On the Placement of Clock Reference Nodes for Time Synchronization in Sensor Networks

Yong Liao, Lixin Gao
Department of Electrical & Computer Engineering
University of Massachusetts
Amherst, MA 01003, USA
Email:{yliao, lgao}@ecs.umass.edu

***Abstract*— The limited resource of sensor nodes requires light-weight and efficient schemes to synchronize clocks on different nodes. Network-wide time synchronization is an even more challenging problem because a sensor network usually has a large number of sensor nodes. In this paper, we consider the problem of synchronizing the clocks of all nodes in large scale sensor networks. Making all sensor nodes synchronize with one root node is limited in synchronization accuracy. We assume that multiple highly accurate clock reference nodes can be placed in the network. The clock reference nodes have some external means such as GPS to make sure they have the accurate time. Sensor nodes will adjust their system time according to the time of those clock reference nodes. The network-wide synchronization accuracy depends on where to place the clock reference nodes in the network. We formulate this placement problem to a K-Median problem, which has been proven to be NP-Complete. To make this problem tractable, we introduce simplifications to the problem and solve it using integer programming. Our numerical experiment results show that properly placing multiple clock reference nodes in the network can greatly improve the network-wide synchronization accuracy, especially for those networks where sensor nodes are unevenly distributed. In reality, the clock reference nodes may not be placed in the exact points where they should be. Our simulations show that even the placement of clock reference nodes has some deviations, the global synchronization still has good accuracy.**

I. Introduction

The distributed nature of sensor networks requires fine-grained coordinations among sensor nodes [1]. The functions of many layers in sensor networks depend on the time synchronization between different sensor nodes, such as data fusion, TDMA scheduling, localization, and power-saving duty cycling. Each sensor node, such as the mote [2], has a physical real time clock which is triggered by a quartz oscillator. The clocks on different nodes usually have different values and we need to synchronize the clocks on different sensor nodes.

Synchronizing two neighboring nodes has been extensively studied. Many schemes have been proposed in literature [3] [4] [5] [6] [7]. How to synchronize sensor nodes multi-hop away is a more challenging problem. Sophisticated schemes such as NTP [8] cannot be used because the sensor nodes must be small-size, low-cost, and power efficient. One feasible solution to synchronize two nodes multi-hop away is synchronizing all intermediate nodes one by one.

The majorities of previous works on network-wide/global time synchronization [4] [6] [7] assume there exists one accurate clock reference node and a spanning tree rooted at that node is built to synchronize all nodes along the edges of the tree in the hop-by-hop manner. Synchronizing two neighboring nodes is the basic operation in synchronizing two nodes multi-hop away. But we should notice that existing schemes for synchronizing two neighboring nodes are not perfect. There are always some inaccuracies existing. A potential problem of synchronizing two nodes multi-hop away in the hop-by-hop manner is that the inaccuracies of synchronizing every two neighboring nodes can be accumulated and the synchronization inaccuracies between two nodes multi-hop away can be quite large. Therefore, building one single tree to synchronize all nodes in the network has a limit in accuracy, because some nodes may be far away from the tree root.

Besides making the synchronization between two neighboring nodes more accurate to achieve fine global synchronization, there is another straightforward but efficient way to overcome the potential problem of global synchronization using the hop-by-hop method. We can deploy multiple clock reference nodes in the network and make each sensor node synchronize with the closest one. The clock reference nodes are special nodes which have the accurate time, such as being equipped with GPS receivers, or being synchronized before deploying them in the network. By placing multiple clock reference nodes to proper locations, sensor nodes can be closer to one of those clock reference nodes than the single spanning tree-based schemes. So all sensor nodes in the network can be more accurately synchronized. The challenge here is where to place the clock reference nodes, because the locations of the clock reference nodes can greatly affect the synchronization accuracy. We plot an example in Fig. 1 to demonstrate how the number and the placement of the clock reference nodes can affect the network-wide time synchronization.

We follow the assumption in [3] that the local clock value of each sensor node is a linear function of the actual time. Synchronizing two clocks includes measuring the skew and offset. We also assume there are some mechanisms by which the clocks of two neighboring nodes can be synchronized, as the schemes proposed in [3] [4] [5] [6]. The inaccuracies of measuring the skew and offset between two neighboring nodes can be modeled as zero mean Gaussian random variables. When a node multi-hop away from the clock reference nodes synchronizes his clock to the clock reference node in the hop-by-hop manner, the means of the skew and offset measurement

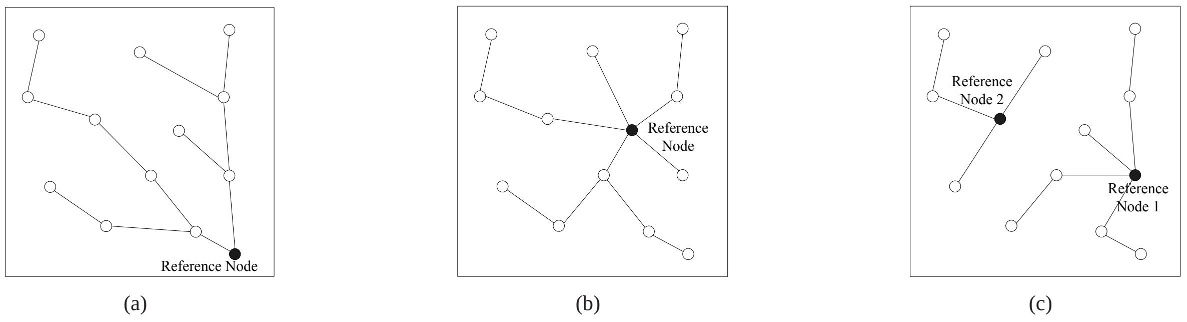


Fig. 1. A simple example demonstrates how the placement of clock reference nodes can affect the global time synchronization accuracy. The solid nodes represent the clock reference nodes. Intuitively, the closer a node is to the clock reference node, the more accurate its time synchronization can be. So we use the sum of all nodes' number of hops away from the closest clock reference node as the metric of synchronization accuracy. In (a), the sum of all nodes' number of hops away from the solid node is 28. If we change the location of the clock reference node as shown in (b), the sum is 22. If we place two clock reference nodes like (c), the sum can be reduced to 15.

