A Secure and Scalable Time Synchronization Protocol in IEEE 802.11 Ad Hoc Networks

Lin Chen, Jean Leneutre
École Nationale Supérieure des Télécommunications
46, Rue Barrault, 75634 Paris Cedex 13, France
{Lin.Chen, Jean.Leneutre}@enst.fr

Abstract

IEEE 802.11 standards support ad hoc mode which are ad hoc networks with all nodes within each other's transmission range. Time synchronization is crucial in such ad hoc networks. A distributed Timing Synchronization Function (TSF) is specified by IEEE 802.11 to provide synchronization service, but it suffers from the scalability problem due to its inefficient synchronization mechanism. Furthermore, TSF is designed without taking into account security. Such an insecure time synchronization protocol may cause serious problems on the applications and protocols based on synchronized time. To the best of our knowledge, currently no secure time synchronization mechanisms are proposed for such environments. In this paper we propose our scalable and secure time synchronization protocol SSTSP. We prove that SSTSP can synchronize the networks with the maximum clock difference under $20\mu s$ *without any uncontinuous leaps in clocks. We also conduct simulations to evaluate the performance of SSTSP. The results show that without attacks, the performance of our approach is significantly superior to TSF and is among the best of currently proposed solutions in terms of accuracy and scalability. Besides, our approach can maintain the network synchronized even under malicious attacks.*

1 Introduction

IEEE 802.11 standards support the peer-to-peer mode, Independent Basic Service Set (IBSS), which is an ad hoc network with all nodes within each other's transmission range. Time synchronization is crucial in such ad hoc networks. It is a key function to perform power management and to support the medium access control protocol in the Frequency Hoping Spread Spectrum version of the physical layer. It also plays an important role in the support of QoS in ad hoc networks, particularly for real-time applications.

The dynamic nature of ad hoc networks, the non-determinism of the wireless channel and the lack of reference nodes make time synchronization a challenging task in ad hoc networks. An ideal synchronization mechanism for ad hoc networks should be robust to mobility and topology changes, efficient in terms of traffic and processing cost, scalable and secure. We pay a special emphasis on the security aspect because recently many mechanisms have been proposed to address the time synchronization problem in ad hoc networks [1], [4], [5], but most of them do not take into account security, although ad hoc networks are much more vulnerable to various attacks than traditional wired networks. As a result, the proposed solutions are very vulnerable to various kinds of attack. An attacker can easily modify or replay a time synchronization message. It can also send forged times to desynchronize the network or disturb the receiver's clock. Such an insecure time synchronization protocol further causes serious problems on the applications and protocols based on synchronized time. Nodes may fail to be activated because of the incorrect time estimation, which may further cause serious problems such as failing to respond to important events and packet loss.

In this paper we propose our secure and scalable synchronization protocol. We start by analyzing the core problems of existing synchronization mechanisms for ad hoc networks. Based on the analysis, we propose our scalable and secure time synchronization protocol SSTSP. We then provide security analysis and conduct simulations to evaluate the performance of SSTSP. Finally we conclude the paper in Sec.6.

2 Related Work

IEEE 802.11 standards specify the ad-hoc-mode Timing Synchronization Function (TSF) for IEEE 802.11 ad hoc networks (IBSS) [3] in which time synchronization is achieved by periodical time information exchange through beacons containing timestamps and other parameters. Each

Proceedings of the 2006 International Conference on Parallel Processing Workshops (ICPPW'06)
0-7695-2637-3/06 $20.00 © 2006 IEEE

node maintains a local TSF clock counting in increments of microseconds. All nodes in the IBSS compete for beacon transmission every Beacon Period (BP). At the beginning of each BP, there is a beacon generation window consisting of $w + 1$ slots each of length $aSlotTime$, where w is a parameter defined by system. Each node calculates a random delay uniformly distributed in $[0, w] \times aSlotTime$ and schedules to transmit a beacon when the delay timer expires. If a beacon is received before the random delay timer has expired, the node cancels the pending beacon transmission. Upon receiving a beacon, the node sets its local clock to the timestamp of the beacon if the value of the timestamp is later than its local clock.

In spite of its distributed nature and its efficiency in terms of communication cost, TSF has the following problems when applied to large scale ad hoc networks:

- *Fastest node*[1] *asynchronization:* As identified in [4], the clock of the fastest node may drift away, because it may not get a chance to transmit its beacon. Since the fastest node does not synchronize itself to other nodes, its clock will keep drifting away from others. The problem becomes more severe when the number of nodes of the network increases.
- *Beacon collision:* As the number of nodes increases, the synchronization beacon transmission contentions uprise accordingly. As a result, in large networks, due to repeated collisions, synchronization beacons can hardly be successfully transmitted and some nodes may fail to synchronize with others.

