

# Time-Diffusion Synchronization Protocol for Wireless Sensor Networks

Weilian Su, *Member, IEEE*, and Ian F. Akyildiz, *Fellow, IEEE*

**Abstract**—In the near future, small intelligent devices will be deployed in homes, plantations, oceans, rivers, streets, and highways to monitor the environment. These devices require time synchronization, so voice and video data from different sensor nodes can be fused and displayed in a meaningful way at the sink. Instead of time synchronization between just the sender and receiver or within a local group of sensor nodes, some applications require the sensor nodes to maintain a similar time within a certain tolerance throughout the lifetime of the network. The *Time-Diffusion Synchronization Protocol (TDP)* is proposed as a network-wide time synchronization protocol. It allows the sensor network to reach an equilibrium time and maintains a small time deviation tolerance from the equilibrium time. In addition, it is analytically shown that the TDP enables time in the network to converge. Also, simulations are performed to validate the effectiveness of TDP in synchronizing the time throughout the network and balancing the energy consumed by the sensor nodes.

**Index Terms**—Sensor networks, time synchronization, timing.

## I. INTRODUCTION

**I**N the near future, small intelligent devices will be deployed in homes, plantations, oceans, rivers, streets, and highways to monitor the environment [1]. Events such as target tracking, speed estimating, and ocean current monitoring require the knowledge of time between sensor nodes that detect the events. In addition, sensor nodes may have to time-stamp data packets for security reasons. With time synchronization, voice and video data from different sensor nodes can be fused and displayed in a meaningful way at the sink.

Instead of time synchronization between just a sender and a receiver or within a local group of sensor nodes, some applications require all the sensor nodes to maintain a similar time within a certain tolerance throughout the lifetime of the network. Combining with the criteria that sensor nodes have to be energy efficient, low-cost, and small in a multi-hop environment, this requirement becomes a challenging problem to solve. In addition, the sensor nodes may be left unattended for a long period of time, e.g. in deep space or on an ocean floor. Note that in short distance multi-hop broadcast, the data processing time and

its variation contribute the most to time fluctuations and differences in the path delays. Also, the time difference between two sensor nodes may become large over time due to the wandering effect of the local clocks.

A proposed solution called *Reference-Broadcast Synchronization (RBS)* [6] aims to provide instantaneous synchronization among a set of receivers that are within the reference broadcast of the transmitter. It is argued that RBS can provide multi-hop synchronization by translating the time between different broadcast domains. However, the impact of the translation errors and delays on the synchronization still needs to be studied. In addition, the energy dissipation and effects of node mobility in large scale sensor networks, e.g., 300, 1000, and 2000 nodes, need to be addressed.

In the Internet, the *Network Time Protocol (NTP)* [14] is used to discipline the frequency of each node's oscillator. It is used to provide network-wide agreement among a large group of nodes in the Internet. It maybe useful to use NTP to discipline sensor nodes, but the sensor nodes may be off when power management and topology maintenance protocols, e.g., SPAN [4] and LEACH [9], are employed. In addition, disciplining all the sensor nodes in the sensor field may be a problem due to interferences from the environment and large variation of delays between different parts of the sensor field. The interferences can temporarily disjoint the sensor field into multiple smaller fields causing undisciplined clocks among these smaller fields.

A more recently developed *Time-Sync protocol for Sensor Networks (TPSN)* [7] is based on similar methodology as the NTP, where the sensor nodes are organized into multiple levels and synchronized to the root node of the hierarchy. Unlike the Internet, the root node and nodes at different levels responsible for synchronization may fail often, which may cause synchronization problems. In addition, mobile nodes may disrupt the predefined level-by-level synchronization procedure.

To provide network-wide time synchronization, the time differences among the sensor nodes must be minimized before protocols requiring time-stamps, e.g., security applications, flow control protocols, target tracking, voice fusion, video fusion, and environmental data fusion, are realizable. In addition, the time synchronization protocol must be robust to node failures as well as energy consumption in the network. Also, node mobility must be taken into account.

As a result, we propose the *Time-Diffusion Synchronization Protocol (TDP)*. The motivations for a network-wide time synchronization are as follows:

- Enable applications to coordinate sensor nodes, e.g., target tracking, data fusion, and decision fusion.

Manuscript received October 4, 2002; revised October 20, 2003; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor G. Pacifici. This work was supported by the National Science Foundation under Contract ECS-0225497.

W. Su was with the Broadband and Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA. He is now with the Advance Networking Laboratory, Department of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA 93943 USA (e-mail: weilian@nps.edu).

I. F. Akyildiz is with the Broadband and Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: ian@ece.gatech.edu).

Digital Object Identifier 10.1109/TNET.2004.842228

- Enable users to perceive events in the same time frame, e.g., multiple fire outbreaks at different locations of the sensor field.
- Enable protocols that require time-stamps, e.g., security, flow control, and medium access protocols.

The TDP is used to maintain the time throughout the network within a certain tolerance. The tolerance level can be adjusted based on the application of the sensor networks. One of the benefits of TDP is that the performance of voice and video applications can be improved when multiple sources are sending data back to the sink through flooding or *directed diffusion* [11]. The TDP enables the sink to detect the time difference between multiple sources, so that the temporal differences can be adjusted. In addition, it allows the sink to issue a start time to the sensor nodes allowing interactive sensing and monitoring.

The design issues and system architecture of TDP are described in Section II. The TDP protocol is presented in Section III. The analytical performance and simulation results are discussed in Sections IV and V, respectively. Finally, the paper is concluded in Section VI.

## II. DESIGN ISSUES AND SYSTEM ARCHITECTURE

Some of the factors influencing time synchronization in large systems such as *personal computers (PCs)* also apply to sensor networks [12]; they are *temperature*, *phase noise*, *frequency noise*, *asymmetric delays*, and *clock glitches*.

- **Temperature:** Since sensor nodes are deployed in various places, the temperature variations throughout the day may cause the clock to speed up or slow down. For a typical PC, the clock drifts few *parts per million (ppm)* during the day [15]. For low-end sensor nodes, the drifting may be even worse.
- **Phase noise:** Some of the causes of phase noise are access fluctuations at the hardware interface, response variation of the operating system to interrupts, and jitter in the network delay. The jitter in the network delay may be due to medium access and queueing delays.
- **Frequency noise:** The frequency noise is due to the instability of the clock crystal. A low-end crystal may experience large frequency fluctuation, because the frequency spectrum of the crystal has large sidebands on adjacent frequencies.
- **Asymmetric delay:** Since sensor nodes communicate with each other through the wireless medium, the delay of the path from one node to another may be different than the return path. As a result, an asymmetric delay may cause an offset to the clock that can not be detected by a variance type method [12]. If the asymmetric delay is static, the time offset between any two nodes is also static. The asymmetric delay is bounded by one-half the round trip time between the two nodes [12].
- **Clock glitches:** Clock glitches are sudden jumps in time. This may be caused by hardware or software anomalies such as frequency and time steps.

Besides dealing with these factors, a time synchronization protocol for sensor networks should be *automatically self-configured* and be *sensitive to energy requirement*. These are the

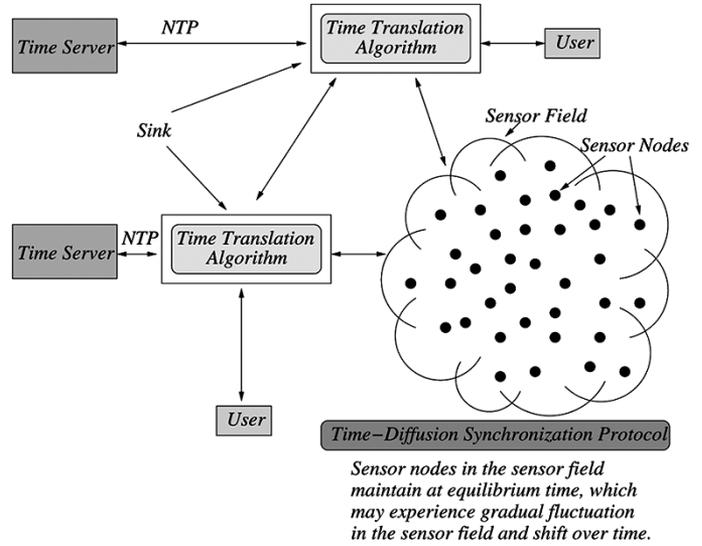


Fig. 1. System architecture.

two design criteria that the TDP is engineered around. The TDP self-configures and self-organizes to address the frequent network partitioning caused by sensor node failures. In addition, the TDP does not depend on any particular node to be a time server/master node, so the workload can be spread to all sensor nodes.

In the following paragraphs, the TDP is described for both cases: (i) *with precise time servers* and (ii) *without precise time servers*.

- With precise time servers:* The overall system architecture of how TDP interacts with the outside world is shown in Fig. 1. *The main objective of the TDP is to enable the time of the sensor nodes to reach an equilibrium time.* The sinks may act as precise time servers for the sensor nodes residing in the sensor field. They broadcast a reference time to all the master nodes in the sensor network; master nodes are sensor nodes randomly elected to synchronize their neighbors. In turn, the master nodes use the received reference time to synchronize their neighbor nodes by using the TDP. In essence, the equilibrium time that the sensor network reaches is the reference time broadcast by the sinks.
- Without precise time servers:* Although the TDP can be used with precise time servers, it is more important to discuss about the autonomous nature of TDP since the line-of-sight or connection to all master nodes from the sinks may not be possible. Also, the sensor network may be deployed in areas that may not be accessed by the sinks for a long period of time, e.g., caves and ocean floor. Consequently, the sinks may not be used as time servers; fortunately, the autonomous nature of the TDP allows the sensor network to reach an equilibrium time that is independent from the time used by the Internet, e.g., *Universal Coordinated Time (UTC)*.

Since the time in the sensor network reaches an equilibrium value, it still may drift over time and has fluctuation throughout the sensor network. Although the time variation throughout the sensor network may be very small, it is necessary to translate

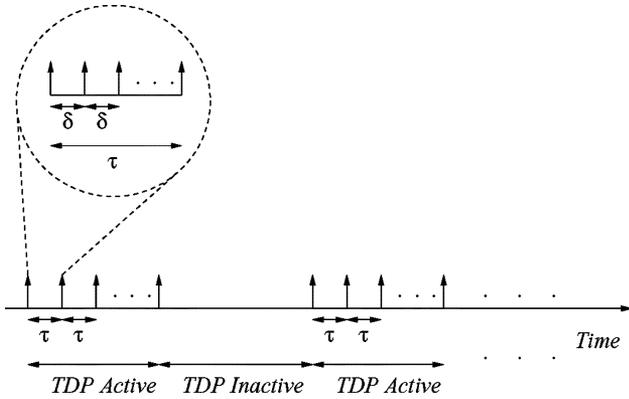


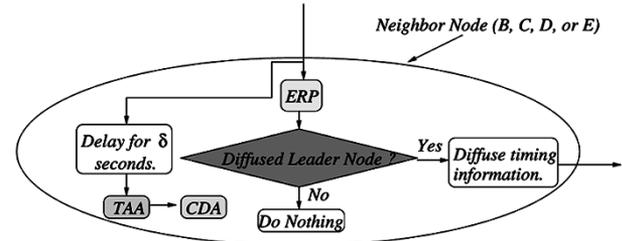
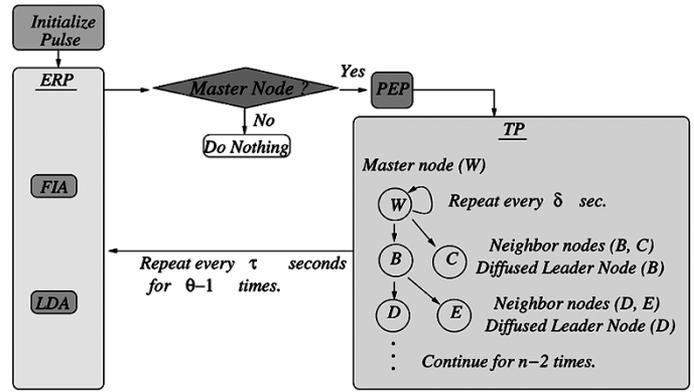
Fig. 2. TDP active/inactive schedule.

the time in the sensor network to a common time, e.g., UTC, used by the users. The sinks as shown in Fig. 1 take care of such translation with the *Time Translation Algorithm* [18], which is based on playout buffer technique similar to *Jitter Time Stamp* [17]. In addition, they serve as interfaces to the sensor network and synchronize to the time servers using the NTP [16]. Note that the focus of this particular paper is on the synchronization problem in the sensor field without precise time servers.

