Ultra-wide Bandwidth Timing Networks

S. Niranjayan and Andreas F. Molisch
Wireless Devices and Systems Laboratory, University of Southern California, CA, USA
Email: {niranjay@usc.edu, molisch@usc.edu}

***Abstract*—We present an algorithm for ultra-precise timing in large wireless networks. Our approach uses physical-layer UWB round-trip time-of-flight measurements to achieve precise timing between any two nodes, and fast re-timing based on UWB pulse broadcasting and diversity combining, allowing the precise timing to "propagate" through even large-scale networks. The algorithm also ensures that a virtual timing network evolves from a large set of randomly placed nodes. Simulation results show that the algorithm converges and maintains network timing, regardless of the size of the network.**
***Index Terms*—Network timing, synchronization, ultra-wide band.**

I. INTRODUCTION

Timing, which we define here as the process of aligning the time origins/references (offsets) and the rates (skews) of two physically separate time keeping devices, is a fundamental task in wireless networks. One of its components, namely frequency (clock rate) synchronization, has been well explored and can be achieved by devices using phase-locked loops to adjust their local oscillator frequency to a master clock (base station, GPS satellite, or another device within an ad-hoc network). However, the alignment of time offsets (including propagation delays) is more involved. It can be achieved by many different techniques, motivated by the fact that different degrees of precision are required for different applications.

The aim of our work is to time a large (both in number of nodes and physical dimension) network of randomly distributed wireless nodes with very high precision. Specifically, we wish to achieve timing with an accuracy *much* better than the runtime of the signals between nodes (better than, e.g., 100 ns), in a network with many (tens or hundreds of) nodes. Such a precisely timed network of wireless nodes can be a key component in distributed transmit beam forming, coordinated wireless jamming, precision wireless localization, tracking, navigation, and wireless channel characterization by real-time measurements.

Timing a network of wired/wireless nodes is not a new problem. For example, computers connected to the Internet [8] and cellular phones connected to a base station are examples of networked devices that attempt to keep a common time reference (also [10], [9], and references therein). However, such approaches have limitations in terms of precision or network size. Small networks can broadcast a high-precision reference clock to all nodes in the network (as is done, e.g., in GPS, or the transmission of a clock to cellular users). Many communication systems establish synchronization only within the accuracy of a symbol duration and/or the cyclic prefix duration, which often obviates the need for compensating for the runtime of signals [11]. Narrowband coherent superposition can be achieved by phase synchronization [11], which, however, contains an ambiguity in timing. However, when the expected precision of timing is much smaller than the propagation delays in the system (potentially even smaller than the inverse carrier frequency) and the network is large, the problem requires new solutions.

Precise timing of large, randomly deployed wireless networks is challenging. Each node's timer is derived from an independent oscillator, which is affected by long/short term frequency drifts and jitter [13]. For example, if a timer is left uncorrected, its timing error growth can be modeled by a one dimensional random walk (sometimes modeled as a Wiener process). Even if two oscillators are perfectly matched in frequency and phase at the time instance t_1^*, the timing drift will lead to an error at time t_2^* with a variance $\propto t_2^* - t_1^*$ (t^* denotes absolute time). Mitigating the effect of this jitter, which is critical for essential timing, requires very fast re-timing of the clocks. A second issue is the limitation on the precision of timing information exchange in finite-bandwidth channels. Transmission of information of a 'point' in the time axis requires infinite bandwidth. With finite bandwidth, time of arrival estimation error (TOA) - a measure of timing information - is directly related to the transmission bandwidth (BW) [12]. If a node's TOA estimation error variance is σ_{TOA}^2, it cannot achieve timing accuracy any better than σ_{TOA}^2 (unless some diversity is used). Due to the high TOA estimation accuracy, ultra-wide band (UWB) pulses are desirable for time transfer [1], [2]; we will henceforth also assume this approach.

