Extensions to the IEEE 802.11 TSF for Efficient and Reliable Network Synchronization in Large Scale MANETs

P. Rauschert, A. Honarbacht, and A. Kummert
University of Wuppertal, Germany
Communication Theory
{rauschert|honarbacht|kummert}@uni-wuppertal.de

***Abstract*— Designing new protocols for Mobile Ad hoc Networks (MANETs) is a great challenge due to their distributed and self organized nature. Though, aspects of approved algorithms for hierarchical topographies may be carried over to these flat networks. The IEEE 802.11 protocol supports ad hoc networks in small scale applications, but its performance in large scale environments is still under investigation. Besides the Distributed Coordination Function (DCF), the Timer Synchronization Function (TSF) can be significantly improved in order to increase the performance in large scale multihop networks. This paper presents systematic extensions to the TSF that allow increasing the overall reliability and disburdening the network load at the same time. The presented scheme may be tailored to specific applications and even supports mobile stations and herewith MANETs.**

I. Introduction

Classical wireless broadband networks are hierarchically organized, e.g. in mobile communications using the Universal Mobile Telecommunication System (UMTS), or Wireless Local Area Networks (WLANs). In these fixed backbone networks, all members connect to a superordinate base station, which is placed at a well selected position that has been planned carefully in order to optimize cell coverage, traffic aspects, and other network constraints.

Networks that base on a fixed backbone, work stable and communicate in a very efficient manner. One reason for this reliable communication is the rigid organization that is also a major drawback in changing environments, since the network organization may not be adapted dynamically. Here, self organizing networks give several advantages over hierarchical networks, but also face designers with new problems that must be solved by appropriate protocols [1]. The structure is flat and a reliable communication structure over multiple hops must be established initially. An even greater challenge are Mobile Ad hoc Networks (MANETs) with moving members that may cause a lot of topology changes as they move around. In order to ensure reliable operation, all protocols in ad hoc networks must be designed carefully in a fully distributed fashion. This is a great challenge, but also enables various new fields of applications, since there are manifold applications, where the coverage of a disaster area with destroyed infrastructure is only one example.

The IEEE 802.11 [2] is the de-facto standard for WLANs nowadays. This standard has been proven by several applications to work well in WLANs, where each station connects directly to an adjacent base station. An interesting aspect is the basic support for an ad hoc mode, which allows a station without direct neighborhood of a base station to connect to adjacent neighbors in order to reach remote base stations. This operation mode is called Independent Basic Service Set (IBSS) and is restricted by a maximum distance of just a single hop between adjacent stations. This limitation to one hop is a significant restriction, since a higher number of hops would extend the area of coverage considerably.

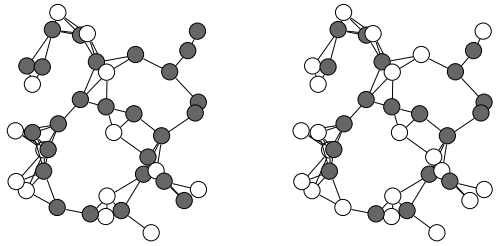
The performance of the IEEE 802.11, when being applied to larger scaled and multihop networks, is of great interest. Here, the Distributed Coordination Function (DCF) and specific problems of ad hoc networks, like the hidden and exposed terminal problem, are common reasons for unfairness [3]. In high density networks, further problems may arise in the Timer Synchronization Function (TSF) [4]. With respect to the TSF there are two major drawbacks in multihop scenarios: packet collisions and network utilization. This paper introduces a scheme that has been initially derived from the TSF, but has been well tailored in order to improve the applicability on ad hoc networks where stations may even move dynamically in a random way and cause topology changes.

II. Connected Dominating Sets

MANETs are flat initially. All stations are equal with the same rights and no priority is predefined. This gives the network a maximum of flexibility to adapt to changing operating conditions. Hierarchical topographies are the most common solutions nowadays and reliable protocols are available. Basic ideas from approved algorithms may be carried over to ad hoc systems, if a well adapted priority scheme can be introduced.

