Wireless Sensor Network, 2011, 3, 39-53
doi:10.4236/wsn.2011.32005 Published Online February 2011 (http://www.SciRP.org/journal/wsn)

Network-Wide Time Synchronization in Multi-Channel Wireless Sensor Networks

Jari Nieminen[1], Lijun Qian[2], Riku Jäntti[1]
[1] *Department of Communications and Networking, Aalto University, Espoo, Finland*
[2] *Department of Electrical and Computer Engineering, Prairie View A&M University, Prairie View, USA*
E-mail: {*jari.nieminen*, *riku.jantti*}*@aalto.fi*, *liqian@pvamu.edu*
Received December 21, 2010; *revised December* 29, 2010; *accepted February* 9, 2011

Abstract

Recent advances in wireless sensor technology have enabled simultaneous exploitation of multiple channels in wireless sensor systems. In this paper, a novel time synchronization algorithm is proposed for multi-channel Wireless Sensor Networks (WSNs) called Multi-Channel Time Synchronization (MCTS) protocol. Time synchronization is critical for many WSN applications and enables efficient communications between sensor nodes along with intelligent spectrum access. Contrary to many existing protocols that do not exploit multi-channel communications, the protocol takes advantage of potential multiple channels and distributes the synchronization of different nodes to distinct channels and thus, reduces the convergence time of synchronization processes significantly.

Keywords: Time Synchronization, Multi-Channel Communications, Wireless Sensor Networks

1. Introduction

Evolution of high-tech and tiny Micro-Electro-Mechanical Systems (MEMS) has provided the platform for successful implementation of Wireless Sensor Networks (WSNs). Wireless communications are free from the costs and physical constraints of communication cables, and the development of small, low-cost wireless sensor nodes has enabled designing of various wireless sensor network applications such as military, health care, habitat monitoring and industrial applications [1]. These versatile sensors can be used to measure movement, humidity, pressure and temperature [2], just to name a few. Thus, it is natural that WSNs have gained a lot of attention from both, academic and industrial partners. However, the design challenges of WSNs include reliability, robustness, interference and scalability issues in addition to resource constraints. Multi-channel communications can be used to address these challenges and to provide high performance and trustworthy delivery of packets [3]. Naturally, exploitation of multiple channels at the same time requires more intelligence from sensor nodes and novel WSN protocols than those of single-channel systems.

Currently, wireless sensor systems usually utilize unlicensed frequency bands and hence, coexistence with various wireless systems such as WLAN introduces significant interference problems for low-power sensor nodes. Because of the crowded spectrum, the performance of WSNs is deteriorated and reliable communications cannot be guaranteed [4]. Hence, a new sensor networking paradigm called Cognitive Radio Sensor Networks (CRSNs) has been proposed [5]. Sensors equipped with cognitive radios are aware of their environment and internal state, and can make decisions about their radio operating behavior based on that information and predefined objectives. This enables more reliable packet delivery and more efficient utilization of scarce spectrum resources. In the near future, we will see the deployment of this kind of smart wireless sensor systems which enable dynamic spectrum access and opportunistic channel usage. In the field of CRSNs, our previous work has considered energy efficient adaptive modulation [6] and energy efficient spectrum access [7]. This paper investigates time synchronization in such networks where cognitive radio can be the enabling technology for the exploration of multiple channels and the focus is on the design of a time synchronization protocol that can be used after available channels have been identified.

Time synchronization plays a crucial role in various WSN applications. Precise time synchronization has been identified as one of the most important design ob-

jectives of WSN communication protocols, such as for industrial automation applications [8]. Industrial control applications require accurate time synchronization in order to achieve predictable data collection and enable reliable event logging [9]. Another promising application for WSNs is structural health monitoring which requires simultaneous vibration measurements [10]. Moreover, the impact of synchronization errors on damage detection in structures was studied in [11], where the authors show that even small timing misalignments will cause time shifts in sensor data which lead to problems in shape reconstruction. Time synchronization is mandatory for other applications as well, such as detection and tracking of various objects. Hence, the importance of time synchronization in WSNs motivated us to propose a novel protocol for such systems.

Furthermore, utilization of multiple frequency bands simultaneously enhances system throughput since many data transmissions can take place in parallel channels. On the other hand, wireless sensors are often energy constrained and power consumption is an important design issue. In order to save energy, devices can turn their transceivers off if they do not need to transmit or receive. Sleeping times of nodes can be maximized by minimizing transmission times when applying multi-channel communications. The realization of this so called *sleep mode* will require accurate time synchronization for coordination since otherwise nodes would not be able to sleep or wake up at the correct time. Moreover, many existing multi-channel MAC schemes use time frame/slot structures and therefore require precise time synchronization for effective operations, e.g. [12] and [13]. Time synchronization also enables the use of Time Division Multiple Access (TDMA) techniques that are generally considered to be more efficient than contention based Media Access Control (MAC) layer techniques.

Radio communication networks can be synchronized using in-band or out-of-band solutions. Currently Global Positioning System (GPS) is the most common out-of-band synchronization method and can provide precise timing. However, GPS may suffer from availability problems due to failure, blockage or jamming. In addition, GPS does not work in all situations such as indoors. Furthermore, cost and power consumption of GPS receivers makes it an infeasible solution for energy constrained WSNs [14]. Nonetheless, the proposed Multi-Channel Time Synchronization (MCTS) protocol can be exploited as an augmentation method for GPS as well.

In this paper, a network-wide time synchronization protocol is proposed especially for multi-channel WSNs called MCTS. The fundamental idea behind the proposed time synchronization algorithm is that multiple channels can be used simultaneously in order to minimize convergence time of the synchronization process. The main contributions are: 1) Using the proposed MCTS, network-wide synchronization is achieved in a fully distributed manner; 2) The proposed protocol takes advantage of the potential multiple frequency bands and distributes the synchronization of different pairs of nodes to distinct channels which reduces the synchronization time significantly; 3) MCTS also exploits multiple transceivers if available and thus, the capacity of multi-transceiver devices can be fully exploited using MCTS; 4) Detailed theoretical analysis of synchronization error and convergence time are provided and by simulations we show that the proposed MCTS is robust and outperforms other protocols such as TPSN in multi-channel WSNs in practice.

The paper is organized as follows. In Section 2 a brief overview on synchronization in communication systems is presented. Section 3 introduces our proposed MCTS. Performance analysis is provided in Section 4 which includes theoretical convergence time analysis and simulation results. Root node selection in case of MCTS will be discussed in Section 5. Section 6 contains the concluding remarks.

2. Related Work

Synchronization in communication systems includes physical layer synchronization and network time synchronization. Physical layer synchronization is required for successful transmissions between two radios. However, physical layer synchronization offers only phase synchronization between two radios and therefore, does not provide global time synchronization across entire WSNs.

Network Time Protocol (NTP) [15] has been widely used in the Internet to provide time synchronization in accuracy of milliseconds. However, WSNs need more accurate time synchronization for efficient operations and to comply with time synchronization requirements of various applications. Moreover, NTP is not designed for rapidly deployable distributed wireless networks and requires predefined hierarchy where low quality clocks synchronize to higher quality clocks. This is usually not the case in WSNs since nodes typically have similar clocks and no predefined assumptions on hierarchy can be made. Time synchronization in Mobile Ad Hoc Networks (MANET) was studied in [16]. The proposed algorithm calculates the time difference between the transmitter and the receiver so that time stamps from the transmitter can be mapped to correspond to the receiver's clock. Consequently, network-wide time synchronization is not provided and only time stamp transformation is carried out.

Time synchronization in wireless sensor networks has been widely studied and several protocols have been proposed such as Reference Broadcast Synchronization (RBS) [17] and Timing-sync Protocol for Sensor Networks (TPSN) [18]. In RBS, all the receivers time stamp a synchronization packet from the same transmitter individually and then exchange receive time stamps with neighbors. This scheme offers only relative time synchronization among neighboring receivers, not time synchronization with the transmitter. TPSN utilizes classical two-way synchronization between a master and a slave node. In the beginning, synchronization hierarchy is formed and after that masters and slaves exchange messages periodically to achieve time synchronization. Moreover, one-way time synchronization schemes for WSNs include for example Flooding Time Synchronization Protocol (FTSP) [19] and Time Diffusion Protocol (TDP) [20] which both exploit one-way messaging between masters and slaves.

