WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2010; **10**:158–170
Published online in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/wcm.905

Behavior of clock-sampling mutual network synchronization in wireless sensor networks: convergence and security

Ereth McKnight-MacNeil[1], Carlos H. Rentel[2] and Thomas Kunz[1*,†]

[1]*Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada*
[2]*Creare Inc., Hanover, New Hampshire, U.S.A.*

Summary

Clock synchronization is an important component of wireless sensor networks (WSNs) both for co-ordination of node communications and for time stamping sensor data. The previously presented clock sampling mutual network synchronization (CS-MNS) algorithm is simple, has low communication and processing overhead, and allows fully decentralized operation. We present some simulation results that indicate the potential of CS-MNS to achieve high clock synchronization accuracy in mobile multi-hop wireless networks. Past work has shown clock convergence under specific conditions in single-hop networks. We show analytically that in the absence of offset errors, the network clocks converge. In the presence of offset errors, we present conditions on the degree of clock asynchrony under which the network clock rates show convergent behavior. The analysis is applicable as long as the network topology is connected and, thus, is of interest in both single-hop and multi-hop environments. As a side result, we also show how a network designer can use these conditions to add a bias term to the CS-MNS algorithm and, thus, improve the start-up dynamics of the algorithm. Furthermore, we discuss the algorithm from a security standpoint. Finally, we propose a method for adding external reference synchronization that is compatible with our security discussion. Copyright © 2009 John Wiley & Sons, Ltd.

KEY WORDS: wireless sensor networks; clock synchronization; simulation results; analytical results; convergence; security

1. Introduction

Clock synchronization is an important foundation of networked systems. The goal is to align the time processes of a network of clocks. Clock synchronization is a key functionality for time-slotted multiple access strategies, data aggregation, security protocols [1], power management, and localization techniques among others. The IEEE 802.11 standard [2], for instance, utilizes network synchronization for power management and frequency hopping, whereas the IEEE 802.16 and the IEEE 802.15 [3] standards depend on network synchronization for their time-slotted medium access control (MAC) protocols. In

*Correspondence to: Thomas Kunz, 1125 Colonel By Drive, Ottawa, Ontario, Canada K1S 5B6.
†E-mail: tkunz@sce.carleton.ca

addition, many wireless sensor networks (WSNs) rely upon distributed clocks to allow correct analysis of collected sensor data.

However, the *ad hoc* and dynamic nature of WSN topologies prevents the straightforward application of traditional centralized, hierarchical clock synchronization strategies. In addition, wireless nodes typically have limited energy and have to be built from cheap, custom hardware, further limiting the applicability of traditional algorithms [4].

Various self-organizing centralized and decentralized algorithms for WSN synchronization have been proposed [5]. Of the many algorithms in the literature, the clock sampling mutual network synchronization (CS-MNS) algorithm [6–8] has many desirable properties. It is fully decentralized, in that all nodes execute the same algorithm at all times. Furthermore, the algorithm does not require knowledge of, and makes no assumptions about the network topology. These properties allow CS-MNS to be applied easily in randomly deployed networks and in dynamic networks (obviously, the performance of the algorithm will vary based on the network topology, but the algorithm itself does not, for example, require to identify a centrally located node as master). Furthermore, the algorithm requires no additional overhead or energy to run adaptation procedures neither in response to changing radio propagation conditions nor in response to nodes leaving the network through battery depletion or otherwise.

The simple beacon format of CS-MNS is compatible with the beacon format and beacon contention method used in IEEE 802.11 or IEEE 802.15.4. The CS-MNS algorithm can therefore be implemented in software with standard radio and clock hardware [8]. This makes CS-MNS applicable for use in networks of currently available hardware and in cost-sensitive consumer devices that must use commodity radio hardware.

In this paper, we briefly review the basic algorithm and show some simulation results that argue for the good performance of the algorithm. We then further analyze the behavior of the CS-MNS algorithm in WSNs. We apply Moreau's main theorems [9] to CS-MNS in the absence of clock offset errors to show that the network clocks converge as long as the network satisfies a weak connectivity constraint. Furthermore, in the presence of offset errors we give a relationship for the relative magnitude of the offset error and rate error under which the essential properties of the update law required for convergence are preserved. This leads to the statement of conditions under which the behavior of CS-MNS is convergent in the presence of offset errors. We show how the designer can use this understanding of the behavior of CS-MNS to improve the initial transient behavior of CS-MNS by adding a bias term to the algorithm.

We discuss the possibility of security, based primarily on limiting attacker influence and algorithm robustness, as opposed to energy intensive cryptographic functions. We then propose how CS-MNS could be extended to allow for external synchronization that is compatible with our discussion of security.

2. Related Work

Gersho and Karafin [10] were one of the first to analyze a system that mutually synchronized the phases of a group of geographically separated oscillators connected by communication links. The classical mutual network synchronization approach [10] was originally designed for wired networks, and it requires dedicated circuitry to generate either narrow pulses or continuous waveforms that can potentially occupy a considerable portion of the allocated frequency spectrum. If used over wireless channels, Gersho and Karafin's mechanism can impose tight restrictions on the turn-around times of the radios if a half-duplex strategy is used since nodes are expected to transmit and receive the timing signals simultaneously. Otherwise, full-duplex radios are needed, which are more complex, more expensive, and require more energy to operate. Using methods that require continuous transmission of waveforms over a standard, such as IEEE 802.11 or IEEE 802.15.4, is not only prohibitive due to potentially large energy and bandwidth requirements, but also very difficult to implement in practice. That is, access to the physical clocks is required in order to adjust their frequency, or possibly different PHY layer components are necessary in order to transmit and receive the required waveforms. CS-MNS' nonlinear discrete control law differs from the ones used by Gersho and Karafin [10] and other mutual synchronization approaches proposed in the past [11–14], which are either controlling the frequency of the clocks directly, or are designed and analyzed in a continuous-time domain. CS-MNS' nonlinear control law is a consequence of adjusting time process drifts rather than time offsets in the time domain. The control law has the interesting property of using a limited amount of information. The latter has the advantage of reducing the complexity of the algorithm and ensuring its compatibility to existing PHY/MAC standards.

The reference broadcasting synchronization (RBS) scheme can be extended to provide multi-hop support [15]. RBS achieves multi-hop synchronization through intermediate nodes that translate the times among different neighborhoods. RBS achieves clock adjustment through linear regression, and it utilizes the idea of receiver-synchronization to eliminate the timing inaccuracies caused by medium access uncertainty.

