INTECH OPEN ACCESS PUBLISHER

OPEN ACCESS ARTICLE

International Journal of Radio Frequency Identification and Wireless Sensor Networks

Adaptive Time Synchronization for Homogeneous WSNs

Regular Paper

Siddhartha Chauhan* and Lalit Kumar Awasthi

Department of Computer Science and Engineering, National Institute of Technology, India
* Corresponding author E-mail: siddharthachauhan1@gmail.com

Received 22 Jul 2011; Accepted 01 Nov 2011

Abstract Wireless sensor networks (WSNs) are being used for observing real-world phenomenon. It is important that sensor nodes (SNs) must be synchronized to a common time in order to precisely map the data collected by SNs. Clock synchronization is very challenging in WSNs as the sensor networks are resource constrained networks. It is essential that clock synchronization protocols designed for WSNs must be light weight i.e. SNs must be synchronized with fewer synchronization message exchanges. In this paper, we propose a clock synchronization protocol for WSNs where first of all cluster heads (CHs) are synchronized with the sink and then the cluster nodes (CNs) are synchronized with their respective CHs. CNs are synchronized with the help of time synchronization node (TSN) chosen by the respective CHs. Simulation results show that proposed protocol requires considerably fewer synchronization messages as compared with the reference broadcast synchronization (RBS) protocol and minimum variance unbiased estimation (MUVE) method. Clock skew correction mechanism applied in proposed protocol guarantees long term stability and hence decreases re-synchronization frequency thereby conserving more energy.

Keywords Clock synchronization, time synchronization, wireless sensor networks (WSNs).

1. Introduction

The SNs are powerful to enough to process data and to transmit it in a limited range. SNs are placed near the environment for observing real-world phenomena. WSNs can be used for vide variety of applications and can be placed in dangerous or hostile environments. WSNs can be homogenous or heterogeneous networks. In homogenous WSNs all the nodes are identical in terms of battery life and other hardware resources. Heterogeneous WSNs are composed of nodes having different battery life and hardware.

The information collected by various sensors has to be pieced together in order to have a broader picture of the phenomenon being sensed by individual SNs. It is important for WSNs applications that the SNs agree on a common time. In time slotted MAC protocols, the multiple access to shared medium is achieved by assigning time slots to the nodes. Hence, to prevent collision the nodes need to be synchronized. Nodes also require synchronization in order to coordinate their sleep/listen periods.

Data fusion is one of the most important and basic operations of WSNs, where data from individual sensor is combined so that a single meaningful result is formed [1, 2, and 3]. Fusion of readings from various sensors is possible if the messages (about the readings of individual sensors) are time stamped by each SN. Hence, it is important that all the sensors must be properly synchronized.

Designing clock synchronization algorithms for WSNs is very challenging as WSNs are different than other networks. These networks are resource constrained hence algorithms designed for sensor networks should be energy efficient. WSNs can have several thousands of sensors and nodes can only communicate with its neighbors. The protocols for sensor network must be scalable to adapt to such networks. Traditional protocols developed for Adhoc networks and wired networks cannot be used in WSNs. If they are to be used in WSNs, it has to be with modifications keeping in view the characteristics of WSNs.

Clock synchronization in WSNs requires new and better approaches. Since sensor networks are used for various applications hence application specific protocols can be designed for better performance. Synchronization protocols can be application specific as some applications can have tight constraints of time or of energy consumption etc. Time synchronization protocols must address to the issues of sensor networks like energy consumption, limited bandwidth, and limited hardware. The restriction on transmission range, computational capabilities of the nodes and storage space also puts a challenge for designing light weight energy efficient mechanisms. Clock synchronization protocols should have less communication and message overheads for WSNs.

In this paper we propose a clock synchronization algorithm for WSNs. The process of synchronization is initiated by the sink, which synchronizes the CHs. CNs are synchronized by their respective CHs through time synchronization node (TSN) chosen by the CHs. The proposed algorithm uses message broadcasts within clusters to reduce the exchange of synchronization messages amongst SNs. The proposed protocol is intended to be used for WSNs deployed in industrial applications including process control. This deployment requires tight time limits for synchronization and the proposed scheme suits to such deployment. It can also be used for other applications like fire hazard detection, habitat monitoring etc.

Rest of the paper is organized as follows: Section 2 presents related work. Section 3 presets the analytical model and problem setup. Section 4 presents our proposed algorithms and explains the working of our proposed algorithms. Section 5 presents results from our simulation experiments of our proposed protocol. Section 6 concludes the work and highlights future work.

2. Related work

A lot of protocols already exist in literature for synchronizing the nodes in WSNs. Some protocols are designed to synchronize the nodes with respect to a global clock while others synchronize the nodes with respect their own local time. The diverse applications of WSNs pose a challenge on the design of such protocols.

Reference Broadcast Synchronization (RBS) is a clock synchronization protocol which tries to reduce non-deterministic latency [4]. In RBS, a base node broadcasts a reference message to all the other nodes and the delays in message transmission paths are removed after offset compensation via least-squares linear regression employed at receivers. To achieve a network-wise synchronization, RBS uses a time-conversion scheme based on its receiver–receiver synchronization nature with the help of routing algorithms. RBS offers relatively high accuracy in offset estimation by removing the delays of the first three delay elements in the message-transmission process. This Protocol depends only on the packet receive time. This scheme uses multiple reference broadcasts in order to observe the variations between two nodes. Due to large number of messages exchanged its convergence time is high and message overheads for RBS are more.

The protocol presented in by Romer [5] uses time transformation algorithm for synchronization. This protocol was designed for Adhoc networks and has been applied for WSNs. The key idea of Romer's protocol is to generate local timestamps using local clocks which are transformed to local time of the receiving node. This transformation increases synchronization error as the number of hops increase along the path of the message timestamp. The synchronization achieved by this approach is localized and short lived.

