Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 19 (2013) 348 – 355

Procedia Computer Science

The 4th International Conference on Ambient Systems, Networks and Technologies

(ANT 2013)

Implementation of Rank Based Sleep Scheduling (RBSS) Protocol for WSNs in a Fixed Grid Topology

Tarek Sheltami [a], Abdul Jabbar Siddiqui, Hamza Ijaz Abbasi, Uthman Baroudi, and Lahouari Ghouti

Department of Computer Engineering, King Fahd University of Petroleum and Minerals,Dhahran 31261, Saudi Arabia

Abstract

One of the most significant challenges in wide-scale Wireless Sensor Networks (WSNs) is to achieve energy efficiency in order to increase their lifetime and ensuring responsiveness. A very widely applied approach in this regard is to overload the region with numerous low-cost sensing devices that can communicate with each other wirelessly and coordinate amongst themselves in developing schedules for being asleep or active. The literature has numerous research works in this regard which aim to develop efficient sleep scheduling schemes. In this paper, we present an implementation of a grid-topology based cooperative Rank Based Sleep Scheduling (RBSS) protocol in which the nodes of every cell coordinate in a distributive manner, the decision making process of selecting the nodes that should stay active, while others sleep. The proposed protocol guarantees the WSN connectivity as well as the required coverage. Finally, we implement RBSS on a small scale grid using TelosB motes.

Selection and peer-review under responsibility of Elhadi M. Shakshuki

Keywords: Wireless Sensor Networks, Grid Topology, GAF Protocol, Sleep Scheduling, SPAN Protocol, Crossbow TelosB motes, TinyOS, nesC

1. Introduction

In most wireless sensor networks, sensing devices are ubiquitously deployed in a desired region and configured to establish networking and communication amongst each other and report or relay all the

[a] Corresponding author. Tel.: +966-3-860-4678; fax: +966-3-860-3059.
E-mail address: tarek@kfupm.edu.sa

Selection and peer-review under responsibility of Elhadi M. Shakshuki
doi:10.1016/j.procs.2013.06.048

sensed data to a base station. Such sensors are extremely energy constrained as they rely mostly on batteries or other alternative sources of power (such as energy harvesting etc.) but replacing or repairing these energy-sources is expensive and not practical in large scale WSNs, in many of which, the exact locations of the sensors may not be known after deployment. However, since wireless sensor networks are desired to run for a considerably long period of time (months or years), these factors inspired researchers to develop energy efficiency protocols and ideas to be incorporated in the sensors themselves, thereby improving the robustness and energy consumption and hence the overall lifetime of the WSN.

Several experimental studies, such as those in [1], [2], [3], etc., have showed that a lot of energy is wasted during the idle listening state of sensor nodes and that the consumption of power by the sensor nodes is very similar in both the idle listening (non-transmitting or non-receiving) and the receiving/transmitting states. Considering most WSN applications, where continuous data delivery and reporting is not essential, the nodes are mostly not involved in sending/receiving the desired sensed data. Hence, there have been many research attempts to develop efficient ways to turn off the node radios when not required to be part of the network or gathering data, but at the same time ensuring there are at least the minimum number of active nodes in their area to take up the duty of ensuring networking and communicating (eg: NAC [7][9], S-MAC [10], T-MAC[11], GAF and CEC [12], SPAN [13], etc. which are based on the periodic data gathering paradigm [8]).

Such proposed and developed techniques can be broadly classified into three categories: On-demand wake-up, scheduled wake-up, and asynchronous wakeup [4]. Firstly, the on-demand wake-up mechanisms require the nodes to have an additional low-power radio besides the primary radio, and the former would wake the latter up when required to. Secondly, the scheduled wake-up mechanisms involve nodes which have synchronized clocks and wake-up periodically after synchronized time intervals. In this approach, the wireless nodes turn off their radios while sleeping at pre-determined intervals to save energy, a technique labelled as "**duty cycling**" or "**sleep scheduling**". In overloaded WSNs, several nodes can communicate and coordinate with each other in framing their sleeping-waking schedules [5]. Thirdly, asynchronous wake-up mechanisms involve nodes which do not require synchronized clocks and wake up at pre-defined time intervals to ensure overlapping with active-state intervals of other nodes in the vicinity [6]. These are highly appropriate for cases where it is impractical to achieve global synchronization of time throughout the WSN.

