A Scalable and Adaptive Clock Synchronization Protocol for IEEE 802.11-Based Multihop Ad Hoc Networks

Dong Zhou Ten H. Lai
Department of Computer Science and Engineering
The Ohio State University
{zhoudo, lai}@cse.ohio-state.edu

Abstract

This paper studies the fundamental problem of clock synchronization in IEEE 802.11-based multihop ad hoc networks. Clock synchronization is important for power saving, network throughput and many basic operations of 802.11 protocols in a multihop ad hoc network (MANET). The scalability problem of 802.11 timing synchronization has been studied extensively in single hop ad hoc networks and good solutions are available. However these solutions do not perform well in the MANET environment. A few multihop solutions were proposed; but the performance is still not very good. The maximum clock offset is still over 200 μs for these protocols. In this paper, we propose an adaptive protocol through beacon transmission prioritization, frequency adjustment and construction of dominating set. The frequency adjustment is proved to be bounded. Simulation shows that we are able to control the maximum clock offset under 50μs after protocol stabilization. The improvement is more than 400% over the current solutions with similar complexity. The new protocol also shows great long-term stability.
Keywords: IEEE 802.11, ad hoc networks, scalability, multihop, clock synchronization

1 Introduction

The IEEE 802.11 standards support the peer-to-peer mode Independent Basic Service Set (IBSS), which is an ad hoc network with all its stations within one another's transmission range. 802.11-based MANET has been proposed to extend the coverage of the network. Clock synchronization is important for frequency hopping, power management and many basic operations of 802.11 networks.

IEEE 802.11 standard [1] specifies a Timing Synchronization Function (TSF) for the ad hoc mode. 802.11 TSF in a single hop ad hoc network is not scalable [2]. The maximum clock offset between stations can be over 4000 μs for a large IBSS running the 802.11 TSF. Several protocols have been proposed during the past few years to resolve the scalability issue in a single hop ad hoc network[3, 4]. TATSF was proposed in [3]. It classifies the stations into multiple tiers with different beacon contention frequencies for stations in each of the tiers. The maximum clock offset is contained under 125 μs. Another solution called SATSF was proposed in [4]. The maximum clock offset can be controlled under 25μs. SATSF allows the fastest station(s) to compete every beacon period and inhibit slower stations from beacon contention. It achieves very accurate clock synchronization through a bounded frequency adjustment scheme. However these solutions cannot be adopted directly for MANETs. Simply giving priority to the fastest station(s) is not enough, as it may take very long time for the fastest station's timing information to reach remote stations. The accuracy of these types of TSFs will suffer in multihop networks. Recently a few clock synchronization protocols for MANETs have been proposed[5, 6]. But the maximum clock offset is over 200 μs after protocol stabilization.

In this paper, we propose a new clock synchronization protocol in a MANET called MATSF by implementing three strategies. Beacon transmission prioritization ensures good scalability. Bounded frequency adjustment helps MATSF to achieve good accuracy and long-term stability. The construction of dominating set allows our protocol to converge faster than existing protocols. For MATSF, the maximum clock offset can be controlled within 50 μs. The improvement is more than 400% over the current solutions of similar complexity. The protocol converges in less than 10 seconds. It converges several times faster than current solutions.

The rest of the paper is organized as follows. Section 2 reviews 802.11 TSF and related works. Section 3 proposes a new protocol to address the scalability issue with all the desirable requirements. We also analyze the protocol to show why it outperforms other protocols. Simulation results are discussed in section 4. Section 5 concludes the paper.

2 Overview and Related Works

In 802.11 TSF, each station maintains a TSF timer counting in increments of microseconds (μs). All stations in the IBSS compete for beacon transmission every aBeaconPeriod second (BP). At the beginning of each BP, there is a *beacon generation window* consisting of $W+1$ slots. Each station calculates a random delay uniformly distributed in $[0, W]$ and is scheduled to transmit a beacon when the delay timer expires. If a beacon arrives before the random delay timer has expired, the station cancels the pending beacon transmission and the remaining random delay. Upon receiving a beacon, a station sets its TSF timer to the timestamp of the beacon if the value of the timestamp is later than the station's

TSF timer.

Several protocols for clock synchronization in a MANET have been proposed recently. A time synchronism algorithm is proposed in [7] to deal with the partitioning problem in sparse ad hoc networks. RBS is presented in [8]. A reference broadcast does not contain an explicit timestamp; instead, its arrival time is used by the receivers as a point of reference for comparing their clocks. RBS uses nontrivial statistical methods such as regression to estimate the clock phase offset and clock frequency offset of any two stations. These protocols are interesting, but they cannot be easily implemented under the 802.11 TSF framework.

A protocol called ASP is proposed in [5] for time synchronization in 802.11-based multihop ad hoc networks. The basic idea of the protocol is to adjust clocks' frequencies. But it does not address the scalability problem fully. The maximum clock offset is still over $200\mu s$. The reason is that ASP trusts the face value of the timestamp of the beacon from the sending station. The result is that the slower station may over-adjust its clock frequency and becomes faster than the original fastest station. This process may keep repeating that the frequencies of the clocks get faster and faster and eventually out of bound (i.e. beyond the 802.11 specified $\pm 0.01\%$).