The time schedule for applying TDP is illustrated in Fig. 2. During the active period, the master nodes are reelected at every  $\tau$  seconds, which is a design parameter that depends on the types of sensor networks. The master nodes broadcast the timing information to their neighbors, which use this time as timing reference. The neighbor nodes self-determine to become diffused leader nodes that further broadcast the timing information to their neighbors. The master nodes repeat this process every  $\delta$  seconds as shown in Fig. 2. The duration of the TDP active period depends on the range of time variation allowed throughout the sensor network. On the other hand, the inactive period depends on the amount of clock drifts allowed before TDP is activated again. The active and inactive periods are design parameters that can be tailored for different types of sensor networks. In addition, they are synchronized among all the master nodes. An overview of the procedures and functionality of the TDP is described in the following paragraphs.

The TDP architecture consists of many algorithms and procedures as illustrated in Fig. 3. The TDP protocol focuses on all the algorithms and procedures except the *clock discipline algorithm*. The clock discipline algorithm may use the adaptive hybrid clock discipline algorithm intended for NTP Version 4 [15]. The hybrid clock discipline algorithm uses a combination of phase lock loop (PLL) and frequency lock loop (FLL), which are usually implemented in hardware to minimize the noise. For low-end sensor nodes, it may not be possible to have a combination of PLL and FLL due to monetary cost of each node. As a result, there may still be room for a different type of clock discipline algorithm specifically designed for low-end sensor nodes.

The algorithms and procedures in Fig. 3 are used to autonomously synchronize the nodes, remove the false tickers (clocks that deviate from their neighbors), and balance the load required for time synchronization among the sensor nodes. Initially, the sensor nodes may receive an *Initialize pulse* from the sink either through direct broadcast or multi-hop flooding.



ERP = Election/reelection of master/diffused leader node procedure  
 FIA = False ticker isolation algorithm  
 LDA = Load distribution algorithm

PEP = Peer evaluation procedure  
 TP = Time diffusion procedure  
 TAA = Time adjustment algorithm  
 CDA = Clock discipline algorithm

Fig. 3. TDP architecture.

Then they self-determine to become master nodes with the *election/reelection of master/diffused leader node procedure* (ERP), which consists of the *false ticker isolation algorithm* (FIA) and *load distribution algorithm* (LDA) as shown in Fig. 3. At the end of procedure ERP, the elected master nodes start the *peer evaluation procedure* (PEP) while others do nothing. The procedure PEP helps to remove false tickers from becoming a master node or a diffused leader node.

After procedure PEP, the elected master nodes (denoted by  $W$  in Fig. 3) start the *time diffusion procedure* (TP), where they diffuse the timing information messages at every  $\delta$  seconds (round interval) for a duration of  $\tau$  seconds. Each neighbor node (e.g., node  $B$  or  $C$  in Fig. 3) receiving these timing information messages self-determines to become a diffused leader node using the procedure ERP. Furthermore, all neighbor nodes adjust their local clocks using *time adjustment algorithm* (TAA) and *clock discipline algorithm* (CDA) after waiting for  $\delta$  seconds as shown in Fig. 3.

The elected diffused leader nodes (e.g., node  $B$ ) will further diffuse the timing information messages to their neighboring nodes (e.g., nodes  $D$  and  $E$ ) within their broadcast range. Note that these timing information messages are diffused by each elected diffused leader node for  $n$  hops from the master nodes, where each hop represents one level from the master nodes (e.g., nodes  $B$  and  $C$  are at Level 1 while nodes  $D$  and  $E$  are at Level 2). This diffusion process enables all nodes to be autonomously synchronized. In addition, the master nodes are re-elected at every  $\tau$  seconds using the procedure ERP, which is repeated for

$\theta - 1$  times, where  $\theta\tau$  is equal to the length of the TDP active period.

The functionality and novelties of the procedures *PEP*, *TP*, and *ERP* as shown in Fig. 3 are described in the following section. The procedures *PEP* and *TP* are explained before the procedure *ERP*, because the algorithms *FIA* and *LDA* of procedure *ERP* require the understanding of these procedures.

### III. TIME-DIFFUSION SYNCHRONIZATION PROTOCOL

The TDP is composed of components as illustrated in Fig. 3.

#### A. Peer Evaluation Procedure (PEP)

The purpose of procedure *PEP* is to allow neighbor nodes to evaluate the stability of the local clock by using the Allan variance [2]. The Allan variance is used to estimate the deviations between two clocks. The steps are as follows:

*Step 1:* Initially, the elected master nodes broadcast  $\eta$  number of time-stamped *SCAN* messages to the neighbor nodes within a short time interval, so the phase and frequency noises are almost white [12].

*Step 2:* The neighbor nodes use these *SCAN* messages to calculate the 2-sample Allan variance  $\sigma^2(\iota)$  [2], [15] of the local clock from the clock of the master nodes as follows:

$$\sigma^2(\iota) = \frac{1}{2\iota^2(N-2)} \sum_{g=1}^{N-2} (x_{g+2} - 2x_{g+1} + x_g)^2 \quad (1)$$

where  $\iota$  is the time difference between two time deviation measurements;  $N$  is the total number of time deviation measurements, and  $x$  is the measurement value.

*Step 3:* The calculated Allan variances  $\sigma^2(\iota)$  by (1) are sent back to the master nodes with the *REPLY* messages.

*Step 4:* The master nodes then calculate the outlier ratio  $\gamma_{yz}$ , which indicates the amount of clock deviation, between the sensor node  $y$  (e.g., master node) and  $z$  (e.g., neighbor node) by

$$\gamma_{yz} = \left| \frac{\sigma_{yz}^2(\iota) - \sigma_{avg}^2(\iota)}{\sigma_{avg}^2(\iota)} \right| \quad (2)$$

and the average  $\sigma_{avg}^2(\iota)$  of the Allan variances by

$$\sigma_{avg}^2(\iota) = \frac{\sum_{z=1}^M \sigma_{yz}^2(\iota)}{M} \quad (3)$$

where  $\sigma_{yz}^2(\iota)$  is the Allan variance between nodes  $y$  and  $z$  calculated by (1); and  $M$  is the number of Allan variances received. In addition, the average  $\sigma_{avg}(\iota)$  of the Allan deviations ( $\sqrt{\sigma^2(\iota)}$ ) is determined as

$$\sigma_{avg}(\iota) = \frac{\sum_{z=1}^M \sqrt{\sigma_{yz}^2(\iota)}}{M} \quad (4)$$

As a result, the outlier ratio  $\gamma_{yz}$  (2) and the average Allan deviation  $\sigma_{avg}(\iota)$  (4) are sent back to the neighbor nodes using the *RESULT* messages.

*Step 5:* Repeat steps 1 to 4 for  $n$  hops from the master nodes, where in each hop the elected diffused leader nodes are the ones that broadcast  $\eta$  number of time-stamped *SCAN*

messages and perform the evaluation of their neighbor nodes.

*Step 6:* At the end of  $\tau$  period, reset the Allan variances (1) and outlier ratios (2) to zero and start the same procedure with them from the master nodes.

After the procedure *PEP*, all sensor nodes receive the outlier ratios  $\gamma_{yz}$  (2) and the average Allan deviation  $\sigma_{avg}(\iota)$  (4), which are used to evaluate the quality of their clocks with respect to their neighbors by the procedure *ERP*. In the following section, the procedure *TP* is described.

#### B. Time Diffusion Procedure (TP)

As shown in Fig. 3, the procedure *TP* diffuses timing information messages from the master nodes to the neighboring nodes, where the timing information messages are further diffused by the elected diffused leader nodes for  $n$  hops from the master nodes. Effectively, a temporary tree-like structure is created when the master nodes diffuse timing information messages. At the end of this procedure, the timing information is used by the algorithm *TAA* to adjust the local clocks.

*1) Timing Information Handshake:* The timing information messages are diffused from one level to the next starting from the master nodes, and it contains the following fields:

- a) *Master node local ID (M-LID)* (ID of the master node of which it is originated),
- b) *Source LID* (the LID of the node that broadcasts the timing information message),
- c) *Value n* (the number of levels that the timing information message is to be diffused; it is twice the minimum number of hops that covers the sensor field given a number of evenly distributed master nodes. This allows a sensor node to receive multiple timing information messages from different master nodes.),
- d) *Time  $t_{M,i}$*  (the diffused time from the master LID  $M$  that neighbors should synchronize to at Round ( $i$ )), and
- e) *Value  $\beta_{M,k}$*  (the value used by the algorithm *TAA* to calculate the weight for the diffused time  $t_{M,i}$  at Level  $k$ ).

The elected diffused leader nodes at Level 1 respond to the timing information messages with *ACK* messages (Round ( $i$ ) in Fig. 4). Afterwards, the round trip time  $\Delta_j$  between the master node and diffused leader node  $j$  is calculated by

$$\Delta_j = (t_1 - t_0) \quad (5)$$

where  $t_1$  is the arrival time of the *ACK* message and  $t_0$  is the broadcast time of the timing information message at Round ( $i$ ) in Fig. 4.

Since each master node may receive multiple *ACK* messages, the average  $\Delta$  of the round trip delays  $\Delta_j$  ((5)) is calculated and used to estimate the one-way delay between the master node and the neighboring nodes. As a result, the diffused time  $t_{M,i}$  from the master nodes can be calculated as

$$t_{M,i} = t_{M,i} + \frac{\Delta}{2} + \delta \quad (6)$$

where  $\Delta/2$  is the estimated one-way delay, and  $\delta$  is the amount of time that the neighboring nodes wait before adjusting their local clocks.

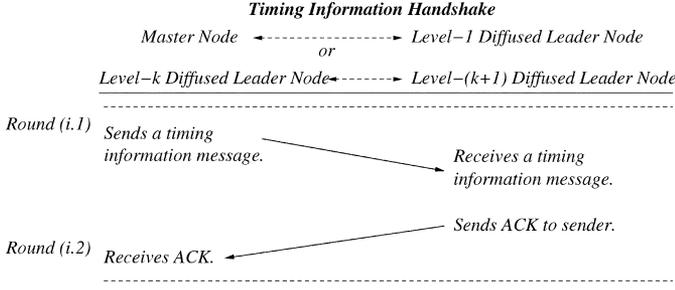


Fig. 4. Handshake of timing information message.

