Secure and Resilient Clock Synchronization in Wireless Sensor Networks

Kun Sun, Peng Ning, *Member, IEEE*, and Cliff Wang, *Member, IEEE*

Invited Paper

***Abstract*—Wireless sensor networks have received a lot of attention recently due to its wide applications. An accurate and synchronized clock time is crucial in many sensor network applications. Several clock synchronization schemes have been proposed for wireless sensor networks recently to address the resource constraints in such networks. However, most of these techniques assume benign environments, but cannot survive malicious attacks in hostile environments, especially when there are compromised nodes. As an exception, a recent work attempts to detect malicious attacks against clock synchronization, and aborts when an attack is detected. Though this approach can prevent incorrect clock synchronization due to attacks, it will lead to denial of clock synchronization in such situations.**

This paper adopts a model where all the sensor nodes synchronize their clocks to a common source, which is assumed to be well synchronized to the external clock. This paper seeks techniques to provide redundant ways for each node to synchronize its clock with the common source, so that it can tolerate partially missing or false synchronization information provided by compromised nodes. Two types of techniques are developed using this general method: *level-based clock synchronization* and *diffusion-based clock synchronization*. Targeted at static sensor networks, the level-based clock synchronization constructs a level hierarchy initially, and uses (or reuses) this level hierarchy for multiple rounds of clock synchronization. The diffusion-based clock synchronization attempts to synchronize all the clocks without relying on any structure assumptions and, thus, can be used for dynamic sensor networks. This paper further investigates how to use multiple clock sources for both approaches to increase the resilience against compromise of source nodes. The analysis in this paper indicates that both level-based and diffusion-based approaches can tolerate up to s colluding malicious source nodes and t colluding malicious nodes among the neighbors of each normal node, where s and t are two system parameters. This paper also presents the results of simulation studies performed to evaluate the proposed techniques. These results demonstrate that the level-based approach has less overhead and higher precision, but less coverage, than the diffusion-based approach.

Manuscript received September 1, 2005; revised October 1, 2005. The work of K. Sun was supported in part by the Army Research Office (ARO) under Grant W911NF-04-D-0003-0001. The work of P. Ning was supported in part by the National Science Foundation (NSF) under Grant CAREER-0447761, in part by the Army Research Office (ARO), in part by Advanced Research and Development Activity (ARDA), and in part by NCSU/Duke Center for Advanced Computing and Communication (CACC).

K. Sun and P. Ning are with the Department of Computer Science, North Carolina State University, Raleigh, NC 27695 USA (e-mail: ksun3@ncsu.edu; pning@ncsu.edu).

C. Wang is with the U.S. Army Research Office, Research Triangle Park, NC 27709-2211 (e-mail: cliffwang@ncsu.edu)

Digital Object Identifier 10.1109/JSAC.2005.861396

***Index Terms*—Computer network security, fault tolerance, synchronization, wireless sensor networks.**

I. Introduction

WIRELESS sensor networks have received a lot of attention recently due to its wide applications, such as target tracking, monitoring of critical infrastructures, and scientific exploration in dangerous environments. Sensor nodes are typically resource constrained, and usually communicate with each other through short range wireless links.

An accurate and synchronized clock time is crucial in many sensor network applications, particularly due to the collaborative nature of sensor networks. For example, in target tracking applications, sensor nodes need both the location and the time when the target is sensed to correctly determine the target moving direction and speed (e.g., [1] and [2]). However, due to the resource constraints on sensor nodes, traditional clock synchronization protocols (e.g., network time protocol (NTP) [3]) cannot be directly applied in sensor networks.

Several clock synchronization protocols (e.g., [4]–[11]) have been proposed for sensor networks to achieve *pairwise* and/or *global* clock synchronization. Pairwise clock synchronization aims to obtain a high-precision clock synchronization between pairs of sensor nodes, while global clock synchronization aims to provide network-wide clock synchronization in a sensor network. Existing pairwise or global clock synchronization techniques are all based on *single-hop* pairwise clock synchronization, which discovers the clock difference between two neighbor nodes that can communicate with each other directly. Two approaches have been proposed for single-hop pairwise clock synchronization: *receiver–receiver synchronization* (e.g., reference broadcast synchronization (RBS) [4]), in which a reference node broadcasts a reference packet to help pairs of receivers to identify the clock differences, or *sender-receiver synchronization* (e.g., timing-sync protocol for sensor networks (TPSNs) [5]), where a sender communicates with a receiver to estimate the clock difference. Multihop pairwise clock synchronization protocols and most of the global clock synchronization protocols (e.g., [4], [5], and [9]) establish multihop paths in a sensor network, so that all the nodes in the network can synchronize their clocks to the source based on these paths and the single-hop pairwise clock differences between adjacent nodes in these paths.

Alternatively, diffusion-based global synchronization protocols [7] achieve global synchronization by spreading local synchronization information to the entire network.

Most of these techniques assume benign environments; however, malicious intruders may certainly attack the clock synchronization protocols due to the importance of synchronized clock time. Though it is possible to use authentication to defend against external attacks, an attacker may still attack clock synchronization through compromised nodes. A compromised node has limited impact on single-hop clock synchronization between neighbor nodes, since with sender–receiver protocols such as TPSN [5], the compromised node can only affect the clock difference between itself and a normal node (rather than between normal nodes). However, when a pair of nodes are synchronized through a multihop path (e.g., [4], [5], and [9]), a compromised node in the path can introduce arbitrary errors. This implies multihop pairwise and global clock synchronization using multihop paths are vulnerable to compromised nodes. Even when the diffusion-based global clock synchronization techniques [7] are used, compromised nodes may fluctuate their clock information periodically to prevent the convergence of the clocks.

It is natural to consider fault-tolerant clock synchronization techniques, which have been studied extensively in the context of distributed systems (e.g., [12]–[16]). However, these techniques require either digital signatures (e.g., HSSD [16]and CSM [15]), exponential copies of messages (e.g., COM [15]), or a completely connected network (e.g., CNV [15]) to prevent malicious nodes from modifying or destroying clock information sent by normal nodes. Thus, they are not practical in wireless sensor networks.

A recent work [17] attempts to detect malicious attacks against clock synchronization, and aborts clock synchronization when such an attack is detected. Though this approach can prevent incorrect clock synchronization due to malicious attacks, it will also lead to denial of clock synchronization in such situations. Thus, it is necessary to seek additional techniques to protect clock synchronization in sensor networks.

In this paper, we develop secure and resilient clock synchronization techniques for wireless sensor networks. We adopt a model where all the sensor nodes synchronize their clocks to a common source, which is assumed to be well synchronized to an external clock. Our basic idea is to provide redundant ways for each node to synchronize its clock with the common source, so that it can tolerate partially missing or false synchronization information provided by the malicious nodes. Using this general method, we develop two types of clock synchronization techniques: *level-based clock synchronization* and *diffusion-based clock synchronization*. The level-based scheme builds a level hierarchy in the sensor network, and then synchronizes the nodes in the network level by level. The diffusion-based scheme allows each node to diffuse its clock to its neighbor nodes after it has synchronized to the source node. Our approaches guarantee that normal nodes can synchronize their clocks to the common source node even if each normal node has up to t colluding malicious nodes among its neighbor nodes. Our analysis and simulation results indicate that these two approaches are complementary. The level-based approach is suitable for static sensor networks, while the diffusion-based approach is suitable for dynamic sensor networks. The level-based approach has less overhead and higher precision, but less coverage than the diffusion-based approach.

To improve the synchronization precision and reduce the communication overhead in large sensor networks, we propose to deploy multiple source nodes in the network, so that the sensor nodes can synchronize to the nearest source node. Moreover, we extend this approach to increase the resilience of such clock synchronization. As a result, a sensor node can obtain the correct clock time even if up to the half of the source nodes to which it can synchronize are compromised. This approach can tolerate up to s colluding malicious source nodes in addition to t colluding malicious nodes among the neighbors of each normal node, where s and t are two system parameters.

In summary, this paper makes the following contributions.

- We develop a model for resilient clock synchronization in sensor networks by adapting traditional fault tolerant techniques.
- We develop two complementary approaches for secure and resilient clock distribution (from a single source) in sensor networks. These approaches consider the practical constraints (e.g., low-power, low bandwidth communication, and the lack of general broadcast authentication techniques) in the current generation of sensor networks, and have proved security properties.
- We develop multisource-based clock synchronization techniques for sensor networks based on the above clock distribution approaches. These techniques further consider practical issues such as medium access control (MAC) layer message collision and potentially compromised source nodes.
- We perform substantial experiments to evaluate various aspects of the proposed techniques, including synchronization precision, synchronization rate (i.e., percentage of synchronized nodes), synchronization time (i.e., time required by synchronization), and communication overhead. The results indicate that these approaches are practical for the current generation of sensor networks.

The rest of this paper is organized as follows. Section II motivates our approaches with an example, and describes our global clock synchronization model. Section III presents two secure and resilient approaches for level-based clock synchronization and diffusion-based clock synchronization. Section IV presents the clock synchronization with multiple source nodes in sensor networks. Section V presents the simulation experiments used to evaluate the proposed techniques. Section VI discusses related work, and Section VII concludes the paper and points out some future research directions.

II. Model for Clock Synchronization

A. Motivating Example

Consider Fig. 1, in which there are multiple, interleaved paths between node S and node D. Assume node D needs to estimate the clock difference between itself and node S. Suppose that each pair of nodes connected by an edge in the network are neighbors, and have synchronized with each other using a

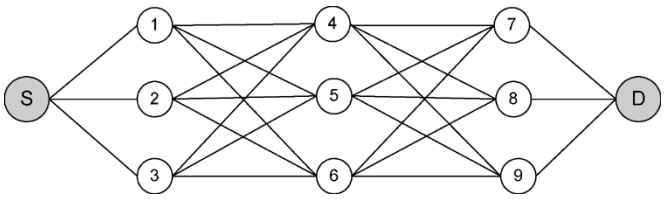


Fig. 1. Mesh network between nodes S and D.

single-hop pairwise clock synchronization scheme (e.g., RBS [4] and TPSN [5]). For convenience, we denote the pairwise clock difference between any two nodes i and j as $\delta_{i,j}$. Specifically, $\delta_{i,j} = C_j - C_i$, where C_i and C_j are the local clock of node i and node j, respectively. We assume some nodes may have been compromised and, thus, may lie about any information needed by other nodes.

