Abstract—This paper presents the Clock-sampling Mutual Network Synchronization (CSMNS) algorithm. CSMNS is a distributed and autonomous network time-synchronization (NTS) approach ultimately proposed for the support of QoS-aware protocols in mobile wireless Ad Hoc networks. The fundamental principles of CSMNS are the non-hierarchical influence of the nodes in achieving the network-wise time-synchronization of the clocks, combined with the convergence and simplicity of the procedure. CSMNS makes no use of master clocks that are external or internal to the network, such as the ones located in the Global Positioning System (GPS) or any cluster of broadcasting, centralized or reference nodes. Moreover, the exchange of timing information among the nodes is performed explicitly through the periodic transmission of beacons à la IEEE 802.11. CSMNS is compatible with the IEEE 802.11 physical layer and over-the-air procedures. Numerical and analytical performance-evaluations are presented that show the potential of CSMNS as a scalable and accurate NTS approach for single-hop or multi-hop wireless networks. Our numerical results show steady-state accuracy in the order of few microseconds. This accuracy is achieved after a transition period during which the time-processes of the clocks converge to a common time-process. We compare the performance of CSMNS with that of the standardized IEEE 802.11 Timing Synchronization Function (TSF) in the Independent Basic Service Set (IBSS) mode [1]. Additionally, we evaluate some modifications for the enhancement of the basic CSMNS algorithm and the TSF. To the best of our knowledge, we also present the first numerical performance evaluation of the TSF in a multi-hop scenario.

Keywords: Network time-synchronization, wireless Ad Hoc networks.

I. INTRODUCTION

A wireless Ad Hoc network is a distributed and autonomous computer network comprised of (mobile or fixed) nodes that perform the creation, management and regular user data-communication operations over a wireless transmission medium. All, or part of the nodes, have the capability to be host and router simultaneously. A wireless Ad Hoc network is not necessarily tied to the use of a back-bone infrastructure. Advantages of the Ad Hoc network architecture include the possibility of multi-hop mode of communication, self-configuration, and the deployment flexibility gained through the lack of a pervasive infrastructure. These advantages made them initially attractive in battlefield and disaster relief scenarios. However, the Ad Hoc paradigm also suggests a better spatial re-use, and therefore better capacity and coverage performance. These are the main reasons the Ad Hoc or relaying principle is also considered an important enhancement for future wireless networks.

We are concerned with the fundamental problem of achieving network time-synchronization (NTS) in a wireless Ad Hoc network. The proposed algorithm is not tied to the specific use of the network. It is simply assumed that the nodes are trying to establish communication links among themselves.

In line with our fundamental goal, we also outline the fact that NTS can serve as one of the foundations over which mechanisms at all levels of the protocol stack can build better Quality of Service (QoS) solutions. It is particularly interesting to note that the majority of Medium Access Control (MAC) protocols that try to satisfy the QoS requirements of real-time applications (e.g., voice, streaming video) utilize a slotted-time structure (e.g., [2], [3]). However, slotted-time is usually assumed to be achievable through the atomic clocks located in the GPS satellite network. Depending on GPS signals is, in most cases, less flexible and more costly due to the need of additional hardware and clear-sky line-of-sight. The more autonomous Timing Synchronization Function (TSF) of the IEEE 802.11 standard presents itself as an alternative. However, its performance has also been questioned recently in terms of its poor scalability performance [4]. Results in this paper also show that the TSF performs poorly in a multi-hop scenario. Therefore, there is increasing need for a NTS approach that fits into the paradigm of providing QoS support for wireless Ad Hoc networks in a simple, flexible, scalable and robust manner.