In this paper, we propose a Rank Based Sleep Scheduling Protocol for a wireless sensor network setup in a fixed grid topology. These sensor nodes coordinate among themselves in deciding which one(s) must stay active, and which one(s) must sleep, based on their ranks, while achieving the objective of having at-least one node active per cell at any given time. "Rank" is an abstract term that can be interpreted differently according to the ongoing application. For instance, "available energy" in each sensor node can be used to rank each sensor in each grid. The following sections describe the implemented protocol in greater detail.

The rest of the paper is organized as follows. Section 2 describes the system description. The proposed protocol is presented and discussed in section 3. In section 4, we describe the experimental setup used for implementing RBSS followed by section 5 where we show few snapshots of our experimental results. Finally, the paper is concluded in section 6.

2. System Description

We assume that the monitored area is divided into grids. An example scenario could be a huge parking facility where one square/rectangular grid-cell is the area enclosed between four columns (each at the corner of the square/rectangular area). The division of the area of interest into a grid-layout is not the

focus of this paper, and it depends on the requirements of the application and kinds of devices used. In each grid cell, a number of sensors are positioned to cover the grid as shown in Figure 1. Each sensor node within a grid cell would intercommunicate with its partner nodes in the same cell in order to decide that which nodes should be allowed to sleep (to conserve energy) and which node needs to be active in the cell using Rank Based Sleep Scheduling (RBSS) protocol. For experimental analysis and without loss of generality, we consider only 2 sensor nodes per grid cell of which only 1 node is allowed to be active at a time within the cell. The active node in each cell would then communicate with the other active nodes in its adjacent neighbouring cells to carry out routing, data delivery, and other kinds of network communication purposes as and when needed. In the following figure, an example of a grid-based wireless sensor network has been illustrated:

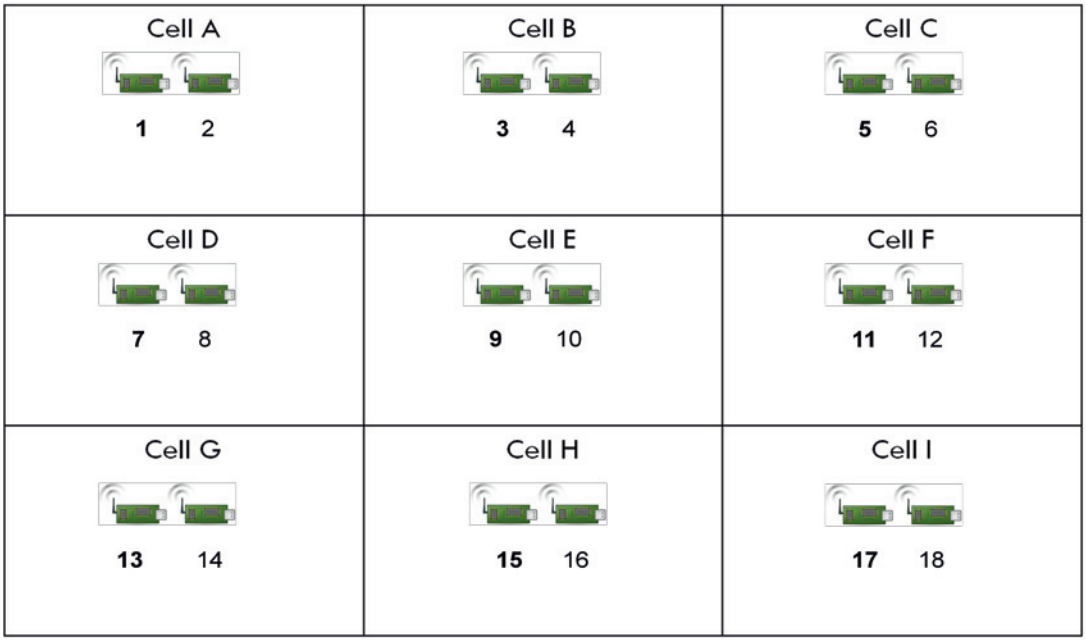

Figure 1. Example of Grid Topology

In Figure 1, let us consider Cell E. The two sensor nodes 9 and 10 in cell E would intercommunicate with each other using the RBSS Protocol to determine which node will stay active and which one would go to sleep for energy conservation purposes. The active node in cell E would then communicate with the active nodes in its adjacent neighbouring cells i.e. Cell B,D,F and H to carry out routing, data delivery and for other networking purposes.

