E-SATS: An Efficient and Simple Time Synchronization Protocol for Cluster-based Wireless Sensor Networks

G. S. S. Chalapathi, *Graduate Student Member, IEEE,* Vinay Chamola, S. Gurunarayanan, *Member, IEEE* and Biplab Sikdar, *Senior Member, IEEE*

***Abstract*—Over the past decade, the Internet of Things (IoT) has attracted enormous interest from the research community and industry. IoT requires a synergy of various technologies, and Wireless Sensor Networks (WSNs) are poised to play a critical role in many IoT applications like weather monitoring, smart-grid, smart-city, etc. Synchronization of local clocks of the WSN nodes is essential in many network functionalities and thus a time synchronization protocol is required in WSNs. Although several synchronization protocols have been proposed for WSNs, most of them are simulation-based works. They make many assumptions at a high abstraction level and do not take into account the conditions of the Line-of-Sight (LOS) in the network. These factors significantly affect the performance of these protocols. Thus, conclusive experimental proof of the effectiveness of these protocols for different LOS conditions is required. In this direction, this work proposes a time synchronization protocol called *Efficient and Simple Algorithm for Time Synchronization* (E-SATS) for a cluster-based WSN. E-SATS has been tested on a large-sized WSN testbed in different LOS scenarios in this work and compared with the existing state-of-the-art protocols. E-SATS outperformed existing protocols by achieving up to 6 times better accuracy as compared to existing protocols with significantly lesser computations and energy consumption.**

***Index Terms*—Time synchronization protocol; Wireless Sensor Networks; Cluster-based WSN; WSN testbed; Line-of-sight (LOS) conditions; Non-line-of-sight (NLOS) condition**

I. Introduction

The Internet of Things (IoT) paradigm connects different objects over the Internet and allows humans to interact with them by bringing together the physical realm and the virtual environments. IoT is made functional by a combination of many underlying technologies such as sensors, communication networks, Application Program Interfaces (APIs), back-end servers with data analytics, remote data/service access technologies, etc. Hence, WSNs have become an important part of the IoT. WSNs are used in several applications such as home automation [2], precision agriculture [3], [4], personal health monitoring [5], environmental monitoring [6], [7], asset tracking [8], etc. Some of the features of WSN which make them suitable for IoT applications are low cost (for large scale deployment of devices), bidirectional communication (for sensing and actuation) and low bandwidth requirements [9].

A WSN is a network of several devices which are generally referred to as nodes. The WSN nodes cooperatively perform the sensing, monitoring and actuation tasks in a given location. Most of the WSN nodes in the network rely on batteries (or solar cells) for their power supply requirements. Thus, a WSN is typically an energy-constrained network deployed in harsh environments like industries [10], mines [11], volcanoes [12], etc. Therefore, WSN nodes are designed to be robust and survive without battery replacement for few months and even few years. A WSN is an example of a distributed network where the nodes run in a coordinated way carrying out a sensing and monitoring tasks. Thus, it is very important for the participating nodes to have a common notion of time which is achieved through a time synchronization protocol. WSN nodes use cheap crystal oscillators for running the local clock at each node. These cheap crystal oscillators bring inaccuracies in the local time due to crystal aging, environmental factors like temperature, humidity, etc. [13]. Therefore, a time synchronization protocol is essential to synchronize the local clocks of the WSN nodes.

Several synchronization protocols for WSN have been proposed till date like TPSN [15], RBS [16], TDP [17], GTSP [18], CCTS [19], CMTS [20], etc. However, many of them have been demonstrated on simulators only. Simulation-based demonstrations have limitations like neglecting or approximating the delays in the communication of the packets, not accounting for the packet loss in the communication, making assumptions at high abstraction level, etc. [21]. Thus, they cannot fully prove the merit of these synchronization schemes for practical deployment. Also, the fact whether the nodes are Line-of-Sight (LOS) to each other or not significantly affects the communication between any two nodes, which in turn affects the synchronization protocol's performance. However, seldom do the synchronization protocols mention about the LOS conditions considered while evaluating their protocols or consider the effect of the environment on their synchronization protocol. Therefore, a simple time synchronization protocol called *Efficient and Simple Algorithm for Time Synchronization (E-SATS)* for a cluster-based WSN is proposed in this work. E-SATS is able to achieve micro-second level synchronization which is very essential for many automotive, health and industrial applications [22]. Further, E-SATS uses simple and fewer computations while consuming lesser energy compared to

Corresponding Author: Biplab Sikdar. G. S. S. Chalapathi, Vinay Chamola and S. Gurunarayanan are with Department of EEE, Birla Institute of Technology & Science (BITS), Pilani, Rajasthan India 333031. (email:{gssc,vinay.chamola,sguru}@pilani.bits-pilani.ac.in) and Biplab Sikdar is with Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117583 (email:bsikdar@nus.edu.sg).

the state-of-the-art synchronization protocols. Thus it is aptly suited for resource-constrained WSN nodes. The performance of E-SATS has also been tested on a large size WSN testbed of 30 nodes in different LOS conditions to prove its merit in realistic conditions.

This paper makes the following contributions:

1) A simple and efficient synchronization protocol for a cluster-based WSN named E-SATS is presented.
2) E-SATS is shown to achieve micro-second level synchronization on a hardware testbed.
3) The effect of LOS conditions on the accuracy of synchronization protocol is also presented.
4) E-SATS is also shown to have significantly better synchronization accuracy, lower computational complexity and lesser energy consumption compared to the existing synchronization protocols used in cluster-based WSNs.

The organization of this paper is as follows. Section II gives a brief background of time synchronization in WSNs. Section III discusses the existing synchronization protocols related to this work. The network model, clock model and the mathematical basis of E-SATS is discussed Section IV. Section V describes the testbed and the methodology used for the evaluation of E-SATS. Section VI presents the results and analysis of the performance analysis carried out on the testbed in different LOS conditions. This section also presents a comparison of computational complexity and energy consumption of E-SATS with some of the existing synchronization protocols. Finally, conclusions are presented in Section VII.

II. Background of Time Synchronization in WSN

Time synchronization plays an important role in many operations of a WSN. For example, in sensor data monitoring applications, the sensor data reported at different locations has to be accompanied by the timestamp at which the data was recorded. During analysis of the sensor data, it is essential for this timing information to be according to a clock which is common to all the participating nodes. If this synchronization is absent, it will lead to incorrect analysis of the sensor data. Apart from this data fusion and data-gathering operation, time synchronization is also essential for duty-cycled packet communication, for synchronizing the sleep-wake patterns among the nodes, for time-based localization protocols, etc.

The following are the different types of synchronization in WSN nodes according to the degree of synchronization that is required:

I) All the nodes of the network are synchronized to a common clock.
II) Only a part of the network, i.e., only some of the nodes are synchronized.
III) A chronology of different events at various locations of the network is determined without determining the actual time of occurrence of the events.

The degree of synchronization needed in a WSN depends on the application in which it is used.

A time synchronization protocol in a WSN involves exchange of packets among the participating nodes. These packets contain timing information like the local clock of the sender, reception time of the previous packets, etc. There are different delays which a packet experiences in the course of this transmission-reception operation. These delays can be categorized as deterministic delays and non-deterministic delays. These delays are elaborately discussed in [23] and [13]. A synchronization protocol has to take into account these delays as they directly influence the accuracy of the synchronization achieved.

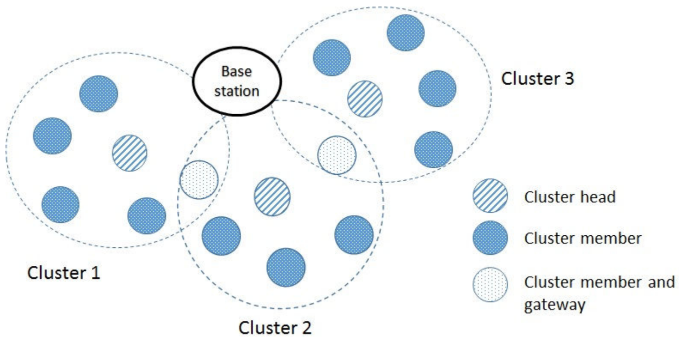


Fig. 1: A typical network considered.

As mentioned before, the local clock of a node is maintained using a crystal oscillator. There are two common models used to model the local clock of a node. The first model is the offset only model and the second one is the skew offset model. In offset-only model, the local time at two nodes are assumed to differ each other by a constant (called the offset). This model does not account for the fact that oscillators on two nodes tick at a different rate in reality. Some of the works based on offset-only model are [15], [24], [25]. Thus this model is less-accurate than the clock-offset model which accounts for this fact. Clock-offset model based time synchronization protocols are more accurate and required less frequent re-synchronizations [21]. Some of the works which are based on clock-offset model are [16], [26], [27], [30]. These models are elaborately explained in [21].

III. Related work

This work focuses on time synchronization in WSNs with a cluster-based topology. It has been shown in works like [1], [35], [36] that cluster-based topology is a very energy-efficient topology for WSNs, particularly for operations like data-gathering, data dissemination, route-formation, etc. which are the most common operations in typical WSN deployments. Thus, it is pertinent to design time synchronization protocol specific to cluster-based WSNs. A typical cluster-based WSN is shown in Figure 1. A cluster consists of a cluster head and a few cluster members. There are a few cluster members called the gateway nodes which are cluster members of more than one cluster. These gateway nodes help in inter-cluster communication. Except for the gateway nodes, a cluster member communicates only to its cluster head via a unicast communication.