In [6] the IEEE 802.11 master–slave protocol has been extended to provide fault tolerance and reduce message complexity. The protocol improves precision by exploiting the tightness of the communication medium, similar to RBS and also tolerates message loss. Continuous clock synchronization is performed and local clock time is corrected by gradually speeding up or slowing down the clock rate. However, this approach suffers from a high run-time overhead since clocks need to be adjusted every clock tick.

Network-wide Time Synchronization protocol [7] proposed for networks with large node density. The algorithm works in two phases. In the first phase a hierarchical topology is created, where every node is assigned a level in the hierarchy. In the second phase, every node synchronizes itself to a node belonging to exactly one level above in the hierarchy. Eventually all nodes in the network synchronize their clocks to a reference node. The algorithm is scalable and synchronization accuracy is not degraded significantly by increasing the number of nodes.

Delay Measurement Time Synchronization (DMTS) [8] protocol merges individual sensor readings on the basis of global timestamp. Local clocks of nodes are maintained and local clocks are synchronized over the whole network to create a network time. It has less computation cost though it is less accurate than RBS Protocol. The Probabilistic Clock Synchronization Service [9] for sensor networks has extended the RBS Protocol. This Protocol provides probabilistic bounds on the accuracy of clock synchronization [9]. It provides reasonable synchronization accuracy with lower computational and message overheads as compared to deterministic algorithms. The protocol is sensitive to message loss and cannot be used for safety critical applications.

Timing-Sync Protocol for Sensor Networks (TPSN) [10] synchronizes the clock using sender receiver synchronization. Sender communicates with the receiver to estimate the clock difference. TPSN logically organizes sensor network into hierarchical structure similar to [7]. This makes it inapplicable for mobile sensor networks. Nodes synchronize their clocks to the root node's clock using the hierarchical structure. Pairwise synchronization is then performed along the edges of the hierarchical structure established in the earlier phase according to the classical approach of the sender-to-receiver synchronization. The protocol is suitable for sensor networks that are highly constrained in bandwidth and computational power but its convergence time is high.
Pair-wise synchronization is used in Lightweight Time Synchronization [11] protocol which uses Gaussian error properties. In this protocol a packet is exchanged between two nodes, both nodes store transmitting and receiving time locally. After the packet exchange, offset is calculated by using stored time and clock is synchronized.

In [12], a global clock synchronization scheme is presented. This paper present's four methods to achieve synchronization. A node based approach, a hierarchical cluster based method, a diffusion based method and a fault tolerant diffusion based method. Node-based method transmits a packet around a cycle composed of all the nodes in the network. In clusters-based method, initially clusters head is synchronized, second synchronization round synchronize the members within each cluster with their cluster head. In diffusion-based method every node exchanges and updates information locally with its neighbors. The fault-tolerant diffusion-based method assumes the presence of malicious nodes that exhibit Byzantine faults.

Time-diffusion synchronization Protocol (TDP) [13] uses an iterative, weighted averaging technique which is based on a diffusion of messages in order to synchronize the network. In TDP nodes synchronize asynchronously with respect to each other. TDP maintains multiple master nodes which are distributed across the network, hence can function properly without external time servers. The message overhead for TDP is high and convergence time tends to be high when no external time servers are used. In [14] the time complexity of diffusion methods was analyzed.

Cluster-based power efficient (CPBE) [15] is an energy saving time synchronization protocol in which first a local synchronization synchronizes two connected cluster heads with each other. A virtual backbone network is established and each cluster head synchronizes to a virtual clock, which is maintained by two root nodes. Finally, all nodes in the cluster are synchronized with the cluster head.

Ilkay Sari et. al. have proposed a joint maximum likelihood (JML) protocol[16] which estimates clock offset and skew under exponential noise model in the RBS protocol. In [17] three estimator algorithms which are based on the two-way message-exchange mechanism have been proposed. These three estimators jointly estimate the clock offset and skew without knowledge of the fixed delay. The derived estimator includes the maximum-likelihood estimator (MLE), a generalized estimator of the MLE, and a newly proposed low-complexity estimator. Roxana Albu et.al. [18], proposed an IEEE1588-PBS Hybrid Protocol which minimizes the energy consumption while ensuring proper synchronization for WSNs.

Energy-Efficient Time Synchronization (EETS) [19] is a scheme for synchronizing the clocks of node by reducing the number of transmissions and message forwarding amongst SNs. EETS procedure is divided into two phase's namely level discovery phase and synchronization phase. In discovery phase hierarchical order is created and every node assigned a level. Synchronization Phase starts after discovery phase. Any broadcasted messages can be delivered to multiple nodes hence EETS uses an advantage of broadcast schemes to reduce transmissions. EETS is very efficient in reducing the number of generated messages between SNs and in delivering the messages rapidly during synchronization.

Feedback-based synchronization (FBS) [20] scheme uses the proportional–integral (PI) control principle to compensate the clock drift. The feedback-based synchronization scheme solves sleep-clock-synchronization problem. The first phase is spanning tree-construction phase, which is energy conserving and provides multihop synchronization accuracy. Second phase is iterative synchronization phase which is iteratively conducted, thereby increasing message overheads.

Temperature compensated time synchronization (TCTS) [21] exploits temperature information to increase the synchronization intervals. In synchronization of different clocks environmental issues like temperature is considered as an excuse but TCTS helps to overcome this problem by sensing the temperature and then calibrate the clocks. The differences between the clocks can be adjusted and temperature does not affect the clock. This results in overall network power savings since fewer synchronization messages have to be transmitted.

Clock synchronization problem is addressed in [22] by two-way timing packets exchange. Symmetric and asymmetric exponential link delay circumstances give the linear unbiased estimate of the clock offset between two nodes. The minimum variance unbiased estimation **(**MVUE) has been matched it with the maximum likelihood estimator (MLE) only in the symmetric link delay situation. In the region around the point of symmetry MLE performs better than MVUE as it has lesser mean-square error.

Clock synchronization protocols for WSNs not only have to be energy efficient and accurate but should also have less computational and message overheads. Various clock synchronization protocols have been evaluated in terms of accuracy, energy efficiency and complexity (computational and message overheads) in table 1.