An ad hoc network can be formally represented by a graph that can be described mathematically by $G(V, E)$, where V denotes a set of vertices and E denotes all edges (u, v) and (v, u) between two bidirectionally connected vertices $u, v \in V$ when station u and v are in communication range [5]. V can be split into two distinct subsets of vertices. A dominating set S is defined as a subset of V, where each node in $(V - S)$

is adjacent to at least one member of S. Elements of S are called to be dominators over the absorbers that are elements of $V - S$. A connected subgraph of S is called a Connected Dominating Set (CDS) and denoted by C.



(a) CDS (b) MCDS

Fig. 1. Connected dominating set and CDS with minimum cardinality

A Minimum Connected Dominating Set (MCDS) is a CDS with minimum cardinality. Figure 1 shows two examples for (M)CDSs for an identical flat topology. Constructing an MCDS requires global network knowledge and computing its solution is NP-hard. Constructing an MCDS is an optimal solution, but its implementation into communication systems is not advisable due to the complexity. A suited solution is the utilization of CDSs, since they can be calculated in a fully distributed fashion, their calculation is P-hard, and stations just need local knowledge of the topography. Here, various algorithms are available [5] [6] while their selection depends on boundary conditions that are beyond the scope of this paper.

III. Time and events in distributed systems

In fully distributed and autonomous digital systems, stations usually use oscillator vibrations for time measurement. Members of a closed system may use an own definition of time that bases on a common model. As long as all members use the same definition this works well and allows synchronized operation as needed for object tracking or scheduled channel access like with Time Division Multiple Access (TDMA).

Naturally, humans denote the world time by t in seconds. This gives a fixed reference when describing time in technical systems. Digital systems have a physical oscillator that has a fixed frequency. In ideal systems it would be sufficient to count the vibrations and divide the sum by the frequency constant to get the time progress in seconds. But due to inaccuracies, caused by outside influences and physical variations, the progress of two oscillators is never identical and therefore they drift away from each other. This drift is the main reason why there is the need for network synchronization in networks consisting of several autonomous members.

Time in systems may be described on the base of world time t as follows. Let's consider a network with N stations. The physical time $p_i(t)$ of one station $i \in N$ is the discrete function that represents the local time and is measured by oscillator vibrations. Due to the oscillator inaccuracies, all N stations have a varying progress of $p_i(t)$ and should use an artificial model to balance these differences. This balanced model is denoted by $v_i(t)$ for each station i and describes the progress of virtual time that should be common to all stations in the ideal synchronized case.

On basis of this representation of time, discrete events can be described. Since synchronization is usually implemented by transmitting and receiving synchronization packets (so called beacons), there is one event when beacons are generated and transmitted by a station k_1. The event when station k_1 generates its nth beacon is denoted by $T_B(k_1, n)$ and releases further events when being received by adjacent stations where k_2 denotes one of them. Herefrom we call $T_U(k_2, m)$ to be the event when k_2 receives its mth beacon frame from adjacent stations. In this case, with $T_B(k_1, n)$ releasing $T_U(k_2, m)$, the times of these events in t only differ by propagation delay.

IV. Timer Synchronization Function

In infrastructure IEEE 802.11 networks, a base station k periodically sends out beacon frames, carrying the local time $p_k(t)$, to $(N - 1)$ adjacent stations within its transmission range. When a station $i \in (N - 1)$ receives a beacon frame, it compares the transmitted timestamp to its local virtual time $v_i(t)$. If these times are different, the station adopts $v_i(t)$ to the received timestamp that is denoted by $v_k(T_B(k, n))$ and represents the time of the base station when generating the beacon frame.

For IEEE 802.11 IBSS networks, not all stations may be in transmission range of the base station and therefore the Timer Synchronization Function (TSF) is used that follows a fully distributed scheme. Here, each station maintains a periodic timer in order to schedule the Target Beacon Transmission Time (TBTT). For each TBTT, stations follow the given scheme:

a) Suspend decrementing the back-off timer for any pending non-beacon or non-ad hoc traffic indication transmission.
b) Calculate a random delay that is uniformly distributed in the range between zero and twice aCWmin × aSlotTime (which are station constants).
c) Wait for the period of the random delay while decrementing the random delay timer.
d) Cancel the remaining random delay and the pending beacon transmission if a beacon arrives before the random delay timer has expired.
e) Send a beacon if the random delay has expired and no beacon has arrived during the delay period.

This algorithm results in a scheme, where all stations compete uniformly for beacon generation each TBTT. Figure 2 shows a two station example for distributed beacon that follows the described algorithm.