Furthermore, the standard deviation  $\alpha$  of the round trip delays  $\Delta_j$  ((5)) is obtained and used to estimate the quality of the diffused time  $t_{M,i}$ . Note that a large  $\alpha$  value means that the diffused time  $t_{M,i}$  may have a larger error. Hence, the standard deviation  $\alpha$  is accumulated at every hop starting from the master node. This accumulated deviation value  $\beta_{M,k}$  is calculated as

$$\beta_{M,k} = \beta_{M,k-1} + \alpha \quad (7)$$

where  $k$  is the distance from the master node in terms of the number of hops.

The elected diffused leader nodes follow the same handshake procedure when they propagate the timing information message from Level  $k$  to Level  $(k + 1)$  as shown in Fig. 4 with the following modifications:

- The time  $t_{M,i}$  is calculated and only adjusted by  $\Delta/2$  as

$$t_{M,i} = t_{M,i} + \frac{\Delta}{2} \quad (8)$$

- The value  $n$ , the number of levels to be diffused, is decreased by one after each broadcast. A diffused leader node will not propagate the timing information message if the value  $n$  stored in the received timing information message is set equal to 0.
- The source LID in the timing information message is set to the LID of the diffused leader node.
- The value  $\beta_{M,k}$  is calculated by (7).
- The M-LID value stays the same in the timing information message.

Moreover, the master nodes diffuse a new timing information message every  $\delta$  seconds (i.e., time between two rounds) for a duration of  $\tau$  seconds to address the clock wandering and mobility of nodes. In addition, the timing information message handshake is repeated level-by-level by the elected diffused leader nodes as shown in Fig. 4.

Furthermore, a sensor node may receive multiple timing information messages from different master nodes. The information in these timing information messages are stored in a *timing data table* and used later by the algorithm TAA to calculate the new time for the node. If a node receives multiple timing information messages along different paths from the same master node during  $\delta$  period, only the information from the message that has the lowest  $t_{M,i}$  value is stored. In addition, a node may only diffuse the timing information message at most once per master node during the  $\delta$  period. As a result, only one entry per master node is stored in the timing data table. Since the timing data table only stores one entry per neighboring master node, the size of the timing data table should be very small, e.g., less than 10 entries.

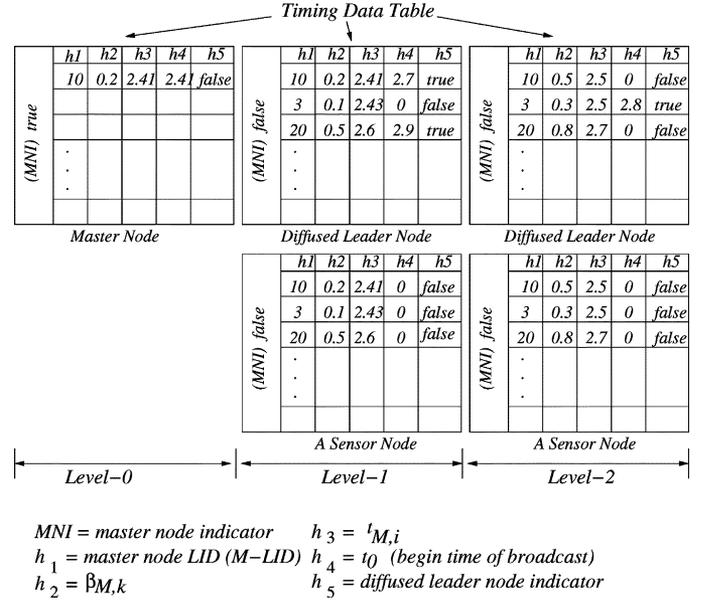


Fig. 5. Example of timing information stored in the nodes.

An example of the timing information stored in the timing data table is illustrated in Fig. 5. It shows the contents of a master node, Level 1 neighboring nodes, and Level 2 neighboring nodes. The details of the contents are described as follows:

**Level 0:** A master node sets the *master node indicator* (MNI) to true in the timing data table. Each row of the table contains the information related to the timing information message diffused by the master node. For example, the M-LID,  $\beta_{M,k}$ ,  $t_{M,i}$ , and  $t_0$  values are stored under columns  $h_1$  to  $h_4$  (e.g., M-LID = 10;  $\beta_{M,k} = 0.2$ ;  $t_{M,i} = 2.41$ ; and  $t_0 = 2.41$ ). In addition, column  $h_5$  is set to false since a master node can not be elected as a diffused leader node. Also, a master node should only have one row of data in the table, because it is diffusing its time.

**Level 1:** There may be many nodes, which are one hop away from the master nodes. Some of them may be elected diffused leader nodes while others are not. The contents of both categories of nodes are shown in Fig. 5. In addition, the MNI of both categories is set to false indicating that the nodes are not master nodes. The example illustrated in Fig. 5 shows that both categories of nodes receive three timing information messages from three master nodes, e.g., M-LID = 10; M-LID = 3; and M-LID = 20. The only differences between the contents of both categories are under columns  $h_4$  and  $h_5$ . For each received timing information message, a node may be elected as a diffused leader node. If it is elected, it is specified in column  $h_5$  with the value true (e.g., true for M-LID = 10 and M-LID = 20 while false for M-LID = 3 of the elected diffused leader node). If a node is not elected, column  $h_5$  is specified with the value false and  $h_4$  is set to 0 since the node is not diffusing any timing information message. As a result, a diffused leader node may have both false and true indicator values while a node elected not to diffuse

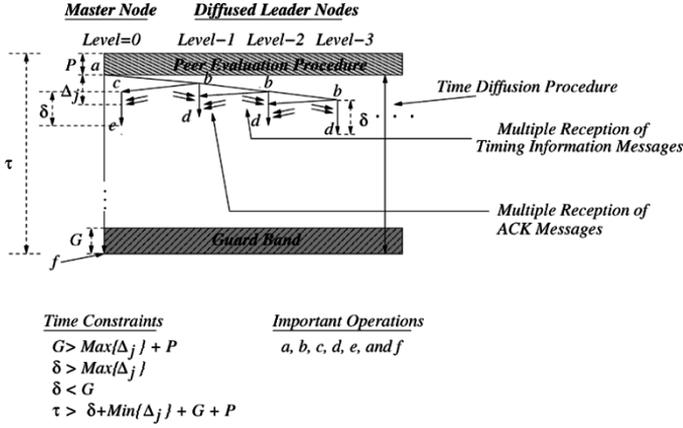


Fig. 6. Timing diagram of the timing information handshake.

any timing information message has only false values for all rows in column  $h_5$ .

**Level 2:** The method of storing information in the timing data table is the same as Level 1. Note that the values under columns  $h_2$  and  $h_3$  are different than the values in Level 1, because they are adjusted by (7) and (8). In addition, the diffused leader node indicators at column  $h_5$  are different, because a sensor node at Level 2 may be elected differently than a node in Level 1 to diffuse the timing information message, which originates from the same master node. For example, a diffused leader node at Level 2 is elected to diffuse the timing information message from  $M\text{-LID} = 3$  at time 2.8 s while the diffused leader node at Level 1 is not elected although both of these nodes receive the timing information message from  $M\text{-LID} = 3$ .

Since the timing information message handshake involves message exchanges as shown in Fig. 4, it is important to understand the time constraints between these message exchanges. As a result, the timing diagram of the timing information message handshake is described in the following section.

**2) Timing Diagram of the Timing Information Handshake:** The TDP consists of three procedures *ERP*, *PEP*, and *TP* as shown in Fig. 3. Since the procedure *ERP* requires a small amount of processing time comparing to procedures *PEP* and *TP*, it is not included in the timing diagram as shown in Fig. 6, which captures the timing relationship of events within  $\tau$  period. The procedure *PEP* given in Section III-A requires  $P$  seconds of processing time while the procedure *TP* occupies the rest of the  $\tau$  period. In the following paragraphs, the timing relationship of the procedure *TP* is described.

As shown in Fig. 6, the procedure *TP* consists of the operations  $a$ ,  $b$ ,  $c$ ,  $d$ ,  $e$ , and  $f$  as well as the *guard band*. The time constraints of the operations and *guard band* are described as follows:

- **Guard band:**  $G$  seconds long, where  $G > \text{Max}\{\Delta_j\} + P$ ; it prevents operations  $a$ ,  $b$ ,  $c$ , and  $e$  from occurring and generating events that may spill over to the next  $\tau$  period. The operations are initiated when an ACK message is received at  $\Delta_j$  ((5)) seconds after operation  $a$ .
- $b \rightarrow d$  and  $c \rightarrow e$ :  $\delta$  seconds long, where  $\delta > \text{Max}\{\Delta_j\}$  and  $\delta < G$ . This is to ensure that all timing information

and ACK messages are received within  $\delta$  seconds when diffusing a timing information message.

- **$\tau$  period:**  $\tau > \delta + \text{Min}\{\Delta_j\} + G + P$ ; it ensures that a timing information message is diffused for at least one round.

Within the  $\tau$  period, the operations  $a$  through  $f$  are performed during each round, which is kept tracked by the  $\theta$  counter, where  $\theta\tau$  is the TDP active period. The functionality of each operation is as follows:

**Operation a:** The master nodes broadcast timing information messages.

**Operation b:** The sensor nodes perform the procedure *ERP* detailed in Section III-C, which elects the diffused leader nodes as described in Section III-C. The elected diffused leader nodes send an ACK message and broadcast a timing information message as shown in Fig. 6. In addition, they initiate an operation  $d$  occurring  $\delta$  seconds later. Before operation  $d$  takes place, the diffused leader nodes may receive multiple ACK messages. Every time an ACK message is received, the round trip delay  $\Delta_j$  is measured using (5). In addition, the average  $\Delta$  and standard deviation  $\alpha$  of the round trip delays  $\Delta_j$  are calculated and stored in the timing data table as described in Section III-B1.

**Operation c:** The master nodes receive an ACK message from Level 1 diffused leader nodes and initiate operation  $e$  occurring  $\delta$  seconds later as shown in Fig. 6. Before executing operation  $e$ , multiple ACK messages may be received. As a result, the average  $\Delta$  and standard deviation  $\alpha$  of the round trip delays  $\Delta_j$  are calculated.

**Operation d:** The sensor nodes adjust their local clocks with the algorithm *TAA*. They also remove all rows in the timing data table. In addition, the  $\Delta_j$  values are cleared while the  $\alpha$  and  $\Delta$  values are kept, which are used to calculate  $\beta_{M,k}$  and  $t_{M,i}$  with (7) and (8), respectively.