There are many time synchronization protocols (TSPs) for cluster-based WSNs like PC-Avg [37], CHTS [38], CCSN [39], L-SYNC [33], SLTP [34], CCTS [19] and most recently CMTS [20]. PC-Avg uses average of the local time of the cluster members to synchronize the cluster members. This protocol exhibits a high synchronization error and also does not consider the non-deterministic and deterministic delays involved in communication. It also requires frequent resynchronizations as it is based on the offset-only model.

CHTS [38] is one of the earliest reported TSPs for cluster-based WSNs. It assumes the presence of few nodes in the network with high performance oscillator nodes (HPNs) along with rest of the nodes which are low performance oscillator nodes (LPNs). The HPNs become the cluster heads and the LPNs become cluster members. This protocol has a significant overhead as it involves cluster head tree formation and cluster member tree formation phases. The cluster heads first get synchronized to the reference node (which provides a time reference to the whole network) in a hierarchical manner and then the cluster heads synchronize their cluster members again in a hierarchical manner. Since CHTS uses a offset-only clock model and does not account for non-deterministic and deterministic delays, it shows poor synchronization accuracy and requires frequent resynchronization. Also, it has high overhead due to cluster head tree formation and cluster member tree formation phases. Also, it requires specialized HPNs thereby increasing the cost of deploying the network.

CCSN [39] is another hierarchical synchronization protocol. It also performs hierarchy of cluster heads like CHTS and uses a synchronization scheme like CHTS. The main difference between CCSN and CHTS is that CCSN uses a reference broadcast method among the cluster members to synchronize the cluster members to the cluster head. Since CCSN also uses an offset-only clock model and does not capture the deterministic and non-deterministic delays, it exhibits a very high synchronization error of the order of milliseconds.

SLTP [34] and L-SYNC [33] both use a similar synchronization procedure. They however differ in the cluster formation methods. In both of these protocols, the cluster members exchange packets with their cluster heads and record the transmission and reception timestamps of these packets. The cluster members then perform a linear regression on these timestamps to synchronize themselves to their respective cluster heads. Both SLTP and L-SYNC simulate their performance over large-sized networks. However, they exhibit very high synchronization error of the order of milliseconds. Like PC-Avg, both SLTP and L-SYNC do not take into consideration the non-deterministic and deterministic delays in the packet exchanges.

CCTS [19] is a recent consensus-based time synchronization protocol. CCTS uses two different virtual clocks, i.e., intra-cluster virtual clock and inter-cluster virtual clock. The cluster members and cluster head maintain an intra-cluster virtual clock while the cluster heads maintain the inter-cluster virtual clock. The virtual clock at a node (either the cluster member or cluster head) is related to the local time by the skew compensation parameter and offset compensation parameter. The intra-cluster virtual time C_{vj} of a node j is given by

$$C_{vj} = \alpha_{vj} C_k(t) + \beta_{vj}, \tag{1}$$

where $C_j(t)$ is the current local time of the node and α_{vj} and β_{vj} are the skew and offset compensation parameters of the intra-cluster virtual clock. The cluster head first synchronizes the skew of the intra-cluster virtual clock and then it synchronizes the offset of the intra-cluster virtual clock. Then the cluster heads of all the clusters exchange packets among themselves using the gateway nodes to synchronize the inter-cluster virtual clock. CCTS has a very slow convergence and requires a large number of iterations [20]. The reason for this slow convergence is attributed to the fact that CCTS first synchronizes the skew (in both the virtual clock synchronization phases) and then synchronizes the offset [20]. A large number of iterations (as seen in Figure 9 and Figure 6 of [19]) are used by CCTS to achieve microsecond accurate synchronization which cannot be afforded by energy-constrained WSN nodes. Also, CCTS does not capture the deterministic and non-deterministic delays that occur in the communication of packets and thus it does not give good synchronization accuracy in practical WSNs (which we will prove in Section VI of this paper).

CMTS [20] is another recent consensus-based synchronization protocol for cluster-based WSNs. CMTS also has two different synchronization phases, i.e., intra-cluster and inter-cluster. However, unlike CCTS, CMTS maintains a single virtual clock at each node. CMTS uses the maximum value of skew and offset compensation parameters of the virtual clock of all the nodes in a cluster instead of the average value (as used in CCTS) to achieve a consensus. By using maximum value, it achieves faster convergence compared to CCTS. An improved version of CMTS called 'Revised-CMTS' is also presented in [20] which considers the communication delays. Revised-CMTS assumes that the communication delays have an upper-bound and achieves simultaneous synchronization of skew and offset of the virtual clocks. Though CMTS and Revised-CMTS have faster convergence than CCTS, they need a large number of broadcasts in inter-cluster synchronization (Fig. 8 and Fig. 9 of [20]). It needs around 2000 broadcasts for a 50 node network (Fig. 9 of [20]) which is extremely energy draining for energy-constrained WSN nodes.

A common drawback of all the above-mentioned synchronization protocols is that they are all simulation-based works, i.e., they have not been tested on real WSN platform. Simulation-based works do not give a complete and accurate understanding of the synchronization protocol in practical WSN deployment as mentioned before. Also, these works have not taken into account the LOS conditions among the nodes in practical deployments. The performance of a synchronization protocol is greatly affected by the LOS conditions in which it operates. Therefore, E-SATS which is presented in this work presents a simple yet accurate time synchronization protocol tested on a WSN testbed in different LOS conditions. The preliminary version of E-SATS is presented in [30]. The present work extends and enhances [30] in the following ways:

1) The synchronization algorithm in a cluster has been made more efficient by reducing the number of messages used in the synchronization process and by employing a specific message exchange schedule.
2) The mathematical model along with the cluster formation phase used has been presented in detail.
3) The synchronization protocol has been tested in different

LOS conditions and a comparison with existing synchronization protocols in terms of synchronization error, computational complexity and energy consumption has been presented.
4) An analysis of the effect of LOS conditions on the synchronization protocol has also been presented.

IV. Efficient and Simple Algorithm for Time Synchronization (E-SATS)

Before describing the E-SATS algorithm, we first describe the clock model and the network considered in this work.

A. Clock Model and the Network Considered

The network considered in this work is shown in Figure 1. The WSN nodes in the network are organized into clusters each of which has a cluster head and few cluster members. There are gateway nodes as explained before which are part of more than one cluster and help in communication between its cluster heads. The cluster heads are connected to a node called the "Base Station" or the "sink" node. This Base Station node acts as an interface between the WSN and other networks like the Internet when the WSN is used in IoT applications.

We consider a static network in E-SATS, i.e., the nodes in the network are stationary. Also, the links between any two nodes are symmetric links, i.e., if node p can communicate with node q, then q can also communicate with p. Some other works like [26], [27], [19] have also made this assumption that links in WSN are symmetric. The nodes which have higher computation and memory capabilities are identified as the cluster heads.

Let there be n nodes in a network and the set $\mathcal{N}$ represent all those n nodes. Among the nodes in the network, nodes having higher computational capabilities and memory availability are chosen as cluster heads. If such nodes are unavailable, we choose some of the nodes as cluster heads and employ a cluster head rotation mechanism by which each member of a cluster becomes a cluster head in each synchronization phase. By employing cluster head rotation method, we can avoid the scenario where a cluster head drains its battery energy because of becoming a cluster head for prolonged duration. The cluster heads in the network are represented by $\mathcal{D} \triangleq \{d_1, d_2, ..., d_h\}$ where h is the total number of cluster heads in the network. Let the total number of cluster members in the whole network be u. Therefore $n = h + u$. The set $\mathcal{T}_i$ represents the cluster members of the cluster head d_i, i.e.,

$$\mathcal{T}_i = \{m \mid m \text{ is a cluster member of } d_i, \text{ where } d_i \in \mathcal{D}\}. \tag{2}$$

The number of cluster members of d_i is given by M_i. In the rest of the paper, we denote the jth cluster member of cluster head d_i as m_{ij}.

The skew-offset model mentioned in Section III is considered in E-SATS. Each cluster member is synchronized to its cluster head. The local time of a cluster member m_{ij} with respect to its cluster head d_i at time t, represented as $C_{ij}(t)$, is given by

$$C_{ij}(t) = \alpha_{ij}\ C_i(t) + \beta_{ij}, \tag{3}$$

TABLE I: Notation Summary

Notation	Meaning
$\mathcal{N}$	Set representing all nodes in the network
n	total number of nodes in the network
$\mathcal{D}$	Set representing all the cluster heads
h	Total number of cluster heads in the network
u	Total number of cluster members in the network
i	index of the cluster head
d_i	ith cluster head
$\mathcal{T}_i$	Set representing all the cluster members of d_i
M_i	Number of cluster members in the ith cluster
j	index of the cluster member
k	index of the iteration of communication between a cluster member and its cluster head
m_{ij}	jth cluster member of ith cluster head (i.e. d_i)
α_{ij}	Relative skew of m_{ij} with respect to d_i
β_{ij}	Relative offset of m_{ij} with respect to d_i
ζ^j_{ik}	deterministic delays in communication between m_{ij} and d_i in the kth iteration
$\eta^j_{ik}, \omega^j_{ik}$	non-deterministic delays in communication between m_{ij} and d_i in the kth iteration
Q	Total number of iterations in Synchronization phase
S	Total number of common event packets sent by *tester node* in Synchronization evaluation phase
r_i	number of overlapping clusters for ith cluster

where $C_i(t)$ is the local time of the cluster head d_i. In the above equation, α_{ij} and β_{ij} are the relative skew and relative offset, respectively, of node m_{ij} with respect to d_i. Table I summarizes the notations used in this paper.