Protocols	**Accuracy**	**Energy efficiency**	**Complexity**
RBS [4]	High	High	High
Romer [5]	Average	High	Low
Mock et al. [6]	High	Low	Low
Network Wide TSP [7]	Average	High	Low
DMTS[8]	Average	Low	Low
TPSN[10]	High	Average	Low
Pair wise Lightweight Protocol [11]	High	High	Low
TDP [13]	High	Average	High
CPBE [15]	High	Low	High
JML [16]	High	High	High
PBS Hybrid [18]	Average	Average	Low
EETS [19]	High	High	High
FBS [20]	High	Average	High
TCTS [21]	High	High	Low
MVUE [22]	High	Average	Average

Table 1. Comparison of clock synchronization protocols.

3. Analytical model

Wireless SNs are power constrained nodes and hence communication range of a sensor node is dependent on the transmission power. Synchronization algorithms for sensor networks are to be energy efficient and re-synchronization frequency should be less. Clock skew correction mechanism can guarantee stability of synchronization for longer time.

3.1 Cluster head synchronization

The two cluster heads can synchronize among themselves. Let us assume that *CHP* be the parent cluster head which synchronizes another cluster head *CHS*. Let the clock offset between both cluster heads is denoted by δ. The clock model for two way message exchange is depicted in Fig.1.

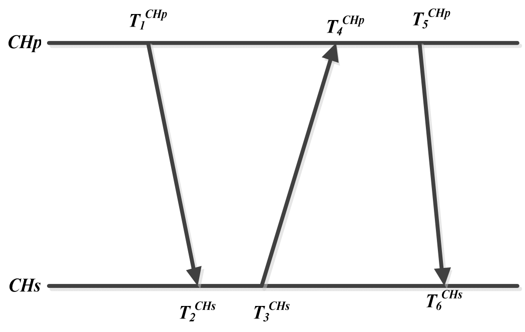

Figure 1. Cluster head synchronization.

CHP transmits a synchronization beacon to node *CHS*, at time$T_1^{CH_P}$. *CHS* receives it at its own clock time $T_2^{CH_S}$ and transmits an acknowledgement packet to *CHP* at $T_3^{CH_S}$ with time stamp$T_3^{CH_S}$. Acknowledgement is received at time $T_4^{CH_p}$ by *CHP*. *CHP* sends an acknowledgement to *CHS* at time $T_5^{CH_p}$ with time stamps $T_4^{CH_P}$ and$T_5^{CH_P}$. It is received at time $T_6^{CH_S}$ by *CHS*. Received packet will have delay component which can further be divided into fixed delay (*dfixed*) and variable delay (*dVar*). The time $T_6^{CH_S}$ and $T_4^{CH_P}$can be expressed as

$$T_6^{CH_S} = T_5^{CH_P} + \delta + d_{fixed} + d_{Var} \quad (1)$$

$$T_4^{CH_P} = T_3^{CH_S} - \delta + d_{fixed} + d_{Var} \quad (2)$$

If d is the total delay then $d = d_{fixed} + d_{Var}$. Total delay calculated from the equations (1) and (2) is given by the following equation (3).

$$d = ((T_6^{CH_S} + T_4^{CH_P}) - (T_5^{CH_P} + T_3^{CH_S}))/\,2 \quad (3)$$

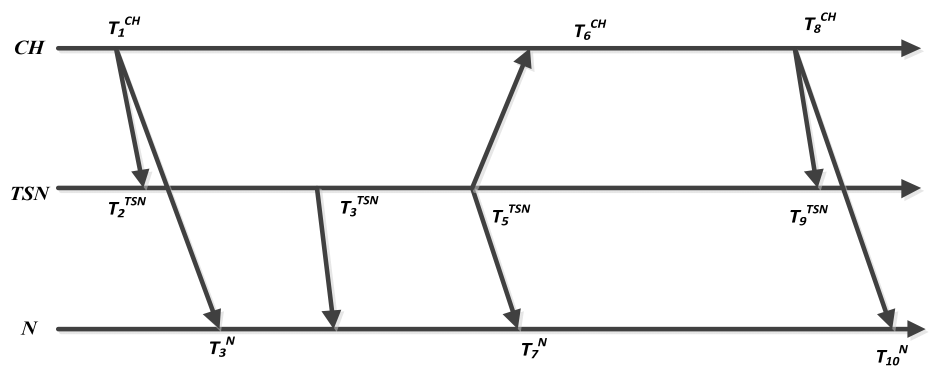

Figure 2. Node synchronization within cluster

3.2 Cluster node Synchronization

Nodes in a cluster can be synchronized with the cluster head. In Fig.2 the message exchange for synchronization of a node is shown. Cluster head (*CH*) synchronizes node (*N*) with the help of time synchronization node (*TSN*). One node in a cluster is chosen by *CH* as *TSN*. *CH* broadcasts a message with time stampT_1^{CH}. *TSN* on receiving it records its own clock time T_2^{TSN}. Similarly *N* also records its own timeT_3^N. *TSN* sends a message to *N* with a time stamp T_2^{TSN} at time T_3^{TSN} which is received by *N* at its timeT_4^N.