**Operation e:** The master nodes clear the timing data table and initiate operation  $a$ .

**Operation f:** All sensor nodes in the sensor network reset their variables. For instance, the master node indicator (*MNI*) is set to true, and all the rows in the timing data table are cleared. In addition, the  $\Delta_j$  values are cleared while the  $\Delta$  and  $\alpha$  values are kept. Furthermore, the  $\theta$  counter is decreased by one, and the master nodes broadcast a *SYNCH* message containing the value of the  $\theta$  counter. The *SYNCH* message is intended for new sensor nodes that have been just added into the network. Once these new sensor nodes receive the *SYNCH* message, they set the  $\theta$  counter to the value specified in the *SYNCH* message. Only new sensor nodes that have received the *SYNCH* message can participate in becoming a master or diffused leader node with the procedure *ERP*. The rest of the new sensor nodes has to wait until it has received the *SYNCH* message that only occurs at every  $\tau$  seconds. Since the operation  $f$  is at the end of the  $\tau$  period, the procedure *ERP* is performed, which elects new master nodes for the next  $\tau$  period.

The above operations are carried out during each round of timing information message diffusion and continue until the  $\theta$

counter reaches 0. In the following section, the procedure *ERP* is described specifying how a master or diffused leader node is elected.

### C. Election/Reelection of Master/Diffused Leader Node Procedure (ERP)

As shown in Fig. 6, the diffused leader and master nodes are elected in operations *b* and *f*, respectively. Both types of elections depend on the outputs of the algorithms *FIA* and *LDA* to automatically self-configure the nodes as described in Sections III-C1 and III-C2. The master nodes are elected at the beginning of every  $\tau$  period to balance the workload of being master nodes and allow the network to reach an equilibrium time. The diffused leader nodes are elected every  $\delta$  period to balance the workload of being diffused leader nodes and ensure neighboring nodes receive the diffused timing information messages. The elections of both master nodes and diffused leader nodes at every  $\tau$  and  $\delta$  periods provide a robust mechanism that does not depend on specific sensor nodes to be operational.

1) *False Ticker Isolation Algorithm (FIA)*: The algorithm *FIA* uses the outlier ratio outputs ((2)) of the procedure *PEP* as described in Section III-A to self-determine if a node is a false ticker or not. If the average of these received outlier ratios is greater than the threshold  $\phi$ , the sensor node is an outlier. The value  $\phi$  controls the quality of the selected clocks. For instance, a small  $\phi$  value means that the selected clocks have small deviations with the clocks of the neighbor nodes. When the average outlier ratio is greater than 1, it means that the local clock deviates from the clocks of the neighbor nodes by more than twice the average Allan variance given by (3).

If a node self-determines to be an outlier, it is a false ticker. The false ticker does not become a diffused leader node during the current  $\tau$  period or a master node at the beginning of the next  $\tau$  period. The algorithm *FIA* aims to remove nodes that have high frequency noise clocks or high access fluctuation due to either network jitter or access variations from becoming master or diffused leader nodes. If a node is not a false ticker, then it uses the algorithm *LDA* as described in Section III-C2 to determine if it is elected as a master or diffuse leader node.

2) *Load Distribution Algorithm (LDA)*: Besides allowing the sensor network to achieve an equilibrium time, the TDP needs to be energy efficient and capable of distributing the energy consumption for diffusing time to all sensor nodes in the network. It achieves them by reelecting master and diffused leader nodes at every  $\tau$  and  $\delta$  seconds, respectively. During the reelection, the nodes randomly choose a value  $\lambda$  that is between 0 and 1. The value  $\lambda$  is then shifted by the value  $(1 - \zeta)$ , where  $\zeta$  is the ratio of current energy level over the maximum allowed energy level, and calculated as

$$\lambda = \lambda - (1 - \zeta) \quad (9)$$

If the value  $\lambda$  is greater than the threshold  $\varphi$ , then the node is either a master or diffused leader node depending if the master or diffused leader node is being reelected. The threshold  $\varphi$  determines the number of sensor nodes participating as a master or diffused leader node. For example, if  $\varphi$  is set equal to 0.7, it means on the average that 30 percent of the deployed sensor nodes is a master node or diffused leader node. As a result,

$\rho = 1 - \varphi$  represents the fraction of deployed sensor nodes that is a master or diffused leader node. For this case,  $\rho$  is set equal to 0.3.

Since the shifting of the randomly selected value  $\lambda$  is based on the current energy level of the sensor node,  $\rho$  decreases if the threshold  $\varphi$  is not adjusted appropriately. As a result, the threshold  $\varphi$  stored in all sensor nodes is adjusted at every  $\tau$  seconds according to

$$\varphi = \varphi - \varepsilon \quad (10)$$

where  $\varepsilon$  is the amount that needs to be adjusted, which is based on  $\mu$  (energy consumed per round of timing information message diffusion). The value  $\mu$  can be approximated by

$$\mu \approx \frac{\text{Amount of energy consumed during } \tau \text{ seconds}}{\lceil \frac{\tau}{\delta} \rceil - 1} \quad (11)$$

where  $\tau$  is the master node reelection period, and  $\delta$  is the time between each round of timing information message diffusion.

As a result, the value  $\varepsilon$  (see the Appendix) is calculated as

$$\varepsilon = \rho - \sum_{m=1}^i \Phi_{i,m} \rho^{m-1} (1 - \rho)^{(i-m)} (\rho - (m-1)\varepsilon) \quad (12)$$

where  $\rho$  is the fraction of sensor nodes that can become a master or diffused leader node;  $i$  is the number of rounds within a  $\tau$  period, which is approximated by  $\lceil \tau/\delta \rceil - 1$ ;  $\varepsilon$  is the ratio of  $\mu$  ((11)) over the maximum energy level; and the coefficient  $\Phi_{i,m}$  is calculated as

$$\Phi_{i,m} = \begin{cases} 1, & \text{for } m=1 \\ \sum_{j=1}^{i-1} 1, & \text{for } m=2 \\ \left( \sum_{v_{m-1}=1}^{i-(m-1)} \left( \sum_{v_{m-2}=1}^{i-(m-2)-v_{m-1}} \right. \right. \\ \left. \left. \times \left( \dots \left( \sum_{v_1=1}^{i-1-\sum_{k=1}^{m-2} v_{m-k}} 1 \right) \dots \right) \right) \right), & \text{for } m \geq 3 \end{cases} \quad (13)$$

with  $i \geq m$  and  $m - 1$  levels of summation for  $m \geq 3$ , e.g.,  $(\sum(\sum \cdot))$  and  $(\sum(\sum(\sum \cdot)))$  are 2 and 3 levels, respectively.

### D. Time Adjustment Algorithm (TAA)

As shown in Fig. 4, a sensor node at operation *d* adjusts its local clock with time  $t_{M,i}$  and deviation  $\beta_{M,k}$ , which are obtained from columns  $h_2$  and  $h_3$  of its timing data table. First, the node sums up all the deviations in column  $h_2$  of the timing data table, where  $\beta_T$  denotes the sum and set  $\mathbf{S}$  contains all the deviations  $\beta_{M,k}$ . In addition, set  $\mathbf{T}$  contains all the times  $t_{M,i}$  in column  $h_3$  of the timing data table, where  $|\mathbf{T}| = |\mathbf{S}|$ .

Let  $u$  be the  $u$ th element stored in sets  $\mathbf{S}$  and  $\mathbf{T}$ , where both  $u$ th elements in sets  $\mathbf{S}$  and  $\mathbf{T}$  are obtained from the same timing information message, which occupy the same row in the timing data table. For example,  $S_u$  and  $T_u$  are the  $u$ th elements in sets  $\mathbf{S}$  and  $\mathbf{T}$ , respectively.

As a result, the weight  $\omega_u$  for the diffused time  $T_u$  is determined as

$$\omega_u = \frac{\xi_u}{\xi_T} \quad (14)$$

where  $\xi_u$  (the unnormalized weight for the diffused time  $T_u$ ) is calculated as

$$\xi_u = \beta_T - S_u \quad (15)$$

and  $\xi_T$  (the normalizing factor for the weight  $\xi_u$ ) is determined as

$$\xi_T = \sum_{u=1}^{|\mathbf{S}|} \xi_u \quad (16)$$

The weight  $\omega_u$  is large when the deviation  $S_u$  is small. As a result, the sensor node uses more of the time diffused by a master node that is a hop away than two hops away. In addition, the effect of the asymmetric delay is lowered since the asymmetric delay is bounded by one-half of the round trip delay [12], and the round trip delay of one hop should be rather small, i.e., milliseconds order.

Once the weight  $\omega_u$  for each diffused time  $T_u$  is obtained from (14), it is used to calculate the new time  $t_{\text{new}}$  for the node. If the set  $\mathbf{T}$  is empty, then the new time is just the current local time ( $t_{\text{local}}$ ), without any change. If the set has only one element  $T_1$ , then the new time  $t_{\text{new}}$  is set equal to that element. Otherwise, all the elements in set  $\mathbf{T}$  are weighted by  $\omega$  ((14)), and they are summed up to provide the new time  $t_{\text{new}}$ . In summary,  $t_{\text{new}}$  is calculated as

$$t_{\text{new}} = \begin{cases} t_{\text{local}}, & \text{for } |\mathbf{T}| = 0 \\ T_1, & \text{for } |\mathbf{T}| = 1 \\ \sum_{u=1}^{|\mathbf{T}|} \omega_u \cdot T_u, & \text{for } |\mathbf{T}| \geq 2 \end{cases} \quad (17)$$

As the new time  $t_{\text{new}}$  is calculated by (17), the local clock is not updated with the new time if  $|t_{\text{local}} - t_{\text{new}}|$  is smaller than the average of the received Allan deviations, which are the outputs of procedure *PEP* as described in Section III-A. This is to prevent unnecessary updates to the local clock since the new time  $t_{\text{new}}$  is within the range of clock deviation among the neighbors.

#### IV. ANALYTICAL PERFORMANCE EVALUATION

It is important to show that TDP can allow the time in the sensor nodes to converge to an equilibrium time with a small variation that is equal to the round trip time  $v$  between two adjacent neighboring nodes. The value  $v$  consists of two components: 1) the processing delays and 2) the propagation delays. Since the propagation delays may be in the order of microseconds, the time precision between nodes is gated by the processing delays. As a result,  $v$  can be controlled by varying the processing delays, which consist of the medium access and queueing delays.

Each node is assumed to have received at least 2 timing information messages. The minimum  $\rho$  value ( $\rho_{\text{min}}$ ) required to satisfy this assumption is calculated as

$$\rho_{\text{min}} = \frac{l \cdot w}{\kappa \pi (R \cdot n)^2} \quad (18)$$

where  $l$  and  $w$  are the length and width of the sensor field, respectively;  $\kappa$  is the number of nodes deployed in the sensor field;  $R$  is the broadcast radius of a sensor node; and  $n$  is the number of levels that the timing information message is to be diffused.

Although  $\rho_{\text{min}}$  gives the minimum value of  $\rho$ ,  $\rho$  is best to be few times larger than  $\rho_{\text{min}}$ . This is to account for uneven distribution of nodes in the sensor field. By requiring the nodes to receive at least 2 timing information messages, the time in the sensor nodes can be diffused more effectively by using TDP.