The above issues affect the timing between any pair of nodes. A significant further complication occurs in large networks with random deployment of nodes. Due to the size of the network, direct transmission of timing pulses from a "master" to all other nodes in the network is not possible. Thus, the timing information has to propagate through the network, possibly aggregating errors. In order to decrease the error aggregation and accumulated jitter we suggest to exploit the node diversity present in the network. Due to the random deployment, the network topology is unknown, and in order to be scalable and be robust to changes, a solution should be distributed. A number of distributed algorithms were proposed for network timing. Reference [3] describes the mathematical foundation for an averaging based distributed consensus algorithm for timing, though neither a fully functional algorithm, nor fast re-timing was within the scope of this paper. A packet based distributed timing algorithm proposed in [4] uses an elegant clock table structure to acquire the timing information from nodes that are multiple hops away. However, due to the assumption of negligible TOA errors, and the sharing of clock tables and packet based time transfer, it is not suitable for the aforementioned target applications. Other examples of packet based time transfer algorithms are [5], [6], however fast re-timing was not within their scope; furthermore [5] neglects propagation delay differences, and rate adjustment is not dealt with in [6]. We do not cite here many other references which are intended for applications that do not require ns level accuracy; such algorithms either explicitly designed for packet based timing, neglects propagation delay (consider only the random transmit delays due to the MAC layer), or correct only either skew or offset (not both).

The main contributions of this paper are 1. A fully functional, completely distributed (in the view of the secondary or slave nodes) algorithm for the propagation of high-precision timing information through a large network. 2. Employing a pulse based physical layer broadcast technique that will work in a distributed fashion[1] for fast re-timing. 3. Nodes harvesting and propagating timing information from and to the network by combining timing information both from multiple neighbors and multiple paths (exploiting the dense multipath properties of UWB). 4. Integrating the strengths of physical layer pulse broadcasting, UWB signals, distributed consensus, and node intelligence in a single set of algorithms. A distributed collision avoidance scheme prevents interference between timing signals from multiple nodes.

II. Problem Definition and System Model

Consider the deployment of a large number (N_S) of wireless sensor nodes ("slaves") with low-precision clocks, such as crystal oscillators. Additionally there are N_M master nodes with high precision clocks (e.g., atomic clocks) that serve as distributed timing references and initiators for key steps in the distributed timing algorithms.

Figure 1 shows the important components of a typical node (masters and slaves alike). Each node has its own oscillator and frequency correction circuit, yielding an adjustable frequency output. Let f_k denote node k's oscillator's nominal frequency. This oscillator output is used to run a timer, which we assume in this paper to be digital, and counts the zero crossings of the oscillator output.[2] At the time (absolute) instance t^*, node k's current value of this local timer is denoted by $t_k(t^*)$. Our goal is to ensure that, at any given time instance t^* (absolute), $E\{(t_k(t^*) - t_j(t^*))^2\} < \sigma_{tol}^2,\ \forall i \neq j$, where σ_{tol}^2 is the timing error tolerance. Local time instances, and local time differences are denoted by t, and τ, respectively. Corresponding values on an absolute time axis are denoted by t^*, and τ^*. For many applications, it is sufficient to time the whole network to some common local reference, i.e., in our case the common local time axis of all the master nodes. Mapping to an absolute reference (if required) can then be achieved simply by mapping the timing of the master nodes to the global reference. To simplify notation, we will henceforth call the time of the master nodes the "absolute" time.

[1]Although pulse based broadcast was considered in [2], its approach of averaging out the propagation delays will require very large node densities for the law of large numbers to take effect.

[2]The counter resets to 0 once it reaches its maximum possible value, t_{max}.

Inside the physical network, a virtual network (namely the *timing virtual network*, TVN) consisting of the nodes, links, and the algorithms that keep timing, is defined. The purpose of TVN is to establish and maintain timing in the network. The network used for data communication can be different, both in the physical layer and in topology. In our proposed solution, the node hardware is built on ultra-wide band transceivers.

Within the scope of this paper the link matrix $\boldsymbol{L}$ defining the connectivity (topology of the network) is defined by (also known as the adjacency matrix in graph theory)

$$\{\boldsymbol{L}\}_{i,j} = 0,\ \ if\ i = j\ or\ \gamma_{i,j} < \gamma_{TH}$$
$$\{\boldsymbol{L}\}_{i,j} = 1,\ \ if\ \gamma_{i,j} \geq \gamma_{TH}$$

where γ_{ij} is the signal-to-noise ratio (SNR) of the link between nodes i and j, γ_{TH} is the SNR threshold for wireless connectivity. This threshold is the minimum SNR that is required for (i) signal acquisition and (ii) meaningful transmission of timing information. The nodes in the network are grouped into sets called tiers, T_i where a node $k \in T_i$ iff there is at least one $l \in T_{i-1}$ such that $\{\boldsymbol{L}\}_{k,l} = 1$. By definition, tier 0 consist of all the master nodes. We assume stationary or quasi stationary channel conditions. Node movements and changes of the propagation channel could be handled easily by our framework, by a subset of the aspects in Phase I, and a node's "personal responsibility", described in Section III.