If one station k won this competition, it accesses the shared medium and transmits its virtual time that is denoted by $v_k(T_B(k, n))$, by means of a timestamp within the beacon frame. All adjacent neighbor stations i receive this beacon and compare it to their local TSF time

$$v_i(t) = p_i(t) + \Delta_i \tag{1}$$

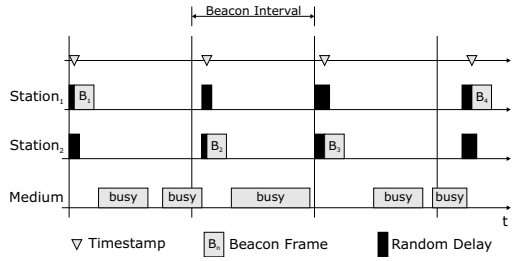


Fig. 2. Timer Synchronization Function

where Δ_i alway carries the latest station-to-station asynchronism. If the timestamp is greater than the virtual time the offset is corrected until a further update carries a timestamp that is again later than the time represented by $v_i(t)$. Adopted timestamps are always greater than the TSF counter and therefore the offset between these two values is positive in all cases. The offset Δ_i between station k and i after processing the nth beacon is

$$\Delta t_i = v_k(T_B(k,n)) - p_i(T_U(k,n)). \tag{2}$$

The performance of the IEEE 802.11 TSF has been analyzed in a MANET environment [7]. It was found that it allows to synchronize all connected stations within an accuracy of $30\mu s$ to the median node for the example scenarios with up to 200 stations over an area of $4000m \times 4000m$. The TSF allows to synchronize MANETs, but its efficiency still lacks in performance. First, typical problems like the hidden/exposed terminal problem will counteract periodic beacon generation. For this reason, beacons get lost and offset compensations will not be done periodically. Therefore, the offset increases with each lost update and stations will drift further away from each other [4]. Another important aspect is the high number of nodes that are trying to submit simultaneously onto a shared medium. Here, a high number of collisions appear which also counteract reliable beacon transmission. This effect even reduces the lifetime of stations dramatically, since each transmission needs energy and energy efficiency is an essential aspect in autonomous wireless systems that are powered by batteries with a limited capacity.

V. Predictive Timer Synchronization

In order to improve the reliability we propose a novel approach - the Predictive Timer Synchronization Function (PTSF). The PTSF bases on a modified scheme of distributed beacon generation like the TSF, but it allows to enlarge the beacon interval significantly. The PTSF compensates not only the offset between stations periodically, but even allows to equalize the clock drifts between two updates. This results in a more reliable algorithm where even missed beacon frames can be interpolated and therefore further driftage is reduced significantly.

The PTSF uses dedicated parameters at each station i to provide a drift compensated model in $v_i(t)$. Each update allows to adapt parameters that are necessary for this model.

TABLE I

An example for $\mathbf{d_i(k)}$ with 4 entries

Address	0x..10002	0x..10023	0x..10147	0x..15003
$p_i(T_U(k,m))$	101	105	87	95
$v_k(T_B(k,m))$	107	113	91	101
$p_k(T_U(j,n))$	78	0	78	72
Lifetime	5	98	87	34

These parameters are always kept up-to-date when receiving a beacon whose timestamp is greater than the actual virtual time. Whenever an update happens, it is necessary to proceed through the following steps.

Let's assume that station i receives the mth beacon from station k while the last valid update from the same source has been the nth beacon. First, the local physical time $p_i(T_U(k,m))$ is stored. Furthermore, the actual offset is measured by calculating the difference between beacon timestamp and physical time

$$\Delta t_i = v_k(T_B(k,m)) - p_i(T_U(k,m)). \tag{3}$$

The PTSF uses beacons with an additional 64bit timestamp that represents the last physical time when the sender itself has been updated by another station j. This trailer is denoted for instance by $p_k(T_U(j,n))$. By means of this timestamp, all stations maintain a list with one station vector for each neighbor. This station vector contains the following information

$$\mathbf{d_i(k)} = \begin{pmatrix} p_i(T_U(k,m)) \\ v_k(T_B(k,m)) \\ p_k(T_U(j,n)) \end{pmatrix} \tag{4}$$

and several of these station vectors are combined to a station matrix. An example for such a matrix with 4 single station vectors is given in table I.