An interesting approach for large scale wireless sensor networks was presented in [21]. The purpose is to adopt a synchronization scheme used by biological agents, for example fireflies, and the protocol synchronizes nodes by periodically sending pulses. However, the approach only offers phase synchronization, similarly to synchronously flashing fireflies, but no time synchronization since no time stamps are taken nor sent.

On the other hand, an algorithm for minimizing energy consumption of nodes during the synchronization process was presented in [22] called Pairwise Broadcast Synchronization (PBS). The innovative idea behind this approach is that multiple nodes can exploit timing information they overhear during the synchronization process and hence, the amount of messages required for synchronizing the network is minimized. Time stamps are exchanged between two "super nodes" (one master and one slave) and all other nodes that can hear both of these messages will synchronize according to this message exchange. It is shown that PBS performs much better than RBS and TPSN in terms of energy consumption. However, PBS introduces additional timing errors because of the additional synchronization path between receivers. Hence, even though this scheme is important in order to minimize the number of synchronization messages, two-way synchronization between all master and slaves nodes should be performed for highest accuracy. This work was extended in [23] to cover multicluster networks as well, however, the same problem with accuracy remains.

Comparing to PBS whose goal is mainly on energy saving, the proposed MCTS targets the cases where the availability of multiple channels may be exploited to reduce convergence time. Since the goals of the two are completely different, they can be considered as orthogonal approaches. Furthermore, they can be integrated to achieve greater performance for large sensor networks by taking advantages from both PBS and MCTS. For instance, in large sensor networks that need multiple super nodes and have multiple channels available for communications, MCTS may be used among the super nodes while PBS can be used for the rest of the sensor nodes.

To the best of our knowledge, utilization of multiple channels in the context of network-wide time synchronization in WSNs has not been studied previously, even though multi-channel wireless systems in general are continuously attracting more attention in the research community. For instance, local time synchronization using multiple channels was considered by So *et al*. in [24]. In their work, the authors considered parallel rendezvous-based multi-channel MAC approaches and introduced a synchronization protocol to synchronize one hop neighbor pairs in time. Network-wide time synchronization is not provided and the method in [24] is infeasible for many WSN applications.

We conclude that even though many time synchronization protocols have been designed for various WSN applications to provide network-wide time synchronization, none of those exploits multiple frequency channels and thus, the full capacity and advantages of multi-channel WSNs are not exploited. By using multiple bands for synchronization, the convergence time of synchronization processes can be minimized and hence, the demand for a new time synchronization protocol designed particularly for multi-channel WSNs clearly exists.

Our previous work concentrated on time synchronization of cognitive radio networks [25]. In this paper we extend that work significantly by considering important theoretical issues, such as the convergence time bounds for an individual node and for the networks. In addition, we analyze the performance of the proposed protocol by showing new simulation results with respect to different critical operation parameters such as the number of available channels, network density and transmission range of nodes. Furthermore, we also consider the root node selection problem and propose a suitable solution for the problem.

3. Multi-Channel Time Synchronization Protocol

Multi-Channel Time Synchronization protocol (MCTS) is a master-slave protocol where all slave nodes synchronize to a pre-selected root node. In small wireless sensor networks, the gateway (GW) node can act as a root node and provide time reference for the entire net-

work. However, in moderate sized WSNs the root node should be in the middle of the network to minimize convergence time and synchronization errors. How to select a root node in moderate sized networks will be discussed in detail in Section 5. In general, all time synchronization schemes need a functioning MAC protocol to operate and so does MCTS. To be more specific, MCTS requires a functioning multi-channel MAC that uses periodic beaconing, such as [13]. The synchronization protocol has three phases, *Hierarchy Discovery* (*HD*), *Synchronization Negotiation* (*SN*) and *Synchronization Execution* (*SE*). First, HD phases are used to create a synchronization hierarchy and keep the hierarchy up to date in order to cope with topology changes and node mobility. SN and SE phases always follow a HD phase.

An example of operations of MCTS in a multi-channel network is illustrated in **Figure 1**. We assume that the gateway node will set one channel as a Common Control Channel (CCC). τ is the synchronization slot length. T_{BI} means the end of the Beacon Interval (BI) and T_T stands for the total time that it takes to finish the synchronization process. HD phase is carried out during BI and SN and SE phases are carried out during the Negotiation Interval (NI). In the figure operations of MCTS are illustrated in discrete time in order to enable theoretical analysis later on. However, in practice the operations do not have to be bounded by this kind of discrete time presentation, instead the nodes can work without dividing time into synchronization slots.

In the beginning of HD phase, a selected root node will broadcast a *Hierarchy Beacon* (*HB*) message during the Beacon Interval (BI) that includes a root node's ID, synchronization level 1 and a list of available channels in addition to a send time stamp. All the nodes that receive this beacon message will set their synchronization level to 2 and broadcast a similar HB message with a new send time stamp. At this point the nodes at level 2 will set the root node as their master and synchronize to it in a coarse manner by using the time stamps they received. This process goes on until every node in the network has found out its level in the hierarchy and broadcasted a HB message. Since HD phase can be included into multi-channel MAC protocols that utilize periodic beaconing, overhead can be minimized and only the addition of node's synchronization level to a beacon is required.

After creating the synchronization hierarchy, MCTS proceeds to NI. In NI phase we have four different messages, *Synchronization Negotiation Message* (*SNM*), two *Synchronization Execution Messages* (*SEMs*) and *Data Negotiation Messages* (*DNMs*). SNM and DNM messages are similar and should be defined by the MAC layer protocol. NI starts with the root node announcing on the CCC that it is ready to start the synchronization process. After this, all its slaves will contact it and negotiate a channel for synchronization. After agreeing on the used synchronization channel, both the master and the slave will tune to the synchronization channel and carry out the actual time synchronization as illustrated in **Figure 2**. Each node has to wait until they have synchronized themselves to an upper level node before synchronizing others in order to prevent distribution of false synchronization information in the network.

After a slave has been synchronized it will announce on the CCC that it is ready to be a master for other nodes, if necessary, and all its slaves will contact it and negotiate synchronization channels. Later on in the case study we will show more precisely how this works. The exact operations depend on the number of transceivers per node and available channels as well as network topology. SE phase is initiated by a slave and it consists of two messages, *Synchronization Request* (*Sreq*) and *Synchronization Response* (*Sres*). The slave first transmits a Sreq message to the master, including the slave and master IDs, and a send time stamp (T_1). Send time stamps are taken at the MAC layer in order to mitigate send and

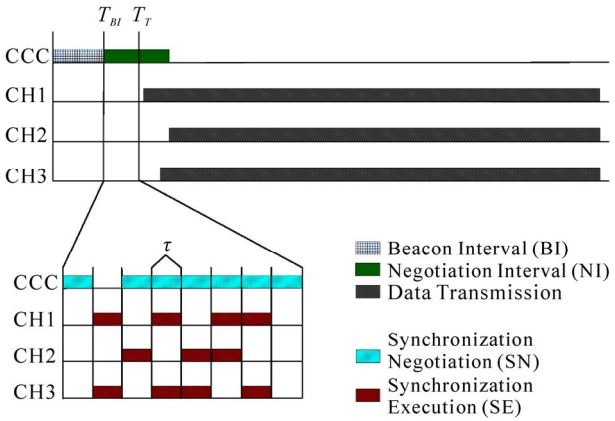

Figure 1. Demonstration of MCTS operations.

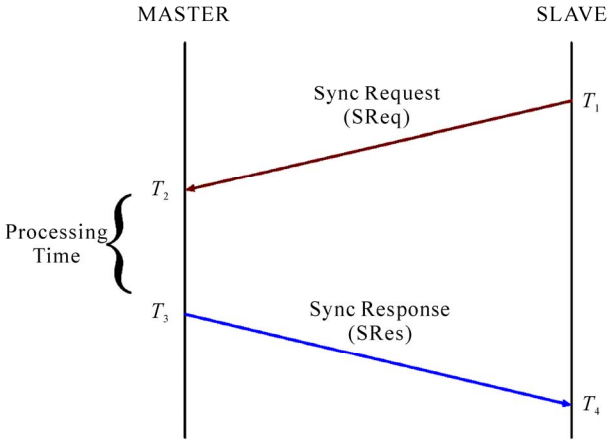

Figure 2. Synchronization Execution (SE).

access delays. At the receiver side, the master should time stamp (T_2) the incoming Sreq message as close to the physical layer (PHY) as possible, even before opening the packet, to diminish the impact of receive delay variation to synchronization accuracy. After receiving the packet, the master will create and transmit a Sres message which contains master ID, slave ID, T_1, T_2 and T_3. Send time stamp (T_3) is again taken at the MAC layer.

