Volume 1, No. 1, May-June 2010

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

EETS: AN ENERGY-EFFICIENT TIME SYNCHRONIZATION ALGORITHM FOR WIRELESS SENSOR NETWORKS

Kavi Kumar Khedo*
Department of Computer Science and Engineering
University of Mauritius
Reduit, Mauritius
k.khedo@uom.ac.mu

Dhiraj Lobin
Department of Computer Science and Engineering
University of Mauritius
Reduit, Mauritius
dhirajlobin@gmail.com

Abstract: High precision and real-time communications are increasingly becoming important in Wireless Sensor Networks as such networks are required to support highly time-critical applications. The sensor nodes need to communicate data to each other at the correct time. However, the clocks of sensor nodes run at different speed and very often drift from the real time value. In this paper we are proposing EETS, an energy efficient time synchronization algorithm. Energy efficiency is achieved by synchronizing only nodes that are communicating with each other when an event occurs. EETS is performed in two phases namely the Level discovery phase and the Synchronization phase. A simulation study has been carried out and the results shows that EETS is more energy efficient than other existing algorithms. Particularly, EETS has been compared to Network Wide Time Synchronization (NWTS) which is a widely used algorithm for time synchronization in WSN. Simulation results have shown that NWTS uses about 10% more energy than EETS. NWTS synchronizes 90% of the network nodes but reduces the lifetime of the network considerably while EETS synchronizes only nodes that need to be synchronized and thus saving energy.

Keywords: Time Synchronization, Energy Efficiency, Sensor Networks, Clock Drift, Network Delay

I. INTRODUCTION

A Wireless Sensor Network (WSN) can be defined as a network of autonomous devices with built-in sensors that cooperate to monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different locations. Unlike traditional networks, WSN usually have an ad-hoc deployment and node volatility is a common occurrence. In addition to this, sensor nodes need to operate at significantly low energy level as these nodes have limited amount of energy thus making their deployment a real challenge [1, 2]. Sensor nodes are becoming smaller with lower production cost and the ability to place sensor nodes in remote and dangerous areas without worrying about communication lines is definitely a key aspect over traditional networks. Each node in the network monitors a region by sensing the events occurring in their surroundings. This sensory input, when gathered from all the nodes and analyzed by more traditional computers, paints a comprehensive, high-resolution picture of the surroundings in real time.

With technological advances and the rise of its importance, WSN has been used in more critical applications where the data needs to be communicated at the right time [3]. One common problem in traditional networks as well as in WSN is the clock drift problem [4, 5]. The clocks of different node run at different speed. After some time, the clock readings of the nodes will be different. Now if the nodes need to communicate based on time, a node can reject a packet considering it to be outdated compared to its clock.

In traditional network, the time can be synchronized with the use of methods like Network Time Protocol [6] as well as Global Positioning System. But for WSN these methods are not suitable. The sensor nodes in WSN have limited amount of energy. So sensor nodes cannot communicate with a base server or a GPS very often. Maintaining a connection with a base server or a GPS will use up the energy of the nodes and after some time all the nodes in the network will be dead. Another constraint is that the sensor node being small devices have low computational power. Complex algorithms cannot be run on these processors as it will use considerable amount of energy. With respect to all the constraints of the WSN, an algorithm which not just synchronize time efficiently but should also do it with the use of minimum amount of energy is required.

II. TIME SYNCHRONIZATION IN WIRELESS SENSOR NETWORKS

Communication between the nodes is the key factor in a WSN. It is important that the nodes provide the right information at the right time. This is why time synchronization in WSN plays an important role. Time synchronization consists of setting the clocks of each node in the network to the same global time. At the start when the WSN is deployed, all the nodes have their clocks synchronized. But the problem is, as time goes by, the clocks of the nodes will have different values. This problem arises because each node's clock run at different speed and some might be faster and some slower. The clocks are said to be drifting apart and this phenomenon is known as Clock Drift [4].

Time Synchronization is a critical factor in WSN where the nodes have to communicate with each other [7, 8]. It is very important to keep the time of each sensor node to be synchronized with a global time so that when the sensor reading is taken it is in the order to determine the correct undergoing of the monitored event. If the time of the sensor nodes are not synchronized the base station can have problems in reordering of the sequence of an event.