inaccuracies are zero, but the variances of measuring the skew and offset are related with the number of hops between them. Intuitively, the farther two nodes are away from each other, the larger the inaccuracies of the skew and offset will be, because the inaccuracies of every one-hop synchronization can be accumulated. In Section III we will show that the variances are directly proportional to the number of hops.

Since we consider the network-wide time synchronization, we have to take all sensor nodes into account. We consider a scenario here that all sensor nodes have equal importance, so the objective of our time synchronization is defined as minimizing the *sum* of all sensor nodes' inaccuracies. The locations of the clock reference nodes are essential to the network-wide synchronization accuracy. We formulate the clock reference node placement problem to a K-median problem, which is NP-Complete. In order to make this problem tractable, we simplify the problem by partitioning the network into grids and treat each grid as a weighted vertex. The vertices' weights are the sensor node deployment densities or the number of nodes in those grids. Besides, we use the Euclidean distance between two vertices to approximate the number of hops between nodes within those two grids. After these simplifications, an 0-1 integer programming is used to describe the clock reference nodes placement problem and we solve it by a branch-and-bound technique.

Extensive simulation experiments have been conducted to test the impact of those simplifications and the network-wide synchronization accuracy in case of using multiple clock reference nodes. The experimental results show that partitioning the network into grids and using Euclidean distance to approximate the number of hops have little impact to the synchronization accuracy and the network-wide time synchronization accuracy can be greatly improved by deploying multiple clock reference nodes in the network, especially for those networks where sensor nodes are unevenly distributed. In reality, we may not be able to deploy the clock reference nodes to their exact locations. We will show by simulations that even the deploying locations of the clock reference nodes cannot be precisely controlled, the synchronization accuracy will not degrade too much.

The rest of this paper is organized as followings: Section II presents a brief overview of existing works on time synchronization in sensor networks. In Section III we will analyze the source of inaccuracies when synchronizing a node multi-hop away from the clock reference node. We formulate the clock reference nodes placement problem in Section IV and solve it using 0-1 integer programming. Section V presents the numerical experiment results. Section VI concludes this paper.

II. Background and Related Works

There have been many research activities in time synchronization for wireless sensor networks. For message exchanging based synchronization schemes, the inaccuracy comes mainly from the measurement of various delays in sending and receiving messages, including send delay, MAC access delay, transmission delay, propagation delay, reception delay, and receive delay [4]. The measurement or estimatation of each component can have inaccuracy.

A. Synchronization Two Neighboring Nodes

In [3] the authors propose a sender-based deterministic scheme. The local time of a node is modeled as a linear function of the actual time and the synchronization involves how to find the slope and y-intercept of the linear fucntion by measuring message delays. In order to accurately measure variance delays in sending and receiving synchronization messages, the MAC layer time stamping is proposed in [7]. The basic idea is using the exact time when the MAC layer sends or receives frames as the time stamps for messages. The RBS (Reference-Broadcast Synchronization) protocol [5] is a receiver-based scheme for synchronizing nodes within the same broadcast region. Two nodes within the same broadcast region use the same broadcast message from a third-party node as a beacon to synchronize their clocks.

B. Network-wide Time Synchronization

The TPSN (Timing-sync Protocol for Sensor Networks) protocol is proposed in [4], which aims at providing network-wide synchronization. It first establishes a hierarchical structure with a root having the standard time and then a pair wise synchronization is performed along the edges of this structure to build a global timescale throughout the networks. The authors of [7] propose using hierarchical broadcasting to provide network-wide synchronization. The root node broadcasts messages to synchronize its neighbors. After those neighbors are synchronized, they broadcast timing message as well to get other nodes synchronized. A *post-facto* scheme is proposed in [9]. The idea is using RBS to synchronize nodes hop-by-hop only when an event occurs. In [10], the author propose not to synchronize the local clocks of different nodes but transform time stamps using unsynchronized local clocks. As a message propagates from source to destination, all intermediate nodes know the time differences between the previous hop. After the destination node receives the message, it can compute the time difference between the source and the destination nodes.

C. How Our Work is Different from Previous Works

Our work is different from previous work in that instead of focusing on improving the synchronization of two neighboring nodes, we choose to place multiple clock reference nodes in the network. By carefully selecting the deploying locations of the clock reference nodes, we can improve the synchronization accuracies of all sensor nodes without too much cost. We formulate the placement problem to a K-median problem and solve it using reasonable heuristics and integer programming. To our knowledge, our work is the first one on placing multiple clock reference nodes to achieve accurate time synchronization in sensor networks.

III. The Inaccuracies of Global Synchronization

In this section, we will show that the inaccuracy of synchronizing the clock of a node multi-hop away from the clock reference node is directly proportional to the number of hops between them. Similar to the work in [3], we assume the local clock value of each node is a linear function of the actual time. Let t_i denote the local time of node i and t is the time of the clock reference node, which is supposed to be the actual time. The relation between t_i and t can be shown in (1), where s_i is called the *skew* of node i's clock and o_i is the *offset*, as referred to the clock reference node.

$$t_i = s_i \cdot t + o_i \tag{1}$$

Once node i has known s_i and o_i, node i is said to be synchronized to the clock reference node. If the current reading of his clock is t_i, node i should adjust his system time to $\frac{(t_i - o_i)}{s_i}$.