The slave will time stamp (T_4) the incoming Sres message. It is important to take both send and receive time stamps at the same place in both transceivers, respectively, to mitigate delay variations. After collecting all the time stamps, $\{T_1, T_2, T_3, T_4\}$, the slave node can be synchronized to the master and the propagation delay (d) and the clock offset (θ) can be calculated as follows

$$d = \frac{(T_2 - T_1) + (T_4 - T_3)}{2} \quad (1)$$

$$\theta = \frac{(T_2 - T_1) - (T_4 - T_3)}{2} \quad (2)$$

Since the response from the master comes instantly, it is safe to assume that clock drift will be constant during the synchronization procedure. By using linear regression for several time stamps it is possible to determine deterministic upper and lower bounds for clock offset and drift and so, synchronization accuracy can be improved. Processing of time stamps increases robustness of the synchronization process as well since false time stamps can be neglected. For time stamp processing, the algorithms presented in [26] or in [27] can be exploited.

Current wireless sensor nodes have only one transceiver which means that in order to avoid multi-channel hidden node problem, the nodes have to sense the particular channel before transmitting and exchange Request-to-Send (RTS) and Clear-to-Send (CTS) messages before synchronization execution. However, in the near future wireless sensor nodes may have at least two transceivers and in that case one transceiver can be tuned on to the CCC all the time and thus, RTS/CTS messages will not be needed on the synchronization execution channels. Added to this, MCTS is suitable for any system that utilizes multiple channels and so the nodes may even have 2 transceivers and a dedicated receiver tuned on to the CCC or more. MCTS enables simultaneous synchronization of numerous slaves if multiple transceivers are available.

The entire synchronization procedure is executed periodically so if a node moves or a new node wants to join, they will find their place in the synchronization hierarchy quickly. However, synchronization does not have to be performed every beacon interval and therefore, we introduce a design parameter M (a positive integer) that determines how often synchronization is carried out. This is an important enhancement particularly for WSNs since wireless sensors usually have small batteries and hence, energy consumption should be optimized. In **Figure 3** the idea of *Synchronization Interval* (*SI*) is depicted. In the figure M is set to 4 so synchronization is done every fourth beacon round. This way the synchronization overhead can be reduced and the operation of the protocol can be optimized with respect to the desired synchronization accuracy.

Nodes will use the information recorded from the BI to determine whether they may have slaves or not. If a node did not hear any HBs after sending one, it can start negotiation for a data transmission immediately after it has been synchronized since it does not have any slaves. However, if a node hears a HB from a lower level node after sending one, it has to announce on the CCC that it is ready to act as a master for other nodes after it has been synchronized. Furthermore, masters have to wait for one slot after synchronizing all their slaves before starting the data negotiation to ensure that synchronization negotiation is prioritized over data negotiation.

Because the wireless channel may have deep fade and suffer severe packet losses, a node may not be synchronized during the synchronization process while it is still able to negotiate for data transmissions after the channel recovers from deep fade. In this case, we suggest a back-up synchronization execution, where two-way synchronization messaging is carried out before data transmissions.

It is worth noting that MCTS is locally executed at each individual node, *i.e.*, all the definitions including BI and NI, are local to each node because of the fact that a node only needs to consider the nodes within its transmission range. For instance, times T_{BI} and T_T are not fixed and may be different for different nodes. Hence, each node acts individually and the proposed protocol is fully distributed. Therefore, NI phase will start propagating after the root node has sent out the HB and noticed that all of its neighbors have sent out their HBs as well. In other words, synchronization process moves on as a "wave" from the root node, which means that momentarily synchronization only impacts the nodes that are two hop away.

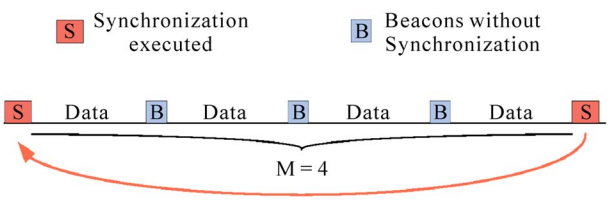

Figure 3. Synchronization Interval (SI).

3.1. Case Study

In order to clarify the detailed operation of MCTS, we present a case study according to the example scenario shown in **Figure 4**. To simplify the operations we assume that there exists a global CCC. We discuss two cases where each node has one transceiver in the first case and two transceivers in the second case. The focus will be on NI since it is the most important part of the protocol. The scenario consists of 9 nodes and the root node is denoted by 1. In the figure available channels for each node are presented next to each node. It is assumed that the required time for SN is larger than the time for SE.

First in the beginning of the HD phase the root node will send a HB in order to start the synchronization process. Nodes {2, 3, 4} will set their level as 2 and the root node will be their master. Now, all the nodes on level 2 send a HB and nodes {5, 6, 7, 9} will find out that they are on level 3 and send a HB as well. Finally, node 8 will receive a HB from node 7 and the hierarchy is completed. However, node 8 still has to broadcast a HB since there may be additional nodes. After this, the protocol proceeds to the NI phase.

At this point all the nodes have found out their levels in the synchronization hierarchy as described before. The root node starts the NI phase by announcing that it is ready to proceed with synchronization. After receiving this announcement, all the nodes on level 2 will negotiate with the root node and schedule a channel to carry out synchronization. Up to this point, the protocol operation has been the same regardless of the number of transceivers per node. However, in the next step, MCTS will take advantage of multiple transceivers if available. It is noticed that all nodes should listen to the CCC during NI in order to prevent overlapping allocations or alternatively use RTS/CTS message exchange on the chosen synchronization channel.

Let us first consider the one transceiver plus one receiver case presented in **Figure 5**. All nodes tune their receivers on the CCC and listen to that all the time. When a node negotiates transmissions it has to take into account the number of transceivers it has. Naturally, in case of one transceiver, a master can synchronize only one slave at a time. In our example, node 2 will first synchronize with the root node on channel 1. After this, node 2 will announce in the third slot that it is ready to synchronize its slave nodes and negotiate with node 5 on which channel to use, and the root node will synchronize another slave (node 3) on channel 2 simultaneously.

Similarly, in the fourth slot, nodes 1 and 4 will synchronize on channel 3 and nodes 2 and 5 will synchronize on channel 1. At the same time node 3 will negotiate

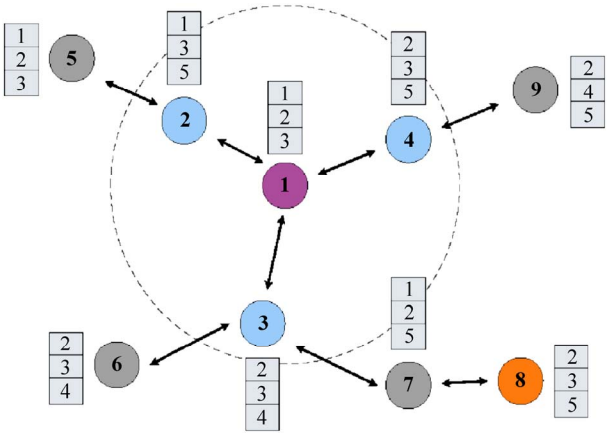

Figure 4. Network topology of the case study scenario.

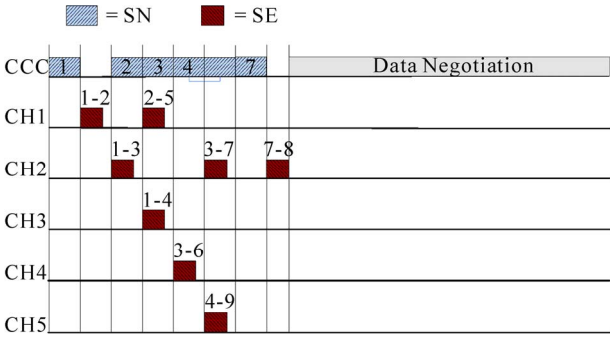

Figure 5. Negotiation Interval (NI) (1 transceiver).

with nodes 6 and 7. So in the fifth slot, node 4 will announce that it is ready to serve as a master. Node 5 does not need to announce on the CCC since it did not hear any HBs after sending one. Same applies for nodes 6, 8 and 9 as well. In the sixth slot, node 3 will also synchronize node 7 since it was unable to synchronize both, nodes 6 and 7, in the fifth slot. Finally, the last node to be synchronized is node 8. After the synchronization process, negotiations for data transmissions begin. Again, the ending of the synchronization for each node, T_T, could be different. In this case the length of the NI, $T_{NI} = T_T - T_{BI}$, for node 5 is four slots and for nodes 1, 2 and 6 five slots.