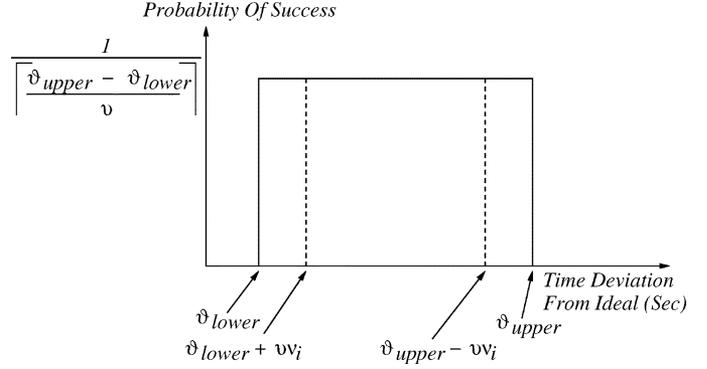


Fig. 7. Probability distribution of the deviated time from the ideal.

The time deviation from the ideal time is assumed to be uniformly distributed between lower bound  $\vartheta_{\text{lower}}$  and upper bound  $\vartheta_{\text{upper}}$ . The range between  $\vartheta_{\text{lower}}$  and  $\vartheta_{\text{upper}}$  is separated into discrete sections with step size of  $v$ . As a result, each section has the probability of  $1/[(\vartheta_{\text{upper}} - \vartheta_{\text{lower}})/v]$  as shown in Fig. 7. Since after each round of diffusion from the master nodes, the range between  $\vartheta_{\text{lower}}$  and  $\vartheta_{\text{upper}}$  shrinks. For the analysis, it is assumed that the range is shrunk by  $v \cdot \nu_i$  seconds from the upper bound  $\vartheta_{\text{upper}}$  and lower bound  $\vartheta_{\text{lower}}$ , where  $\nu_i$  is the shrink in multiples of  $v$  at Round ( $i$ ). After the range is shrunk, the  $\vartheta_{\text{upper}}$  and  $\vartheta_{\text{lower}}$  values are set to the new shrunk upper and lower bound values.

The probabilities  $Q_{L,i}$  and  $Q_{U,i}$  are the probabilities of shrinking by  $v \cdot \nu_i$  from the lower and upper bounds at Round ( $i$ ), respectively. Since the upper and lower bounds are assumed to shrink the same range,  $Q_{L,i}$  is equal to  $Q_{U,i}$  and is calculated as

$$Q_{L,i} = Q_{U,i} = \frac{\nu_i}{\left\lceil \frac{(\vartheta_{\text{upper}} - \vartheta_{\text{lower}})}{v} \right\rceil} \quad (19)$$

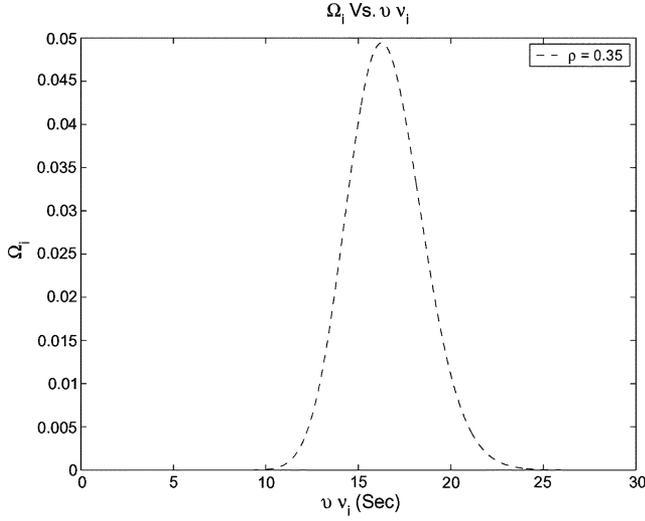
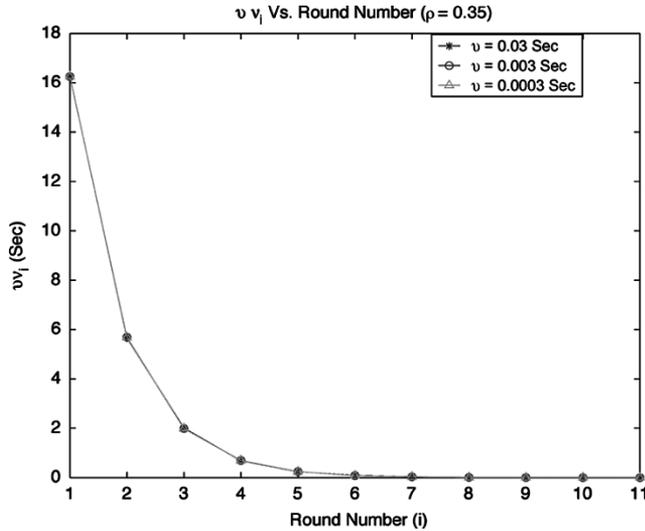
The objective of each round of timing information diffusion is to shrink the range of the deviated time distributed throughout the sensor network. The amount of shrinkage at Round ( $i$ ) is  $v \cdot \nu_i$ . As a result, the probability  $\Omega_i$  that all sensor nodes are shrunk by  $v \cdot \nu_i$  at Round ( $i$ ) is calculated as

$$\begin{aligned} \Omega_i &= \Omega_{(\vartheta_{\text{lower}} + v \cdot \nu_i) < t < (\vartheta_{\text{upper}} - v \cdot \nu_i)} \\ &= \binom{\kappa}{\lceil \rho \kappa \rceil} (1 - (Q_{L,i} + Q_{U,i}))^{\rho \kappa} (Q_{L,i} + Q_{U,i})^{(1-\rho)\kappa} \\ &= \binom{\kappa}{\lceil (1-\rho)\kappa \rceil} (Q_{L,i} + Q_{U,i})^{(1-\rho)\kappa} \\ &\quad \cdot (1 - (Q_{L,i} + Q_{U,i}))^{\rho \kappa} \end{aligned}$$

where  $\kappa$  is the number of nodes deployed in the sensor field;  $\rho$  is the fraction of nodes that will become master nodes or diffused leader nodes; and  $Q_{L,i}$  and  $Q_{U,i}$  are probabilities calculated by (19). Furthermore,  $\Omega_i$  can be approximated by the Poisson distribution if  $\kappa$  is large and  $\Lambda$  is small

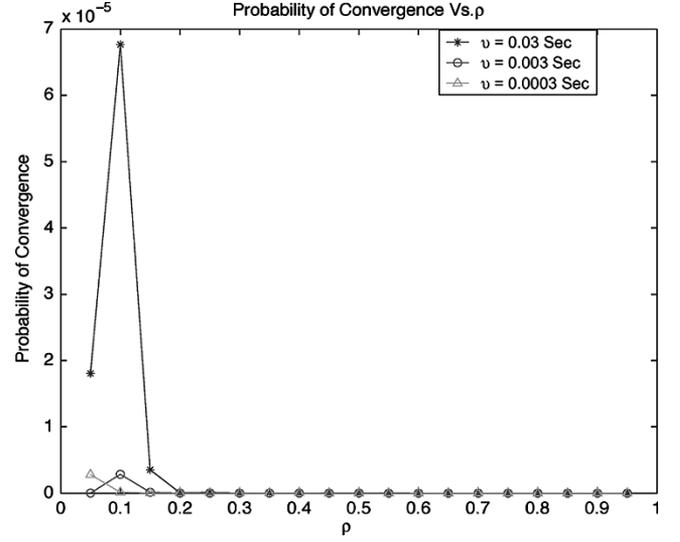
$$\Omega_i \simeq \frac{\Lambda^{\lceil (1-\rho)\kappa \rceil}}{\lceil (1-\rho)\kappa \rceil!} e^{-\Lambda}, \quad \text{where } \Lambda = Q_{L,i} + Q_{U,i}. \quad (20)$$

To show that the time distributed in the sensor network will converge to an equilibrium time, the probability given by (20) is plotted in Fig. 8 for  $\rho = 0.35$ ,  $\kappa = 100$ ,  $\vartheta_{\text{upper}} = 60$  s, and  $\vartheta_{\text{lower}} = 10$  s. The maximum value of  $\Omega_i$  occurs at around  $v \nu_i = 16$  s. From (20) and Fig. 8, there is a shrink range  $v \nu_i$  that

Fig. 8.  $\Omega_i$  versus  $v v_i$ .Fig. 9.  $v v_i$  versus diffusion round.

will give a maximum value of  $\Omega_i$  for each round of timing information diffusion. After each round, the upper and lower bounds of the deviated time, i.e.,  $\vartheta_{\text{upper}}$  and  $\vartheta_{\text{lower}}$ , are decreased by the shrink range  $v v_i$ . Once the deviated time range is shrunk, the probability distribution is still assumed equally distributed for the new range, and (20) is used to find the next best shrink range  $v v_i$  at Round ( $i$ ). This process is repeated for every round until the deviated time range is  $2v$  or less. The convergence of the proposed protocol based on this process is shown in Fig. 9. It represents the best case in convergence. Since the proposed protocol diffuses timing information message at each round to synchronize the neighboring nodes, the actual shrink range  $v v_i$  maybe different than the best case. Although with this difference, the TDP still converges but at a longer time since  $\vartheta_{\text{upper}}$  and  $\vartheta_{\text{lower}}$  are decreased at each round.

As shown in Fig. 9, the shrink range decreases exponentially as the number of rounds of timing information message diffusion increases. This means that the range of deviated time throughout the network is slowly reaching its equilibrium time. Note that the convergence does not depend on the  $v$  value, i.e.,

Fig. 10. Probability of convergence versus  $\rho$ .

the round trip time between nodes. This means that the time in the network can reach different level of precision, e.g., milliseconds or microseconds order. In reality, the attainable order of precision may only be in milliseconds, because the error budgets for processing and queuing delays in real systems are in the microseconds range. Also, the convergence exhibited by TDP only depends on the number of timing information message diffusion. As an example, the time takes 7 rounds to converge when  $v = 0.03$  s while 9 and 11 rounds when  $v = 0.003$  s and  $v = 0.0003$  s, respectively. Note that these number of rounds are obtained based on the best probability  $\Omega_i$  at each Round ( $i$ ). In addition, the analysis is based on  $\kappa = 100$ , i.e., the number of nodes being deployed, but it is also valid for higher values of  $\kappa$ .

The choice of  $\rho$  in (20) is also important. For  $\kappa = 100$ , the best  $\rho$  value with the highest probability of convergence is around 0.1 as shown in Fig. 10. The probability of convergence is calculated as the product of  $\Omega_i$  for  $i = 1$  to ( $i$  when  $v v_i$  is equal to  $2v$  or less). As  $\kappa$  increases to 200, the best  $\rho$  value shifts to around 0.3. In addition, the  $\rho$  value should be less than 0.5. Hence, half of the deployed nodes can be adjusted by the TDP since master nodes do not adjust their time although they receive the timing information messages from other master nodes.