III. General Description of the Timing Algorithm

The algorithm has two phases. Phase I (Section IV) creates and initializes a TVN among a group of deployed nodes. Phase II (Section V) maintains accurate timing in the network through continuous adjustments.

The network timing algorithm handles the following scenarios. 1. Fresh deployment: Deploying a group of masters and slaves on an area where no prior TVN is established. Establishment of a new timing network. 2. Continuous re-timing: Continuous timing adjustment of a slave after network is established. 3. Slave addition: Adding slaves to an existing sensor network. 4. Master addition: Adding master nodes to an existing network. 5. Group addition: Adding a group of masters and slaves to an existing network. Phase I handles the scenarios 1,3,4 & 5, and becomes important since it creates the TVN. The second scenario is handled by Phase II. Due to space limitations, this paper omits the aspects of Phase I covering 3,4, & 5; they can be found in [7].

In Phase I, each node learns its neighbors (nonzero entries in a row/column of the link matrix, $\mathbf{L}$) and the propagation delays of the associated links. This is achieved through two-way packet exchanges with neighbors motivated by the algorithm of [1] (note that packet transmissions are not used during Phase II). Each node k keeps a parameter, $ClkQ_l$, as a measure of the latest quality of the timing information of its neighbor l. $ClkQ$'s are continuously updated. Node k is initialized when it knows its neighbors and the correct propagation

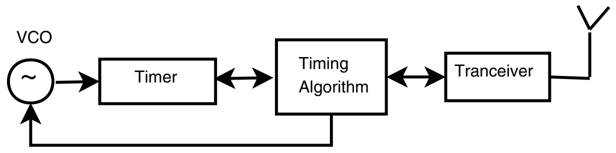

Fig. 1. Components of a node hardware. The timing algorithm interacts with the timer and the oscillator.

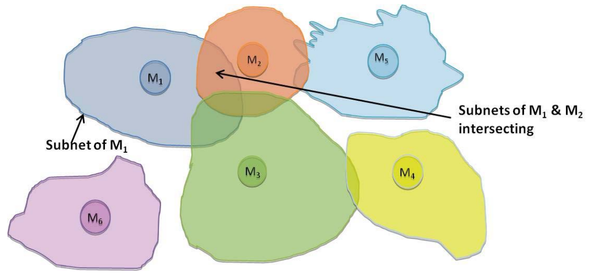

Fig. 2. Master subnetworks. Illustrating the collision between neighboring masters. Some masters are collision are free and can start the TP right after sending SOTPR.

delays. [3] Node k also records its neighbors' current channel signatures, and learns its tier. Since the link matrix and tier structure is not yet known at the beginning of Phase I, all of this is achieved through a distributed algorithm that contains mechanisms to handle collisions during the packet exchanges. We use a suitable modification of Carrier Sense Multiple Access (CSMA). More details are given in Sec. IV.

Once the TVN is formed, the timing information from the masters can be propagated repeatedly through the network. In a naive approach, this would be done in a sequence, one tier transmitting at a time. This would mean each tier has to idle for $(N_T - 1)\tau_B$ (where N_T is the number of tiers and τ_B is the time gap between broadcasts). We propose, however, that the least idle time is achieved by letting odd and even tiers transmit alternatively so that the delay between timing updates for each node is only $2\tau_B$. This approach has the further advantage that it can be implemented in a completely distributed fashion, as knowledge of N_T is not necessary. Nodes use physical layer broadcasting, and *nodes in the same tier can transmit simultaneously*, leveraging the fine delay resolution of UWB signals. In other words, the received signal seen by a node consists of multiple resolvable pulses from both multiple neighbors and multipaths. Since the receiving node knows (from Phase I) the multipath signatures from each neighbor separately, it can now harvest timing information of each resolvable multipath from the aggregate received signal.

The following two sections will describe Phase I and Phase II algorithms in detail.

IV. Timing Algorithm - Fresh deployment (Phase I)

A. Master Starting Timing Procedure with Collision Avoidance

It is assumed that all master are timed to the same reference.

[3]To handle node mobility, each node is also able to detect its own movements (e.g., by an accelerometer), and includes a position flag (PF) in its transmissions to indicate to other nodes whether it moved after it's last initialization. If PF is changed, the node re-initializes itself with its neighbors.