The time T_3^N and T_2^{TSN} can be expressed as

$$T_3^N = T_1^{CH} + \delta_N + d_N \qquad (4)$$

$$T_2^{TSN} = T_1^{CH} + \delta_{TSN} + d_{\,TSN} \qquad (5)$$

Where δ_N and $\mathrm{d_N}$ is the offset and total delay of cluster node with respect to Cluster head, similarly δ_{TSN} and $d_{\,TSN}$ is the offset and total delay of *TSN* with respect to cluster head. *N* can calculate the offset (δ′) with respect to *TSN* if message delay is same for *N* and *TSN* from *CH* as

$$\delta' = T_3^N - T_2^{TSN} = (\delta_N - \delta_{TSN}) \qquad (6)$$

If the delay is not same then the offset will have a component of delay, hence δ' can be calculated as

$$\delta' = T_3^N - T_2^{TSN} = (\delta_N - \delta_{TSN} + d_N - d_{TSN}) \qquad (7)$$

3.3 Calculation of skew

Clock skew can be calculated by a *N*. As shown in Fig.2, *CH* sends a message to *N* with timestamp T_1^{CH}. The message is received by *N* at T_3^N . *CH* waits for a random time(Δt_{CH}) and sends a message to the *N* at time T_8^{CH}, which is recived at time T_{10}^N by the *N*. Node *N* can calculate the random time as $\Delta t_N = T_{10}^N - T_3^N$. If both the nodes are synchronized or have a common clock then Δt_{CH} and Δt_N *will* be same. In WSNs because of the skew Δt_{CH} and Δt_N are different. Hence the skew can be calculated by *N* as $\rho_N = (\Delta t_N - \Delta t_{CH}) / \Delta t_N$. The above equation gives clock difference between *CH* and *N* for a unit time. If k rounds are performed for the synchronization then *N* calculates the skew for every round. If upper bound of the clock skew of *N* is known to *CH*, then *CH* can take a decision whether to initiate a new synchronization round or not. The makes this process self-learning and adaptive.

3.4 Energy consumption

If $E_{i,Initial}$ is the Initial energy of a node; which is full energy. $E_{i,Process}$ is the processing energy per bit, $E_{i,TX}$ is the transmission energy per bit, $E_{i,RX}$ is the per bit reception energy. Then the energy consumed by a node at particular time interval say Δt which is the interval between two consecutive synchronization rounds for a particular node is given by equation (7) as

$$E_{i,\Delta t} = (k\,(E_{i,RX})\,(PackLEN)) + (l(E_{i,Process})(PackLEN)) + (m(E_{i,TX})(\,PackLEN)) \qquad (8)$$

Where *k* is the number of synchronization packets received by the nodes, $PackLEN$ is the size of synchronization packet in bits, *l* is the number of synchronization packets processed by the node and *m* is the number of synchronization packets transmitted by the node.

4. Adaptive Time Synchronization protocol

Let us suppose that a WSN having *N* nodes is divided into *k* clusters as shown in Fig.3. Cluster heads of neighboring clusters can communicate with each other through gate node. All the nodes in a network have unique ID and nodes can identify each other. The clock in a sensor network can be inconsistent due to several reasons. The nodes in a sensor network may not be synchronized well initially, when the network is deployed. The sensors may be turned on at the different times and their clocks may be running according to different initial values. The results of events on specific sensors may also affect the clock.

The time difference between these nodes is called offset (δ). The time is measured as a function of the hardware oscillator $C(t) = k\int_{t_0}^{t}\omega(\tau)d\tau + C(t_0)$, where $\omega(\tau)$is the angular frequency of the oscillator, k is a constant for the oscillator, and t is the time [12]. The oscillator of the clock may be affected by temperature, pressure, battery voltage etc. The clock will have a skew (ρ) due to above stated reasons. In this paper we calculate nodes offset and clock skew with respect to sink and set the logical clock by using this parameter. Logical clock is denoted by *H(t)* and logical clock is incremented after every clock cycle *f(1)*.Suppose Δt is the real time of one clock pulse and Δh is the time of logical clock which is incremented after one clock pulse. We can calculate Δh by using Δt and ρ as follows:

$$\Delta h \;=\; (1 \pm \rho)\Delta t \qquad (8)$$

Initially offset (*δ)* can be calculated and logical clock can be set accordingly as *H(t)* = *C(t)* + δ. The clock will then be incremented after every clock pulse according to the following equations (9) and (10).

$$H(t + \Delta t) = H(t) + \Delta h \qquad (9)$$

$$H(t + \Delta t) = H(t) + (1 \pm \rho)\Delta t \qquad (10)$$

Our proposed protocol i.e. Adaptive Time Synchronization Protocol (ATS) follows the following steps for the synchronization:1) the Sink synchronizes the cluster heads, 2) cluster heads change their clock accordingly, 3) Cluster head synchronizes the nodes accordingly. Fig. 4 shows how synchronization is performed by ATS. The offset and skew calculated by ATS through message passing is shown in Fig.4. In Fig.4, *Sink* is the cluster head of cluster 0. It broadcasts a message M_{Sink}^{1} to all the nodes in the cluster. This synchronization message has time stamp T_{Sink}^{1} and ID of the node chosen as Time Synchronization Node (*TSN*). Any node can be chosen to be *TSN*. All the cluster nodes know their *TSN*. The message M_{Sink}^{1} will reach the cluster nodes with some delay. This delay will be almost same for all the cluster nodes but practically it will be different for each node. As shown in Fig.4, M_{Sink}^{1} is received at different times by different nodes. *TSN* records its own time T_{TSN}^{1} at which it receivedM_{Sink}^{1}. *TSN* now broadcasts a synchronization message M_{TSN}^{2}, which is received by all the nodes. *TSN* broadcasts in M_{TSN}^{2} timeT_{TSN}^{1}. All the nodes calculate offset and synchronize their clocks with respect to *TSN*. *TSN* again sends message M_{TSN}^{3}. All the nodes record their own time at which M_{TSN}^{3} is received. *Sink* broadcasts it's time T_{Sink}^{3} of reception of M_{TSN}^{3} in message M_{Sink}^{4} . All the nodes and *TSN* calculate offset and skew and set their clocks according to cluster head.