Negotiation interval for two transceiver case is presented in **Figure 6**. Now a master can synchronize two slaves simultaneously so the convergence time of the synchronization process becomes smaller and it is possible to utilize multiple frequency bands even better. For instance, the root node can synchronize nodes 2 and 3 at the same time in the second slot, and in the fourth slot, three pairs of nodes can synchronize simultaneously. In this case, nodes 2 and 3 are spatially separated so they can share the SN slot. Compared to the one transceiver case, the entire NI has reduced from eight slots to six slots.

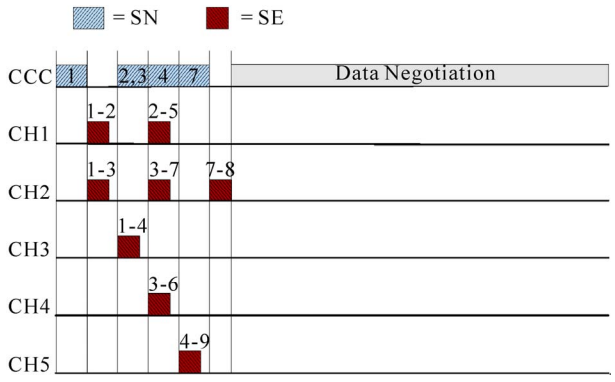

Figure 6. Negotiation Interval (NI) (2 transceivers).

3.2. Practical Considerations

In practice, MCTS can be implemented on top of various multi-channel MAC protocols. MCTS can be implemented together with any multi-channel MAC protocol that utilizes periodic beaconing and a common control channel. For example, multi-channel MAC protocols which are based on the dedicated control channel approach can be used. Dedicated control channel designs reserve one channel for distributing control information, see for example Dynamic Channel Assignment (DCA) [28], while data transmissions take place on different data channels simultaneously. MCTS can be easily implemented together with different dedicated control channel MAC protocols since the frame structure is essentially the same in both. Moreover, since these MAC protocols take care of the most complex operations, such as channel selections and resource reservations, the implementation of MCTS on top of dedicated control channel based MAC protocols should be straightforward. On the other hand, Multi-channel MAC (MMAC) [13] is an example of a split phase based approach in which time has been divided into two intervals. The scheme uses separate fixed time intervals for contention and data transmissions. Resource reservations are carried out on a common control channel during the contention period. MMAC provides periodic beaconing and enables collision-free transmission by using contention periods. Thus, MMAC and other split phase approaches are suitable for MCTS as well. We conclude that MCTS can be implemented on top of different MAC protocols. More importantly, we want to emphasize that the proposed multi-channel time synchronization algorithm can be used together with various MAC designs and it is not tied to any specific multi-channel MAC protocol.

Exploitation of multi-channel communications naturally introduces some additional complexity. However, since the used multi-channel MAC handles resource reservations and channel allocations, MCTS does not generate much complexity itself. Creation of the synchronization hierarchy is straightforward and only the addition of hierarchy levels to beacons is required. Moreover, during the actual synchronization phase two-way synchronization is carried out. This does not require heavy calculations and hence, requirements for processing power are light and the protocol can be implemented on existing wireless sensor nodes.

4. Performance Analysis

In this section, we analyze the performance of MCTS with respect to the convergence time of the synchronization process. We derive analytical results for the convergence time of an entire network and convergence time bounds for individual nodes. We also study the performance of MCTS under interference using simulations. We show that the convergence time of the synchronization process using MCTS is much less than that with serial two-way synchronization, such as TPSN [18], and MCTS performs well under interference as well. In this section we concentrate on the convergence time analysis since it is the main advantage of MCTS.

In case of MCTS time synchronization errors are similar to TPSN and thus, error analysis will be bypassed for now. However, for completeness we derive the results for synchronization errors and show in general that the accuracy of the two-way synchronization in the proposed MCTS is much better than that of a simple one-way synchronization in Appendix A.

4.1. Convergence Time Bounds for an Individual Node

Since MCTS is locally executed in each node, convergence time of MCTS may be different for different nodes. In this subsection we derive theoretical upper and lower bounds for the convergence time of MCTS in case of individual nodes. We assume that the interference range is twice the transmission range and therefore, for an individual node the convergence time of MCTS is determined by its two hop neighborhood. In this model we also assume that the data rate of each of the channels is the same regardless of the amount of channels, which means that each additional channel uses additional (orthogonal) frequency band, respectively. The number of additional masters in the two hop neighborhood is denoted by M. With additional masters we mean all other masters in two hop neighborhood of a node except its master and the node itself. For simplicity, all nodes have the same channels available for synchronization execution.

In general, the higher the data rate the smaller the

length of synchronization slots and hence, the length of the synchronization slot τ is inversely proportional to the data rate. As a consequence, the convergence time will be inversely proportional to the data rate as well. From **Figure 1** the convergence time of MCTS for an individual node is

$$T_{NI} = T_T - T_{BI} \tag{3}$$

Clearly, if a node does not have any slaves, the minimum convergence time will be achieved if its master can immediately carry out synchronization. In this case, the node only has to successfully negotiate and execute synchronization which takes two synchronization slots in the optimum case. Hence, the convergence time is lower bounded as follows

$$T_l \geq 2\tau \tag{4}$$

Now, we denote the number of available channels by *N*. If there exists some other master nodes in the two hop neighborhood, the convergence time will depend on the ratio of channels to additional masters (*N*/*M*). If (*N*/*M*) ≥ 1, the lower bound can be achieved. However, if (*N*/*M*) < 1, additional delay of $\lceil N/M \rceil$ may be induced, where $\lceil \ \rceil$ is the ceiling function. Moreover, if this particular node has one or more slaves, the convergence time will be extended. In case of idle channel conditions and only one slave, *S* = 1, the convergence time will increase by two slots and is given by

$$T_m = T_l + 2\tau \ . \tag{5}$$

However, if a node has more than one slave, the number of transceivers *X* will have an impact as well. Hence, in idle channel conditions the convergence time in general form is

$$T_X = T_l + \tau + \left\lceil \frac{S}{X} \right\rceil \cdot \tau \ , \tag{6}$$

given that the number of available channels is large enough (*N* ≫ *S*). By taking into account the fact that additional masters will have multiple transceivers as well, we find out that the maximum number of occupied channels is now $N_{occ} = M \cdot X$. If $N_{occ} < N$, a node can find at least one free channel for synchronization execution. Naturally, in the worst case a node cannot synchronize any of its slaves and has to wait until there is a channel available. Hence, we can summarize the upper bounds of convergence times for a master node as follows[1]:

$$T_u = \begin{cases} T_l + \tau + \left\lceil \dfrac{S}{N - N_{occ}} \right\rceil \cdot \tau \ , & N - N_{occ} < X \\ T_l + \tau + \left\lceil \dfrac{S}{X} \right\rceil \cdot \tau \ , & N - N_{occ} \geq X \\ \infty^{1} \ , & N_{occ} = N \end{cases} \tag{7}$$

If the amount of free channels is smaller than the number of transceivers, the convergence time of MCTS will be upper bounded with respect to the amount of free channels. Furthermore, if the amount of free channels is larger than the number of transceivers the convergence time of MCTS will be upper bounded with respect to the number of transceivers. Finally, if all the channels are occupied, no upper bound can be found.

4.2. Network Convergence Time

In this subsection we present a theoretical framework that can be used to analyze the overall time duration of a MCTS process in multi-channel networks. The overall convergence time in multi-channel systems depends on various design parameters. First of all, the number of transceivers per node determines how many slave nodes one master can synchronize simultaneously provided that the number of available channels is large enough. This leads to the second critical parameter, which is the number of available channels. Furthermore, network topology has an impact as well since it determines the number of levels in the synchronization hierarchy *L*. Another critical parameter is the maximum number of slave nodes that a master node may have *S*, which is closely related to network density (nodes/area). We denote the number of nodes by *η* and the transmission range by *r*. In the following analysis it is assumed that the transmission range is fixed for all nodes.