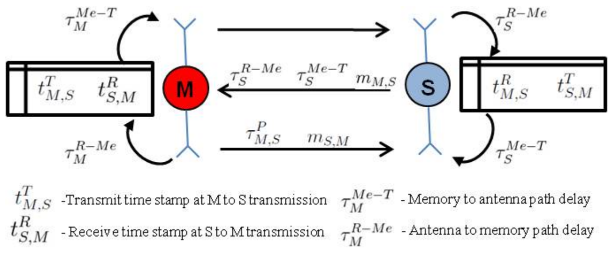
τ_M^{Me-T}
τ_S^{R-Me}
$t_{M,S}^T$ $t_{S,M}^R$
M
τ_S^{R-Me} τ_S^{Me-T} $m_{M,S}$
S
$t_{M,S}^R$ $t_{S,M}^T$
τ_M^{R-Me}
$\tau_{M,S}^P$ $m_{S,M}$
τ_S^{Me-T}
$t_{M,S}^T$ -Transmit time stamp at M to S transmission
τ_M^{Me-T} - Memory to antenna path delay
$t_{S,M}^R$ - Receive time stamp at S to M transmission
τ_M^{R-Me} - Antenna to memory path delay

Fig. 3. Illustration of a two-way timing technique. Time stamps in the node's memory and the device and propagation time delays are shown. $m_{S,M}$ is master's estimate of slave's clock rate.

A.1 Masters simultaneously broadcast a *start of timing procedure request* (SOTPR) signal. SOTPR has sufficient information for the receiver to estimate the master clock rate relative to the receiver clock rate (skew) and the start of a time slot.

A.2 Surrounding slaves that identify the SOTPR signal estimate the SNR of the link between them and the master. If the SNR is larger than, γ_{TH}, the slave does a skew calculation, adjusts its clock, estimates the starting point of a time slot, and accepts the sender as its master. It can furthermore perform synchronous data detection on the master's signal, which contains the master ID and other information.

A.3 If SNR is smaller than γ_{TH}, the slave will continue to listen, in order to identify a set of preferred candidate neighbors to rely on for timing. In this state, a slave will be listening to all the transmissions on the air, which could originate from other masters or slaves).

A.4 Slaves with SNR $> \gamma_{TH}$ that could not identify SOTPR will report collisions between master packets (*complaint*) in the subsequent time slots. This collision reporting is done during a *complaint window* of length N_{col} time slots, after the SOTPR. The position of the complaint signal within the complaint window is determined by random time hopping in this way a master will not only learn of the existence of collisions, but will have a rough estimate of their number.

A.5 If the number of complaints in the complaining period $C \leq C_{TH}$ (collision threshold), the master sends *start of timing procedure* message (SOTP) in the next time slot.

A.6 Slaves that synchronized themselves to a master (w.r.t rate and time slot alignment) now broadcast a SOTPBcast message in the subsequent N_{bro} time slots using a random time hopping technique similar to the one used for complaining. Masters that backed off use this message to learn that their competing master has started timing procedure.

A.7 If $C > C_{TH}$, the master waits for a random backoff of $1 + N_{bro} + (2 + N_{col} + N_{bro})b$ time slots, where b a random integer $(0 \leq b \leq b_{max})$, and retransmits SOTPR if no SOTPBcast message is detected during the waiting period.

A.8 If SOTPBcast message is detected during the waiting period in A.7, master pauses its attempt to start a timing procedure and goes to *pause* mode (although it is not described here, algorithm facilitates recovery from *pause* mode and beginning timing procedure).

A.9 After transmitting SOTP, the master waits N_{bro} time slots and enters its timing procedure (TP). This is described in IV-B.

B. Master Timing Procedure (TP)

B.1 Master saves its current time stamp, $t_{M,S}^{T}$, (Fig.3) and broadcasts a SYNCH message carrying sufficient information for extracting it's clock rate (e.g., a sequence of UWB pulses at predetermined spacing).

B.2 A slave receiving SYNCH extracts the master clock rate and corrects its own clock accordingly. It further estimates the time of arrival (TOA) of the SYNCH message and creates a time stamp $t_{M,S}^{R}$.

B.3 All the receiving slaves, using a carrier sense multiple access collision avoidance (CSMA/CA) strategy, reply to the master with a SYNCH_R message, which contains τ_S^{R-Me}, τ_S^{Me-T}, and time stamp, $t_{M,S}^{R}$[4].

B.4 Upon receiving a valid SYNCH_R message, master calculates the propagation delay τ_p of the master-slave link, and the timing offset (τ_e) of the slave using its own $\tau_M^{R\rightarrow Me}$, $\tau_M^{Me\rightarrow T}$, and the TOA estimate of the SYNCH_R message, $t_{S,M}^{R}$.