Another importance of time synchronization in WSN is when sensor nodes have to transfer data to the base station. Sensor nodes communicate with the neighboring node to pass on the data which in turn passes the data to the other node until it reaches the base station. Sensor nodes are low energy devices and cannot be running all the time. When the sensor nodes are not running they are said to be sleeping so that they can conserve energy. In a multi-hop system the neighboring nodes should be awaken at the same time. This means the time at which they awake or go asleep should be synchronized [9]. If it is not synchronized, when a sensor node has to send a packet to the base station, the propagation delay will be high as some nodes will be asleep when they have to relay a packet.

Wireless Sensor Networks requires more precision in time synchronization than traditional networks due to their close coupling with the physical world and energy constraints [10, 11]. There are many domain specific applications which require precisely synchronized time and in addition to these domain specific applications, sensor network applications rely on time synchronization as traditional distributed systems do. Time synchronization is needed for secure cryptographic schemes, ordering logged events during system debugging, coordination of future action, and so forth.

The protocols and algorithms used for traditional networks do not work efficiently in WSN. For example if the clock of a sensor node is to be synchronized, it is difficult to know with which other node to synchronize and whether the other node's clock is showing the right time. The algorithms used for traditional networks should be enhanced or an algorithm which not only synchronizes the time in WSN, but also meet the challenges like energy efficiency, scalability, robustness, and ad-hoc deployment of WSN, should be devised. Many algorithms have been proposed for time synchronization in WSN. Some of the algorithms are discussed below.

A. Delay Measurement Time Synchronization (DMTS)

DMTS is a flexible and lightweight technique for both single and multi hop WSN [12]. The nodes in DMTS are synchronized based on the concepts of event timestamp and network event scheduling [7]. The main idea of DMTS is to choose a leader which broadcast its local time value to other nodes. The receiver nodes synchronize with each other as they can synchronize with each other better than synchronizing with the sender.

B. Reference-Broadcast Synchronization (RBS)

The RBS aims at using the broadcast property of WSN to build up an algorithm for time synchronization [13]. In WSN, two receivers found in the listening distance of the same sender, receives a message sent by the sender at approximately the same time. If the receivers record their local time at the moment the message is received, the sensor nodes can compare and synchronize their local clocks precisely.

C. Global Clock Synchronization

Global clock synchronization is very important for sensor network applications that require interfacing with time of the events in the sensor networks. Li and Rus [14] have proposed a rate-based algorithm with both synchronous and asynchronous implementations of the algorithm. Global synchronization is achieved by flooding the local synchronization information to the whole system. Various global values can be chosen to synchronize the network. The easiest one will be to take either the maximum or the minimum value. However a faulty or malicious node can provide an incorrect high or low clock reading. In order to have a more robust algorithm, the global average is used as the ultimate clock reading.

D. Network-wide Time Synchronization

Network-wide time synchronization [15] concentrates more on the scalability of WSN. It aims at making sure that synchronization accuracy is not degraded as the number of nodes in the network increases. The main idea in network-wide time synchronization is to create a hierarchical structure in a WSN. In this structure a node can be acting simultaneously as a server to a number of nodes while acting as a client to another node. The network-wide time synchronization algorithm works in two phases: The level discovery phase and the synchronization phase.

E. Probabilistic Clock Synchronization Service

Probabilistic clock synchronization [16] is an extension to Reference Broadcast Synchronization protocol [13]. It makes use of the higher precision of the receiver to receiver synchronization and extends RBS to provide a probabilistic bound on accuracy of clock synchronization.

F. Flooding Time Synchronization Protocol (FTSP)

For high precision applications, Flooding Time Synchronization Protocol (FTSP) [17] is a good technique to go for. FTSP was developed for a sniper localization application where very high precision is required. FTSP uses customized MAC-layer time stamping and calibration to eliminate delays. FTSP synchronizes time between a sender and multiple nodes using a single radio message which is time-stamped at both sender and receiver sides. The message consists of the sender's time representing the global time at the transmission of a byte. When the receivers get the message, the sensor nodes get their local time and thus it forms a global-local time pair for each receiver. The receiver's clock offset is obtained from the difference between the global and the local time.

G. Comparison of existing algorithms

Table I. Comparison of Time Synchronization Protocols

	Energy Efficiency	***Accuracy***	***Scalability***	***Robustness***
DMTS	High	High	Good	Good
RBS	High	High	Good	Good
Global Clock Synchronization	Average	High	Good	Average
Network-Wide Synchronization	Average	High	Good	Average
Probabilistic Synchronization	High	Average	Good	Average
FTSP	High	High	Good	Good

From the comparison table, it can be observed that DMTS, RBS, Probabilistic Synchronization and FTSP have high energy efficiency compared to Global clock synchronization and Network-Wide Synchronization. In terms of accuracy, except probabilistic synchronization, all the protocols provide a high accuracy in their synchronization algorithm. All of the protocols provide good scalability but only DMTS, RBS and FTSP are robust enough to manage loss of nodes.