The IEEE 802.11 timing synchronization function (TSF) [2] utilizes clock-sampling to explicitly distribute time throughout the network. Clock sampling refers to the use of messages carrying explicit time stamps rather than signals with embedded timing information such as a train of pulses. The time is simply read from the clock and transmitted in a message called beacon. The TSF is used to support power management and the channel hopping procedure used by the frequency hopping spread spectrum (FHSS) version of IEEE 802.11. The TSF utilizes the same principle of clock correctness introduced by Lamport [16]. Part of that principle states that a clock's time shall not move backward. Therefore, nodes implementing the TSF in the *ad hoc* mode of operation adjust their time only to faster clocks in the network.

The lack of scalability of the IEEE 802.11 TSF was first analyzed by Huang and Lai [17] for single-hop networks. The authors also proposed a method to improve scalability by giving higher priority to the beacon transmissions of the node with the fastest clock in the network [17,18]. However, the latter approaches are not suitable for multi-hop wireless *ad hoc* networks.

Sheu *et al*. presented a synchronization method for multi-hop wireless *ad hoc* networks based on an automatic self-time-correcting procedure (ASP) [19]. ASP alleviates the scalability problem of TSF and also works for multi-hop wireless *ad hoc* networks. However, ASP needs to modify the beacon of the standard IEEE 802.11 TSF in order to work properly. Furthermore, clock adjustment is achieved through a modified additive time offset rule, which cannot achieve better accuracy than methods that correct the frequency or skews of clocks. The self-adjusting TSF (SATSF) [20] improves the accuracy of the ASP without changing the IEEE 802.11 beacon format, but for a single-hop network. Later, the Manet TSF (MATSF) was proposed for multi-hop IEEE 802.11-based *ad hoc* networks [21]. MATSF, similar to ASP, also modifies the standard beacon frame used in the IEEE 802.11 standard. However, MATSF shows better accuracy than ASP due to the fact that it adjusts the frequency of the clocks. As we will show later, CS-MNS performs better than either of these protocols.

Many protocols [17–21] depend on the identification of 'appropriate' clocks for beacon transmission prioritization, which may be a difficult task in a mobile *ad hoc* network. CS-MNS does not depend on correctly identifying a suitable subset of nodes with good properties (centrally located, fastest clocks, etc.), as the algorithm is fully distributed and all nodes contribute equally to the resulting network-wide synchronization.

The basic CS-MNS protocol was originally published in [6], explaining the core protocol, introducing variants, and comparing CS-MNS to IEEE 802.11 TSF in static scenarios. Additional simulation results were presented in Reference [7], demonstrating the performance of CS-MNS in the presence of node mobility. Finally, Rentel and Kunz [8] provide a more detailed description of the protocol and its performance, and proves necessary conditions for algorithm stability and convergence for two specific scenarios, using the discrete Lyapunov direct method. However, as stated in Reference [8], stability and convergence for any number of nodes and arbitrary topology remained an open problem. In this paper, we add on these results by showing that in the absence of offset errors the network clocks converge, based on a theorem originally published in Reference [9]. In the presence of offset error, we present conditions on the degree of clock asynchrony under which the network clock rates show convergent behavior. The analysis is applicable as long as the network topology is connected and thus is of interest in both single-hop and multi-hop environments. As a side result, we also show how a network designer can use these conditions to add a bias term to the CS-MNS algorithm and, thus, reduce the initial transient increase in maximal clock difference that the original CS-MNS protocol suffers from, as shown in the simulation results presented later. Furthermore, we discuss the algorithm from a security standpoint. Finally, we propose a method for adding external reference synchronization that is compatible with our security discussion.

3. Basic Algorithm Operation

We adopt a model for the node clock termed an affine clock [22], that presents the clock at each node as an affine transformation of a reference clock. Thus, the time process at node *j* is given by

$$T_j(t) = \alpha_j t + \beta_j \tag{1}$$

where t represents the reference time process. We assume that the clock rates and offsets, α_j and β_j,respectively, remain constant in time. The corrected clock then becomes

$$\widehat{T}_j(t) = s_j T_j(t) = s_j(\alpha_j t + \beta_j) \tag{2}$$

where the CS-MNS algorithm controls the correction scale factor s_j.

As discussed in more detail later, CS-MNS performs better when the initial clock offsets (β_j) are close to each other. CS-MNS achieves this by using a coarse synchronization step at the beginning of the algorithm. A new node, hearing a CS-MNS protocol message for the first time, will set its internal clock to the advertised time stamp. As a result, the initial clock offsets are determined primarily by message latency and they thus are close to each other. The clock rates/drift of a local clock is defined by $\alpha_j - 1$. Many wireless technologies bound the allowed drift on local clocks. In IEEE 802.15.4, for example, the clock drift is bound by ±40 ppm, and in IEEE 802.11 the bound is ±25 ppm.

The CS-MNS algorithm running on node j, upon receiving a clock sample from another node i at reference time τ, updates the factor s_j using the update law

$$s_j^+ = s_j + k_p \frac{\widehat{T}_i(\tau) - \widehat{T}_j(\tau)}{T_j(\tau)} \tag{3}$$

where s_j^+ is the updated value and k_p is a control gain.

Substituting the updated correction factor from Equation (3) into Equation (2), rearranging arrives at

$$\widehat{T}_j^+(t) = \left(1 + k_p \frac{\widehat{T}_i(\tau) - \widehat{T}_j(\tau)}{\widehat{T}_j(\tau)}\right) \widehat{T}_j(t) \tag{4}$$

as the new time process for node j.

Immediately, it is evident that the clock model incorporates clock error with two degrees of freedom, rate, and offset, while the adjustment mechanism has only a single degree of freedom. However, practical considerations require that CS-MNS bases adjustments on a single input sample and, thus, works with a single input dimension.

Extracting the relative clock rate in order to generate a second input dimension is theoretically possible if the adjustments are based on multiple received samples. However, in dense networks with many neighbors, tracking each neighbor has excessive memory complexity and requiring multiple beacons decreases the likelihood that a received beacon will be useful. These considerations are especially important for mobile nodes whose neighbors change quickly.

Another approach to clock adjustment introduces a second degree of freedom by including an integration term in the corrected clock model. However, this introduces a phase lag that grows as synchronization information is passed from node to node. If the information flow travels in a sufficiently large loop, the resulting closed loop system will have no phase margin remaining and become unstable. Again, the complexity that would be required to avoid these unstable conditions appears to be prohibitive.

Thus, side-stepping the above complexities and selecting the single adjustment factor as in Equation (2) allow both rate and offset to be adjusted—albeit not independently—while retaining simplicity.

