Evaluation of an Uncertainty Aware Hybrid Clock Synchronisation System for Wireless Sensor Networks

Christoph Steup, Sebastian Zug and Jörg Kaiser
Department of Distributed Systems
Faculty of Computer Science
Otto-von-Guericke-University
Magdeburg, Germany
Email: {steup,zug,kaiser}@ivs.cs.ovgu.de

***Abstract*—Wireless Sensor Networks, aiming to monitor the real world's phenomena reliably, need to combine and post-process the detected individual events. This is not possible without reliable context information for each event. One important aspect of this context is time. It enables ordering of events as well as deduction of further data like rates and durations. Consequently, an unreliable time base affects all aspects of further processing. Any uncertain time information generates uncertain values and decisions, which jeopardize the correct behaviour of the system unless the system knows them. Therefore, the synchronisation of the clocks of the individual nodes is of high importance to the reliability of the system. On the other hand, tight and reliable synchronisation typically induces a large message overhead, which is often not tolerable in WSN scenarios. This paper evaluates a new hybrid synchronisation mechanism enabling tight synchronisation in single-hop environments and looser synchronisation in multi-hop environments. The lack of a guaranteed synchronisation precision is mitigated by an explicit synchronisation uncertainty, which is passed to the application. This enables the application to react to changes in the current synchronisation precision. The new approach is evaluated using the network Simulator OMNET++ and a small scale wireless sensor network to verify the expected performance and assumptions. The new method showed excellent performance in single-hop environments and a decreasing synchronisation precision based on the topological distance in multi-hop scenarios, which are useful results for Wireless Sensor Networks.**

***Keywords*–Wireless Sensor Networks; Clock Synchronisation; Uncertainty**

I. INTRODUCTION

Time is a relevant dimension in all aspects of the physical world. Consequently, measuring, comparing and computing time is a very important capability of any technical system observing or interacting with the real world. The approach presented by Steup et al. [1] proposed a method to include a low-overhead uncertainty aware clock synchronisation mechanism for hierarchical networks. The proposed method provides tight synchronisation in single-hop environments and looser synchronisation with a decreasing precision based on the topological distance. In this paper this approach will be explained, compared to existing approaches and evaluated using a simulated environment and a small-scale Wireless Sensor Network (WSN).

WSN gain increasing attention by researchers as well as industry and governments. They provide the ability to monitor large areas for events efficiently and with small effort. One example is the SafeCast Project [2], which aims to provide people with the ability to cheaply monitor radiation in their vicinity and share this data with others. The forest fire detection system described by Yu et al. [3] is another example aiming to protect people and nature using real-time information fed by a WSN. A third example is the Avalanche system of Michahelles et al. [4] enabling easier detection and rescue of people caught in an avalanche using medical sensor attached to the person. There are many more applications of WSN using real-time data streams to react to environmental conditions, hazards or catastrophes.

As described by Römer and Mattern [5] WSN come in the different forms. The design space consists of multiple dimensions, which contain network topology (from single-hop to graph), granularity (high-power to smart dust), coverage (dense, sparse and redundant), heterogeneity (heterogeneous or homogeneous), communication interface and mobility (from static to high speed mobile). There are also commonalities that all WSN share like perceiving the environment using equipped sensors and a limitation in energy, processing power and network bandwidth. Generally WSN are considered to be disseminating data from the sensor nodes to the sinks. Depending on the application and the limitations of the WSN it is often necessary to handle evaluation or decision making of the data in the network itself. Some WSN like the above described systems need extend this basic dissemination. They either need to deliver decisions to sinks within the network or need to process the data within the network to limit the needed bandwidth. These systems need robust decision making and processing mechanisms to be accepted by people, since a missed warning may cause fatalities, whereas too many wrong warnings lead people to ignore them. Therefore, reliable post processing and context detection are crucial to provide a robust output. In this context time together with space are the most important context attributes a WSN needs to deliver to enable evaluation and decision making. Consequently, the quality of the time base is an important aspect governing the reliability of the output, since an unreliable time base influences the ordering as well as the deduction of events, which in consequence become unreliable as well. Therefore, a reliable, precise global time base is a must-have for each WSN detecting safety relevant events.