In E-SATS, there is a cluster formation phase when the network is initialized. This phase is followed by a synchronization phase.

B. Cluster Formation Phase

The cluster heads initiate a cluster formation phase. A cluster head broadcasts a *CM_assignment* packet containing its *nodeID*. Note that the cluster head can adjust the transmit power level according to the cluster area and the cluster size that is desired. When the nodes (which are not cluster heads) hear this packet, they identify themselves to be the cluster member of this cluster head and store the *nodeID* of the cluster head. If a cluster member hears the *CM_assignment* packets of more than one cluster head, it becomes a cluster member of all these cluster heads and identifies itself to be a gateway node. Further, the gateway cluster member unicasts the information about the *nodeID* of the cluster heads it is associated with to all its cluster heads. This helps a cluster head in identifying the gateway node to the neighboring cluster heads. Since the nodes are stationary, this cluster formation phase is performed only while initializing the network. However, a newly joining node

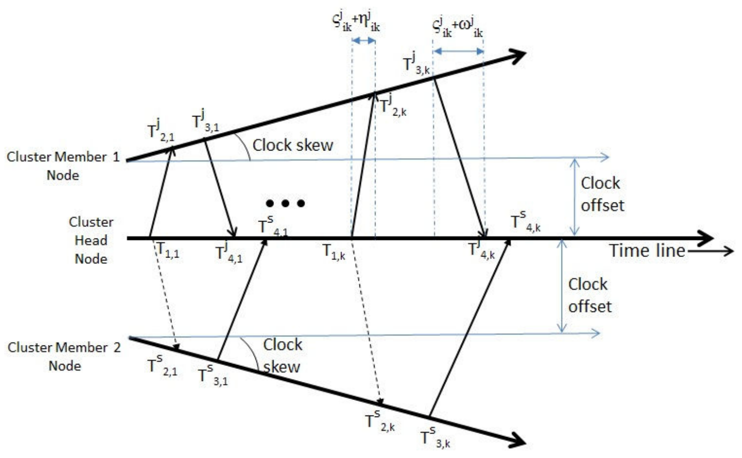


Fig. 2: Synchronization message exchanges between a cluster head and two cluster member nodes m_{ij} and m_{is}.

can initiate a cluster discovery by sending a *CH_Discovery* packet. A cluster head which hears this packet, responds to this node by unicasting its acknowledgment to allow this node to join its cluster. If the newly joining node receives an acknowledgment from more than one cluster head, it will become a gateway node and it sends this information to all its cluster heads.

C. Synchronization Phase

The cluster head (say d_i) then synchronizes the cluster members by sending a *Synch_msg* packet at time $T_{1,1}$ as per its local clock. This packet is broadcasted to all its cluster members and it contains the timestamp $T_{1,1}$. On receiving this packet, the cluster members record the reception time. For example, cluster member m_{ij} will record the reception time as $T^j_{2,1}$. All the cluster members then respond to this *Synch_msg* by sending an acknowledgment packet at $T^j_{3,1}$ after backing off for a pre-determined amount of time. Note that this back-off time is different for each cluster member so that collisions of the acknowledgment packets sent by them to d_i can be avoided. The acknowledgment sent by the cluster member contains $T_{1,1}$, $T^j_{2,1}$, $T^j_{3,1}$. Also, the back-off times observed by each cluster member is chosen in such a way that the duration of each iteration is kept small. Also, the cluster head knows the back-off time observed by each cluster member. The cluster head d_i receives this packet at $T^j_{4,1}$. This iteration of cluster head sending the *Synch_msg* and cluster member sending an acknowledgment is repeated Q number of times. Each iteration collects four-time stamps for each cluster member m_{ij}, i.e., $T_{1,k}$, $T^j_{2,k}$, $T^j_{3,k}$ and $T^j_{4,k}$ (where the subscript k represents the iteration index). Note that the time stamp $T_{1,k}$ does not have the superscript j because it is common for all the cluster members. The message exchanges for two cluster members m_{ij} and m_{is} with their cluster head are depicted in Figure 2. The superscript in the timestamps specifies the node to which the timestamp corresponds to. Thus, in E-SATS, in the ith cluster with cluster head d_i a total of M_i+1 packets are required in each iteration (1 broadcast packet and M_i acknowledgment packets from the cluster members). In contrast, SATS (the preliminary version of E-SATS) which was proposed in [30] needs $2M_i$ packets in each iteration. This is because in SATS, a cluster head unicasts the *Synch_msg* and synchronizes the cluster members one-by-one. Thus, E-SATS is more energy efficient than SATS. The timestamps collected for an iteration k of cluster member m_{ij} are related to each other as follows:

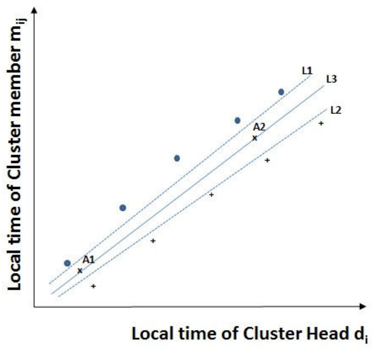


Fig. 3: A diagram to illustrate the approximate estimate used to calculate relative skew and and offset.

$$T^j_{2,k} = \alpha_{ij}(T_{1,k} + \zeta^j_{ik} + \eta^j_{ik}) + \beta_{ij}, \tag{4}$$

$$T^j_{3,k} = \alpha_{ij}(T^j_{4,k} - \zeta^j_{ik} - \omega^j_{ik}) + \beta_{ij}. \tag{5}$$

In the above equations, ζ^j_{ik} represents the deterministic delay and, η^j_{ik} and ω^j_{ik} represent the non-deterministic delays measured by local clock of d_i. Equations (4) and (5) can be written as follows:

$$T^j_{2,k} = \alpha_{ij}T_{1,k} + \beta_{ij} + \alpha_{ij}(\zeta^j_{ik} + \eta^j_{ik}), \tag{6}$$

$$T^j_{3,k} = \alpha_{ij}T^j_{4,k} + \beta_{ij} - \alpha_{ij}(\zeta^j_{ik} + \omega^j_{ik}). \tag{7}$$

Let us consider the case where the timestamps of cluster member m_{ij} and the timestamps of cluster head d_i are plotted in Cartesian plane as shown in Figure 3. The Y-axis represents the local time at m_{ij} and the X-axis represents local time at d_i. From (6), it can be seen that since α_{ij}, ζ^j_{ik} and η^j_{ik} are positive quantities, the line $L1 = \alpha_{ij}T_{1,k} + \beta_{ij}$ lies below the points $(T_{1,k}, T^j_{2,k})$. Also it can be observed from (7) that the line $L2 = \alpha_{ij}T^j_{4,k} + \beta_{ij}$ will lie above the points $(T^j_{3,k}, T^j_{4,k})$.

So a good approximate estimate of the relative skew and offset of m_{ij} with respect to d_i can be obtained by fitting a line (say L3) that passes in between L1 and L2. To obtain this line L3, we identify two points A1 and A2 such that

$$A1 = \{0.5(T^j_{4,b} + T_{1,b}),\ 0.5(T^j_{2,b} + T^j_{3,b})\}, \quad (8)$$

$$A2 = \{0.5(T^j_{4,a} + T_{1,a}),\ 0.5(T^j_{2,a} + T^j_{3,a})\}, \quad (9)$$

where $b = arg \min_{1 \le k \le Q} (T^j_{4,k} - T_{1,k} - \text{back-off time of } m_{ij})$ and $a = arg \min_{1 \le k \le Q, k \ne b} (T^j_{4,k} - T_{1,k} - \text{back-off time of } m_{ij})$.

The point A1 represents the iteration for which the deterministic and non-deterministic delays are minimum. Similarly, A2 represents the iteration for which these delays are the next minimum. Thus, the line L3 passes through the points A1 and A2. The slope of L3 gives the relative skew and its y-intercept gives the relative offset. So we can obtain the estimated values of relative skew ($\hat{\alpha}_{ij}$) and relative offset ($\hat{\beta}_{ij}$) as

$$\hat{\alpha}_{ij} = \frac{(T^j_{2,b} + T^j_{3,b}) - (T^j_{2,a} + T^j_{3,a})}{(T^j_{4,b} + T_{1,b}) - (T^j_{4,a} + T_{1,a})}, \quad (10)$$

$$\hat{\beta}_{ij} = \frac{(T^j_{2,b} + T^j_{3,b})}{2} - \hat{\alpha}_{ij}\frac{(T^j_{4,b} + T_{1,b})}{2}. \quad (11)$$

Though this gives a simple and approximate method to calculate relative skew and offset, it gives an accuracy comparable to more sophisticated and computationally intensive maximum likelihood estimates as shown in [29].

In E-SATS we use the gateway nodes to do the time translation whenever a packet is sent from one cluster to another. We do not employ a network-wide synchronization when it is not required. Thus, the inter cluster synchronization is performed on a need-to basis and we reduce the overhead involved in network-wide synchronization when it is not needed. Algorithm 1 summarizes the E-SATS algorithm.

