. 1. Event-driven state transition diagram of an application.

3 Event-Driven Power Management Techniques

3.1 Application-Specific Protocol Customization

The parameters of a networking protocol must be customized according to the performance requirements of a target application during the network design stage. We call this process *protocol customization*. The application characteristics that can influence the protocol customization include the size of the network field, the maximum tolerable latency for each event class, and the size of data types. In addition, the network designer must consider the physical characteristics of the sensor node.

3.2 Event-Driven Dynamic Power Management

3.2.1 Scheduled events

For a scheduled event, a node needs to wake up only during the scheduled report period. Thus, the usual periodic wakeup and sleep employed by an existing MAC protocol can be completely eliminated if there are no other event classes in the application. The wakeup and sleep of a node can be controlled precisely by the report timer since the report schedule is prescheduled. This is called the *schedule-driven power control*. Both periodic and non-periodic events can be handled by the same technique. To meet the delay requirement of the event, the wakeup time must account for the worst-case communication delay, which can be computed by using the protocol customization procedure described in Section 4.2. A node must wake up if it is on the communication path from a source to a sink. If a single node is generating a report, all the nodes in the path from the source to the sink must wake up. All the non-participating nodes may not need to wake up at all. Thus, the idle listening can be completely eliminated if a node is not participating. This is called a *single-source event*. If multiple nodes are generating reports, the event is called a *multiple-source event*. In this case the worst-case communication delay must consider the delay due to the contention. If all the nodes in the network are generating reports, the scheduled event is called a *global event*.

In the case of a global event, all the nodes may start transmitting the report messages simultaneously. This is called *parallel transmission*. Although this might reduce the message delivery latency for each node, this may cause a significant delay in the MAC layer due to the contention caused by simultaneous transmissions. An alternative approach is that each node waits until it combines the messages from all of its descendants in the routing tree. If the event has a spatial locality, an aggregation can also be applied and the size of the report message can be further reduced. This is called *global coordination*. This technique reduces the number of transmissions by reordering the transmissions in the sensor field.

3.2.2 Source-driven non-scheduled events

A node cannot predict the timing of a source-driven non-scheduled event. In addition, a node may not determine how many sources are participating in such an event. Thus, neither the global coordination nor the schedule-driven power control can be applied. Instead, a node must periodically wake up to check if such an event has occurred.

For the source-driven events, we can apply the *source-driven power control*. In this scheme, a source detecting an event can notify the occurrence of the event to all the nodes in the routing path from the source to the sink by sending a *wakeup signal* to the sink. After receiving the signal, a node must fully wake up to process the message. This full wakeup not only reduces the message delivery latency but also can increase the cycle time of the periodic wakeup and sleep during an idle state. If the event has a spatial locality, multiple sources can report the same event. Thus, a node must wake up as long as there is an active wakeup signal from any of its descendants. After the report is complete, then each source can send a *sleep signal* to request the nodes on the routing path to go back to an idle state. The RTS packet [10] can be used as a wakeup signal since the destination address field of the packet can be used to designate the recipient to wake up. In addition, the fragment flag of the normal data packet header can be used as a sleep signal since a source node can indicate if the current packet is the last data packet of a message by setting this flag.

3.2.3 Non-scheduled sink-driven events

A sink node may generate a sink-driven event by sending a query. Like a source-driven event, a sensor node must check the occurrence of a sink-driven event by employing a periodic wakeup since it cannot predict the timing of such an event.

For the sink-driven events, we can apply the *sink-driven power control*. In this scheme, the sink node sending a query can notify the occurrence of the event to the sensor field. Thus, the query itself is considered as a wakeup signal and all the nodes receiving the query must wake up to process the query message. The sink sends a sleep signal after it receives all the report messages, signaling the completion of the query processing. Thus, a separate power down message is needed to implement the sink-driven power control.

4 Experimentation and Results

To evaluate both the energy efficiency and the network performance of EPM, we have implemented all the proposed EPM techniques in the NS-2 simulator framework. As a baseline routing protocol, we used the virtual sink rotation (VSR) routing [4] which is able to support multiple mobile sinks using the tree-based routing topology. Such a tree-based routing protocol is used since the global coordination assumes a tree-based topology. In addition, we use S-MAC [10] as an underlying MAC protocol, which assumes a periodic wakeup. We have implemented the schedule-driven, source-driven, and sink-driven power control to the underlying S-MAC protocol.