We first estimate the clock differences between S and the nodes close to S (in a fault-tolerant way), then gradually use these clock differences to estimate those between S and the nodes farther away from S, and eventually derive the clock difference between S and D. According to the assumption, nodes 1, 2, and 3 have obtained $\delta_{1,S}$, $\delta_{2,S}$, and $\delta_{3,S}$, respectively. Now, consider node 4. Node 4 may estimate $\delta_{4,S}$ through 1, 2, or 3. To deal with potentially malicious nodes, node 4 can estimate $\delta_{4,S}$ through all three nodes. When node 1 is chosen, node 4 can easily compute $\delta_{4,S}^{(1)} = \delta_{4,1} + \delta_{1,S}$. Similarly, node 4 can compute $\delta_{4,S}^{(2)}$ and $\delta_{4,S}^{(3)}$ through nodes 2 and 3, respectively. Then, node 4 chooses the median of the three values as $\delta_{4,S}$. As a result, if only one of nodes 1, 2, and 3 is malicious and attempts to attack clock synchronization, its effect will be removed.

This process may continue for nodes 7, 8, and 9, assuming 4, 5, and 6 have obtained $\delta_{4,S}$, $\delta_{5,S}$, and $\delta_{6,S}$, respectively. Eventually, node D can obtain the correct clock difference $\delta_{D,S}$ if there is at most one malicious node in each level in the mesh network between S and D. In general, if there are $2t+1$ nodes in each level of the mesh network between nodes S and D and all the neighboring nodes can communicate with each other, this approach can tolerate up to t colluding malicious nodes in each level.

B. Our Model

We develop our secure clock synchronization techniques by generalizing the above motivating example. We assume there is a *source node* S that is well synchronized to the external clock, for example, through a global positioning system (GPS) receiver. We would like to synchronize the clocks of all the sensor nodes in the network to that of the source node. We assume the source node is trusted, and all the other nodes know the identity of the source node.

We adopt the following model for secure and resilient clock synchronization.

1) Each node i maintains a *local clock* C_i. The local clock of the source node (i.e., C_S) is the desired global clock.
2) For each neighbor node j, each node i maintains a *single-hop pairwise clock difference* $\delta_{i,j} = C_j - C_i$ with a method that is secure for single-hop pairwise clock synchronization (e.g., RBS [4] and TPSN [5]).
3) Each node i also maintains a *source clock difference* $\delta_{i,S}$ between its local clock and the clock of the source node S. Node i can directly obtain it if it is a neighbor node of S. Otherwise, node i needs to estimate $\delta_{i,S}$.
4) To tolerate up to t malicious neighbor nodes, each node i needs to compute at least $2t+1$ *candidate* source clock differences through different neighbor nodes. Specifically, the candidate source clock difference obtained through neighbor node j is $\delta_{i,S}^{(j)} = \delta_{i,j} + \delta_{j,S}$. Node i then chooses the median of the candidate source clock differences as $\delta_{i,S}$. We assume the sensor network of concern is dense so that each node has enough number of neighbor nodes to obtain $2t+1$ candidate source clock differences.
5) Each node i can estimate the *global clock* C_S by using its local clock and its source clock difference (i.e., $C_S = C_i + \delta_{i,S}$).

We assume there are malicious nodes (e.g., compromised nodes that possess valid cryptographic keys) in the network, which may collude together to disrupt clock synchronization. A malicious node i may affect a normal node j by affecting node j's measurement of $\delta_{i,j}$ and/or lying about $\delta_{i,S}$. An attacker may also affect the single-hop pairwise clock difference between two normal nodes by launching, for example, pulse-delay attacks [17] or wormhole attacks [18]. Such attacks, however, can be subsumed by the scenarios where one of the two nodes is compromised. For brevity, when the single-hop pairwise clock difference between two normal nodes is impaired by the attacker (via, e.g., wormhole attack), we consider the node closer (in terms of the number of hops) to the source node as compromised. When one of the nodes is the source, we consider the other as compromised.

Our goal is to provide secure clock synchronization so that even if a certain number of malicious nodes collude together to disrupt clock synchronization, each normal node i can still synchronize its local clock to the source node.

It is natural for sensor nodes to communicate through broadcast. In hostile environments, this requires broadcast authentication to ensure the authenticity and integrity of the broadcast messages. There are two ways to provide broadcast authentication: digital signatures and TESLA-based approaches [19]–[21]. However, both types of approaches are difficult to use for clock synchronization in wireless sensor networks. Though it is shown recently that it is feasible to perform public key cryptography (including digital signatures such as ECDSA) on low-end sensor nodes [22], such operations still cost substantial computational and power resources, and are subject to DoS attacks. The TESLA-based approaches can provide broadcast authentication by using efficient symmetric cryptography. However, the TESLA-based approaches require at least loose synchronization among all the nodes and, thus, cannot be used to for global clock synchronization directly.

Due to the above reasons, we assume each pair of nodes communicate through unicast, and any two nodes that need to communicate with each other share a unique pairwise key, so that the messages between them are authenticated. One node can also identify the other node based on the unique pairwise

key. Such pairwise keys can be provided by several key predistribution schemes proposed for sensor networks recently (e.g., [23]–[25]). For brevity, we assume all pairwise clock difference $\delta_{i,j}$ between two neighbor nodes i and j is obtained with a single-hop pairwise clock synchronization technique (e.g., RBS [4] and TPSN [5]). These single-hop pairwise clock differences may be incorrect due to compromised nodes [18].

We give the following recursive definition to further clarify the correctness of secure and resilient clock synchronization.

Definition 1: With a unique source node S, a source clock difference $\delta_{i,S}$ obtained by node i is *correct* if:

- node i is a neighbor node of node S;
- $\delta_{i,S}$ is computed as $\delta_{i,S} = \delta_{i,j} + \delta_{j,S}$, where node j is a neighbor of node i, and either: 1) node j is a normal node and $\delta_{j,S}$ is correct or (2) node i has two other normal neighbor nodes m and n such that $\delta_{m,S}$ and $\delta_{n,S}$ are correct and $\delta_{i,m} + \delta_{m,S} \leq \delta_{i,S} \leq \delta_{i,n} + \delta_{n,S}$.

Intuitively, Definition 1 says that a normal node can obtain a correct source clock difference in two cases: 1) either it computes the source clock difference through a normal node with a correct source clock difference and 2) or it computes this value through a compromised node, but this value happens to be between two source clock differences that this node could have computed through two normal nodes with correct source clock differences.

It is easy to see that if node i has a correct source clock difference, it can estimate the global clock C_S "correctly."

III. Secure and Resilient Clock Synchronization

In this paper, we develop two secure and resilient clock synchronization schemes for wireless sensor networks based on the general method discussed in Section II: level-based clock synchronization and diffusion-based clock synchronization. In the level-based scheme, a level hierarchy is established initially in the sensor network, and each node obtains the clock differences from its parent nodes in the level hierarchy. In the diffusion-based scheme, a node can obtain the clock differences from any neighbor nodes. The level-based scheme is suitable for static sensor networks, where sensor nodes stay in the same places after deployment. In contrast, the diffusion-based scheme is more suitable for dynamic sensor networks, where sensor nodes may move frequently.

A. Level-Based Clock Synchronization

Level-based clock synchronization aims at static sensor networks, where the network topology does not change frequently. Level-based clock synchronization consists of two phases: *level discovery phase* and *synchronization phase*. The level discovery phase is to organize sensor nodes into a hierarchy rooted at the source node S so that two nodes connected in the hierarchy are neighbors. Each node except for the root has a set of parent nodes in the hierarchy, and each nonleaf node has a set of children nodes. Each node is also associated with a *level*, which is the number of hops in the longest path from the root to itself. We refer to this hierarchy as the level hierarchy. In the synchronization phase, all the sensor nodes obtain the source clock differences through their parent nodes, estimate their own source clock differences, and then help their children nodes to synchronize their clocks.

1) Level Discovery Phase: To establish the level hierarchy, each node maintains three variables: `level`, `parents`, and `children`. The variable `level` records the level of the node. `Parents` and `children` record the parents and the children of the node in the level hierarchy, respectively. After the level hierarchy is established, a node i can obtain the candidate source clock differences from the nodes in its parent set, and may help the nodes recorded in its children set to obtain their source clock differences.

We assume all the sensor nodes have discovered their neighbors before the level discovery phase. Consider the source node S. Initially, $S.\texttt{level} = 0$, $S.\texttt{parents} = \emptyset$, and $S.\texttt{children} = \{x | x \text{ is a neighbor of } S\}$. The variables of all the other nodes are unknown. The source node S initiates the level discovery phase by unicasting a *level discovery message* to each of its neighbor nodes. A level discovery message contains the sender's identity and its level number, authenticated (and optionally encrypted) with the pairwise key shared between the sender and the receiver. After receiving an authenticated level discovery message from S, each neighbor i of S sets $i.\texttt{level}$ as 1, and $i.\texttt{parents}$ as $\{S\}$. It then unicasts a level discovery message to each of its neighbor nodes except for S.

