# Distributed Low Power Scheduling in Wireless Sensor Networks

Taehong Kim, Noseong Park[*], Poh Kit Chong, Jongwoo Sung, Daeyoung Kim
Information and Communications University, Electronics and Telecommunications Research Institute[*]
{damiano, chongpohkit, jwsung, kimd}@icu.ac.kr, behack@etri.re.kr[*]

*Abstract*—**Wireless sensor networks are becoming more and more common in the ubiquitous computing era. The applications of sensor networks are diverse and they are employed in areas such as smart homes, military, ITS, fire monitoring, factory automation, and so on. However, sensor networks are generally battery powered and thus, low power consumption is required for prolonged use. In this paper, we propose a distributed low power scheduling algorithm for sensor nodes to determine its active time slots in a TDMA mechanism working on top of a slotted CSMA network. The performance evaluation shows that an extremely small duty cycle is achieved by the proposed algorithm with a tradeoff in data delay in a sensor network application.**

***Index Terms*—low power, scheduling, sensor network**

## I. Introduction

Wireless sensor networks are becoming more and more common in the era of ubiquitous computing. Sensor networks are a kind of ad hoc network that is used for event and environment monitoring. Sensor networks have a wide range of possible usage from military applications to health monitoring to smart homes to forest fire monitoring and many other applications. Due to the requirements of small size and ease of deployment, sensor nodes are battery powered thus, low power consumption is very important as it is difficult or, in some cases, impossible to replace the battery in deployed sensor nodes.

Sensor network applications commonly used in real life such as in smart homes, habitat monitoring, and smart farms, all require a long lifetime as a fundamental characteristic of the sensor network. The main source of energy in a sensor node is a small battery but its main components such as the microcontroller unit (MCU), radio frequency (RF) transceiver, and many various types of sensors consume high power when turned on. For example, a zigbee transceiver that is popularly used nowadays draws about 20mA of current when active, so the lifetime of a zigbee device with 2AA alkaline batteries can not exceed 1 week without wake-up scheduling. Therefore, low power scheduling is an essential element for the longevity of sensor networks.

The low power scheduling algorithm depends on the sensor network application. Sensor network applications can be divided into event driven and continuous monitoring types [5]. In the event driven based application such as fire detection, gas leakage monitoring, and intrusion detections, cycle (referred to as 'epoch') duration should be as small as several hundred milliseconds in order to detect them. Moreover, the synchronization among sensor nodes has to be considered in the scheduling algorithm. In order to report the real-time data to the base station, the other sensor nodes should be waken up in order to relay the event information generated by a sensor node that detects event. On the contrary, the sensor network application for continuous monitoring requires neither small cycle duration nor synchronization between nodes. Since the delay from sensor nodes to the base station is not strictly bounded in the continuous monitoring system, time synchronization between sensor nodes is enough for the wakeup scheduling in the network. That is, not all the nodes need to wake up at the same time for a real-time data transmission. So, a well designed wakeup scheduling algorithm can optimize the needed wakeup time for a given delay bound of an application. The various environment monitoring, remote inspection systems and telematics applications are examples of continuous monitoring applications.

The low power scheduling algorithm can also be classified by the scheduling method. In the centralized scheduling algorithm, the base station collects all the topology information of sensor nodes in a network and calculates the optimal schedule for each node. Even if the scheduling in a centralized method achieves the optimal wakeup schedule, it cannot be applied to large scale sensor networks because of the control packet overhead. Whereas the centralized method is based on the collection of global network topology and distribution of scheduling information for each node, the distributed method adopts the method that each sensor node itself determines its wakeup schedule based on the local topology information. Since the control packet overhead is much smaller than that in the centralized method, the distributed method based scheduling algorithm is preferred in a large network.