In very large scaled networks, memory utilization for a matrix build by several $\mathbf{d_i(k)}$ may be quite extensive. Therefore, a lifetime is used to remove each entry after expiration. Herefrom, the matrix memory may be reused in order to observe more faster stations than available entries by sequential memory reuse.

The station vector that corresponds to the actually received beacon, is updated right after adjusting slope a_i. Slope a_i is set to 1.0 after powering. Afterwards, it is adjusted according to the following algorithm. First, an unknown node (station vector $k \notin$ station matrix) is sending a timestamp that is greater than the virtual time. Here, only the offset is compensated (3) but a station vector is generated and inserted into the list and therefore known in the future. If receiving further timestamps from station k that are still greater than $v_i(t)$ there are two possibilities. First, the received trailer may be unequal to the one represented by the entry of the corresponding station vector. In this case, the transmitting station has been updated in the meanwhile and asynchronism is not caused by the drifting local oscillators only. In the case that the trailers are identical, station s has not been updated in the meanwhile and the drift can be compensated by adjusting the slope

$$a_i = \frac{v_k(T_B(k,m)) - v_k(T_B(k,n))}{p_i(T_U(k,m)) - p_i(T_U(k,n))}. \quad (5)$$

After obtaining a valid update and following the shown procedure the PTSF virtual time is defined by

$$v_i(t) = p_i(T_U(k,m)) + \Delta t_i + a_i\big[p_i(t) - p_i(T_U(k,m))\big] \quad (6)$$

until a beacon timestamp has been received which is later than $v_i(t)$.

For clarification, figure 3 shows an example for the operation of PTSF where station 1 synchronizes to stations 2-4. It can be seen, that $v_1(t)$ adapts to $v_4(t)$ after receiving two consecutive beacons from station 4.

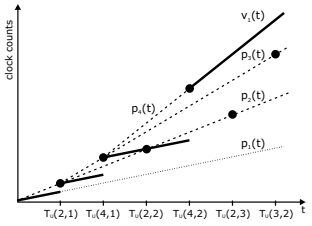


Fig. 3. Times when using PTSF synchronization

VI. PTSF Extension for CDSs

The PTSF that has been proposed in section V, allows to enlarge the beacon interval and to interpolate $v_i(t)$ between two updates. This allows to solve a major drawback of the TSF, since its performance relies on short and periodical beacon periods.

Using CDSs is helpful and enables efficient routing, broadcasting, and other network services like in infrastructure networks. By means of one dominator over several absorbers, the efficiency can be raised significantly. Using a CDS is even more efficient when it is used for several protocols, since the traffic overhead for CDS construction is more than compensated by saving a multiple of overhead with efficient hierarchical protocols. Therefore, a modification of the PTSF is proposed that supports an efficient operation and eliminates the drawback of the PTSF in high density scenarios.

Generally, CDSs introduce a twofold priority scheme that classifies all stations V into the set of dominators S and absorbers $(V-S)$. In CDSs, all dominators are connected as a network backbone and information may be distributed globally using this infrastructure build by vertices in C. In order to avoid packet collisions, only stations belonging to C are allowed to schedule the TBTT in a periodic manner. Vertices belonging to $(V-C)$ are usually only listeners. The algorithm of beacon reception that has been proposed in section V, is slightly modified. When a vertex that belongs to $(V-C)$ receives a timestamp $T_B(k,n)$, that is later than $v_i(T_U(k,n))$, it proceeds through the procedure as shown before. The initial scheme is only modified when receiving a timestamp that is earlier than the local virtual time. In this case, station i schedules a single TBTT in order to update the transmitter. Hereupon, the dominator receives this update and adapts to it. Once that the dominator as a clusterhead has adapted to the fastest absorber in its cluster, it will only be updated by other dominators that also supervise other absorbers with possibly a faster clock drift.

This is just a slight modification to the original scheme and requires only minor changes to the initial algorithm. Therefore, it may be easily integrated in a seamless manner when being useful in order to prolong battery lifetime or reduce the number of packet collisions.