The benefits of achieving NTS are unquestionable, particularly in terms of providing support for more efficient MAC, security, and network management protocols. It is, for instance, a key function in the IEEE 802.11 standard to perform power management and support of the channel hopping mechanism in the Frequency Hoping Spread Spectrum (FHSS) version of the physical (PHY) layer [1]. NTS is also a crucial feature for the coordination of packet transmissions in the more recent mesh version of the IEEE 802.16 standard. We argue that network synchronization can play a fundamental role in the support of QoS for future wireless Ad Hoc networks. However, more understanding of its advantages and associated challenges is needed.
In this paper we present Clock-sampling Mutual Network Synchronization (CSMNS). CSMNS is a distributed and autonomous NTS algorithm intended for the use of QoS-aware protocols in wireless Ad Hoc networks. The remainder of this paper is as follows: In Section II we present some related work. In Section III the TSF is discussed along with an analysis of a straightforward modification to the original TSF algorithm (the extended-TSF). In Sections IV.A, B we describe and analyze the basic-CSMNS approach along with a reduced-overhead version (CSMNS-RMN). The numerical performance evaluation of CSMNS and that of the TSF in a multi-hop scenario is presented in Section IV.C. The concluding remarks and future research directions are presented in Section V.

II. RELATED WORK

CSMNS is based on a mutual network synchronization approach of which the early work in [5] is one of the most representatives. The stability of such scheme was proved through classical control theory. Geographically separated oscillators are controlled in a distributed manner through a multiple-input phase-locked-loop (MI-PLL) co-located in every oscillator of the network. Each input of a MI-PLL collects the timing information exchanged with neighbor oscillators. Every oscillator (node) sends a train of periodic pulses that characterize its timing process; the combined received signal in every node is averaged and used in a feedback-loop to adjust the frequency of the local oscillator. All the nodes share the responsibility of network synchronization in a non-hierarchical way. One important parameter for the stability of the approach is the variability of link latency. Link latency affects the phase of the transmitted pulses and ways to estimate it might be needed. A classical mutual NTS approach, such as the one presented in [5], implies the use of special circuitry to generate short pulses that can potentially occupy wide portions of the spectrum. The mechanism also imposes tight restrictions on the Rx-Tx and Tx-Rx turn-around times of the radios if a half-duplex scheme is used, otherwise, full-duplex radios are needed to simultaneously listen and send the continuous train of pulses. The most recent work in [6] proposes a scheme for sensor networks in which a centrally located sensor initializes the NTS approach by sending pulses that are relayed by neighboring nodes in a form of flooding or broadcasting. This approach inherits the PHY layer requirements from the earlier work in [5] through the use of pulses. Moreover, it is a hierarchical approach since it depends on the central sensor or on any other sensor of the network. In particular, it is unclear how to select and discover the centrally located sensor, or how to replace it in case it fails or its battery power is drained beyond a point where it can no longer transmit pulses. CSMNS differs from these previous works in that there is no direct physical control of the clocks in every node. CSMNS does not require access to the PHY layer, nor does it require special circuitry to send continuous train of pulses that could potentially occupy a large portion of the frequency spectrum. However, as in the original work in [5], CSMNS does not depend on any particular node attributes (e.g., centrally located, fastest clock, etc.). It is a non-hierarchical approach. Furthermore, one of goals of CSMNS is to be used as a tool for QoS support protocols in wireless Ad Hoc networks. The latter is fundamentally different to the goal of NTS approaches aimed at solving the more specific multi-sensor data-fusion problem in sensor networks.

Mutual network synchronization approaches are proposed for inter-vehicle wireless networks in [7] and [8]. However, no study is made of the performance of these approaches in a multi-hop network, and the timing information is exchanged using short pulses or spreading sequences that require explicit PHY layer support. The Reference Broadcasting Synchronization (RBS) scheme is proposed in [9] with multi-hop support. RBS achieves the synchronization of multi-hop neighborhoods through the exchange of messages with an intermediate node (i.e., a node in between broadcasting neighborhoods). The purpose is to obtain a logical ordering of events rather than achieving real synchronization of the clocks in a multi-hop scenario, which is fundamentally the same approach of the work in [10] for data-fusion in sparse sensor networks. In RBS, the network is divided in broadcasting neighborhoods in which a single “reference node” sends packets that are used to achieve local synchronization (i.e., the reference node’s neighborhood). The fundamental idea is to exploit the fact that the packets sent by the reference node arrive at approximately the same time in every node of the neighborhood. After the packet is sent, the receiving nodes exchange their observations in order to establish synchronization based on the common packet reception time. RBS is also a hierarchical approach since it depends on a reference node. A hierarchical approach might be suitable for networks in which the nodes barely move relative to one another. However, it is not the most suitable approach for highly dynamic wireless Ad Hoc networks with mobility.