In this paper, we focus on the sensor network application for continuous monitoring and try to minimize the wakeup time of each node to prolong the lifetime of the network. The proposed algorithm is a distributed low power scheduling algorithm along with tree topology construction by the nodes. When constructing the tree topology, every node chooses its active time slot by itself. The topology and scheduling is optimized for the application of sensor nodes where every node monitors and reports information to the base station.

This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Advancement)" (IITA-2006-(C1090-0603-0047))

This paper is organized as follows. Section II describes related works on the low power scheduling algorithm. Section III describes the network mode which we use, and section IV shows the proposed algorithm including tree construction and active time slot assignment algorithm. We evaluate the proposed algorithm and conclude in section V and VI.

## II. Related Works

There have been many researches on low power scheduling since research into sensor networks started. Among the many kinds of low power consumption techniques, wakeup and sleep scheduling of nodes has been the area most researched on to extend the lifetime of sensor networks. Shin [1] and Lee [2] suggested beacon scheduling algorithm on IEEE 802.15.4. In their papers, every node transmits a beacon which includes its neighbor's addresses and allocated slots, periodically. When a node attempts to join the network, it receives the beacons from it neighbors and obtains 2-hop neighbor information by interpreting beacons. It selects one of the available slots that are not used in 2-hop neighbors; therefore, every node can achieve time synchronization and low power schedule as well as avoiding beacon collision. However, the beacon only period that is used for wakeup scheduling consumes a big portion of the superframe duration; this causes every node to waste energy during the beacon only period in every cycle. The proposed algorithm also causes a large delay in the sensor network application when all sensor nodes send data to the base station because it requires many cycles to forward data from the lower levels to the base station.

There are also many papers [3-6] that concentrate on the low power scheduling algorithm in the sensor network application for continuous monitoring. The common application that almost papers assume is that all the sensor nodes report one data in a cycle under the tree topology. Choi et al [3] proposed a scheduling mechanism on the sensor network assuming the above sensor network application. They show that the optimal number of slots for both string and tree topology is $3(n\text{-}2)$ slots for $n$ nodes. Under the condition that all the nodes send the data one time in a cycle, they minimized both the needed number of slots in a cycle and active slots. However, the base station computes the scheduling for all nodes, and sends it back. As we mentioned in the introduction, this kind of centralized approach cannot be applied to large scale sensor networks.

The papers [4-6] proposed tree construction and slot allocation algorithm to achieve low power consumption. [4] proposed a way to assign predefined slots to each sensor node assuming both single and multiple base stations. Sichitiu [5] suggested a scheduling algorithm where every node can set its own schedule by using the RTS/CTS based control packet. When the node joins to the network, it sets up the route path and its transmission time to the parent node. When the acknowledgement for RTS/CTS based control packet is returned at a given time, the joining node sets it as its transmission time. However, the number of available slots is restricted and the network setup time is delayed if duration of a cycle is relatively smaller than the required time for the number of nodes in a network.

Our proposed low power scheduling algorithm fully uses the characteristics of tree topology such as hierarchical routing, maximum number of children of a parent node, and maximum depth of a tree. So, we can predefine the number of slots in a cycle that can support all the data transmission of all the sensor nodes in a network. By using predefined active time slots for each level, sensor nodes can easily choose its active node slot as well as avoid collisions with packets from a node in another tree level. Moreover, the slot assignment algorithm is designed to avoid the collision with the packets from a node in a same tree level.

## III. Network Model

The proposed algorithm assumes that every sensor node reports its sensing data once every cycle. The network is constructed by the nodes themselves; the tree topology is designed to make reporting from sensor nodes to the base station efficient. The minimum duration of a cycle, which we use in the proposed algorithm, can be determined by the maximum number of nodes in a network. Because all the nodes know the constraints such as maximum number of children, maximum depth of a tree, they can allocate the memory for storing the schedule information. The schedule information considers the situation that the network has an acceptable maximum number of nodes. For each time schedule, information is already given for sending, receiving, and forwarding from a specific sensor node. Each node can be differentiated by its tree level and its active time slot selected when joining. Thus, those slots may be set as idle in so far as a specific node does not join.

