A Low-Energy Adaptive and Distributed MAC Protocol for Wireless Sensor-Actuator Networks

April 5th, 2006

Muhammad Farukh Munir and Fethi Filali

---

1Institut Eurécom’s research is partially supported by its industrial members: Bouygues Télécom, Fondation d’entreprise Groupe Cegetel, Fondation Hasler, France Télécom, Hitachi, Sharp, ST Microelectronics, Swisscom, Texas Instruments, Thales.
A Low-Energy Adaptive and Distributed MAC Protocol for Wireless Sensor-Actuator Networks

Muhammad Farukh Munir and Fethi Filali

Abstract

Sensor-actuator networks are often limited in battery capacity and processing power. Therefore, it is exigent to develop solutions that are both energy and delay efficient. In this paper, we propose a low-energy and delay-sensitive TDMA based MAC for wireless sensor-actuator networks (WSANs). These networks are organized into clusters and each cluster is managed by a single actuator. To avoid inter-cluster interference, we employ an easy use of CDMA codes. The actuators schedule sensors to reduce the total network energy consumption per successful transmission. We identify the advantages of our proposal over existing TDMA-MAC schemes for cluster-based sensor networks. Our simulation results demonstrate that the proposed MAC greatly improves the WSAN lifetime and achieves a good trade-off between the packet delay and sensor energy consumption.
Keywords: WSANs, MAC, CSMA, TDMA, CDMA, wakeup protocols, energy efficiency, delay guarantee.
Contents

1 Introduction and related work 1
2 Assumptions 2
3 Network Model 3
4 Design of LEAD-MAC 4
  4.1 Network Learning Phase .................................. 5
  4.2 Scheduling Phase ........................................... 5
  4.3 Adjustment Phase ........................................... 6
5 LEAD-Wake up Protocol 6
  5.1 Adaptivity to Network Conditions ......................... 6
  5.2 Analysis of LEAD Wake up protocol .................... 7
6 Simulation Results 9
7 Conclusions and Future work 9
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Architecture of SANETs</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Energy Savings through adaptive duty cycle</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>Average lifetime of a cluster as a function of increasing the number of nodes</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>Average delay in a cluster as a function of the number of nodes</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>Average energy consumption in a cluster as a function of the number of nodes</td>
<td>11</td>
</tr>
</tbody>
</table>
1 Introduction and related work

SANETs\(^1\) (Sensor-Actuator Networks) shown in Fig. 1, are appealing to researchers due to their wide spread of application potential, ranging from densely deployed habitat-monitoring setup to the real-time security applications. Sensor nodes are small, cheap devices with reduced processing and communication capabilities. They usually gather information about the physical world. Whereas, the actuator\(^2\) nodes are expensive, resource rich, and higher communication capability devices.

The multi-actuator architecture raises many interesting issues such as cluster formation, cluster-based sensor organization, network management and task allocation among the actuators. In this paper, we only focus on the issues of network management within the clusters, particularly energy-aware medium access control (MAC) layer protocol and inter-cluster interference issues. The energy efficiency at the MAC layer has recently received attention, especially with the increasing interest in the applications of unattended sensor networks. The S-MAC [3] enables low-duty-cycle operation in a multi-hop sensor network. Nodes form virtual clusters based on **fixed common sleep schedules** to reduce control overhead and enable traffic-adaptive wake-up. T-MAC [5] extends S-MAC by adjusting the length of time sensors are awake between sleep intervals based on the communication of neighboring sensors. To achieve low power operation, B-MAC [6] employs an **adaptive preamble sampling scheme** to reduce duty cycle and minimize idle listening which is a basic source of energy drain. Whereas, the Z-MAC proposal [7] combines the strengths of TDMA and CSMA while offsetting their weaknesses by switching the MAC to CSMA and TDMA at low and high contention periods, respectively. The performance of Z-MAC falls even below B-MAC in the case of low contention, so it is a more suited protocol for medium to high data rate applications.