Algorithm 1 E-SATS Algorithm

Cluster formation Phase

```
1: Each cluster head $d_i$ where $d_i \in \mathcal{D}$ broadcasts CM_assignment packet along with its nodeID.
2: A node $p$ (where $p \in \mathcal{N}$ and $p \notin \mathcal{D}$) hears this packet and assigns itself as cluster member of $d_i$, i.e., $\mathcal{T}_i = \mathcal{T}_i \cup \{p\}$.
3: If $p$ hears CM_assignment packet of more than one cluster head, $p$ becomes a gateway node and joins the cluster of all those cluster heads .
4: $p$ sends the information of its cluster heads to all $d_l \in \mathcal{D}$ where $p \in \mathcal{T}_l$.
```

Synchronization Phase

```
1: for $i$=1:$h$ do
2:     for $k$=1:$Q$ do
3:         $d_i$ broadcasts a Synch_msg at $T_{1,k}$
4:         for $j$=1:$M_i$ do
5:             Node $m_{ij}$ (where $m_{ij} \in \mathcal{T}_i$) receives it at $T^k_{2,j}$.
6:             Node $m$ responds at time $T^k_{3,j}$ after backing-off for a predetermined time.
7:             $d_i$ receives the packet at $T^k_{4,j}$
8:         end for
9:         $d_i$ waits till it receives the acknowledgment from all cluster members .
10:    end for
11:    for j=1:$M_i$ do
12:        calculate $\hat{\alpha}_{ij}$ and $\hat{\beta}_{ij}$ using (8), (9), (10) and (11)
13:    end for
14: end for
```

V. WSN Testbed and Methodology Used

E-SATS was implemented on a WSN testbed consisting of TelosB WSN nodes [31]. TelosB nodes have MSP-430, a 16-bit microcontroller and CC2420 [32], an IEEE 802.15.4 compliant transceiver. We use an RF frequency of 2.480GHz for communication among the nodes during these experiments. In the first set of experiments, (which we refer to in the rest of the paper as *Scenario-1*), there were a total of 30 WSN nodes used for testing E-SATS and comparing it with other synchronization protocols. These nodes were organized into 6 clusters each with 4 cluster members and 1 cluster head. In the current implementation, all the 30 nodes used in the network are homogeneous, i.e., they have same computational and memory capabilities. Therefore, a homogeneous sensing network is considered in this implementation. A total of 6 nodes were chosen and programmed as cluster heads. Since both the cluster members and cluster heads are battery powered, we can consider a cluster-head rotation scheme so that each node in a cluster can become a cluster head in turns so that a single node is not overloaded. Further, for generating the microsecond timestamps on the TelosB nodes, we use a timer running at 1 MHz. This 1 MHz frequency is obtained by dividing the SMCLK of MSP430 by 4. SMCLK of MSP430 on TelosB runs at 4MHz.

In the second set of experiments (which we refer to in the rest of the paper as *Scenario-2*), we increase the number of cluster members to 10 per cluster. Thus, each cluster consists of 10 cluster members and 1 cluster head. We deploy 3 clusters one-by-one, i.e., the network at most consists of 33 nodes and analyze the performance of E-SATS and other protocols when the WSN nodes are densely deployed.

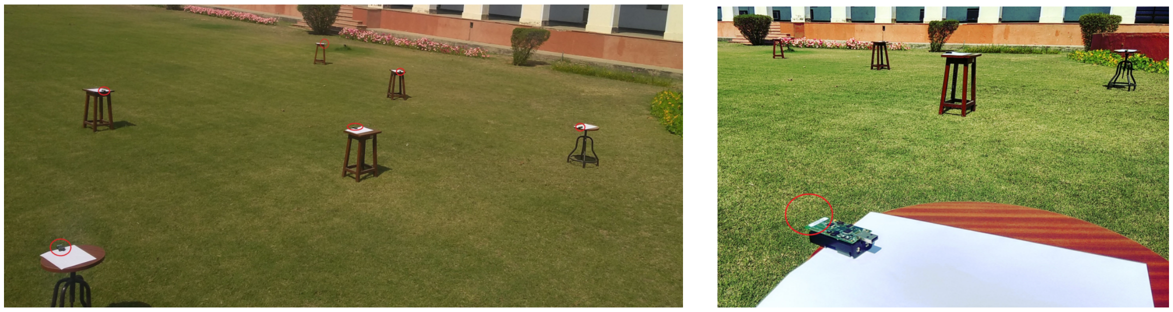
To evaluate the performance of E-SATS, the WSN nodes were deployed in two different environments:

Line-of-Sight (LOS): In this kind of deployment, all the nodes can communicate Line-of-Sight (LOS) without any obstacles between them.

Mixed-LOS: In this kind of deployment, some of the nodes can communicate LOS without any obstacles between them while others were Non-Line-of-Sight (NLOS).

A. LOS environment

Figure 4 shows the deployment of nodes in a LOS environment. The area of each cluster in this deployment was 20 sq. meters and the distance between two clusters was about



(a) (b)

Fig. 4: Set-up for LOS environment (a) Picture showing part of the network (one cluster) deployed with WSN nodes encircled. (b) A zoomed picture showing a node's radio antenna encircled.

4 meters. Such a dense deployment is used to cause intense traffic which would show the behavior of the protocol even in challenging situations. The clusters were activated one-by-one, i.e., initially only one cluster was active and gradually other clusters were powered-on one-by-one. When more than one cluster was operational, the packet exchanges in one cluster caused packet drops in other cluster(s). We employ the same procedure for both *Scenario-1* and *Scenario-2*.

B. Mixed-LOS environment

In this experiment, the cluster head could communicate to some of its cluster members with LOS while other cluster members were NLOS to the cluster head. Some of the nodes were deployed inside a lab while others were deployed in a gallery which was separated from the lab by thick concrete walls. While Figure 5a shows an indicative diagram of the deployment, while Figure 5b shows some of the nodes deployed in the lab. Note that Figure 5a shows an indicative diagram for Scenario-1 (i.e., when there are 4 cluster members per cluster). In the case of Scenario-2, we had 5 cluster members LOS with the cluster head and 5 cluster members NLOS with the cluster head. The presence of thick concrete walls will increase the communication delays among the nodes which are NLOS. In this experiment also, the clusters were turned on one-by-one and the performance of E-SATS along with other protocols was analyzed in each case.

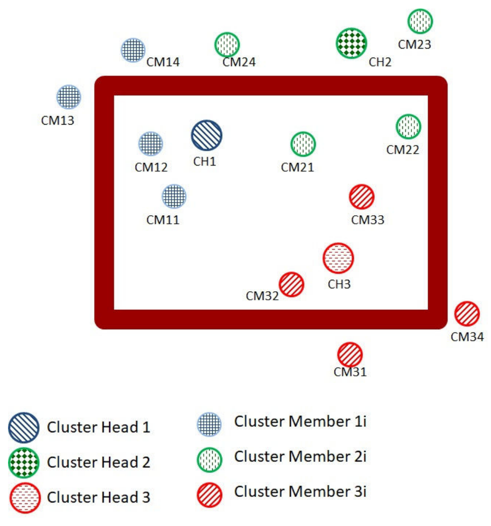
C. Methodology used

The methodology used to implement E-SATS is described in this subsection. The same methodology was followed for LOS environment and mixed-LOS environment. E-SATS was implemented in two phases, i.e., cluster formation phase and synchronization phase. Further, we have performed synchronization evaluation to calculate the synchronization error after the nodes were synchronized.

1) Cluster Formation phase: The cluster formation phase is performed as described in Section IV-B. Same methodology was employed for both Scenario-1 and Scenario-2. We have also estimated the energy consumed by E-SATS to perform the cluster formation. Along the lines of [40], [41], we have used the information about the supply votage used, current consumption and the data rate during transmission and reception operations of the radio of a node from the datasheets of CC2420 [32]. The TelosB mote uses a voltage of 2.92V for transmission and 2.88V for reception [40], [42]. Also the size of the header and footer for any data transmission is 18 bytes for TelosB motes [42], [43]. The header consists of preamble sequence (4 bytes), start of frame delimiter (1 byte), frame-length (1 byte) and MAC header (9 bytes). The footer, i.e., frame check sequence is 2 bytes long. A detailed description of the packet structure can be found in [42]. The cluster head transmits *CM_assignment* packet as explained in IV-B with its *nodeID* and a sequence number. Thus, the length of this packet along with the above-mentioned header and footer is 20 bytes.

The CC2420 radio can be programmed to have different transmission energy levels [32]. We use a transmission energy of -15dBm in these transmissions. At this transmission level, the radio consumes 9.9mA for a transmission and 18.8mA for reception [32]. Since the radio uses a data rate of 250 kbps, it takes 6.4 ms each for both transmission and reception operations. Thus, we can calculate the energy consumed for a transmission and reception of this packet as 18.5 and 34.65 μJ respectively. The total energy consumption for the 30-node network for all the packet transmissions and reception is 1289.18 micro Joules (μJ). It must be noted that this calculation is an approximate estimation without accounting for the energy consumed by the processor of the nodes which will be much smaller in this case.

2) Synchronization phase: After clusters were formed, the cluster heads synchronize their respective cluster members by broadcasting the *Synch_msg* to the cluster members. Since the *Synch_msg* was broadcasted by the cluster head, all the cluster members in that cluster were able to hear this packet. They recorded the reception of this packet and responded to it by backing-off for a predetermined quantum of time. The cluster head then calculates the relative skew and relative offset of each cluster member as discussed in section IV-C and unicasts these values to the corresponding cluster member. In the synchronization phase, 17 iterations were performed, i.e., $Q = 17$. For monitoring, testing and debugging, the timestamps collected by the cluster head were also forwarded to the Base Station node which was connected to a PC as shown in Figure 6b. The right portion of this figure also shows one of the iterations of a cluster member encircled in the enlarged portion of the screen. The nodes were synchronized



(a)

(b)

Fig. 5: Set-up for mixed LOS environment (a) An indicative diagram depicting part of the deployment for mixture of LOS and NLOS environment (b) A picture showing some of the nodes deployed in the lab.

once every 1000 seconds.