Another protocol called MTSF is proposed in [6] for 802.11 MANETs. The basic idea is to create a spanning tree rooted at the fastest station. Each station selects the fastest neighbor as its "parent". Once the tree is constructed, the station schedules its beacon transmission in a way so that it competes in a beacon period only if its parent does not compete. A problem with this solution is that its beacon transmission schedule relies heavily on the spanning tree. When the fastest station moves to another location, the tree needs to be reconstructed and the schedule will be messed up in the process. Another issue with the protocol is that the maximum clock offset is still quite large (over 200 μs) after stabilization.

3 Our Solution: MATSF

In this section, we will propose a multihop adaptive timing synchronization function called MATSF. First let us briefly analyze the reasons behind the poor scalability of 802.11 TSF, and then we will show how MATSF resolves these issues.

3.1 Root Cause of Scalability Problem

- Beacon collision: Large number of stations contend for beacon generation within a small contention window.

- Diverse clock frequencies: The IEEE 802.11 specifications only require clock accuracy to be within $\pm 0.01\%$. There are several elements that contribute to the inaccuracy of clocks [9]: starting frequency offset (due to frequency calibration error), frequency drift (due to aging) and oscillator phase noise/offset/drift. Given the typical operating environment and short life span of an 802.11 MANET, frequency drift and oscillator phase errors are negligible compared to the starting frequency offset.

- Inaccurate timestamp: It takes time for a beacon to travel from the sender's MAC layer (which sets the value of the timestamp on the beacon) to the receiver's MAC layer (which uses the timestamp). Thus, when the receiver synchronizes its clock according to the timestamp's value, the latter is already different from the sending TSF timer's actual value as of that moment. To mitigate this problem, according to the 802.11 standard, a station sending a beacon shall set the value of the beacon's timestamp so that it equals the value of the station's TSF timer at the time that the first bit of the timestamp is transmitted to the PHY plus the transmitting station's delays through its local PHY from the MAC-PHY interface to its interface with the wireless medium. This mechanism maintains the value of the beacon and the value of the sending TSF timer to be within 4 μs plus the maximum propagation delay of the PHY, which adds up to about $\pm 5\mu$s.

3.2 Overview of our Protocol

Before presenting our protocol, we list the desirable requirement for a good MANET TSF first. In our opinion, an acceptable solution to the 802.11 TSF's scalability problem must be accurate, low cost, scalable, and able to handle mobility.

Given the reasons of asynchronism (as discussed in Section 3.1), three strategies suggest themselves as natural approaches to improving the 802.11 TSF's scalability. The first strategy is to reduce the number of stations competing for beacon generation. The second strategy is to adjust clock frequencies so that all clocks in the MANET have about the same accuracy. The third strategy is to establish a path for each station to reach the fastest station(s) quickly through beacon exchange. This can be achieved through the construction of a dominating set.

Beacon transmission prioritization

To implement strategy 1, we classify the stations into two tiers according to their frequencies. It is hard and unnecessary to obtain a precise ranking; an estimate is sufficient for our purpose. Each station divides the time line into *observation periods*, each period consisting of OBS BPs, where OBS is a pre-specified integer. In each observation period, each station i computes the number of *different* stations from which i has received at least one faster beacon. We declare a beacon faster only if the timestamp minus the maximum estimation error (5 μs) is larger than i's current TSF timer value. We record this number as $ahead(i)$. If $ahead(i) < THD$, then station i competes in every beacon period.

Bounded frequency adjustment

To implement strategy 2, we need the following definitions.

Definition 1 Clock i's *native frequency*, denoted as $f_n(i)$, is the number of ticks the clock accumulates per second driven by its oscillator.

Note that each 802.11 timer has a native frequency within the range of $(1 \pm 0.01\%)$ mega hertz (MHz).

(Each timer ticks once per microsecond, but there is an inaccuracy of up to ± 0.01%.)

Definition 2 Clock i's *frequency adjustment*, denoted as $f_a(i)$, is the number of extra ticks per second that is added to the clock.

Definition 3 The sum $adj\_freq(i) = f_n(i) + f_a(i)$ is the *adjusted frequency* of clock i.

For each station i, we will calculate a frequency adjustment $f_a(i)$, such that $f_n(i)+f_a(i) \approx f_n(fastest\ station)$. The basic idea behind the calculation of frequency adjustment is as follows. Suppose station i receives two beacons from station j, say, at times t_1 and t_2, respectively. From the timestamps on the beacons, station j's TSF timer values as of time t_1 and t_2 can be estimated — let them be $TS^1(j)$ and $TS^2(j)$, respectively. If it can be ensured that j's timer had never been synchronized between the times it sent the two beacons. Then from the values $TS^1(j)$ and $TS^2(j)$, station j's clock frequency can be estimated. If this estimated frequency is higher than station i's own clock frequency, then i' adjusts its frequency to match j's. This way, all stations can eventually have their adjusted frequency matching the fastest station's clock frequency.

There are two design issues in the above frequency adjustment mechanism. First, how to ensure that j's clock had not been synchronized between the times it sent the two beacons? It can be resolved by attaching a sequence number field at the end of the beacon frame. The sequence number is incremented each time a station is synchronized by another station. If the sequence numbers are the same, then the station is not synchronized by other stations. The second issue is that when station i adjusts its clock frequency to match j's, it is important to ensure that i does not overdo it. That is, i's adjusted frequency must not be higher than j's frequency.