The main problem with the above algorithms is that all the nodes in the network are synchronized, even the nodes that which are currently not active. Synchronizing the nodes will cause them to move from a sleep state to an active state, perform synchronization and then go back to sleep state. In this process the nodes use up energy. This gives rise to the need of a new time synchronization algorithm which will synchronize only the nodes that are going to communicate and send data. It will avoid unnecessary use of energy and will increase the lifetime of the network.

III. PROPOSED TIME SYNCHRONIZATION ALGORITHM (EETS)

A good time synchronization algorithm should take into consideration the parameters of energy consumption in the sensor nodes. It should be able to handle and minimize energy usage and thus perform energy efficient time synchronization. In addition to minimize energy consumption, the algorithm should be able to synchronize time accurately. For some critical applications it is very important that the clocks of the nodes are synchronized to the maximum accuracy. WSNs are highly dynamic with large number of nodes joining or leaving the network. Thus the time synchronization algorithm should be scalable and robust.

In order to satisfy the above requirements, Energy Efficient Time Synchronization (EETS) algorithm is being proposed which is a new approach for time synchronization in WSN. The goal of EETS is not just to synchronize the time in WSN but it should be doing it in such a way that the minimum amount of energy is used. EETS consists of two phases: The Level discovery phase and the Synchronization phase. The Level discovery phase is performed the first, followed by the Synchronization phase. But the important part is that synchronization is performed only when a node needs to be synchronized.

A. Level Discovery Phase

In the level discovery phase, a root node is selected in the network. This root node is given the Level 0. The Level 0 node will broadcast a level discovery packet which contains its own level. When the neighboring receives this packet, they assign themselves one level higher than the level in the packet. That is the neighbor of node level 0 will be assigned level 1. The level 1 node then broadcast a level discovery packet which contains their level. Again the nodes receiving this packet will assign themselves one level higher than in the packet. The neighbors already having a level will ignore the level discovery packet. This process goes on until all the nodes in the network have been assigned a level.

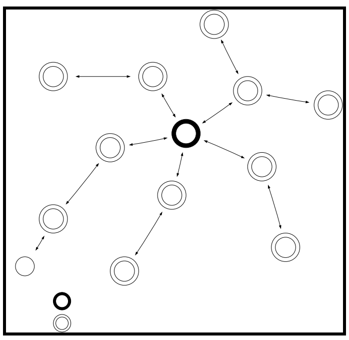

Figure 1. Level discovery process

B. Synchronization Phase

The synchronization phase will be run on only nodes that need to be synchronized. Synchronization will take place between a node and its neighbor which is found at a lower level. That is node at level N will synchronize with node at level N-1.

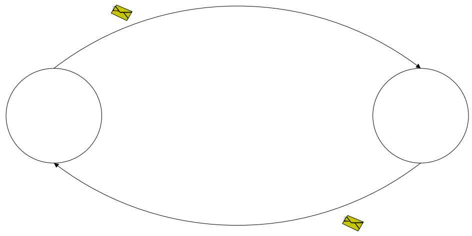

Figure 2. Synchronization phase of EETS

The steps for the synchronization between two nodes A and B found in the network are as follows:

- Node A transmits the first packet which contains a timestamp t1 with respect to its local time.
- When node B receives the first packet, it records the time t_2. Time t_2 is defined as time t1 plus the transmission time D from node A to B plus the offset d between the node A and B's clock.
- Node B then transmits a second packet to node A. The second packet contains the value of t_1 and t_2 as well as t_3, the time at which the packet was sent with respect to the local time of node B.
- Node A will receive the second packet at time $t_4 = t_3 + D + d$.
- The offset d can now be calculated at node A by subtracting t_4 from t_2
  - $t_2 - t_4 = t_1 - t_3 - D + D + 2d$
  - $d = 0.5*(t_2 - t_4 - t_1 + t_3)$
- Once node A has found the offset it can adjust its clock to be synchronized with node B

C. Calculation for Network Delay (D)

Network Delay in WSN networks depends on the following values:

- Send time: The time needed for assembling the message and includes processing and buffering time.
- Propagation time: The time taken for the signal to travel from the sender to the receiver. It is proportional to the distance between the sender and receiver and thus propagation time is the same in both directions.
- Receive time: The time for receiving the message from the channel and notifying the host of the receipt of the message.
- Access time: The time associated with accessing the channel including carrier sensing.