We use two metrics: average dissipated energy and average message delay. The *average dissipated energy* measures the ratio of total dissipated energy per node in the network to the number of distinct events seen by sinks. This metric computes the average work done by a node in delivering useful sensor data to the sinks. The *average message delay* measures the average latency observed from the time when an event is detected to the time when the last packet has arrived at the sink.

4.1 Benchmarks and Network Configurations

We use seven application scenarios as benchmarks as shown in Table 3. Applications 1 through 4 are a single event class application. Application 1 consists of a scheduled, non-real time, single data event class with spatial locality and simulates a climate monitoring system. Application 2 has the same event class as the application 1 but without locality. Application 3 consists of a non-scheduled, source-driven, hard real time, burst data event class and simulates an intrusion detection system with a camera. Application 4 is an example of a non-scheduled sink-driven, soft real time, burst data event class. This application can be viewed as a wildlife animal tracking system. Applications 5, 6, and 7 are various combinations of these single-event classes.

Table 3 also shows the characteristics of the network configurations used for the simulation. A 100-node sensor field is generated by placing the nodes in a 10x10 grid. A sink node is located at the center of the field. The size of a MAC packet is 200 bytes. A sensor node spends 33mW, 15mW, and 0mW at transmit, receive/idle, and sleep mode respectively. This is consistent with previous studies [10].

Table 3. Benchmarks and network configurations used for the simulation.

<table>
<tr><th></th><th>App. 1 (scheduled)</th><th>App. 2 (scheduled)</th><th>App. 3 (source-driven)</th><th>App. 4 (sink-driven)</th><th>App. 5</th><th>App. 6</th><th>App. 7</th></tr>
<tr><td>Report period</td><td colspan="2">1 hour</td><td colspan="2">-</td><td rowspan="4">App. 1+ App. 3</td><td rowspan="4">App. 2 + App. 4</td><td rowspan="4">App. 2 + App. 3 + App. 4</td></tr>
<tr><td>Data size</td><td colspan="2">32B</td><td colspan="2">64KB</td></tr>
<tr><td>Aggregation</td><td>no</td><td>yes</td><td>no</td><td>no</td></tr>
<tr><td>latency tolerance</td><td colspan="2">1 hour</td><td>10 minutes</td><td>30 minutes</td></tr>
<tr><td>Network Top.</td><td colspan="7">10 * 10 grid</td></tr>
<tr><td>Num. of nodes</td><td colspan="7">101 nodes (100 sensor nodes + 1 sink node)</td></tr>
<tr><td>MAC packet size</td><td colspan="7">200 bytes</td></tr>
<tr><td>Routing protocol</td><td colspan="7">VSR</td></tr>
<tr><td>MAC protocol</td><td colspan="7">S-MAC</td></tr>
<tr><td>Power Cons.</td><td colspan="7">Tx: 31mW, Rx/Wakeup: 15mW, Idle: 0mW</td></tr>
</table>

4.2 Protocol Customization

At the network design stage, the network designer must determine the sleep and wakeup schedule of each node which can guarantee the maximum tolerable latency. For a MAC protocol employing a periodic wakeup and sleep, such as S-MAC, the sleep and wakeup schedule can be expressed by the node's cycle time (t). A *cycle* is

defined as the periodic interval, which consists of an active period and a sleep period [7]. The *cycle time t* should be long enough to accommodate a single data packet transaction, i.e. the sequence of SYNC, RTS, CTS, DATA and ACK packets. This is called the *minimum cycle time* (t_{min}), which can be calculated from the size of each packet, the contention window size for each packet, the transmission delay, and the RF transmission parameters of a given node. To derive the maximum cycle time permitted by the application, the longest path length in the network (N), the maximum tolerable latency for each event class (L), and the size of a message (S) need to be considered. Assuming that there is no other traffic, the latency of a single packet over N hops [10] under both S-MAC and EPM can be given by

$$\text{single-packet message delay under S-MAC / EPM} = N*t \quad (\mathbf{1})$$