The nodes that are more than one hop away from the source node may receive more than one level discovery messages from their neighbors. To tolerate up to t malicious parent nodes in the synchronization phase, a node needs to record $3t+1$ parent nodes. When a normal node has $3t+1$ parent nodes in the level hierarchy, even if up to t malicious parent nodes keep silent during the synchronization phase, the node still can receive $2t+1$ candidate source clock differences and synchronize its clock.

We have two options for a sensor node to obtain its level and parent set. In the first option, after receiving authenticated level discovery messages from the first $3t+1$ different neighbor nodes, node i chooses these nodes as its parent nodes. In the second option, node i may wait for a period of τ time units after getting the first $3t+1$ candidate parent nodes, and then choose the $3t+1$ nodes with the least levels as the parent nodes. When using the second option, the convergence time of the level discovery phase is longer than that for the first option, but the average level of the sensor nodes is smaller. Because the source node runs level discovery process infrequently, we adopt the second option in our level-based scheme. Assuming the maximum level of the parent nodes is l, node i then sets $i.\texttt{level}$ as $l+1$.

After determining its level, a node i unicasts level discovery messages to its neighbors from which it has not received any authenticated level discovery message. Node i also unicasts messages to its parent nodes to add itself as a children node. Node i will drop subsequent level discovery messages.

The level hierarchy needs to be maintained when there are slight changes in the network (e.g., node joins and failures). The maintenance may be performed locally without re-executing the level discovery phase. When a new node joins the network, it needs to determine its level and find its parent nodes in the level hierarchy. To do it, it unicasts *level query messages* to all its neighbor nodes. A neighbor node will send back a *level reply*

message, containing its identity and its level. All the messages are authenticated by the shared pairwise key. The new node can determine its level and parent nodes. In the synchronization phase, when a node fails to receive from at least $2t+1$ parent nodes in several rounds of synchronization, it will send level query messages to its neighbor nodes that are not its parent or children nodes, and recruits new parent nodes according to the level reply messages.

2) Synchronization Phase: Due to the clock drift of sensor nodes, the source node S periodically initiates the synchronization phase by unicasting *synchronization messages* to its neighbor nodes. A synchronization message contains the sender's identity, a sequence number, and the sender's source clock difference. Each node maintains a sequence number, and increases it in each round of synchronization. These nodes then further send synchronization messages to their children nodes. All the relevant messages are authenticated with a key shared between the communicating nodes.

After receiving a synchronization message from node S, level one nodes start the single-hop pairwise clock synchronization with the source node. Then, they unicast synchronization messages to their children nodes. Consider a node i at a level greater than 1. When it receives a synchronization message from a parent node j, after obtaining the single-hop pairwise clock difference from node j, node i calculates a candidate source clock difference by $\delta_{i,S}^{(j)} = \delta_{i,j} + \delta_{j,S}$. To tolerate up to t malicious nodes in its parent nodes, it has to collect at least $2t+1$ candidate source clock differences through its parent nodes. Node i sets the source clock difference $\delta_{i,S}$ as the median of the $2t+1$ candidate source clock differences. Then, node i unicasts its source clock difference to its children nodes.

3) Effectiveness: We first introduce Lemma 1 to facilitate the analysis.

Lemma 1: Assume a normal node i has at least $2t+1$ neighbor nodes, among which there are at most t colluding malicious nodes. Node i can obtain a correct source clock difference if it receives from each neighbor node the source clock difference and all the normal neighbor nodes provide their correct source clock differences.

Proof: According to our model, node i computes a candidate source clock difference with the source clock difference provided by each neighbor node, and then chooses the median as its source clock difference $\delta_{i,S}$. Suppose the source clock difference is obtained through node j, that is, $\delta_{i,S} = \delta_{i,j} + \delta_{j,S}$. There are two cases.

Case 1) If node j is a normal node, both $\delta_{j,S}$ and $\delta_{i,j}$ must be correct according to the assumption, and $\delta_{i,S} = \delta_{i,j} + \delta_{j,S}$ is correct according to Definition 1.

Case 2) Suppose node j is malicious. Because there are at most t malicious nodes, $\delta_{i,S}$, which is the median of the $2t+1$ candidate source clock differences, must be between two candidate source clock differences obtained through two normal nodes. Thus, the source clock difference $\delta_{i,S}$ is still correct, according to Definition 1. ■

Based on Lemma 1, we have the following results on the effectiveness of level-based clock synchronization.

Lemma 2: The level-based clock synchronization can synchronize all the normal nodes correctly, if each normal node at level l ($l > 1$) receives at least $2t+1$ source clock differences from distinct parent nodes and at most t out of these parent nodes are colluding malicious nodes.

Proof: This is equivalent to proving that each normal node i can obtain the correct source clock difference $\delta_{i,S}$ if the given conditions are satisfied. We prove it by induction.

Each level one node i can obtain the correct source clock difference $\delta_{i,S}$, which is the single-hop pairwise clock difference. (Note that a level one node that cannot obtain the correct single-hop pairwise clock difference with the source is considered compromised.) Suppose each normal node at a level less than or equal to level k ($k \geq 1$) has obtained the correct source clock difference. Consider a normal node j at level $k+1$. All parents of node j have levels less than or equal to k. If node j receives source clock differences from at least $2t+1$ distinct parent nodes and at most t out of them are colluding malicious nodes, then by Lemma 1, node j can obtain its correct source clock difference $\delta_{j,S}$. ■

B. Diffusion-Based Clock Synchronization

With level-based clock synchronization, all the sensor nodes synchronize to the source node by using the level hierarchy. The following diffusion-based clock synchronization scheme allows sensor nodes to obtain source clock differences through any neighbor nodes without requiring any level hierarchy.

In the diffusion-based scheme, the source node S initiates the synchronization process periodically by unicasting synchronization messages to its neighbor nodes. After obtaining a source clock difference from the source node, the neighbor nodes of S update their source clock differences, and then unicast synchronization messages to their neighbors except for S. To tolerate up to t colluding malicious nodes among its neighbor node, a node more than one hop away from the source node needs to receive at least $2t+1$ candidate source clock differences through different neighbor nodes, and updates its source clock difference as the median of the $2t+1$ source clock differences. The node then sends synchronization messages to its neighbors from which it has not received synchronization messages.

We have the following results on the effectiveness of diffusion-based clock synchronization.

Lemma 3: The diffusion-based clock synchronization scheme can synchronize all the normal nodes correctly, if each normal node that is more than one hop away from the source node receives the source clock differences (of the neighbor nodes) from at least $2t+1$ distinct neighbor nodes among which at most t nodes are colluding malicious nodes.

Proof: This is equivalent to proving that each node i can obtain the correct source clock difference $\delta_{i,S}$ if the given conditions are satisfied. We prove it by induction.

Each normal neighbor node i of the source node can obtain the correct source clock difference $\delta_{i,S}$, which is the single-hop pairwise clock difference. Assume at a certain time, all the normal nodes that have been synchronized have correct source clock differences. Consider a normal node j that is more than one hop away from the source node. From the assumption, if it can receive the source clock differences (of the neighbor nodes)

from at least $2t + 1$ distinct neighbor nodes, among which at most t nodes are colluding malicious nodes, then by Lemma 1, node j can obtain its own correct source clock difference. ■

The benefit of the diffusion-based scheme is that all communication is localized without depending on a distributed level hierarchy. However, a node has to send synchronization messages to all its neighbor nodes from which it has not received synchronization messages. The diffusion-based scheme potentially has higher communication overhead than the level-based one, but is more suitable for dynamic sensor networks, where the network topology changes frequently.

C. Security Analysis

In both lemmas 2 and 3, a normal node can correctly synchronize its clock to the source nodes when the following two conditions are satisfied.

- Condition 1: Each normal node can receive $2t + 1$ source clock differences.
- Condition 2: Among the $2t + 1$ source clock differences, there exist at most t malicious source clock differences.

Our schemes require these two conditions to provide correct global clock synchronization. Now, consider the first condition. Our schemes are suitable for dense sensor networks in which a normal node can receive at least $2t + 1$ source clock differences. In one round of clock synchronization, a malicious node may refuse to provide its source clock difference to its neighbor nodes. In the level-based scheme, a normal node can tolerate such attacks by recording $3t + 1$ parent nodes in its parent set, so that even if up to t malicious nodes keep silent, the normal node can still receive $2t + 1$ source clock differences. This attack has little effect on the diffusion-based scheme when a normal node can obtain source clock differences from any $2t+1$ neighbor nodes, though the malicious nodes keep silent. Our schemes fail when an attacker can launch signal jamming attacks, since normal nodes cannot receive any synchronization messages. Nevertheless, no scheme that requires internode communication can survive such attacks.

Let us consider the second condition. Our schemes guarantee correct clock synchronization as long as the normal node accepts at most t malicious source clock differences from its malicious neighbor nodes. An attacker may attack the level discovery phase of the level-based clock synchronization, aiming at increasing the impact of malicious nodes and corrupting the level hierarchy, or directly attack the synchronization phase in the diffusion-based clock synchronization. Assume a given normal node has at most t malicious nodes that appear to be its neighbors. These malicious nodes may be nodes physically located near the normal node, remote malicious nodes that pretend to be in this local area through wormholes [18], or normal nodes whose single-hop pairwise clock differences (with the given node) are distorted by, for example, wormholes. Though we cannot immediately identify malicious nodes physically located near the normal node, remote malicious nodes and normal nodes tunneled through wormholes can be detected with their locations and/or the message transmission delays, as indicated in [18] and [26]. Further considering the difficulty of physically deploying malicious nodes, it is in general difficult for an attacker to have many malicious neighbor nodes interfere with the clock synchronization of normal nodes.

A malicious node may certainly attempt to forge multiple identities by launching Sybil attacks [27]. By using unique pairwise keys to authenticate messages, our schemes can prevent malicious nodes from impersonating uncompromised normal nodes. If colluding malicious nodes can exchange their keying materials, one malicious node may impersonate other remote malicious nodes in its local network. Such colluding malicious nodes may be detected and removed by using the techniques proposed in [28].