The description of the approach starts with a short introduction to basic terms and concepts of clock synchronisation in general in Section II. Afterwards we discuss some relevant existing clock synchronisation protocols in Section III, followed by the description of our concept in Section IV. In Section V, we describe our implementation within the Omnet++ network simulator and our real world evaluation, the tests carried out and their results. The paper closes with a conclusion of our results and some ideas on future work in Section VI.

II. Foundations of Clock Synchronisation

The time base of any digital system is provided by a clock. The clock itself can be modeled as linearly increasing counter, with a defined time period between the clock events. This period is called *granularity (g)* of the clock [6]. Since a clock is a sensor measuring time, it has an *accuracy (α)* describing its difference to the real value of time and a *precision (π)* describing the maximum difference between any two clocks in a system [6]. The accuracy of a clock directly defines the precision of this clock to be: $\pi = 2\alpha$ [6].

Precision and accuracy are typically unbounded for unsynchronized clocks. Therefore, these parameters can only be evaluated after the clocks of a system are synchronized. To be able to evaluate the quality of a single clock other parameters need to be considered. The most important parameter is the *drift (ρ)* [6]. It describes the maximum deviation of the frequency of the clock towards and ideal clock. The frequency deviation between two real clocks is measured by the *skew*. Over time the drift will increase the *offset (δ)* between the real clock and the ideal clock [6]. The skew will increase the offset between any pair of clocks. Consequently, the aim of any synchronisation mechanism is to limit the offset to a certain value defining either precision or accuracy of the clocks of the system. Since drift and clock skew will increase the offset again after a successful synchronisation, the synchronisation needs to be repeated periodically. This *resynchronisation interval* is a very important parameter for most synchronisation protocols as it defines the achievable precision or accuracy [6]. Ideally, the resynchronisation interval depends only on the drift or clock skew, but in real system acquiring, distributing and comparing time stamps takes time itself. This time is another important parameter for synchronisation protocols and is determined by the *critical path*. Estimating the length of this path or minimizing it are two general approaches to increase synchronisation precision or accuracy.

A. Internal and External Synchronisation

The easiest solutions for an accurate, precise and reliable time base are external time sources. One exemplary source is the DCF77 [7] standard used in Germany to distribute the time of the atomic clock in Braunschweig. This standard uses a very simple protocol transmitted over a 77.5 kHz wave to supply whole Germany with only a single sender.

Another possible source is the Global Positioning System (GPS) as described by Dana [8]. GPS is a Time-of-flight based positioning system, which needs a very accurate time reference to infer the time between transmission and reception of the signal. For this purpose one satellite is typically used as a time reference, whereas at minimum three others are used to derive the position. Consequently, GPS always provides a time reference additionally to the position.

GPS and DCF77 are examples of external synchronisation, where a high accuracy reference source is used to synchronize all nodes. This approach can guarantee precision and accuracy based on the used communication and quality of the reference source [9].

An alternative approach is internal synchronisation, which aims to provide a bounded precision without accuracy [9]. This is done by synchronizing the nodes with each other without any external reference. Typical protocols providing internal synchronisation are the Network Time Protocol (NTP) [10] and the Precision Time Protocol (PTP) [11].