The schedule information for every time slot at the start time is initialized as *idle slot*. When sensor nodes join the network, they choose an active time slot in their tree level. According to a node's tree level and active time slot, they can calculate specific time slot to send the data and update their time schedule information for this time as *send slot*. Then, they notify their new joining to its ancestor node, from the parent to the base station. When the ancestor nodes receive the notification message, they calculate the specific time slot for receiving and forwarding data, and set them as *receive slot* and *forward slot* respectively. After finishing the tree construction phase, every sensor nodes including the base station wake up at the time slots labeled as *send slot*, *receive slot*, or *forward slot* to report periodic information.

In order to achieve the main goal of minimizing power consumption, our proposed algorithm has the following characteristics:

- **Optimal number of active slots**: All the sensor nodes including base station wake up only when they are scheduled to send, receive, or forward the data.

- **Collision free schedule**: No node can transmit data that possibly interferes with a transmission from a certain node to its parent node.

The network scheduling is a form of TDMA working on top of a slotted CSMA network. During the tree construction phase, the control packets used in joining procedure are transmitted as CSMA manner, and data gathering phase works as TDMA because sensor nodes are activated at a start of their active slot.

## IV. Proposed Scheduling Algorithm

The proposed scheduling algorithm is a kind of distributed scheduling algorithm based on a tree topology that fully utilizes the characteristics of the tree topology such as the level of each node, the maximum number of children a parent can have (*maxChild*) and the maximum number of tree level (*maxDepth*). In the proposed algorithm, the basic structure and duration of a network schedule is predefined according to the network constraints *maxChild* and *maxDepth*. For example, a network schedule is divided into *maxDepth* number of periods, and the size of each period is determined by the value of *maxChild*. In order to understand the proposed scheduling algorithm, the basic structure of a network schedule should be understood first.

### *A. Structure of Schedule*

The structure of a network schedule for a string topology in Fig. 1 can help us understand the structure of a schedule for a tree topology. Suppose that the transmission range cannot reach the node in a 2-hop distance, the schedule for a string topology can be calculated like Fig. 2. [3]

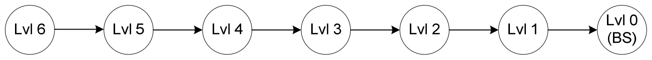
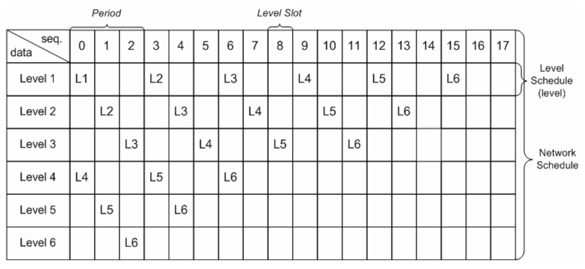


Fig. 1 String Topology

The network schedule shown in Fig.2 is composed of a *level schedule* for every level. These in turn are composed of *level slots*. Since all the data from each level will finally reach the base station at level 0, the nodes at the lower level wake up more frequently. In order to avoid collision, each sending level must be separated by two other levels. In other words, only one level is active every 3 levels; so, it requires 3 level slots for every level to send the data. For example, the data at level 1 and 4 goes up to upper level at the first level slot, data at level 2 and 5 goes up at the second level slot, and data at level 3 and 6 goes up at the third level slot. At the next three level slots, all the data goes up to upper level again. Therefore, the generated data at each level goes up to upper level every 3 level slots. So, we define 3 level slots as a period. For instance, the network in Fig. 2 above requires 6 periods in order to relay the data from the last level 6 to the base station.