We overcome this problem by taking into consideration the inaccuracy of timestamps on beacons. According to the 802.11 specifications, the timing information carried by the timestamp may be inaccurate by as much as $\pm 5\mu$s. Thus, denoting by $T^1(j)$ and $T^2(j)$ station j's actual timer values as of times t_1 and t_2, respectively, where t_1 and t_2, as before, are the times when j's two consecutive beacons arrive at a station, we have $|T^1(j) - TS^1(j)| \leq 5$ μs and $|T^2(j) - TS^2(j)| \leq 5$ μs. To avoid over-adjusting a station's frequency, we will estimate station j's frequency conservatively by

$$\frac{(TS^2(j) - \delta) - (TS^1(j) + \delta)}{t_2 - t_1} \leq \frac{T^2(j) - T^1(j)}{t_2 - t_1} = adj\_freq(j) \quad (1)$$

where $\delta = 5\mu$s. By adjusting i's clock frequency to this conservatively estimated frequency of j, we guarantee the i will never over-adjust its frequency.

Construction of dominating set

To implement strategy 3, we try to construct a dominating set for the stations in the MANET. We put the fastest station(s) into the dominating set, we also put gateway stations and selected bridge stations (to be defined later) into the dominating set and let them compete for beacon transmission every BP; while the rest of the stations compete only once in a while. The timing information of the fastest node will be propagated through the MANET very quickly along a path of dominating nodes. Collision is reduced due to the fact that nodes not in the dominating set compete for beacon transmission less frequently.

Represent the MANET as an undirected graph, $G = (V, E)$. Let DS be any subset of V. Relative to this set, a node u in V is defined to be inDS, covered (by DS), uncovered (by DS), a bridge, or a gateway as follows:

- inDS: if $u \in DS$;
- uncovered: if $u \notin DS$ and there is no edge joining u to any node in DS;
- covered: if $u \notin DS$ and there is an edge $(u, v) \in E$ for some $v \in DS$;
- bridge: if u is covered and has at least one uncovered neighbor;
- gateway: if there are two connected components in $G(DS)$, say, C_1 and C_2, such that $hop(u, C_1) + hop(u, C_2) \leq 3$, where $G(DS)$ denotes the subgraph induced by DS, and $hop(u, C_1)$ denotes the (minimum) number of hops between u and C_1.

We try to construct a dominating set starting with one node in the DS (for example, the initiator of the MANET), and adding bridge/gateway nodes into the DS. The following lemmas will guide our protocol design.

Lemma 1 *DS becomes a dominating set whenever there is no bridge in G.*

Proof. It follows directly from the definition of dominating set.

Lemma 2 *A dominating set in G is connected iff there is no gateway in G.*

Proof. ($\Rightarrow$) If DS is connected, $G(DS)$ has only one connected component and, therefore, no node can be a gateway.

($\Leftarrow$) Now suppose DS is a disconnected dominating set. By definition, There are no uncovered nodes in G. Let S_1 and S_2 be two "nearest" connected components of DS; i.e., the hop number between S_1 and S_2 is smallest among all pairs of connected components. The hop number between S_1 and S_2 is either two or three. (For if the hop number is less than two, S_1 and S_2 would be connected; if it is more than three, there would be uncovered nodes.) The one or two nodes on the shortest path between S_1 and S_2 are gateways. ■

We add to the beacon frame a 3-bit field, called *status*, for the sender to indicate its present status.

Each station, i, continuously observes the beacons received in the past OBS beacon periods. #inDS_bcn(i) and #uncovered_bcn(i) are the number of inDS beacons and the number of uncovered beacons, respectively, received by i in the past OBS beacon periods. At the end of

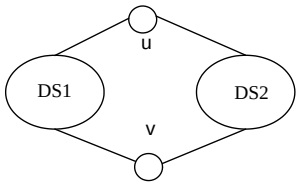

Figure 1: It is difficult to recognize gateways.

each beacon interval, If $status(i)$ is not inDS or *gateway*, it updates its status as follows:

$$\text{status}(i) := \begin{cases} \text{uncovered} & \text{if } (\#\text{inDS\_bcn}(i) = 0) \\ \text{covered} & \text{if } (\#\text{inDS\_bcn}(i) > 0) \text{ and } (\#\text{uncovered\_bcn}(i) = 0) \\ \text{bridge} & \text{if } (\#\text{inDS\_bcn}(i) > 0) \text{ and } (\#\text{uncovered\_bcn}(i) > 0) \end{cases} \quad (2)$$

To help speed up the process of recognizing bridges, an uncovered node competes for beacon transmission every interval. To avoid adding too many bridge nodes into the DS, the following equation is used to calculate when a bridge station enters DS (after $bridge\_T$ BPs). $bridge\_T$ is recalculated whenever the number of uncovered neighbors changes.

$$bridge\_T(i) = \frac{T_{max}}{\text{number of uncovered neighbors of } i} \quad (3)$$

Gateways are considerably harder to recognize. For example, consider the situation as illustrated in Fig. 1, where suppose DS_1 and DS_2 are two disjoint sets of inDS nodes, node u is not inDS, and node v may or may not be inDS. By definition, u is a gateway iff v is not inDS. A challenging question is, without knowing v's status, can u determine its own status? We develop a heuristic for a gateway to recognize its gateway status.