Using additional hardware to supply each node of a WSN with an accurate time is often not feasible because of limitations to cost and energy consumption the nodes need to adhere to. Therefore, pure external synchronisation is seldom used. Internal synchronisation protocols like NTP and PTP are too expensive considering message count and are typically not robust enough for the unreliable wireless links. As a consequence, specially adopted time synchronisation protocols for WSN were developed. These protocols can be divided in two groups: Averaging and Non-Averaging protocols. The averaging protocols exchange time stamps between nodes without any hierarchy. Afterwards, they try to achieve consensus on the resulting time by averaging the timestamps [9]. Non-averaging protocols typically have a master distributing its time in the network with slaves adopting this time to mitigate the offset between pairs of nodes [9]. Obviously, the averaging approaches have problems tolerating malicious or faulty clocks, since clock values with large offsets will strongly influence the average. On the other hand, non-averaging protocols using master nodes have problems tolerating a failure of their master node.

In general, all approaches try to provide a trade-off between message overhead and synchronisation precisions. Additionally, they try to tolerate message losses and changes in the topology of the network. However, most of the existing protocols either try to provide a tight synchronisation in a single hop environment and degrade heavily in multi-hop environments or provide a generally looser synchronisation in both. Unfortunately, none of the existing protocols provide the application with information on the current status of the synchronisation, which might be degraded by errors in communication or heavy changes in topology. The next section will discuss some existing clock synchronisation protocols and evaluate their fitness towards a hybrid uncertainty aware clock synchronisation for WSN.

III. State of the Art

In order to assess the current state of clock synchronisation for WSN, we describe six approaches representing basic concepts in the following section.

A. Reference Broadcast Synchronisation (RBS)

Reference Broadcast Synchronisation as described by Elson et al. [12] is an averaging internal synchronisation mechanism exploiting a physical broadcast in a shared medium. The synchronisation starts with one node transmitting a *NOW*-message to all other nodes. This message serves as an indication for all nodes to take a local time stamp. Afterwards, the timestamps are exchanged between all nodes. This mechanism

reduces the critical path to the transmission time of the *NOW*-message and the local processing time on each node until the local time stamp is taken. This provides very tight synchronisation in single-hop scenarios as long as the computation time is bounded. However, in worst case for n nodes $\mathcal{O}(n^2)$ messages are needed for a single synchronisation round, as visible in Figure 1.

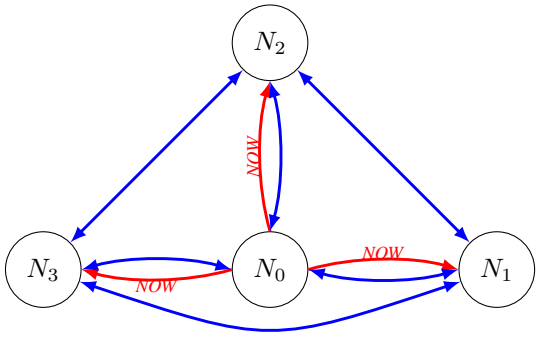


Figure 1. Overview of a single synchronisation round of RBS. The red arrows indicate the *NOW*-message, whereas the blue arrows indicate the messages containing the timestamps of the nodes.

Finally, all nodes individually compute their offset towards the mean of all exchanged timestamps. Towards this end, each node n_i computes its phase offset to each other node n_j over all received broadcasts by another node n_k based on (1). Additionally, a linear regression analyses based on the estimated offsets is used to estimate the clock skew of each node towards the global clock. Using this information all receivers agree on a uniform clock rate and time within the single-hop environment.

$$\delta_{i,j} = \frac{1}{m}\sum_{k=1}^{m}(ts_{j,k} - ts_{i,k}), i \neq j \neq k \quad (1)$$

In summary, RBS provides tight synchronisation bounds for single-hop environments. However, it reacts very sensitive to mobile or faulty nodes. This is caused by the used averaging mechanism of the protocol. The contribution of all nodes to the averaged new time and clock rate may create large shifts of agreed global clock whenever one node's clock is far off. This is especially problematic whenever a node is only a temporary member of the broadcast group.