If multiple packets are transmitted consecutively on the same path, each packet needs to be separated at least 3 hops apart to avoid collision [10]. All the packets except the first one suffer from this additional delay for a multi-packet message. The delay of a multi-packet message assuming zero traffic can be given by

$$\text{multi-packet message delay under S-MAC} = N*t + 3(S-1)*t \quad (\mathbf{2})$$

While the multi-packet message delay of S-MAC can be expressed by (2), the message delay of EPM is lower, because a node works at the full duty-cycle after receiving the first packet. The multi-packet message delay of EPM can be given by

$$\text{multi-packet message delay under EPM} = N*t + 3(S-1)*t_{min} \quad (\mathbf{3})$$

Since the message delay must be smaller than L, the cycle time t can be derived from the equations (1) through (3). Table 4 shows the derived cycle times and the corresponding duty-cycles for our benchmarks under S-MAC and EPM. For applications 5, 6, or 7, the cycle time must be determined by the event class which requires the lowest latency. Note that the cycle time is not applicable to EPM for scheduled events since it does not employ the period wakeup and sleep for such events. However, due to the distributed clock synchronization required by S-MAC, each node wakes up at least every 30 seconds even under EPM.

Table 4. The cycle times and the duty cycles of S-MAC and EPM derived for our benchmarks.

	App. 1	App. 2	App. 3	App. 4	App. 5	App. 6	App. 7
S-MAC	15s(1%)	15s(1%)	0.6s(25%)	1.8s (8%)	0.6s(25%)	1.8s(8%)	0.6s(25%)
EPM	N.A.	N.A.	48s(0.3%)	288s(0.05%)	48s(0.3%)	288s(0.05%)	48s(0.3%)

4.3 Simulation Results

Figure 2 shows the average dissipated energy and the average message delay for each event class in our benchmarks. For the scheduled events, EPM eliminates unnecessary wakeup during an idle state, reducing the energy consumption by up to 39% compared to S-MAC. Note that the global coordination is only effective for application 2 but the additional energy savings are relatively small. The average

message delay of EPM is substantially smaller than that of S-MAC since each node can act with full performance during the event processing. For the source-driven event, the much higher duty cycle (25%) required by the S-MAC substantially increases the idle listening compared to EPM, which has a duty cycle of only 0.3%. As a result, an idle node with S-MAC requires 36.5 times more energy and a busy node with S-MAC requires 2.4 times more energy compared to EPM in this simulation. The average message delay of EPM is smaller than the maximum tolerable latency since the clock synchronization requires each node to wake up more frequently than that required for the tolerable latency. Like the source-driven power control, the sink-driven power control increases the network performance and reduces the energy consumption for the sink-driven event. EPM can eliminate 92% of the idle listening energy in S-MAC.

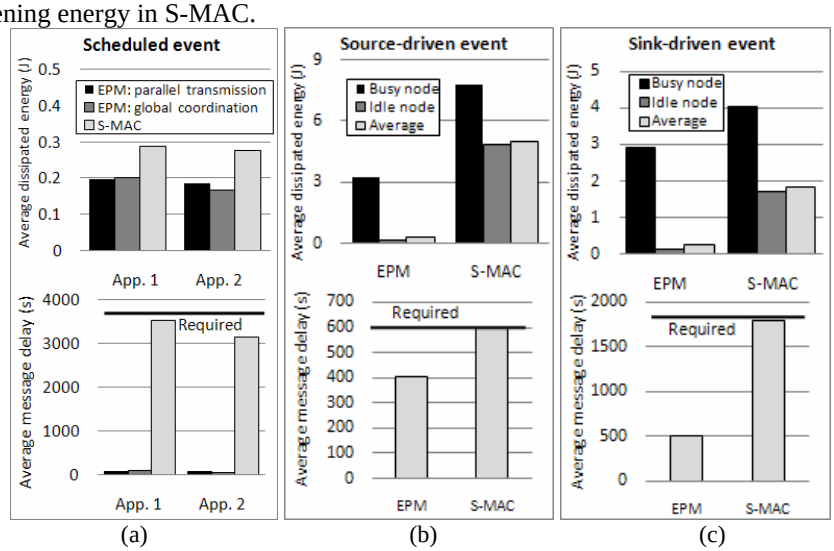

Fig. 2. Average energy consumption and average message delay (a) for a scheduled event, (b) for a non-scheduled source-driven event, and (c) for a non-scheduled sink-driven event.