In [8], the authors presented two scheduling schemes (breadth-first and depth-first assignment) for a **cluster based** sensor network. The proposed TDMA-MAC is shown to perform well in terms of energy-efficiency and end-to-end delay depending on the choice of scheduling scheme. The gateway nodes transmit the schedule in their cluster using larger transmission power. This introduces a new problem of schedule **interference** among neighboring clusters and is not discussed in the paper. PEDAMACS [4] proposal for sensor networks has utilized the presence of a powerful node called **access-point** (AP) among sensors which takes the transmission load from the constrained sensors and is further responsible for the reliable delivery of sensor data toward the sinks. In case of multiple APs, the neighboring APs should not transmit their coordination packets at the same time to avoid **inter-cluster interference**. The APs should take into account the sensors that are outside their longest range while generating the schedule. The power levels of the APs are adjusted so that the schedule can reach all the sensors in the cluster. If all the sensors cannot be reached by the schedule, they can still be scheduled at the

---

\(^1\)The terms SANET and WSAN can be interchangeably used if SANETs are wireless.

\(^2\)In design, actuators/access-points/gateways can all be thought of similar devices with higher processing and communication capabilities. Whereas, in practice, the actuators need to have some extra actuation capabilities.
cost of an increased synchronization overhead apart from the increased delay and energy consumption\textsuperscript{3}.

We propose LEAD-MAC (Low-Energy, Adaptive and Distributed-MAC) protocol for SANETs, which ensures to minimize the end-to-end delay, improves throughput, and conserves sensor energy by using an adaptive wakeup scheme. The actuators compute a hybrid (involves both depth-first and breadth-first scheduling) collision-free TDMA schedule for the sensors in their local-cluster and avoid interference using CDMA\textsuperscript{4} codes.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{architecture.png}
\caption{Architecture of SANETs}
\end{figure}

In Section 2, we detail the assumptions for the design of LEAD-MAC. Section 3 describes the network model for SANETs. The detailed design of LEAD-MAC is presented in Section 4. A delay-energy analysis of the Wakeup protocol is given in Section 5. In Section 6, we present the simulation results. Section 7 concludes the paper and outlines the future directions.

\section{Assumptions}

The assumptions under which we have designed the LEAD-MAC protocol are as follows:

\textsuperscript{3}We remark here that our present observations are not aimed at questioning the significance of [4] and the related TDMA based MAC proposal for wireless sensor networks. Most of these studies never aimed at looking the problem of multiple APs from system design view point. In our work, however we are trying to get the best system performance, hence we need to manage each cluster optimally without sacrificing system performance in terms of end-to-end delay and energy consumption.

\textsuperscript{4}A sensor selects a CDMA code based on the identity of the actuator, so the available codes are limited to the number of actuators in the network. All the sensors in any given cluster would generate the same CDMA code.
1. We consider a wireless ad hoc network that consists of a large number of sensor nodes along with a few uniformly distributed actuator nodes. Sensors do the application dependent sensing and transmit their sensed data towards their optimal actuators.

2. Both sensors and actuators are static, capable of adjusting their transmission power, and a link between any two nodes is bidirectional.

3. The actuators can reach all the sensors in their local cluster in one-hop using maximum transmit power.

3 Network Model

Consider a static wireless sensor network with \( n \) sensor nodes and \( m \) actuator nodes as shown in Fig. 1. Given is an \( (n + m) \times (n + m) \) neighborhood relation matrix \( R \) that indicates the node pairs for which direct communication is possible. We will assume that \( R \) is a symmetric matrix, i.e., if node \( i \) can transmit to node \( j \), then \( j \) can also transmit to node \( i \). For such node pairs, the \((i, j)^{th}\) entry of the matrix \( R \) is unity, i.e., \( R_{ij} = 1 \) if node \( i \) and \( j \) can communicate with each other; we will set \( R_{ij} = 0 \) if nodes \( i \) and \( j \) can not communicate. For any node \( i \), we define \( N_i = \{ j : R_{i,j} = 1 \} \). Which is the set of neighboring nodes of node \( i \). Similarly, a set of interference nodes (cannot be reached by one-hop) for node \( i \) (from where the transmissions can be heard at node \( i \)), and is defined as

\[
S_i = \{ K \notin N_i \cup \{i\} : R_{k,j} = 1 \text{ for some } j \in N_i \}
\]

Note that \( S_i \) does not include any of the first-hop neighbors of node \( i \).

The topology of the network is represented by a graph \( G = (V,E) \), in which \( V \) is the set of nodes (both sensors and actuators). The edges \( E \in V \times V \) are such that \((i, j) \in E \) if nodes \( i \) and \( j \) can transmit to each other.