B. CesiumSpray (CS)

CesiumSpray by Verissimo et al. [13] is a pseudo-hierarchical non-averaging hybrid clock synchronisation mechanism providing strong failure resilience in real-time networks. The baseline idea of this system is the synchronisation of groups of nodes within a single-hop environment towards an external reference. Consequently, the approach uses internal and external synchronisation mechanism. The authors proposed GPS receivers as the external reference, but other time sources like DFC77 receivers or even precise local clocks are possible. An example network employing CesiumSpray is depicted in Figure 2. As visible, the GPS satellites together with the GPS receivers in the single-hop environments act as a multi-hop backbone to sync the individual clusters.

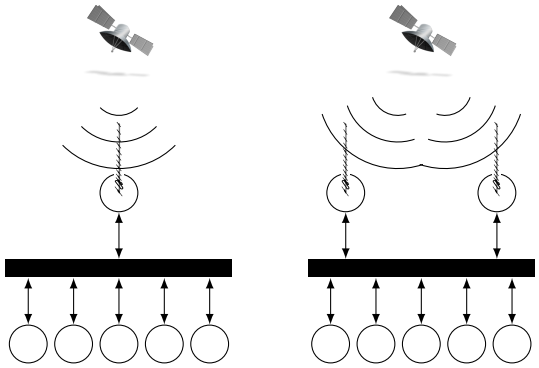


Figure 2. The pseudo-hierarchical structure of an example CesiumSpray clock synchronisation network.

Within each cluster, so called tight broadcasts are used to spray the GPS reference to the other nodes. Special care was taken by the authors to handle faulty GPS references or faulty GPS receiver nodes. This is achieved by acknowledging the broadcast of each GPS receiver node from all other nodes to enable a homogeneous view of the whole cluster. After the reception of a tight broadcast from all local GPS receiver nodes, an agreement is necessary to select a single broadcast. Afterwards, all local nodes sync against the chosen tight broadcast. In case no GPS reference is available, a purely internal synchronisation mechanism is used to preserve precision and accuracy.

The precision of the approach is based on the accuracy of the GPS receivers, which typically is better than $110ns$, and the precision of the internal synchronisation mechanisms. The authors tested their approach in a real-time network using very old Motorola 68020 CPUs and a token-bus network. As a result they achieved a precision of $500\mu s$ with a resynchronisation interval of $150s$.

CesiumSpray is a resilient multi-hop clock synchronisation system for distributed real-time systems. It provides excellent failure resilience with a proven upper bound on the synchronisations precision and accuracy. The precision is dependent on the tightness of the network and the real-time capabilities of the platform used. The drawbacks of the approach consist in the need for real-time capable networks and operating systems to use the strong failure resilience and the need for an external time source providing the "multi-hop" capability. Due to the provided failure tolerance the synchronisation costs are quite high.

C. Delay Measurement Time Synchronisation Protocol (DMTS)

The Delay Measurement Time Synchronisation Protocol described by Ping [14] modifies RBS, see Section III-A, by exploiting low-level hardware access and a non-averaging computation. It extends the *NOW*-message with a time stamp ts_0 taken and inserted just before sending. An averaging is not used anymore, since the transmitted time stamp ts_0 can directly be used by each receiving node. Therefore, the exchange of the individual local time stamps is omitted and the message count is heavily reduced, as visible in Figure 3.

The time between the transmission time ts_M and the reception time ts_S is composed of the interrupt service time

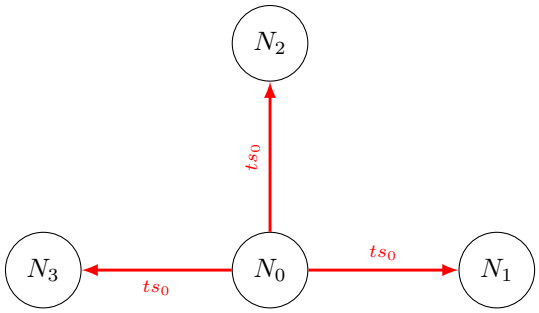


Figure 3. Overview of a single synchronisation round of DMTS. The red arrows indicate the time stamp messages distributed by the master node N_0.

and the propagation time of the network. To reach similar synchronisation precision as RBS, the author estimates the duration of the interrupt service t_{comp} of the receiving node analytically and modifies the inserted time stamp just before transmission, as shown in Figure 4. If the analysis is correct only the transmission time τ on the medium remains as the critical path.