ATS works by progressively synchronizing the cluster heads with the sink and then the nodes within the cluster. ATS will work according to algorithm1, where firstly the sink synchronizes the cluster heads algorithm 2. Sink then synchronizes nodes of its cluster according to algorithm 3. The cluster heads of the neighboring clusters to sink calculate the delay according to algorithm 2. The cluster heads sends a message to sink and records it's time t_1. The sink node on reception of this message records its own time t_2. The sink node sends a message to cluster head at time t_3 with both recorded times t_2 and t_3, which records its own clock time t_4 on reception of this message. Delay is then calculated by the cluster head accordingly. The cluster heads which are not adjacent to sink will calculate the delay with respect to cluster heads synchronized by the sink. In Fig.3, cluster head CH$_6$ will calculate the delay with respect to CH$_3$ and set its time accordingly. The cluster heads synchronize all the cluster nodes within the cluster according to algorithm 3, by choosing any one node as TSN. Firstly all nodes adjust their time according to TSN and then their clocks are set according to cluster head. The skew is calculated individually by each node. Total number of message broadcasts required is only four. The average skew is calculated by the cluster head and it can take a decision whether to initiate a new synchronization round or not.

Algorithm 1. Adaptive Time Synchronization Algorithm

1. Network is organized into clusters by any clustering algorithm.
2. Sink synchronizes the cluster heads according to algorithm2.
3. Synchronize each cluster nodes (within cluster) with cluster head according to algorithm3.

Algorithm 2. Inter Cluster Head Synchronization Algorithm

1. Network is organized into clusters according to clustering algorithm.
2. Sink sends synchronization message to cluster head.
3. Cluster heads sends a synchronization message to sink and records the time t$_1$.
4. Sink records its own time t_2(Sink's time when it received message from cluster head).
5. Sink sends back message to cluster head at its time t_3 along with time t_2 and t_3.
6. Cluster head records its own time t_4 and calculates delay as

$$d \;= \frac{\left((t_4 - t_1) - (t_3 - t_2)\right)}{2}$$

7. Sink sends its current time $H_{sink}(t)$ to cluster heads adjust their clock as

$$H_{CH}(t) = H_{sink}(t)+d$$

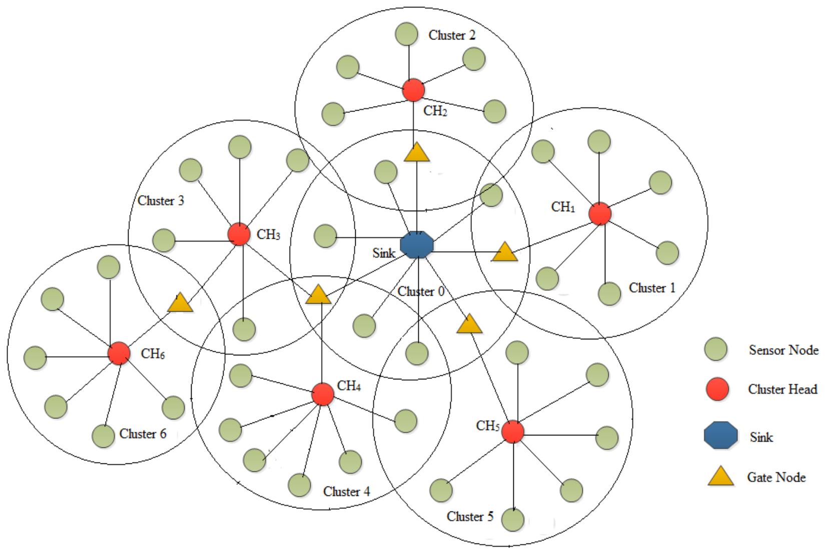

Figure 3. A clustered sensor network.

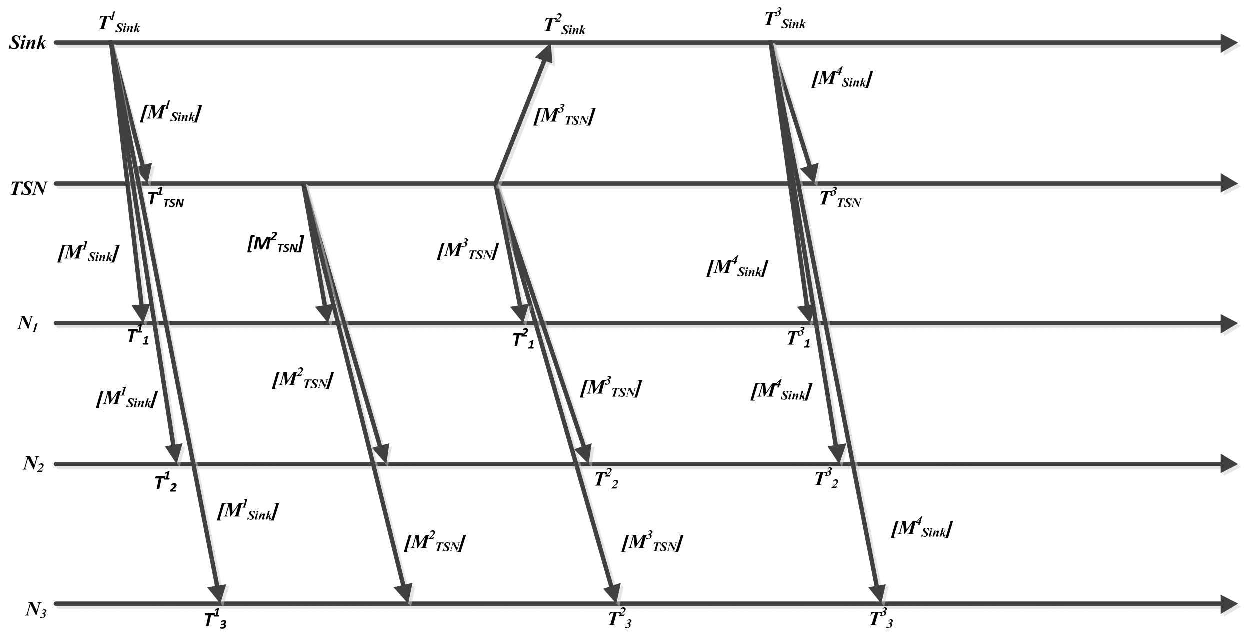

Figure 4. Clock synchronization model of ATS protocol for cluster 0.