As mentioned in Section IV-C, during the synchronization phase, the cluster head broadcasts a *Synch_msg* packet to synchronize the cluster members for every iteration. The cluster members backoff for a pre-determined amount of time and then send their acknowledgment to cluster head. In Scenario-1, this backoff time for the 4 cluster members in a cluster was 1ms, 5ms, 10ms and 15ms. In Scenario-2, the backoff time observed for the 10 cluster members in a cluster was 1ms, 5ms, 10ms, 15ms, 20ms, 25ms, 30ms, 35ms, 40ms and 45 ms. These backoff time periods were determined in such a way that the acknowledgments from the nodes do not collide.

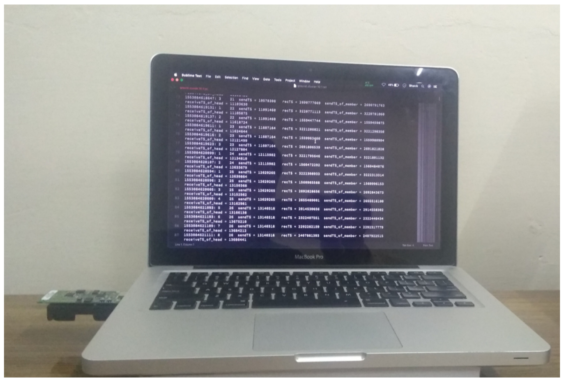
3) Synchronization Evaluation: After the cluster members were synchronized to the cluster head, a good method to evaluate the synchronization error is that both d_i and m_{ij} observe a common event and record the occurrence of this event as per their local clock. These timestamps will be used to calculate the synchronization error. Let us say that the common event was observed by d_i and m_{ij} at $t_e^{'}$ and $t_e^{''}$, respectively, as per their respective local clocks. Then m_{ij} can estimate the local time of d_i when d_i observed this event using (3). Let us say this estimate is $\tilde{t_e^{''}}$. The difference of $\tilde{t_e^{''}}$ and $t_e^{'}$ will give the magnitude of synchronization error. Since the difference in the propagation time of this common-event packet to different nodes is very small, it will be a realistic assumption to say this packet was observed by all the nodes in a cluster at the same time. This method of evaluating the synchronization error was used in these experiments while evaluating the synchronization error of E-SATS and comparing it with other synchronization schemes.

A WSN node which was not part of the network was used to generate the common events by broadcasting a packet to all the nodes in the cluster. We call this node as *tester-node*. The transmission power of the tester-node was adjusted such that it can communicate with all the nodes in the cluster. This tester-node was used to test the synchronization accuracy of each cluster one-by-one. The tester-node sends $\mathcal{S}$ common-event packets and the average synchronization error was calculated. The synchronization evaluation was performed 10 seconds after the synchronization phase was completed in Scenario-1. In Scenario-2, it was performed 25 seconds after the synchronization phase was completed.

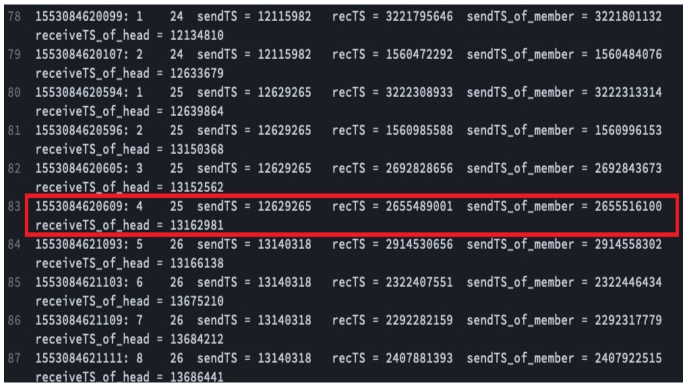
VI. Results and Analysis

A. Performance Analysis in terms of Synchronization Error

As described in the previous section, the WSN nodes were deployed in a LOS environment and in a mixed-LOS environment. The performance of E-SATS was compared in terms of synchronization error (in microseconds) with the regression-based method, CCTS [19] and Revised-CMTS [20]. It must be noted that E-SATS performs synchronization within a cluster only. As explained in Section IV-C, it performs synchronization among the clusters through the gateway nodes only when there is a packet exchange from one cluster to another. Thus, to have a fair comparison of E-SATS with CCTS and Revised-CMTS, we compare only the intra-cluster synchronization phases of Revised-CMTS and CCTS. The regression-based method is the core of the synchronization schemes like SLTP [34] and L-SYNC [33]. The synchronization error observed in LOS-environment for Scenario-1 is shown in Fig. 7. Note that in the rest of the paper we refer to the regression-based method as 'regression-method' for brevity. We observe that when there were only 5 nodes in the network, i.e., one cluster, the synchronization error for all four synchronization protocols compared here was very close to each other. However, as more clusters were added to the network, the difference in the synchronization error widened. As the number of nodes in the network increases, the network traffic (i.e., number of packets exchanged) increases causing increased delays (especially non-deterministic delays). To have a fair comparison among the protocols being analyzed, the number of iterations have been kept the same while evaluating them. Regression-method takes



(a)



```
1553084620099: 1    24  sendTS = 12115982   recTS = 3221795646  sendTS_of_member = 3221801132
receiveTS_of_head = 12134810
1553084620107: 2    24  sendTS = 12115982   recTS = 1560472292  sendTS_of_member = 1560484076
receiveTS_of_head = 12633679
1553084620594: 1    25  sendTS = 12629265   recTS = 3222308933  sendTS_of_member = 3222313314
receiveTS_of_head = 12639864
1553084620596: 2    25  sendTS = 12629265   recTS = 1560985588  sendTS_of_member = 1560996153
receiveTS_of_head = 13150368
1553084620605: 3    25  sendTS = 12629265   recTS = 2692828656  sendTS_of_member = 2692843673
receiveTS_of_head = 13152562
1553084620609: 4    25  sendTS = 12629265   recTS = 2655489001  sendTS_of_member = 2655516100
receiveTS_of_head = 13162981
1553084621093: 5    26  sendTS = 13140318   recTS = 2914530656  sendTS_of_member = 2914558302
receiveTS_of_head = 13166138
1553084621103: 6    26  sendTS = 13140318   recTS = 2322407551  sendTS_of_member = 2322446434
receiveTS_of_head = 13675210
1553084621109: 7    26  sendTS = 13140318   recTS = 2292282159  sendTS_of_member = 2292317779
receiveTS_of_head = 13684212
1553084621111: 8    26  sendTS = 13140318   recTS = 2407881393  sendTS_of_member = 2407922515
receiveTS_of_head = 13686441
```

(b)

Fig. 6: (a) The timestamps collected by each cluster head forwarded to the Base Station connected to a computer. (b) Timestamps of an iteration encircled in screenshot.

into account all the packets to calculate the relative skew and offset. Thus, it will be affected by the delays in all the iterations and consequently its synchronization error is more than that of E-SATS. On the other hand, E-SATS is affected by just two iterations which have minimal delays and thus it shows the least synchronization error. The reason for Revised-CMTS showing an increase in the synchronization error can be attributed to two facts. Firstly, it uses an upper bound of the communication delays in its computations which is just an estimated value. Thus, its results depend on the accuracy of this estimated value. Since the communication delays vary from iteration to iteration, it will lead to inaccurate estimation of compensation parameters. Secondly, as the results of [20] also show, it takes a long time for Revised-CMTS to converge as the network size increases. As mentioned in [20], Revised-CMTS converges when minimum and maximum delays occur in successive iterations and the number of iterations we have taken may not be sufficient for it to converge. However, we cannot afford to perform a large number of broadcasts to allow the nodes to converge as it would drain the batteries of the WSN nodes. In the case of CCTS, it does not account for communication delays in its model and these delays significantly affect the synchronization accuracy. Therefore, as expected, CCTS has large synchronization error in a practical scenario. Further, CCTS also has the problem of large convergence time. The difference in the synchronization error of E-SATS and the consensus methods (both CCTS and Revised-CMTS) increases with increase in network size. It can be noted that E-SATS shows better synchronization accuracy compared to other three protocols for all the network sizes considered.

Similarly, the performance of E-SATS and other protocols was analyzed for clusters containing 10 cluster members for Scenario-2 and the synchronization error for varying network size is shown in Fig.9. From this figure we can observe that as the network size increases, the synchronization error increases for all the four protocols. However, this increase is very steep for CCTS, Revised-CMTS and regression method. The increase in synchronization error for E-SATS is much lower as compared to other three protocols. The increase in synchronization error is attributed to high traffic density in the network leading to increase in packets drops and delays during packet transmission. Since E-SATS uses only two iterations to calculate the relative skew and offset, it shows better performance than other protocols.