A. The Inaccuracy of Synchronizing Two Neighboring Nodes

Synchronizing two neighboring nodes is the basic operation in synchronizing two nodes multi-hop away in the hop-by-hop manner. For some reasons, the synchronization of two neighboring nodes cannot be perfect. The measurement of both the skew and the offset may have inaccuracies. When we adjust the clock of node j according to the clock of another node i who has been synchronized and is closer to the clock reference node, the skew and offset of node j should be s_j and o_j but the measurement results are $s_j + \Delta_{s,j}$ and $o_j + \Delta_{o,j}$, respectively.

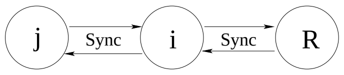


Fig. 2. A simple example shows how a node two-hop away from the clock reference node synchronizes his clock.

So the time value of node j we get from measurement is as the following:

$$t_j = (s_j + \Delta_{s,j})t_i + (o_j + \Delta_{o,j}) \tag{2}$$

In (2), t_j is the time value of node j and t_i is the time value of the node i, which is supposed to be the accurate time. $\Delta_{s,j}$ and $\Delta_{o,j}$ are the inaccuracies of measuring s_j and o_j. We assume $\Delta_{s,j}$ and $\Delta_{o,j}$ are IID (Independent and Identical Distribution) Gaussian random variables, whose means are zeros and variances are σ_s^2 and σ_o^2.

B. The Inaccuracy of Synchronizing Two Nodes Multi-hop away from Each Other

Suppose node j is two-hop away from the clock reference node R and the intermediate node between them is node i, as shown in Fig. 2. Node i first synchronizes his clock to the clock reference node R; then j adjusts his clock by synchronizing to i. Generally, each node among a path from node j to the clock reference node synchronizes his clock to the neighboring node which is closer to the clock reference node and has already been synchronized. For the simplicity in mathematics, we assume the measurements of skew and offset are independent processes. If the synchronization between node i and R occurs shortly before node j synchronizes with node i, we can ignore the additional offset between node i and R because of those two clocks' rate difference. Therefore, the measurement of the offset between node j and R's clock is $(s_i + \Delta_{o,i} + s_j + \Delta_{o,j})$, so the inaccuracy is $(\Delta_{o,i} + \Delta_{o,j})$.

Now we consider the measurement inaccuracy of the skew. Suppose that according to the synchronization process of two neighboring nodes, the frequency of node i's clock is $(s_i + \Delta_{s,i})$ times faster than node R's clock and node j's clock is $(s_j + \Delta_{s,j})$ times faster than node i's clock, we can know that the clock frequency of node j is $(s_i + \Delta_{s,i})(s_j + \Delta_{s,j})$ times faster than node R's clock [1].

In general, for a node n hops away from the clock reference node, the measured offset O_n and skew S_n should be two random variables as follows:

$$\begin{cases} O_n = \sum_{i=1}^{n}(o_i + \Delta_{o,i}) \\ S_n = \prod_{i=1}^{n}(s_i + \Delta_{s,i}) \end{cases}$$

The random variable O_n is the sum of n independent Gaussian random variables. We can write the mean and variance of random variable O_n as the following:

$$\begin{cases} E(O_n) = \sum_{i=1}^{n} o_i \\ Var(O_n) = n\sigma_o^2 \end{cases} \tag{3}$$

[1]For simplicity, we do not count in the offset here. Since $t_i = (s_i + \Delta_{s,i})t_R$ and $t_j = (s_j + \Delta_{s,j})t_i$, we have $t_j = (s_i + \Delta_{s,i})(s_j + \Delta_{s,j})t_R$.

The random variable S_n is the product of n independent Gaussian random variables. The mean and variance are:

$$\begin{cases} E(S_n) = \prod_{i=1}^{n} s_i \\ Var(S_n) = E(S_n^2) - E(S_n)^2 \end{cases} \tag{4}$$

Because $\Delta_{s,i}$ are zero mean independent random variables, the first term of $Var(S_n)$ in (4) is equal to $\prod_{i=1}^{n} E(s_i^2 + 2s_i\Delta_{s,i} + \Delta_{s,i}^2) = \prod_{i=1}^{n}(s_i^2 + \sigma_s^2)$. The second term of $Var(S_n)$ in (4) can be written as $\left(\prod_{i=1}^{n} E(s_i + \Delta_{s,i})\right)^2 = \prod_{i=1}^{n} s_i^2$. Then we can write the variance of measuring the clock skew of two node n hops away as

$$Var(S_n) = \prod_{i=1}^{n}(\sigma_s^2 + s_i^2) - \prod_{i=1}^{n} s_i^2 \tag{5}$$

We assume $s_i \approx 1$ and $\sigma_s \ll 1$, which means the clock of each sensor node has only little drift and the measurement of the skew between two neighboring nodes' clocks has considerable accuracy. Then we can conclude that the variance of adjusting the skew is directly proportional to the number of hops from the clock reference node, as shown in (6).

$$Var(S_n) \approx \prod_{i=1}^{n}(\sigma_s^2 + 1) - 1 = n\sigma_s^2 + o(\sigma_s^2) \approx n\sigma_s^2 \tag{6}$$

From (3) and (4) we see that the means of offset and skew have no inaccuracy, if the inaccuracies of measuring the skew and offset of two neighboring nodes are modeled by zero mean Gaussian random variables. On the other hand, (3) and (6) show that the variances of measuring the skew and offset of two nodes multi-hop away are determined by the number of hops between them and the variances of measuring the skew and offset of two neighboring nodes' clocks. We use the variances of synchronizing a node's clock skew or offset according to a clock reference node as the metrics of the inaccuracy. Because of the limited resource of sensor nodes, there is a limit in reducing σ_o^2 and σ_s^2. Once we have chosen the scheme to synchronize two neighboring nodes, σ_o^2 and σ_s^2 should be fixed. In order to achieve accurate network-wide time synchronization, we have to reduce the number of hops, or the distance, from sensor nodes to the clock reference nodes.