The lack of scalability of the IEEE 802.11 TSF is first analyzed in [4]. A method is proposed to improve the TSF based on giving higher priority for beacon transmission to the node with the fastest clock in the network. However, the study is made in a single-hop scenario, and the fundamental idea leads towards a hierarchical and non-convergent solution.

CSMNS only needs partial connectivity of the network. That is, single-hop or multi-hop communication is supported. We assume negligible link delay since the maximum separation of neighboring nodes is assumed to be of a few hundred meters (i.e., <1μsec). Different from RBS, CSMNS utilizes the same periodic beacon transmissions used in the IEEE 802.11 standard; it does not require additional over-the-air procedures.

III. 802.11 TIMING SYNCHRONIZATION FUNCTION

The TSF utilizes the clock-sampling method to exchange timing information. It is summarized next for the Independent Basic Service Set (IBSS) mode. Each node sends a beacon at a Target Beacon Transmission Time (TBTT) with period $a_{\text{beaconPeriod}}$ (e.g., 0.1sec [1]). At each TBTT each node shall:

1. Suspend the back-off timer of any pending non-beacon transmission.
2. Calculate a random delay uniformly distributed in the range between zero and $2 \cdot a_{\text{CWmin}} \cdot a_{\text{SlotTime}}$ (Table 1 shows the value of these parameters for the IEEE 802.11 standard and its different PHY layer versions).
3. Wait for the random delay before transmitting the beacon.
4. Cancel the remaining random delay and the pending beacon transmission if a beacon arrives before the random delay timer has expired.
5. Send a beacon if the random delay has expired and no beacon has arrived during the delay period.

Upon reception of a beacon, a node will adjust the received timestamp to take into account its PHY layer delay. The receiving node will set its clock to the value of the adjusted timestamp if it is later than its own. Therefore, all nodes will try to gradually synchronize to the fastest clock.

An approximate analysis of the TSF in a single-hop scenario was first attempted in [4]. The probability of sending one beacon successfully regardless of the node that sent it ($P_{\text{given}}$), and the probability of sending a beacon successfully by a given node ($P_{\text{given}}$) were found under the assumption of perfect synchronization (i.e., the beacon contention window of every node starts at the same time). The analysis proves the inefficiency of the TSF to scale. The scalability problem is blamed on the beacon collisions, which cause $P_{\text{given}}$ to decrease substantially as the number of nodes in the network increases. Figure 1 shows $P_{\text{given}}$ [4]. The parameters used in Fig. 1 correspond to the FHSS PHY layer of IEEE 802.11. A beacon transmission takes 11 slots. As seen in Fig. 1, the probability of a given node to transmit its beacon is approximately 0.05 for a network of 20 nodes. In other words, the probability of receiving a beacon from a node with the fastest clock is 0.05 in this case. The low probability of the fastest node to send a beacon successfully translates into a severe a-synchronism due to the relative time-deviations of the clocks. One tempting way to improve the TSF is to allow a node to transmit its beacon even after successfully receiving a beacon in a given contention window. However, this approach is not the most ideal since we are increasing the probability of beacon transmission at the expense of increasing the overhead. Furthermore, as the next analysis shows, the increase in probability may still prove to be insufficient in this extension to the TSF.

The extended-TSF is analyzed next by modifying the analysis in [4]. The probability of a given node to transmit its beacon in the extended-TSF ($P_{\text{given}}$) is given by

$$
P_{\text{given}}(n, W, k) = \frac{1}{W+1} \sum_{i=0}^{W} \hat{P}_{\text{given}}^k(n, W, k)
$$

(1)

Where $\hat{P}_{\text{given}}^k(n, W, k)$ is the conditional probability that the given node successfully transmits a beacon given that it is scheduled to transmit in slot $k$; $W+1$ is the contention window size (there are $W+1$ slots labeled 0 through $W$), and $n$ is the number of nodes in the network. $\hat{P}_{\text{given}}^k(n, W, k)$ is based on the same events outlined in [4] plus an additional one allowing one node to transmit even after a successful beacon reception.