Recall that in our timing synchronization protocol, stations in DS generate beacons far more frequently than those not in DS (except for uncovered nodes). Thus, in Fig. 1, if v is in DS, it is supposed to transmit beacons frequently enough to synchronize DS_1 with DS_2. On the other hand, if v is not in DS, it is possible that DS_1 and DS_2 be getting out of synchronization. We use this difference as a heuristic for u to determine whether it is a gateway. Thus, we adopts the following heuristic.

A covered station regards itself as a gateway if it receives a beacon with a timing "considerably" earlier than the station's own timer. (A practical definition of "considerably earlier" which we used in our simulation is, timing t_1 is said to be *considerably* earlier than t_2 if $t_1 < t_2 - 80\%\Delta$, where Δ is the maximum time difference between clocks that one wishes to achieve. $\Delta = 50\mu$s in our simulation.) Our heuristic may not recognize every gateway node. For example, if we have a graph with 3 nodes in a line. The middle node has the slowest frequency. The node to the left and the node to the right have the same fast frequencies. We will not be able to detect the middle node as the gateway. However, it will not impact the accuracy of our timing synchronization protocol. We fail to detect some gateway nodes only if their clocks are already synchronized very accurately.

3.3 A Scalable Solution for Clock Synchronization: MATSF

Now we are ready to introduce a new scheme by integrating the three strategies to meet all the requirements of clock synchronization for an 802.11 MANET.

We first define the terms and variables to be used the protocol, and then describe the protocol MATSF itself.

- $T(i)$: TSF timer value of station i.
- $NT(i)$: native timer value of station i based on the oscillator.
- $SK(i)$: clock frequency adjustment for station i. The amount $SK(i) * 10^6$ is an estimate of $f_a(i)$, and will be added to $T(i)$ per second.
- $TS(j)$: the adjusted timestamp value obtained from the beacon sent by j and received by i. It is the sum of the timestamp value in the beacon plus the receiving PHY layer delay and the time since the first bit is received at the receiver's PHY/MAC interface.
- $seq(j)$: the sequence number obtained from the beacon sent by j and received by i.
- $ahead(i)$: number of stations from which faster beacons have been received by i during an observation period.
- $\pm\delta$: the maximum error that may occur when we estimate the value of $T(j)$ by $TS(j)$. That is, $|T(j) - TS(j)| \leq \delta$ μs at the time when j's beacon arrives at the receiving station's MAC layer.
- $\Delta NT(i)(j)$: the amount of time (i's native time) since the last time station i received a beacon from station j with a faster timestamp.
- $last\_NT(i)(j)$: the native timer value of station i the last time it received a beacon from station j with a faster timestamp.
- $last\_sync\_T(i)(j)$: the timer value of station i the last time it was synchronized by a faster beacon from station j.
- $last\_seq(i)(j)$: the sequence number station of j the last time when station i received a beacon from station j with a faster timestamp.
- $BC(i)$: station i competes for beacon generation once every $BC(i)$ beacon periods.
- $C(i)$: counter of beacon periods.
- $status(i)$: 3-bit addition to the beacon representing the dominating set status. The value can be inDS, covered, uncovered, a bridge, or a gateway.
- $bridge\_T(i)$: number of BPs for a bridge node to wait before entering DS.

Now we present MATSF in Fig. 2. In this algorithm, we update three variables under the proper conditions to meet our design requirements. We change the TSF timer value of the receiving station i ($T(i)$) when we are sure the clock time difference is out of the margin of

MATSF: Multihop Adaptive Timing Synchronization Function

1. Initially, when station i joins the MANET,
   if station i is the initiator of the MANET, let
   $NT(i) \leftarrow T(i) \leftarrow 0$, $status(i) \leftarrow inDS$, $BC(i) \leftarrow 1$
   else if i joins in by hearing a beacon from station j,
   $NT(i) \leftarrow T(i) \leftarrow TS(j) - \delta$,
   if $status(j) = inDS$, then
   $status(i) \leftarrow covered$, $BC(i) \leftarrow I_{max}$
   else $status(i) \leftarrow uncovered$, $BC(i) \leftarrow 1$.
   In both cases, let $SK(i) \leftarrow 0$, $C(i) \leftarrow rand(1, BC(i))$, and start an observation period of OBS.
2. If $C(i) < BC(i)$, station i will not compete for beacon transmission.
3. If $C(i) \geq BC(i)$, $C(i) \leftarrow 0$, station i is ready to compete.
4. At each TBTT each station that wants to compete for beacon transmission calculates a random delay uniformly distributed in the range $[0, w]$.
5. The station waits for the period of the random delay.
6. When the random delay timer expires, if the medium is idle, the station transmits a beacon with a timestamp; else if it receives a beacon before its random delay timer expiration, it cancels the pending transmission and the remaining random delay.
7. Upon receiving a beacon from station j, a station i performs:
   if $status(j) = inDS$ and $status(i) = uncovered$
   $status(i) = covered$.
   if $status(j) = uncovered$ and $status(i) = covered$
   $status(i) = bridge$, calculate $bridge\_T(i)$.
   if $status(i) = covered$ and $T(i) > T(j) + \delta + 0.8\Delta$
   $status(i) = gateway$.
   if $T(i) \leq TS(j) - \delta$ then
   $seq(i) \leftarrow seq(i) + 1$
   $T(i) \leftarrow TS(j) - \delta$
   $\Delta NT(i)(j) \leftarrow NT(i) - last\_NT(i)(j)$
   $tmp \leftarrow TS(j) - last\_sync\_T(i)(j) - \Delta NT(i)(j) - 3\delta$
   if $seq(j) = last\_seq(i)(j)$ and $tmp \geq 0$
   $SK(i) \leftarrow \max\{SK(i), tmp/\Delta NT(i)(j)\}$
   $last\_NT(i)(j) \leftarrow NT(i)$
   $last\_sync\_T(i)(j) \leftarrow TS(j) - \delta$
   $last\_seq(i)(j) \leftarrow seq(j)$
8. If $SK(i) > 0$, station i increments $T(i)$ by 1 every $\lceil 1/SK(i) \rceil$ μs (in terms of station i's native time).
9. At the end of a beacon period, $C(i) \leftarrow C(i) + 1$,
   if $bridge\_T(i) > 0$, $bridge\_T(i) \leftarrow bridge\_T(i) - 1$.
   update $status(i)$ using Eq. 2.
   if $status(i) = bridge$ & $bridge\_T(i) = 0$
   $status(i) \leftarrow inDS$.
   if $status(i) = gateway$, $status(i) \leftarrow inDS$.
   if $status(i) = inDS$ or $status(i) = uncovered$
   $BC(i) \leftarrow 1$
   else $BC(i) \leftarrow I_{max}$, $C(i) \leftarrow rand(1, BC(i))$.
10. At the end of observation period:
    if $ahead(i) < THD$, then $status(i) \leftarrow inDS$.
    Reset $ahead(i)$, #inDS_bcn(i), #uncovered_bcn(i).
11. $NT(i)$ increments by 1 whenever station i's oscillator ticks.