ATSP was proposed in [4] to solve the *fastest node asynchronization* problem. The basic idea is to let the fastest node compete for beacon transmission every BP and let other nodes compete only every I_{max} BPs. The parameter I_{max} should be carefully chosen to reach a compromise between scalability and stability. As an improved version of ATSP, the authors propose TATSP in which the nodes are dynamically classified into three tiers according to the clock speed. The nodes in tier 1 compete for beacon transmission in every BP and the nodes in tier 2 compete once in a while and the nodes in tier 3 rarely compete. SATSF is another synchronization protocol proposed in [10] compatible with TSF. In SATSF, node i competes for beacon transmission every $FFT(i)$ BPs. $FFT(i)$ is adjusted at the end of each BP in the way that fast nodes will gradually increase their FFT value, thus competing more frequently than slow nodes.

ASP is proposed in [9] to synchronize multi-hop ad hoc networks. The basic idea is to synchronize the whole network by fulfilling two tasks: to increase the successful transmission probability for faster nodes and to spread the faster time information throughout the whole network. The first task is achieved by increasing the beacon transmission priority of a node who has faster time and by cutting down the priorities of the others. When some slower nodes get enough information to accomplish synchronization by themselves, their beacon transmission priorities are increased to carry out the second task.

[1] proposes a mechanism which differs from the idea of giving faster nodes higher priorities. In the mechanism all nodes participate equally in the synchronization of the network. The authors define a controlled clock, which is an adjusted clock of the real clock, and a parameter $s = \frac{controlled\ clock}{real\ clock}$. Each node participate the contention with probability p every T_DELAY BPs if no beacons are received within last T_DELAY beacons. When receiving a beacon, the node updates s and p to synchronize to the sender of the beacon.

In spite of the numerous time synchronization protocols proposed for ad hoc networks, most of them have not been built with security in mind. To our knowledge, there exist very few propositions on the secure time synchronization protocols in the literature among which [7] mainly focus on a specific type of attack called delay attack. The authors propose two approaches to detect and accommodate the delay attack. One approach uses the generalized extreme studentized deviate (GESD) algorithm to detect multiple outliers (malicious time offset). The other uses a threshold based on a time transformation technique to filter out the outliers. [8] proposes several protocols for sensor networks to secure pairwise time synchronization over single hop and multiple hops. The author further extend their efforts to secure group time synchronization. They propose the lightweight secure group synchronization protocol to counter external attacks and the secure group synchronization protocol to counter both external and internal attacks but at the price of the heavy traffic overhead and the lack of scalability.

3 Scalable Secure Time Synchronization Procedure (SSTSP)

3.1 Design Philosophy

Although TSF provides an efficient distributed synchronization mechanism for WLAN in terms of traffic overhead, it suffers from the scalability problem due to its beacon contention scheme. Besides, it is vulnerable to various malicious attacks. In this section we address the above two problems which are vital to build a scalable and secure synchronization protocol.

First we argue that the root of the scalability problem in TSF lies in the fact that the increase in the number of nodes in the network decreases the synchronization beacon

[1]Throughout the paper we mean by the fastest node the node whose clock advances the fastest. The argument to give priority to the fastest node in TSF is to avoid backward leaps in time, which is a necessary requirement for some applications.

emission opportunity of the fastest node or a subset of the fastest nodes. Some protocols improving TSF increase the successful emission probability of the fastest nodes by attributing them priority with respect to other nodes in the network. These mechanisms are significantly more scalable than TSF, but since they follow the same contention mechanism as TSF, the scalability problem is not totally solved. Furthermore, they usually depend on the observation of the beacons to find and locate the fast nodes, which may increase the latency of synchronization.

Our approach, however, addresses the scalability problem from another angle. In our approach, all nodes content to emit the synchronization beacon at the beginning following the contention mechanism of TSF. The winner becomes the reference node and emits a beacon in the beginning of every BP without random delay. Other nodes synchronize their local clocks to the reference node until the reference node leaves the network, when another round of contention begins. To synchronize to the reference node, a node adjusts its clock parameters to gradually catch up with the pace of the reference clock in order to avoid backward and uncontinuous leaps in time. All nodes have the equal chance to become the reference node, but the contention takes place only when the formal reference node leaves and once the reference node is established, other nodes disable their beacon emission and synchronize their local clocks to the reference node each BP. The proposed mechanism maintains the distributed nature of the synchronization process while removes the scalability problem from its root. By making the full use of every received synchronization beacon and adopting a fine adjustment mechanism, we achieve significantly better synchronization accuracy than TSF and avoid all the backward or other uncontinuous leaps in local adjusted clock. Furthermore, by carefully choosing parameters, our mechanism is robust to the change of the reference node and the loss of synchronization beacons.