### Table I. Beacon Contention Window Parameters in IEEE 802.11

<table>
<thead>
<tr>
<th>Parameter</th>
<th>FHSS</th>
<th>DSSS</th>
<th>OFDM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$aCW_{\text{min}}$ (slots)</td>
<td>15</td>
<td>31</td>
<td>15</td>
</tr>
<tr>
<td>$a\text{SlotTime}$ ($\mu$sec)</td>
<td>50</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

![Figure 1. $P_{\text{given}}$ in the IEEE 802.11 TSF, and the extended-TSF](image)

The boundary conditions in (2) are: $\hat{P}_{\text{given}}^k(0, W, k) = \hat{P}_{\text{given}}^k(n, 0, k) = (\hat{P}_{\text{given}}^k(n, W, k) \forall W < k) = 0$, and

$$
C_x^n = \frac{n!}{x!(n-x)!}.
$$

The first expression in (2) takes into account the event that no beacon transmissions occurred before slot $k$. The second expression is the event that a single successful transmission, or no transmissions occurred before slot $k$ (at slot $i$) when there are only 2 nodes in the network. The third term is the event that exactly $x$ beacon transmissions occurred in slot $i$, where $(1 \leq x \leq n-1)$, and that exactly $y$ nodes $(0 \leq y \leq n-1-x)$ are scheduled to transmit in slots $i+1$ through $i+b-1$. The nodes scheduled to transmit during
the latter interval will defer their transmissions for \( b \) consecutive slots due to the beacon transmission that started at slot \( i \) (due to carrier sensing).

Equation (1) is also plotted in Fig. 1 along with simulation points. The simulation result was obtained after 30min of real-time simulation using Matlab. As can be seen, (2) corresponds to the simulation data quite well. The extended-TSF does achieve a better probability of beacon transmission than TSF. However, the improvement is insufficient to provide a scalable solution. Therefore, although to a lesser degree, the extended-TSF suffers from the same scalability problems of the original TSF. One could try to improve the extended-TSF even further by allowing extra beacon transmissions only from those nodes that had a larger time-stamp than the time-stamp received in the previous beacon (and same contention window). We performed simulations of this latter approach in which the fastest clock deviates linearly at +25ppm (i.e., gains 2.5usecs with respect to real time every BeaconPeriod = 0.1secs), and the rest of the clocks deviate at –25ppm. Figure 2 shows the cumulative distribution function (c. d. f) of the maximum time difference among the nodes of the network using FHSS parameters. The modified TSF achieves better performance (i.e., the maximum time difference is smaller); however, the small accuracy is gained at the expense of more beacon transmissions. The extended-TSF presented so far might prove to be an approach suitable for networks in which energy is not a constrained.

The probability \( P_{\text{any}} \) can give an indication of the potential improvement of a mutual NTS algorithm over a more hierarchical approach such as the TSF. An analytical result is obtained in [4] that match our simulation result shown in Figure 3. Figure 3 suggests that a mutual network synchronization algorithm based on the same beacons transmitted in the IEEE 802.11 standard can greatly improve over the TSF. For instance, with 100 nodes (DSSS PHY layer parameters), the probability of sending a beacon by any node is approximately 80%, while \( P_{\text{given}} \) is almost zero. This points out that our efforts should be focused on non-hierarchical NTS approaches that exploit the information carried by every beacon transmitted. The IEEE 802.11 TSF however, has the advantage of being relatively simple, which is an important factor in the design of a practical NTS approach.

IV. CLOCK-SAMPLING MUTUAL NETWORK SYNCHRONIZATION

A. Basic-CSMNS Algorithm Model

In this paper an accurate clock is considered to deviate inside the range of ±50ppm. This is well within the range of typical clocks based on quartz-crystals. For instance, the deviation range for the IEEE 801.11clock that drives the Pseudo-Noise (PN) sequence generator must be ±25ppm [1]. Equation 3 models the time process of an accurate clock [11], [12]

\[
T(t) = \beta \cdot t + \xi(t) + T(0)
\] (3)

\( T \) is the time process of the clock, \( t \) is real-time, \( \beta \) depends on the time-drift of the clocks with respect to real-time, \( \xi \) is a random process that models time jittering, and \( T(0) \) is the initial time of the clock. For accurate clocks, the time-drift coefficient can be approximated by a time-invariant constant. \( \xi \) is usually modeled as a random process with a power-law spectra [11], [12]; however its effects can be neglected for our purposes.