Figure 2: Algorithm MATSF

error. We modify the BC value to control how often a station competes for beacon transmission. We reset frequency adjustment (SK) only when we are sure the sender's clock frequency is faster.

When station i receives a beacon from station j, it first checks whether $TS(j) - \delta \geq T(i)$. If that is true, i knows that station j's clock value is definitely larger than its own timer and i will adjust its TSF timer value. To make sure that j's clock was not reset between these two beacons, the protocol further checks whether the sequence number in the beacon from j is the same as the sequence number recorded when the last time station i received a faster beacon from j. If $TS(j) - last\_sync\_T(i)(j) - \Delta NT(i)(j) \geq 3\delta$, we are sure station j advances faster than station i's native clock (we will prove it in Theorem 1), then $SK(i)$ is re-calculated.

$BC(i)$ is used to decide how often a station competes for beacon transmission. A station competes for beacon transmission every BC BPs. $BC(i)$ is 1 for $inDS$ and $uncovered$ stations; $BC(i)$ is set to I_{max} for other stations.

3.4 Analysis of the protocol

We will first prove that our frequency adjustment is bounded, and then analyze how quickly a station can recognize its status. The latter analysis will give us some guidance on how to set some parameters used in MATSF.

3.4.1 Bounded frequency adjustment

Suppose that station i synchronizes its timer, $T(i)$, on receiving another station j's beacon. We show in the following lemma that at the moment right after the synchronization (i.e., when $T(i)$ is updated to $TS(j) - \delta$ in step 7 of the algorithm), the values of the two stations' timers are related by $T(i) \leq T(j) \leq T(i) + 2\delta$.

Lemma 3 *If station i synchronizes its clock with station j, then at the moment right after the synchronization, it holds that $T(i) \leq T(j) \leq T(i) + 2\delta$.*

Proof. At the moment of synchronization, it is known by assumption (or by the definition of δ) that

$$|T(j) - TS(j)| \leq \delta. \quad (4)$$

During synchronization, station i sets its timer as

$$T(i) := TS(j) - \delta. \quad (5)$$

The lemma follows directly from Eqs. 4 and 5. ■

Lemma 4 *For each station i, $f_a(i) = f_n(i) * SK(i)$ and $adj\_freq(i) = f_n(i) + f_n(i) * SK(i)$.*

Proof. Based on the calculation of $SK(i)$, $SK(i)$ is the adjustment per microsecond in terms of station i's native time. $f_n(i)$ is the number of native microseconds per reference time second. $f_n(i) * SK(i)$ is the number of ticks added per reference time second. Therefore, $f_a(i) = f_n(i) * SK(i)$ and, thus, $adj\_freq(i) = f_n(i) + f_n(i) * SK(i)$. ■

The following theorem shows that the adjusted frequency of each station is bounded by the frequency of the station with the highest native frequency.

Theorem 1 *For all stations i in the MANET, $adj\_freq(i)$ is non-decreasing as a function of time. Furthermore, $adj\_freq(i) \leq f_n(s_0)$ for all i, where s_0 is the station with the highest native frequency that has ever joined the MANET. (Note that s_0 may still be in the MANET or may have left).*

Proof. Since $SK(i) := \max\{SK(i), ...\}$, $SK(i)$ is obviously non-decreasing as a function of time. It then follows from Lemma 4 that $adj\_freq(i)$ is also non-decreasing.

To establish the bound $adj\_freq(k) \leq f_n(s_0)$ for all k during the life-span of the MANET, it suffices to show the following: (1) the bound holds at the MANET's inception, which is obviously true, and (2) if the bound holds up to the time when a station is about to update its SK, then after the update the bound is still valid. To prove the latter statement, consider the event that station i adjusts $SK(i)$ upon receiving a beacon from station j. As the induction hypothesis, assume $adj\_freq(k) \leq f_n(s_0)$ for all k (in particular, $adj\_freq(j) \leq f_n(s_0)$) before the $SK(i)$ update. Since $adj\_freq(j)$ does not change during the event, it is sufficient for us to show $adj\_freq(i) \leq adj\_freq(j)$ after the $SK(i)$ update.