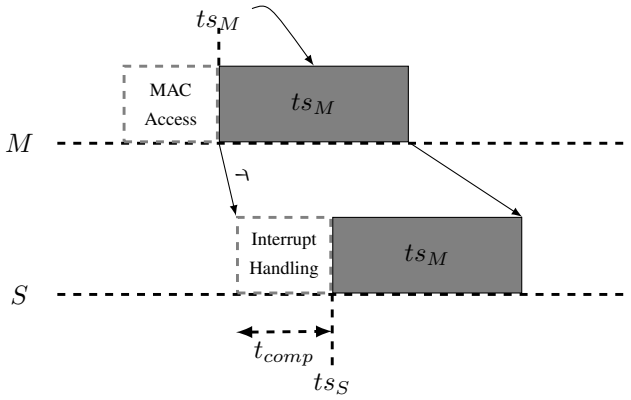


Figure 4. Minimization of the critical path of DMTS.

For this approach to be feasible, a master election is necessary, since a single node needs to transmit its time stamp for multiple rounds until synchronisation in all nodes is reached. After that, the time stamp providing node can freely be chosen from the set of synchronized nodes.

The multi-hop synchronisation of DMTS is based on a hierarchical distribution of the master's time. Each node in the single-hop neighbourhood synchronized with the master will act itself as the master for its own single-hop neighbourhood using its synchronized time. In consequence, the time of the master will distribute over the whole network over time.

The hierarchical structure is non-deterministic, since nodes will always sync themselves to the first node broadcasting a time beacon they hear. Typically, this will create shortest routes from the time master to the individual notes, which may decrease the reliability of the link and therefore lower robustness.

DMTS provides a high precision clock synchronisation in single-hop neighbourhoods as well as multi-hop synchronisation with slowly degrading performance. It solves the large amount of message necessary for a single synchronisation of RBS. However, in-depth knowledge of the needed hardware and communication mechanisms as well as low-level hardware access is needed to use it. The protocol has very limited robustness as only a single time stamp is communicated from master to its neighbourhood, which leaves the protocol open to omission failures and faulty nodes spreading wrong clock values when elected.

D. Continuous Clock Synchronisation in Wireless Real-time Applications (CCS)

The Continuous Clock Synchronisation for Wireless Real-time Applications by Mock et al. [15] is a non-averaging master-slave synchronisation method extending the basic clock synchronisation of the 802.11 standard [16]. It is itself the adoption of the approach of Gergeleit and Streich [17] towards 802.11 networks.

In contrast to the standard time synchronisation mechanism of the 802.11 protocol, the clocks of the slaves are not simply set to the time stamp of the master, but are gradually adapted by adjusting their rate. Consequently, the authors propose the concept of virtual clocks (VC) to enable dynamic adjustment of frequency and offset. The behaviour of such a virtual clock is depicted in Figure 5.

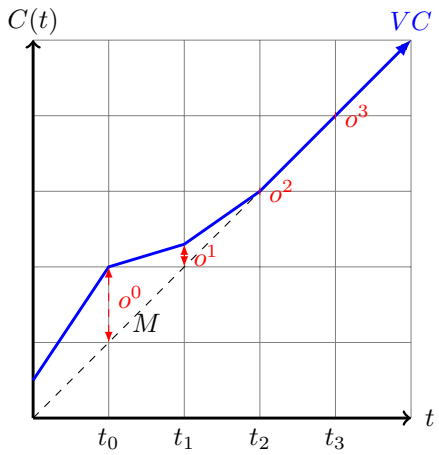


Figure 5. A virtual clock (VC) of a time slave adapting its frequency to compensate offset δ and clock skew towards its master clock M.