4. Initial Simulation Results

To demonstrate the capabilities of CS-MNS, we briefly present some simulation results that compare CS-MNS to the IEEE 802.11 TSF. The simulations were conducted in Matlab, modeling the PHY *via* the simple two-ray ground propagation model commonly used in NS2 (i.e., perfect reception of messages within a given transmission range, here 250 m). Randomly about 1% of beacons are dropped to model wireless medium impairments. At the MAC layer, we model a Carrier Sense Multiple Access (CSMA) operation; nodes can only receive if the transmitter is at most 250 m away, but will detect transmission by other nodes (and therefore defer transmission) up to a distance of 500 m, similar to common IEEE 802.11 simulation scenarios. Performance results are presented for different network sizes and two mobility models. The performance is measured in terms of the maximum time difference between any two clocks at every beacon transmission time. This is denoted as $T_{\max}$ (given in microseconds—μs). We also compare the accuracy of CS-MNS to the results reported for MATSF [21] and ASP [19]. To demonstrate the dynamic behavior of CS-MNS and TSF, we plot the detailed results for single simulation runs. However, the algorithms behave qualitatively the same when repeated multiple times.

Mobility makes synchronization more challenging for several reasons. First, mobility can cause network partitions, which in turn cause two or more portions of the network to time-evolve differently. Second, methods that synchronize to a master or reference clock usually utilize ranked topology strategies, or bridge and gateway nodes to serve as proxies that synchronize different portions of a multi-hop network (e.g., [15,19,21]). These methods usually require for each node to keep a record of its neighbors in order to decide which one will be used as a link to the reference clock. However, this is more challenging (requiring additional protocol messages) when the nodes move. Finally, mobility can cause more time and frequency (i.e., Fading and Doppler) wireless channel variations that affect the PHY layer performance and, therefore, the efficient exchange of timing information in a synchronization method. A mutual network synchronization approach is particularly attractive because it is more resilient than a master–slave approach to the second impairment mentioned above. That is, since no hierarchies (i.e., ranking of nodes), reference, or master clocks are needed, a node participating in a mutual network synchronization approach does not need to keep track of its position or keep a record of any network topology information.

The random waypoint (RWP) mobility model [23] is simulated with a maximum speed of 5 m/s, and maximum pause time of 50 s. The boundless area (BA) mobility model [24] does not make use of pauses, it has a maximum speed of 5 m/s, a maximal acceleration of 0.5 m/s^2, and a maximum change in direction of 0.5 rad/s. In both models the area is 1000 m × 1000 m. The BA model area is effectively a toroid, therefore, nodes leaving one side will reappear on the other side, whereas nodes in the RWP model will move inside the area if they hit the edges. Studies have shown problems with the convergence of the RWP model [25] when the minimum speed is zero. We therefore use a minimum speed greater than zero in the simulations. Accuracy of the clocks in the network vary randomly in the range [−25 ppm, +25 ppm] (i.e., all α_j are in the range $[1 - 25/10^6, 1 + 25/10^6]$, and all the clocks start asynchronous to one another in the range (0, 200 µs]. As discussed above, this fairly tight initial synchronization can be achieved *via* an initial synchronization step, where new nodes simply adjust their local clock to the time stamp carried in a CS-MNS message. Achieving a fairly tight initial synchronization is important to allow nodes to roughly agree (in time) on the time window during which synchronization messages are exchanged. Beacon messages are exchanged every 100 ms, the default beacon interval in the IEEE 802.11 TSF. The rest of the simulation parameters are the same as in Reference [6].

4.1. Network Size and RWP Mobility

Figures 1 and 2 show $T_{\max}$ (µs) using TSF and CS-MNS respectively under RWP mobility. The network sizes are 100, 300, and 500 nodes. In all cases the transmission and detection ranges are 250 and 500 m, respectively. TSF (Figure 1) shows average $T_{\max}$ of 233, 644, and 730 µs and a maximal $T_{\max}$ of 520, 925, and 1,026 µs for 100, 300, and 500 nodes, respectively. CS-MNS' behavior (Figure 2) is different from that of TSF, there is a transition period that reaches peaks of 667, 805, and 917 µs for 100, 300, and 500 nodes, respectively. These peaks occur as the initial few rounds will not adjust the clock drifts enough to enforce convergence right away, however, after approximately 40, 120, and 250 s, CS-MNS converges to a value less than 10 µs. As will be discussed later, this initial divergence can be avoided in most cases by selecting an appropriate bias factor. CS-MNS reduces the time difference among the clocks as time progresses, a property TSF cannot claim due to the fact that it adjusts time based on time-offsets only. Note that as the density of the network increases, it is more difficult to have successful beacon transmissions, and the average maximum deviation between the clocks tends to increase. CS-MNS however, is less affected by the density increase because nodes make use of any beacon transmitted, not only of those from faster clocks.

Figure 2 also shows that $T_{\max}$ does not continuously improve. As beacons are lost due to collisions or channel impairments, nodes loose the opportunity to update their local clocks. As long as clocks drift at different rates, a prolonged loss of beacons will cause $T_{\max}$ to grow, explaining the local maxima in the graph.

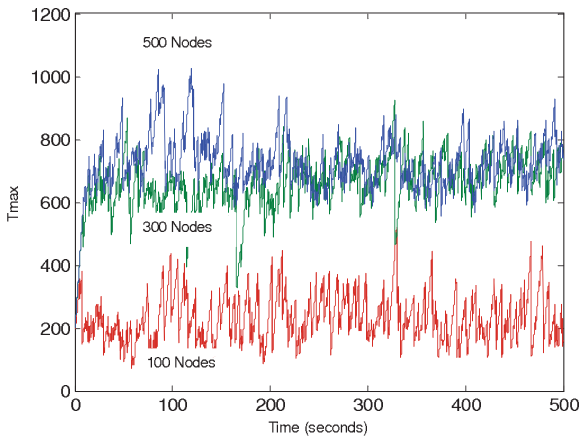


Fig. 1. TSF $T_{\max}$ for different network sizes, RWP mobility model.

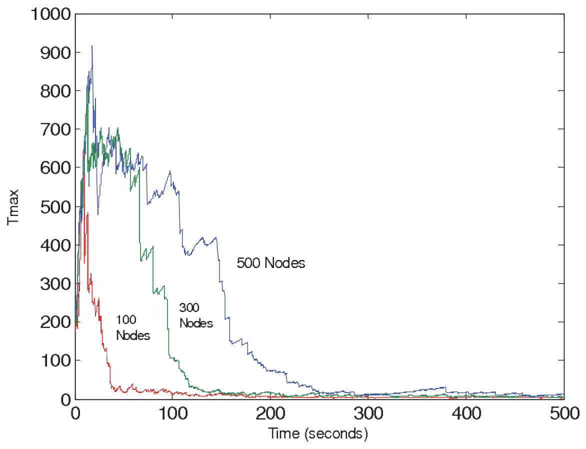


Fig. 2. CS-MNS T_{max} for different network sizes, RWP mobility model.