Algorithm 3. Intra Cluster Head Synchronization Algorithm

```
1.  Cluster head(CH) broadcast a message $M_{CH}^1$ with timestamp $T_{CH}^1$ and ID
    ($M_{CH}^1$=[Sync_message, $T_{CH}^1$, $ID_{TSN}$])
2.  Nodes on receiving $M_{CH}^1$, record their own clock time $T_i^1$ (TSN records $T_{TSN}^1$) where [$i$=1 to $N$-2] and
    also record$T_{CH}^1$.
3.  TSN broadcast a message $M_{TSN}^2$
    ($M_{TSN}^2$ = [Sync_message, $T_{TSN}^1$])
4.  Nodes upon receiving $M_{TSN}^2$ calculate offset ($\delta_i'$)
    $\delta_i'$= |$T_i^1$ - $T_{TSN}^1$|.
5.  Nodes adjust clock according to $\delta_i'$
    If $T_i^1$ - $T_{TSN}^1$>0
      then $N_i$ set its logical clock H(t) = $C(t)$ - $\delta_i'$
    Else
      H(t) = $C(t)$ + $\delta_i'$
6.  TSN broadcast a message $M_{TSN}^3$.
7.  All nodes and cluster head receive $M_{TSN}^3$ and note their clock time $T_i^2$ and $T_{CH}^2$
8.  CH broadcast message $M_{CH}^4$ ($M_{CH}^4$= [$T_{CH}^2$,$T_{CH}^3$]).
    Where $T_{CH}^3$ is the time stamp of CH.
9.  Nodes within cluster upon receiving this message, note their own clock time $T_i^3$ and calculate its clock
    skew $\rho_i$ with CH as follows
    $\Delta t_{CH} = T_{CH}^3 - T_{CH}^1$
    $\Delta t_i = T_i^3 - T_i^1$ where $i$= 1 to $N$-1
    $\rho_i = (\Delta t_{CH} - \Delta t_i)/\ \Delta t_i$
10. Nodes calculate offset $\delta_i$with respect to CH
    $\delta_i = |T_{CH}^2 - T_i^2|$ where $i$= 1 to $N$-1
```

```
        If $(T_{CH}^2 - T_i^2) > 0$,
          then nodes set $H(t) = H(t) + \delta_i$
        Else
          $H(t) = H(t) - \delta_{CH}$
11.  Nodes will set logical clock accordingly as
        $H(t+\Delta t)=H(t)+(1\pm\rho)\Delta t$.
12.  Nodes $N_i$ transmit $\rho_i$ to cluster head.
13.  CH calculates average skew as
```

$$\rho_{Avg} = \sum_{i=0}^{N-1} \left(\frac{\rho_i}{N-1}\right)$$

5. Simulation results

We consider a sensor network composed of SNs deployed uniformly. The nodes are divided into clusters according to energy-efficient multi-level clustering algorithm (EEMC) [23]. The parameters used in simulation for evaluating the performance of ATS protocol are given in table 2. The number of neighbors and node density varies with experiments. The initial energy per node is assumed to be 2J. The sink is also assumed to be a cluster head. Sink and cluster heads communicate through gate nodes. Simulation area for most of the experiments is same but changes for experiment where the number of nodes increases while node density is constant. Simulation was performed using OMNeT++.

Parameter	**Value**
Simulation area	10 x 10/Variable
Number of nodes(*N*)	Fixed/Variable
Initial energy per node	2J
Data transmission rate	250 k bits/second
$E_{i,RX}$	50 nJ/bit
$E_{i,Proocess}$	20 nJ/bit
$E_{i,TX}$	50 nJ/bit
Node density	Fixed/variable
Stop error for Algorithm	0.01 percent and 0.1 percent
Transmission range	1m

Table 2. Parameters and Values.

Fig.5 shows a graph of phase offset of nodes within a cluster before synchronization. The figure shows that before synchronization there is a difference in the clocks of nodes with respect to TSN within a cluster. The phase offset can be due to the drift in the clocks of different nodes or because initially the nodes are not synchronized. Phase offset is reduced after the synchronization of the nodes. Fig.6 shows the phase offset within the cluster with respect to cluster head in first round when nodes have been synchronized with respect to TSN and not with CH. There is no significant difference in the phase offset of nodes within a cluster after round 1, it is because ATS has been designed to synchronize the nodes on the time stamps it receives and is not based on calculation of the delay. Fig.6 shows that the phase offset is very less in round 1 when the nodes have been synchronized with TSN. ATS is highly accurate as fewer rounds will be required for synchronization hence ATS is energy efficient.

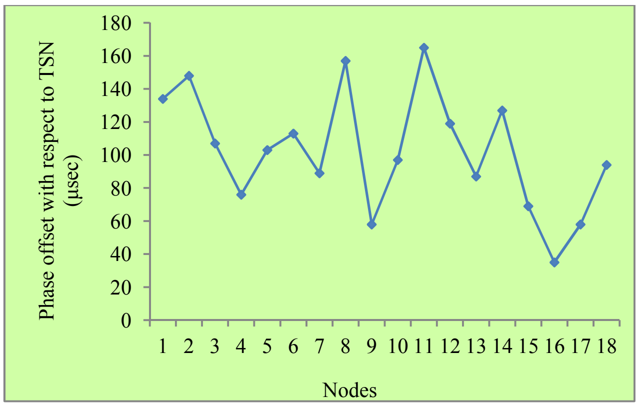

Figure 5. Phase offset of nodes with TSN within a cluster before synchronization.