VII. Integration into 802.11 MAC

The IEEE 802.11 TSF is a MAC sublayer management entity. In order to replace the TSF by the proposed PTSF, some minor changes to the MAC layer must be done. The PTSF introduces a prediction algorithm that bases on clock stability of adjacent transmitters. The stability of a transmitter k is evaluated by observing the last remote update that is denoted by $p_k(T_U(j,n))$ as explained in section V. In order to transmit this timestamp, the beacon frame format has to be changed by insertion of an additional 64bit field. The modified format of a PTSF beacon frame and the size of its single entities is shown in figure 4.

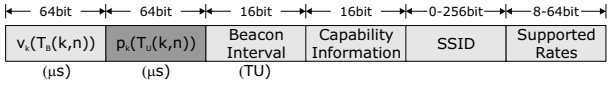


Fig. 4. Beacon Frame Format for PTSF

This is the only modification to all of the standardized 802.11 packet frames and enables to replace the TSF by the PTSF.

In order to utilize a CDS for PTSF as shown in section VI, further changes to the MAC layer must be done. The construction of CDSs relies on neighborhood information. Here, the Neighborhood Discovery Protocol (NDP) [8] may be used that integrates into the MAC Layer Management Entity (MLME) and gathers topography information in the scope of a predefined n-hop distance around each node. An example of 3-hop neighborhood is shown in figure 5, where the source vertex is colored in light gray and the hop counts are marked by the edges.

An NDP client gets two indications, `NDP-NEWCONNECTIONS.indication` and `NDP-LOSTCONNECTIONS.indication`. These NDP MLME indications allow to maintain a neighborhood table. Since neighborhood information may be reused for several protocols, the number of hops that are used for NDP depends on the client that requires maximum neighborhood resolution. In order to construct a CDS for instance, the required resolution depends on the CDS algorithm itself. For instance, the authors of [9] propose a marking and pruning (restricted k-dominant pruning) process that demands 2-hop neighborhood information. Table II shows an example

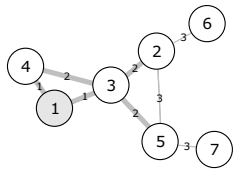


Fig. 5. 3-hop NDP discovery

Station	Neighbor	Hops
1	1	0
1	2	2
1	3	1
1	4	1
1	5	2
2	2	2
3	1	2
3	2	2
3	3	1
3	4	2
3	5	2
4	1	2
4	3	2
4	4	1
5	5	2

TABLE II. 2-hop neighborhood table

where the 3-hop neighborhood information from figure 5, as might be needed for Radio Resource Management (RRM) based on Transaction-Based Soft-Decision RRM (TBSD-RRM), is reused by the CDS algorithm in order to build a 2-hop neighborhood table. Apart from TBSD-RRM, the CDS protocol is a further client of NDP and gets the same indications. All the negligible data that is indicated by distances greater than 2-hops, can be ignored. Based on this table, the CDS MLME operates and informs clients by a `CDS-MARKERCHANGE.indication`. The PTSF is registered as a CDS client and gets this indication. In combination with the slight changing of the beacon frame format, this allows to proceed according to the algorithm that has been proposed in section VI.

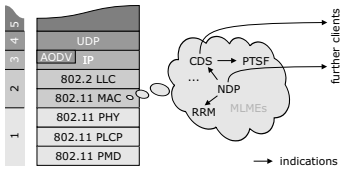


Fig. 6. Protocol stack with proposed MAC extensions

Figure 6 shows an example for a suited protocol stack at layers 1-5. At layer 2, the described insertion of NDP, CDS, and the modification for PTSF can be found. Further clients may be registered at layer 2 and 3 to get indications from NDP and CDS.