Period
Level Slot
Level Schedule (level)
Network Schedule

| data \ seq. | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Level 1 | L1 | | | L2 | | | L3 | | | L4 | | | L5 | | | L6 | | |
| Level 2 | | L2 | | | L3 | | | L4 | | | L5 | | | L6 | | | | |
| Level 3 | | | L3 | | | L4 | | | L5 | | | L6 | | | | | | |
| Level 4 | L4 | | | L5 | | | L6 | | | | | | | | | | | |
| Level 5 | | L5 | | | L6 | | | | | | | | | | | | | |
| Level 6 | | | L6 | | | | | | | | | | | | | | | |

Fig. 2 Network Schedule

In the *network schedule* in Fig 2, sensor nodes wake up to send and forward the data at the marked level slots. The number of active level slots in a level schedule for each level can be computed by *3k + (level-1) mod 3*, where *k* is between *0* and *maxDepth-level*. Nodes sleep during their inactive level slot times.

In order to extend the network schedule for a string topology to a tree topology, the number of node slots in a level slot (*NumNodeSlots*) and data size at each period should be considered. Because there are more nodes in a level compared to a string tolopology, for *NumNodeSlots*, the minimum requirement is the number of maximum children that a parent can have *(maxChild)*. This is because only one child can send data to its parent at a time. If more slots than the maximum number of same level neighbors are given, all the neighboring nodes in the same level have different node slots. As shown in Fig. 3, the last joined node selects a node slot in order not to overlap with other node's node slot in the same level if *NumNodeSlots* is enough.

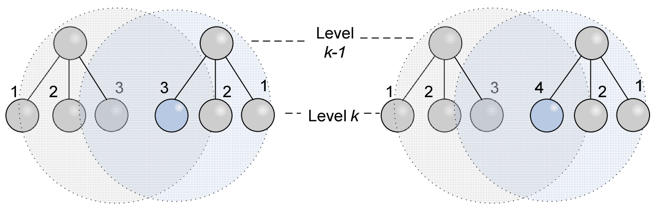


Fig. 3 (a) Node Slots at NumNodeSlots 3 (b) Node Slots at NumNodeSlots 4

In the tree topology, the amount of transmitted data increases as the data arrives from higher levels. For instance, the node at level 1 sends the data amount of 1 in the first period, because it transmits only its own data. In the second period, the node will forward the data which was received from its children in the first period. The size of the data in the second period is a maximum of *maxChild*. In this manner, the data size at each period can be computed as $(maxChild)^{period-1}$.

We can compute the level schedule for every level in the general tree topology by considering both *NumNodeSlots* and data size at each period, as follows:

$$\boxed{\begin{array}{l}\sum_{t=1}^{k}(3\cdot maxChild^{t}\cdot NumSlots)+maxChild^{k}\cdot NumSlots\cdot((level-1)\%3)\\ (0\le period\ \ k\le \max Depth-level\,)\end{array}}$$

Fig. 4 Equation of Level Schedule for a Tree Topology

Every node chooses one active node slot among available node slots (1 ≤ active node slot ≤ *NumNodeSlots*) when joining the network. The detailed algorithm to construct the tree topology and to choose the active node slot will be explained in the next section IV. (B). Based on the level schedule equation in Fig. 4, they can compute their own node schedule by adding *(active node slot-1)·the data size at each period* into the level schedule. The parent node can calculate its children's active time in a similar manner, and set related slots as active slot in order to receive and forward the data. The proposed scheduling algorithm in a view point of a node is described in the Fig. 5.

## B. Tree Construction and Slot Assignment Algorithm

In this algorithm, every node sets the schedule information for each time slot. The schedule information that is used in the proposed algorithm is as follows.

- **idle slot**: The node keeps the idle mode to save the power consumption.

- **send slot**: The node generates its own data to report to a base station.

- **receive slot**: The node receives data from its child or descendant node.

- **forward slot**: The node forwards data stored in the receive buffer.

Because all the sensor nodes and base station keep the information of network configuration such as *maxDepth*, *maxChild*, and *NumSlots* before it is started, they allocate the memory for the schedule and set the schedule information for all the time slots as *idle slot* when they are turned on.