Assume a network of \( N \) nodes, each with a clock that has a different time-drift coefficient and initial time. This will result in a set of \( N \) equations of the form

\[
T_i(t) = \beta \cdot t + \xi_i(t) + T_i(0), \quad i = \{1, 2, ..., N\}
\] (4)

The goal is to synchronize all the clocks in such a way that after some time \( t_c \), \( |T_i(t > t_c) - T_j(t > t_c)| \leq \Delta T, \forall \ i \neq j \). Where \( t_c \) is the convergence time of the network synchronization algorithm, and \( \Delta T \) is the tolerable time-error. The different \( \beta \) in (4) make all the time processes \( T_i(t) \) diverge as \( t \to \infty \). The main goal of CSMNS is to minimize the relative time-drifts of the time processes in (4). This is achieved by multiplying every time process by a correction factor \( s_i(t) \); transforming (4) into

\[
\text{Figure 2. c.d.f of the maximum time deviation for the IEEE 802.11 and modified-TSF}
\]

\[
\text{Figure 3.} \ P_{\text{any}} \text{ and } P_{\text{given}} \text{ for the IEEE 802.11 TSF}
\]
\[ \bar{T}_i(t) = s_i(t) T_e t + s_i(t) T_e (0), \quad i = \{1, 2, \ldots, N\} \] (5)

The correction factor can be computed in every node based on the difference between the time-stamp of the received beacon, and the time-stamp of the local node. Node \( i \) contends to send its time process \( \bar{T}_i(t) \) (\( \forall i = \{1, 2, \ldots, N\} \) ) in periodic beacon transmissions in the same way as the IEEE 802.11 TSF. Therefore, \( s_i(t) \) can be written in the following discrete form

\[ s_i(nT) = s_i(nT - 1) + K_p \frac{(\bar{T}_{rx\text{-timestamp}}(nT - 1) - \bar{T}_i(nT - 1))}{\bar{T}_i(nT - 1)} \] (6)

Where \( T \) is the sampling period (e.g., equal to \( a\text{BeaconPeriod} \) in IEEE 802.11), \( K_p \) is the proportional design gain, \( \bar{T}_{rx\text{-timestamp}} \) is the time-stamp of the node that successfully transmitted the beacon. The proportional gain \( K_p \), if chosen appropriately, can average the time processes of all the clocks in the network to achieve synchronization. Substituting (5) in (6) and normalizing the sample time yields,

\[ s_i(n) = s_i(n - 1) + K_p \frac{(s_j(n - 1) \beta_j + s_j(n - 1) T_j(0) - s_i(n - 1) \beta_i - s_i(n - 1) T_i(0))}{s_i(n - 1) \beta_i + s_i(n - 1) T_i(0)} \] (7)

The \( j \) and \( i \) sub-indexes identify the node that transmitted and received the beacon respectively. Note that \( s_j(n) = s_j(n - 1) \). Equation (7), is a non-linear stochastic difference equation that can be solved numerically in \( s_i(n) \) given the initial conditions \( s_i(0) = 1 \) and \( T_i(0) \). \( P_{any} \) is needed in order to determine whether a node successfully transmitted a beacon or not. That is, node \( j \) is randomly selected based on \( P_{any} \). Using the result for \( P_{any} \) [4], we averaged one thousand solutions of (7), and found the estimate of the ensemble average of the maximum time difference (\( T_{max} \)).

Figure 4 shows \( E[T_{max}] \) in a network using FHSS parameters. \( T_{max} = \max (\|T(i) - T(j)\|) \), \( T_i(0) \) is randomly chosen in the range \([0, 100]\), and the time-drift coefficients \( \beta_i \) are randomly chosen in the range \( \pm \frac{25}{1e6} T \). Note the increase in convergence time with the increase in the number of nodes. The effect of \( K_p \) on the convergence of the algorithm was also studied. Figure 5 shows \( E[T_{max}] \) for \( 0.05 \leq K_p \leq 0.3 \) in a FHSS network of 100 nodes. The algorithm presents its fastest convergence time at approximately \( K_p = 0.15 \). At \( K_p > 0.3 \) the solution of (7) becomes unstable with higher probability. The values of \( K_p \) for which instability was observed depends, among other things, on the initial conditions.