B.5 Master sends τ_p and τ_e to the slave.

B.6 All slaves that can overhear the transmission in B.3 and B.5, use the information for their self learning of their neighborhood. *This can dramatically increase the efficiency of the protocol, as it provides to the slave nodes information about their neighbors' positions (delays) without requiring additional packet exchanges.*

B.7 Masters allow N_{TP} time slots for the slaves to complete timing procedure through CSMA. After N_{TP} time slots, the master broadcasts another message (MTR) querying whether any node still requires timing procedure.

B.8 Slaves still requiring timing procedure will reply to MTR with a request for timing procedure (RTP). RTP uses similar time hopping technique used for transmitting complaints.

B.9 If at least one RTP message is detected by the master, it re-enters timing procedure (B.1-B.9).

B.10 If no RTP message is detected, master ends its timing procedure by broadcasting end of timing procedure message (EOTP). Assigns broadcast order for slaves (time hopping code).

B.11 Slaves distributedly broadcast EOTP in a EOTPBcast message composed similar to the SOTPBcast message in A.6, but with predetermined time hops. This ends a master's timing procedure.

At the end of the procedures A and B, the following is achieved: 1. Masters know their slaves, and conflicting masters 2. Slaves know their candidate master/s and some neighboring slaves 3. The propagation delays for all (not-strictly) master-slave links and some slave-slave links (slave-slave propagation delays can be estimated through triangulation or slave-slave indirect timing) are known by the nodes sharing the link. Note that the slaves' operational mode during A and B is named as *Dinit* mode.

C. Slave Self Timing

After the completion of the initial round (A & B), there will be slaves that were left out (unsuccessful in doing a TP), either not being in the coverage of a master or failed to do TP due to some unresolved collisions. These left-out slaves (which make up a large portion of a practical network) are responsible for joining a TVN and pulling the timing information from the network (pulling of information is a key aspect of a self evolving network as opposed to centralized pushing of the timing information).

C.1 Left-out slaves move to self timing mode (SelfDinit) after a wait period (T_{Dinit}).

C.2 In both Dinit and SelfDinit modes, slaves continuously listen to transmissions on the air to learn their neighbors (CandList).

C.3 Slave sends a request for a timing procedure (RLTP) to the best candidate in it's list (ranked by the $ClkQ$s). This step is repeated with random backoffs until it is successful.

C.3 A slave receiving an RLTP replies with a timing information message (TI) if it has already joined the TVN through at least one link.

C.4 The requesting slave calculates the link propagation delay, τ_p and shares it with the helping slave. Again, all slaves that can overhear this message will utilize it to, increase the knowledge of the neighborhood and enable computation of the delays to neighbor nodes.

C.5 A slave repeats this process until it times the links with all the nodes in its CandList. Note that at the end of Phase I, the candidate list should be consistent with the nonzero entries of the link matrix associated with a particular node.

As the result of A, B, and C a TVN will evolve, each node will have a list of it's neighbors and estimates of the propagation delays for the network links. Also, the nodes in the TVN have achieved coarse timing which is necessary for the operation of the next, and the most important, stage of the timing algorithm.

V. Timing Algorithm - Continuous Adjustment (Phase II)

D.1 Before beginning the continuous adjustment algorithm (named as the *Blink* algorithm), the nodes should learn their tiers. This tier-learning algorithm is not explained here due to space limitation.

D.2 Masters initiate the *Blink* cycles by simultaneously broadcasting short duration timing pulses (UWB pulses), this process is named as *Blinking*. The Blink signal is, in

[4]The delay between, the radio signal arriving at the antenna and the latching of the time stamp, $t_{M,S}^{R}$, in the memory is $\tau_S^{R\rightarrow Me}$. The delay between the latching of the time stamp, $t_{S,M}^{T}$, in the memory and the radio wave actually leaving the antenna is $\tau_S^{Me\rightarrow T}$.

the simplest case, two short pulses transmitted at pre-determined time gap, allowing the receiving node to extract the rate and timing offset of the transmitting node. Upon receiving this *Blink* signal (distinguishable from usual packet transmissions) from tier 0, the nodes in tier 1 do a *Blinking* after a constant time delay τ_B. This process will continue towards the last tier, $N_T - 1$. Once started, the tiers repeat blinking every $2\tau_B$ time. Continuing this creates a pattern of alternating *Blinking* in the network, where odd and even tiers *Blink* out of phase and every $2\tau_B$ time.