The tree topology is constructed from the broadcast of the base station. The base station broadcasts the 'construction message' to build the tree topology. In the 'construction message', the node's tree level and its active node slot information is included.

On receiving the 'construction message', the node waits a random period in order to find the potential parent node which has shortest-hop to the base station and to overhear the tree level and active node slots of neighbors. After the period is over, the node chooses its parent, determines the tree level, collects unused node slot between 1 to *NumNodeSlots* among its neighbors in a same level, and sends a 'join request message' to the parent node. The parent node, on receiving the 'join request message', sends a 'join response message' after assigning the active node slot among available slots in both parent node and child node. The reason the parent node assigns the child's active node slot is to prevent a hidden terminal problem. Although the child node tries to avoid the used slots by overhearing the 'construction message' from its neighbors, children which can not communicate directly may choose the same active node slot. If the parent node has already *maxChild* number of children, it denies the new joining and recommends finding another parent node.

When the joining node receive the 'join response message' from the parent node, it sets its active node slot, calculates its active time based on level schedule and its active node slot, and update the schedule for that time to *sent slot*. Then, it sends 'notify joining message' to its ancestor nodes from its parent to the root node. The ancestor nodes finishing the joining procedure transmit the 'construct message' to allow joining of child nodes.

When the ancestor nodes receive 'notify joining message', it updates the schedule for the time slots prepared to receive or forward data of descendant nodes, to *receive slot* or *forward slot*. The reason we use the feedback mechanism like 'notify joining message' is to reduce the number of active slots. When the nodes prepare the slots to receive and forward data, it can be wasted if the slots are not utilized. With this mechanism, the needed active time slot for any network topology can be said to be the minimum.

***Distributed Low Power Scheduling Algorithm***

```
Begin
 1. receive the 'construction message'
 2. while (random period is not expired)
 3.   record tree level, active node slot from rebroadcast of neighbor
 4. end while
 5. while (period is expired and parent node denied)
 6.   send join request message after choosing the parent node and
      its depth, and collecting unused slots among its neighbors in
      same depth
 7. end while
 8. record its active node slot the parent node assigned
 9. schedule to send data based on level schedule and its active
    node slot, and notify its ancestor node
10. ancestor nodes updates the schedule to receive and forward the
    data from descendant node
11. transmit 'construction message' to accept its child node
12. if (data gathering cycle is started)
13.   send and forward the data in its active time every cycle
14. end if
End
```

Fig. 5 Tree Construction and Slot Assignment Algorithm

## V. Performance Evaluation

We evaluate the performance of the proposed algorithm based on the duty cycle at data gathering period and average delay from sensor node to base station. For the simulation, we used the NS-2 simulation tool. The networks agents we used in the simulation are IEEE 802.15.4 and ZigBee. Since the ZigBee network protocol has many commonalities with the proposed algorithm such as limiting maximum children (*Cm*) and maximum tree depth (*Lm*), following hierarchical routing, we selected IEEE 802.15.4 and implemented ZigBee's tree routing for multi-hop communication from sensor nodes to the base station. In the simulation environment, we set the network size as 100mx100m and transmission range as 10 meters. Every node has identical transmission range and they are randomly deployed. The packet size generated by every sensor node is 100bytes, and the duration of one slot is 20msec. We set the network configuration *maxChild* and *maxDepth* as 4 and 5 respectively for all simulation.

For the tree construction procedure in the proposed algorithm, we modified frame structure of several commands packet such as beacon, association request, association response. The beacon, association request, association response packet are matched with 'construction message', 'join request message', join response message' in the proposed algorithm respectively. So, in the beacon payload, the node's tree level and its active slot is added to let neighbor nodes know. Association request packet and response packet is added as the available slot list and assigned node slot respectively. Once the coordinator that has the role of base station starts to construct the network, other sensor nodes discover the neighbor information such as their tree level and active time slots by using the scan procedure. After selecting the potential parent node, it sends association request with the available slot list. The parent node checks its available slots for children and the lists the child node that wants to be assigned, and sends an association response with a matched slot.