## V. PERFORMANCE EVALUATION BY SIMULATION

The performance of the TDP is evaluated with an event driven simulation. Two hundred sensor nodes are deployed randomly in a 80 m by 80 m sensor field. Each of the sensor nodes can receive and transmit messages to its neighbors by executing the TDP independently, i.e., each sensor node is emulating a physical sensor node where it has its own memory. In addition, it keeps track of its own local time with a randomly selected drift rate that is between  $\pm 100$  ppm. Since each node keeps track of its local time, simulations with large number of nodes, e.g., 1000, 2000, and 3000, may become difficult. It is because the simulation has to create an event for every clock tick. As a result, only 200 sensor nodes are deployed with the targeted precision of  $10^{-1}$  seconds order. It is shown in Section IV that TDP will work for higher order of precision. To show that TDP is

TABLE I  
CONFIGURATION OF EACH SENSOR NODE

Parameters	Value
Transmission radius	10 meters
Available energy	1 J
Transmission cost	600 mW
Receiving cost	200 mW
Clock frequency	2 MHz
Clock fluctuation	within $\pm 100$ ppm
Transmission rate	1 Mbps
Signal propagation speed	$3 * 10^8$ meters/second
Processing delay (saturated)	0.05 seconds $\pm$ $N(0, 1)$ msec access fluctuation
Processing delay (not saturated)	0.01 seconds $\pm$ $N(0, 1)$ msec access fluctuation
Peer Evaluation Scan Message Length	98 bits
Peer Evaluation Reply Message Length	158 bits
Peer Evaluation Result Message Length	139 bits
Timing Information Message Length	152 bits
ACK Message Length	20 bits
Local ID Range	260
Mobility of Mobile Nodes	$8.3 * 10^{-4}$ meters/second
Mobility of Static Nodes	0 meters/second

able to reach its equilibrium time and maintain a small variation of the deviated time throughout the network, the local time of each sensor node is initially shifted by a random amount ranging from 10 s to 60 s from the ideal time. Since the local times of the neighbor nodes are quite different, this setup also shows how TDP recovers from network partitioning. In essence, this setup represents the worst case scenario in synchronizing time, where each node may be drifted far apart from each other.

When a node receives and transmits messages, it will consume power. It is assumed that a node does not go into the idle state while running the TDP since the nodes are active during the short period of time when TDP is running. All the nodes participate in TDP, and the timing information messages are diffused 3 hops from the master nodes for all simulations. The configuration of each node is listed in Table I, which has the parameters as in [8] but with energy set to 1 J.

As specified in Table I, the *processing delays* are 0.05 s and 0.01 s for saturated and not saturated sensor network, respectively. They are composed of delays incurred at the lower layers, i.e., medium access and physical layers. For example, a sensor node, which is equipped with an 802.11 MAC, may have a processing delay of around 0.01 s when the network is not saturated [13]; the delay value does not change for different node densities, e.g., 5, 10, and 20 nodes, and it is near constant until the network becomes saturated at around 75% of the channel capacity. At saturation, the processing delay of an outgoing message, the number of retransmission, and the end-to-end delay flatten at different values [3], [5], [13], [19] depending on the 802.11 MAC parameters, e.g., node density and congestion window size. The processing delay for a saturated sensor network is assumed to be 0.05 s [13].

In addition, the processing delays for both saturated and not saturated networks have access fluctuations that are normally

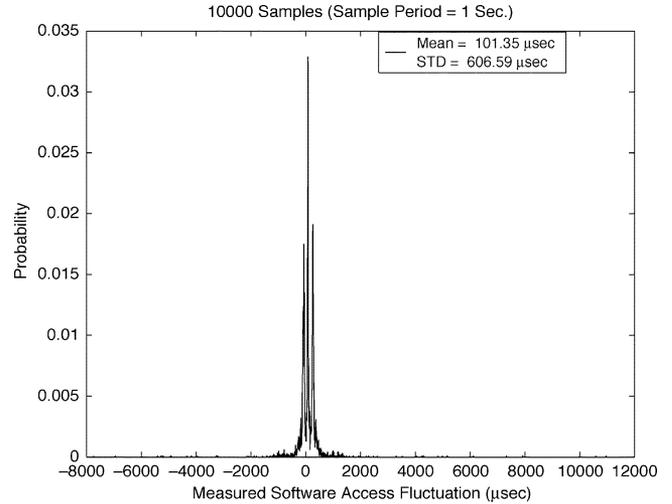


Fig. 11. Software access fluctuation.

distributed with mean and standard deviation of 0 and 1 ms. The access fluctuations are lumped values of both the medium access and software access fluctuations. As shown in Fig. 11, the software access fluctuation of a Sparc machine running Solaris operating system is around 600  $\mu$ s while the mean access time is around 100  $\mu$ s. Experiments are run to test the medium access fluctuation of 802.11 MAC by running Windows 98 operating system in Compaq Presario and Sony laptops. The round trip fluctuations consisting forward and reverse medium accesses as well as software accesses are between 1 and 2 ms. As a result, the medium access fluctuation is in the order of few hundred  $\mu$ s. Since sensor nodes are designed for the low-end regime, the medium access and software access may fluctuate even more. For the simulations, the lumped access fluctuations are normally distributed as given in Table I.

The performance of TDP is evaluated for both static and mobile sensor nodes by varying  $\tau$  and setting  $\delta$  and  $\rho$  values to 2 s and 0.3, respectively. First, the TDP is evaluated with and without the procedure *PEP* for both mobile and static nodes in Section V-A. In addition, the TDP is compared to TPSN to show its novelties. As described in [7], TPSN performs better than RBS in single hop as well as multiple hops. As a result, the TDP is compared to TPSN in a network-wide scenario with parameters given in Table I. Afterwards, both time convergence and energy dissipation of TDP are studied in depth for both static and mobile nodes in Section V-B.

#### A. Performance Comparison

1) *With/Without Peer Evaluation Procedure*: As shown in Fig. 12, the performance of TDP is evaluated for both with and without the procedure *PEP* for static and mobile nodes. The procedure *PEP* is designed to prevent the false tickers from participating in becoming master nodes. As the time throughout the sensor network converges, there is still a small time fluctuation within the network when the procedure *PEP* is not applied. This is illustrated in Fig. 12 as the converged time wiggles after 400 s. On the other hand, the converged time is stable when the procedure *PEP* is used.

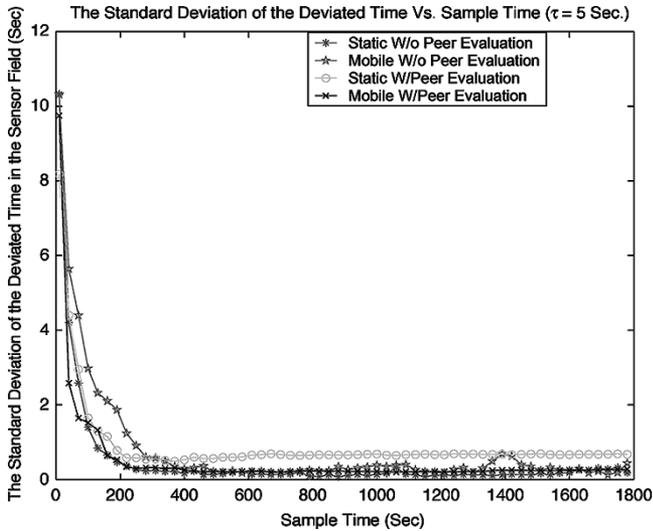


Fig. 12. Comparison of time convergence.

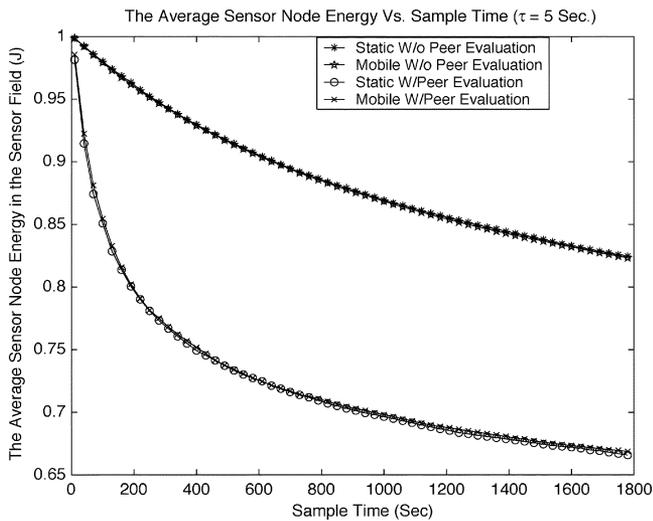


Fig. 13. Comparison of energy consumption.

Although the procedure *PEP* provides a cleaner time throughout the network, the amount of energy consumed when the TDP is used with this procedure exceeds the TDP without it as shown in Fig. 13. The TDP without the procedure *PEP* consumes 20% less energy, but the trade-off is allowing the time to fluctuate a little after convergence. If energy is more critical than time accuracy, the TDP without the procedure *PEP* may be a better choice. As a result, the performance of TDP without the procedure *PEP* is evaluated in detail in the following sections.

2) *TDP Versus TPSN*: The performance of TDP is compared with *Time-Sync protocol for Sensor Networks* (TPSN) [7] in a network-wide scenario to show how the diffusion process helps to synchronize the time in the network. For TPSN, there are three sinks trying to synchronize the network. After the nodes are synchronized, the histogram of the sensor nodes' time is calculated and shown in Fig. 14. There are three large islands of time occurring approximately at 30 s, 37 s, and 54 s. These islands of time are known to occur [7] when three sinks are used to synchronize the network. These islands of time may cause

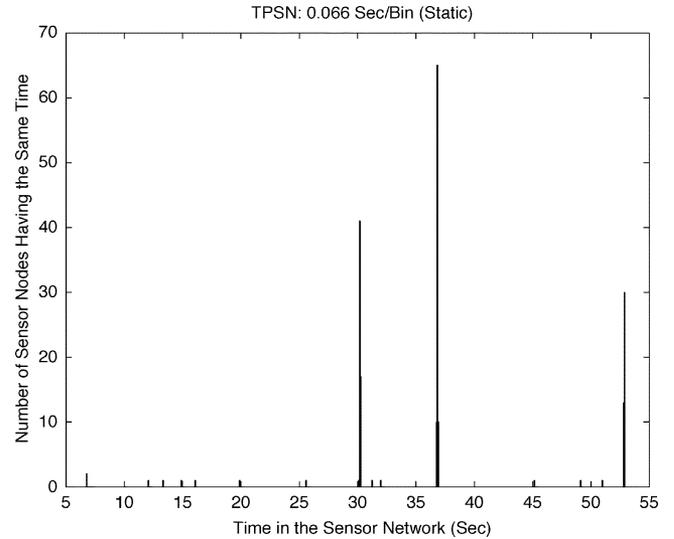


Fig. 14. TPSN: histogram of time distributed in the network (static nodes).

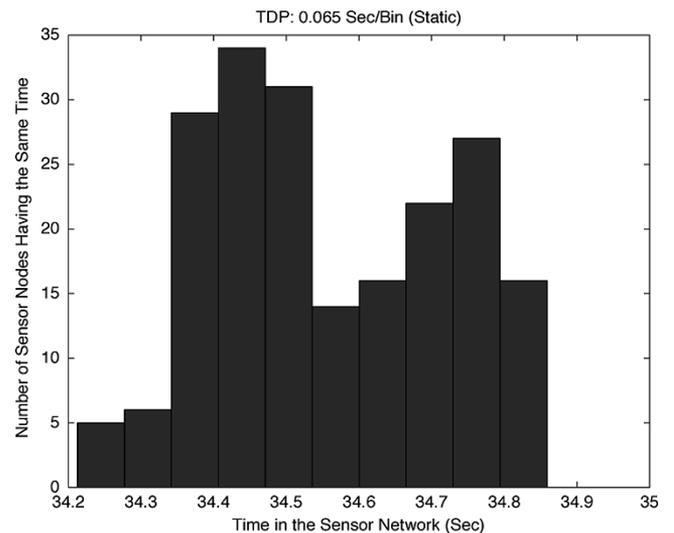


Fig. 15. TDP: histogram of time distributed in the network (static nodes).

problems when the users want all sensor nodes to perform a task at a specific time. Although most of the sensor nodes are synchronized to either one of the three sinks, there are still some nodes that remain unsynchronized. For example, some of the sensor nodes have time values that are within the range of 5 s and 27 s. This anomaly may be due to (1) the broadcast radius not being large enough and (2) the timing offset of synchronization messages between two levels in the hierarchy.