VIII. Results

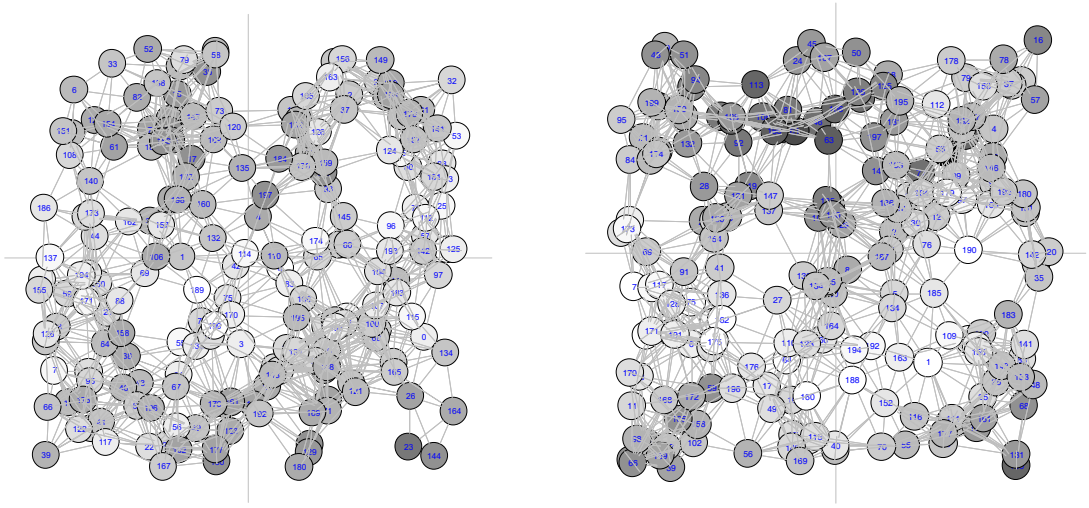
The Wireless Wide Area Network Simulator (WWANS) supports a realistic simulation of scenarios using the IEEE 802.11 PHY and MAC layers. In order to simulate the PTSF, the MAC layer has been modified according to the proposed changes that were pointed out in sections V and VII. Herefrom, new MLMEs for NDP and CDS were added to the MAC in order to evaluate the PTSF with CDS support. This section summarizes essential results that were found after several simulations and presents typical examples. Since the performance of the PTSF with the proposed extension for CDSs depends on two algorithms, namely CDS construction and PTSF synchronization, a two fold evaluation has been done. For all simulations and the results shown in this section, an oscillator model has been used that simulates a random clock drift between $\pm 100ppm$, since this is a typical limit for low budget off-the-shelf oscillators.

In a first step, the PTSF was simulated without CDS support in a highly mobile environment with station using the random walk mobility model. This model was configured to generate movements between $10\frac{m}{s}$ and $50\frac{m}{s}$ and billiard like reflections at the borders that are defined by $4000m \times 4000m$. Figure 7(a) shows the results of such a simulation with 200 stations using a beacon interval of 1.0 second. Here, the synchronization accuracy is visualized by grayscaling the stations in a linear way. In this scheme, white vertices indicate a station with an accuracy that is identical to the network median and otherwise black vertices indicate a node that has a deviation of more than $30\mu s$ apart from the median. It can be seen, that no station deviates more than $30\mu s$. The worst accuracy can be found at station 23 with $16\mu s$. The histogram in figure 8(a) gives more details of the results visualized in figure 7(a). Here, the distribution of synchronization accuracy within certain thresholds is outlined. Furthermore, it summarizes the results that were found by means of another simulation with 100 stations using the same area and nodes with identical clock and mobility model as used in the 200 nodes scenario. The results are almost alike and approve the reliable operation of the PTSF in MANETs.

In order to simulate and evaluate the performance of the PTSF using a CDS, further scenarios were used. A suited protocol stack, as shown in figure 6, has been used. Since the construction of a CDS directly influences the PTSF performance and the evaluation of CDS algorithms is beyond the scope of this paper, a static configuration has been used in order to avoid falsifications by permanent CDS reconstruction in environments with mobile stations. The results of such a simulation with 200 stations in a $4000m \times 4000m$ area is visualized in figure 7(b) by means of an identical grayscale scheme as described before. In this scenario, the CDS was build by 91 stations on basis of the marking and pruning process described in [9]. Here, it can be found that station 42 synchronizes within an accuracy of $30\mu s$ to the median, while all other stations are even closer. A histogram for this result is shown in figure 8(b) that gives again the distribution of synchronization accuracy within certain thresholds. The histogram also shows additional results that were found after simulating an almost identical set-up with the number of nodes reduced to 100. Following these results, it can be shown that the PTSF in conjunction with CDS scales well by means of the proposed extensions. These results show that existing CDSs may be (re-)used for synchronization by PTSF, without any additional requirement regarding their construction.