4.2. Boundless Area (BA) Mobility

Figure 3 shows T_{max} for CS-MNS and TSF for 500 nodes (which is the most demanding for any network synchronization protocol) using the BA mobility model. The transmission and detection ranges are again 250 and 500 m, respectively. CS-MNS converges for both the RWP and BA mobility models, even for 500 nodes. The BA model shows a slightly larger overshoot in the transition period, which lasts approximately 200 s. The final T_{max} is approximately 9 µs when using the BA mobility model, and 6 µs when using the RWP mobility model. The average T_{max} for TSF (Figure 3) is 721 µs, this is slightly smaller than the result obtained with the RWP model (730 µs). There is no substantial difference observed in the performance under both mobility models, though more work is required to analyze the slight performance variations as a function/property of the mobility model.

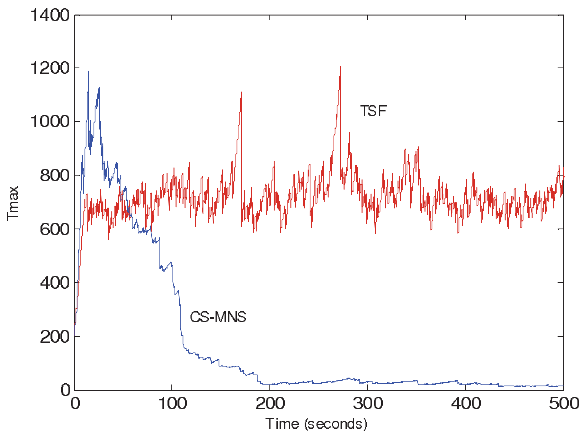


Fig. 3. CS-MNS and TSF T_{max} for 500 nodes and BA mobility.

Table I. T_{max} (µs) for CS-MNS, MATSF, and ASP.

N	CS-MNS	MATSF	ASP
100	2	24	214
300	3	24	$\approx$ 200
500	6	43	$\approx$ 200

4.3. Comparison to MATSF and ASP

The previous simulations show the ability of CS-MNS to converge under different network sizes and mobility patterns. The T_{max} values reported for MATSF [21] and ASP [19] for similar parameters are summarized in Table I, together with our experimental results for CS-MNS. CS-MNS is up to approximately five times more accurate than MATSF, and 100 times more than ASP, according to the reported accuracies of these two methods. In addition, CS-MNS achieves better accuracy while being compatible to the beacons of IEEE 802.11 or IEEE 802.15.4.

5. Algorithm Convergence

The simulation results show that CS-MNS is capable of achieving very good clock synchronization. Yet the results also show that the algorithm has an initial divergent behavior, and the simulation results do not conclusively demonstrate that the algorithm will always converge. This is addressed in the following subsections.

5.1. Convergence Without Offset Errors

Hoping to gain insight into the behavior of a network running CS-MNS, we consider the special case when all $\beta_1 = \cdots = \beta_n = 0$ in Equation (1) above. That is, the case when clock error consists entirely of relative clock rate error. In this case, the corrected time process of Equation (2) becomes simply $\widehat{T}_j(t) = s_j\alpha_j t$. Following through, Equation (4) simplifies to

$$\widehat{T}_j^+(t) = ((1-k_p)s_j\alpha_j + k_p s_i\alpha_i)t \tag{5}$$

which has a rate that is a convex combination[‡] of the previous corrected clock rates at nodes i and j.

The above convex combination of Equation (5) emits exactly the requirements given by Moreau as Assumption 1 [9]. Thus, from Theorem 1 of Reference [9] we can immediately conclude that, in the absence of offset errors, the CS-MNS adjustments cannot increase the overall level of clock rate error regardless of which nodes communicate.

In order for convergent behavior to emerge, the communication between nodes must, in a finite period of time, allow at least one node to pass information to all other nodes [9]. Obviously, this condition is trivially satisfied for single-hop networks. Furthermore, any network that is 'connected' in the usual multi-hop sense satisfies this communication requirement. Thus, for 'connected' networks in the absence of offset errors, Moreau's Theorem 2 of Reference [9] applies and the 'system [...] is uniformly globally attractive with respect to the collection of equilibrium solutions.'[§]

In summary, if clock offsets errors are absent, then the application of CS-MNS has a positive effect on the clock synchronization among nodes. If the network is not connected, then CS-MNS will not increase the overall level of disagreement. However, if the network is connected, then CS-MNS will cause the clock rates to converge under this model. Of course, if offsets are absent then converging clock rates imply converging clocks.

5.2. Convergence in the Presence of Offset Errors

In addition to the result in the previous subsection, we would like to arrive at a result that applies in the presence of offset errors. Clock rate makes an increasingly large contribution to synchronization error as time progresses while the contribution of the offset error remains constant over time. Thus, we intuitively expect that the contribution of clock rate errors begin to dominate over the contribution of the offset and hope that CS-MNS will exhibit the same convergent behavior shown in the previous section.

[‡]We assume $k_p \in (0, 1)$, as k_p exceeding 1 will guarantee instability of the control law.

[§]The precise definition of uniform global attractivity in this case is defined by Moreau in Appendix I of Reference [9].

We proceed by adding an error term to clock rate in Equation (5) so that it is equal to the rate in Equation (4)

$$\frac{\mathrm{d}\widehat{T}_j^{+}(t)}{\mathrm{d}t} = (1-k_p)s_j\alpha_j + k_p s_i\alpha_i + k_p s_i\alpha_i\epsilon_{i,j}(\tau) \tag{6}$$

where in the error term,

$$\epsilon_{i,j}(\tau) = \frac{\beta_j - \frac{\alpha_j}{\alpha_i}\beta_i}{\alpha_j\tau + \beta_j} \tag{7}$$

arises from the non-zero values of β_i and β_j.

By defining

$$\lambda = \min(1-k_p, k_p) \tag{8}$$

then if Equation (7) satisfies

$$\frac{k_p}{\lambda}\epsilon_{i,j}(\tau) < \left|\frac{s_j\alpha_j - s_i\alpha_i}{s_i\alpha_i}\right| \tag{9}$$

the clock rate in Equation (6) must lie strictly between $s_i\alpha_i$ and $s_j\alpha_j$. In other words, despite the error term, the updated clock rate will lie in the interior of the range between the previous corrected clock rates. Thus, under the condition (9) the essential property that led to convergent behavior in the case without offset error is preserved.