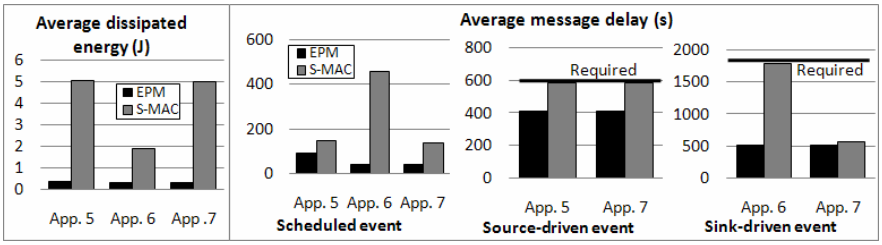

Fig. 3. (a) The average energy consumption for applications with multiple event classes. (b) The average message delay for applications with multiple event classes.

Figure 3 shows the case for multiple-event class applications. In the figure we only show the result of EPM assuming global coordination. As show in Figure 3(a) EPM

can successfully eliminate up to 94% of the per-node dissipated energy compared to S-MAC. Since the event class with the minimum tolerable latency determines the cycle time of a node, usually a non-scheduled real-time event plays a dominant role in determining both the dissipated energy and the message delay of an application.

Figure 3 (b), (c), and (d) compares the average message delay of both S-MAC and EPM for a scheduled, source-driven, and sink-driven events respectively. Scheduled events have lower delay compared to single-event cases since the protocol is customized to meet the lowest latency required by the non-scheduled events.

5 Conclusion

In this paper we explore ways of exploiting the application state information at runtime to efficiently manage the energy and the performance of the networking protocols. To accomplish this we model a sensor network application as a set of application-specific events and propose various event-driven power management (EPM) techniques. EPM dynamically controls the operating mode of the protocols depending on the event currently processed. Our detailed simulation results show that EPM can substantially reduce the energy consumption of a node by successfully removing unnecessary wakeups during an idle state while it also reduces the message delay by employing a full duty-cycle operation during a busy state.

References

1. CodeBlue: http://www.eecs.harvard.edu/~mdw/proj/codeblue/
2. CORIE: http://www.ccalmr.ogi.edu/CORIE/
3. James Reserve Extensible Sensing System: http://www.jamesreserve.edu/
4. Lynn Choi, Kwangseok Choi, Jungsun Kim, Byung Joon Park: Virtual Sink Rotation: Low-Energy Scalable Routing Protocol for Ubiquitous Sensor Networks: In Proceedings of the USN, (2005) 1128 – 1137
5. Ramanathan, N., Yarvis, M., Chhabra, J., Kushalnagar, N., Krishnamurthy, L., Estrin, D.: A Stream-Oriented Power Management Protocol for Low Duty Cycle Sensor Network Applications: In Proceedings of the EmNets, (2005) 53 – 62
6. Robert Szewczyk, Alan Mainwaring, Joseph Polastre, John Anderson, David Culler: An analysis of a large scale habitat monitoring application: In Proceedings of the 2nd international conference on Embedded networked sensor systems, (2004) 214 – 226
7. Sang Hoon Lee, Joon Ho Park, Lynn Choi: AMAC: Traffic-adaptive Sensor Network MAC Protocol through Variable Duty-Cycle Operations: To appear, The IEEE International Conference on Communications, (2007)
8. Sensor network for volcanic monitoring: http://www.eecs.harvard.edu/~mdw/proj/volcano
9. Traffic pulse technology: http://mobilitytechnologies.com/index.html
10. Ye, W., Heidemann, J., Estrin, D.: Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks: IEEE Transactions on Networking, (2004)