According to the protocol, station i may adjust its SK only if i's timer has been synchronized twice by station j with the same sequence number and

$$TS(j) - last\_sync\_T(i)(j) - \Delta NT(i)(j) \geq 3\delta \quad (6)$$

Let t_1 be the time (of a reference clock) of the last timing synchronization and t_2 be that of this timing synchronization. Let $T^1(j)$ and $T^2(j)$ be the values of $T(j)$ (j's TSF timer) as of t_1 and t_2, respectively. $TS(j)$ in Eq.6 is the timestamp value at t_2. We denote it as $TS^2(j)$. Let $\Delta t = t_2 - t_1$.

During the period (t_1, t_2), station j's clock has never been synchronized by other stations. This is due to the fact that j's sequence number has not changed. Thus, over the period (t_1, t_2), timer $T(j)$ has been ticking at its adjusted frequency, $adj\_freq(j)$, and has advanced from $T^1(j)$ to $T^2(j)$. Therefore,

$$adj\_freq(j) = \frac{T^2(j) - T^1(j)}{\Delta t}. \quad (7)$$

By Lemma 3, $T^1(j) \leq last\_sync\_T(i)(j) + 2\delta$. By Eq. 4, $T^2(j) \geq TS^2(j) - \delta$. Substituting these into Eq. 7 yields

$$adj\_freq(j) \geq \frac{TS^2(j) - last\_sync\_T(i)(j) - 3\delta}{\Delta t}. \quad (8)$$

On the other hand, $adj\_freq(i)$ can be computed as follows. Station i's native clock advances $\Delta NT(i)(j)$ between time t_1 and t_2. Therefore,

$$\begin{aligned} adj\_freq(i) &= \frac{\Delta NT(i)(j) + \Delta NT(i)(j)/(1/\lceil SK(i) \rceil)}{\Delta t} \\ &\leq \frac{\Delta NT(i)(j) + \Delta NT(i)(j) * SK(i)}{\Delta t} \end{aligned} \quad (9)$$

By definition,

$$SK(i) = \frac{TS^2(j) - last\_sync\_T(i)(j) - \Delta NT(i)(j) - 3\delta}{\Delta NT(i)(j)}$$

Substituting this into Eq. 9, we obtain

$$adj\_freq(i) \leq \frac{TS^2(j) - last\_sync\_T(i)(j) - 3\delta}{\Delta t} \quad (10)$$

Eqs.8 and 10 together imply $adj\_freq(i) \leq adj\_freq(j)$. ■

3.4.2 Probability of recognizing a bridge within OBS beacon periods

Consider a bridge B. We are interested in estimating the amount of time it would take for B to recognize its bridge status.

Let M be the number of stations in the (1-hop) neighborhood of B — M_1 of them are black and M_2 are white — which are contending for beacon transmission in slot window $[0, W]$. Let $E'(m_1, m_2, w, s)$ denote the event that exactly s slots in $[0, w-2]$ are busy, that slot $w-1$ is idle, and that m_1 black and m_2 white stations remain to contend for beacon transmission (i.e., the random numbers picked by these stations are all in the range $[w-s, W]$). Denote by $p(m_1, m_2, w, s)$ the conditional probability that given $E'(m_1, m_2, w, s)$ at least one of the m_1 black stations successfully transmits a beacon.

Lemma 5 *Given M stations in the (1-hop) neighborhood of B — M_1 of them are black and M_2 are white — the probability that at least one black station successfully sends a beacon within a window is $p(M_1, M_2, 0, 0)$, which can be calculated recursively as in the proof below.*

Proof. Given $E'(m_1, m_2, w, s)$, let E be the event that at least one of the m_1 black stations successfully transmits a beacon. Event E can be partitioned into four disjoint sub-events — E_1, E_2, E_3, E_4 (as specified below) — and therefore,

$$p(m_1, m_2, w, s) = p_1 + p_2 + p_3 + p_4 \quad (11)$$

where p_i, as specified below, denotes the conditional probability of event E_i given $E'(m_1, m_2, w, s)$.

• E_1: There is no beacon transmission in slot w, but at least one successful black beacon transmission in window $[w+1, W]$. This event occurs iff all the m stations, where $m = m_1 + m_2$, picked a random number in $[w-s+1, W]$ and at least one of the black stations succeeds. Thus, the probability of its occurrence (given $E'(m_1, m_2, w, s)$) is

$$p_1 = \left(\frac{W-w+s}{W-w+s+1}\right)^m p(m_1, m_2, w+1, s) \quad (12)$$

• E_2: There is a successful black beacon transmission in slot w. This event occurs iff a unique black station chooses slot $w-s$. Therefore,

$$p_2 = m_1 \left(\frac{1}{W-w+s+1}\right)\left(\frac{W-w+s}{W-w+s+1}\right)^{m-1} \quad (13)$$

• E_3: There is a successful white beacon transmission in slot w and at least one successful black beacon transmission thereafter. It's probability can be easily calculated as

$$p_3 = m_2 \left(\frac{1}{W-w+s+1}\right)\left(\frac{W-w+s}{W-w+s+1}\right)^{m-1} p(m_1, m_2 - 1, w+b+1, s+b) \quad (14)$$

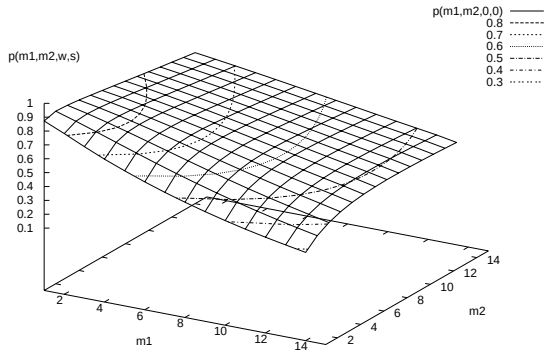

Figure 3: Probability of a successful black beacon.

where b is the beacon length in terms of slots.