Stability can be achieved within wider ranges of \( K_p \) by reducing the initial time errors. Therefore, it is important to set the clocks of incoming nodes as close as possible to the clocks of the nodes already in the network. This can be achieved by listening for beacons before attempting to join the network. An error will always be present, which explains the reason we added an initial random time-offset in the previous results.

B. CMSNS-RMN

A non-hierarchical approach has the additional advantage that not all nodes need to contend for beacon transmission in every TBTT. It is sufficient if only a sub-set of all the nodes contend at any given time. In this case all nodes still have equal opportunity to transmit, but rather in a larger time-span. The following method takes advantage of this feature:

Assume each node has a counter \( C_i \) with maximum value \( C_i^{max} \). In every TBTT, all the nodes perform the following operations independently and in a distributed way:

1. \( C_i = \max \{C_i - 1, 0\} \)
2. If \( C_i = 0 \), contend to send the beacon following the same procedure of the IEEE 802.11 TSF, otherwise if \( C_i > 0 \), wait until next TBTT and return to step 1 without attempting to transmit a beacon.
3. If a beacon is successfully received before the local node sends its beacon (assuming $C_i = 0$), then set $C_i = C_i^{\text{max}}$, adjust the correction factor based on the time-stamp received, wait for next TBTT, and return to step 1.

4. If a beacon is successfully received and $C_i > 0$ then, adjust the correction factor based on the time-stamp received and wait for next TBTT, return to step 1.

Every node will contend to send its time-stamp $T_i(t)$ embedded in a beacon if $C_i = 0$ and no beacon has been received from any other node in the present contention window. If $C_i > 0$, or $C_i = 0$ and node $i$ receives a beacon before it is able to send its own, then node $i$ will not contend to send its beacon in the present contention window. For a while, a group of nodes in the same locality will be listening to beacons coming from the single node winner of the contentions. We call this node the Rotating Master (RM) node. The RM node holds the master status temporarily until a new contention randomly replaces it by another node. In this way, all nodes have the opportunity to be a RM node for a number of TBTTs. CSMNS-RMN reduces the number of beacons transmitted with respect to the basic-CSMNS and the TSF. This translates into energy savings, and beacon collision reduction.

An example of CSMNS-RMN is illustrated in Table II. There are 4 nodes in the same neighborhood. At TBTT = 1, nodes 1 and 4 contend to send their beacons ($C_1 = C_4 = 0$), node 1 wins and becomes the RM. Node 4 sets its counter to 5 as soon as it receives the beacon from node 1 at TBTT = 1; at TBTT = 2, $C_4 = 5 - 1 = 4$. At TBTT = 4, node 2's counter reaches zero, and it contends to send its beacon against node 1. Node 2 wins the contention, but does not win RM status since in the next TBTT node 3 also contends. It is easy to see what the procedure is about afterwards. Note that on average, fewer nodes are contending in every TBTT. The initial values of the counters where chosen different for every node; if the values are the same, all the nodes will contend at the same TBTTs, and a RM node will be determined after probably some collisions, then, after $C_i^{\text{max}}$ TBTTs all nodes will contend again to try to become the next RM node. The previous description does not take into account the wireless medium impairments, which can a beacon to be destroyed even when there are no collisions.

C. Numerical Performance Evaluation of CSMNS-RMN

We performed simulations of the IEEE 802.11 TSF and CSMNS-RMN in Matlab. The simulations were performed assuming no capture and clock models given by (4). A 1% beacon error rate is assumed to model wireless medium impairments. That is, a beacon might not be received properly even if it did not collide with other beacon(s). The time it takes to transmit a beacon is 11 slots, and $d\text{BeaconPeriod} = 0.1$ secs. The total real-time simulated was 30min per simulation [13].