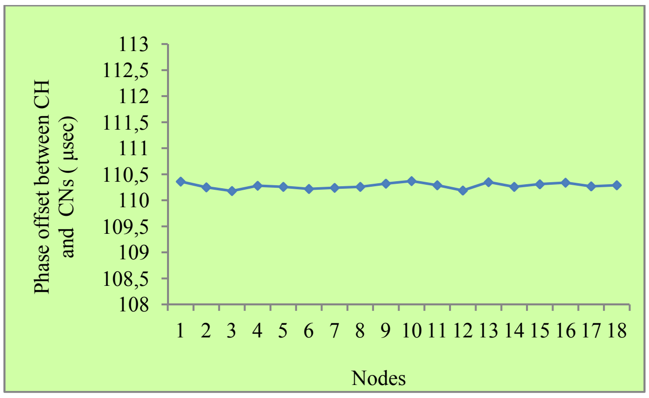

Figure 6. Phase offset of nodes within a cluster w.r.t. CH after synchronization with TSN (after round 1).

The relative error with the number of rounds executed by the SNs is shown in Fig.7. The cluster size is taken to be of twenty (20) nodes and the total number of nodes is taken to be 200. The simulation area is 10x10 and transmission range is 1 meter. The stop error for the algorithm is 0.01 percent. The result is similar as of averaging algorithm. As seen from the graph the error rate decreases as the number of rounds increase. The decrease in the error rate is exponential. This means that as the number of rounds increase the error rate reduces significantly. Simulation results show that ATS outperforms the other protocols in terms of the relative error. ATS is more accurate as compared to other protocols and will require less synchronization rounds to achieve given stop error.

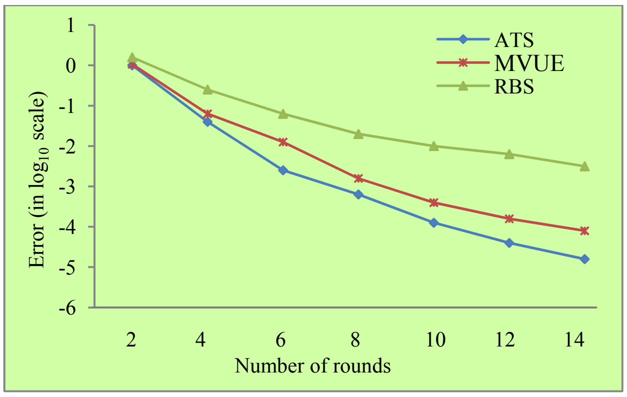

Figure 7. Relative error with the number of rounds.

The comparison of synchronization overheads in terms of number of synchronization messages required per round for synchronizing the network is shown in Fig.8. The results are obtained by keeping the node density constant. As the number of nodes increase then the number of clusters also increase. The number of synchronization messages required for RBS and MVUE are more as compared to our proposed protocol. ATS requires less synchronization messages. Figure 8 shows that as the number of nodes increase there is a substantial increase in the synchronization messages required by RBS protocol. There is an increase in the synchronization messages required per round for ATS protocol as the nodes increase, though the increase is very less as compared to other protocols. This is because of the reason that ATS broadcasts synchronization message and all the nodes synchronize exchanging fixed number of messages as discussed in section 4.

Average energy consumption per round for variable number of nodes is shown in Fig. 9. The graph is plotted for fixed density of nodes. Fig.9 shows that as the nodes in a cluster increase the average energy consumed per node per round increases for all protocols. The energy consumption in RBS and MUVE protocols is more as compared to ATS. This is because fixed message broadcasts are required for the synchronization of the nodes for ATS protocol. ATS has very less effect of increase in the node density on per node energy consumption.

Fig.10 shows the convergence speed of ATS protocol for variable number of nodes. The graph is plotted for fixed node density. The stop error for the algorithm is 0.1 percent. The graph is plotted for the number of rounds to achieve stop error of 0.1percent. It can be seen from the figure that less number of rounds are required if the nodes are less hence the convergence speed is more. The convergence speed is more for less number of nodes at fixed node density because fewer clusters are formed.

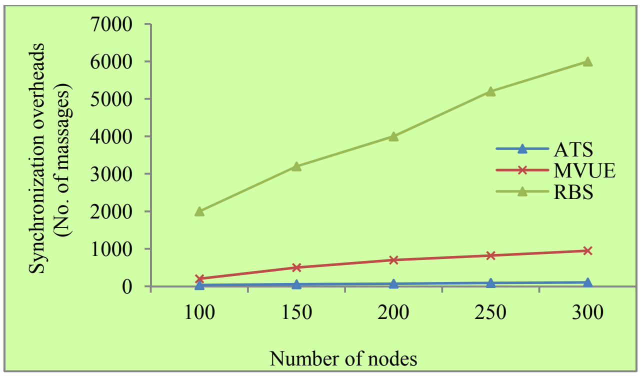

Figure 8. Synchronization overheads per round.

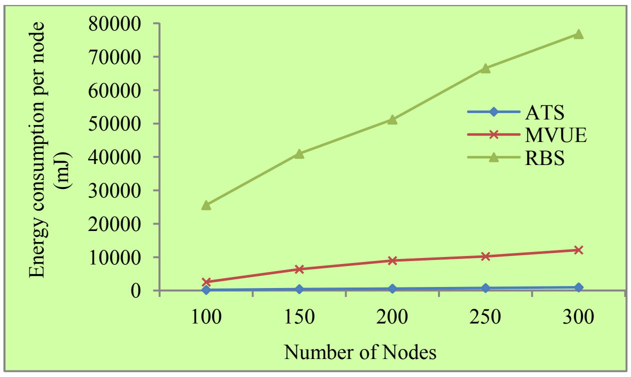

Figure 9. Average energy consumption per round.