For the implementation of EETS we will consider that the send time, propagation time, receive time and access time are all the same for every node in the network. Thus the time taken to send a message from the sender to the receiver will be the same as the time taken for the receiver to send a message to the sender.

$D_{Sender} = D_{Receiver}$ ………………………………….. (1)

D. Calculation of Clock Offset (d)

It is very important for EETS to know the clock difference between two nodes in order to be able to synchronize them. The clock offset (d) between the nodes will be calculated by subtracting the time at which the receiver receives the packet (t_2) from the time at which the sender receives the packet (t_4).

t2 = t1 + D + d …….. (2)
t4 = t3 + D – d …….. (3)

Therefore:

t2 - t4 = t1 – t3 – D + D + 2d ……. (4)
d = 0.5*(t2 – t4 – t1 + t3) …….. (5)

IV. IMPLEMENTATION OF EETS

In order to provide a full simulation of the EETS, the algorithm was implemented and the following assumptions were made for the implementation:

- The WSN consists of clusters with varying number of nodes.
- The nodes communicate with nodes within a specified distance to it.
- The greater the distance between two nodes the greater will be the energy used to transmit messages.
- The initial energy of all the nodes is equal at the start of the simulation.
- EETS will be run on a cluster of nodes and not on the entire network.

A. Energy Model

We use the assumptions in [26] as the basis to calculate the energy dissipation for our simulations, which are as follows:

- Energy consumption for modulating or demodulating one bit:
  $E_{elec} = 50nJ/bit$
- Energy consumption for spreading one bit to an area of radius r = 1 meter (i.e., πm2):
  $\epsilon_{amp} = 100pJ/bit/m2 = 0.1nJ/bit/m2$
- Data rate = 2000bits/s
- Data package size = 2000-bit
- Signal package size = 64-bit
- The radio board consumes 100 µJ for each received data message:
  $E_{Rx\_data} = E_{elec} * k\text{-}bit/message = 50nJ/bit * 2000\ bits/message = 100\ \mu J/message$
- The radio board consumes 3 µJ for each received signal message:
  $E_{Rx\_signal} = E_{elec} * k\text{-}bit/message = 50nJ/bit * 64\ bits/message = 3.2\ \mu J/message \sim= 3\ \mu J/message$
- The radio board consumes (100 + 200*d^2) µJ for transmitting a data message to a distance d:
  $E_{Tx\_data} = E_{elec} * k\text{-}bit/message + \epsilon_{amp} * k * d^2 = 50\ nJ/bit * 2000\ bits/message + 0.1\ nJ/bit*2000\ bits/message * d^2 = (100\ \mu J + 200 * d^2)/message$
- The radio board consumes (3 + 64* d^2) µJ for transmitting a signal message to a distance d:
  $E_{Tx\_signal} = E_{elec} * k\text{-}bit/message + \epsilon_{amp} * k * d^2 = 50\ nJ/bit * 64\ bits/message + 0.1\ nJ/bit*64\ bits/message * d^2 = (3\ \mu J + 6.4 * d^2)/message$
- Assuming that the optimized communication radius of nodes is 60m. The radio board consumes 820 µJ for transmitting a data message to distance d<= 60m:
  $E_{Tx\_data} = 2000(bit)*(50(nJ) + 0.1(nJ)*60*60) = 820\ \mu J/message$
- The radio board consumes 26 µJ for transmitting a data message to a distance d<= 60m:
  $E_{Tx\_signal} = 64(bit)*(50(nJ) + 0.1(nJ)*60*60) = 26.2\ \mu J/message \sim= 26\ \mu J/message$
- If the radio board is in receiving mode, it consumes 100 µJ at each second:
  $E_{Radio} = Eelec * data\_rate = 50nJ/bit * 2000\ bits/s = 100\ \mu J/s$
- If the sensor board is in full operation mode, it consumes 66 µJ at each second:
  $E_{Sensor} = E_{Radio} * 2/3 = 66\ \mu J/s$
- In general, the MCU (Memory board, CPU board) is in sleep mode, it just switches to active when having an external interrupt. So, in the simulation, we assume that the processor is in sleep mode. It turns to full operation when having an event. It means when creating a new message, energy consumption is calculated as follows:
  $E_{MCU\_data} = 2000(bit)*50(nJ) = 100\ \mu J/message$
- The MCU board consumes 3 µJ for creating a signal message.
  E_{MCU_signal} = 64(bit)*50(nJ) = 3.2 ~= 3 µJ/message

Derived from the above calculations, Table 1 summarizes the operations and their respective energy consumption.