Under the same simulation scenario, the TDP is applied. Since the TDP does not depend on specific sensor nodes to be master nodes, it enables the network time to reach an equilibrium value by diffusion process. As shown in Fig. 15, the equilibrium time is around 34 s. The time variation throughout the network is around 0.6 s. This variation may be much tighter when the master nodes are synchronized to a time server.

When the sensor nodes are mobile, the TPSN exhibits more noise in the time throughout the network. Since TPSN synchronizes the nodes in the network hierarchically, the node movement breaks the hierarchy causing nodes to be unsynchronized.

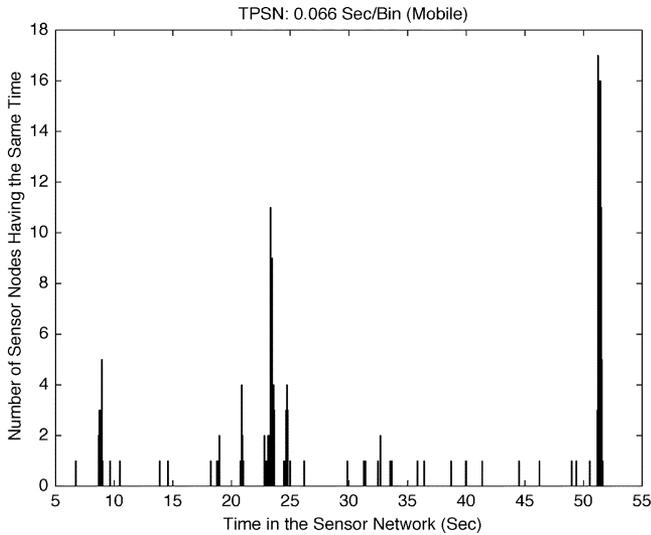


Fig. 16. TPSN: histogram of time distributed in the network (mobile nodes).

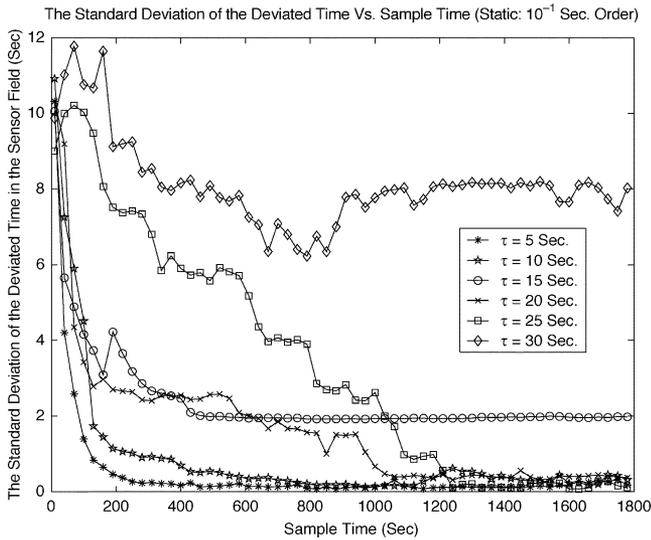


Fig. 17. The standard deviation of time for different  $\tau$  values (static nodes).

As shown in Fig. 16, there are still three islands of time but more nodes are becoming unsynchronized due to the movements. As for TDP, the movement does not affect the diffusion process. The time throughout the network still reaches an equilibrium value. A more detailed evaluation of TDP is given in the following section. The performance of TDP over time is evaluated for both static and mobile nodes.

**B. Performance Over Time**

1) *Static Sensor Nodes:* The design objective of  $\tau$  is to control the speed that the time in the sensor nodes reaches an equilibrium time. As shown in Fig. 17, the convergence of time is illustrated for different values of  $\tau$ . The standard deviation of the deviated time approaches  $10^{-1}$  s order for  $\tau = 5$  s and  $\tau = 10$  s. For  $\tau = 15$  s, the standard deviation flats out around 2 s. This is due to sensor nodes being topologically inaccessible with a broadcast radius of 10 m. As for  $\tau = 30$  s, the time in the sensor network fluctuates at a much faster rate than it can be synchronized.

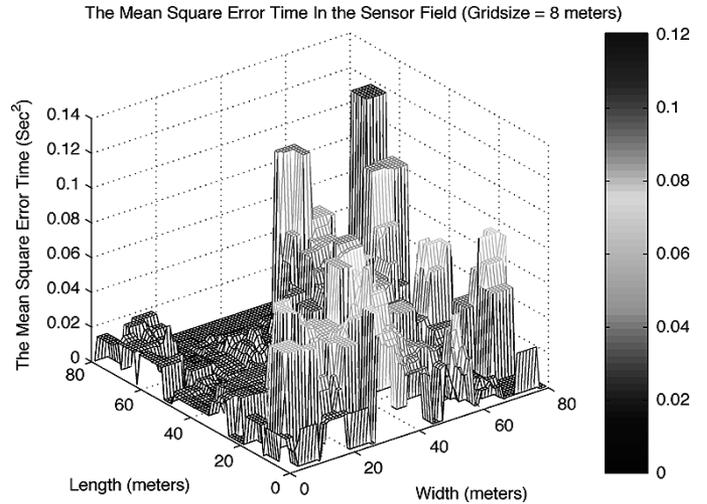


Fig. 18. The MSE time of the network for  $\tau = 5$  s (static nodes).

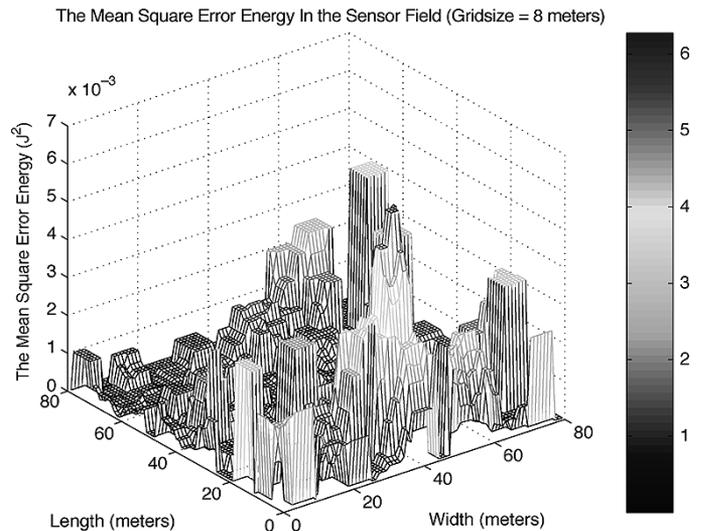


Fig. 19. The MSE energy of the network for  $\tau = 5$  s (static nodes).

The rate of the convergence depends on the  $\tau$  value being used. The time in the sensor network converges the fastest when  $\tau = 5$  s as shown in Fig. 17. This corresponds to the analytical performance evaluation. The convergence rate depends on the number of rounds of timing information message diffusion within a period of time. As a result,  $\tau = 5$  s gives the highest number of rounds of timing information message diffusion.

To further show that TDP is performing as it should be, a three-dimensional view of the mean square error (MSE) time distributed throughout the sensor field for  $\tau = 5$  s at simulation time 400 s is illustrated in Fig. 18. A grid size of 8 m by 8 m is used to scan the whole sensor field. The grid is shifted at 1 meter increment horizontally and vertically until the whole sensor field is covered. After each grid movement, the MSE time of nodes within the grid are calculated. The MSE of the time deviated from the average is illustrated in Fig. 18. It is in the  $10^{-2}$  s<sup>2</sup> order. In the sensor network, the time difference between neighbor nodes is important for some applications, e.g., speed tracking. As a result, a smooth transition of time throughout the sensor field is important. The TDP does

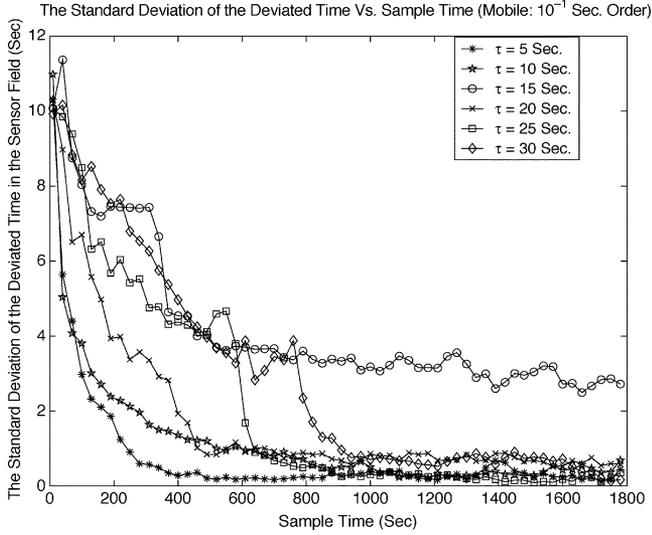


Fig. 20. The standard deviation of time for different  $\tau$  values (mobile node).

enable the time in the network to have a smooth transition as shown in Fig. 18. At location (40,40) on the sensor field, the smooth transition is shown more prominently.

A detailed view of the MSE energy in the sensor field for  $\tau = 5$  s at simulation time 400 s is illustrated in Fig. 19, where the energy variation is in  $10^{-3} J^2$  order. In addition, the energy is fairly distributed within the sensor field. The MSE time and energy is highest at location (80,80). This is because the sensor nodes may not be easily accessible.

2) *Mobile Sensor Nodes:* When the sensor nodes are mobile, the time difference between the neighbor nodes can be higher than when the nodes are static. As a result, the frequent change of positions and sharper time difference among neighbors cause the TDP to converge slower as shown in Fig. 20. As the  $\tau$  value increases, it takes a longer time to converge. In addition, the converged time is at a higher value for large  $\tau$  values. It is because the timing information message diffusion rate can not keep up with the mobility of the nodes. This suggests that the  $\tau$  value has to be small for high mobility or else the time in the network will have a hard time to converge. For instance, the time converges at around 400 s for  $\tau = 5$  s while it is at 1700 s for  $\tau = 30$  s as shown in Fig. 20. When  $\tau = 15$  s, the time converges to around 3 s. This is due to nodes not topologically accessible with a broadcast radius of 10 m.

The load of participating in the TDP is also distributed to all the nodes. The standard deviation of a sensor node energy in the sensor field is slowly approaching a constant value over time. It is not graphically illustrated due to space limitation. Regardless of the time in the network converges or not, the energy consumption of the nodes are fairly distributed.