For the 'notify joining' in the tree construction phase and data reporting in data gathering phase, the routing follows the ZigBee's tree routing. All the sensor nodes set the destination as 0 to send the packet to the base station; then, sensor nodes and its ancestor nodes forward it to their parent node according to destination address.

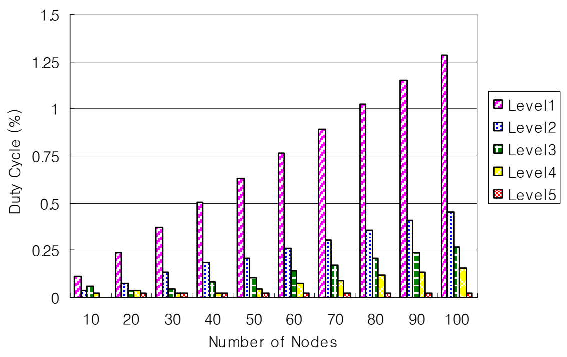


Fig. 6 Average Duty Cycle in Each Level

Fig. 6 shows the average duty cycle in each tree level. The network configuration *NumNodeSlots*=5, and the number of slots in a cycle is calculated as 5115. As the number of nodes in a network increase, the duty cycle in each level increase together. It is because the active slot is allocated only when the nodes joins. Since the nodes in a lower level have to relay the data of higher levels, average duty cycle in lower level is higher than that in higher level. Even if the duty cycle increases as the number of nodes in a network increases, the duty cycle is still less than 1.5% for 100 nodes.

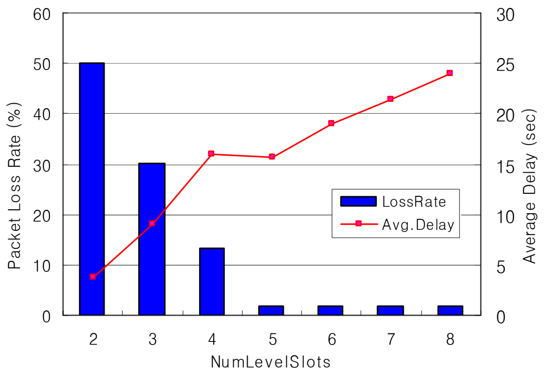


Fig. 7 Packet Loss Rate versus Average Delay

Fig. 7 shows the relation between *NumNodeSlots* and packet loss rate and tradeoff between *NumNodeSlots* and average delay. The number of nodes in a network is 60 nodes. As we mentioned in Section VI, if *NumNodeSlots* is not enough to assign unique active node slot to every node in a tree level, they may select the same active node slot and send the data at the same time. Therefore, the packet loss rate is higher as the *NumNodeSlots* is smaller. However, when *NumNodeSlots* is higher than 5, the packet loss rate becomes stable with 1%. The average delay is measured as the average delay from sensor node to the base station. Since *NumNodeSlots* affects both duration of a cycle and the size of each period, the average delay increases as *NumNodeSlots* increases. The cycle duration for *NumNodeSlots* 5 and 8 is about 100 sec and 180 sec respectively. Even though such a big cycle may be an issue in event driven sensor network applications, it is not a problem in sensor network applications for monitoring.