Table II. Energy consumption of operations

Operations	*Energy*
Create/Receive a data message	100 µJ
Create/Receive a signal message	3 µJ
Send a data message (d<= 60m)	820 µJ
Send a signal message (d<=60m)	26 µJ
Send a message (d > 60m)	100 µJ + 0.1* d^2
Sensor board (full operation)	66 µJ/s
Radio board (idle/receive mode)	100 µJ/s

V. SIMULATION STUDY

The simulation study of EETS has been performed using varying parameters under different scenarios. There are some parameters that have been set using random functions and as a result the simulation depends on the randomness of these functions. During the simulation study, the average energy level of the WSN was observed relative to the amount of time the simulation has been running. Moreover EETS has been compared to Network Wide Time Synchronization (NWTS) for the different scenarios. EETS is simulated in an environment where sensor nodes are spread randomly on a

field of given size. The sensor nodes will be reporting any event they sense to the cluster head.

A. Input parameters

- Size of the field
- Network size that is the number of nodes
- Number of random events occurring
- Initial battery level
- Duration of simulation
- Clock drift rate

B. Output parameters

- Number of remaining node
- Time of each sensor node
- Remaining energy of each node
- Average energy of the WSN

C. Scenarios

EETS has been simulated by establishing different scenarios. In these scenarios the input parameters has been varied and graphs of the average remaining energy of the WSN is plotted against time. In order to examine the efficiency of EETS, NWTS has been run under the same scenarios, and the results have been compared.

D. Scenario 1

Scenario 1 consists of varying the number of nodes in the network while keeping the other parameters constant. This scenario shows how EETS and NWTS behave when the network size increases.
Size of the field = 300m X 300m
Network size, that is the number of nodes = 100, 200, 400, 600, 800, 1000
Number of random events occurring = 100
Initial battery level = 2000 mJ
Duration of simulation = 120s
Clock drift rate = 1s

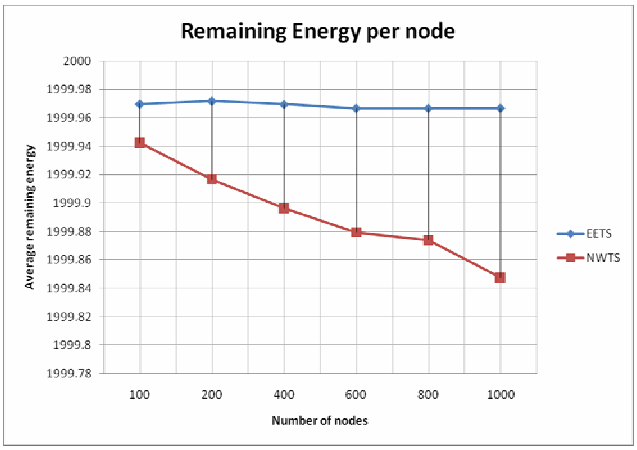

Figure 3. Comparison between EETS and NWTS

Form the above graph it can clearly be seen that EETS is much more energy efficient than NWTS. As the number of nodes increase, the average remaining energy for EETS remains approximately the same but NWTS uses about 10% more energy compared to EETS.

E. Scenario 2

In this scenario, the size of the field as well as the number of nodes in the network is varied. For each size of the network, the number of nodes is varied from 100 to 1000 during each simulation.
Size of the field = 300m X 300m, 600m X 600m, 900m X 900m
Network size, that is the number of nodes = 100, 400, 800, 1000
Number of random events occurring = 100
Initial battery level = 2000 mJ
Duration of simulation = 60s
Clock drift rate = 1s

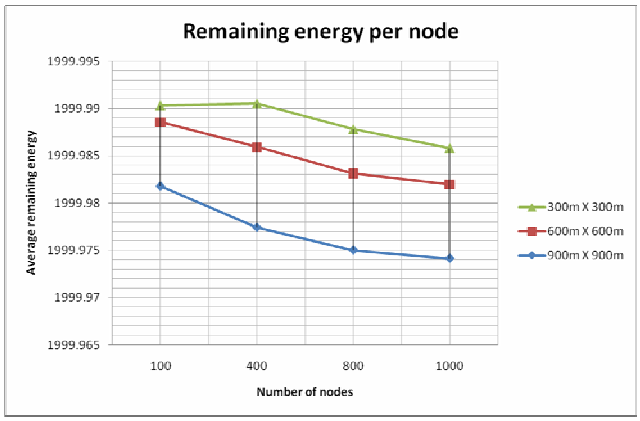

Figure 4. Varying the network size

From this simulation it can be seen that as the size of the network increases, the amount of energy required synchronizing the nodes increases.