3. Rank Based Sleep Scheduling (RBSS) Protocol

In order to decide that which sensor node within a grid cell would remain active and which would go to sleep, we are proposing Rank Based Sleep Scheduling (RBSS) Protocol. RBSS controls active/sleep schedule based on the current rank of each sensor node. “Rank” is an abstract that can be interpreted according to the ongoing application. For example, residual energy in each sensor can be used to rank the sensor. Since the rank associated with each sensor is dynamic in nature, it is essential to update and communicate these changes to all sensors positioned in the same grid cell. Therefore, RBSS

defines four distinct states or modes: *Time Synchronization, Discovery, Sleep and Active*. The following sub-section explains these modes.

3.1. Description of the Protocol Sates

Figure 2 illustrates the transitions from state to state per specific conditions. Initially, any sensor node starts off in the time synchronization state. Then it moves on to the discovery state after receiving the time beacon from the base station. From the discovery state, the node either proceeds to the sleeping state or the active state depending upon the rank value that it has received from its cell partner node and the timer value T_d. Choosing the value for the Discovery-State-Timer (T_d) depends on the users' requirements and must be sufficient enough to ensure that every node within the grid-cell have received info from all other nodes within that cell (i.e. info on rank statuses). From the active state, the node either goes to discovery state or sleep state depending on either the discovery message that it receives from its partner node or the local timer T_a. From the sleep state, the node will proceed to the discovery state once the local timer T_s expires. The values of Active-State-Timer, (T_a) and Sleep-State-Timer (T_s) are determined by how long the user wants a node to stay active or asleep, respectively. They should be chosen so as to maximise overall energy savings. In this way, the node goes from one state to other, while ensuring that there should always be an active node in each grid cell.

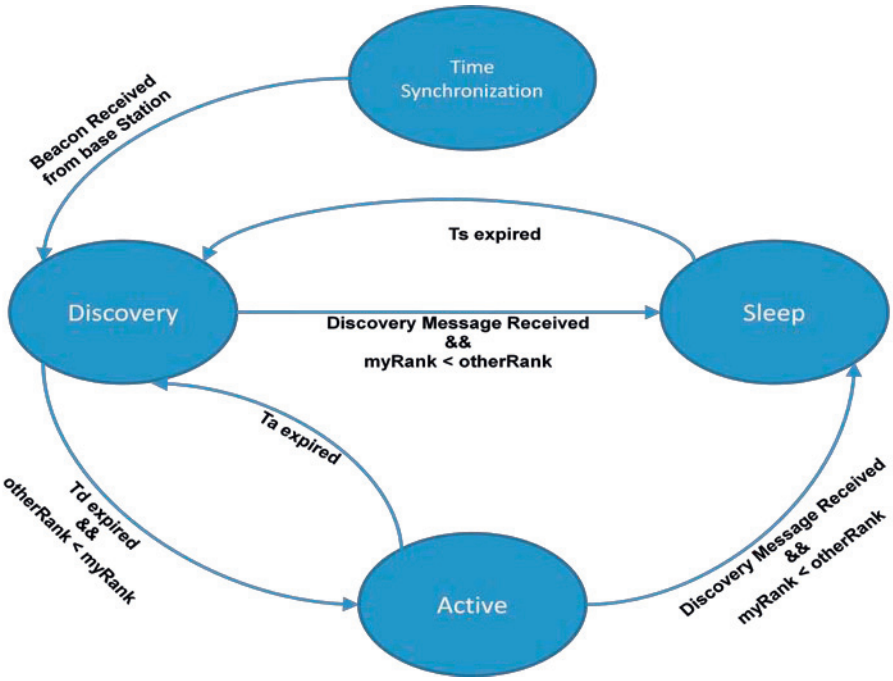

Figure 2. State Diagram of RBSS Protocol.

- *Time Synchronization Mode:* Initially, after the sensor node starts it keeps on waiting until it receives the beacon message from the base station containing time stamp. The sensor node then sets its local time according to the time received from the base station. Each node is assigned a predefined rank at start-up, which is hardcoded initially.