We approximate the number of hierarchy levels in a square network, x^2 m^2, as follows. Naturally, the distance from the root node, which is in the center of the network, to the edge is $y = x/\sqrt{2}$. Hence, the expected number of hierarchy levels is

$$E[L] = \frac{y}{r} = \frac{x}{\sqrt{2}r} \ . \tag{8}$$

Furthermore, we set network density as $\delta = \eta / x^2$ and thus, each node has $\delta \pi r^2$ neighbors on average. Clearly, all neighbors of the root node are its slaves and the nodes on the edges of the network do not have any slaves. Since the nodes are randomly positioned in the network, we estimate that in the vicinity of each node one half of the neighbor nodes can be on a lower level at maximum. Thus, the maximum amount of slave nodes is

[1]In theory all the channels can be occupied for infinity by other nodes. However, in practice the channels will be freed eventually since other masters cannot have an infinite number of slaves and thus, the process converges in any case at some point. Determination of the upper bound in this case would require more assumptions.

$$S = \frac{\delta A_r}{2} = \frac{\eta \pi r^2}{x^2} \cdot \frac{1}{2}, \quad (9)$$

where A_r is the area of transmission. Now, if the number of available channels is large enough, *i.e.* $N >> S$, the convergence time of a typical synchronization process of MCTS can be approximated as follows

$$O_{MCTS} = L + \frac{S}{X} \cdot (L-1). \quad (10)$$

Figure 7 compares theoretical and simulated results. As the figure demonstrates, theoretical results match well with simulation results if we have moderate network density, *i.e.* $1/(20\text{ m} \times 20\text{ m}) \le \delta \le 3/(20\text{ m} \times 20\text{ m})$, since the area was set as 200m×200m in this simulation. Even though in this theoretical analysis it is assumed that the network topology is wide spread, *i.e.* slave nodes of each master are not in the transmission range of each others, the theoretical results match simulation results perfectly in one transceivers case.In case of two transceivers the results do not match exactly but since the difference is relatively small, theoretical results can be used to give guidelines of the performance[2].

4.3. Simulation Results for WSNs under Interference

We performed simulations to determine convergence times of wireless sensor networks in case of MCTS and TPSN in practice. Since wireless sensors are often colocated with Wireless Local Area Networks (WLANs), coexistence of IEEE 802.15.4 based sensor networks and IEEE 802.11 b/g/n systems has gained a lot of attention recently [29,30]. This motivated us to test the performance of MCTS under interference in a real world scenario. In the simulations one WLAN transmitter was randomly placed in the area of 200 m × 200 m with varying number of wireless sensors. The maximum interference range of WLAN transmitters and the transmission range of WSN nodes was set as 200 and 50 meters, respectively, and the total amount of channels was set as 16. The WLAN transmitter occupied one randomly selected channel and thus, four channels were unavailable for some WSN nodes at a time. One channel was assigned as CCC to ensure the operation of both time synchronization protocols. Again, convergence times are calculated in slots.

In the first simulation all 16 channels were available for all WSN nodes, except the channels that are occupied by the WLAN transmitter, and we studied the impact of network size on the performance of MCTS and TPSN. Convergence times as a function of network size are shown in **Figure 8**. The convergence time of TPSN grows linearly when the amount of nodes in the network is incremented. MCTS performs similarly as a function of network size, however, since neighboring master nodes can use different channels for synchronization execution the angular coefficient of MCTS curve is significantly smaller. This leads to much smaller convergence times when using MCTS, even in one transceiver case. In general, the benefit gained using MCTS grows as the size of the network increases. In this scenario, MCTS with two transceivers uses less than 20% and even with one transceiver less than 40% of the resources compared to serial two-way synchronization when the amount of wireless sensors is 200. Moreover, the performance of MCTS is quite stable as the network size grows whereas the convergence time of TPSN is heavily affected by the number of nodes. However, other network parameters have an impact on the performance of MCTS as we will see next.

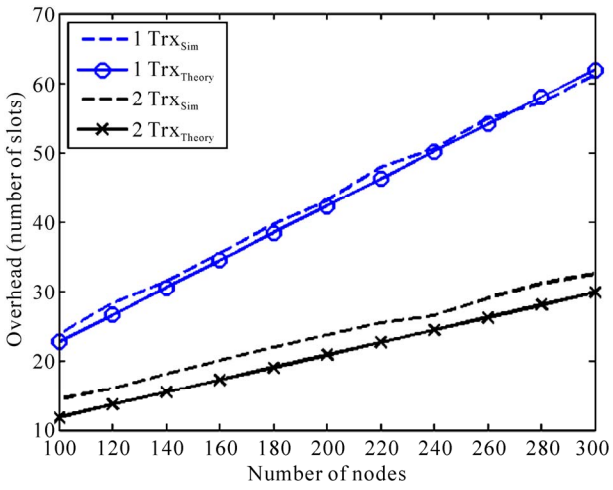

Figure 7. Convergence time as a function of network size.

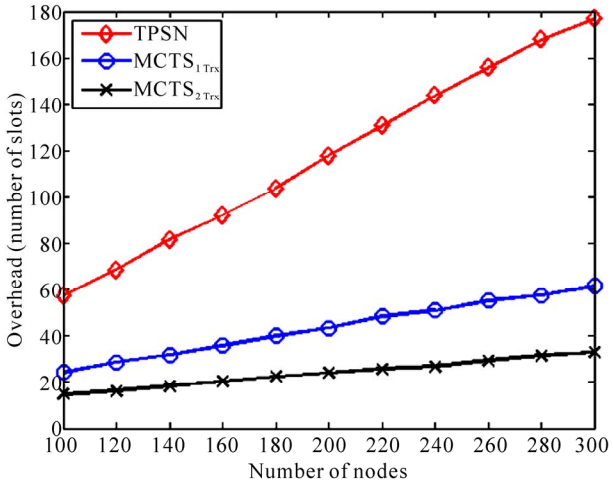

Figure 8. Convergence time as a function of network size.

[2]However, since this is only a simple approximation, it may not give as accurate results in all possible cases.

Secondly, we simulated the impact of transmission range on the performance of MCTS. Convergence times as a function of wireless sensors' transmission range are presented in **Figure 9**. Interference range of a WLAN transmitter was fixed as 200 meters. The amount of WSN nodes was set as 100. As we can see, the transmission range of wireless sensors has a bigger impact on the performance of serial two-way synchronization since all masters have to synchronize on the same channel and can only synchronize one slave at a time. The results imply that MCTS performs significantly better than TPSN in case of small transmission ranges but the difference shrinks while transmission range is increased. The reason for this is that in case of large transmission ranges each master node will have many slaves and thus, multi-channel communications cannot be fully exploited because of the small number of transceivers. Hence, as the transmission range grows the achieved gain from using two transceivers increases.

Finally, we simulated the effect of available channels on the performance of MCTS in general without considering any specific interference sources. Convergence times as a function of available channels are presented in **Figure 10**. Again, the amount of wireless sensors was set as 100. Naturally, the number of available channels does not have any impact on the convergence time of serial two-way synchronization. When the amount of available channels is low, the amount of transceivers has negligible impact on the performance of MCTS since the probability that a master and its slaves would share many available channels is extremely low. However, when the amount of available channels is increased, the performance of MCTS quickly improves. In this scenario, the performance of MCTS saturates when 40% of the channels are available. Consequently, only a small amount of available channels is enough to ensure close to the optimal performance in case of MCTS.

As the simulation results show, MCTS performs much better than serial two-way synchronization in most of the cases. Only when the amount of available channels is very low, MCTS and serial two-way synchronization perform similarly. Furthermore, the benefit gained from the use of MCTS depends on various network parameters. In one transceiver case MCTS outperforms serial two-way synchronization clearly and with two transceivers, the difference in performance is even larger. MCTS is scalable with respect to the number of nodes in the network as well as to the transmission range. This is a very promising and important property because it implies that MCTS can be applied to large scale multi-hop multi-channel wireless sensor networks. Simulation results show that MCTS is robust and performs well under interference.

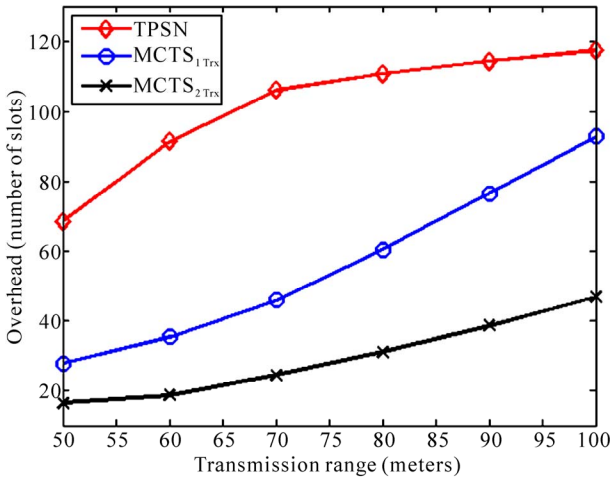

Figure 9. Convergence time as a function of transmission range.