A malicious node may launch replay attacks during the synchronization process. Specifically, a malicious node may record a synchronization message in one round of clock synchronization, and replay it to normal nodes in later rounds. As a result, the normal nodes may accept the replayed message, and derive a false source clock difference.

This attack can be prevented by including a per-node sequence number in the synchronization messages. Specifically, each node maintains a sequence number for itself, and keeps a copy of the most recent sequence number received from *each* of its parent nodes (or neighbor nodes in case of diffusion-based scheme). In a new round of clock synchronization, each node increments its sequence number and includes it in all the messages sent to its neighbor nodes. Accordingly, a node only accepts a message from a neighbor node (and update the corresponding sequence number) if the sequence number in the message is greater than the recorded one. Note that we cannot use a global sequence number to prevent replay attacks. Otherwise, a malicious node that has the right keying materials may launch denial-of-service (DoS) attacks.

Besides the efforts to violate the above necessary conditions on clock synchronization, attackers may attempt to launch resource consumption attacks to deplete the limited battery power of sensor nodes in a period of short time. In level discovery phase, a malicious node may make itself the children node of all its neighbors. In the synchronization phase, all its neighbor nodes will have to unicast synchronization messages to this malicious node, which is a waste of their battery power. However, such a malicious node can only force each of its neighbor nodes to transmit a few messages in each synchronization round and, thus, has limited impact.

There is a potentially more serious resource consumption attack. In the synchronization phase, a malicious node may unicast synchronization messages to its neighbor nodes at any time, without receiving any synchronization message. In other words, the malicious nodes may attempt to start a synchronization round without being triggered by the source node. Fortunately, a normal node sends synchronization messages only after receiving at least $2t + 1$ synchronization messages from distinct neighbors. As a result, the malicious nodes may convince its normal children nodes to request synchronization messages from other parent nodes, but will not convince them to further send synchronization messages, as long as the victim normal node has less than $2t + 1$ malicious neighbor nodes.

D. Performance Analysis

We discuss the performance of the proposed schemes using communication overhead, synchronization precision, and memory requirement.

Communication Overhead: To facilitate the analysis of communication overhead, we consider a sensor network as a graph $G = \{V, E\}$, in which each vertex in V stands for a node in the network, and each edge in E represents that the two vertices of the edge are neighbor nodes.

In the level discovery phase of level-based approach, after a node determines its level, it unicasts level discovery messages to the neighbors that have not sent level discovery messages to it. Assuming there is no communication failure and all the nodes are included in the level hierarchy, all the edges in the graph will be covered exactly once by one level discovery message in both approaches. Thus, the overhead is $O(|E|)$. In the synchronization phase, we assume that there is no communication failure and all the nodes in the network can synchronize their clocks. Suppose there are n_1 nodes in level one. Since the nodes at levels more than 1 will receive $3t + 1$ synchronization messages, the total number of messages transmitted in one round of clock synchronization can be estimated as

$$n_1 + (|V| - n_1 - 1)(3t + 1). \tag{1}$$

In the diffusion-based scheme, the number of messages transmitted in one round of clock synchronization is the same as that in the level discovery phase of the level-based schemes, that is $O(|E|)$. Suppose each node has k neighbor nodes in average in a large dense sensor network. We have $|E| = |V| \cdot k/2$. Compared with the level-based schemes, the diffusion-based scheme has a higher communication overhead when $k \geq 2(3t + 1)$.

Note that in real sensor networks, the communication overhead in both schemes will be higher than what we estimated earlier due to message collisions.

Synchronization Precision: The synchronization precision at a node i can be measured by the clock error between node i's estimated global clock and the actual global clock (i.e., the clock of the source node) when node i adjusts its local clock. Specifically, $\text{Error}_i = |C_i + \delta_{i,S} - C_S|$, where C_i and C_S are the local clock values of node i and the source node S, respectively, and $\delta_{i,S}$ is the estimated source clock difference.

A high precision pairwise clock synchronization scheme is critical for our schemes, since the synchronization error may accumulate for nodes that are multiple hops away from the source node. It is suggested in TPSN [5] that we can use MAC layer timestamp to minimize the clock error. In our scheme, the major clock error is mostly caused by the clock drift between the time when the source node starts one round of clock synchronization and the time when a node obtains its source clock difference. Suppose the source node S initiates one round of synchronization at time t_s and node i adjusts its clock at time t_i, where $t_i > t_S$. We denote the maximum time duration $t_i - t_S$ of all the nodes as the *synchronization time*. By [16], when the maximum clock drift of all the clocks is ρ, the maximum clock drift during $t_i - t_s$ between node S and node i is up to $2\rho(t_i - t_s)$. It seems that a sensor node may receive $2t + 1$ messages sooner in the diffusion-based scheme than in the level-based scheme, since it can receive from any neighbor node in the diffusion-based scheme. However, due to the higher communication overhead in the diffusion-based scheme, there are more message collisions and message retransmissions. Hence, the diffusion-based scheme has a longer synchronization time and a worse synchronization precision than the level-based scheme.

After obtaining the synchronization precision, we can decide the *synchronization interval* accordingly. The synchronization interval is about how often the source node initiates one round of clock synchronization. Suppose the maximum clock drift rate of all the sensor nodes is ρ. Given the synchronization precision δ and the required precision Δ of an application, the synchronization interval R must satisfy that $R \leq (\Delta - \delta)/\rho$.

Memory Usage: Memory usage is a critical issue for resource constrained sensor nodes. In the level discovery phase, the level-based approach requires memory to record a node's level, its parent nodes, and its children nodes. To tolerate up to t malicious nodes among its neighbor nodes, a normal node has to have a certain amount of memory set aside for children node so that the malicious nodes cannot prevent it from having normal children nodes by consuming this memory. In the synchronization phase of both level-based and diffusion-based approaches, each node only needs to record $2t + 1$ single-hop pairwise clock differences and $2t + 1$ source clock differences from its neighbor nodes.

IV. Clock Synchronization With Multiple Source Nodes

In our initial experiments, we observe that it took tens of seconds to synchronize a large sensor network that contains hundreds of sensor nodes, and some nodes cannot be synchronized. Our investigation revealed that this is mostly due to message propagation delays and increased occurrences of message collisions. Moreover, the nodes far away from the source node were not synchronized with a high precision due to the clock drift during the synchronization process. To reduce the synchronization time and improve the synchronization rate and synchronization precision, we propose to distribute multiple source nodes into the network, and make sensor nodes synchronize to the nearest source nodes. This approach is in essence similar to the typical approach (e.g., [29]–[32]) for localization in wireless sensor networks, where multiple anchor nodes that know their locations are deployed to help the other nodes estimate their locations.

Having multiple source nodes can also increase the robustness of the clock synchronization, so that sensor nodes can get synchronized from other source nodes even if the nearest source node fails. In hostile environments, it is possible for malicious attackers to compromise a small portion of the source nodes, though the source nodes are typically better protected from attacks than the normal ones. Having multiple source nodes also offers an opportunity to tolerate a number of compromised source nodes.

A. Extended Model

We assume all the normal source nodes are well synchronized to an external clock, for example, through GPS receivers. Suppose the IDs of the source nodes are well known by all the sensor nodes. We extend the clock synchronization model in Section II-B to accommodate synchronization with multiple source nodes.

1) Each node i maintains a *local clock* C_i.

2) Each node i may obtain a source clock difference δ_{i,S_j} between its local clock and the clock of a source node S_j by following the model in Section II-B.
3) To tolerate up to s malicious source nodes, each node i needs to obtain at least $2s+1$ source clock differences from distinct source nodes. Node i then chooses the median of the source clock differences as its final source clock difference $\delta_{i,S}$.
4) Each node i can estimate the *global clock* C_S by using its local clock and its source clock difference (i.e., $C_S = C_i + \delta_{i,S}$).

When all the source nodes are normal (i.e., $s = 0$), sensor nodes may synchronize to any source node.

B. Hop-Count Threshold

When multiple source nodes are used for clock synchronization, each node only needs to synchronize to the nearest $2s+1$ source nodes. Thus, it is unnecessary for each source node to propagate its clock synchronization messages to the entire network. As a result, we can significantly reduce the message propagation delay and message collisions. Therefore, we propose to limit the coverage area of each source node. Specifically, we set a suitable hop-count threshold on the maximum hop-count that a synchronization message can be forwarded. We certainly still need to guarantee that each sensor node can synchronize to $2s+1$ source nodes.

In the level-based approach, we can set the hop-count threshold by limiting the maximum level in each source node's level hierarchy. In the level discovery phase, a sensor node only chooses the neighbor nodes whose level are less than the hop-count threshold as its parent nodes. After receiving from $3t+1$ parent nodes, it sets its level as the median of these parent nodes' levels plus one. It is possible that a children node sends a smaller level number than some of its parent nodes; however, since a node sends its level discovery messages only after it has decided its parent set, there will be no loop in the level hierarchy. If a sensor node's level equals to the hop-count threshold, it stops sending level discovery messages to its neighbor nodes. In the synchronization phase, a sensor node may send synchronization messages only if its level is less than the hop-count threshold.

In the diffusion-based approach, we set an upper bound threshold on the maximum hop-count for the synchronization messages to be forwarded. We add a hop-count field in the synchronization messages. When a source node initiates one round of synchronization, it sets the hop-count in the messages to 0. Each sensor node only accepts a synchronization message whose hop-count field is less than the threshold. To tolerate up to t malicious neighbor nodes, a sensor node needs to receive $2t+1$ messages from neighbor nodes. Suppose the median of the hop-counts in the $2t+1$ messages is L. If L is less than the hop-count threshold, the sensor node sends out its synchronization messages, in which the hop-count equals to $L+1$; otherwise, it does not send any synchronization message.