• E_4: There are unsuccessful (collided) beacon transmissions in slot w, but at least one successful black beacon transmission in window $[w+b+1, W]$.

$$p_4 = \sum_{i=2}^{m} C_i^m \left(\frac{1}{W-w+s+1}\right)^i \left(\frac{W-w+s}{W-w+s+1}\right)^{m-i} \sum_{x=0}^{\min(m_1, m-i)} \frac{C_x^{m_1} C_{m-i-x}^{m_2}}{C_{m-i}^m} p(x, m-i-x, w+b+1, s+1)$$

The boundary condition is $p(m_1, m_2, w, s) = 0$ if $m_1 = 0$, $m_2 < 0$, $w > W$, or $s > W$. ■

Using the formulas developed in the proof of Lemma 5, we can calculate p_1 (resp. p_2), the probability that at least one inDS (resp. uncovered) node successfully transmits a beacon in a beacon interval.

Theorem 2 *Let $N = OBS$, the length of an observation period. With probability $p = 1 - {q_1}^N - {q_2}^N + {q_1}^N {q_2}^N$ a bridge can recognize its bridge status in a single observation period, where $q_1 = 1 - p_1$ (resp. $q_2 = 1 - p_2$) is the probability of B not even hearing an inDS (resp. uncovered) beacon in a beacon generation window.*

Proof. Let X (resp. Y) be the number of beacon generation windows until the first successful inDS (resp. uncovered) beacon transmission. Both X and Y are geometrically distributed with marginal probability functions $f_X(x) = p_1 {q_1}^{x-1}$ and $f_Y(y) = p_2 {q_2}^{y-1}$, where $q_1 = 1-p_1$ and $q_2 = 1-p_2$. Let $N = OBS$ be the length of an observation period. The probability that a bridge recognizes its status in a single observation period is

$$p = P(\max(X, Y) \le N) = 1 - {q_1}^N - {q_2}^N + {q_1}^N {q_2}^N$$

■

This theorem is useful for determining the values of *OBS* and *THD*. In our performance studies (to be presented in Section 4), we choose $N = OBS = 10$. p_1 and q_1 can be found using Fig. 3. With this value of *OBS*, the probability of recognizing a bridge in an observation period is 0.85 if there is only one inDS node and 13 uncovered nodes. The probability is 0.98 or higher when there are 2 or more inDS nodes and 12 uncovered nodes. In our simulations, the average number of BPs needed to recognize a bridge is 10.2, which amounts to 1.1 seconds.

If $THD = 2$, then the number of inDS nodes is at least 2. Combining this with $OBS = 10$, we can achieve very high probability to recognize a bridge in an observation period.

The probability to recognize a gateway is very hard to analyze. But we observed through simulation that the protocol reaches a stable state where all uncovered nodes become covered by *inDS* nodes. Therefore, it is not very critical to have an accurate analysis for gateway recognition.

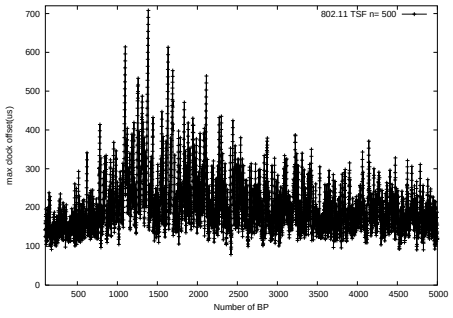

Figure 4: Maximum clock offset for 802.11 TSF

4 Simulation Study on 802.11 TSF, ASP and MATSF

In the simulation, we let the clock frequency be uniformly distributed in the range of $[-d, d]$. We pick $d = 0.01\%$. The number of slots needed to send a beacon is called beacon length. We run the simulation for OFDM system with bit-rate of 54 Mbps: $W = 30$ and beacon length $b = 4$. The MANET has n stations. We set the packet error rate to be 0.01%. We run the simulation for 5000 BPs (500 seconds). We randomly distribute these n stations in a $1000m \times 1000m$ region. Each station has a transmission range of 250 meters. We adopt the same random way-point mobility model as in [5]. The maximum speed is 5 m/s and the pause time is 50 seconds. We show the maximum clock offset starting from beacon period 100 to ignore the initial clock offset.

802.11 TSF is not scalable for MANETs. When $n = 500$, the maximum offset can be over $700\mu s$ and the average over $200\mu s$ as shown in Fig. 4.

We ran simulations for ASP with $\alpha = 3$ as it is the preferred value by [5]. Fig. 5 shows the maximum clock offset during each beacon period for ASP when $n = 500$. It shows good improvement over the 802.11 TSF but it still has large clock offset (over $300\mu s$). It does not address the scalability issue completely.