By choosing

$$\alpha_{\min} = \min_i \alpha_i$$

$$\beta_{\min} = \min_i \beta_i$$

$$\gamma = \max_{i,j}\left|\beta_j - \frac{\alpha_j}{\alpha_i}\beta_i\right|$$

then $\epsilon_{i,j}(\tau)$ can be bounded over all i, j as

$$\epsilon_{i,j}(\tau) \le \epsilon_{\max}(\tau) = \frac{\gamma}{\alpha_{\min}\tau + \beta_{\min}} \tag{10}$$

which becomes smaller with increasing time. Indeed, $\lim_{\tau\to\infty}\epsilon_{\max}(\tau) = 0$.

Finally, using $\epsilon_{\max}(\tau)$ and Equation (9) we arrive at the condition

$$\frac{k_p}{\lambda}\epsilon_{\max}(\tau) < \min_{i,j}\left|\frac{s_j\alpha_j - s_i\alpha_i}{s_i\alpha_i}\right| \tag{11}$$

under which all possible CS-MNS rate updates exhibit the required properties for convergent behavior. Thus, regardless of network topology, any period in which all relative clock rate errors are greater than a threshold will be periods in which the clock rates show convergent behavior. In addition, Equation (10) shows that this threshold grows smaller as time passes.

Equation (11) can fail to be satisfied if only a few nodes have very similar clock rates. However, Equation (9) is stronger. In cases where Equation (11) is not satisfied among all nodes, the updates between nodes with largely differing clock rates will still satisfy Equation (9) for each update. Thus, in practice many nodes continue convergent behavior even when Equation (11) cannot be used to guarantee convergent behavior of the system.

5.3. Introducing a Bias Term to Improve Start-Up

When a synchronization period begins, the value of τ in Equation (7) is small, leading to a large value for $\epsilon_{i,j}(\tau)$. Thus, during the initial synchronization period many updates do no satisfy Equation (9) and the CS-MNS algorithm makes large changes to the clock rates based primarily on the clock offsets. This causes divergent behavior of the clock rates until $\epsilon_{i,j}(\tau)$ shrinks sufficiently. However, this divergent period serves to desynchronize the clocks and once convergent behavior emerges the algorithm must synchronize clocks that are further apart than their initial misalignment.

To avoid this initial divergent behavior Equation (1) can be modified to

$$T_j(t) = \alpha_j t + \beta_j + \beta_{\text{BIAS}} \tag{12}$$

where β_{BIAS} is a constant. With this modification $\epsilon_{\max}(\tau)$ in Equation (10) becomes

$$\epsilon_{\max}(\tau) = \frac{\max_{i,j}\left|\beta_j - \frac{\alpha_j}{\alpha_i}\beta_i + \left(1 - \frac{\alpha_j}{\alpha_i}\right)\beta_{\text{BIAS}}\right|}{\alpha_{\min}\tau + \beta_{\min} + \beta_{\text{BIAS}}} \tag{13}$$

and the designer can control the initial magnitude of $\epsilon_{\max}(\tau)$ by selecting β_{BIAS}.

In order to select an appropriate value of β_{BIAS}, the system designer can use Equation (11). By substituting the expected initial values from hardware tolerance and initial synchronization accuracy, and substituting the maximum initial rate error between two nodes on the right hand side, the designer can solve Equation (11) for τ. Simulation indicates that this value of τ then makes a good starting point for β_{BIAS}.

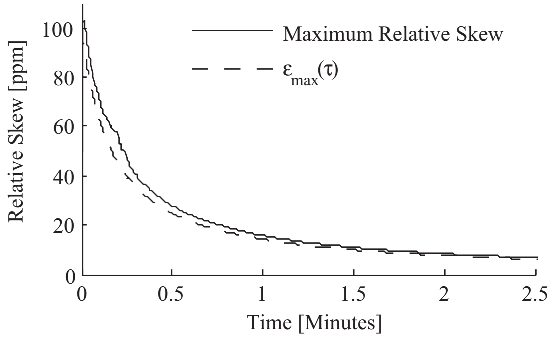


Fig. 4. Simulation results for maximum relative rate error plotted with the predicted convergence limit $\epsilon_{\max}(\tau)$.

5.4. Simulation Results

The maximum clock rate errors from simulation of a small single-hop network of five nodes is plotted against the calculated value of $\epsilon_{\max}(\tau)$ in Figure 4. A bias term is used and the maximum rate error can be seen to track $\epsilon_{\max}(\tau)$. This illustrates the intuitive understanding that as $\epsilon_{\max}(\tau)$ shrinks, nodes with the largest rate error will satisfy Equation (9) and start to move closer to the other nodes, thus reducing the maximum rate error.

The beneficial effect of a bias term calculated as explained above can be seen in Figure 5. A 25 node multi-hop network is arranged in a 5 × 5 regular grid with the transmission radii equal to grid spacing. The initial synchronization is within 1 ms and clocks rates are accurate to ±50 ppm. The beacon rate is 10 Hz (i.e., every 100 ms). For the system without the bias term, the maximum deviation between clocks peaks at 133 ms. For the system with $\beta_{\text{BIAS}} = 10$ s, the maximum deviation does not go above the initial value of 1 ms. The maximum deviation for the system with

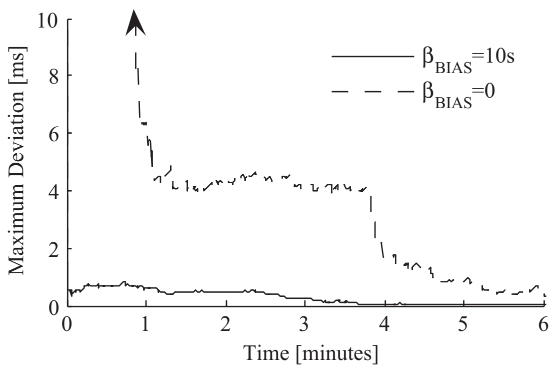


Fig. 5. Simulation results for a multi-hop network arranged in a 5 × 5 regular grid with transmission radii equal to grid spacing.