IV. Choose the Locations of Multiple Clock Reference Nodes

A. Problem Formulation

Because we aim at the network-wide synchronization, we have to take into consideration the synchronization inaccuracies of all sensor nodes. We set the objective of network-wide synchronization as minimizing the sum of the variances in measuring the skew (or offset) of all sensor nodes in the network. From (3) and (6) we see that minimizing the sum of the variances is equivalent to minimizing the sum of the number of hops from each node to its corresponding clock reference node. The clock reference nodes should be placed to proper locations to achieve accurate network-wide time synchronization, as the example in Fig. 1 shows. After the sensor network is deployed, the location of each sensor node is fixed and we do not consider the case where sensor nodes are mobile. In this scenario, we can define the problem of placing the multiple clock reference nodes as followings. Given a sensor network deployment (each sensor node has its fixed position in the network) and the number of clock reference nodes K, we need to choose K locations in the network to place those clock reference nodes, so the sum of the number of hops from each sensor node to the closest clock reference nodes is minimized.

The clock reference node placement problem can be formulated to the K-median problem. Similar to the K-median problem definition in [11], we formally define the clock reference node placement problem here. Given a sensor network deployment with a set of sensor nodes G. We need to find a set of K points, denoted by $G^{'}$, in the network to place K clock reference nodes. The number of hops from node $i \in G$ to a point $j \in G^{'}$ is $d(i,j)$. The objective is minimize the sum of the number of hops from each node to its nearest clock reference node, as shown in (7),

$$\text{minimize} \sum_{i \in G} \min_{j \in G'} d(i,j) \tag{7}$$

The K-median problem has been proven to be NP-Complete [11]. Since our clock reference node placement problem is not easier than the K-median problem, it is NP-Complete as well. We need to simplify this problem to make it tractable.

B. Problem Simplification

Some simplifications have to be introduced to make this problem tractable in practice. In Section V, we will show by simulations that the simplifications described here will not have much impact on the network-wide time synchronization.

1) Euclidean distance simplification: In the sensor networks where nodes are densely deployed, there are large number of nodes. Because of the dynamics of sensor networks, it is difficult, or even impossible, to know the exact nodes connectivity relationships and the length (number of hops) of the shortest path between two nodes in the network. We use the Euclidean distance to approximate the number of hops from one sensor node to the clock reference node. This simplification is reasonable since we assume the nodes are densely deployed and the communication range of each sensor node is usually fixed, so the physical distance is roughly proportional to the number of hops between two nodes.

2) Partition the network into grids: From the feasibility point of view, there is another issue we should consider in solving the clock reference node placement problem. For a large network with huge number of sensor nodes, the computation cost may be too high if we treat each node individually. To further simplify the clock reference node placement problem, we partition the network into grids and treat each grid as a weighted vertex in a graph. The weight is the node deployment density of the corresponding grid or the number of nodes within it. For instance, suppose the sensor network in concern is deployed in a square area. We evenly partition the square area into $N = n \times n$ equal size grids. The result is a grid network with $n \times n$ vertices and vertex i's weight is w_i. We assume the clock reference nodes can be placed on the vertices only, which is similar to what the work in [12] did.

C. Solve the Problem using Integer Programming

We solve the clock reference nodes placement problem on the partitioned grid-like network using integer programming. The grid-like network has N weighted nodes, assigned the ID from 1 to N and located at the vertices of a $n \times n$ grid-network. The possible locations of the clock reference nodes are limited to the vertices of the $n \times n$ grid-network. We need to deploy K clock reference nodes in the network at proper locations so the sum of the inaccuracies of all nodes is minimized. Let d_{ij} denote the distance from vertex i to vertex j and w_i is weight of the node at vertex i. We define two 0-1 variables x_{ij} and y_i as shown in (8) and (9), respectively. Variable x_{ij} is 1 means all sensor nodes in grid i will adjust their clocks according to a clock reference node located at position j. If the 0-1 variable y_i is 1, there is a clock reference node deployed at position i, otherwise there is no clock reference node at i.

$$x_{ij} = \begin{cases} 1, & \text{the node at } i \text{ uses ref at } j \\ 0, & \text{otherwise} \end{cases} \quad i, j \in [1, N] \tag{8}$$

$$y_i = \begin{cases} 1, & \text{there is a clock ref at } i \\ 0, & \text{otherwise} \end{cases} \quad i \in [1, N] \tag{9}$$

Then the integer programming formulation of the clock reference node placement problem is defined as followings:

$$\text{objective:} \quad \text{minimize} \sum_{i,j \in [1,N]} w_j d_{ij} x_{ij} \tag{10a}$$

$$\text{cons:} \quad \sum_{i \in [1,N]} y_i = K \tag{10b}$$

$$\sum_{i \in [1,N]} x_{ij} = 1, \ \ j \in [1, N] \tag{10c}$$

$$x_{ij} \leq y_i, \ \ i, j \in [1, N] \tag{10d}$$

$$x_{ij} \leq 1, \ \ i, j \in [1, N] \tag{10e}$$

$$y_i \leq 1, \ \ i \in [1, N] \tag{10f}$$

Constraint (10b) restricts the number of clock reference nodes to be K. Constraint (10c) means that each sensor node attaches to only one clock reference node. Constraint (10d) means if node at i synchronizes with a clock reference node at j, there must be a clock reference node at location j. Constraints (10e) and (10f) restrict the variables x_{ij} and y_j to be 0-1 variables. Because the objective is minimizing the sum of the distance from each node to its clock reference node, the optimal solution of the integer programming problem implicitly ensures that each node i attaches to the closest clock reference node. Otherwise, we can further optimize the objective function by attaching node i to a closer clock reference node.