D.3 Receiving nodes make estimates of their neighbors' time of arrivals relative to their own local clocks, $\{t_k^e\}_{k=1}^{N_{nei}}$ where N_{nei} is the number of transmitting neighbors.

D.4 Using its links' propagation delays (learned in Phase I), a receiving node estimates the expected arrival times of its neighbors' *Blink* signals as $\{\tilde{t}_k^e\}_{k=1}^{N_{nei}}$. Comparing this with the actual arrival time estimates, $\{t_k^e\}_{k=1}^{N_{nei}}$, a node calculates its time reference offsets from its neighbors as $\{t_k^e - \tilde{t}_k^e\}_{k=1}^{N_{nei}}$. The receiver adjusts its timer by some function of the computed offsets; in the simplest case by a linear combination, $\sum_{k=1}^{N_{nei}} w_k \left(t_k^e - \tilde{t}_k^e\right)$, where w_k are some real numbers [5].

D.5 This *Blinking* procedure is continued as needed.

Note that since we assumed half-duplex nodes, during the *Blink* procedure, tier i nodes hear only from tiers $i+1$ and $i-1$.

VI. Simulation and Numerical Results

The timing algorithm described in Sections IV and V was implemented in a network simulator using object oriented design in Matlab and C^{++} to perform close-to-system level simulations. The simulator places N_M masters and N_S slaves on a two-dimensional (2D) surface according to some spatial distribution model (Poisson, uniform, etc.), runs algorithms A-C to form a TVN, and runs algorithm D to do continuous re-timing. The physical layer (radio interface and the channel) is abstracted, such that only time of arrival estimation error (σ_{TOA}), clock jitter (σ_J) and receive SNR levels (γ_{ij}, γ_{TH}) are used to characterize the physical layer. Hence it is assumed that the multipath diversity due to UWB signaling is also absorbed into σ_{TOA}.

Note that no units for linear dimensions and time are necessary for the simulations, since the results are scalable. However, to give a practical insight, following parameter settings are done. Simulation results shown in Figs. 4-6 are for $N_M = 5$ and $N_S = 5000$, a 2D Poission spatial spread model was assumed with an average node density, $\lambda = 0.0013m^{-2}$. Neglecting fading effects, we assume that each transmitter has a *radius of coverage* (within which the SNR exceeds γ_{TH}) of $50m$ that results in an average of 10 neighbors per node. With 5000 slaves the network's radius of coverage is approximately $1.118km$. The following values were assumed, $\sigma_{TOA} = 1ns$, $\sigma_J = 7.74 \times 10^{-9} s/\sqrt{s}$ (assuming a Wiener process model),

[5]In the examples presented in Section VI, $w_k = 1/N_{nei}$.

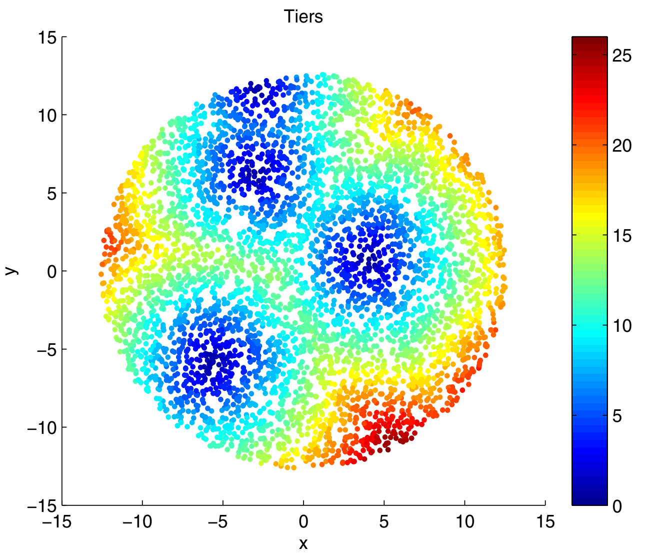

Fig. 4. Tiers of the TVN. This network with $N_m = 5$ and $N_s = 5000$ with 10 neighbors per node has 26 tiers (colored figure).