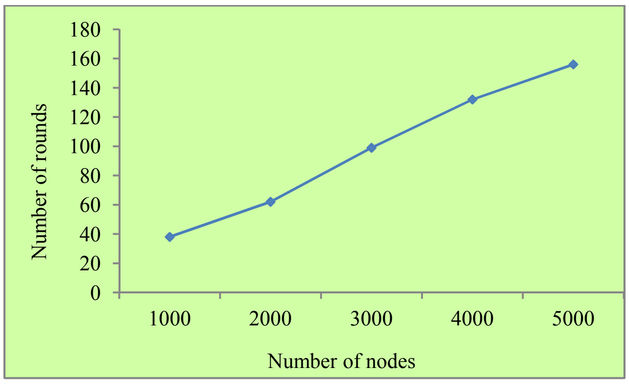

Figure 10. Convergence speed of our proposed protocol

6. Conclusion and future work

Proposed adaptive time synchronization protocol (ATS) makes use of TSN to synchronize the nodes within a cluster. The simulation results show that proposed protocol has less synchronization message overheads as compared to other protocols thus energy consumption in synchronization of nodes is also less. Since proposed protocol uses broadcasts for synchronization hence it's per node energy consumption is not much affected by the node density. ATS requires less synchronization rounds due to high accuracy thereby saving energy. The proposed protocol is suitable for industrial applications and can be used for applications where synchronization time must be less.

In future the proposed protocol can be extended and its performance can be evaluated for nodes using dual radio modes (high and low power transmitters). The main ideas presented in this paper could be fully or partially applied to improve the performance of existing protocols or for the design of new energy efficient solutions.

7. References

[1] A. Dobra, M. Garofalakis, J. Gehrke, and R. Rastogi, "Processing Complex Aggregate Queries over Data Streams," in *Proceedings of SIGMOD International Conference on Data Management,* 2002, pp. 61–72.

[2] S. Madden, R. Szewczyk, M. Franklin, and D. Culler, "Supporting Aggregate Queries over Ad-Hoc WSNs," in *Proceedings of Workshop on Mobile Computing Systems and Applications,* 2002, pp. 49–58.

[3] A. Woo, S. Madden, and R. Govindan, "Networking Support for Query Processing in Sensor Networks," *Communications of the ACM,* vol. 47, pp. 47–52, 2004.

[4] J. Elson, L. Girod, and D. Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts," in *Proceedings of Fifth Symposium on Operating Systems Design and Implementation,* 2002, vol. 36, pp. 147–163.

[5] K. Romer," Time Synchronization in Ad Hoc Networks," in *Proceedings of ACM Symposium on Mobile Ad Hoc Networking and Computing*, 2001, pp. 173–182.

[6] M. Mock, R. Frings, E. Nett, and S. Trikaliotis, "Continuous Clock Synchronization in Wireless Real-Time Applications," in *Proceedings of 19th IEEE Symposium on Reliable Distributed Systems,* 2000, pp. 125–133.

[7] S. Ganeriwal, R. Kumar, S. Adlakha, and M. Srivastava, "Network-Wide Time Synchronization in Sensor Networks," *Technical report of Networked and Embedded Systems Lab, Electronic Engineering Department, UCLA,* 2002.

[8] S. Ping, "Delay Measurement Time Synchronization for WSNs," *Intel Research,* vol.59, pp. 2963–29, 2003.

[9] S. Palchaudhuri, A. Saha, and D.B. Johnson, "Probabilistic Clock Synchronization Service in Sensor Networks," *IEEE Transactions on Networking,* vol. 2, no.2, pp. 177-189, 2003.

[10] S. Ganeriwal, R. Kumar and M.B. Srivastava, "Timing-Sync Protocol for Sensor Networks," in *Proceedings* of *First International conference on Embedded Networked Sensor Systems,* 2003.

[11] Jana Van Greunen and Jan Rabaeay, "Lightweight Time Synchronization for Sensor Networks," in *Proceedings of Second ACM International Workshop on Wireless Sensor Networks and Applications in conjunction with ACM MobiCom,* 2003.

[12] Q. Li, and D. Rus, "Global Clock Synchronization in Sensor Networks, "*IEEE Transactions on Networking,* vol.55, no.2, pp. 214-225, 2006.

[13] W. Su, and I. Akyildiz, "Time-Diffusion Synchronization Protocols for Sensor Networks," *IEEE/ACM Transactions on Networking,* vol. 13, no. 2, pp. 384-397, 2005.

[14] R. Subramanian, and I.D. Scherson, "An Analysis of Diffusive Load-Balancing," in *Proceedings of Sixth Annual ACM Symposium on Parallel Algorithms and Architectures*, 1994, pp. 220-225.

[15] Dong Shao-Long, and Xing Tao, "Cluster-Based Power Efficient Time Synchronization in Wireless Sensor Networks," in *Proceedings of IEEE International Conference on Electro/Information Technology*, 2006, pp. 147-151.

[16] Ilkay Sari, Erchin Serpedin, Kyoung-Lae Noh, Qasim Chaudhari, and Bruce Suter, "On the Joint Synchronization of Clock Offset and Skew in RBS-Protocol," *IEEE Transactions on communications*, vol. 56, no.5, pp.700–703, 2008.

[17] Mei Leng, and Yik-Chung Wu, "On Clock Synchronization Algorithms for Wireless Sensor Networks under Unknown Delay," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 1, pp. 182-190, 2010.

[18] Roxana Albu, Yann Labit, Thierry Gayraud, and Pascal Berthou, "An Energy-Efficient Clock Synchronization Protocol for Wireless Sensor Networks," *Wireless Days, IFIP*, pp.1-5, 2010.

[19] B-Kug Kim, Sung-Hwa Hong, Kyeong Hur, and Doo-Seop Eom, "Energy-Efficient and Rapid Time Synchronization for Wireless Sensor Networks," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 4, 2010

[20] J. Chen, Qing Yu, Yan Zhang, Hsiao-Hwa Chen, and Youxian Sun,"Feedback-Based Clock Synchronization in Wireless Sensor Networks: A Control Theoretic Approach," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 6, 2010.

[21] Thomas Schmid, Zainul Charbiwala, Roy Shea, and Mani B. Srivastava, "Temperature Compensated Time Synchronization," *IEEE Embedded Systems Letters*, vol. 1, no. 2, 2009.

[22] Qasim M. Chaudhari, Erchin Serpedin, and Khalid Qaraqe, "On Minimum Variance Unbiased Estimation of Clock Offset in a Two-Way Message Exchange Mechanism," *IEEE Transactions on Information Theory*, vol. 56, no. 6, 2010.