Furthermore, our approach can detect malicious synchronization attacks and prevent networks from being desynchronized by malicious nodes using erroneous time values. To achieve this goal, we use μTESLA [2], a recently proposed technique base on one-way hash chain, to protect synchronization beacons against external attackers. On the other hand, each node set a threshold called "guard time". A beacon is rejected if the difference between the timestamp in the beacon and local timestamp is beyond the threshold. This mechanism, based on the fact that the difference between any two clocks cannot drift unboundedly within a certain period of time, can counter the internal attacks and other attacks such as delay attacks and replay attacks. Note that performing asymmetric cryptographic operations usually takes up to hundreds of milliseconds depending on the CPU capacity of the nodes, which may increase significantly the synchronization error. In contrast, hash functions

Notations

s_i: Hash chain seed of node i
n: Hash chain length
T_0: Start time of the Hash chain
$h^j(s_i)$: The j^{th} element of node i's Hash chain
BP: Beacon period, typical value is 0.1s
t_i: Local unadjusted time of node i
$c_i(t)$: Local adjusted time of node i at local time t. Our goal is to synchronize $c_i(t)$
t_i^j: Local unadjusted time of node i when receiving the beacon in the j^{th} BP
t_{ref}^j: Local adjusted time of the reference node when emitting the beacon in the j^{th} BP
k^j, b^j: Coefficient and offset parameter to be adjusted when receiving the beacon in the j^{th} BP
t_p: Transmission and propagation delay
T^m: Local expected emission time of the synchronization beacon in the m^{th} BP, $T^m = T_0 + m * BP$ if $m > 1$
ts_{ref}^j: Adjusted timestamp value obtained from the beacon emitted by the reference node in the j^{th} BP. $ts_{ref}^j = t_{ref}^j + t_p$. ts_{ref}^j is estimated at the receiver side.
$(t_i^j)^*$, $(t_{ref}^j)^*$, $(ts_{ref}^j)^*$: Expected values of t_i^j, t_{ref}^j, ts_{ref}^j.
ϵ: Maximum error between ts_{ref}^j and t_{ref}^j, normally $\epsilon < 5\mu s$
δ: Guard time

are three to four orders of magnitude faster than asymmetric operations and can be performed in an on-the-fly way such that it causes almost no additional delay. Furthermore, due to its nature, nodes in ad hoc networks may be resource constrained. It is sometimes expensive even prohibited for such nodes to perform heavy asymmetric cryptographic operations. Another alternative is to use Keyed-hash Message Authentication Codes (HMAC) based on symmetric key shared between each pair of nodes [8]. This choice may pose scalability problem in that it requires N^2 keys in a network of N nodes and $(N-1)$ HMACs should be calculated to protect a beacon, which is clearly impossible to perform in the on-the-fly way when N is large. The above reasons push us to use μTESLA in our approach.

3.2 Assumptions and Requirements

To use one-way hash chains, we need some mechanism for a node to distribute an authenticated element $h^n(s_i)$ in its hash chain. A traditional approach is to let each node use its public key sign the hash chain element. Alternatively, a node can securely distribute an authenticated hash chain element using only symmetric-key cryptography [11] or non-cryptographic approaches [12].

We also assume that the synchronization beacons are timestamped below MAC layer. Thus, we remove the most significant non-deterministic factor of the end-to-end delay of the beacons, medium access waiting time.

3.3 Synchronization Procedure

Node initiation

Each node i picks a random seed s_i and generates its hash chain based on s_i: $h(s_i)$, $h^2(s_i)$, ..., $h^n(s_i)$. The last element $h^n(s_i)$ is authenticated and published within the network. The reference time T_0 corresponding to the start time of the hash chain is also published (e.g. T_0 can be configured and published by the first node arriving in the network or be integrated into the synchronization beacons). Suppose the beacons are expected to be emitted at time $T_0 + j * BP$ ($j = 1, 2, ..., n$). Each element of the hash chain $h^{n-j}(s_i)$ is used as the key to secure the synchronization beacon sent by node i in the time interval $[T_0 + j * BP - BP/2,\ T_0 + j * BP + BP/2]$ if node i is the reference node. Node i, in its synchronization beacon sent in the above time interval, discloses the element $h^{n-j+1}(s_i)$ ($j > 1$), allowing other nodes to authenticate previously received beacons sent by itself in last time interval $[T_0+(j-1)*BP-BP/2,\ T_0+(j-1)*BP+BP/2]$.

Our scalable and secure time synchronization procedure (SSTSP) consists of two phases: the coarse synchronization phase and the fine-grained synchronization phase.

Coarse synchronization phase

This phase is actually the pre-synchronization phase lasting several BPs during which a node only scans beacons in order to acquire coarse synchronization before joining the network. The objective is to enable the application of one-way hash chains to secure the time synchronization procedure in the fine-grained synchronization phase. The coarse synchronization can also be achieved by calibration when a node joins the network.

In this phase the new arriver scans beacons and adjusts its own clock based on the received timestamps. To counter attacks in this phase, a node collects a number of timestamps and computes the offsets between these timestamps and its own timestamps when receiving them. It then eliminates biased offsets. The mechanism proposed in [7] based on threshold can be applied here. Note that a loose threshold is chosen since the objective is to achieve coarse time synchronization. Finally the node averages the rest valid offsets and uses the averaged value to adjust its own clock t_i. In this phase $c_i(t_i)$ is set to t_i.

It is interesting that our approach uses μTESLA to secure the synchronization process while μTESLA itself requires a loose synchronization. It is not contradictory in that μTESLA only requires a loose synchronization which is easy to achieve and once the hash chain scheme is established and the fine-grained synchronization is secured, we can continue to use the hash chain scheme based on the synchronized time to further maintain the time synchronization.