Additionally, the precision of the synchronisation is enhanced by dividing the time beacon in a *NOW*-message indicating surrounding nodes a time stamp ts_i needs to be taken and an additional message containing the time stamp ts_m of the *NOW*-message. The major benefit of this approach is the ability to exactly estimate the time stamp of the *NOW*-message minimizing the critical path and omitting any estimation of media access, transmission or computation delays.

The additional message needed to transmit the time stamp t_m after the *NOW*-message can be omitted if the master's time stamp is incorporated in the next *NOW*-message. Figure 6 shows the timing behaviour of the protocol over two rounds of synchronisation.

The proposed protocol provides failure resilience in case of omission as long as the amount of omitted packets is smaller then the estimated omission degree. Based on this assumption,

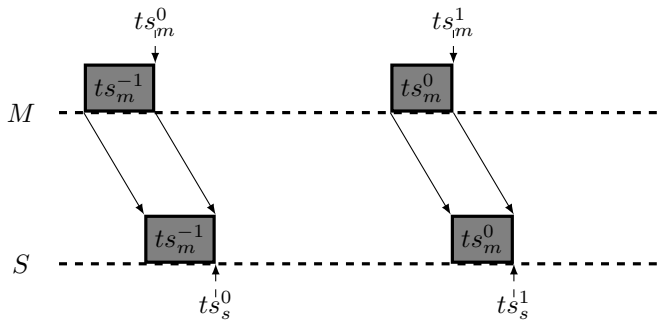


Figure 6. Timing diagram of 2 synchronisation rounds between a time master M and a time slave S.

the authors analytically derived (2) to compute the guaranteed precision π:

$$\pi \leq \frac{2\delta\Delta t(\rho+2)}{(\Delta t)^2-\delta^2}(\text{OD}+\text{INT})+\delta\left(1+\frac{(\Delta t)^2(1+\rho)+\delta^2)}{(\Delta t)^2-\delta^2}\right) \quad (2)$$

In their experiments, they used a synchronisation interval $\Delta t = 10s$ on clocks with a drift of $\rho = 10^{-5}$. The maximum age of their second time stamp was INT $= 1s$. They measured a maximum jitter of the critical path of $\delta = 46\mu s$, resulting in a precision of $\pi = 150\mu s$, which fits with the results expected by (2) in case of an omission degree of OD $= 1$.

Even though multi-hop synchronisation is not covered in the paper, the approach of DMTS, see Section III-C, can also be used for CCS. Consequently, the multi-hop performance shall be similar to DMTS.

This approach enables continuous clock synchronisation without gaps in the time base suitable for real-time applications. Additionally, it provides better precision than the baseline 802.11 synchronisation mechanism without additional message overhead. It provides failure resilience and a guaranteed precision for a known omission degree. However, it is generally only useful for single-hop environments with a dedicated access point, since multi-hop performance will have the same problems as DMTS.

E. Probabilistic Clock Synchronisation Service (PCS)

The Probabilistic Clock Synchronisation Service by PalChaudhuri et al. [18] is an extension of RBS enabling a dynamic trade-off between synchronisation precision and message overhead. This approach transmits n *NOW*-messages in one synchronisation round, which are used to derive the skew of the sender's and the receiver's clock though linear regression. The results are combined and transmitted back to the receivers in range. By comparing their own data with the data received from the sender, they are able to adopt their own clocks. To derive the number of needed *NOW*-messages the authors assumed the synchronisation error to be normal distributed with zero-mean and a standard deviation of σ. Based on this distribution, the authors analytically derive the probability $P(|\epsilon| < \epsilon_{max})$ of the synchronisation error to be less than a specified value ϵ_{max}. For a specified probability of the synchronisation to be more precise than ϵ_{max}, the authors derive the number n of message needed. This number heavily depends on the standard deviation σ of the normal distribution.