We use a branch-and-bound approach to solve the integer programming problem. First we try to find a possible clock reference nodes placement by the guide of some heuristics, such as placing the K clock reference nodes to the K most weighted vertices. Then we have an upper bound of the objective function, denoted by Φ. During searching the solution tree, we use the linear programming relaxation to get the lower bound of one branch. If the linear programming relaxation lower bound is larger than the upper bound we have already gotten, that branch is pruned. If an all-integer solution is found and the result of the objective function in (10a) is ϕ, the upper bound Φ is updated to $\min(\Phi, \phi)$. After finishing the searching, we find the optimal solution for minimizing the objective function in (10a).

D. Operation of the Multiple Clock Reference Nodes Synchronization Scheme

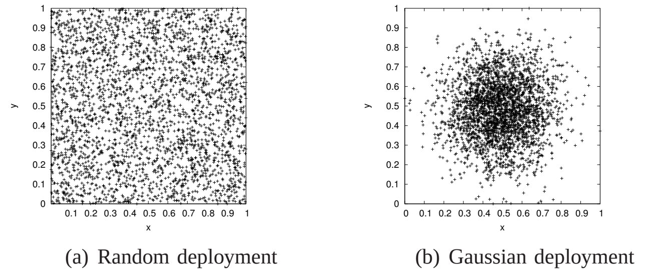
After we deploy the K clock reference nodes at proper locations, they start to synchronize sensor nodes in the network. The clock reference nodes periodically broadcast time synchronization messages to synchronize their neighboring sensor nodes. After the one-hop away sensor nodes are synchronized, those nodes broadcast synchronization messages to synchronize the nodes two-hop away from the clock reference nodes. By this means, eventually all nodes in the sensor network can measure their clock skew and offset according to those clock reference nodes.

When a synchronized sensor node broadcasts the synchronization messages, a *HopNum* field is included in those messages, which indicates how far that node is away from his closest clock reference node. If a sensor node receives multiple synchronization messages from different neighbors, the sensor node uses the message with the minimum *HopNum* value to do synchronization. By this means, we can assure that each sensor node synchronizes with his nearest clock reference node.

V. Evaluations

In this section, we will use simulation results to verify that the simplifications we have introduced do not have much impact on the network-wide time synchronization accuracy. We will also show that properly placing multiple clock reference nodes in the network can achieve good network-wide time synchronization accuracy. Even the locations of clock reference nodes cannot be precisely controlled, the network-wide time synchronization accuracy will not degrade too much and we can still have good synchronization accuracy by placing multiple clock reference nodes in the network.

A. The Network Model



(a) Random deployment (b) Gaussian deployment

Fig. 3. Random sensor nodes deployment with uniformly and Gaussian distributed nodes.

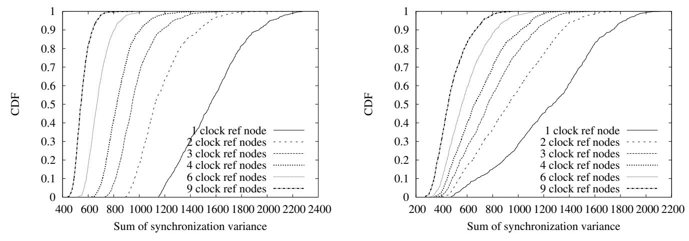
We use two kinds of network models in our experiments. An example of each kind of network is shown in in Fig.

3, both of which are within an 1×1 unit square area. Fig. 3(a) is the example of sensor network where the sensor nodes are uniformly distributed. In reality, this model represents the sensor networks in which sensor nodes are evenly deployed. For example, the sensor network used to monitor a large forest which is not accessible for human beings, so nodes are evenly scattered by airborne means. In Fig. 3(b), we show an example of sensor network where the nodes are distributed according to Gaussian distribution, i.e., the X and Y coordinates of each node are generated according to Gaussian distribution $N(0.5, 0.5^2)$ (we count only nodes whose coordinates are within the unit square). This network model represents that one place in the network need to be more carefully monitored so more sensor nodes are deployed in that "hot" point.

For the sake of simplicity, we assume the variances of measuring the skew and offset between two neighboring nodes are 1, i.e., $\sigma_o = 1$ and $\sigma_s = 1$. Since we have proven that the variance of measuring one sensor node's skew or the offset is directly proportional to the number of hops between that node and the clock reference node, the sum of all nodes' synchronization variances is equal to the sum of all nodes' distance to their closest clock reference node. During this section, when we say *"the sum of synchronization variance"*, we also mean *"the sum of the distance from node to its clock reference node"*, both of which represent the network-wide time synchronization inaccuracy. We also assume that the transmission range of sensor nodes is carefully tuned so that the connectivity of the network is guaranteed.

B. The Impact of the Number of Clock Reference Nodes

Intuitively, the more clock reference nodes are placed in the network, the smaller the sum of synchronization variances should be. We have a set of simulations to test the gain of placing multiple clock reference nodes. Because the variances of measuring the skew and offset are directly proportional to the distance from the sensor node to its clock reference node, we use the sum of the distance from each node to its corresponding clock reference node to represent the sum of each node's variances in measuring offset or skew.



(a) Uniform distribution nodes (b) Gaussian distribution nodes

Fig. 4. The CDF of the sum of synchronization variance of each sensor nodes when there are different number of clock reference nodes in the network. The reference clocks are randomly deployed.