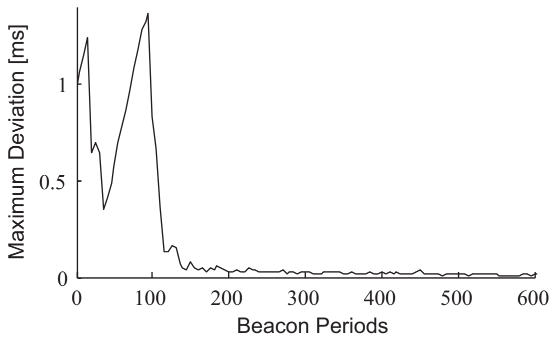


Fig. 6. Simulation results for a single-hop network of 324 nodes representing a large *ad hoc* network.

the bias term does not go above 50 μs from just after 5 min until the end of the simulation at 15 min.

As shown by Zhou *et al.* [26], the IEEE 802.11 TSF is shown to suffer from asynchrony in large *ad hoc* groups consisting of hundreds of nodes. One contributing factor is TSF's inability to make adjustments towards a slower clock. Another shortcoming of TSF is that it does not adjust the clock rates and because of this must continually correct the accumulating offset errors. These deficiencies are corrected in CS-MNS and our analysis does not suggest any upward limit on network size. Since CS-MNS shares a compatible beacon format with TSF, the question of CS-MNS' performance for large *ad hoc* groups is raised.

Figure 6 shows simulation results for a single-hop network consisting of 324 nodes. The node clocks are accurate to 100 ppm and a bias value of 5 s is used. The beacon rate is again 10 Hz. The maximum deviation between nodes remains below 25μs after 455 beacon periods and below 10 μs after 1205 beacon periods (100 ms) until the end of the simulation at 3000 beacon periods. This degree of synchronization compares well with the ad hoc network requirements outlined in reference [26]. Further work is required to study behavior during inter-operation of CS-MNS with legacy nodes running TSF.

6. External Synchronization and Security

The original CS-MNS algorithm [6–8] neither makes any provisions for security nor for external synchronization. Du and Chen [27] provide an overview of the need for, and the challenges of, security in WSNs including a discussion of the need for secure time synchronization. Some secure time synchronization schemes based on cryptographic methods have been proposed, for example Sun *et al.* [28] where the authors use the μTESLA broadcast authentication scheme to achieve security. However, message and processing complexity are necessarily increased over unencrypted protocols.

We begin by discussing the addition of simplistic unsecured external synchronization to CS-MNS. We then proceed by analyzing the security implications of these additions and propose a two-layer external synchronization method that does not introduce large encryption overheads. The basic approach is to limit the disturbance an attacker can introduce and then to rely on algorithm robustness to mitigate the effectiveness of the attack.

6.1. Insecure External Synchronization

One simple method of adding external synchronization to a network running the CS-MNS algorithm would be to introduce reference nodes. These reference nodes would participate in the same beacon contention as CS-MNS nodes but would broadcast the external reference clock as the beacon time stamp. These reference nodes would ignore any received beacons as they would have no need to adjust their clocks.

It is important that the reference nodes participate normally in beacon contention. In a multi-hop network, increasing the probability that reference nodes win the beacon contention means that nodes neighboring a reference node have a lower beacon transmission probability. This, in turn, has the effect that nodes at two hops from a reference node receive fewer beacons.

If all reference nodes are synchronized to the same external source and are functioning correctly, the time stamp beacons from these nodes will be identical and these nodes can be considered as one node for transmission. As the reference nodes discard any incoming clock samples they can be considered as a single unreachable node for reception. Thus, whatever the true network topology, for analysis the topology can be considered to have a single reference node that can transmit to many nodes but cannot receive.

We note that these conditions are exactly the minimum connectivity requirements under which the previous analysis held; the case when there is at least one node that can communicate to all other nodes. Therefore, the addition of any number of externally synchronized reference nodes does not effect the presented analysis of behavior except to control the admissible equilibrium conditions. Preliminary simulation indicates that convergence is slowed when reference nodes are introduced into the network.

However, further work is required to study the effect of differing reference node densities.

The simplicity of this external scheme may make it attractive enough for use in networks that have lax security requirements. However, requiring that the protocol allows nodes that make no clock adjustments to participate creates security concerns as explored below.

6.2. Security Through Outlier Detection

Song *et al*. [29] identified four types of attacks against network time synchronization protocols: masquerade attack, replay attack, manipulation attack, and delay attack. These are in addition to more general denial of service and other jamming attacks that do not target the synchronization protocol specifically.

However, in CS-MNS, where the messages are simple time stamped beacons that are broadcast to all nodes in range of the transmitting node, many of the attacks become equivalent. Since the nodes need not identify themselves in the beacon, masquerading as another node is identical to fabricating a beacon. Similarly, since the beacon is a simple time stamp, replaying a message is also equivalent to fabricating a beacon. Since CS-MNS does not use multi-hop routing of individual synchronization messages, there is no opportunity for an attacker to either manipulate or delay messages en route to other nodes. This leaves one viable type of attack that can be perpetrated against CS-MNS, which is the fabrication of incorrect beacons.

One possible strategy to protect against fabricated beacons is to have each node attempt to detect and ignore outliers in the received beacon stream. As long as nodes joining the network perform coarse initial synchronization when they join, then the time processes of all cooperative nodes are expected to be similar. Song *et al*. [29] explored the outlier rejection approach using both generalized extreme studentized deviate (GESD) and a much simpler threshold approach.

The case of a fabricated beacon attack is slightly different from that in Reference [29], but a similar threshold approach appears promising for CS-MNS. Restricting the maximum acceptable deviation of a received beacon limits the amount of influence an attacker can have on the system. If this influence is small enough, the system will simply tolerate the attacker. Indeed, the effect of a fabricated beacon designed to fall within the threshold is similar to the effect of clock offset errors explored above.

There remains a danger that an intelligent attacker could bias the system towards ever increasing or ever decreasing clock rates. While theoretically the network could remain internally synchronized, an unbounded increase or decrease in network clock rates would eventually cause numerical difficulties. Furthermore, a group of attackers may attempt to exploit the threshold detection to partition the network clocks into groups separated by more than the threshold. Solutions to these problems remain to be explored.

Finally, outlier detection poses two problems for the simple reference node scheme outlined above. Eventually, circumstances will arise where the reference nodes are themselves outliers from the group and they then will become ineffective. Also, an attacker can prevent the reference nodes from bringing the rest of the network into synchrony with the external reference by acting as a reference node supplying an incorrect reference.

6.3. Securing External Synchronization

We propose a different approach to external synchronization that is more compatible with security concerns. We run the CS-MNS algorithm with simple outlier detection but otherwise unsecured. This retains the simplicity of the algorithm and avoids any encryption overhead. If reference nodes act as standard nodes and run the CS-MNS algorithm, the reference nodes can then calculate rate and offset correction factors between the internal network clock and the external clock. At this point, external synchronization of the network is a matter of disseminating this correction data to the non-reference nodes.