F. Scenario 3

Scenarios 3 consist of keeping the field size constant and vary the number of nodes in the network. For each simulation the number of random events occurring is increased. In this scenario, the number of nodes synchronized by EETS and NWTS as well as the amount of energy remaining is observed.
Size of the field = 600m X 600m
Network size, that is the number of nodes = 100, 400, 800, 1000
Number of random events occurring = 100, 400, 800, 1000
Initial battery level = 2000 mJ
Duration of simulation = 60s
Clock drift rate = 1s

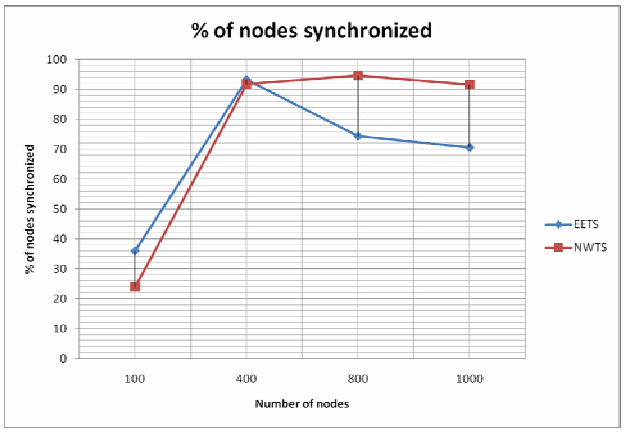

Figure 5. Percentage of nodes synchronized for EETS and NWTS

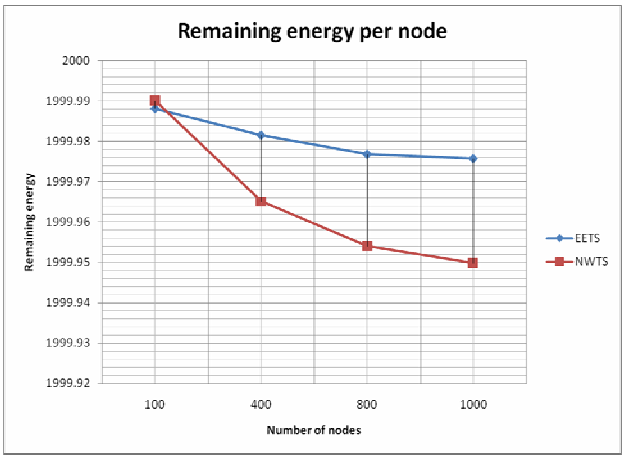

Figure 6. Remaining energy after synchronization

From the two above figures (16 and 17) we can see that NWTS synchronizes more nodes than EETS but it uses more energy. Synchronizing only nodes that needs to be synchronize is much more energy efficient. We can see that EETS saves about 3% more energy than NWTS.

G. Scenario 4

In this scenario we keep all the parameters constant and vary only the number of random events occurring. This will show how energy is consumed as the number of events increases.

Size of the field = 600m X 600m
Network size, that is the number of nodes = 800
Number of random events occurring = 100, 200, 400, 800
Initial battery level = 2000 mJ
Duration of simulation = 60s
Clock drift rate = 1s

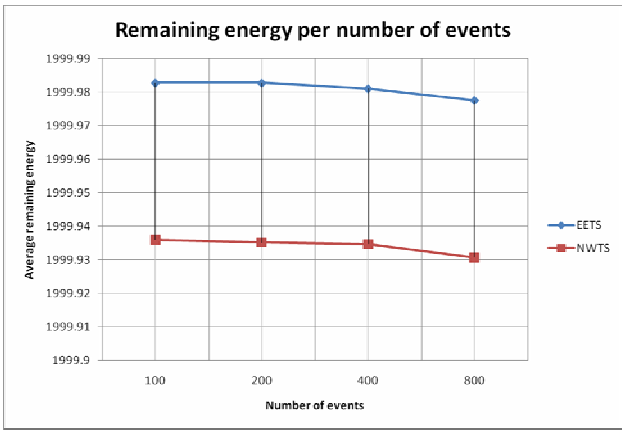

Figure 7. Remaining energy per number of events

As the number of events increase, we can see that EETS uses more energy to synchronize the nodes. But it can be seen that, as the number of events increases, NWTS uses even more energy than EETS.