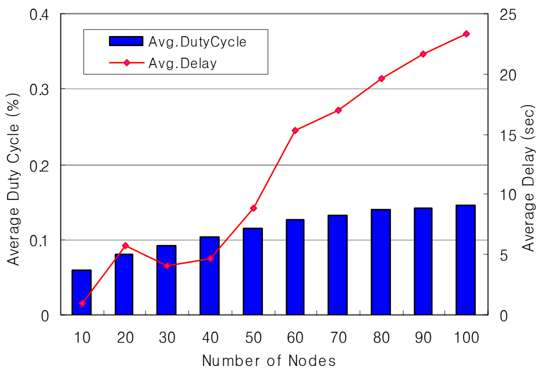


Fig. 8 Average Duty Cycle and Average Delay

Fig. 8 shows the average duty cycle and average delay according to the number of nodes in a network. The network configuration *NumNodeSlots* is set to 5 to simulate the condition that all the data from the sensor nodes comes to the base station. The reason the average duty cycle increases is because the number of nodes in a network increases. However, the average duty cycle is still significantly smaller relative to the increase in the number of nodes in a network, whereas the average delay increases to 23 sec when the number of nodes is 100. The average delay has a tendency to increase as the number of nodes increases. Since the structure of the schedule is already defined, and data from higher levels are delivered in higher periods as in Fig. 2, the average delay increases as the number of nodes in higher levels increases. However, this is allowed for sensor network applications for continuous monitoring because the delivery time is not a critical fact in this type of applications.

## VI. Conclusion

We proposed a distributed low power scheduling algorithm based on the tree topology. Every node can calculate its active time schedule based on the level schedule and its active node slot. The proposed algorithm makes the duty cycle of the nodes optimal by allocating active node slots for only necessary data transmission. It is achieved by dynamically allocating active node slots for the notification of joined nodes and not by allocating the active node slot in advance.

If the *NumNodeSlots* is equivalent to the maximum number of neighbors in the same level, we can guarantee reliable data gathering. Otherwise, in dense networks where there are many interfering nodes, we can expect that much of the sensing data will be duplicated and therefore will be filtered out using CSMA in the node slots in the network. Thus, the user can choose proper *NumNodeSlots* to get reliable data gathering with consideration to the network topology.

If the sensor network application is tolerant to the big duty cycle and large delay, then the network lifetime can be maximized by the proposed low power scheduling algorithm.

## REFERENCES

[1] Yongsik Shin, “MAC/PHY Specifications on MEW (Mewsh-Enabled Wireless Sensor Network) Technology,” proposal of ISO/IEC JTC1

[2] Myung Lee, Huai-Rong Shao, Ho-in Jeon, “Combined Beacon Scheduling Proposal to IEEE 802.15.4b,” in IEEE 802.15.4b proposal

[3] Hongsik Choi, Ju Wang and Esther A. Hughes, “Scheduling on Sensor Hybrid Network,” in IEEE ICCCN 2005

[4] Andreea Berfield and Daniel Mossé, “Efficient Scheduling for Sensor Networks,” The 1st Internation Worshop on Advances in Sensor Netowkrs 2006.

[5] Mihail L. Sichitiu, “ Cross-Layer Scheduling for Power. Efficiency in Wireless Sensor Networks,” INFOCOM,. 2004.

[6] B. Hohlt, L. Doherty, and E. Brewer. "Flexible Power Scheduling for Sensor Networks," IPSN 2004.

[7] Diba Mirza, Maryam Owrang, Curt Schurgers, "Energy-efficient Wakeup Scheduling for Maximizing Lifetime of IEEE 802.15.4 Networks," International Conference on Wireless Internet (WICON’05), Budapest, Hungary, pp. 130 - 137, July 2005

[8] Q. Cao, T. Abdelzaher, T. He and J. Stankovic, “Towards Optimal. Sleep Scheduling in Sensor Network for Rare-Event Detection,” The. 4th International Symposium on Information Processing in Sensor Networks, 2005.

[9] D. Mirza, M. Owrang, C. Schurgers, "Energy-efficient. Wakeup Scheduling for Maximizing Lifetime of IEEE. 802.15.4 Networks", Proc. International Conference on. Wireless Internet (WICON’05), Budapest (Hungary), pp.130-137, July 2005.

[10] Pollin S, et al., “Performance analysis of slotted IEEE 802.15.4 medium access layer,” Technical Report, DAWN Project, Sep. 2005.