We randomly generate a network deployment in which 3000 sensor nodes are located in an 1×1 unit square area network. The number of clock reference nodes is set to 1, 2, 3, 4, 6, and 9. For each case, we randomly choose where to place the clock reference nodes. Then the sum of each node's shortest distance to one of the clock reference node (the synchronization inaccuracy) is computed. This experiment is repeated 1000 times and we plot the CDF of the synchronization inaccuracy in Fig. 4. Fig. 4(a) is the experimental results when using the uniformly distributed sensor nodes network model. The experimental results in case of Gaussian distributed nodes is plotted in Fig. 4(b). We also calculate the average synchronization variances of the 1000 simulation instances for each network model and show the results in TABLE I.

From the simulation results in Fig. 4 and TABLE I, it can be concluded that using multiple reference clocks does have great gain in reducing the synchronization variances. When four clock reference nodes are used, the average synchronization variance can be reduced to half of the value when there is only one clock reference node.

TABLE I
AVERAGE SYNCHRONIZATION VARIANCES WHEN DIFFERENT NUMBER OF REFERENCE CLOCKS ARE USED.

# of ref-clocks	1	2	3	4	6	9
Uniform	1613	1166	952	839	683	553
Gaussian	1228	969	803	710	590	478

C. The Impact of Euclidean Distance Simplification

The Euclidean distance simplification is based on the intuition that in a large scale sensor network with densely deployed nodes, the Euclidean distance between two nodes should be proportional to the number of hops between them, if the communication distance of each node is the same. A set of simulation experiments have been conducted to evaluate the impact of using Euclidean distance to approximate the number of hops between two nodes.

We still randomly generate a network deployment where 3000 nodes are uniformly distributed within an 1×1 unit square network. The communication range of each node is set to 0.03 unit to ensure the network is connected ($\sqrt{\frac{1}{3000}} \approx 0.018 < 0.03$). All nodes within the communication range of one node are considered to be its neighbors. We randomly pick one of the 3000 nodes as the source node and use the Dijkastra shortest path algorithm to calculate the number of hops from the source node to all other nodes in the network. Therefore, we have two parameters for each node to represent his distance to the source node, the number of hops (X_h) gotten from the Dijkastra shortest path algorithm and the Euclidean distance (Y_e). For each node, we plot a point in Fig. 5 at (X_h, Y_e). The least-squares linear curve fit of those points is plotted as the dashed line in Fig. 5. From this figure we see that those points are well fitted by a linear curve, which means the number of hops between two nodes are roughly proportional to the Euclidean distance. So we can use the physical distance between two nodes to substitute the number of hops between them without making much difference for solving the clock reference node placement problem.

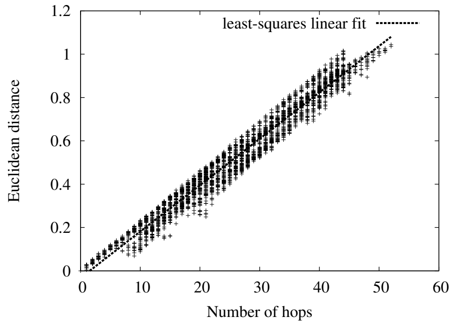


Fig. 5. The impact of Euclidean distance simplification. The X-axis is the number of hops and the Y-axis is the Euclidean distance.

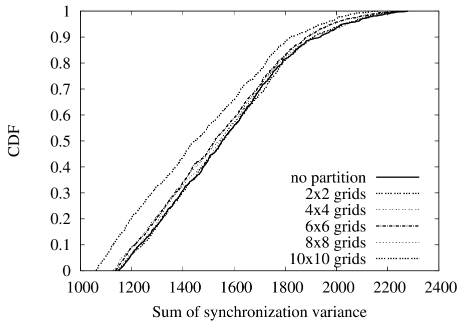


Fig. 6. The impact of grid simplification for uniformly distributed sensor nodes.

D. The Impact of Grid Simplification

We use the following experiments to evaluate the impact of the grid simplification. There are 3000 sensor nodes and 2 randomly deployed clock reference nodes in the network. We calculate the sum of synchronization variances (the network-wide synchronization inaccuracy) when the network is partitioned into 2×2 grids, 4×4 grids, 6×6 grids, 8×8 grids, 10×10 grids, and without partition. This experiment is repeated for 1000 times for both uniformly and Gaussian distributed sensor nodes deployments. The CDF of the network-wide synchronization inaccuracy in case of uniformly distributed sensor nodes is plotted in Fig. 6. The CDF for Gaussian distributed sensor nodes is plotted in Fig. 7.

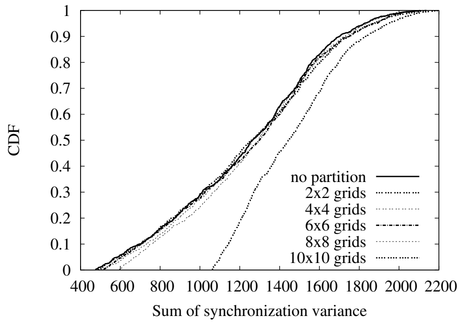


Fig. 7. The impact of grid simplification for Gaussian distributed sensor nodes.

From Fig. 6 and Fig. 7 we see that if sensor nodes are densely deployed and the network is partitioned into enough number of grid, the grid simplification does not make too much difference to the synchronization result. The CDF in case of 6×6 partition is almost the same as the CDF when the network is not partitioned.