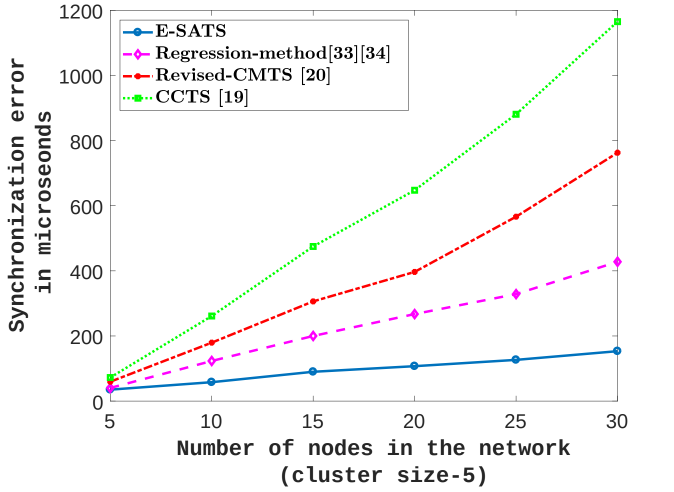


Fig. 7: Synchronization error for varying sized network for different synchronization protocols in LOS environment (with 4 cluster members per cluster)

The results of the experiments performed in the mixed-LOS environment for Scenario-1 are shown in Fig. 8. This figure shows the synchronization error (in microseconds) of E-SATS, regression-method [33], [34], Revised-CMTS [20] and CCTS [19]. In mixed-LOS environment also, when there were only 5 nodes in the network, the difference in the synchronization errors of all the four protocols was close to each other. However, as the number of clusters in the network increases, the difference in the synchronization error also increases due to the reasons mentioned before. However, we see from Figs. 7 and 8 that for the 30 node network, the synchronization error of CCTS is 3 times that of E-SATS in mixed-LOS environment (Scenario-1) while it showed an error which was 6 times that of E-SATS in LOS environment (Scenario-1). In the case of Revised-CMTS, its synchronization error is 3 times that of E-SATS in LOS environment (Scenario-1) and 5 times that of

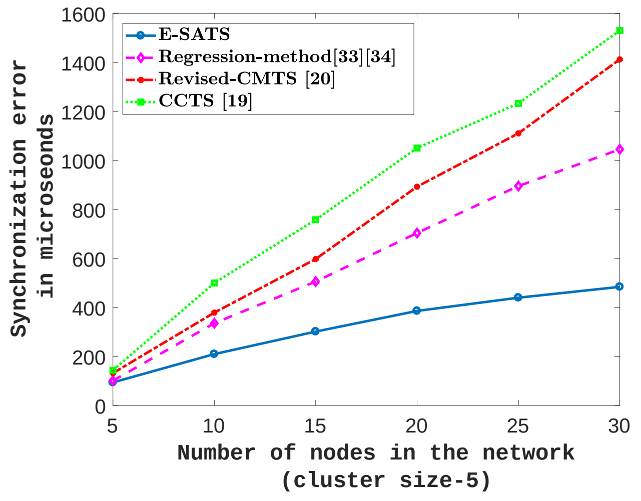


Fig. 8: Synchronization error for varying sized network for different synchronization protocols in mixed-LOS environment (with 4 cluster members per cluster)

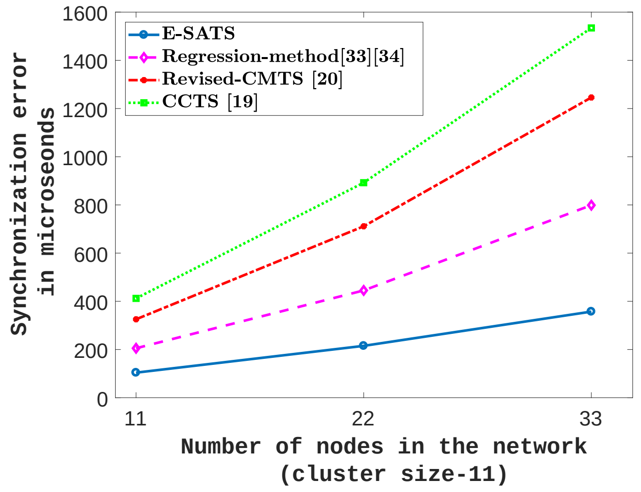


Fig. 9: Synchronization error for varying sized network for different synchronization protocols in LOS environment (with 10 cluster members per cluster)

E-SATS in mixed-LOS environment (Scenario-1).

Similarly, the performance of E-SATS and other protocols was analyzed for clusters containing 10 cluster members for Scenario-2 for mixed-LOS environment and the synchronization error for varying network sizes is shown in Fig. 10. In this scenario also, we observe a similar trend as observed in Fig. 8. However, the synchronization errors for Scenario-2 are much higher than those for Scenario-1. This is due to increase in the network traffic density as explained before.

We see from the above results that in the mixed-LOS environment, all the synchronization schemes experience increased synchronization errors. This can be attributed to the fact that there is an increase in packet loss in NLOS communication. The synchronization error of E-SATS for a 30-node network was 484 μs in the mixed-LOS environment compared to 153 μs in LOS environment in Scenario-1. The performance gain, i.e., improvement of E-SATS over the other protocols is lesser

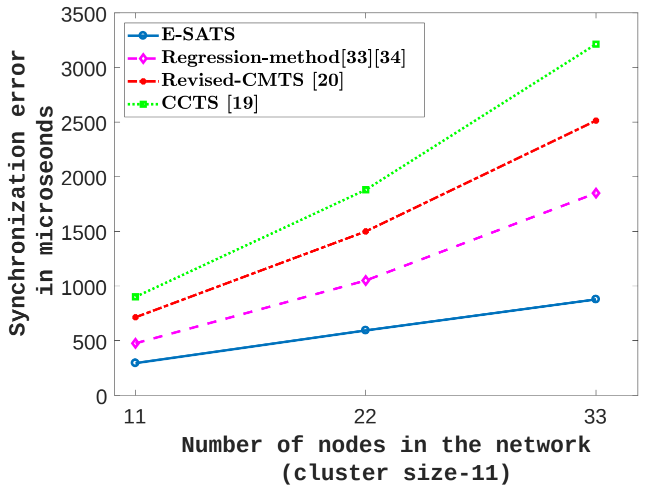


Fig. 10: Synchronization error for varying sized network for different synchronization protocols in mixed-LOS environment (with 10 cluster members per cluster)

in the mixed-LOS environment compared to LOS-environment due to this increased synchronization error. Thus, we see a wide gap in the synchronization accuracy of a protocol in LOS and mixed-LOS environments. Thus, we conclude that NLOS communication significantly affects the performance of synchronization protocols and increases the synchronization error.

B. Comparison of Energy Consumption and Computational Complexity

The computational complexity of E-SATS, regression-method [33] [34], CCTS [19] and Revised-CMTS [20] is shown in Table II. Note that this table is not specific to a particular LOS environment considered in the previous section and does not include computations due to packet drops. In this table, Q represents the number of iterations used in each synchronization phase, h represents the total number of clusters and u represents the total number of cluster members. Thus, for a single cluster network, with Q=17, h=1 and u=4, E-SATS uses 216 addition, 28 multiplication and 4 division operations. It can be seen that the number of computations for E-SATS are much lesser than other protocols.

In Figs. 11, 12 and 13, we have only compared the computational complexity for Scenario-1, i.e., when $u = 4$ due to

TABLE II: Computational Complexity of E-SATS and other protocols (refer Table I for notation meanings)

Protocol	**Additions**	**Multi-plications**	**Divisions**
E-SATS	(3Q+3)hu	7hu	hu
Regression-based method [34][33]	(4Q-1)hu	(2Q+6)hu	2hu
CCTS [19]	Qh(5u+3)	Qh(4u+4)	(3Qh)
Revised-CMTS[20]	(9Qhu)	(5Qhu)	Qhu

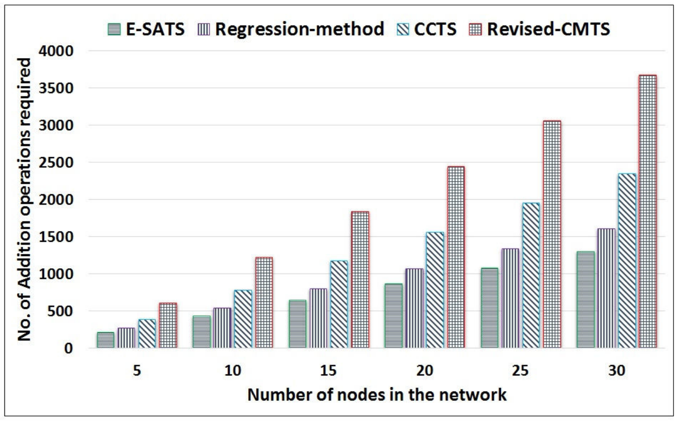


Fig. 11: Number of addition operations in E-SATS and other protocols with varying network size (with 4 cluster members per cluster).

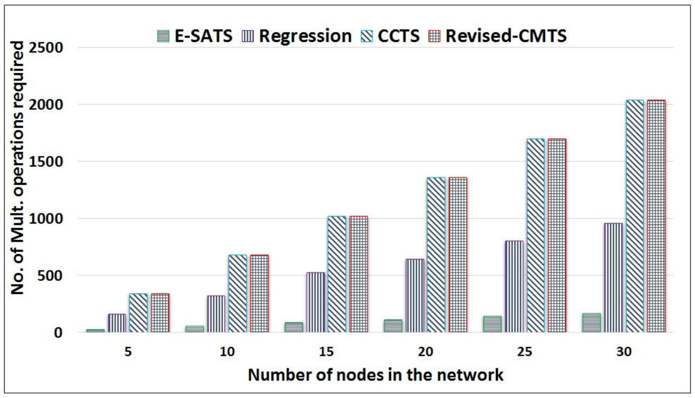


Fig. 12: Number of multiplication operations in E-SATS and other protocols with varying network size (with 4 cluster members per cluster).