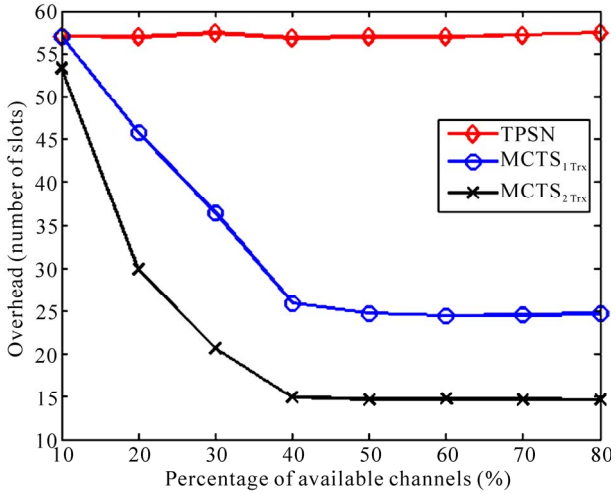

Figure 10. Convergence time as a function of available channels (16 channels in total).

It is evident that if less time is spent to carry out the synchronization process, more time for data transmissions is available. If multiple synchronization messages are required to estimate the clock drift, this time saving is even more important since multiple messages are sent and thus, the achievable gain by using MCTS increases. Hence, we conclude that utilization of MCTS is very important if multiple messages are required for precise estimation of the drift.

5. Root Node Selection for MCTS

Root node selection is an essential topic since it is important to choose a node in the topological center of the network as the root node in order to minimize the convergence time and worst case synchronization error of MCTS. We show that in order to minimize the conver-

gence time of MCTS, proper root node selection is required. Moreover, since synchronization error in multi-hop communications generally grows as a function of hops, the root node selection algorithm should be based on location in case of MCTS. It is assumed that all nodes in the network have similar clocks so there is no need to compare clock attributes of different nodes. For now, we consider only moderate sized networks where only one root node should be selected.

Root node election problem considered in this work is similar to the well-studied leader election problem in mobile ad hoc networks. The multicast operation of the Ad-hoc On-Demand Distance Vector (AODV) routing protocol [31] performs leader election to elect a new multicast group leader when a partition occurs. After the multicast tree becomes disconnected due to a network partition, there are two group leaders. If the components reconnect, the multicast operation of the AODV protocol ensures that only one of the group leaders eventually becomes the leader of the reconnected tree. A random leader election algorithm is proposed in [32]. Two distributed leader election algorithms, based on the routing algorithm TORA, are designed for operation in ad hoc networks. Both leader election algorithms guarantee that every connected component in the network will eventually have a unique leader. The first algorithm works when only a single topological change occurs. The second algorithm handles multiple concurrent topological changes.

In the above two approaches, the leader is *randomly* chosen without considering any specific requirements on the leader. On the contrary, extrema-finding leader election algorithms for mobile ad hoc networks have been proposed in [33], however, these algorithms are unrealistic as they require nodes to meet and exchange information in order to elect a leader. A more practical extrema-finding leader election algorithm is proposed in [34] based on self-stabilizing systems that is highly adaptive to arbitrary (possibly concurrent) topological changes. Since there will be frequent topological changes in WSNs due to the disruptions resulted from interfering users' activities, wireless sensors running out of battery and possibly sensors' mobility, this type of algorithms are highly desirable. Hence, we may adapt the algorithm proposed in [34] to find a root node in MCTS by assigning proper values to each node based on their topological locations. The difficulty is that it is non-trivial to obtain such topological information, and the mapping between the topological locations and the values assigned to each node needs to be designed.

In order to obtain proper values of each node based on their topological locations, we review two possible solutions based on the classical k-center problem and the minimum Connected Dominant Set (CDS), respectively. Then we highlight how to use these algorithms to fit our needs. We denote the WSN topology by a graph $G = (V, E)$, where V is the set of nodes and E is the set of links. The k-center problem identifies a subset S of V containing k nodes such that the distance from the rest of the nodes (in V-S) to S is minimized [35,36]. It formulates the scenario where a known number (k) of service facilities are to be deployed in the network so that they are "close to every client". The problem itself is NP-complete but can be approximated within a factor of 2 [37,38]. The root node selection problem in MCTS can be formulated as a 1-center problem [39]. However, most of the existing algorithms are centralized and many of them are only for a tree topology.

Another possible solution is using the well-developed distributed algorithms for computing minimum Connected Dominant Set (CDS) repeatedly (trim one layer of nodes in each round) to find the center node. The generic minimum dominating set problem in graphs is to find a minimum subset S of V such that any node in V-S has at least one neighbor in S, *i.e.*, dominated by S. S is a minimum CDS if it is also a connected subgraph. The problem of finding minimum CDS in a graph is NP-complete and cannot be approximated better than $log(n)$, where $n=|V|$ [40]. However, there are distributed approximation algorithms that meet such an approximation ratio with small overhead, such as [41] and [42]. For our problem of finding a center node, we apply distributed algorithm to compute a minimum CDS repeatedly until there is only one node left in the minimum CDS. In other words, we obtain a minimum CDS of V, S_1, in the first round. For all the nodes in V-S_1, their values are assigned as b_1. Then a minimum CDS of S_1, S_2, is obtained in the second round, and for all the nodes in S_1-S_2, their values are assigned as $b_2 = b_1 + 1$. We repeat this process until there is only one center node left in the final minimum CDS.

Both the k-center problem and minimum CDS problem assume static network topology and do not consider frequent topology changes that may happen very often in WSNs. Hence, we propose the following scheme:

1) Firstly, each node needs to obtain the set of their 1-hop neighbors.

2) Then we compute minimum Connected Domi-nant Set (CDS) repeatedly (trim one layer of nodes in each iteration) to compute the center node as well as map the topological locations to their respective values. CDSs can be calculated in a distributed manner using the algorithm presented in [42] for example.

3) After that, we apply the leader election algo-rithm in [34] to find the root node under highly dynamic scenarios if needed.

If the network topology changes slower than the con-

vergence of the scheme, then only steps 1 and 2 are necessary on the condition that the topology discovery algorithm in step 1 is able to discover network partitions due to link failures or networks merging due to new link formations. When the network is highly dynamic, *i.e.*, there are frequent topology changes induced by primary users' activities or node mobility, steps 1 and 2 and 3 will be executed periodically to accommodate the dynamic nature of a WSN. The scheme will guarantee that there will be a unique root node for each group of connectednodes when the algorithm converges, even under frequent network partitions or networks merging.

Figure 11 demonstrates the importance of root node selection on the performance of MCTS. In the figure, *RAND* stands for the case when the root node is randomly selected instead of using the proposed method for MCTS. The simulation scenario was the same as in Section 4 except the network density was fixed as 1/(20 m × 20 m). This is typical in WSN deployment and we wanted to study how MCTS scales with increased network size. The results imply that in order to minimize the convergence time of MCTS, proper root node selection is essential especially as the number of nodes grows.

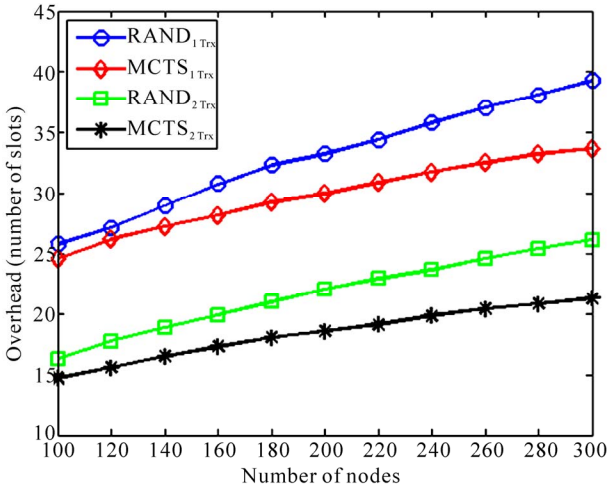

Figure 11. Impact of root node selection on the convergence time of MCTS.