Fine-grained synchronization phase

In this phase, each node competes to be the reference node if it has not heard the synchronization beacon in the last l BPs. A larger value of l makes the mechanism more robust since the failure to receive a beacon may be due to collision or temporary wireless channel instability other than the leave of the reference node. As price, a larger l increases the synchronization error when the reference node changes. In case of collision, the contention may last several BPs. The contention mechanism is the same as in TSF. The winner becomes the reference node and emits a beacon in the beginning of every BP without random delay. Other nodes synchronize their local clocks to the reference node until the reference node leaves the network, when another round of contention begins. A node joining the network does not participate in the contention until it is synchronized with the network. We use μTESLA scheme to protect the beacons. The synchronization beacon sent by the reference node ref in time interval j is:

$$< B,\ j,\ h_{ref}^{n-j}(B,\ j),\ h^{n-j+1}(s_{ref}) >$$

where B is the original unsecured synchronization beacon, $h_{ref}^{n-j}(B,\ j)$ denotes the HMAC output using $h^{n-j}(s_{ref})$ as the key applied to $(B,\ j)$, $h^{n-j+1}(s_{ref})$ is the disclosed key corresponding to the last interval (interval $(j-1)$).

Each node i temporarily stores the recently received beacons. Upon receiving a new beacon from the reference node ref, node i performs the following checks:

- Node i checks whether interval j corresponds to the current time interval.
- If the above check passes, node i further checks the validity of the disclosed key $h^{n-j+1}(s_{ref})$ in the beacon by verifying whether $h^{j-1}(h^{n-j+1}(s_{ref}))$ equals to the published element $h^n(s_{ref})$. In case of success, i checks the authenticity and the integrity of the beacon received in last interval using disclosed key $h^{n-j+1}(s_{ref})$. Node i can store previously authenticated disclosed key $h^{n-j+2}(s_{ref})$ to reduce processing overhead. In this case only one hash operation is needed instead of $j-1$.
- If above two checks pass, i checks whether the difference between the timestamp in the beacon and local timestamp is under the threshold δ. Otherwise the beacon may be replayed or delayed or the timestamp is forged by an internal attacker. We choose a tighter threshold here than that in the coarse synchronization phase. [8] and [7] discuss the parameter δ.

If all the above tests pass, node i then adjusts its local clock using the authenticated beacon $(j-1)$ and $(j-2)$. Note that beacon j cannot be used for clock adjustment until its integrity is verified. In SSTSP each node i has two clocks: an original clock and an adjusted clock. The original clock is the hardware clock of the node, e.g. a 64-bit counter with the resolution of $1\mu s$ in the IEEE 802.11 stan-

dard. The adjusted clock takes t_i, the time of the original clock as input and adjusts its value $c_i(t_i)$ according to the following relation:

$$c_i(t_i) = k^j * t_i + b^j, j = 1, 2, ... \quad (1)$$

Our objective is to synchronize the adjusted clocks of all the nodes in the network by repeatedly adjusting k^j and b^j ($k^j = 1$, $b^j = 0$ if $j \leq 2$) at each node when receiving the beacon in the j^{th} BP from the reference node. Compared with TSF, SSTSP has the following features:

- SSTSP achieves better accuracy than TSF via a more sophisticated adjustment scheme in which both the offset and the coefficient parameters are adjusted.
- There is no backwards or other uncontinuous leaps in local clock, which are important in some applications. TSF only guarantees that no backwards leaps exist.

The following equations illustrate the clock adjustment of SSTSP:

$$k^{j-1} * t_i^j + b^{j-1} = k^j * t_i^j + b^j \quad (2)$$

$$c_i((t_i^{j+m})^*) = k^j * (t_i^{j+m})^* + b^j = (ts_{ref}^{j+m})^* \quad (3)$$

$$\frac{t_i^{j-1} - t_i^{j-2}}{ts_{ref}^{j-1} - ts_{ref}^{j-2}} = \frac{(t_i^{j+m})^* - t_i^{j-1}}{(ts_{ref}^{j+m})^* - ts_{ref}^{j-1}} \quad (4)$$

$$(ts_{ref}^{j+m})^* = T^{j+m} \quad (5)$$

(2) follows the argument that the adjusted clock $c_i(t_i)$ is continuous at t_i^j. (3) indicates that the adjusted clock of node i is expected to converge to the reference clock at the expected receiving time of the beacon $(j + m)$. Before the convergence, the synchronization error is expected to decrease monotonously. m ($m > 1$) is the parameter of aggressiveness. A larger value of m increases the synchronization latency since the local clock converges slower to the reference node, while it avoids the synchronization error to be increased significantly when the reference node changes. (4) establishes the relation of the local clock and the adjusted clock of the reference node based on the linearity of the clocks[2]. (5) follows that the expected emission time of the $(j + m)^{th}$ beacon is T^{j+m}. By solving the equations (2), (3), (4), (5) containing 4 variables k^j, b^j, $(ts_{ref}^{j+m})^*$ and $(t_i^{j+m})^*$, we get:

$$k^j = \frac{(T^{j+m} - (k^{j-1} * t_i^j + b^{j-1})) * (ts_{ref}^{j-1} - ts_{ref}^{j-2})}{(t_i^{j-1} - t_i^{j-2}) * (T^{j+m} - ts_{ref}^{j-1}) + (t_i^{j-1} - t_i^j) * (ts_{ref}^{j-1} - ts_{ref}^{j-2})}$$

$$b^j = -\frac{(T^{j+m} - (k^{j-1} * t_i^j + b^{j-1})) * (ts_{ref}^{j-1} - ts_{ref}^{j-2}) * t_i^j}{(t_i^{j-1} - t_i^{j-2}) * (T^{j+m} - ts_{ref}^{j-1}) + (t_i^{j-1} - t_i^j) * (ts_{ref}^{j-1} - ts_{ref}^{j-2})} + k^{j-1} * t_i^j + b^{j-1}$$

[2] In this paper, the original clock is regarded as a linear function of real time within a short period of time. The adjusted clock is regarded as linear as long as no adjustment occurs during that period of time. A detailed clock model is described in [1].

By repeatedly updating k^j and b^j using received beacons from the reference node, the local adjusted clock of each node i gradually catches up with the pace of the reference clock and the network is hence synchronized.

3.4 Analysis of SSTSP

Lemma 1 shows that regardless of the initial value, the adjusted clock of i, c_i, converges to the adjusted timestamp of the reference node ts_{ref}, where $ts_{ref} = t_{ref} + t_p$.

Lemma 1: For any node i, its local adjusted clock, c_i, converges to ts_{ref}.

Proof: Let D_i^n be the difference between $c_i(t_i^n)$ and ts_{ref}^n when receiving the n^{th} beacon: $D_i^n = c_i(t_i^n) - ts_{ref}^n$. Let $(n + q) * BP + d_{n+q}$ be the time when the reference node emits the $(n + q)^{th}$ beacon ($q \geq 1$), where d_{n+q} is the time elapsed between the scheduled emission time of the beacon and its actual emission time. The timestamp in the beacon is adjusted at node i by adding t_p to ts_{ref}^{n+q}: $ts_{ref}^{n+q} = (n + q) * BP + d_{n+q} + t_p$.
By (3) we have:

$$k^n * (t_i^{n+m})^* + b^n = (ts_{ref}^{n+m})^* = (n+m) * BP + t_p \quad (6)$$

Apply (1) at t_i^n and t_i^{n+1} we get:

$$\begin{aligned} c_i(t_i^n) &= k^n * t_i^n + b^n = ts_{ref}^n + D_i^n \\ &= n * BP + d_n + t_p + D_i^n \quad (7) \\ c_i(t_i^{n+1}) &= k^n * t_i^{n+1} + b^n = ts_{ref}^{n+1} + D_i^{n+1} \\ &= (n + 1) * BP + d_{n+1} + t_p + D_i^{n+1} \quad (8) \end{aligned}$$

By the linearity of the clock we have:

$$\frac{(t_i^{n+m})^* - t_i^n}{(ts_{ref}^{n+m})^* - ts_{ref}^n} = \frac{(t_i^{n+m})^* - t_i^{n+1}}{(ts_{ref}^{n+m})^* - ts_{ref}^{n+1}} \quad (9)$$

Combining (6), (7), (8) and (9), we get:

$$\frac{D_i^{n+1}}{D_i^n} = \frac{(m-1) * BP - d_{n+1}}{m * BP - d_n} < \begin{cases} \frac{d}{m*BP-d} & m = 1 \\ \frac{(m-1)*BP}{m*BP-d} & m > 1 \end{cases}$$

Where $d = max(d_j)$, $(j > 1)$. Recursively we get:

$$\frac{D_i^{n+q}}{D_i^n} < \begin{cases} (\frac{d}{m*BP-d})^q & m = 1 \\ (\frac{(m-1)*BP}{m*BP-d})^q & m > 1 \end{cases}$$

Given any synchronization error threshold Δ and D_i^n, after $[log_{d/(m*BP-d)}(\Delta/D_i^n)]$ BPs (if $m = 1$) or $[log_{(m-1)*BP/(m*BP-d)}(\Delta/D_i^n)]$ BPs (if $m > 1$), the difference between the local adjusted clock of node i and the clock of the reference node will drop below the threshold. The adjusted clock thus converges to ts_{ref}.

Apply lemma 1 and $|ts_{ref} - t_{ref}| < \epsilon$, it is easy to prove that the maximum synchronization error is bounded by 2ϵ, typically $10\mu s$.

Lemma 2 studies the difference between c_i and ts_{ref} when the reference node changes.

Lemma 2: For any node i, let D_i^- and D_i^+ be the difference between c_i and ts_{ref} (ref is the old reference node) before and after the reference node changes, then $D_i^+ < (l+2) * D_i^-$.