IX. Conclusions

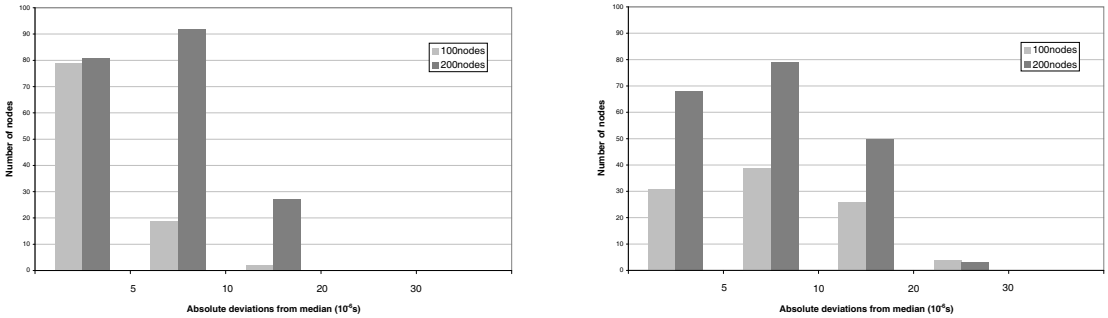
The PTSF allows to synchronize MANETs in a reliable and efficient manner. In comparison to the original IEEE 802.11 scheme, the beacon interval can be enlarged significantly by the proposed algorithm. While it predicts the clock drift



(a) PTSF with random walk model ($10\frac{m}{s}$ to $50\frac{m}{s}$)

(b) PTSF with CDS

Fig. 7. Simulation with 200 stations on $4000m \times 4000m$ and randomized clock drift within ± 100 ppm



(a) Histogram of the simulation with mobility

(b) Histogram of the simulation with CDS

Fig. 8. Histograms of two simulations with 200 and 100 nodes on $4000m \times 4000m$ and random clock drift within ± 100 ppm

between two TBTTs, it also allows equalizing lost beacon frames. Furthermore, an extension of the proposed algorithm was presented that allows using a CDS and therefore enables resource efficient network utilization. The proposed algorithm may reuse any CDS, without any specific requirements concerning its construction algorithm. Simulations of the proposed algorithms were done based on a random clock model in the range of $\pm 100ppm$ and the results show at least an accuracy of $30\mu s$ related to the median. Adaptive control of the beacon interval may improve performance of the algorithms, since faster beacon generation in the start-up phase will accelerate the convergence. On the other hand beacon intervals can be prolonged after a certain time that has been sufficient for initial convergence. Furthermore, in high density networks the beacon interval must be balanced even more carefully. Adaptive control of the beacon interval is an interesting option and therefore further research concerning this aspect will be carried out in the future.

REFERENCES

[1] I. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Elsevier Computer Networks*, vol. 47, no.4, 2005.

[2] *IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, ANSI/IEEE Standard 802.11, ISO/IEC 8802-11:1999(E).

[3] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?" *IEEE Communications Magazine*, vol. 39, no. 6, pp. 130–137, June 2001.

[4] L. Huang and T. Lai, "On the scalability of IEEE 802.11 ad hoc networks," in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking and computing (MOBIHOC)*, Lausanne, Switzerland, June 2002, pp. 173 – 182.

[5] J. Blum, A. Ding, M. Thaeler, and X. Cheng, *Handbook of Combinatorial Optimization: Connected Dominating Sets in Sensor Networks and MANETs*. Springer, 2005, vol. B (Supplement), ch. 8, pp. 329–369.

[6] J. Yu and P. H. J. Chong, "A survey of clustering schemes for mobile ad hoc networks," *IEEE Communications Surveys & Tutorials*, vol. 7, no.1, 2005.

[7] P. Rauschert, A. Honarbacht, and A. Kummert, "On the IEEE 802.11 IBSS and its timer synchronization function in multi-hop ad hoc networks," in *Proceedings of the International Symposium on Wireless Communication Systems (ISWCS)*, Port Louis, Mauritius, Sept. 2004, pp. 304 – 308.

[8] A. Honarbacht, "Contributions to scalable wireless ad hoc networks supporting quality-of-service," Ph.D. dissertation, University of Wuppertal, Germany, Dec. 2004, Logos Verlag.

[9] F. Dai and J. Wu, "Distributed dominant pruning in ad hoc networks," in *Proceedings of the IEEE 2003 International Conference on Communications (ICC)*, Anchorage, AK, USA, May 2003, pp. 353 – 357.