The presence of reference nodes in the network actually protects against coordinated attacks that aim to partition the network. As long as each partition contains a reference node, even when the network clocks between partitions do not agree, the reference nodes will distribute appropriate corrections to each partition to synchronize all nodes to the external reference.

As the CS-MNS internal network clock adjusts smoothly over time, the correction factors calculated by the reference nodes can be updated at a slower rate than the beacon rate required to synchronize the network. Therefore, more energy intensive cryptographic procedures can be tolerated to validate this information. For example, a signature scheme based on asymmetric cryptography would make it difficult for an attacker to introduce incorrect corrections even if the attacker had full access to a non-reference node.

If the corrections from each reference node are disseminated through the network, each node can defend against counterfeit reference nodes by eliminating outlier correction factors and using the average of accepted correction factors. This type of algorithm can be chosen to tolerate a given ratio of attackers to reference nodes which may be sufficient for many applications.

6.4. Preliminary Simulation Results

Figures 7 and 8 show preliminary simulation results for CS-MNS using threshold beacon filtering. The acceptance window was heuristically set at a constant $\pm 100\,\mu$s of the adjusted clock at the receiving node. This acceptance window is smaller than the initial offset errors and initially falsely rejects beacons from some co-operative nodes.

However, by modifying the beacon contention mechanism, adverse effects of these false rejections can be mitigated. If a node rejects a received beacon, the node continues to compete for beacon transmission instead of backing-off until the next beacon period. This allows multiple beacons to be transmitted per beacon period when the population of network clocks is widely dispersed and increases the likelihood that a node will receive a beacon within its acceptance threshold.

Figure 7 shows that an attacker that respects the protocol but introduces a random offset error into their beacon data can continually disrupt synchronization. However, the disruption is contained within the range of the acceptance window and the overall convergent behavior of the network is not impacted.

Figure 8 shows a simple attack that attempts to partition the network by introducing two malicious 'reference' nodes. These attackers broadcast beacons normally but do not adjust their differing, diverging clocks. The attack fails to partition the network.

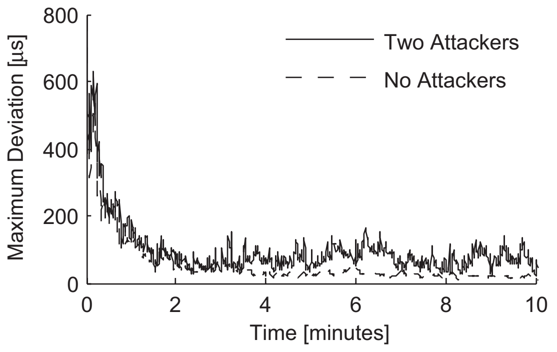


Fig. 7. Simulation results for a 5 × 5 regular grid network with and without attackers located at opposite corners with random intentional errors in their beacon values.

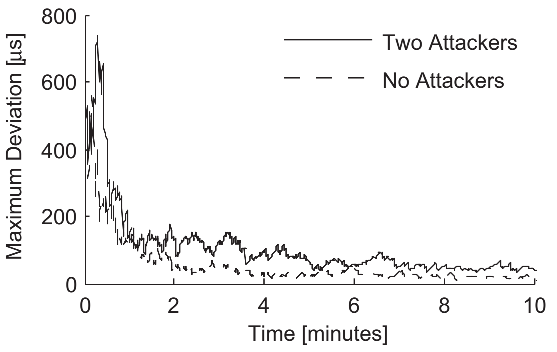


Fig. 8. Simulation results for a 5 × 5 regular grid network with and without attackers located at opposite corners with differing fixed clock rates.

7. Conclusions and Future Work

In this paper we have applied the results from Reference [9] to show the convergence of CS-MNS in the absence of offset errors. We develop conditions under which CS-MNS shows convergent behavior in the presence of offset errors. We then apply the understanding of CS-MNS behavior to improve start-up by introducing a bias term to the CS-MNS algorithm. We then show simulation results for the improved algorithm. Furthermore, we show simulation of CS-MNS converging in a network of over 300 nodes.

We present a discussion of the introduction of external synchronization to the CS-MNS algorithm. We discuss the possibility of security based on algorithm robustness as opposed to encryption.

Preliminary simulation results show the performance of threshold-based attacker rejection to the CS-MNS algorithm. The discussed ideas would benefit from future work that develops a tighter, dynamic threshold for beacon rejection, perhaps based on the expected network convergence rate.

We suggest that other future work could attempt to apply the same analysis approach to other synchronization algorithms. The independence from topology, and thus application to networks with changing connectivity, makes Moreau's results in Reference [9] particularly well suited to the domain of multi-hop sensor networks.

Work currently underway related to the CS-MNS algorithm includes implementation on a testbed of wireless sensor nodes running TinyOS for verification of the theoretical and simulation results. Furthermore, the implementation of the discussed security ideas and the integration of reference nodes for external synchronization forms part of this work. The preliminary results to date indicate that CS-MNS does indeed perform as predicted through analysis and simulation. For example, the basic CS-MNS protocol shows the initial transient behavior that is evident in Figures 2 and 3, but the clocks ultimately converge. Choosing a bias factor will, as predicted, avoid this transient phase. In addition, we achieve better results (tighter clock synchronization) than the flooding time synchronization protocol (FTSP) [30] that is implemented in TinyOS as well.

Finally, once CS-MNS achieves a certain level of clock synchronization, the beacon frequency can be reduced. As CS-MNS adjusts not simply the clock offsets, but also the clock drifts, it will take longer for two clocks to drift apart a pre-determined amount. As described in Reference [31], users or applications can tolerate clock differences up to a specific amount. Using this knowledge can, therefore, allow CS-MNS to adjust the beacon interval to reduce network load and conserve energy.

Acknowledgement

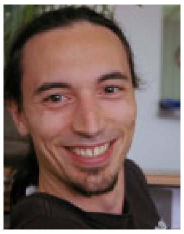
The authors would like to thank the National Sciences and Engineering Research Council of Canada for the financial support that enabled this research.