Proof: It takes $(l+3)$ BPs before node i can re-adjust its local clock to the new reference clock: during the first $(l+1)$ BPs, the new reference node is elected via contention; during the following two BPs, each node validates the timestamp sent by the new reference node in previous BP and gets enough validated timestamps to adjust its local clock. Let $t_{ref}^n = n*BP + d_n$ (d_n is defined the same as in lemma 1) be the time when the last beacon is emitted by the old reference node. The beacon is received by i at local unadjusted time t_i^n. The difference between $c_i(t_i^n)$ and ts_{ref}^n is D_i^-. Let $t_{ref}^{n+l+3} = (n+l+3)*BP + d_{n+l+3}$ be the local time of the old reference node when the new reference node emits its beacon in the $(l+3)^{th}$ BP with which node i begins to adjust its local clock to the new reference clock. The difference between $c_i(t_i^{n+l+3})$ and ts_{ref}^{n+l+3} is D_i^+. Between ts_{ref}^n and ts_{ref}^{n+l+3}, the synchronization error cannot be controlled since no adjustment is done during this period. After ts_{ref}^{n+l+3} all the nodes synchronize to the new reference node, and the synchronization error decreases. We prove in the following that $D_i^+ < (l+2) * D_i^-$.
By (3) we get:

$$k^n * (t_i^{n+m})^* + b^n = (ts_{ref}^{n+m})^* = (n+m)*BP + t_p \quad (10)$$

Apply (1) at t_i^n and t_i^{n+l+3} we get:

$$\begin{aligned} c_i(t_i^n) &= k^n * t_i^n + b^n = ts_{ref}^n + D_i^- \\ &= n * BP + d_n + t_p + D_i^- \quad (11) \\ c_i(t_i^{n+l+3}) &= k^n * t_i^{n+l+3} + b^n = ts_{ref}^{n+l+3} + D_i^+ \\ &= (n+l+3) * BP + d_{n+l+3} + t_p + D_i^+ \quad (12) \end{aligned}$$

By the linearity of the clock we have:

$$\frac{(t_i^{n+m})^* - t_i^n}{(ts_{ref}^{n+m})^* - ts_{ref}^n} = \frac{(t_i^{n+m})^* - t_i^{n+l+3}}{(ts_{ref}^{n+m})^* - ts_{ref}^{n+l+3}} \quad (13)$$

Combining (10), (11), (12) and (13), we get $\frac{D_i^+}{D_i^-} = \frac{(m-l-3)*BP - d_{n+l+3}}{m*BP - d_n}$. Note that $d_n, d_{n+l+3} \ll BP$, we get $\frac{D_i^+}{D_i^-} = \frac{m-l-3}{m} + o(1)$. We can see from the proof that the optimal value of m in terms of the performance when the reference node changes is $l+3$ in that the adjusted clock of each node is expected to converge to the same time when a new round of synchronization begins. Even in the worst case where $m = 1$, D_i^+ can be bounded by $(l+2) * D_i^-$.

It is further easy to prove that the synchronization error after the change of the reference node is bounded by $|\frac{m-l-3}{m}| * syn\_err + 2*\epsilon$ where syn_err is the synchronization error before the reference node changes. Combining Lemma 1 and Lemma 2, we prove that by carefully choosing m and l, the synchronization error of SSTSP can be bounded by $2 * \epsilon$, which is normally under $20\mu s$.

In terms of traffic overhead, the number of synchronization beacons emitted in SSTSP is the same as in TSF, while the size of each beacon increases from 56 bytes (24 bytes of preamble and 32 bytes of data) in TSF to 92 bytes (suppose 128-bit hash values are used) in SSTSP due to the hash values and the interval index included to secure the beacons.

Besides, each node is required to store its own hash chain. It can either create the hash chain all at once and store all the elements or only store the last element and compute the new element on demand. [6] proposes a hybrid storage efficient mechanism to reduce storage with a small recomputation penalty: a one-way hash chain with n elements only requires $log_2(n)$ storage and $log_2(n)$ computation to access an element. Each node is also required to buffer temporarily the synchronization beacons received during last 2 BPs. In most cases $300-500$ bytes of memory can meet the requirement. We argue that the storage requirement as well as the increase in the beacon size is reasonable considering the gain in performance and security that SSTSP achieves.

In this paper, we focus on detecting malicious attacks and preventing the network from being desynchronized by malicious nodes using erroneous time values. However, we do not provide recovery mechanisms when an attack is detected but simply discard the erroneous beacon. Possible solutions includes restarting the synchronization procedure or sending an alert and eliminating the attackers from the network. We leave it for our future work.

4 Security Analysis

The attacks to the synchronization protocols can be classified as external attacks and internal attacks based on the information the attackers have. External attacks are launched by external attackers who do not have the cryptographic credentials (authenticated hash chain in SSTSP) that are necessary to participate in the synchronization procedure. Internal attacks are launched by internal attackers who have compromised legitimate nodes, and therefore have access to the cryptographic credentials of those nodes. Obviously internal attacks are far more difficult to detect and sometimes cannot be countered by pure cryptographic primitives.