VI. INTERPRETATION OF RESULTS

In scenario 1, the number of nodes deployed in the network is varied, keeping all other parameters constant and the amount of energy used by EETS and NWTS is observed. It has been observed that EETS uses about the same amount of energy as the number of nodes increase. This is because EETS will use a little more energy to set the levels of more nodes. But if the number of occurring events remains the same, EETS will use approximately the same amount of energy to synchronize the nodes. The difference in the amount of energy required will depend on the distance of the nodes.

NWTS uses much more energy because it tries to synchronize the whole network. In a small network (100 nodes) NWTS uses less energy. But when the network is big (1000 nodes) it uses considerable amount of energy. While trying to synchronize the whole network, the average amount of remaining energy in the network decreases. Thus the lifetime of the network also decreases. In order to save energy it is better to synchronize only the nodes that need to communicate instead of synchronizing the whole network.

In scenario 2, the network size as well as the number of nodes in the network is varied. It is observed that as the network grows in size, the amount of energy required to synchronize the nodes also increases. This is because the distance between the nodes increases as the network size increases. With increase in distance the nodes require more energy to communicate and thus the average remaining energy of the network also decreases.

Scenario 3 is set up to observe what percentage of the network is synchronized by EETS and NWTS. We can see from figure 17 that as the number of nodes in the network increases the percentage of nodes synchronized by EETS is less than that of NWTS. This is because NWTS runs regularly at an interval of time. The more the number of cycles, the more nodes synchronized. But EETS synchronizes only nodes that have detected an event and that have to communicate with the sink node. So the percentage of node synchronized as the network grows bigger is lower than NWTS. On the other hand, looking at figure 18, it is observed that NWTS uses considerable amount of energy to synchronize unnecessary nodes. At this rate the lifetime of the network will decrease rapidly. So it is better to synchronize only nodes that need to be synchronized and save energy.

In scenario 4, the number of events occurring is varied for a fixed size network to evaluate the energy consumption of EETS and NWTS. EETS does not use much energy to synchronize the nodes as the number of events increases. But NWTS has to handle both the occurring event as well as the synchronization. While synchronizing nodes only involved with the event, EETS saves much more energy than NWTS which synchronizes the whole network as well as handles the event.

It has been seen from the simulation results that EETS is much more energy efficient than NWTS. This is because EETS synchronizes only the nodes that need to be synchronized. NWTS uses about 10% more energy than EETS for synchronization. The nodes remain in a sleep state until an event occurs. When an event occurs before handling this event, EETS synchronizes the nodes that are on the path to the sink node. Thus it is not necessary to synchronize all nodes in the network as this will reduce the network lifetime.

VII. CONCLUSION

This paper aimed at addressing the problem of time synchronization in Wireless Sensor Networks. It has been seen that WSN are used in many different fields like agriculture, security, industrial, transport, asset tracking and ubiquitous computing. Most of these applications are time critical applications. With increase in the use of WSN, the importance of accurate communication has gain importance. The sensor nodes in the network should be able to communicate at the right time to give the correct results. But

very often, the clocks of the sensor nodes have different values. The clocks of the sensor nodes have different values as each of them run at a different speed. At the beginning, the clock values will be the same, but after some time the values will differ from each other. This is known as clock drift whereby the clock values drift away from the correct value. Time synchronization is a very important factor in communication between the nodes. Before coming up with a solution for time synchronization, the different challenges for designing an algorithm for WSN have been studied. The sensor nodes of a WSN are small devices with low amount of energy. An algorithm that will be running on a WSN should take into account that it does not use up all the energy of the sensor nodes.

A comparative study of different existing algorithms for time synchronization has been made. The study of different existing algorithms has lead to setting up the requirements for a new time synchronization algorithm and this algorithm has been named Energy Efficient Time Synchronization (EETS). EETS aims at performing time synchronization with the minimal use of energy in the WSN. The algorithm works in two phases. The first phase is the Level discovery phase. In this phase, each node of the network is assigned a level. The second phase is the synchronization phase. Synchronization takes place only between nodes that needs to be synchronized. The other nodes in the network stay in a sleep state. When an event occurs in the network, the node sensing the event will synchronize with a node at a lower level before starting to send the collected data.