A malicious nodes may attack the hop-count mechanism by manipulating the hop-count field in its messages; however, such attack has little impact on both schemes. Because each normal node decides its hop-count (or level) by using the median of the values from $2t+1$ neighbor nodes, it can tolerate the attacks from up to t malicious nodes.

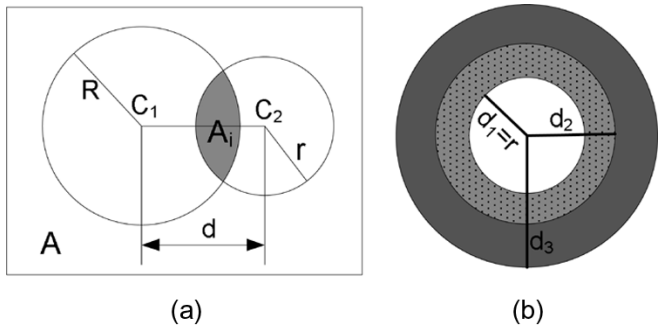


Fig. 2. Determining the hop-count threshold.

Determining the Hop-Count Threshold: In the following, we present a method to estimate the hop-count threshold. In Fig. 2(a), we assume that n sensor nodes are uniformly distributed in a rectangle field of area A.

We first calculate the maximum distance d_i from level i nodes to a source node in the level-based approach. Suppose a source node locates at point C_1 in Fig. 2(a). The source node has a transmission radius R, and all the other nodes have the same transmission radius r. Because all the level 1 nodes are in the transmission range of the source node, we have $d_1 = R$.

Now, consider the maximum distance d_2 from a level 2 node that locates at C_2 to the source node at C_1 in Fig. 2(a). Since this sensor node needs to find $3t+1$ level 1 nodes to be its parent nodes, we need guarantee that there exist at least $3t+1$ nodes in the shadow area A_i. That is

$$A_i \cdot \frac{n}{A} > 3t+1 \tag{2}$$

where n/A is the node density of the sensor network.

Suppose the distance between the two nodes is d. We can calculate the shadow area A_i of the circle intersection by

$$A_i = r^2 \cos^{-1}\left(\frac{d^2+r^2-R^2}{2dr}\right) + R^2 \cos^{-1}\left(\frac{d^2+R^2-r^2}{2dR}\right) - \frac{1}{2}\sqrt{(r+R-d)(d+r-R)(d+R-r)(d+r+R)}. \tag{3}$$

By combining (3) and (2), we can calculate the maximum distance $d_2 = d$ from level 2 nodes to the source node.

Similarly, as Fig. 2(b) shows, we can estimate the maximum distance d_3 for level 3 nodes by using $R = d_2$, $r = r$ in (3). Given the maximum distance d_i from level i nodes to the source node, we can estimate the maximum distance d_{i+1} for level $i+1$ nodes.

Given a maximum distance D from the farthest node to the source node, we can calculate the a level threshold L that satisfies $d_L \geq D$. In the level-based scheme, the level threshold functions as the hop-count threshold. Because the shadow area A_i in Fig. 3(a) increases along with R, we have $d_{i+2} - d_{i+1} > d_{i+1} - d_i$, where $i \geq 1$. This guarantees that we can find a hop-count threshold L given D.

In the diffusion-based approach, we can perform a similar calculation. But we should use (4) instead of (2), since a node

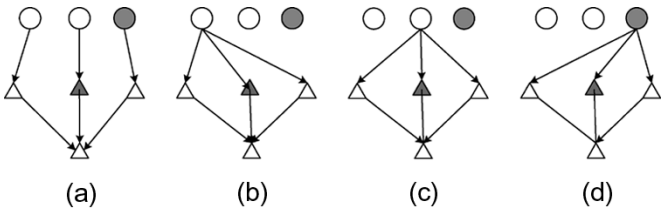


Fig. 3. Parents synchronize to multiple source nodes.

needs to receive $2t+1$ synchronization messages from neighbor nodes

$$A_i \cdot \frac{n}{A} > 2t + 1. \tag{4}$$

C. Multiple Source Nodes

We consider two situations, when all the source nodes are normal, and when some of the source nodes are potentially compromised. In both cases, each normal node can uniquely identify each source node using the unique pairwise key shared between them. Thus, an attacker cannot pretend to be these source nodes.

All Source Nodes are Normal: When all the source nodes are normal, sensor nodes can synchronize to the nearest source node. We can improve the synchronization performance by deploying multiple source nodes. First, sensor nodes may receive from a source node in shorter paths, so the accumulated synchronization error on the nodes along the path can be reduced. Second, multiple source nodes can reduce message collisions and, thus, shorten the synchronization time and improve the synchronization precision. Moreover, multiple source nodes may increase the synchronization rate in a randomly distributed sensor network.

In the level-based scheme, each source node builds a level hierarchy rooted at itself. For the neighbors of a source node, they choose the source node as the unique parent node. For a node more than one hop away from any source node, to tolerate up to t malicious neighbor nodes, it can choose either: 1) a set of $3t + 1$ parent nodes that synchronize to the same source node or 2) a set of $3t + 1$ parent nodes that may synchronize to different source nodes. In the synchronization phase, a node may obtain a source clock difference after receiving synchronization messages from $2t + 1$ parent nodes.

In the diffusion-based scheme, the neighbors of the source nodes can directly synchronize their clocks to a source node. For other nodes, they can synchronize their clocks after receiving synchronization messages from any $2t + 1$ neighbors.

Some Source Nodes are Potentially Compromised: To tolerate up to s malicious source nodes, a normal sensor node has to receive at least $2s + 1$ source clock differences from distinct source nodes. It is important that each node obtains each source clock difference for a given source node from a set of parent nodes that synchronize to the same source node.

Consider Fig. 3, in which the circles stand for source nodes, and the triangles stand for sensor nodes. Suppose the shadow nodes are malicious. For the bottom sensor node, one of its three neighbor nodes is malicious, and one of the three source node is malicious. Suppose malicious nodes may collude with each other. Consider Fig. 3(a). Since the malicious source node can control the source clock difference computed by the normal node below it, and the malicious neighbor node in the middle can modify the source clock difference received from the normal source node, these two malicious nodes can actually control two out of the three source clock differences received by the bottom node. As a result, even if the bottom node uses the median of the three source clock differences to synchronize its clock, there is no guarantee that it can correctly synchronize its clock. In contrast, if the bottom node synchronizes to each source node separately, as shown in Fig. 3(b)–(d), it can successfully filter out the effect of the malicious neighbor node in the cases of Fig. 3(b) and (c), but get an incorrect source clock difference in the case of Fig. 3(d). By further choosing the median from the source clock differences for the three sources, the bottom node can still synchronize its clock correctly.

Based on the above discussion, we revise the level-based and the diffusion-based clock synchronization as follows. In both level-based and diffusion-based schemes, the source node adds its identity into the messages that it initiates. In the level-based scheme, each source node independently builds a level hierarchy rooted at itself. When a sensor node's level in a source node's level hierarchy is no more than the level threshold, the sensor node records parent/children sets for the source node. In the synchronization phase, after one sensor node obtains a source clock difference from one source node, it sends synchronization messages to its children nodes that synchronize to the same source node. After obtaining $2s + 1$ source clock differences from different source nodes, the sensor node uses the median of the $2s+1$ source clock differences to adjust its clock. Similarly, in the diffusion-based scheme, to tolerate s malicious source node, a sensor node synchronizes its clock after obtaining $2s+1$ source clock differences from different source nodes separately.

By synchronizing sensor nodes to multiple source nodes, we can increase the robustness of our schemes to the malicious source nodes; however, the performance of our schemes become worse. To tolerate s malicious source nodes, a normal node needs to obtain $2s + 1$ source clock differences from different source nodes. For each source clock difference, the node needs to receive from $2t + 1$ neighbor nodes to tolerate up to t malicious neighbor nodes. Thus, the communication overhead is increased along with s and t. Because the coverage areas of different source nodes have overlaps, the messages from different source nodes may collide frequently. Due to the increased occurrences of message collisions, both the communication overhead and the synchronization time may increase substantially.

In the level-based scheme, each node needs to allocate memory to record the parent/children sets for multiple source nodes. In both level-based and diffusion-based clock synchronization, each node needs to record the candidate source clock differences from different neighbor nodes and different source nodes. Each node also records its neighbors' sequence numbers. Each node will receive $(2t+1)(2s+1)$ candidate source clock differences. Thus, the memory consumption for these source clock differences is bounded by $(2t + 1)(2s + 1)$. In the worst case when all the source nodes start clock synchronization at the same time, each normal node requires the amount of memory for $(2t+1)(2s+1)$ source clock differences. However, typically a normal node does not need to records all these values

TABLE I
SIMULATION PARAMETERS

Number of Nodes	50, 100, 150, 200
Simulation Area	60m × 60m
Transmission Range	20m
Physical Link Bandwidth	250 kbps
mac layer	802.11 with DATA/ACK
Clock Drift Rate ($\mu s/s$)	uniformly distributed in [0, 10]
Malicious neighbors	t = 0, 1, 3
Total Source nodes	S = 1, 9
Malicious source nodes	s = 0 when S=1 s=0,1,3 when S=9

at the same time and its actual memory consumption is usually less. This is because that a node can release the memory for the $2t+1$ candidate source clock differences from one source node after obtaining the source clock difference.

V. SIMULATION RESULTS

We studied both level-based and diffusion-based clock synchronization through simulation in ns2 [33]. Our goal is to gain a better understanding of the performance issues of the proposed techniques, which cannot be obtained through theoretical analysis. We implemented a new agent in ns2 to provide global clock synchronization for sensor nodes. We used a simple "Hello" protocol for nodes to discover their neighbor nodes.