and $2\tau_B = 60\mu s$. Relative slave clock frequencies are assumed to always have some error and modeled by a normal random variable with mean 1 and a standard deviation 10^{-6}. Figure 4 shows the tiers in the TVN. This network, with a number of 5000 nodes with 5 masters, and the above defined parameters has 26 tiers in this realization. It is observed that the blink algorithm converges [6] and gives nearly the same steady state performance regardless of the number of tiers in the TVN. However, the time to reach the steady state increases with number of tiers, i.e., nodes in the outer most tiers have to wait longer to achieve timing, where tier 1 will achieve it in few cycles of the blink protocol. Figure 5 shows the variation of the root-mean-square-error (rmse) of the timing offsets in each tier, calculated as $rmse_i = \sqrt{\sum_{n \in T_i} \epsilon_n^2}$, where ϵ_n is the timing offset of the node n w.r.t the master time axis. In the middle tiers, $rmse_i \approx 0.65ns$ which is lowered from $\sqrt{\sigma_{TOA}^2 + \sigma_J^2 * 2\tau_B}$ due to the diversity in the neighbor hood. However, full diversity from all the neighbors ($\approx$ 10) will not be available since the diversity from the neighbors in the node's own tier is not useful. The value of $rmse_1$ is lower than those in the middle tiers since the 1^{st} tier is directly connected to the masters. The $rmse_i$ at the outer most tiers should be higher than the middle tiers, since those tiers have less diversity available. However, this effect is not observed in the plots. One should note that the outer most tiers only have a small number of nodes and hence make it difficult to calculate a reliable statistical ensemble average. In fig. 6 a snapshot of the timing errors of the nodes during the steady state is shown. A comparison of tier wise $rmse$ timing offsets for average number of neighbors 10 and 50 is shown in Fig. 7 with $Ns = 1000$ (all other parameters remain unchanged). The improvement in timing performance with node density is demonstrated.

[6]Theoretical analysis of the blink algorithm, proof of convergence, and further insights are in [7].

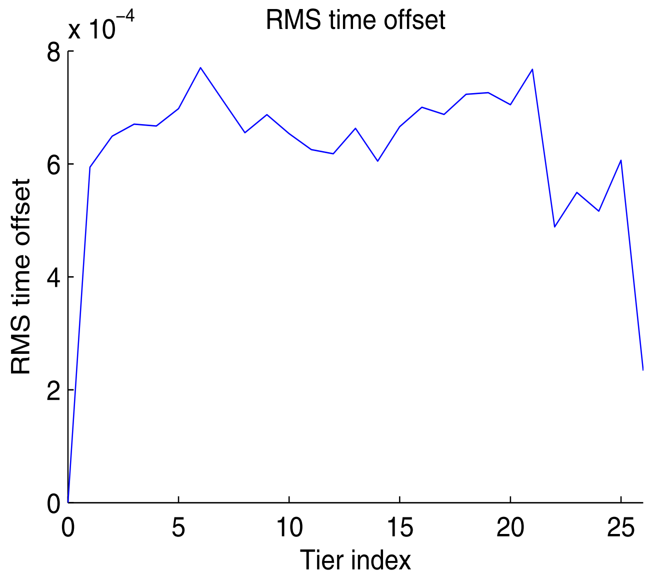

Fig. 5. Variation of RMS time offsets along the tiers in μs. In the middle tiers, the RMS offset is closer to $0.65ns$ regardless of the tier index.

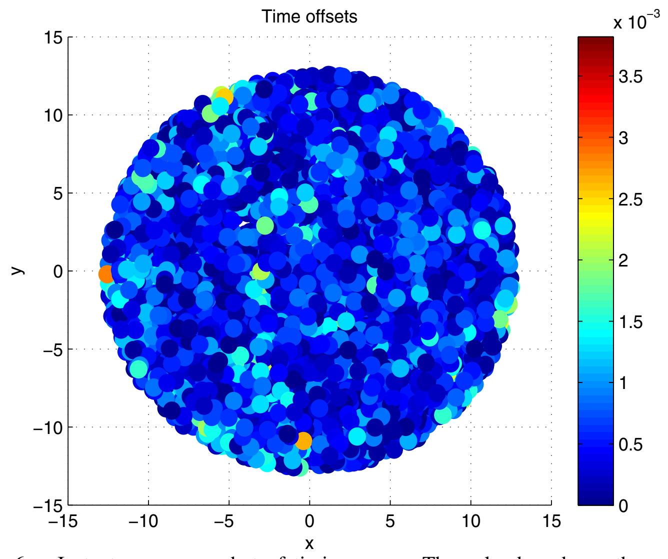

Fig. 6. Instantaneous snapshot of timing errors. The color bar shows the instantaneous absolute time errors in μs (colored figure).

The initialization algorithm was able to create the TVN regardless of the size of the network. It also provides initial timing for almost all nodes connected to the TVN with no dependency on the size of the network. Additional results and interpretations can be found in [7].