We also have simulations to test the impact of the grid simplification when the sensor nodes are not densely deployed. We vary the number of sensor nodes in the network from 300 to 4000 and calculate the difference between the synchronization inaccuracies when there is no partition and when it is partitioned into a 6×6 grid-network. The percentage of the difference as compared with the synchronization inaccuracy in case of no partition is plotted in Fig. 8. From this plot, we see that even when there are only 300 nodes in the network, partitioning it into 6×6 grid leads to less than 3% difference from the synchronization inaccuracy without partition. The benefit of using such a partitioned grid-like network is that the computation overhead is greatly reduced. How many grids the network should be partitioned depends on the sensor density. What we should notice is that the nodes are usually densely deployed for typical sensor networks, so partitioning the network into a few grids should be enough.

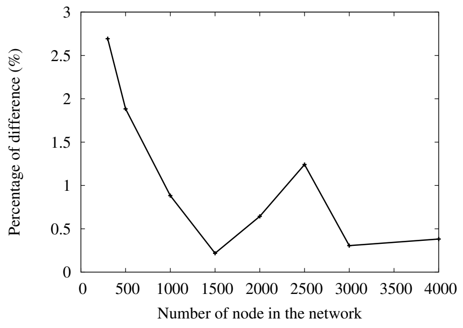


Fig. 8. Percentage of difference between the average synchronization variance when using the original network and the 6×6 grid-network.

E. The Comparison between Different Clock Reference Nodes Placement Strategies

We have compared the sum of all nodes' synchronization inaccuracies when three different strategies are used. The first strategy is random deployment, in which the clock reference nodes are randomly placed in the network. The second strategy is greedy deployment, which means the K clock reference nodes are placed in the centers of the top K most crowded grids, i.e., the K grids with more sensor nodes than other grids. The third strategy is the sub-optimal placement in which the locations of the clock reference nodes are gotten from solving the integer programming problem described in Section IV-C. The term "*sub-optimal placement* is used here because we use the simplified grid network to find the locations of the clock reference nodes. The solution may not be optimal for the original network.

We still randomly generate 1000 sensor network instances and plot the CDFs of the time synchronization inaccuracies when different strategies are used. Fig. 9 shows the experimental results when the sensor nodes are uniformly distributed in the network and 2 clock reference nodes are placed in

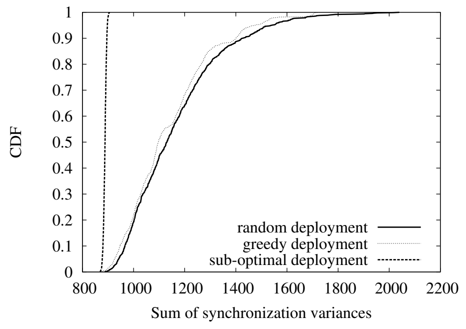


Fig. 9. The comparison between random deployment, greedy deployment, and sub-optimal deployment of two clock reference nodes. Sensor nodes are uniformly deployed in the network.

the network. Fig. 10 plots the simulation results when the sensor nodes deployment are generated according to Gaussian distribution. Both Fig. 9 and Fig. 10 tell us that choosing the right places to deploy the clock reference nodes has much better synchronization accuracy than just randomly picking up the deploying places. Besides, greedy deployment is a good heuristic to find the places to deploy the clock reference nodes in case of unevenly distributed sensor nodes. Fig. 10 shows that the CDF of greedy deployment is almost the same as the CDF of sub-optimal deployment.

We have other sets of experiments in which the number of clock reference nodes are set to 1, 3, and 4, respectively. For each number of clock reference nodes, we repeat the experiment for 1000 times and calculate the average synchronization variances. The results are shown in TABLE II and TABLE III, respectively.

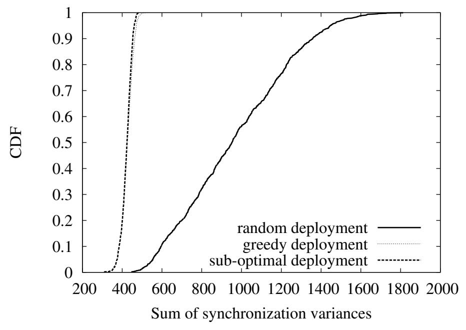


Fig. 10. The comparison between optimal deployment, random deployment and greedy deployment of two clock reference nodes. Sensor nodes are deployed according to Gaussian distribution.

From the simulation results in TABLE II and III we see that by carefully choosing the locations of the clock reference nodes, the overall synchronization inaccuracies can be greatly reduced, especially for networks where sensor nodes are not evenly distributed. For those sensor networks where nodes are uniformly distributed, the gain of carefully choosing the locations of the clock reference nodes is not so significant compared with networks having unevenly distributed nodes. That is because nodes are uniformly distributed so there is no specially crowded places in the network, the sum of synchronization variances in case of randomly deploying clock reference nodes may not be too much different from the greedy deploying and the sub-optimal deploying. On the contrary, when nodes are unevenly distributed, those places where nodes cluster together are the "special" points in the network. If we place the clock reference nodes far away from those clusters, the sum of the synchronization variances will be greatly impacted, since many nodes are far away from the clock reference nodes.

TABLE II
AVERAGE SYNCHRONIZATION VARIANCES IN CASE OF UNIFORMLY DISTRIBUTED SENSOR NODES

# of ref-clocks	1	2	3	4
random deploy	1613	1166	952	839
greedy deploy	1571	1140	920	772
sub-optimal deploy	1178	888	687	536

TABLE III
AVERAGE SYNCHRONIZATION VARIANCES IN CASE OF GAUSSIAN DISTRIBUTED SENSOR NODES

# of ref-clocks	1	2	3	4
random deploy	1228	969	803	710
greedy deploy	599	430	324	251
sub-optimal deploy	580	421	321	250

F. Tolerant to the Clock Reference Nodes Placing Deviation

In reality, we may not be able to place the clock reference nodes at the exact points where they are supposed to be. We use the following simulations to test how robust the time synchronization can be if the deployment points of the clock reference nodes have some deviations. Here we assume that the clock reference nodes cannot be precisely placed. Suppose the ideal place to deploy a clock reference node is at point p, the actual deploying point is randomly chosen from a circle which is centered at p and with radius r, as shown in Fig. 11.