We use a grid topology as shown in the inset of Figure 6. We refer to a specific simulation by its size (e.g., 5x5, 10x10). The ordering of the nodes will follow the same pattern as the one shown in Figure 6 (i.e., node 1 at lower left corner and incrementing towards the up-right direction). The lines joining the nodes correspond to the communication links. For instance, node 1 is able to correctly decode a beacon from node 2, but not from node 7, however, node 7 can interfere with node 1’s reception since the transmission range is taken to be half the detection range. A beacon will not be decoded successfully if there is a collision, or the beacon is received in error due to wireless medium impairments. The hidden and exposed node problems are taken into account in the simulation. An exposed node will defer its beacon transmission if another node transmits within its detection range; a hidden node will cause a collision, if it is within transmission range of a receiving node, but out of the detection range of the corresponding transmitting node. We compare CSMNS-RMN with the TSF using FHSS parameters. Figure 6 shows the c.d.f of $T_{\text{max}}$ in a 5x5 network using TSF. Node 1 has the fastest clock with a time-drift of +25ppm, and the rest of the nodes all drift at -25ppm. The maximum $T_{\text{max}}$ is approximately 900μsecs. A sample of the time difference between nodes 1 and 7, and nodes 1 and 25 is shown in Figure 7. Note that even under no mobility TSF requires the continuous exchange of beacons in order to keep the nodes synchronized; this is because it does not have any mechanism that can achieve convergence. Figure 8 shows a sample of $T_{\text{max}}$ in a 10x10 network using CSMNS-RMN. Figure 8 corresponds to a network that is four times the size of the one used to obtain the previous TSF results. This shows promising scalability and accuracy performance for CSMNS. The time of convergence is approximately 250secs using $C^{\text{max}} = 10$ and $K_p = 0.5$, 147secs for $C^{\text{max}} = 2$ and $K_p = 0.5$, 80secs for $C^{\text{max}} = 2$ and $K_p = 0.8$. Decreasing $C^{\text{max}}$ tends to improve the convergence time at the cost of increasing the overhead. The increase in the proportional gain

<table>
<thead>
<tr>
<th>$T_{\text{TBTT}}$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_i$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>$C_i$</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>$C_i$</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$C_i$</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 6. c.d.f of $T_{\text{max}}$ in a 5x5 network using the TSF
Figure 7. Sample of the time difference between nodes 1-7, and nodes 1-25 in a 5x5 network using the TSF

![Graph showing time difference between nodes](image)

Figure 8. Sample $T_{max}$ in a 10x10 network using CSMNS-RMN

![Graph showing maximum time difference](image)

$K_p$ reduces the convergence time as well, but the algorithm becomes more susceptible to instability as previously discussed. Figure 9 shows the average of $T_{max}$ after performing fifty separate simulations in a 5x5 network with $C_{max} = 10$ and $K_p = 0.5$. The dotted curves show the 95% confidence interval. As can be observed, convergence is achieved after a transition that lasts few seconds.

V. CONCLUSION

We propose CSMNS as a non-hierarchical and mutual network synchronization algorithm for wireless Ad Hoc networks. CSMNS shows superior performance to TSF in terms of accuracy, scalability, and robustness. Table III shows an overall view of the differences between the two approaches. CSMNS is compatible with the beacon messages used in the IEEE 802.11 standard, and it is PHY transparent. CSMNS-RMN is proposed in order to further reduce beacon collisions and overhead. Stability is a factor that must be considered in CSMNS. However, values of the proportional gain below 0.3 suggest a good stability performance. The use of larger $C_{max}$ values in more dense networks, and/or the use of techniques that randomly prioritize the transmission of beacons can further reduce the overhead and risks of instability [13]. Future work includes the implementation of these algorithms in a wireless Ad Hoc network test-bed.

![Table showing comparison between IEEE 802.11 TSF and CSMNS](image)

Table III. Comparison between the IEEE 802.11 TSF and CSMNS

<table>
<thead>
<tr>
<th></th>
<th>TSF</th>
<th>CSMNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (order of)</td>
<td>≈10^{-3}</td>
<td>≈10^{-6}</td>
</tr>
<tr>
<td>Scalability (order of)</td>
<td>Max. 10-20 nodes</td>
<td>Hundreds</td>
</tr>
<tr>
<td>Stability</td>
<td>Yes</td>
<td>By design</td>
</tr>
<tr>
<td>Complexity</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Overhead w.r.t the TSF</td>
<td>-</td>
<td>Less than or equal</td>
</tr>
<tr>
<td>Multi-hop capability</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Convergence</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

REFERENCES