VII. Conclusion

We presented a network timing algorithm for large wireless networks. The proposed algorithm creates a timing virtual network from a set of randomly deployed nodes. The algorithm learns the neighbors, their link propagation delays, channel signatures, position flags, and clock qualities. Using this information, the algorithm performs a physical layer fast pulse broadcasting to achieve fast and accurate re-timing. In the presented example, timing is achieved through averaging-based consensus. Simulation results show the algorithm provides accuracy that is essentially independent of the number

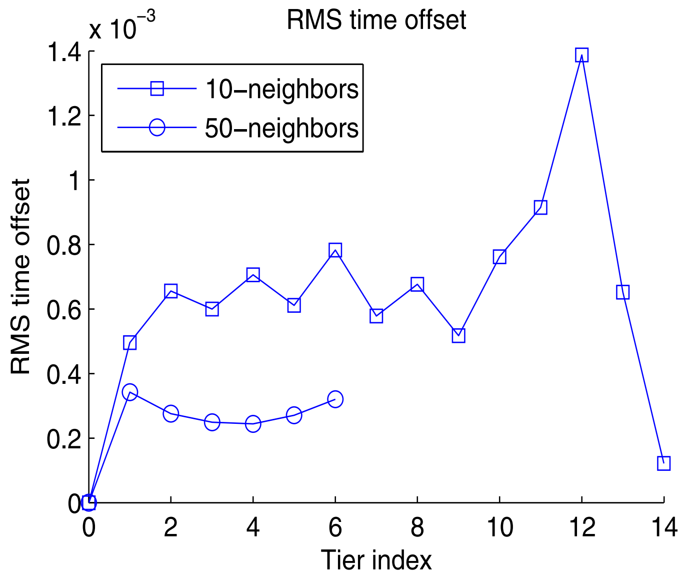

Fig. 7. Variation of RMS time offsets along the tiers in μs for $Ns = 1000$ with 10 and 50 average neighbors per node. Performance improved with node density.

of nodes in the network, and thus provides outstanding scalability. Given the possible accuracy of TOA determination in UWB networks, nanosecond accuracy timing in networks with thousands of nodes become feasible.

Acknowledgment: Authors thank Ahmad Golmohammadi for his help with programming.

References

[1] C.-C. Chui and R. Scholtz, "Time transfer in impulse radio networks," *Communications, IEEE Transactions on*, vol. 57, no. 9, pp. 2771–2781, september 2009.
[2] A.-S. Hu and S. Servetto, "On the scalability of cooperative time synchronization in pulse-connected networks," *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2725–2748, june 2006.
[3] A. Giridhar and P. Kumar, "Distributed clock synchronization over wireless networks: Algorithms and analysis," in *Decision and Control, 2006 45th IEEE Conference on*, dec. 2006, pp. 4915–4920.
[4] B. J. Choi, H. Liang, X. Shen, and W. Zhuang, "Dcs: Distributed asynchronous clock synchronization in delay tolerant networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 3, pp. 491–504, march 2012.
[5] L. Schenato and G. Gamba, "A distributed consensus protocol for clock synchronization in wireless sensor network," in *Decision and Control, 2007 46th IEEE Conference on*, dec. 2007, pp. 2289–2294.
[6] M. Leng and Y.-C. Wu, "Distributed clock synchronization for wireless sensor networks using belief propagation," *Signal Processing, IEEE Transactions on*, vol. 59, no. 11, pp. 5404–5414, nov. 2011.
[7] S. Niranjayan and A. F. Molisch, "An algorithm for high-precision timing of large-scale wireless networks", *to be submitted*, 2012.
[8] Network Time Protocol project, http://www.ntp.org/, 2012.
[9] P. Ranganathan and K. Nygrad, "Time synchronization in wireless sensor networks: A survey," *Int. Journal of UbiComp*, vol. 1, no. 2, pp. 92–102, july 2010.
[10] W. Lindsey, F. Ghazvinian, W. Hagmann, and K. Dessouky, "Network synchronization," *Proceedings of the IEEE*, vol. 73, no. 10, pp. 1445–1467, oct. 1985.
[11] A. F. Molisch, *Wireless communications*, second ed., Chichester, West Sussex, England: John Wiley & Sons Ltd., 2011.
[12] Y. Shen and M. Z. Win, Fundamental Limits of Wideband Localization Part I: A General Framework, *Information Theory, IEEE Transactions on*, vol.56, no. 10, oct. 2010, pp. 4956–4980.
[13] J. Esterline, Oscillator phase noise: theory vs practicality, *Greenray industries, inc. report*, Apr. 2008.