- *Discovery Mode:* After setting its time using the time beacon received, the sensor node goes into discovery mode. During the discovery mode, the sensor starts the timer T_d. Apart from this, during the discovery mode; the node also transmits its discovery message to its neighbouring node within the same cell. Within the discovery message, the node transmits its rank along with its local time. Upon receiving the discovery message from its partner node, the current node sets its local time as that of the partner node in order to keep time synchronization between the two nodes. We assume that the two sensor nodes are relatively close to each other and that the time of propagation of the messages (between them and from the base station to either of them) is negligible. Therefore, they achieve time synchronization (as described above) by sending each other their local times. Apart from this, the node also decides about its future mode (active or sleep) after it receives the discovery message based on its own rank value and its cell partner's rank value. If the node's own rank value is greater than the rank value of its partner node (whose discovery message has been received), it would wait for T_d to be expired which was started in the beginning. Then, the node would go to active mode. On the other hand, if the current node's own rank value was less than the rank value of the other partner node, the node would proceed to sleep mode. However, if no discovery message has been received during the entire T_d period, the node will automatically go to the active mode after T_d expires. Therefore, if one node in the grid cell malfunctions or goes down, its other partner node can take the entire role on itself and stay active the whole time. This makes our protocol very reliable.

- *Active Mode:* After a node goes into active mode, it starts the timer T_a. During this mode, the sensor node is responsible for routing, data delivery and carrying out networking communication with its neighbouring cells' active sensor nodes. If a sensor node during its active mode, receives a discovery message from its cell partner node which has a higher rank than its own rank, then this active node would directly proceed to sleep mode giving the other node the chance to go to active mode. However, if the node doesn't receive any discovery message from a higher ranked node, it waits until the T_a timer to expire. Once T_a finishes, the node goes to the discovery mode.

- *Sleep Mode:* When the node enters sleep mode, it starts the timer T_s. The aim of entering in sleep mode is to conserve the energy of the sensor node so that it can be better utilized later on. Therefore, the node should turns its radio off without sending or receiving any kinds of messages or doing any kind of processing during the sleep mode. After the timer T_s expires, the node wakes up and proceeds to the discovery mode.

In this manner, the node goes from one mode to another, while ensuring that there should always be at least one active node in each grid cell monitoring its activity besides acting as a router for the other grids.

Figure 3 depicts the flow chart for the four different modes of RBSS Protocol (*Time Synchronization, Discovery, Sleep and Active*) along with the different conditions under which a sensor node transits from one mode to another:

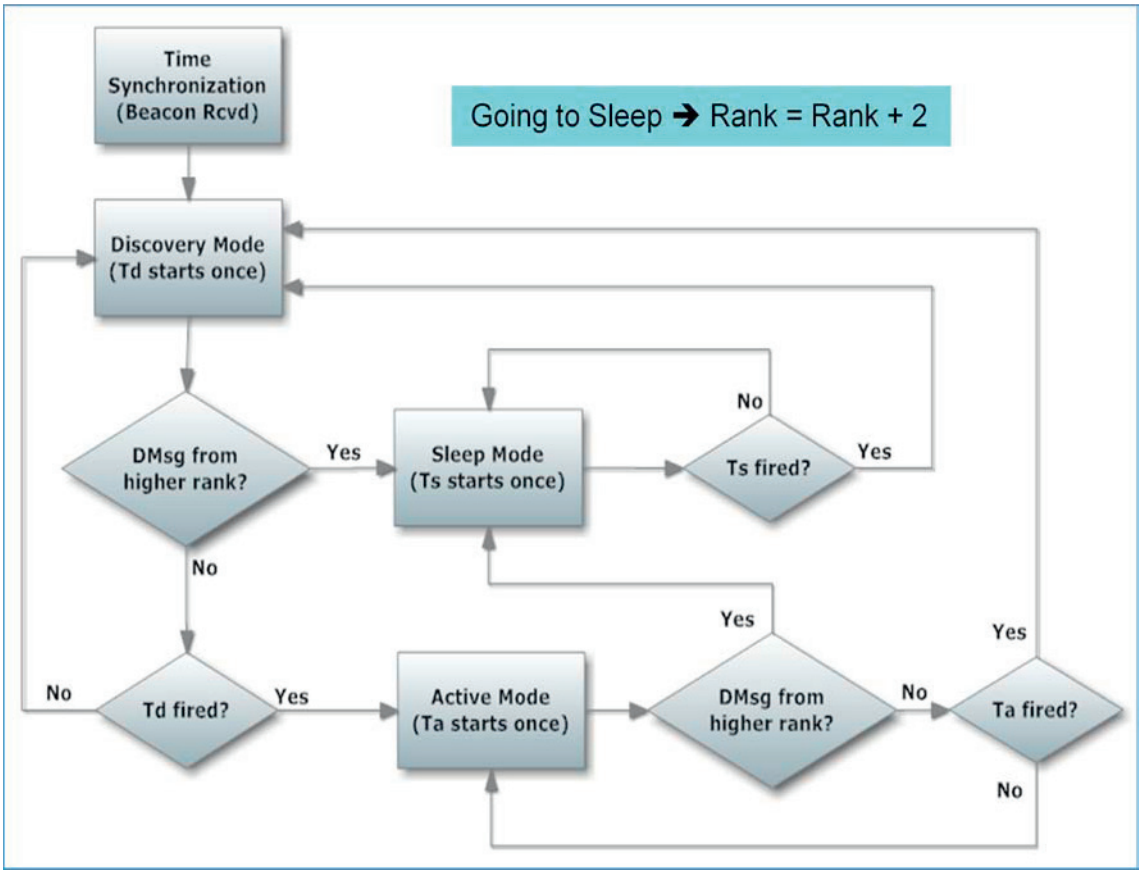

Fig. 2. Flow Chart of RBSS Protocol.

4. Experimental Setup

In this section, we describe the experiment setup along with the hardware and software that has been used for the implementation of RBSS Protocol in WSNs.

4.1. Hardware Components (Crossbow TelosB motes)

The sensor nodes we have used are the Crossbow's TelosB motes which are widely used by the WSN community. They use the IEEE 802.15.4/ZigBee compliant RF transceiver, a globally compatible ISM band of 2.4-2.4835 GHz, provides a data rate of 250 kbps, had an onboard integrated antenna, 1 MB external flash for logging data, an 8 MHz TI MSP430 microcontroller with 10kB of RAM, along with optional sensor suite such as integrated light, temperature and humidity sensor (TRP2420). These motes run on TinyOS 1.1.10 or higher and can be programmed via USB [14].

4.2. Software Platform (TinyOS)

The software platform that we used to implement the RBSS protocol in WSNs is TinyOS. TinyOS is a small, open-source, energy-efficient software operating system developed by UC Berkeley which supports and is designed for large scale, self configuring designed for wireless embedded sensor

networks. The source code software development tools are publicly available on [15]. It features a component based architecture (event-driven) which enables rapid innovation and implementation while minimizing code size as required by the severe memory constraints inherent in sensor networks. TinyOS's component library includes network protocols, distributed services, sensor drivers, and data acquisition tools – all of which can be used as-is or be further refined for a custom application. It tries to minimize code size [16], [17]. For comprehensive guide and explanations on programming in TinyOS, [19] is an excellent resource.

4.3. Programming Language (nesC)

The programming language that was used is nesC. The "nesC" which stands for "Networked Embedded System C" is an extension to C language designed and modified to handle the special data structures and execution model of TinyOS. TinyOS has been re-implemented in nesC. The manual in [18] describes v1.1 of nesC. The basic concepts behind nesC are: (1) Separation of construction and composition, (2) Component Behaviour Specification in terms of set of Interfaces, (3) Bi-directional Interfaces, (4) Static Linking of Components, (5) Whole-program Compilers, (6) The Concurrency Model of nesC [18].

5. Experiments and Tests

We have experimented and tested the implementation of the algorithm on TelosB motes, with a pair of TelosB motes, representing one grid-cell and observe the behaviour of our protocol on these nodes. For experimental purposes, in which rank determination of motes is not practically feasible to be based on remaining-energy in batteries, etc., we initialise one of the two motes of a grid-cell, say Node_1 with an initial rank of '11' (odd) whereas its partner node in the same grid-cell, say Node_2, is given an initial rank of '2' (even).

Initially, the nodes 1 and 2 are switched ON one after the other to have a difference in their times, as shown in Fig. 4. Once started or reset, the nodes wait for a beacon from the base station to receive the global time. When the base station (BS) sends the beacon to these nodes, they reset their timers to the timestamp included in the beacon message. The BS could be a single stationary node (which would require all the nodes to be in its range for the time-synchronization to work), or could be a node that moves across the region sending out beacons every now and then, thereby ensuring that all nodes receive the BS time. After achieving time synchronization through the BS beacon, both nodes enter DISCOVERY mode to exchange their ranks and states. If a node is at a higher rank, it turns ACTIVE while the node at lower rank enters SLEEP mode (see section 3).

The nodes remain in their respective states for specific times (for T_a in ACTIVE mode, for T_s in SLEEP mode) and then enter the discovery mode again. In this manner, the whole cycle of events is repeated, with only one of the two nodes going ACTIVE at a time.