There are still 3000 sensor nodes in an 1×1 unit square area and we range the number of clock reference nodes from 1 to 4. In each case, the clock reference nodes are placed with deviation described in the previous paragraph. After placing the clock reference nodes, we compute the sum of all nodes' synchronization inaccuracy. This experiment is repeated for 100 times and the average results are plotted in Fig. 12.

We can see from Fig. 12 that even the deployment of clock reference nodes has considerable deviations, randomly placing each clock reference node in a circle with radius 0.2 while the network is within an 1×1 square area, the network-wide time synchronization accuracy is still much better than randomly

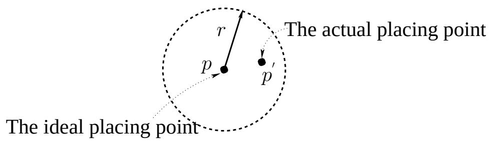
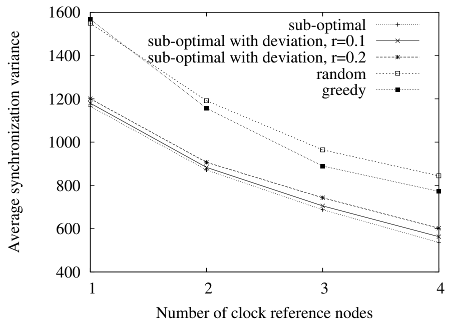
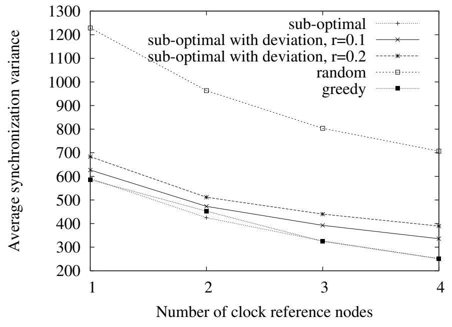


Fig. 11. The actual placing point of a clock reference node is a random point in a circle centered at the ideal placing point with radius r.



(a) Uniform nodes deployment



(b) Gaussian nodes deployment

Fig. 12. The average synchronization variances when the deployment of clock reference nodes has some deviation.

placing the clock reference nodes. If sensor nodes are evenly distributed in the network, the network-wide synchronization accuracy is not too much sensitive to the locations of the clock reference nodes. When sensor nodes are not evenly distributed, the clock reference nodes placement deviation can have more impact on the synchronization accuracy, but the synchronization accuracy with clock reference nodes deployment deviations is still better than randomly placing the clock reference nodes in the network.

VI. Conclusion

In this paper, we have studied placing multiple clock reference nodes in sensor networks to achieve accurate network-wide time synchronization. We have analyzed the source of time synchronization inaccuracies in sensor networks. Our analysis shows that the inaccuracies of measuring the skew and offset of a node multi-hop away from the clock reference node are directly proportional to the number of hops between them. We formulate the clock reference node placement problem to a K-median problem, which is NP-Complete. To make this problem tractable, some simplifications to the problem are introduced. We describe the clock reference node placement problem by 0-1 integer programming and use a branch-and-bound approach to solve it. Extensive numerical experiments have been conducted to evaluate the performance of placing multiple clock reference nodes for time synchronization. The experimental results show that the network-wide synchronization accuracy can be greatly improved by deploying multiple clock reference nodes and carefully choosing the locations of those nodes, especially for the networks where sensor nodes are unevenly distributed. Our simulations also demonstrate that even we cannot have good control on where the clock reference nodes are actually placed, deploying multiple clock reference nodes in the network can still achieve much more accurate network-wide time synchronization.

References

[1] R. Karp, J. Elson, D. Estrin, and S. Shenker, "Optimal and global time synchronization in sensornets," UCLA, Technical Report 0012, Apr. 2003.

[2] J. Hill and D. Culler, "A wireless embedded architecture for system level optimization," CS department, UC Berkeley," Technical Report, 2001.

[3] M. Sichitiu and C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks," in *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC 2003).*, vol. 2. IEEE, Mar. 2003, pp. 16–20.

[4] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2003, pp. 138–149.

[5] 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, 2002.

[6] J. van Greunen and J. Rabaey, "Lightweight time synchronization for sensor networks," in *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*. New York, NY, USA: ACM Press, 2003, pp. 11–19.

[7] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2004, pp. 39–49.

[8] D. L. Mills, "RFC 1305: Network time protocol (version 3) specification, implementation," Mar. 1992. [Online]. Available: http://www.ietf.org/rfc/rfc1305.txt?number=1305

[9] J. Elson and D. Estrin, "Time synchronization for wireless sensor networks," in *2001 International Parallel and Distributed Processing Symposium (IPDPS), Workshop on Parallel and Distributed Computing Issues in Wireless and Mobile Computing*, San Francisco, California, USA, Apr. 2001.

[10] K. Romer, "Time synchronization in ad hoc networks," in *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*. New York, NY, USA: ACM Press, 2001, pp. 173–182.

[11] P. Crescenzi, V. Kann, M. Karpinski, and G. Woeginger, "A compendium of NP optimization problems." [Online]. Available: http://www.nada.kth.se/~viggo/problemlist/compendium.html

[12] A. Bogdanov, E. Maneva, and S. Riesenfeld, "Power-aware base station positioning for sensor networks," in *Proceeding of Infocom'04*, 2004.