Table I shows the parameters used in our experiments. The numbers of nodes n in a sensor network, which do not include the source nodes, are 50, 100, 150, and 200, respectively. All the nodes remain static after being randomly deployed in a 60 m × 60 m simulation area. We assume all the nodes have the same 20 m transmission range. The bandwidth of each physical link is 250 kb/s, as for MICAz motes [34]. Our simulation uses 802.11 with DATA/ACK as the MAC layer, in which an ACK message is sent back for a unicast DATA message, and no ACK message for broadcast DATA message. In our simulation, we did not enable the RTS/CTS/DATA/ACK pattern in the 802.11 protocol, since the control messages will introduce a large extra latency into the synchronization time and lead to a high collision rate. We simulate a node i's local clock as $C_i = (1+\rho_i)\cdot C_S$, where C_S is the clock of the source node S and ρ_i is node i's clock drift rate. Each ρ_i is randomly generated using a uniform distribution between 0 and 10 μs/s.

In our simulation, we first evaluate the proposed schemes when there is only one source node. In these experiments, we deploy a single source node in the center of the simulation area, and assume the unique source node is always trusted. For each sensor node, the number of malicious neighbor nodes t is set to be 0, 1, and 3, respectively. When $t = 0$, our scheme degenerates into an existing clock synchronization scheme (e.g., [5] and [6]), depending on the single-hop pairwise clock synchronization scheme adopted in our scheme. We then evaluate the proposed schemes when there are multiple source nodes. In our experiments, we deploy nine source nodes in the simulation are,a as shown in Fig. 4. The number of malicious source nodes is set to be 0, 1, and 3, respectively.

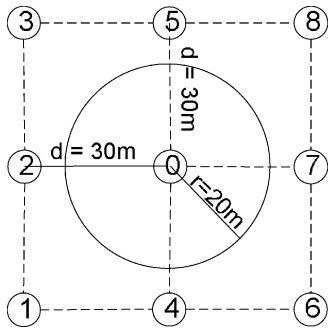


Fig. 4. Topology of multiple source nodes.

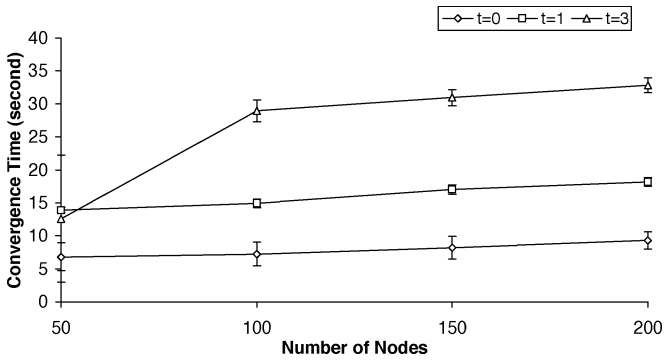


Fig. 5. Convergence time of level discovery.

In the following, we describe the simulation results in detail.

A. Single Source Node

When deploying a single source node, we study the performance of our schemes when they can tolerate up to t malicious sensor nodes. Each data point in the result figures is an average of ten simulation runs with identical configuration but different randomly generated node deployments. The Y axis error bars show the 95% confidence intervals.

We compare the level-based scheme and the diffusion-based scheme on synchronization rate, communication overhead, synchronization time, and synchronization precision. The synchronization rate is to measure the percentage of sensor nodes that can be synchronized. The communication overhead is about the total number of synchronization messages in one round of synchronization. The synchronization time is the duration between the start of clock synchronization and the time when the last sensor node that can be synchronized derives its clock. The synchronization precision is the maximum clock difference between any sensor node and the source node right after the sensor nodes are synchronized.

Convergence Time of Level Discovery: In the level-based approach, the convergence time of the level discovery phase is shown in Fig. 5. In our simulation, in order to reduce a node's level in the level hierarchy, after obtaining its level, each sensor

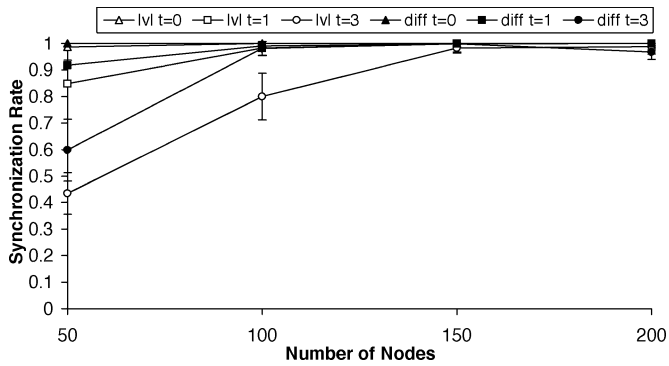


Fig. 6. Synchronization rate.

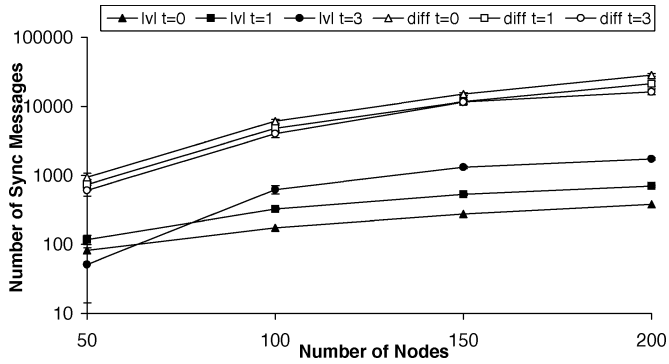


Fig. 7. One round communication overhead.

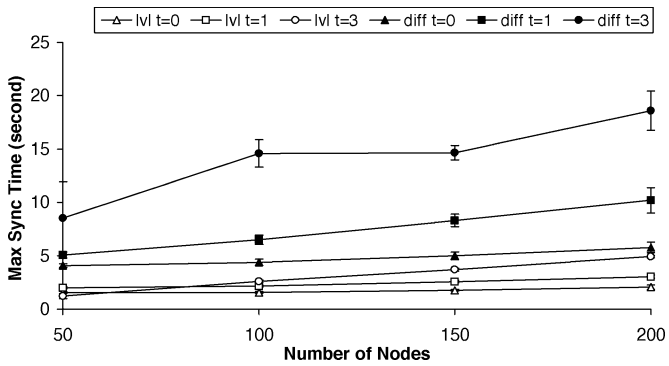


Fig. 8. Maximum synchronization time.

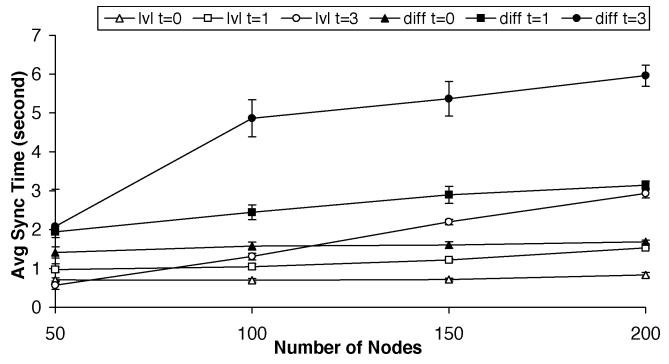


Fig. 9. Average synchronization time.

node waits for 1 s before it sends level discovery messages to its neighbor nodes.

Synchronization Rate: Fig. 6 shows the percentage of sensor nodes that can be synchronized. When $t = 3$ and $n = 50$, due to the relatively low density of the network, the level-based scheme can synchronize only 40% nodes, while diffusion-based scheme can synchronize 60% nodes. When n increases to 150, both schemes can synchronize almost all the sensor nodes. The diffusion-based scheme can synchronize more sensor nodes than the level-based scheme in the sparse sensor networks. When $t = 3$ and $n = 200$, due to the increased message collisions, some sensor nodes may not be synchronized in the level-based scheme.

Communication Overhead: In both schemes, the neighbors of the source node require only one synchronization message from the source node, and the nodes more than one hop away from the source node may receive $3t + 1$ messages from neighbor nodes.

Fig. 7 shows the number of synchronization message sent in one round of clock synchronization. One message can be retransmitted at most four times in our simulation. The diffusion-based approach has a much higher communication overhead than the level-based approach. The communication overheads increase along with the number of the nodes in both schemes. In the level-based scheme, the overhead also increases along with t, since nodes need to receive from more parent nodes.

Synchronization Time: We now examine the synchronization time, which has a significant impact on the synchronization precision.

Figs. 8 and 9 show the maximum and the average synchronization time to finish one round of global clock synchronization. The synchronization time increases along with t in both schemes. The level-based scheme can finish sooner than the diffusion-based scheme.

Synchronization Precision: Our simulation uses MAC layer timestamp by modifying the 802.11 protocol in ns2 to record the exact timestamp when a message is transmitted or received at the MAC layer.

Figs. 10 and 11 show the maximum and the average clock errors of all the nodes immediately after synchronizing their local clocks. The level-based scheme can provide a much better clock precision than the diffusion-based scheme. The major reason is that the clock drift during the synchronization time is greater in the diffusion-based scheme than that in the level-based scheme.

B. Multiple Source Nodes

We deploy nine source nodes in the network, as shown in Fig. 4. When up to s source nodes may be compromised, a sensor node has to obtain source clock differences from $2s + 1$ source nodes. In our simulation, we assume there may exist 0, 1, or 3 malicious source nodes. We study four scenarios $(s = 0)$, $(s = 1, t = 1)$, $(s = 1, t = 3)$, and $(s = 3, t = 3)$. In our configuration, each node is a one-hop neighbor of at least one source node. Thus, we do not need to specify the value of t when $s = 0$. We fix the number of sensor nodes to 200.