6. Future Work

The experience with implementing protocols proposed in literature on real motes taught us that many of the assumptions and requirements of these protocols are practically not feasible. Future works could involve fine-tuning the most significant of those protocols for practical deployment. In our case of sleep-scheduling protocols, we plan to deploy the proposed protocol and several others from literature, on a wider-scale testbed, possibly with different families of sensor motes. Such experiments could give deeper insight on how these protocols would behave in real-life scenarios and in varying environmental

conditions. The variations in throughput, connectivity, sensor lifetime, etc. for the different protocols could be measured more meaningfully if deployed on large-scale WSNs.

7. Conclusion

In this paper, Rank Based Sleep Scheduling (RBSS) Protocol was proposed for a wireless sensor network (WSN) setup in a fixed grid topology. The basic aim of the RBSS Protocol is to enable the sensor nodes to coordinate in a distributed manner the activity (active, sleep) of each node in a gird. The decision is based on the current rank of each node, while achieving the objective of having at-least one node active per cell at any given time. RBSS Protocol attempts to solve one of the most significant challenges in wide-scale WSNs i.e. to achieve energy efficiency in order to increase the lifetime of the sensor nodes and ensuring responsiveness.

Acknowledgements

The authors would like to acknowledge the support of King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.

References

[1] L.M. Feeney and M. Nilsson, “Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment,” Proc. IEEE INFOCOM, Mar./Apr. 2001.
[2] M. Stemm and R.H. Katz, “Measuring and Reducing Energy Consumption of Network Interfaces in Hand-Held Devices,” IEICE Trans. Comm., vol. E80-B, no. 8, pp. 1125-1131, 1997.
[3] A. El-Hoiydi, J. D. Decotignie, C. Enz and E. Le Roux, “WiseMAC: an Ultra Low Power MAC Protocol for the WiseNET Wireless Sensor Network”, Poster, in ACM Sensys 2003.
[4]Bong Jun Choi; Xuemin Shen; , "Adaptive Asynchronous Sleep Scheduling Protocols for Delay Tolerant Networks," Mobile Computing, IEEE Transactions on , vol.10, no.9, pp.1283-1296, Sept. 2011
[5] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, “SPAN: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks,” Wireless Networks, vol. 8, no. 5, pp. 481-494, 2002.
[6] R. Zheng, J.C. Hou, and L. Sha, “Asynchronous Wakeup for Ad Hoc Networks,” Proc. ACM MobiHoc, June 2003.
[7] Sheltami, T.R.; Shakshuki, E.; , "Neighbor-Aware Clusterhead with Different Sleep Scheduling Protocols," Parallel Processing - Workshops, 2008. ICPP-W '08. International Conference on , vol., no., pp.143-147, 8-12 Sept. 2008
[8] Begum, S.; Shao-Cheng Wang; Krishnamachari, B.; Helmy, A.; , "ELECTION: energy-efficient and low-latency scheduling technique for wireless sensor networks," Local Computer Networks, 2004. 29th Annual IEEE International Conference on , vol., no., pp. 60- 67, 16-18 Nov. 2004
[9] Tarek R. Sheltami “Neighbor-Aware Clusterhead for SNET,”Advanced Information Networking and Applications - Workshops, 2008. AINAW 2008. 22nd International Conference on 25-28 March 2008 Page(s):561 – 566
[10] W. Ye, J. Heidemann, D. Estrin, “An Energy-Efficient MAC Protocol for Wireless Sensor Networks", INFOCOM 2002, New York, NY.
[11] T. Dam, K. Langendoen, “An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks", SenSys 2003, Los Angeles, California.
[12] Y. Xu, S. Bien, Y. Mori, J. Heidemann, and D. Estrin, “Topology Control Protocols to Conserve Energy in Wireless Ad Hoc Networks", CENS Tech Report 0006, 2003.
[13] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, “Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad HoC Wireless Networks", ACM Wireless Networks Journal, Volume 8, Number 5, September 2002.
[14] TelosB_Datasheet.pdf (Downloaded from http://www.willow.co.uk/TelosB_Datasheet.pdf, on 14th January 2011, 11:54 AM KSA)
[15] http://www.tinyos.net
[16] http://docs.tinyos.net/tinywiki/index.php/Main_Page
[17] Greg Hackmann, “TinyOS Tutorial”, CSE 521S Fall 2010, Washington University in St.Louis.
[18] David Gay, Philip Levis, David Culler, Eric Brewer, “nesC 1.1 Language Reference Manual”, May 2003.
[19] Philip Levis, “TinyOS Programming”, June 28, 2006.