Attackers may attack the synchronization protocols by forging or modifying synchronization beacons. SSTSP uses

μTESLA to ensure the integrity and authenticity of the synchronization beacons. This prevents the external attackers from modifying or forging the synchronization beacons or impersonating the reference node. A more serious case is when an internal attacker becomes the reference node. In this case, the guard time check serves as a defense line to decrease the effectiveness of the attacks such that the attacker can only forge timestamps whose difference with the receiver's local time is within the guard time, otherwise the beacons containing incorrect timestamps are rejected. We argue that the impact of this attack is limited in that all nodes are synchronized to a virtual clock that may be slightly different to the real clock of the reference node. However, the internal attacker cannot desynchronize the network.

Attackers may replay the out-of-date synchronization beacons to deliberately magnify the offset of the time declared in the replayed message and actual time. As a more delicate version of replay attacks, an attacker may firstly jam the channel between the reference node and the victim node A, then delay the synchronization beacons from the reference node and relay it to A later to make A incorrectly estimate the time of the reference node (This attack is referred to as pulse-delay attack in [8]). These attacks can be countered in our approach by carefully choosing the guard time parameter.

Besides the efforts to violate the proper behavior of the synchronization procedure, attackers can generate a massive amount of messages to jam the wireless channel, impeding the traffic including the transmission of time synchronization beacons. Jamming attacks are beyond the scope of our discussion. Actually under jamming attacks all communications in the network are disabled.

5 Performance Analysis

We further evaluate the performance of SSTSP by simulation. We set the relative clock frequency with respect to real time uniformly distributed in the range of $[1 - 0.01\%, 1 + 0.01\%]$. The delay caused by hash operations is ignored since they can be performed in the on-the-fly way. We run the simulation for 1000s for OFDM system with bitrate of 54Mbps: $w = 30$, $BP = 0.1s$, $l = 1$, the number of nodes $N = 100 - 500$ and the beacon length is 4 slot time in TSP and 7 slot time in SSTSP. We also set the packet error rate to be 0.01%. We let 5% of the stations leave at BP $k * 200$s ($k > 1$). They return after 50s. In order to simulate the impact of changing the reference node, we let the reference node leave at 300s, 500s and 800s.

Fig.1 shows the maximum clock drift of TSF in the network of 100 and 300 nodes. We can see the scalability problem due to the *fastest node asynchronization* and the *beacon collision* problem discussed in Sec.2.

Fig.2 and shows the maximum clock drift of SSTSP in

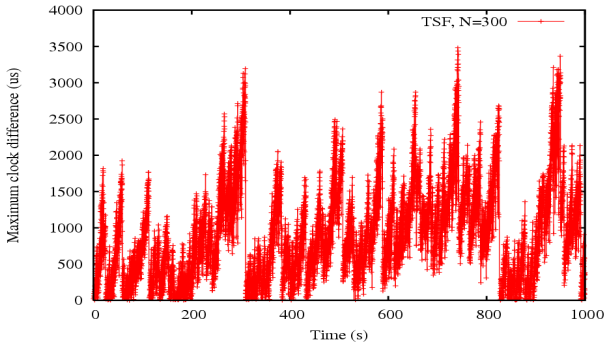

Figure 1. Maximum clock difference: TSF, 300 nodes

the network of 500 nodes with m=4. We can see that SSTSP significantly outperforms TSF by achieving a very precise synchronization with the maximum clock difference below $10\mu s$ after the protocol stabilizes, which is among the best results of currently proposed solutions (see [10], [9] for their detailed results). We then run SSTSP with different m. Tab.1 studies the maximum clock difference and the synchronization latency. We set an initial clock offset in $(-112\mu s,\ 112\mu s)$ to each node. We adopt the industrial expectation that the maximum clock drift should be under $25\mu s$ for an IBBS of any size. We consider the network to be synchronized when the maximum clock difference between any two nodes is under $25\mu s$. We can see that setting $m = 2$ or 3 achieves the best tradeoff between synchronization accuracy and latency.

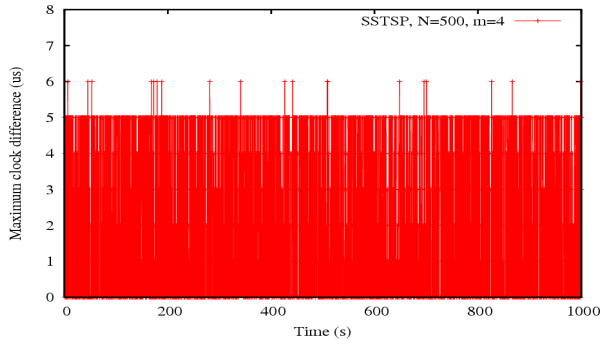

Figure 2. Maximum clock difference: SSTSP, 500 nodes, m = 4

We also simulate TSF and SSTSP in a hostile environment where an attacker attacks the synchronization protocols during $400s$ to $600s$. The attacker attacks by deliberately sending the synchronization beacons at each BP without delay with an erroneous time value slower than its local clock. We carefully configure the erroneous time values such that they can pass the guard time check in SSTSP.

m	Synchronization latency	Synchronization error
1	$0.1s$	$12\mu s$
2	$0.4s$	$7\mu s$
3	$0.6s$	$6\mu s$
4	$0.8s$	$6\mu s$
5	$1.1s$	$6\mu s$

Table 1. Maximum clock difference & synchronization latency Vs m

Fig.3 and Fig.4 show the synchronization error of TSF and SSTSP under the above attack. The synchronization error of TSF uprises to $20000\mu s$ during the attack. The attacker always wins the contentions thus disabling the fast nodes from emitting beacons. Other protocols improving TSF are also vulnerable to the attack because they depend on the fast nodes to spread the timing information. However, with SSTSP the attacker cannot desynchronize the network even though it manages to become the reference.