The EETS has been implemented, executed under different scenarios and compared to Network Wide Time Synchronization (NWTS). Simulations have showed that as the network size increases, NWTS uses about 10% more energy than EETS. NWTS synchronizes more nodes in the network but at the same time uses more energy. EETS synchronized only the nodes that need to be synchronized and at the same time saves the overall energy of the network. As a result EETS has proved to be energy efficient and provides and accurate synchronization of the nodes.

VIII. REFERENCES

[1] M. Molla, and S.I. Ahamed, "A Survey of Middleware for Sensor Network and Challenges," In the Proceedings of the 2006 International Conference Workshops on Parallel Processing, 14-18 August 2006, Columbus, Ohio, USA. Washington: IEEE Computer Society, pp. 223-228.

[2] A. Bharathidasan, and V.A. Ponduru, "Sensor Networks: An Overview", IEEE Potentials, April-May 2003, vol. 22, issue 2, pp. 20- 23.

[3] J. Blumenthal, M. Handy, F. Golatowski, M. Haase, and D. Timmermann, "Wireless Sensor Networks - New Challenges in Software Engineering," Proceedings of 9th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Lissabon, Portugal, September 2003.

[4] R. Tjoa, K.L. Chee, P.K. Sivaprasad, S.V. Rao, and J.G. Lim, "Clock drift reduction for relative time slot TDMA-based sensor networks," 15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, 2004. PIMRC 2004, 5-8 Sept. 2004, pp. 1042- 1047.

[5] D. Sanchez, "Secure, accurate and precise time synchronization for wireless sensor networks," Proceedings of the 3rd ACM workshop on QoS and security for wireless and mobile networks, Chania, Crete Island, Greece, pp. 105 – 112, 2007.

[6] D.Mills, "Internet Time Synchronization: The Network Time Protocol," Global States and Time in Distributed Systems. IEEE Computer Society Press, 1994.

[7] B. Sundararaman, U. Buy, A.D. Kshemkalyani, "Clock synchronization for wireless sensor networks: a survey", Ad Hoc Networks, vol. 3, issue 3, pp. 281-323, May 2005.

[8] C. Lenzen, P. Sommer and R. Wattenhofer, "Optimal clock synchronization in networks," Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, Berkeley, California, USA, pp. 225-238, 2009.

[9] Y. Wu, S. Fahmy, and N.B. Shroff , "Optimal sleep/wake scheduling for time-synchronized sensor networks with QoS guarantees," IEEE/ACM Transactions on Networking (TON) , vol. 17, Issue 5, October 2009.

[10] S. Yoon, C. Veerarittiphan, M.L. Sichitiu, "Tiny-sync: Tight time synchronization for wireless sensor networks", ACM Transactions on Sensor Networks (TOSN), vol.3, issue.2, June 2007.

[11] E. McKnight-MacNeil and T. Kunz, "Behavior of clock-sampling mutual network synchronization in wireless sensor networks", Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing, Leipzig, Germany, pp. 633-638, June 2009.

[12] S. Ping, "Delay measurement time synchronization for wireless sensor networks," Technical Report IRB-TR-03-013, Intel Research, June 2003.

[13] J. Elson, L. Girod, and D. Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts," In Proceddings of the 5th ACM Symposium on Operating System Design and Implementation (OSDI-02), Operating System Review, pp. 147-164, New York, December 9-11 2002. ACM Press.

[14] Q. Li and D. Rus, "Global Clock Synchronization in Sensor Networks," Proc. IEEE Conf. Computer Communications (INFOCOM 2004), vol. 1, pp. 564–574, Hong Kong, China, Mar. 2004.

[15] S. Ganeriwal, R. Kumar, S. Adlakha, and M. Srivastava, "Network-wide Time Synchronization in Sensor Networks," Technical Report, Networked and Embedded Systems Lab, Elec. Eng. Dept., UCLA, 2003.

[16] S. PalChaudhuri, A. Saha, and D. B. Johnson. "Probabilistic Clock Synchronization Service in Sensor Networks," Technical Report TR 03-418, Department of Computer Science, Rice University, 2003.

[17] M. Maroti, G. Simon, B. Kusy, and A. Ledeczi, "The flooding time synchronization protocol," in Proceedings of the 2nd international conference on Embedded networked sensor systems", Baltimore, MD, USA, Nov. 2004, pp. 39–49.