**Power Consumption Model:** For a sensor node, the energy consumption due to wireless communication (i.e. receiving and transmitting) is considered the dominant source in power consumption. If power consumed to receive a single multi-hop packet (for design, we assume all packets to be of same length) is given by \( P_{rx} \) (in \( j/pkt \)), then the power consumed \( P_{r}^i \) (in \( j/s \)) by a sensor node \( i \) for receiving is

\[
P_{r}^i = P_{rx} \sum_{j \in N_i} \alpha_{j,i}
\]

where \( \alpha_{j,i} \) is the rate (\( pkt/s \)) at which node \( j \) is transmitting packets toward node \( i \). If the power consumed to sense and sample a packet is \( P_{sense} \) (in \( j/pkt \)), then the power consumed \( P_{s}^i \) (in \( j/s \)) by a sensor node \( i \) in sampling packets is

\[
P_{s}^i = P_{sense} \lambda_i
\]

---

*Optimal refers to the outcome of a cost-function, e.g., min-hop or min-delay routing.

*Conceptually, we can assume that this actuator is also a sensor node, which has 0 sampling rate.
where $\lambda_i$ is the rate ($\text{pkt/s}$) at which node $i$ performs environmental sensing. If the power consumed to send a packet is given by $P_{tx}$, then the power consumed $P^i_t$ (in $j/s$) by a sensor node $i$ in transmitting its data (both locally originated and forwarded packets) is

$$P^i_t = P_{tx} \sum_{j \in N_i} \alpha_{i,j}$$

When the packets arrive from $S_i$ due to interference, the power loss $P^i_{in}$ (in $j/s$) at node $i$ is

$$P^i_{in} = P_{rx} \sum_{j \in S_i} \alpha_j$$

where $\alpha_j$ in ($\text{pkt/s}$) is the total rate at which node $j$ is transmitting: $\alpha_j = \sum_{k \in N_j} \alpha_{j,k}$

If node $i$ is neither serving its forwarding queue nor sampling a new packet, it is in idle state. If the power consumed in idle state is given by $P_{idle}$, then the power consumption $P^i_{id}$ (in $j/s$) by a sensor node $i$ is given by

$$P^i_{id} = P_{idle} \left(1 - \sum_{j \in N_i} \frac{\alpha_{i,j}}{C} - \sum_{j \in N_i} \frac{\alpha_{j,i}}{C}\right)$$

where $C$ is the transmission capacity in $\text{pkts/s}$.

Network Lifetime $T_{\text{network}}$ is defined as the time spanned by the network before first node death as a result of energy outage. The lifetime of a sensor node $i$ having battery capacity $E_i$ is given by (1).

$$T_{\text{life}} = \frac{E_i}{P_{rx} \left(\sum_{j \in N_i} \alpha_{j,i} + \sum_{k \in S_i} \alpha_k\right) + P_{rx} \sum_{j \in N_i} \alpha_{i,j} + P_{sense} \lambda_i + P_{idle} \left(1 - \sum_{j \in N_i} \frac{\alpha_{i,j}}{C} - \sum_{j \in N_i} \frac{\alpha_{j,i}}{C}\right)}$$

(1)

$$T_{\text{network}} = \min_i T_{\text{life}}$$

The lifetime $T_{\text{life}}$ can also be maximized by controlling the flow coming into a node and its service rate using an adaptive routing protocol. But in this paper, we do not discuss any routing layer details. For the considered model, we try to optimize the system performance using a TDMA-MAC protocol by minimizing the awake periods and power loss due to interference.

4 Design of LEAD-MAC

LEAD-MAC has three operational phases: (i) network learning phase, (ii) Scheduling phase, and (iii) adjustment phase. The following discussion covers the different protocol phases in detail.
4.1 Network Learning Phase

A sensor node finds an optimal actuator using the proposed ADP (Actuator Discovery Protocol, a controlled flooding mechanism [1]), during the initial deployment phase. The sensors start the learning phase by transmitting a one hop broadcast `actuator-search_request`, using their lowest transmit power. When a broadcast reaches an actuator, it is replied with the actuator identity. A random access scheme is used in the topology learning phase, because the sensors do not yet have a transmission schedule. The scheme is designed so that, at the end of this phase, almost all nodes are attached (based on minimum-hop routing) to an actuator and correctly determine their neighbors and interferers with high probability. We adopt a carrier sense multiple access (CSMA) mechanism similar to 802.11 [9]. The sensors listen for a random time before transmitting, and transmit if the channel is idle. A random delay is added before carrier sensing to further reduce collisions. However, because a collision will lead to incomplete cluster information at the actuators, the CSMA scheme itself cannot guarantee that an actuator will receive the full cluster information. Therefore, we proposed to include an implicit acknowledgment from the actuator, which occurs when a sensor transmits a packet to join a particular cluster. Based on this acknowledgment, a sensor selects a CDMA code to communicate with its actuator.