References

1. Hu Y-C, Perrig A, Johnson D. Packet leashes: a defense against wormhole attacks in wireless networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 3, March–April 2003; 1976–1986.
2. IEEE Std. 802.11. Wireless LAN medium access control (MAC) and physical layer specification. 1999.
3. IEEE Std. 802.15.4-2003. Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs). 2003.
4. Ren F, Lin C, Liu F. Self-correcting time synchronization using reference broadcast in wireless sensor network. *IEEE Wireless Communications* 2008; **15**(4): 79–85.
5. Sundararaman B, Buy U, Kshemkalyani A. Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks* 2005; **3**(3): 281–323.
6. Rentel C, Kunz T. A clock-sampling mutual network time-synchronization algorithm for wireless *ad hoc* networks. In *Proceedings of the Wireless Communications and Networking Conference*, Vol. 1, March 2005; 638–644.
7. Rentel C, Kunz T. Clock-sampling mutual network synchronization for mobile multi-hop wireless *ad hoc* networks. In *Proceedings of the Military Communications Conference*, October 2007; 1–7.
8. Rentel C, Kunz T. A mutual network synchronization method for wireless *ad hoc* and sensor networks. *IEEE Transactions on Mobile Computing* 2008; **7**(5): 633–646.
9. Moreau L. Stability of multiagent systems with time-dependent communication links. *IEEE Transactions on Automatic Control* 2005; **50**(2): 169–182.
10. Gersho A, Karafin B. Mutual synchronization of geographically separated oscillators. *Bell Systems Technical Journal* 1966; **45**: 1689–1904.
11. Sourour E, Nakagawa M. Mutual decentralized synchronization of intervehicle communications. *IEEE Transactions on Vehicular Technology* 1999; **48**(6): 2015–2027.
12. Hu A, Servetto S. Asymptotically optimal time synchronization in dense sensor networks. In *Proceedings of the 2nd ACM Workshop on Sensor Networks and Applications (WSNA)*, 2003; 1–10.
13. Lindsey W, Chen J.-H. Mutual clock synchronization in global digital communication networks. *European Transactions on Telecommunications* 1996; **7**(1): 25–37.
14. Tong F, Akaiwa Y. Theoretical analysis of interbase-station synchronization systems. *IEEE Transactions on Communications* 1998; **46**(5): 590–595.
15. Elson J, Girod L, Estrin D. Fine-grained network time synchronization using reference broadcast. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI'02)*, December 2002; 147–163.
16. Lamport L. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM* 1978; **21**(7): 558–565.
17. Huang L, Lai T. On the scalability of IEEE 802.11 *ad hoc* networks. In *Proceedings of the 2002 Mobihoc Conference*, June 2002; 173–182.
18. Lai T, Zhou D. Efficient and scalable ieee 802.11 *ad-hoc* mode timing synchronization function. In *Proceedings of the 17th International Conference on Advanced Information Networking and Applications*, 2003; 318–323.
19. Sheu J, Chao C, Sun C. A clock synchronization algorithm for multi-hop wireless *ad hoc* networks. In *Proceedings of the 24th IEEE ICDCS Conference*, 2004; 574–581.
20. Zhou D, Lai T. Compatible and scalable clock synchronization protocol in 802.11 *ad hoc* networks. In *Proceedings of the International Conference on Parallel Processing*, 2005; 295–302.
21. Zhou D, Lai T. A scalable and adaptive clock synchronization protocol for IEEE 802.11-based multihop *ad hoc* networks. In *Proceedings of IEEE Conference on Mobile Ad-hoc and Sensor Systems*, 2005; 8–558.
22. Freris N, Kumar P. Fundamental limits on synchronization of affine clocks in networks. *Proceedings of the 46th IEEE Conference on Decision and Control*, December 2007; 921–926.
23. Broch J, Maltz D, Johnson D, Hu Y, Jetcheva J. A performance comparison of multi-hop wireless *ad hoc* network routing protocols. In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, October 1998; 85–97.
24. Haas Z. A new routing protocol for the reconfigurable wireless networks. In *Proceedings of the IEEE 6th International Conference on Universal Personal Communications Record*, Vol. 2, October 1997; 562–566.
25. Yoon J, Liu M, Noble B. Random waypoint considered harmful. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 2, March 2003; 1312–1321.

26. Zhou D, Huang L, Lai T. On the scalability of IEEE 802.11 ad-hoc-mode timing synchronization function. *Wireless Networks* 2008; **14**(4): 479–499.
27. Du X, Chen H-H. Security in wireless sensor networks. *IEEE Wireless Communications* 2008; **15**(4): 60–66.
28. Sun K, Ning P, Wang C. Tinysersync: secure and resilient time synchronization in wireless sensor networks. In *CCS '06: Proceedings of the 13th ACM Conference on Computer and Communications Security*, ACM: New York, NY, USA, 2006; 264–277.
29. Song H, Zhu S, Cao G. Attack-resilient time synchronization for wireless sensor networks. *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*, November 2005; 765–772.
30. Maróti M, Kusy B, Simon G, Lédeczi A. The flooding time synchronization protocol. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, ACM: New York, NY, USA, 2004; 39–49.
31. Ganeriwal S, Tsigkogiannis I, Shim H, Tsiatsis V, Srivastava M, Ganesan D. Estimating clock uncertainty for efficient duty-cycling in sensor networks. *IEEE/ACM Transactions on Networking* 2009; **17**(3): 843–856.

Authors' Biographies



Ereth McKnight-MacNeil received his BSc in Mathematics and Engineering from Queen's University, Kingston, Ontario in 2007. He is currently an MSc student in Systems and Computer Engineering at Carleton University, Ottawa, Ontario, working on clock synchronization protocols for wireless sensor networks.

Carlos H. Rentel received a B.S. degree in Electrical Engineering from Universidad Rafael Urdaneta, Venezuela, an M.S degree in Electrical Engineering from the University of Puerto Rico at Mayaguez, an M.S degree in Electrical and Computer Engineering from the University of Alberta, Canada, and a Ph.D. degree in Electrical Engineering from Carleton University, Canada in 2006. He is currently an Engineer at Creare Inc. Hanover, New Hampshire. He has several years of industrial and engineering research experience in the areas of wireless communications, signal processing, and electronic systems. His research interests include medium access control (MAC) aspects of wireless networks, and applied signal processing. He is a holder of ten patent applications and the author of several peer review papers related to wireless communications and signal processing.

Thomas Kunz received a double honors degree in Computer Science and Business Administration in 1990 and the Dr. Ing. degree in Computer Science in 1994, both from the Technical University of Darmstadt, Federal Republic of Germany. He is currently a Professor in Systems and Computer Engineering at Carleton University. His research interests are primarily in the area of wireless and mobile computing. The main thrust is to facilitate the development of innovative next-generation mobile applications on resource-constraint, hand-held devices, exploring the required network architectures (MANETs, wireless mesh networks, wireless sensor networks), network protocols (routing, Mobile IP, QoS support), and middleware layers. He authored or co-authored close to 150 technical papers, received a number of awards, and is involved in national and international conferences and workshops.