6. Conclusions

In general, time synchronization is essential for many WSN applications and makes it possible for sensor nodes to communicate in a smart and efficient manner using sleep modes and TDMA. With cognitive radio as the enabling technology, multiple available channels can be identified by the sensor nodes in a cognitive radio sensor network. In this context, we presented a novel protocol for time synchronization of multi-channel wireless sensor networks, named Multi-Channel Time Synchronization (MCTS) protocol, and explained the operation of the protocol with a detailed case study. The unique features of the proposed MCTS include achieving network-wide synchronization using a fully distributed protocol and exploiting multiple channels to reduce convergence time. MCTS exploits the benefits of using multiple transceivers as well, if available. We studied the convergence time analytically and demonstrate the performance of MCTS through simulations. We show that the simulation results match our analytical results well. In addition, we observe that MCTS works well even only a small number of channels is available. Importance of root node selection was also discussed and a suitable solution is provided. We conclude that MCTS outperforms other existing solutions such as TPSN in multi-channel wireless sensor networks, and it is a promising candidate for time synchronization in future multi-channel cognitive radio sensor networks.

7. Acknowledgements

This research work is supported in part by TEKES (Finnish Funding Agency for Technology and Innovation) as part of the Wireless Sensor and Actuator Networks for Measurement and Control (WiSA-II) program and by the U.S. Army Research Office under Cooperative Agreement W911NF-04-2-0054.

8. References

[1] J. Yick, B. Mukherjee and D. Ghosal, "Wireless Sensor Network Survey," *Computer Networks*, Vol. 52, No. 12, August 2008, pp. 2292-2330. doi:10.1016/j.comnet.2008.04.002

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, Vol. 38, No. 4, March 2002, pp. 393-422. doi:10.1016/S1389-1286(01)00302-4

[3] G. Zhou, J. Stankovic and S. Song, "Crowded Spectrum in Wireless Sensor Networks," *Proceedings of the Third Workshop on Embedded Networked Sensors*, Cambridge, May 2006, pp. 1-5.

[4] L. Stabellini and M. U. Javed, "Experimental Comparison of Dynamic Spectrum Access Techniques for Wireless Sensor Networks," *Proceedings of the IEEE* 71*st Vehicular Technology Conference*, Taipei, May 2010, pp. 1-5.

[5] O. B. Akan, O. B. Karli and O. Ergul, "Cognitive Radio Sensor Networks," *IEEE Network*, Vol. 23, No. 4, July-August 2009, pp. 34-40.

[6] S. Gao, L. Qian, D. R. Vaman and Q. Qu, "Energy Efficient Adaptive Modulation in Wireless Cognitive Radio Sensor Networks," *Proceedings of the IEEE International Conference on Communications*, Glasgow, June 2007, pp.

3980-3986. doi:10.1109/ICC.2007.655

[7] S. Gao, L. Qian and D. R. Vaman, "Distributed Energy Efficient Spectrum Access in Wireless Cognitive Radio Sensor Networks," *Proceedings of the IEEE Wireless Communications & Networking Conference*, Las Vegas, March 2008, pp. 1442-1447.

[8] V. Gungor and G. Hancke, "Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches," *IEEE Transactions on Industrial Electronics*, Vol. 56, No. 10, October 2009, pp. 4258-4265. doi:10.1109/TIE.2009.2015754

[9] S. Chen, A. Dunkels, F. Österlind, T. Voigt and M. Johansson, "Time Synchronization for Predictable and Secure Data Collection in Wireless Sensor Networks," *Sixth Annual Mediterranean Ad Hoc Networking Workshop*, Corfu, June 2007, pp. 165-172.

[10] Y. Uchimura, T. Nasu and M. Takahashi, "Time Synchronized Wireless Sensor Network and Its Application to Building Vibration Measurement," *Proceedings of the* 33*rd Annual Conference of the IEEE Industrial Electronics Society*, Taipei, November 2007, pp. 2633- 2638.

[11] V. Krishnamurthy, K. Fowler and E. Sazonov, "The Effect of Time Synchronization of Wireless Sensors on the Modal Analysis of Structures," *Smart Materials and Structures*, Vol. 17, August 2008, pp. 1-13.

[12] J. Chen, S. T. Sheu and C.A. Yang, "A New Multichannel Access Protocol for IEEE 802.11 Ad Hoc Wireless LANs," *Proceedings of the 14th International Symposium on Personal, Indoor and Mobile Radio Communications*, Beijing, September 2003, Vol. 3, pp. 2291-2296.

[13] J. So and N. H. Vaidya, "Multi-channel MAC for Ad Hoc Networks: Handling Multi-channel Hidden Terminals Using a Single Transceiver," *Proceedings of the* 5*th ACM International Symposium on Mobile Ad hoc Networking and Computing*, Tokyo, May 2004, pp. 222-233.

[14] J. Elson and K. Römer, "Wireless Sensor Networks: A New Regime for Time Synchronization," *ACM SIGCOMM Computer Communication Review*, Vol. 33, No. 1, January 2003, pp. 149-154. doi:10.1145/774763.774787

[15] D. L. Mills, "Internet Time Synchronization: The Network Time Protocol," *IEEE Transactions on Communications*, Vol. 39, No. 10, October 1991, pp. 1482-1493. doi:10.1109/26.103043

[16] K. Römer, "Time Synchronization in Ad Hoc Networks," *Proceedings of the* 2*nd ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Long Beach, October 2001, pp. 173-182. doi:10.1145/501416.501440

[17] J. Elson, L. Girod and D. Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts," *Proceedings of the 5th Symposium on Operating Systems Design and Implementations*, Boston, December 2002, pp. 147-163. doi:10.1145/1060289.1060304

[18] S. Ganeriwal, R. Kumar and M. B. Srivastava, "Timing-sync Protocol for Sensor Networks," *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems*, Los Angeles, November 2003, pp. 138-149.

[19] M. Maróti, B. Kusy, G. Simon and A. Lédeczi, "The Flooding Time Synchronization Protocol," *Proceedings of the* 2*nd ACM Conference on Embedded Networked Sensor Systems*, Baltimore, November 2004, pp. 39-49. doi:10.1145/1031495.1031501

[20] W. Su and I. F. Akyildiz, "Time-Diffusion Synchronization Protocol for Wireless Sensor Networks," *IEEE/ACM Transactions on Networking*, Vol. 13, No. 2, April 2005, pp. 384-397. doi:10.1109/TNET.2004.842228

[21] Y. W. Hong and A. A. Scaglione, "A Scalable Synchronization Protocol for Large Scale Sensor Networks and Its Applications," *IEEE Journal on Selected Areas in Communications*, Vol. 23, No. 5, May 2005, pp. 1085-1099. doi:10.1109/JSAC.2005.845418

[22] K. L. Noh, E. Serpedin and K. A. Qaraqe, "New Approach for Time Synchronization in Wireless Sensor Networks: Pairwise Broadcast Synchronization," *IEEE Transactions on Wireless Communications*, Vol. 7, No. 9, September 2008, pp. 3318-3322. doi:10.1109/TWC.2008.070343

[23] K. L. Noh, Y. C. Wu, K. Qaraqe and B. W. Suter, "Extension of Pairwise Broadcast Clock Synchronization for Multicluster Sensor Networks," *EURASIP Journal on Advances in Signal Processing*, Vol. 2008, 2008, pp. 1-10. doi:10.1155/2008/286168

[24] H. S. W. So, G. Nguyen and J. Walrand, "Practical Synchronization Techniques for Multi-Channel MAC," *Proceedings of the* 12*th Annual International Conference on Mobile Computing and Networking*, Los Angeles, September 2006, pp. 134-145.

[25] J. Nieminen, R. Jäntti and L. Qian,"Time Synchronization of Cognitive Radio Networks", *IEEE Global Communications Conference*, Honolulu, December 2009, pp. 1-6.

[26] S. Yoon, C. Veerarittiphan and M. L. Sichitiu, "Tiny-sync: Tight Time Synchronization for Wireless Sensor Networks," *ACM Transaction on Sensor Networks*, Vol. 3, no. 2, June 2007, pp. 1-34.

[27] M. Lemmon, J. Ganguly and L. Xia, "Model-based Clock Synchronization in Networks with Drifting Clocks," *Proceedings of the Pacific Rim International Symposium on Dependable Computing*, Los Angeles, December 2000, pp. 177-184. doi:10.1109/PRDC.2000.897300

[28] S. L. Wu, C. Y. Lin, Y. C. Tseng and J. L. Sheu, "A New Multi-channel MAC Protocol with On-demand Channel Assignment for Multi-hop Mobile Ad Hoc Networks," *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks*, Dallas/Richardson, December 2000, pp. 232-237.