4.2 Scheduling Phase

The actuator explicitly schedules all the sensors, based on its knowledge of the cluster. An actuator schedules the sensors in the depth-first order for end-to-end routes, and in a breadth-first order for any given parent node $i$, to capture forwarded data from all of its downlink sensors. We don’t provide the scheduling algorithm here. The scheduling frame duration $T$ is divided into slots (a single slot-duration depends on the packet size, available transmission rate, and is typically application dependent). A slot extends the packet duration by a guard interval to compensate for synchronization errors. At the beginning of this phase, an actuator broadcasts the scheduling packet using maximum transmit power. Since the actuator reaches all the sensors at the same time, the error in synchronization from the delay between time-stamping and sending the packet at the transmitter is eliminated. Since the range of an actuator is on the order of hundreds of meters, the propagation delay is also negligible (few $\mu$sec). Based on the assumption that all the nodes run the same software, all of them will time-stamp the packet at the same time. Therefore, the only error of synchronization in this application comes from clock skew, the difference in the clock tick rates of the nodes. Typical clock drifts of a sensor node in 1sec is $10\mu$sec [6]. If the packet generation period of each node is around 30sec, the maximum clock drift will be 0.3msec compared to approx. 20msec (the duration of the packet transmission of a packet of 50 byte at 50kbps). The total time-frame duration of this schedule is given by $T$, which depends on the number of sensors in the cluster. Therefore, the frame duration $T$ is different for
every cluster in the network. The minimum duration for a sensor to stay awake
$T_a$ is $T_a = T_{rx} + T_{tx} + T_g$, where $T_{rx}$ is the time required to receive a packet,
$T_{tx}$ is the time required to transmit one packet to the parent node, and $T_g$ is the
guard interval for synchronization errors. The interval $T_g$ is assumed to be a small
percentage of the total slot duration. The maximum duration for a sensor to stay
awake depends on its sub-tree and can be calculated as a multiple of $T_a$ depending
on the application, e.g., if the application allows for data aggregation: a sensor can
receive forwarded data from its sub-tree, aggregate its own packet and transmit the
resultant packet requiring only one time slot.

The first transmitted packet to contain the CDMA code is the collision-free
TDMA schedule by each actuator in the network, so that the sensors receive the
schedule from their attached actuator only once. There can be multiple data trans-
missions by the non-radio interfering sensors in the cluster at the same time. The
schedule packet contains a current-time field in order for all the sensors in one
cluster to synchronize to a common clock before starting the transmissions and a
next-time field, where all the sensors wakeup once in order to resynchronize to the
common clock.

4.3 Adjustment Phase

If a new node is added to the network or a link level failure is detected in the net-
work, a sensor will try to attach itself by transmitting a one-hop broadcast request
in its neighborhood. All the sensors in the network wakeup at next-time to resyn-
chronize to the network. At this time, a sensor which receives the actuator-search
broadcast replies to the sensor with its actuator id and cost to reach the actuator.
Upon receiving the reply to its broadcast (there can be multiple replies), a sensor
decides it’s optimal actuator (min. hop-count) and transmits an attachment request
to the actuator. The new sensor is added to the transmission schedule and also
acquires the same CDMA code as it’s cluster.

5 LEAD-Wakeup Protocol

The main idea of LEAD-Wakeup protocol is to extend the scheduling for event-
driven sensing applications, where the slots assigned to the nodes do not have to be
used. According to the opted scheduling scheme, all the nodes on one routing path
remain active only for a small duration $T_a$ to check for the possibility of forwarded
data.

5.1 Adaptivity to Network Conditions

A sensor wakes up at the scheduled time to see if it has any new sensed data in
its transmit queue. If it has no data to transmit and also, no data arrives from its
children sensors during a defined interval (which is equivalent to the reception time
for one packet and a guard interval), it immediately goes back to the sleep mode
and saves considerable amount of energy (adaptive duty cycle). The duration of this adaptive validation period is atleast equivalent to \( T_{\text{adapt}} = T_{\text{rx}} + T_{g} \).