We ran the simulation for our new protocol MATSF with $\delta = 5$, $I_{max} = 16$, $\Delta = 50$, $T_{max} = 32$, $OBS = 10$ and $THD = 2$.

Fig.6 shows the maximum clock offset for a MANET with 500 stations. We can see from Fig.6 that MATSF can synchronize the clocks very precisely. The maximum clock offset is under $50\mu s$.

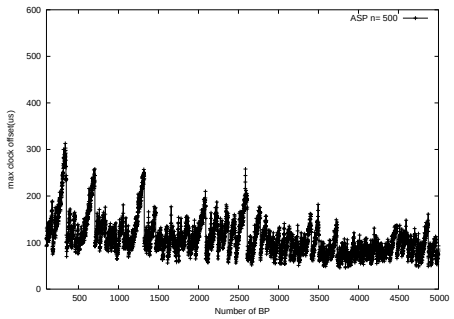

Figure 5: Maximum clock offset for ASP

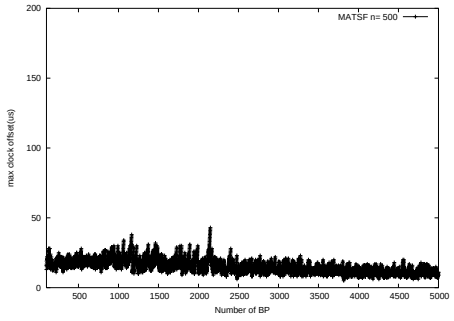

Figure 6: Maximum clock offset for MATSF

MATSF can reduce the maximum clock offset to under $50\mu s$ very quickly (under 10 seconds). ASP takes 500% more time to stabilize. We say that the TSF converges when the maximum clock offset start to stay within a close range and stop going up and down significantly (say 30 μs). Because of the construction of dominating set, the timing information can be propagated to all the stations very quickly for MATSF.

MATSF also shows nice stability after the protocol converges. The maximum offset keeps going down along the time. Fig.6 shows the trend clearly. The reason behind the improvement is that MATSF's clock frequency adjustment is bounded and each frequency adjustment will bring the frequency differences smaller.

Table 1 shows the maximum clock offset comparison between MATSF, ASP and 802.11 TSF for the number of stations (n) ranging from 100 to 500. For MATSF, The maximum clock offset is under $50\mu s$. The maximum clock offset for ASP is about 700% higher than MATSF.

Table 2 shows the average maximum clock offset comparison between MATSF, ASP and 802.11 TSF. The average maximum offset of ASP is about 400% higher than that of MATSF. The reasons for ASP's inferior performance is that the protocol may over-adjust frequencies and that the fastest stations may not form a dominating set. It may take long time for the timing information from the faster stations to reach the slower stations.

n	MATSF	ASP	802.11
100	24 μs	270 μs	277
300	24 μs	220 μs	254
500	43 μs	313 μs	708

Table 1: Maximum clock offset of MATSF, ASP and 802.11 TSF

n	MATSF	ASP	802.11
100	12 μs	59 μs	127 μs
300	12 μs	62 μs	138 μs
500	16 μs	109 μs	212 μs

Table 2: Average maximum clock offset of MATSF, ASP and 802.11 TSF

5 Conclusion

We have studied the scalability problem of clock synchronization protocols for 802.11 ad hoc networks in a multhop environment. The current solutions to the scalability problem are still short of the expectation: the maximum clock offset is over $200\mu s$. We proposed an adaptive protocol called MATSF that can achieve higher accuracy of clock synchronization than any of the current solutions. It can control the maximum clock offset under $50\mu s$. This is an over 400% improvement. We analyzed why the current solutions cannot achieve the desired accuracy. We proved that our frequency adjustment is bounded and non-decreasing. This provides a solid foundation for good long-term protocol stability. Our protocol also converges several times faster than other solutions due to the use of dominating set.

References

[1] IEEE Std 802.11. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification*, 1999 edition.

[2] L. Huang, T.H. Lai, On the Scalability of IEEE 802.11 Ad Hoc Networks, *MobiHoc 2002*, pp. 173-182.

[3] D. Zhou, T.H. Lai, Analysis and Implementation of Scalable Clock Synchronization Protocols in IEEE 802.11 Ad Hoc Networks, *MASS 2004*.

[4] Dong Zhou, Ten-Hwang Lai, A Compatible and Scalable Clock Synchronization Protocol in IEEE 802.11 Ad Hoc Networks, *ICPP 2005*.

[5] J.P. Sheu, C.M. Chao, C.W. Sun, A Clock Synchronization Algorithm for Multi-Hop Wireless Ad Hoc Networks, *ICDCS 2004*, pp. 574-581.

[6] J. So, N. Vaidya, MTSF: A Timing Synchronization Protocol to Support Synchronous Operations in Multihop Wireless Networks, UIUC technical report, 2004.

[7] K. Römer and E. Zurich, Time synchronization in ad hoc networks, *MobiHoc 2001*.

[8] J. Elson, L. Girod, and D. Estrin, Fine-Grained Network Time Synchronization using Reference Broadcasts, *2002 Symposium on Operating Systems Design and Implementation*, pp. 147-163.

[9] S. Bregni, Synchronization of Digital Telecommunications Networks, John Wiley & Sons, 2002.