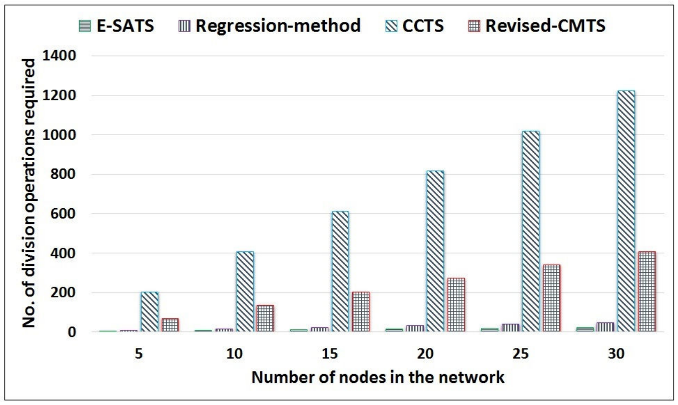


Fig. 13: Number of division operations in E-SATS and other protocols with varying network size (with 4 cluster members per cluster).

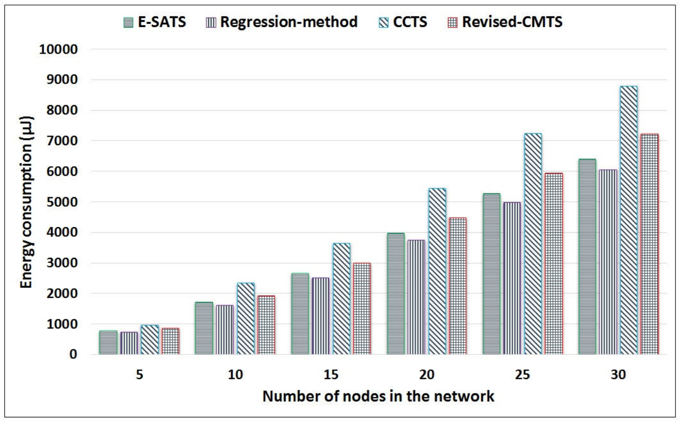


Fig. 14: Energy consumption (in micro Joules (μJ)) of E-SATS and other protocols for the transmission and receptions during one synchronization cycle with varying network size (with 4 cluster members per cluster).

space constraints. However, for Scenario-2, these graphs will scale linearly as the value of u becomes 10 instead of 4.

Figure 11 shows the number of addition operations required in E-SATS and other protocols for varying network size (with 4 cluster members per cluster). E-SATS uses 1296 additions in a 30 node-network while the regression-based method requires 1608, CCTS requires 2346 and Revised-CMTS requires 3672 additions. Thus, E-SATS requires just 55.24% of the number of additions required by CCTS and 35.29% of that required by Revised-CMTS.

Figure 12 shows the multiplication operations required by E-SATS and other protocols for varying network size (with 4 cluster members per cluster). E-SATS has an enormous advantage over other protocols in the number of multiplication operations required to perform synchronization. For a 30-node network, E-SATS requires just 168 multiplication operations which is just 17.5% of the multiplications required by regression-method and 8.23% of that required by CCTS and Revised-CMTS.

Figure 13 shows the division operations required by E-SATS and other protocols for varying network size (with 4 cluster members per cluster). In a 30 node network, E-SATS requires just 24 division operations which is the 50% of what is required by regression, 1.68% of that required by CCTS and 5.88% of what is required by Revised-CMTS. Thus, in all the cases, i.e., from 5 node network to 30 node network, E-SATS requires the least number of additions, multiplications, and division operations. This makes E-SATS the most suitable synchronization protocol for cluster-based WSNs.

The energy consumption of E-SATS, Regression-method, CCTS and Revised-CMTS for the transmissions and receptions during one synchronization phase in μJ for varying network size (with 4 cluster members per cluster) is shown in Fig. 14. It must be noted that these energy estimations do not include any retransmissions due to packet drops and it is not specific to any LOS conditions considered in the previous section. We have estimated the energy consumed for each protocol using the same method used in Section V-C to calculate the energy consumption for the cluster formation phase. The energy consumption of E-SATS is slightly more than that of regression-method. However, this difference is at most 361.4 μJ for a 30-node network. This difference is due to the fact that the regression-method uses smaller data payload for the packets exchanged though the number of packet transmissions and receptions in regression-method and E-SATS are same. It must be noted that this graph shows the energy consumption of the radio only. The total energy consumption of a WSN node during synchronization phase is roughly equal to the summation of the energy consumed by the radio (for transmissions and receptions) and the energy consumed by its processor (to perform the addition, multiplications and divisions). We have not shown the energy consumed by the processor of the node for these arithmetic operations and have just shown the number of operations required by each of the protocols during the synchronization phase. This is because arithmetic operations involving floating-point numbers require variable number of clock cycles in MSP430 depending on the numbers involved in a each operation [44]. To determine the exact number of cycles (and thus the energy consumption) of these operations will require computationally intensive methods like profiling which cannot be performed on the energy-constrained WSN nodes. However, we note that most of these operations are floating-point data operations requiring large number of cycles. For example, according to [44], a C library of MSP430 requires 427 cycles to perform a multiplication of an integer and a floating-point number. The current consumed by MSP430 on TelosB mote which runs at 4.1MHz is 2.33 mA during a multiplication operation [45]. So the regression-method which requires 960 multiplications (in 30-node network) compared to 168 multiplications of E-SATS will consume 569.615 μJ more energy than E-SATS (if the multiplications involve an integer and a floating-point number). We note that this excess energy consumed by regression method for multiplication operations (569.615 μJ) is much more than the energy it has saved compared to E-SATS in the transmission and reception operations (i.e., 361.4 μJ). Further, a multiplication of two floating-point numbers requires many more cycles which only increases the energy consumption of regression-method compared to E-SATS. The multiplications involved in the calculations of these synchronization protocols mostly involve two floating-point numbers. Thus, we can conclude that E-SATS consumes least energy among the compared synchronization protocols. Further, E-SATS offers better synchronization accuracy than the compared protocols.

VII. Conclusion

A simple and efficient time synchronization protocol for WSNs with a cluster-based topology called E-SATS was proposed in this work. E-SATS was tested on a practical WSN testbed in a LOS environment and a mixed-LOS (mixture of LOS and NLOS) environment. It achieves micro-second level accurate synchronization in both the environments. This protocol was found to outperform other state-of-the-art synchronization protocols for clustered WSNs in the two environments considered, both in terms of synchronization accuracy and also in terms of a number of computations. It also consumes lesser energy than the other protocols. These features make E-SATS a suitable synchronization protocol for resource constrained WSNs having cluster-based topology.

Acknowledgment

We would like to thank Raunak Manekar, Bhavik Dhandhaliya, Deependra Singh, Sahil Chandna and Syed Irfan for their help during the experiments.

References

[1] A. Manjeshwar and D. P. Agrawal, "TEEN: A Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks," in *Proc. 15th IPDPS-2001*, San Francisco, CA, USA, 2001, pp. 2009–2015.

[2] N. K. Suryadevara, S. C. Mukhopadhyay, S. D. T. Kelly, and S. P. S. Gill, "WSN-based Smart Sensors and Actuator for Power Management in Intelligent Buildings," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 2, pp. 564–571, April 2015.

[3] S. Ivanov, K. Bhargava, and W. Donnelly, "Precision Farming: Sensor Analytics," *IEEE Intelligent Systems*, vol. 30, no. 4, pp. 76–80, July 2015.

[4] J. J. Estrada-López, A. A. Castillo-Atoche, J. Vázquez-Castillo, and E. Sánchez-Sinencio, "Smart Soil Parameters Estimation System Using an Autonomous Wireless Sensor Network with Dynamic Power Management Strategy," *IEEE Sensors Journal*, vol. 18, no. 21, pp. 8913–8923, Nov 2018.

[5] N. Dey, A. S. Ashour, F. Shi, S. J. Fong, and R. S. Sherratt, "Developing Residential Wireless Sensor Networks for ECG Healthcare Monitoring," *IEEE Transactions on Consumer Electronics*, vol. 63, no. 4, pp. 442–449, November 2017.

[6] C. A. Pérez, F. S. Valles, R. T. Sánchez, M. J. Buendía, F. López-Castejón, and J. G. Cervera, "Design and Deployment of a Wireless Sensor Network for the Mar Menor Coastal Observation System," *IEEE Journal of Oceanic Engineering*, vol. 42, no. 4, pp. 966–976, Oct 2017.

[7] F. Viani, A. Polo, M. Donelli, and E. Giarola, "A Relocable and Resilient Distributed Measurement System for Electromagnetic Exposure Assessment," *IEEE Sensors Journal*, vol. 16, no. 11, pp. 4595–4604, June 2016.

[8] C. Laoudias, A. Moreira, S. Kim, S. Lee, L. Wirola, and C. Fischione, "A Survey of Enabling Technologies for Network Localization, Tracking, and Navigation," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2018.

[9] Z. Song, M. T. Lazarescu, R. Tomasi, L. Lavagno, and M. A. Spirito, *High-Level Internet of Things Applications Development Using Wireless Sensor Networks*, Cham: Springer International Publishing, 2014.

[10] Z. Iqbal, K. Kim, and H. Lee, "A Cooperative Wireless Sensor Network for Indoor Industrial Monitoring," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 482–491, April 2017.

[11] U. I. Minhas, I. H. Naqvi, S. Qaisar, K. Ali, S. Shahid, and M. A. Aslam, "A WSN for Monitoring and Event Reporting in Underground Mine Environments," *IEEE Systems Journal*, vol. 12, no. 1, pp. 485–496, March 2018.

[12] R. A. Lara-Cueva, R. Gordillo, L. Valencia, and D. S. Benítez, "Determining the Main CSMA Parameters for Adequate Performance of WSN for Real-time Volcano Monitoring System Applications," *IEEE Sensors Journal*, vol. 17, no. 5, pp. 1493–1502, March 2017.