### 5.2 Analysis of LEAD Wakeup protocol

In [8], the authors have shown that the depth-first scheduling works better than the breadth-first scheme for end-to-end delay, throughput and forwarding queue size at the sensors, but fails to perform well in energy consumption compared to the breadth-first scheduling. In this work, we will show that the hybrid scheduling scheme with an adaptive duty cycle achieves a good trade-off between the sensor energy consumption and end-to-end delay for sensor-actuator applications.

**Energy Consumption:** For a wireless sensor, typical states are “active”, “idle”, and “sleep”. The energy saved for a sensor by not sending the sensors directly to active state is shown in Fig. 2. The only form of overhead seen in this power management is the time spent in settling from one state to another and is given by

\[
E_{\text{overhead}} = T_{si} \cdot \left( \frac{P_a - P_s}{2} \right) + T_{ia} \cdot \left( \frac{P_a - P_i}{2} \right)
\]

where \( T_{si} \) and \( T_{ia} \) is the time required to change the state from sleep-to-idle and idle-to-awake, respectively. The gain in energy using such an adaptive power management can be seen as

\[
E_{\text{saved}} = (\delta t - T_{si}) \cdot (P_a - P_i) + T_{si} \cdot \left( P_a - \left( \frac{P_i + P_s}{2} \right) \right) + T_{ia} \cdot \left( \frac{P_a - P_i}{2} \right)
\]

where \( \delta t = T_{\text{event}} - T_1 \), \( T_1 = \) start of the adaptive awake period \( T_{\text{adapt}} \) and \( T_{\text{event}} = \) time of arrival of an event (transmission or reception).
Table 1: Useful states for the sensor node with associated power consumption and delay (time to reach $S_4$ from any given state)

<table>
<thead>
<tr>
<th>Operating State</th>
<th>Strong ARM</th>
<th>Memory</th>
<th>ADC</th>
<th>Radio</th>
<th>Power Consumption</th>
<th>Delay (ms)</th>
<th>Notation Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0$</td>
<td>Sleep</td>
<td>Sleep</td>
<td>Off</td>
<td>Off</td>
<td>50 ($\mu$W)</td>
<td>50</td>
<td>$E_{node}^{s_0}$</td>
</tr>
<tr>
<td>$S_1$</td>
<td>Sleep</td>
<td>Sleep</td>
<td>On</td>
<td>Off</td>
<td>5 (mW)</td>
<td>20</td>
<td>$E_{node}^{s_1}$</td>
</tr>
<tr>
<td>$S_2$</td>
<td>Sleep</td>
<td>Sleep</td>
<td>On</td>
<td>$Rx$</td>
<td>10 (mW)</td>
<td>15</td>
<td>$E_{node}^{s_2}$</td>
</tr>
<tr>
<td>$S_3$</td>
<td>Idle</td>
<td>Sleep</td>
<td>On</td>
<td>$Rx$</td>
<td>100 (mW)</td>
<td>5</td>
<td>$E_{node}^{s_3}$</td>
</tr>
<tr>
<td>$S_4$</td>
<td>Active</td>
<td>Active</td>
<td>On</td>
<td>$Tx$, $Rx$</td>
<td>400 (mW)</td>
<td>NA</td>
<td>$E_{node}^{s_4}$</td>
</tr>
</tbody>
</table>

Let’s see the case with Intel strong ARM (Table 1). It is sensing the environment in $S_1$. At scheduled time, it will change its sleep state from $S_1$ to $S_2$. The sensor node will stay in this state until the arrival of event for $T_{adapt}$, if it does not receive a data packet and it has no packet to transmit it will go back to sleep. Otherwise, it will jump to $S_4$ to transmit a packet. The minimum energy consumed by a sensor during one time frame $T$ is given by

$$Min E_i = E_{i1}^{s_1} \cdot (T - T_{adapt}) + E_{i2}^{s_2} \cdot T_{adapt}$$

Similarly, maximum energy consumed by a sensor during $T$ is given by

$$Max E_i = E_{i1}^{s_1} \cdot (T - T_a) + E_{i2}^{s_2} \cdot \delta t + E_{i4}^{s_4} \cdot (T_a - \delta t)$$

The sensors only wake up when a transmission or reception is expected, therefore, there is no energy drain due to interference from two-hop neighbors. Due to an adaptive sleep schedule, a sensor saves energy by configuring its transmitter state to sleep. Therefore (1) becomes

$$T_{life} = \frac{E_i}{P_{rx} \sum_{j \in N_i} \alpha_{j,i} + P_{tx} \sum_{j \in N_i} \alpha_{i,j} + P_{sense} \lambda_i}$$