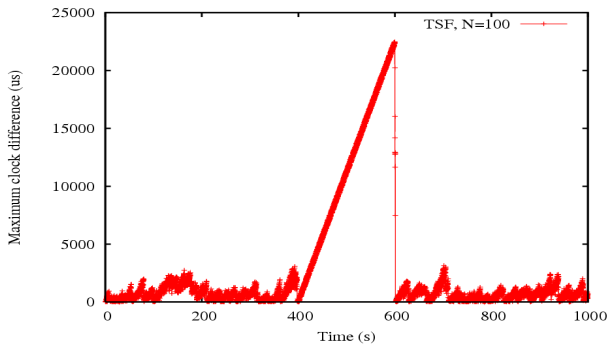

Figure 3. Maximum clock difference: TSF, 100 nodes, an attacker

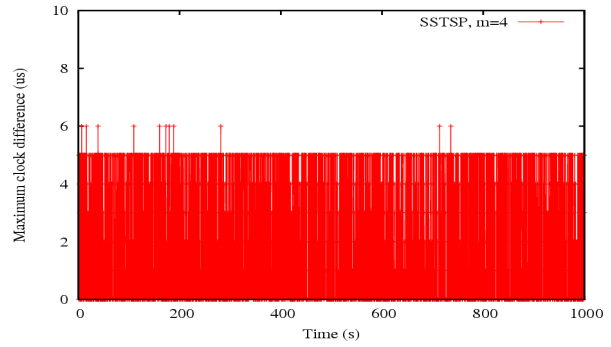

Figure 4. Maximum clock difference: SSTSP, 500 nodes, an attacker

6 Conclusion

In this paper, we address the scalability and the security problems of time synchronization protocols for IEEE 802.11 ad hoc networks. We propose SSTSP, a scalable and secure time synchronization that significantly improves the performance of TSF. We base our approach on one-way Hash chain, a lightweight mechanism to ensure the authenticity and the integrity of the synchronization beacons. With SSTSP, the networks can be synchronized with the maximum error under $20\mu s$ without any uncontinuous leaps in clocks, which is among the best results of currently proposed solutions. Meanwhile, SSTSP can maintain the network synchronized even in hostile environment. Our further work includes extending SSTSP to multi-hop ad hoc networks and proposing a recovery mechanism when a malicious attack is detected.

References

[1] Carlos H. Rentel, Thomas Kunz. "Network Synchronization in Wireless Ad Hoc Networks". Technical Report SCE-04-08, July 2004, Carleton University, Systems and Computer Engineering.

[2] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J. D. Tygar. SPINS: Security Protocols for Sensor Networks. Mobile Computing and Networking. Rome, Italy, 2001.

[3] ANSI/IEEE Std 802.11, 1999 Edition.

[4] T. Lai and D. Zhou. "Efficient and Scalable IEEE 802.11 Ad-Hoc-Mode Timing Synchronization Function". Proceedings of the IEEE 17th International Conference on Advanced Information Networking and Applications, Xi'an, China, March 2003.

[5] J. So, N. H. Vaidya. "A Distributed SelfStabilizing Time Synchronization Protocol for Multi-hop Wireless Networks". Technical Report, UIUC, January 2004.

[6] M. Jakobsson. "Fractal hash sequence representation and traversal". In IEEE Symposium on Information Theory, 2002.

[7] H. Song, S. Zhu, and G. Cao. "Attack-Resilient Time Synchronization for Wireless Sensor Networks". 2nd IEEE Int'l Conf. on Mobile Ad-hoc and Sensor Systems (MASS05), USA, November 2005.

[8] S. Ganeriwal, S. Capkun, C. Han, M. Srivastava. "Secure Time Synchronization Service for Sensor Networks". WiSE, 2005.

[9] J. Sheu, C. Chao, C. Sun. "A Clock Synchronization Algorithm for Multi-Hop Wireless Ad Hoc Networks". ICDCS 2004.

[10] D. Zhou, T. Lai. "A Compatible and Scalable Clock Synchronization Protocol in IEEE 802.11 Ad Hoc Networks". ICPP 2005.

[11] Y.-C. Hu, A. Perrig, and D.B. Johnson. "Ariadne: A Secure On-Demand Routing Protocol for Ad hoc Networks". Proc. 8th ACM Conf. Mobile Computing and Networking (Mobicom02), USA, 2002.

[12] F. Stajano and R. Anderson. "The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks". In Security Protocols, 7th International Workshop, edited by B. Christianson, B. Crispo, and M. Roe. Springer Verlag Berlin Heidelberg, 1999.