## VI. CONCLUSION

The constraint of requiring the nodes to maintain a similar time among the neighbors and throughout the network at conditions where outside timing sources, e.g., high power stations used to discipline the local time of the nodes in the network, may not be available due to distance and location, e.g., inside a cave or under water. With this constraint in mind, we develop

the time-diffusion synchronization protocol (TDP) that allows the nodes in the sensor field to reach an equilibrium time with a small tolerance from each other. Also, we have analytically shown that TDP can be used to provide timing precision that is gated by the round trip delays among neighbor nodes. The convergence to the equilibrium time depends heavily on the rate of timing information message diffusion. This allows the designer to trade-off between convergence time and energy consumption. In addition, we have studied the TDP thoroughly for both static and mobile sensor nodes. In both scenarios, the TDP enables the time in the network to converge to the targeted tolerance. Also, the time differences among neighbor nodes are small allowing smooth transition of time throughout the network. Besides enabling the time to converge and reach the targeted precision, the tasks for this process is distributed among the nodes in the sensor field. An additional advantage of the TDP is that it allows the designer to choose different  $\tau$  values for different types of sensor networks depending on the purpose of the applications.

## APPENDIX

In order to determine  $\varepsilon$  (12), the fraction of deployed sensor nodes  $\psi_i$  that can become a diffused leader node at Round ( $i$ ) in Fig. 4 needs to be determined. In addition, the energy  $\mu$  consumed for each round of timing information message diffusion is required.

As a result,  $\zeta$  is reduced by  $\epsilon$ , i.e., ratio of  $\mu$  over the maximum allowed energy level, after each round of timing information handshake. The value  $\psi_i$  is determined as follows:

$$\text{Round } (i) : \psi_i = \sum_{m=1}^i \Gamma_{i,m} (\rho - (m-1)\epsilon)$$

At Round (1), the fraction of nodes that can become diffused leader nodes is set equal to  $\Gamma_{1,1}\rho$ , where  $\Gamma_{1,1}$  is the coefficient; at Round (2), the fraction of sensor nodes that did not participate as a diffused leader node has probability  $\rho$  of being reelected while the ones that participated has probability  $(\rho - \epsilon)$ , where  $\Gamma_{2,1}$  and  $\Gamma_{2,2}$  are the coefficients of  $\rho$  and  $\rho - \epsilon$ , respectively. The probability of being reelected is decreased by  $\epsilon$  for the elected sensor nodes, because the randomly selected value  $\lambda$  is reduced by  $(1 - \zeta)$  [(9)], where  $\zeta$  is reduced by  $\epsilon$ . By repeating the same evaluation at each round, a pattern emerges and gives the equation for  $\psi_i$ . The value  $\Gamma_{i,m}$  is the  $m$ th coefficient at Round ( $i$ ). For example,  $\Gamma_{4,3}$  is the third coefficient of Round (4), which is the coefficient of  $(\rho - 2\epsilon)$ .

The coefficient  $\Gamma_{i,m}$  represents the fraction of nodes that has probability  $(\rho - (m-1)\epsilon)$ , and it is derived as follows:

$$\Gamma_{i,1} = (1 - \rho)^{(i-1)}$$

$$\Gamma_{i,m} = \Gamma_{i-1,m}(1 - \rho) + \Gamma_{i-1,m-1}(\rho)$$

$$\text{Note : } \Gamma_{i,m} = 0, \text{ for } i < m, m > 1$$

The coefficient  $\Gamma_{i,1}$  is decreased by the fraction of nodes that is selected to be diffused leader nodes at every round. As a result, only  $(1 - \rho)$  of the previous fraction of nodes will have probability  $\rho$ . The coefficient  $\Gamma_{i,2}$  depends on the fraction of nodes that has not been selected to be diffused leader node  $\Gamma_{i-1,2}(1 - \rho)$  and the fraction of nodes that has been selected to be diffused leader node  $\Gamma_{i-1,1}(\rho)$ . Basically,  $\Gamma_{i,m}$  is composed of  $\Gamma_{i-1,m}$  and  $\Gamma_{i-1,m-1}$ . This means that diffused leader

nodes move from having probability  $(\rho - (m - 2)\epsilon)$  to probability  $(\rho - (m - 1)\epsilon)$  since their chances of being reelected as diffused leader nodes at the next round are decreased by  $\epsilon$ . The general form for the  $m$ th coefficient at Round ( $i$ ) is given as  $\Gamma_{i,m}$ .

The  $\Gamma_{i,m}$  coefficients can be further simplified in terms of  $\rho$  and  $i$ . They are given as follows:

$$\Gamma_{i,1} = \Phi_{i,1}(1 - \rho)^{(i-1)}$$

$$\Gamma_{i,m} = \Phi_{i,m}(\rho)^{m-1}(1 - \rho)^{(i-m)}$$

$$\text{Note : } \Phi_{i,m} = 0, \text{ for } i < m, m > 1$$

where the coefficient  $\Phi_{i,m}$  is calculated as

$$\Phi_{i,m} = \begin{cases} 1, & \text{for } m=1 \\ \sum_{j=1}^{i-1} 1, & \text{for } m=2 \\ \left( \sum_{v_{m-1}=1}^{i-(m-1)} \left( \sum_{v_{m-2}=1}^{i-(m-2)-v_{m-1}} \right. \right. \\ \left. \left. \times \left( \dots \left( \sum_{v_1=1}^{i-1-\sum_{k=1}^{m-2} v_{m-k}} 1 \right) \dots \right) \right) \right), & \text{for } m \geq 3 \end{cases} \quad (21)$$

with  $i \geq m$  and  $m - 1$  levels of summation for  $m \geq 3$ , e.g.,  $(\sum(\sum \cdot))$  and  $(\sum(\sum(\sum \cdot)))$  are 2 and 3 levels, respectively.

As a result, the value  $\psi_i$  and  $\epsilon$  are calculated as

$$\psi_i = \sum_{m=1}^i \Phi_{i,m}(\rho)^{m-1}(1 - \rho)^{(i-m)} (\rho - (m - 1)\epsilon) \quad (22)$$

$$\epsilon = \rho - \psi_i$$

$$= \rho - \sum_{m=1}^i \Phi_{i,m} \rho^{m-1} (1 - \rho)^{(i-m)} (\rho - (m - 1)\epsilon) \quad (23)$$

where  $\rho$  is the fraction of sensor nodes that can become a master or diffused leader node;  $\Phi_{i,m}$  is calculated by (21);  $i$  is the number of rounds within a  $\tau$  period, which is approximated by  $\lceil \tau/\delta \rceil - 1$ ;  $\epsilon$  is the ratio of  $\mu$  (11) over the maximum energy level.

#### ACKNOWLEDGMENT

Special thanks to four referees for their outstanding detailed constructive feedbacks, which greatly helped to revise and improve the earlier version of this paper.

#### REFERENCES

- [1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey?" *Computer Networks (Elsevier) J.*, vol. 38, no. 4, pp. 393–422, Mar. 2002.
- [2] D. Allan, "Time and frequency (time-domain) characterization, estimation, and prediction of precision clocks and oscillators," *IEEE Trans. on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 34, no. 6, pp. 647–654, Nov. 1987.
- [3] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE J. Select. Areas Commun.*, vol. 18, no. 3, pp. 535–547, Mar. 2000.
- [4] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "SPAN: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," in *Proc. ACM MobiCom*, Rome, Italy, 2001, pp. 85–96.
- [5] B. P. Crow, I. Widjaja, J. G. Kim, and P. Sakai, "Investigation of the IEEE 802.11 Medium Access Control (MAC) sublayer functions," in *Proc. IEEE INFOCOM*, Kobe, Japan, Apr. 1997, pp. 126–133.
- [6] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *Proc. 5th Symp. Operating Systems Design and Implementation (OSDI 2002)*, Boston, MA, Dec. 2002.

- [7] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," presented at the ACM SenSys 2003, Los Angeles, CA, Nov. 2003.
- [8] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proc. ACM MobiCom*, Seattle, WA, 1999, pp. 174–185.
- [9] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *IEEE Proc. Hawaii Int. Conf. System Sciences*, Jan. 2000, pp. 1–10.
- [10] *IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, 2002.
- [11] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE Trans. Networking*, vol. 11, no. 1, pp. 2–16, Feb. 2003.
- [12] J. Levine, "Time synchronization over the internet using an adaptive frequency-locked loop," *IEEE Trans. Ultrason., Ferroelectr., Freq. Contr.*, vol. 46, no. 4, pp. 888–896, Jul. 1999.
- [13] J. Liu, D. M. Nicol, L. F. Perrone, and M. Liljenstam, "Toward high performance modeling of the 802.11 wireless protocol," in *Proc. 2001 Winter Simulation Conf. (WSC 2001)*, Arlington, VA, Dec. 2001.
- [14] D. L. Mills, "Internet time synchronization: the network time protocol," in *Global States and Time in Distributed Systems*, Z. Yang and T. A. Marsland, Eds. New York: IEEE Computer Society Press, 1994.
- [15] —, "Adaptive hybrid clock discipline algorithm for the network time protocol," *IEEE/ACM Trans. Networking*, vol. 6, no. 5, pp. 505–514, Oct. 1998.
- [16] "Network Time Protocol (Version 3) Specification, Implementation, and Analysis," Network Working Group, RFC 1305, Mar. 1992.
- [17] W. Su and I. F. Akyildiz, "The jitter time stamp approach for clock recovery of real-time VBR traffic," *IEEE/ACM Trans. Networking*, vol. 9, no. 6, pp. 746–755, Dec. 2001.
- [18] —, "Time Translation Algorithm," Georgia Tech, BWN Lab Report, Apr. 2003.
- [19] Y. C. Tay and K. C. Chua, "A capacity analysis for the IEEE 802.11 MAC protocol," *ACM/Kluwer Wireless Networks (WINET) J.*, vol. 7, pp. 159–171, 2001.



**Weilian Su** (M'00) received the B.S. degree in electrical and computer engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1997, and the M.S. and Ph.D. degrees in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, in 2001 and 2004, respectively. During his Ph.D., he did research under the guidance of Dr. I. F. Akyildiz in the Broadband and Wireless Networking Laboratory.

He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA. His current research interests include sensor networks, InterPlaNetary networks, ad hoc networks, Internet QoS, distributed networks, and satellite networks.

Dr. Su has been a member of ACM since 2000. In 2003, he received the IEEE Communications Society 2003 Best Tutorial Paper Award.



**Ian F. Akyildiz** (M'86–SM'89–F'95) received the B.S., M.S., and Ph.D. degrees in computer engineering from the University of Erlangen-Nuernberg, Erlangen, Germany, in 1978, 1981, and 1984, respectively.

He is the Ken Byers Distinguished Chair Professor with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, and Director of Broadband and Wireless Networking Laboratory. His current research interests are in Sensor Networks, InterPlaNetary Internet, Wireless

Networks and Satellite Networks. He is an Editor-in-Chief of *Computer Networks* and the *Ad Hoc Networks Journal*.

Dr. Akyildiz is a Fellow of the ACM. He served as a National Lecturer for ACM from 1989 to 1998. He has received the ACM Outstanding Distinguished Lecturer Award for 1994, the 1997 IEEE Leonard G. Abraham Prize of the IEEE Communications Society, the 2002 IEEE Harry M. Goode Memorial Award, the 2003 IEEE Best Tutorial Award from the IEEE Communications Society, and the 2003 ACM SIGMOBILE Award.