Hop-Count Threshold: We first need to decide the hop-count threshold to allow all the sensor nodes receive from $2s+1$ source nodes. When $s = 0$, all the source nodes are normal. Since the

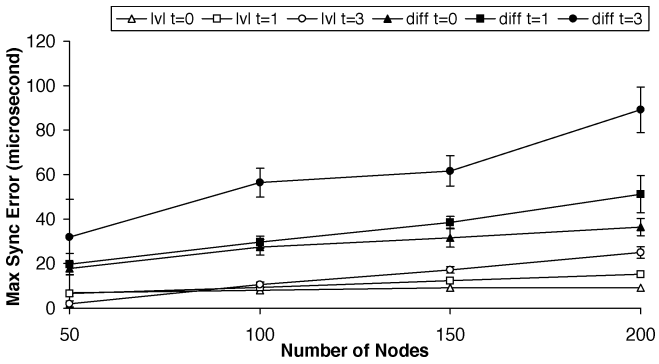


Fig. 10. Maximum synchronization error.

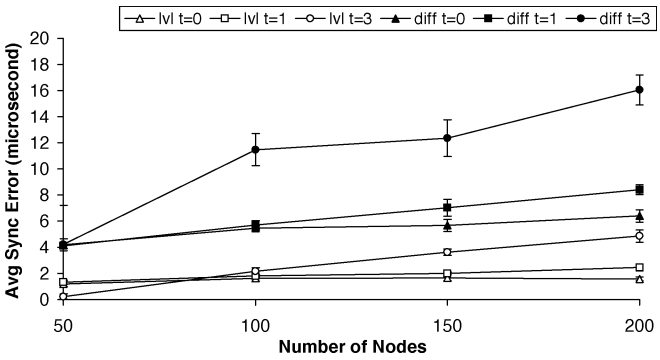


Fig. 11. Average synchronization error.

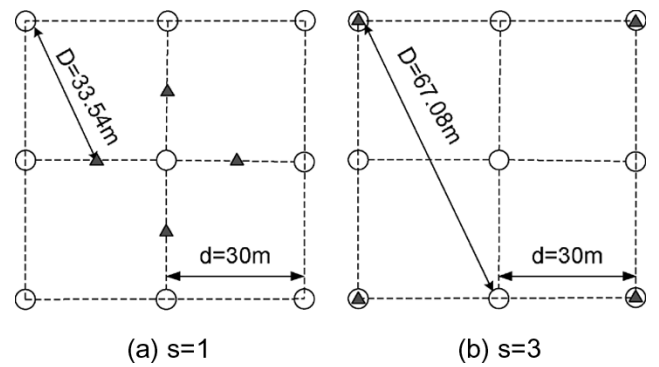


Fig. 12. Maximum distance from $2s + 1$ source nodes. (a) $s = 1$. (b) $s = 3$.

multiple source nodes can guarantee that all the sensor nodes can directly receive from at least one source node in our simulation configuration, the hop-count threshold is set to 1. When $s = 1$, each sensor node needs to receive from at least three source nodes. As Fig. 12(a) shows, the maximum distance for all the sensor nodes to receive from the nearest three source nodes is $D = 15 * \sqrt{5} = 33.54$ m. When $s = 3$, each node needs to receive from seven source nodes. Fig. 12(b) shows that the maximum distance for a sensor node to receive from the nearest seven source nodes is $D = 30 * \sqrt{5} = 67.08$ m. We can then use the method described in Section IV-B to calculate the hop-count thresholds. Table II shows the hop-count thresholds when $n = 200$.

Synchronization Rate: In all the scenarios, all the 200 sensor nodes can be synchronized.

TABLE II
HOP-COUNT THRESHOLDS WHEN $n = 200$ AND $S = 9$

Scheme	s=0	s=1, t=1	s=1, t=3	s=3, t=3
Level-based	1	2	3	6
Diffusion-based	1	2	3	5

Synchronization Time: When $s = 0$ and $s = 1$, due to the small hop-count thresholds, the message collision can be controlled. Therefore, all the source nodes may initiate the synchronization process at the same time. However, when $s = 3$, if all the source nodes synchronize at the same time, there are a large number of message collisions, making it difficult to synchronize the sensor nodes. To reduce the collisions, we divide the nine source nodes into five groups, that is, {1,8}, {2,6}, {3,7}, {4,5}, and {0}. The source nodes in the same group can initiate synchronization at the same time, since they are relatively far from each other and have fewer message collisions. In our simulation, each group initiates the synchronization process in an interval of 20 s in the level-based scheme, and 30 s in diffusion-based scheme. This arrangement increases the synchronization time and the synchronization error, but also improves the synchronization rate. There may exist better ways to arrange the order for source nodes to initiate the clock synchronization, but we consider it out of the scope of this paper.

Fig. 13(a) shows the maximum synchronization time in different scenarios. We can see that the synchronization time increases along with t and s. When $s = 0$, the whole network can be synchronized in 1 s, because all the sensor nodes can be directly synchronized to one source node.

When $s = 1$ and $t = 1$, one round of synchronization can be finished in 16 s by the level-based approach, and in 57 s by the diffusion-based approach. When $s = 1$ and $t = 3$, because a node far away from a source node needs to receive seven clock differences before sending its synchronization messages, the synchronization time increases to around 29 s in the level-based scheme, and around 75 s in the diffusion-based scheme. When $s = 3$ and $t = 3$, the synchronization time is quite long in both schemes. The level-based scheme needs around 2.5 min to finish one round of synchronization, while the diffusion-based scheme needs almost 4 min.

Synchronization Error: Fig. 13(b) shows the maximum synchronization error. When $s = 0$, the maximum synchronization error is less than 10 μs. When $s = 3$ and $t = 3$, this error is less than 1 ms in the level-based scheme, but around 2 ms in the diffusion-based scheme.

Communication Overhead: The communication overhead in the level-based scheme is moderate for sensor nodes, while the communication overhead of the diffusion-based scheme is greater than the level-based scheme. When $s = 0$, the message overheads in both schemes are less than 400. When $s = 3$ and $t = 3$, in one round of clock synchronization, each sensor node sends nearly 100 messages in the level-based scheme, while it sends around 850 messages in the diffusion-based scheme. Considering the resource constraint in sensor nodes, it makes the diffusion-based scheme not scalable to tolerate a large number of malicious source nodes.

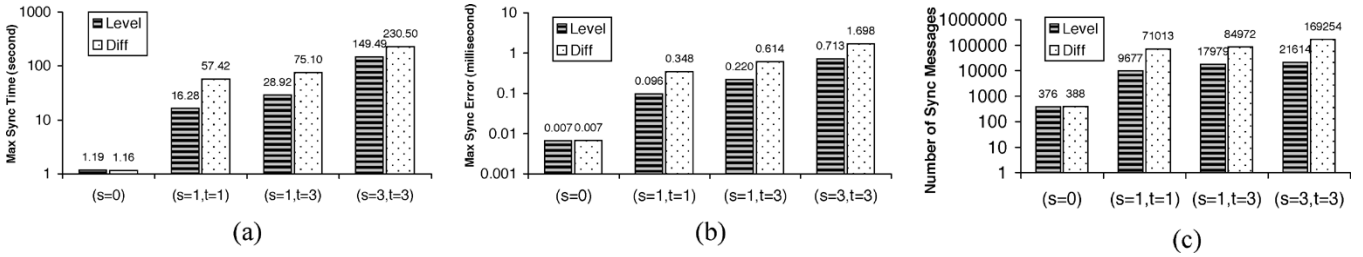


Fig. 13. Experimental results with multiple source nodes. (a) Maximum synchronization time. (b) Maximum synchronization error. (c) Communication overhead.

VI. Related Work

Clock synchronization has been studied for many years. Early techniques (e.g., NTP [3]) are mainly for clock synchronization in wired networks. However, such techniques usually assume there is unlimited computing resource and network bandwidth and, thus, are not suitable for resource constrained sensor networks.

Several clock synchronization techniques (e.g., [4]–[11] and [35]–[38]) have been proposed for sensor networks recently. Elson *et al.* developed the RBS scheme for pairwise, as well as multidomain clock synchronization [4], which eliminates the uncertainty of send time and access time from the clock reading error by using a reference broadcast node. Dai and Han improved RBS by reducing the communication overhead in each broadcast domain to two broadcasts [36]. Palchaudhuri *et al.* proposed a probabilistic clock synchronization based on RBS [10]. Generiwal *et al.* proposed a hierarchical clock synchronization schfeme named TPSN for sensor networks [5], assuming clock synchronization messages are timestamped at mac layer. Sichitiu *et al.* developed a lightweight scheme to deterministically estimate the bounds on both the relative clock drift and offset between two sensor nodes, which can be used to synchronize their clocks [9]. Li and Rus proposed global clock synchronization techniques only based on local diffusion of clock information [7]. However, all of the above techniques assume benign sensor networks, but cannot survive malicious attacks from compromised nodes. The recent result in [17] can detect malicious attacks against clock synchronization, and aborts when such an attack is detected. However, as discussed in Section I, this approach will lead to denial of clock synchronization in presence of attacks. In contrast, the techniques proposed in this paper can tolerate malicious attacks from compromised nodes.

Traditional fault-tolerant clock synchronization in distributed systems has undergone substantial research (e.g., [12]–[16] and [39]). A common theme of these techniques is to use redundant messages to deal with malicious participants that may behave arbitrarily. However, these fault-tolerant schemes usually have very high communication overhead (especially the consistency-based approaches such as COM and CSM). Moreover, to prevent malicious participants from forging messages originated from normal ones, these schemes use either digital signatures (e.g., CSM [15] and HSSD [16]), or a broadcast primitive that requires simultaneous broadcast from multiple nodes, which will result in message collision in wireless sensor networks. Compared with these techniques, our techniques have much less communication overhead, and use pairwise key to authenticate the synchronization messages instead of using the heavy digital signatures.