The multi-hop mechanism in this approach is based on a time transformation in a hierarchically structured network. If a node n_1 is synchronized to the master node n_0 broadcasting its time reference packets, this node can transform the received broadcasts based on arrival time and re-broadcast them to spread the synchronisation within the network. The error induced by the retransmission is again assumed to be normal distributed and analysed the same way as for the single-hop approach.

The robustness of the approach very much depends on the accuracy of the estimated distribution, if the real distribution of the synchronisation error in single-hop or multi-hop environments fits with the assumed distribution, the synchronisation will be very robust, since all failures as well as their results are already contained in the distribution. However, a mismatch of these will result in an unpredictable behaviour of the synchronisation. Consequently, the estimation of the probability distribution is the crucial part of this approach.

The approach provides a dynamic trade-off between message overhead and synchronisation precision even in multi-hop scenarios. Even though the trade-off depends on the standard deviation of the synchronisation error, the acquisition of this value was not covered by the authors. Additionally, only a mathematically prove without any simulation was conducted to evaluate the idea. Consequently, the authors never discussed the effects of non-normal distributed synchronisation errors.

F. Time Synchronisation in Ad-Hoc Networks (TSAN)

Römer's Time Synchronisation in Ad-Hoc Networks [19] is based on Christian's Algorithm [20]. It is a non-averaging internal synchronisation using pair-wise offset estimation. It estimates the round trip time of a message between sender and receiver, as visible through the transmission of e_0 in Figure 7 and ultimately tries to order events created by the system. Whereas Christian's Algorithm proposed a dedicated server for clients to communicate to, Römer attaches time stamps to events communicated in the network. Therefore, Römer's algorithm ideally induces a zero message overhead. However, not all events are acknowledged by the receiver, which might create large durations between events flowing in both directions between two nodes. This is mitigated by the insertion of additional dummy events in case the duration grows too large. This is shown through the communication between N_0 and N_2 in Figure 7.

TSAN supports multi-hop synchronisation directly, since it measures round-trip time from sender to receiver and back. The basic concept makes no assumption on the amount of hops between sender and receiver and works with an arbitrary amount of hops.

One of the major ideas of this approach is the usage of time transformations instead of clock synchronisation. In consequence, no node needs to set their clock to a certain value, but they only transform the timestamps of the events they receive to their own time domain using their estimated offset between sender and themselves. However, this offset estimation is limited to the events flowing by. As a result the first event flowing between two nodes cannot be transformed. The quality of the transformation depends on the amount of

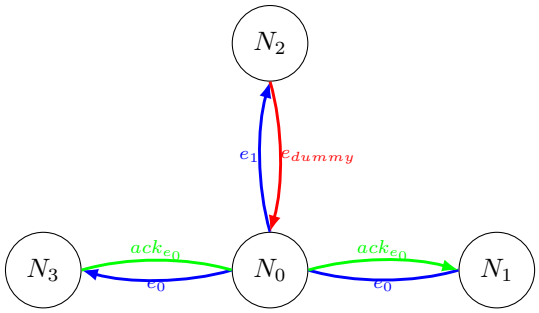


Figure 7. Schematic of a TSAN synchronisation. The blue events e_0 and e_1 are transmitted. e_0 is acknowledged back, showed as a green arrow. e_1 needs a dummy event (red) inserted, since no acknowledgement is done.

events. More events grant a better estimation of the offset between two nodes and therefore increase the transformation's precision.