(2)

**Observed Latency:** The average latency seen by a packet from node $i$ is

$$delay_i = \sum_K \left( \delta_{s_2-s_4} + T_{data} \right)$$

where $T_{data}$ is the time required to actually transmit a packet and $K$ is the number of hops toward the actuator of node $i$. And the worst case latency seen by a packet from node $i$

$$delay_i = \sum_K \left( \delta_{s_2-s_4} + T_{data} \right) + T$$

which can happen if an event arrival in the current awake duration $T_a$ do not reach the actuator due to long paths.
Table 2: Simulation Parameters. The simulation area is set such that there are at least two sensors in each other’s transmission range.

<table>
<thead>
<tr>
<th>Sensors</th>
<th>Area (m²)</th>
<th>Actuators/Base Stations</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>500 * 500</td>
<td>2</td>
</tr>
<tr>
<td>150</td>
<td>600 * 600</td>
<td>3</td>
</tr>
<tr>
<td>400</td>
<td>970 * 970</td>
<td>8</td>
</tr>
</tbody>
</table>

6 Simulation Results

The metrics which are often used to compare sensor/actuator network MAC-protocols are energy efficiency, delay analysis, and network lifetime. LEAD-MAC aims at performing well in terms of all these metrics. The purpose of this section is to demonstrate the effectiveness of LEAD-MAC. A delay and energy consumption analysis is presented for increasing network size. The simulations environment is ns2 [10], a discrete event simulator. Shortest path routing is used in the simulations. The average depth of the resulting routing trees is 4.4, 5.2, and 7 for 20, 30, and 60 sensors per cluster, respectively; correspondingly the average number of neighbors is 4.6, 5.0, and 5.5. The data packet length is 37 bytes. Sensors generate one packet in 30 sec, with a sampling rate of 1 Hz and transmission rate is 50 kbps. The remaining simulation parameters are listed in Table 2 and Table 3. The sensors in the network are always connected to the actuators. A comparison with the analytical model of PEDAMACS is presented for lifetime analysis. We compare LEAD-MAC only with PEDAMACS, because this work has already been shown to perform better compared to other listed MAC proposals for sensor networks. The evaluated lifetime for LEAD-MAC is given in Fig. 3. The lifetime decreases by increasing the number of sensors in a cluster due to longer data paths and hence more load on the sensors closer to the actuator. The maximum delay observed by a network can be seen in Fig. 4. The end-to-end delay is less due to the depth-first scheduling policy of end-to-end routes in the hybrid-schedule. Finally, we present a comparison for the energy consumption in Fig. 5, where sensors consumes less energy due to an adaptive duty cycle and longer sleep periods.

7 Conclusions and Future work

The introduction of actuators [2] in the network pose a hard delay constraint to timely actuate the required actions. Hence, as the base of the communication stack, the MAC layer should support real-time guarantees or QoS features. The LEAD-MAC protocol is shown to perform better in determining network performance in terms of sensor energy consumption, end-to-end delay observed by the
Table 3: Power Consumption in Discrete Operation States for Mica Motes

<table>
<thead>
<tr>
<th>Operation</th>
<th>Power Consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>To transmit one packet</td>
<td>0.92 mJ</td>
</tr>
<tr>
<td>To receive one packet</td>
<td>0.69 mJ</td>
</tr>
<tr>
<td>Listening to channel</td>
<td>29.71 mJ/sec</td>
</tr>
<tr>
<td>Operating radio in sleep mode</td>
<td>15 μJ/sec</td>
</tr>
<tr>
<td>To sample a packet</td>
<td>1.5 μJ/sample</td>
</tr>
</tbody>
</table>

Figure 3: Average lifetime of a cluster as a function of increasing the number of nodes

Figure 4: Average delay in a cluster as a function of the number of nodes
Figure 5: Average energy consumption in a cluster as a function of the number of nodes

sensor transmissions, and network lifetime as a result of distributed learning during initial deployment and a delay-energy aware hybrid scheduling policy. The use of CDMA codes delimits the interference among neighboring clusters and the need for a separate scheduling algorithm to transmit the schedule.

Our future work will consider the development of LEAD-MAC in a TinyOS based simulator known as TOSSIM. We will develop a routing layer protocol that should operate on top of LEAD-MAC to further optimize network lifetime by considering delay-energy issues at routing layer.

References