VII. Conclusion

In this paper, we presented two secure and resilient global clock synchronization techniques for sensor networks. We adopted a model where all the sensor nodes synchronize their clocks to a common source, which is assumed to be well synchronized to an external clock. Our approaches guarantee that normal nodes can synchronize their clocks to the common source node even if each normal node has up to t colluding malicious nodes among its neighbor nodes. We proposed to increase the performance by deploying multiple source nodes, and extend our approaches to tolerate malicious source nodes. The simulation results indicate that these approaches are promising for the current generation of sensor networks.

Several issues are worth further investigation. First, we would like to seek more efficient techniques for clock synchronization in sensor networks. In particular, we would like to study how to exploit broadcast authentication for clock synchronization without incurring DoS attacks. Moreover, we would like to integrate clock synchronization techniques with power-saving techniques in sensor network applications.

Acknowledgment

The authors would like to thank the anonymous reviewers for their useful comments.

References

[1] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho, "Grid coverage for surveillance and target location in distributed sensor networks," *IEEE Trans. Comput.*, vol. 51, no. 12, pp. 1448–1453, Dec. 2002.

[2] D. Tian and N. D. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," in *Proc. 1st ACM Int. Workshop on Wireless Sensor Networks and Appl.*, Sep. 2002, pp. 32–41.

[3] D. Mills, "Internet time synchronization: The network time protocol," *IEEE Trans. Commun.*, vol. 39, no. 10, pp. 1482–1493, Oct. 1991.

[4] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *ACM SIGOPS Operating Syst. Rev.*, vol. 36, pp. 147–163, 2002.

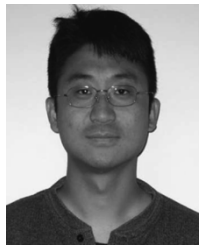
[5] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. 1st Int. Conf. Embedded Networked Sensor Syst.*, 2003, pp. 138–149.

[6] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in *Proc. 2nd ACM Conf. Embedded Networked Sensor Syst.*, Nov. 2004, pp. 39–49.

[7] Q. Li and D. Rus, "Global clock synchronization in sensor networks," in *Proc. IEEE INFOCOM*, Mar. 2004, pp. 214–226.

[8] M. Mock, R. Frings, E. Nett, and S. Trikaliotis, "Clock synchronization for wireless local area networks," in *Proc. 12th Euromicro Conf. Real-Time Syst.*, Jun. 2000, pp. 183–189.

[9] M. Sichitiu and C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2003, pp. 1266–1273.
[10] S. PalChaudhuri, A. Saha, and D. Johnson, "Adaptive clock synchronization in sensor networks," in *Proc. Inf. Process. Sensor Netw.*, Apr. 2004, pp. 340–348.
[11] A. Hu and S. D. Servetto, "Asymptotically optimal time synchronization in dense sensor networks," in *Proc. 2nd ACM Int. Workshop on Wireless Sensor Netw. Appl.*, Sep. 2003, pp. 1–10.
[12] B. Barak, S. Halevi, A. Herzberg, and D. Naor, "Clock synchronization with faults and recoveries," in *Proc. 19th Annual ACM Symp. Principles of Distrib. Comput.*, 2000, pp. 133–142.
[13] A. Olson and K. Shin, "Fault-tolerant clock synchronization in large multicomputer systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, no. 9, pp. 912–923, Sep. 1994.
[14] T. K. Srikanth and S. Toueg, "Optimal clock synchronization," *J. ACM*, vol. 34, no. 3, pp. 626–645, 1987.
[15] L. Lamport and P. Melliar-Smith, "Synchronizing clocks in the presence of faults," *J. ACM*, vol. 32, no. 1, pp. 52–78, 1985.
[16] D. Dolev, J. Y. Halpern, B. Simons, and R. Strong, "Dynamic fault-tolerant clock synchronization," *J. ACM*, vol. 42, no. 1, pp. 143–185, 1995.
[17] S. Ganeriwal, S. Capkun, C. Han, and M. B. Srivastava, "Secure time synchronization service for sensor networks," in *Proc. ACM Workshop on Wireless Security*, Sep. 2005, pp. 97–106.
[18] Y. Hu, A. Perrig, and D. Johnson, "Packet leashes: A defense against wormhole attacks in wireless ad hoc networks," in *Proc. INFOCOM*, Apr. 2003, pp. 1976–1986.
[19] A. Perrig, R. Canetti, D. Song, and D. Tygar, "Efficient authentication and signing of multicast streams over lossy channels," in *Proc. IEEE Symp. Security and Privacy*, May 2000, pp. 56–73.
[20] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar, "SPINS: Security protocols for sensor networks," in *Proc. 7th Ann. Int. Conf. Mobile Comput. Netw.*, Jul. 2001, pp. 521–534.
[21] D. Liu and P. Ning, "Multi-level μTESLA: Broadcast authentication for distributed sensor networks," *ACM Trans. Embedded Comput. Syst.*, vol. 3, no. 4, pp. 800–836, 2004.
[22] N. Gura, A. Patel, and A. Wander, "Comparing elliptic curve cryptography and RSA on 8-bit CPUs," in *Proc. Workshop n Cryptographic Hardware and Embedded Syst.*, Aug. 2004, pp. 119–132.
[23] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proc. 10th ACM Conf. Compute. Commun. Security*, Oct. 2003, pp. 52–61.
[24] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proc. IEEE Symp. Res. Security and Privacy*, 2003, pp. 197–213.
[25] W. Du, J. Deng, Y. S. Han, and P. Varshney, "A pairwise key predistribution scheme for wireless sensor networks," in *Proc. 10th ACM Conf. Comp. Commun. Security*, Oct. 2003, pp. 42–51.
[26] D. Liu, P. Ning, and W. Du, "Detecting malicious beacon nodes for secure location discovery in wireless sensor networks," in *Proc. 25th Int. Conf. Distrib. Comput. Syst.*, Jun. 2005, pp. 609–619.
[27] J. Newsome, R. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: Analysis and defenses," *Proc. IEEE Int. Conf. Inf. Processing in Sensor Netw.*, pp. 259–2168, Apr. 2004.
[28] B. Parno, A. Perrig, and V. Gligor, "Distributed detection of node replication attacks in sensor networks," in *Proc. IEEE Symp. Security and Privacy*, May 2005, pp. 49–63.
[29] A. Savvides, C. Han, and M. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Proc. ACM MobiCom*, Jul. 2001, pp. 166–179.
[30] A. Savvides, H. Park, and M. Srivastava, "The bits and flops of the n-hop multilateration primitive for node localization problems," in *Proc. ACM WSNA*, Sep. 2002, pp. 112–121.
[31] D. Niculescu and B. Nath, "Ad hoc positioning system (APS) using AoA," in *Proc. IEEE INFOCOM*, Apr. 2003, pp. 1734–1743.
[32] A. Nasipuri and K. Li, "A directionality based location discovery scheme for wireless sensor networks," in *Proc. ACM WSNA*, Sep. 2002, pp. 105–111.
[33] The Network Simulator—ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns/
[34] Crossbow Technology Inc. (2005, May) Wireless sensor networks. [Online]. Available: http://www.xbow.com/Products/Wireless_Sensor_Networks.htm
[35] J. Elson and K. Römer, "Wireless sensor networks: A new regime for time synchronization," in *Proc. 1st Workshop on Hot Topics Netw.*, Oct. 2002, pp. 149–154.
[36] H. Dai and R. Han, "Tsync: A lightweight bidirectional time synchronization service for wireless sensor networks," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 8, no. 1, pp. 125–139, 2004.
[37] J. Greunen and J. Rabaey, "Lightweight time synchronization for sensor networks," in *Proc. 2nd ACM Int. Workshop on Wireless Sensor Netw. Appl.*, Sep. 2003, pp. 11–19.
[38] K. Römer, "Time synchronization in ad hoc networks," in *Proc. 2nd ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2001, pp. 173–182.
[39] F. Cristian, "Probabilistic clock synchronization," *Distrib. Comput.*, vol. 3, no. 3, pp. 146–158, 1989.



Kun Sun received the B.S. and M.E. degrees from the Department of Computer and System Science, Nankai University, Tianjin, China, in 1997 and 2000, respectively. He is currently working towards the Ph.D. degree in the Department of Computer Science, North Carolina State University (NCSU), Raleigh.

Before joining NCSU, he was Member of Technical Staff at Bell Laboratories for one year. His research focuses on wireless networks, especially on the security issues in wireless ad hoc networks and wireless sensor networks.

Peng Ning (M'99) received the B.S. degree in information science and the M.E. degree in communication and electronic systems from the University of Science and Technology, Anhui, China, in 1997 and 1994, respectively, and the Ph.D. degree in information technology from George Mason University, Fairfax, VA, in 2001.

He is currently an Assistant Professor of Computer Science in the College of Engineering, North Carolina State University, Raleigh. His research interests are mainly in computer and network security. His recent work is mostly in intrusion detection and security in ad hoc and sensor networks.

Dr. Ning is a founding member of the NCSU Cyber Defense Laboratory and a member of the NCSU/Duke Center for Advanced Computing and Communication (CACC). He is also a member of the Association for Computing Machinery (ACM), the ACM SIGSAC, and the IEEE Computer Society. He is a recipient of the National Science Foundation (NSF) Faculty Early Career Development (CAREER) Award.

Cliff Wang (S'93–M'04) received the Ph.D. degree in computer engineering from North Carolina State University, Raleigh, in 1996.

He is currently the Program Director for the Army Research Office's Information Assurance Program and manages a large portfolio of advanced information assurance research projects. He is also appointed as an Associate Faculty Member of the Computer Science Department, College of Engineering, North Carolina State University. His research interests are in the area of the development of secure protocols and algorithms for wireless sensor networks, wireless mobile ad hoc networks, and optimization of sensor network coverage and deployment.