[13] F. Sivrikaya and B. Yener, "Time Synchronization in Sensor Networks: A Survey," *Netwrk. Mag. of Global Internetwkg.*, vol. 18, no. 4, pp. 45–50, July 2004.

[14] D. L. Mills, "Precision Synchronization of Computer Network Clocks," *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 2, pp. 28–43, Apr. 1994.

[15] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync Protocol for Sensor Networks," in *Proc.SenSys '03*, Los Angeles, CA, USA, 2003, pp. 138–149.

[16] J. Elson, L. Girod, and D. Estrin, "Fine-grained Network Time Synchronization Using Reference Broadcasts," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 147–163, Dec. 2002.

[17] W. Su and I. F. Akyildiz, "Time-diffusion Synchronization Protocol for Wireless Sensor Networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 384–397, April 2005.

[18] P. Sommer and R. Wattenhofer, "Gradient Clock Synchronization in Wireless Sensor Networks," in *Proc.IPSN '09*, San Francisco, CA, USA, 2009, pp. 37–48.

[19] J. Wu, L. Zhang, Y. Bai, and Y. Sun, "Cluster-based Consensus Time Synchronization for Wireless Sensor Networks," *IEEE Sensors Journal*, vol. 15, no. 3, pp. 1404–1413, March 2015.

[20] Z. Wang, P. Zeng, M. Zhou, D. Li, and J. Wang, "Cluster-based Maximum Consensus Time Synchronization for Industrial Wireless Sensor Networks," *Sensors*, vol. 17, no. 1, pp. 141.1–141.16, 2017.

[21] D. Djenouri and M. Bagaa, "Synchronization Protocols and Implementation Issues in Wireless Sensor Networks: A Review," *IEEE Systems Journal*, vol. 10, no. 2, pp. 617–627, June 2016.

[22] A. Elsts, S. Duquennoy, X. Fafoutis, G. Oikonomou, R. Piechocki, and I. Craddock, "Microsecond-Accuracy Time Synchronization Using the IEEE 802.15.4 TSCH protocol," in *2016 LCN Workshops*, Dubai, UAE, 2016, pp. 156–164.

[23] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The Flooding Time Synchronization Protocol," in *Proc. SenSys '04*, Baltimore, MD, USA, 2004, pp. 39–49.

[24] H. Dai and R. Han, "TSync: A Lightweight Bidirectional Time Synchronization Service for Wireless Sensor Networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 8, no. 1, pp. 125–139, Jan. 2004.

[25] K. Noh, E. Serpedin, and K. Qaraqe, "A New Approach for Time Synchronization in Wireless Sensor Networks: Pairwise Broadcast Synchronization," *IEEE Transactions on Wireless Communications*, vol. 7, no. 9, pp. 3318–3322, September 2008.

[26] B. Etzlinger, F. Meyer, F. Hlawatsch, A. Springer, and H. Wymeersch, "Cooperative Simultaneous Localization and Synchronization in Mobile Agent Networks," *IEEE Transactions on Signal Processing*, vol. 65, no. 14, pp. 3587–3602, July 2017.

[27] F. Meyer, B. Etzlinger, Z. Liu, F. Hlawatsch, and M. Z. Win, "A Scalable Algorithm for Network Localization and Synchronization," *IEEE Internet of Things Journal*, pp. 1–1, 2018.

[28] M. L. Sichitiu and C. Veerarittiphan, "Simple, Accurate Time Synchronization for Wireless Sensor Networks," in Proc. WCNC 2003, New Orleans, LA, USA, 2003, pp. 1266–1273 vol.2.

[29] Q. M. Chaudhari, E. Serpedin, and K. Qaraqe, "On Maximum Likelihood Estimation of Clock Offset and Skew in Networks With Exponential Delays," *IEEE Transactions on Signal Processing*, vol. 56, no. 4, pp. 1685–1697, April 2008.

[30] G. S. S. Chalapathi, R. Manekar, V. Chamola, K. R. Anupama, and S. Gurunarayanan, "Hardware Validated Efficient and Simple Time Synchronization protocol for Clustered WSN," in *Proc IEEE TENCON*, Singapore 2016, pp. 2162–2166.

[31] "Telosb-Datasheet," http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf, accessed: 2018-08-14.

[32] "CC2420-Datasheet," http://www.ti.com/lit/ds/symlink/cc2420.pdf, accessed: 2018-09-21.

[33] M. Jabbarifar, A. S. Sendi, H. Pedram, M. Dehghan, and M. Dagenais, "L-SYNC: Larger Degree Clustering Based Time-Synchronisation for Wireless Sensor Network," in Proc. *ACIS International Conference on Software Engineering Research, Management and Applications*, Montreal, QC, Canada,2010, pp. 171–178.

[34] S. Nazemi Gelyan, A. N. Eghbali, L. Roustapoor, S. A. Yahyavi Firouz Abadi, and M. Dehghan, "SLTP: Scalable Lightweight Time Synchronization Protocol for Wireless Sensor Network," in *Mobile Ad-Hoc and Sensor Networks*, Springer Berlin Heidelberg, 2007, pp. 536–547.

[35] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, Jan 2000, pp. 10 pp. vol.2–.

[36] H. Yang, F. Ye and B. Sikdar, "A dynamic query-tree energy balancing protocol for sensor networks," in *Proc.2004 WCNC*, Atlanta, GA, USA, 2004, pp. 1715-1720 Vol.3.

[37] M. Mamun-Or-Rashid, C. S. Hong, and C.-H. In, "Passive Cluster Based Clock Synchronization in Sensor Network," in *Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop (AICT/SAPIR/ELETE'05)*, July 2005, pp. 340–345.

[38] H.Kim, D.Kim and S.Yoo, "Cluster-based Hierarchical Time Synchronization for Multi-hop Wireless Sensor Networks," in *Proc. AINA'06*, Vienna, Austria, 2006, pp. 5 pp.-.

[39] L. Kong, Q. Wang and Y. Zhao, "Time Synchronization Algorithm based on Cluster for WSN," *International Conference on Information Management and Engineering*, Chengdu, China, 2010, pp. 126-130.

[40] L. Shi, J. Yuan, S. Yu, and M. Li, "MASK-BAN: Movement-Aided Authenticated Secret Key Extraction Utilizing Channel Characteristics in Body Area Networks," *IEEE Internet of Things Journal*, vol. 2, no. 1, pp. 52–62, Feb 2015.

[41] F. Marcelloni and M. Vecchio, "An Efficient Lossless Compression Algorithm for Tiny Nodes of Monitoring Wireless Sensor Networks," *The Computer Journal*, vol. 52, no. 8, pp. 969–987, 2009.

[42] M. Amiri, "Measurements of Energy Consumption and Execution Time of Different Operations on Tmote Sky Sensor Motes," Master's thesis, Masaryk University, Brno, Czech Republic, 2010.

[43] M. Abdelaal and O. Theel, "An Efficient and Adaptive Data Compression Technique for Energy Conservation in Wireless Sensor Networks," in *2013 IEEE Conference on Wireless Sensor (ICWISE)*, Dec 2013, pp. 124–129.

[44] "Efficient Multiplication and Division Using MSP430™ MCUs", http://www.ti.com/lit/an/slaa329a/slaa329a.pdf, accessed: 2018-11-02.

[45] A. Prayati, C. Antonopoulos, T. Stoyanova, C. Koulamas, and G. Papadopoulos, "A modeling approach on the Telosb WSN platform power consumption," *Journal of Systems and Software*, vol. 83, no. 8, pp. 1355 – 1363, 2010.

G.S.S. Chalapathi [S'13] obtained his B.E. and M.E. from Birla Institute of Technology and Science (BITS), Pilani, India in 2009 and 2011 respectively. He is currently working for his Ph.D. from BITS-Pilani, Pilani Campus. He is currently working as an Assistant Professor in the Department of Electrical and Electronics Engineering, BITS-Pilani, Pilani Campus. He has been a visiting researcher at Johannes Kepler University from June- Aug. 2016 and from June-July 2017. His research interests include Wireless Sensor Networks, Edge Computing, Internet-of-Things (IoT). He is a Graduate Student Member of the IEEE.

Vinay Chamola received his B.E. degree in electrical & electronics engineeerig and Master's degree in communication engineering from Birla Institute of Technology & Science (BITS), Pilani, India in 2010 and 2013 respectively. He received his Ph.D. degree in electrical and computer engineering from the National University of Singapore, Singapore, in 2016. From June to Aug. 2015, he was a visiting researcher at the Autonomous Networks Research Group (ANRG) at University of Southern California (USC), USA. Currently he is a Research Fellow at the National University of Singapore. His research interests include solar powered cellular networks, energy efficiency in cellular networks, internet of things, and networking issues in cyberphysical systems.

S Gurunarayanan received his Ph.D. from BITS Pilani. He is currently a Professor, in the Department of Electrical and Electronics Engineering, BITS Pilani. He has about three decades of teaching experience at BITS-Pilani. His research interests are multi-core processor architectures, cache and memory architectures and embedded systems. He is also serving as the Dean (University-wide), Practice School Division, BITS Pilani.

Biplab Sikdar [S'98, M'02, SM'09] received the B.Tech. degree in electronics and communication engineering from North Eastern Hill University, Shillong, India, in 1996, the M.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, in 1998, and the Ph.D. degree in electrical engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 2001. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. His research interests include wireless MAC protocols, transport protocols, network security, and queuing theory.