[29] S. Y. Shin, H. S. Park and W. H. Kwon, "Mutual Interference Analysis of IEEE 802.15.4 and IEEE 802.11b," *Computer Networks*, Vol. 51, No. 12, August 2007, pp. 3338-3353. doi:10.1016/j.comnet.2007.01.034

[30] M. Petrova, L. Wu, P. Mähönen and J. Riihijarvi, "Interference Measurements on Performance Degradation between Colocated IEEE 802.11g/n and IEEE 802.15.4 Networks," *Proceedings of the Sixth International Conference on Networking*, Sainte-Luce, April 2007, pp. 93-

98.

[31] E. M. Royer and C. E. Perkins, "Multicast Operations of the Ad-hoc On-Demand Distance Vector Routing Protocol," *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, Seattle, August 1999, pp. 207-218. doi:10.1145/313451.313538

[32] N. Malpani, J. Welch and N. Vaidya, "Leader Election Algorithms for Mobile Ad Hoc Networks," *Proceedings of the Fourth International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, Boston, August 2000, pp. 96-103.

[33] K. P. Hatzis, G. P. Pentaris, P. G. Spirakis, V. T. Tampakas and R. B. Tan, "Fundamental Control Algorithms in Mobile Networks," *Proceedings of the Eleventh Annual ACM Symposium on Parallel Algorithms and Architectures*, Bar Harbor, July 1999, pp. 251-260. doi:10.1145/305619.305649

[34] S. Vasudevan, J. Kurose and D. Towsley, "Design and Analysis of a Leader Election Algorithm for Mobile Ad Hoc Networks," *Proceedings of the* 12*th IEEE International Conference on Network Protocols*, Berlin, October 2004, pp. 350-360. doi:10.1109/ICNP.2004.1348124

[35] E. Minieka, "The m-Center Problem," *SIAM Review*, Vol. 12, No. 1, January 1970, pp. 138-139. doi:10.1137/1012016

[36] M. S. Daskin, "A New Approach to Solving the Vertex p-center Problem to Optimality: Algorithm and Computational Results," *Communications of the Operations Research Society of Japan*, Vol. 49, No. 9, 2000, pp. 428-436.

[37] D. Hochbaum and D. Shmoys, "A Best Possible Heuristic for the k-Center Problem," *Mathematics of Operations Research*, Vol. 10, No. 2, May 1985, pp. 180-184. doi:10.1287/moor.10.2.180

[38] J. Mihelic and B. Robic, "Approximation Algorithms for the k-center Problem: An Experimental Evaluation," *Proceedings of Operations Research*, 2002, pp. 1-6.

[39] N. Megiddo, "The Weighted Euclidean 1-center Problem," *Mathematics of Operations Research*, Vol. 8, No. 4, November 1983, pp. 14-25. doi:10.1287/moor.8.4.498

[40] S. Guha and S. Khuller, "Approximation Algorithms for Connected dominating Sets," *Algorithmica*, Vol. 20, No. 4, April 1998, pp. 374-387. doi:10.1007/PL00009201

[41] J. Wu and H. Li, "On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks," *Proceedings of the* 3*rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, Seattle, August 1999, pp. 7-14.

[42] P. J. Wan, K. M. Alzoubi and O. Frieder, "Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks," *Proceedings of the* 21*st IEEE International Conference on Computer Communications*, New York, June 2002, pp. 1597-1604.

Appendix A: Error Analysis

Many existing MAC layer protocols (such as [13]) use beacons to carry time synchronization information. As a result, only one-way synchronization is performed. In this appendix we derive the error formulas for MCTS and one-way synchronization and show that the accuracy of the two-way synchronization in the proposed MCTS is much better than that of a simple one-way synchronization in general. In order to derive synchronization error formulas for one-way and two-way multi-hop synchronization, let us first consider a simple case where Node 1 synchronizes Node 2, *i.e.* 1-hop case where Node 1 is the master and Node 2 is the slave. MAC layer time stamping is used in all of the following calculations. The slave node initializes the synchronization execution. First, the receive time T_2 for the SReq message at the master (Node 1) can be formulated as follows

$$T_2 = T_1 + T_s^2 + T_p^{2\to 1} + T_{rc}^1 + \delta_{t1}^{2\to 1} + \theta\,, \qquad \text{(A-1)}$$

where T_1 is the send time stamp at Node 2, T_s^2 the send delay at Node 2, $T_p^{2\to 1}$ is the propagation delay from Node 2 to Node 1 and T_{rc}^1 the receive delay at Node 1. $\delta_{t1}^{2\to 1}$ stands for the clock drift between nodes at time $t1$. θ is the clock offset between the nodes. Similarly, the receive time T_4 for the SRes message at the slave (Node 2) is

$$T_4 = T_3 + T_s^1 + T_p^{1\to 2} + T_{rc}^2 + \delta_{t3}^{1\to 2} - \theta\,, \qquad \text{(A-2)}$$

where T_3 is the send time stamp at Node 1, T_s^1 is the send delay at Node 1, $T_p^{1\to 2}$ is the propagation delay from Node 1 to Node 2 and T_{rc}^2 is the receive delay at Node 2. $\delta_{t3}^{1\to 2}$ stands for the clock drift between nodes 1 and 2 at time $t3$. Then, the relative clock drift between nodes during the synchronization message exchange, marked with δ, can be calculated as follows

$$\delta = \delta_{t1}^{2\to 1} - \delta_{t4}^{2\to 1}\,, \qquad \text{(A-3)}$$

since

$$\delta_{t3}^{1\to 2} \approx \delta_{t4}^{1\to 2} \equiv -\delta_{t4}^{2\to 1}\,, \qquad \text{(A-4)}$$

where $\delta_{t3}^{1\to 2}$ is the clock drift between nodes 1 and 2 at time $t3$, $\delta_{t4}^{1\to 2}$ is the clock drift between nodes 1 and 2 at time $t4$ and $\delta_{t4}^{2\to 1}$ stands for the clock drift between nodes 2 and 1 at time $t4$. Next, by subtracting equation (A-2) from (A-1) and by using (2), (A-3) and (A-4), the equation for synchronization error in case of two-way synchronization can be formulated as follows

$$\begin{aligned} E_{MCTS} &= \frac{1}{2}\cdot\left(\left(T_s^1 - T_s^2\right) + \left(T_p^{1\to 2} - T_p^{2\to 1}\right) + \left(T_{rc}^2 - T_{rc}^1\right) + \delta\right) \\ &= \frac{\Delta_{Tp} + \Delta_{Ts} + \Delta_{Trc} + \delta}{2} \end{aligned} \qquad \text{(A-5)}$$

where Δ_{Tp} is the difference in propagation delays and Δ_{Ts} and Δ_{Trc} are differences in send and receive times, correspondingly. Note that by using MAC layer time stamping, access delays are eliminated and most of the send delays as well. For N-hop communications with two-way synchronization the synchronization error is

$$E_{MCTS}^{total} = \sum_{i=1}^{N} \frac{\Delta^i{}_{Tp} + \Delta^i{}_{Ts} + \Delta^i{}_{Trc} + \delta^i}{2}\,, \qquad \text{(A-6)}$$

where δ^i corresponds to the error introduced by ith oscillator on the path. Similarly we can derive the equation for error in case of one-way synchronization. Now the receive time T_2 for the message is the same as in (A-1). From that we can derive the synchronization error in N-hop communications with one-way synchronization the synchronization error is

$$E_{beacon}^{total} = \sum_{i=1}^{N}\left(T_s^i + T_p^i + T_{rc}^i + \delta^i\right). \qquad \text{(A-7)}$$

When using one-way synchronization, the error is cumulative and includes all the delays. Thus, the error grows fast as a function of hop count and distance. On the contrary, when using two-way synchronization, the synchronization error includes only the differences in various delays, which are much smaller than the delays themselves.

In fact, the error with two-way synchronization will be only slightly cumulative. According to practical measurements with wireless sensors reported in [18], the synchronization error is almost the same over multiple hops on average. However, the worst case synchronization error will grow as a function of hops. The measurements performed in [26] show that the one-way delay with wireless sensors is approximately 1.4 milliseconds, which means that the synchronization error with one-way synchronization will be the same. With two-way synchronization, the average error is 17 microseconds and worst case error is 44 microseconds [18]. Naturally, the performance of one-way synchronization can be improved by using FTSP [19] or TDP [20]. However, these methods require multiple messages to achieve microsecond precisions and thus, convergence times are significantly larger and mobility of nodes will cause problems.