Since TSAN assumes no bound on network delays of the events and the acknowledgements it uses, it cannot provide an upper bound of the synchronisations precision. This is mitigated by the usage of time intervals, which provide a duration instead of a singular time stamp. To achieve its main goal of ordering events, the time stamps are compared to infer the order of the events. However, it is possible that the intervals overlap and an ordering is not possible. In such cases, the system will provide a result of *MAYBE* for the order relation. The final order relation of TSAN for two time stamped events e_0 and e_1 containing the time stamps $e_0.ts = [t_0, t_1]$ and $e_1.ts = [t_2, t_3]$ is described in (3).

$$[t_0, t_1] < [t_2, t_3] = \begin{cases} \textit{YES} & : t_1 < t_2 \\ \textit{NO} & : t_3 < t_0 \\ \textit{MAYBE} & : t_2 < t_1 \end{cases} \quad (3)$$

TSAN applies a very loose multi-hop synchronisation with an ideal message overhead of 0 in a multi-hop network. Unfortunately, the real message overhead is heavily dependent on the actual communication in the network and is therefore very hard to estimate for a real system. The used time intervals together with the *MAYBE*-results of the event ordering provide the protocol with a certain robustness against large synchronisation errors caused by message losses or unpredicted delays.

G. Summary

The individual problems and features of the protocols are summarized in Table I. As visible none of the described approaches fully solve the problem of multi-hop uncertainty aware clock synchronisation in wireless sensor networks. However, each approach contains individual features, which might enhance the performance of our approach.

- CS provided the idea of hierarchically structured synchronisation architecture using different synchronisation mechanism on the different layers of the hierarchy.
- DMTS introduced the usage of hardware access and knowledge on low-level behaviour to decrease the jitter of the critical path.
- CCS added the idea of a virtual clock following the clock of another node to provide a steady time base without gaps. Additionally, it provided the differentiation of time stamp transmission and *NOW*-indication to shorten the critical path.
- PCS proposed the estimation of the synchronisation error as a normally distributed random variable with a zero-mean and a known deviation. We will exploit that idea to estimate the uncertainty of our synchronisation.
- TSAN provided the idea of time transformation between nodes on event reception, which we will exploit for our multi-hop synchronisation.

TABLE I. Comparision of the discussed time synchronisation protocols.

Protocol	Synchronisation Precision	Multi Hop Capability	Message Overhead	Robustness
CS	medium	inherent	$\mathcal{O}(n)$	medium
RBS	high	none	$\mathcal{O}(n^2)$	fragile
DMTS	high	possible	$\mathcal{O}(n)$	fragile
CCS	high	possible	$\mathcal{O}(1)$	robust
PCS	medium	inherent	dynamic	medium
TSAN	low	inherent	0 - $\mathcal{O}(1)$	medium

The proposed approach incorporates the beneficial properties of the different clock synchronisation mechanisms in a single clock synchronisation, which is uncertainty- and topology-aware and produces time intervals usable by an application. The next section describes it in detail.

IV. Uncertainty Aware Clock Synchronisation (UACS)

For an efficient synchronisation of clocks in WSN multiple parameters are important. On one hand, the synchronisation needs to be scalable, while on the other hand the overhead may not exceed a certain threshold to safe battery and prevent an overload of the network. Most of the approaches discussed in Section III favour one over the other. However, if we limit our self to certain base topologies better solutions might be found. One interesting topology is the cluster tree structure of IEEE 802.15.4 networks [21] in beacon-enabled mode. This mode divides the nodes in groups called Personal Area Networks (PANs), which have an individual coordinating instance managing the internal communication. The individual PANs communicate only through their respective coordinators, as visible in Figure 8. This hierarchical network structure may also be found in other types of networks like Bluetooth scatternets as proposed by the Bluetooth standard [22]. In the remaining section of the paper we consider an 802.15.4 network, with an already established cluster tree structure. The formation and the handling of dynamic changes in this structure are not considered in this paper.

Based on the initial assumption, that clock synchronisation may have a decreasing precision based on topological distance between nodes in the network, we propose a hybrid clock-synchronisation, consisting of a tight synchronisation mechanism for each individual PAN called Intra Cluster Synchronisation and a loose synchronisation mechanism between the individual PAN Coordinators, called Inter Cluster Synchronisation.

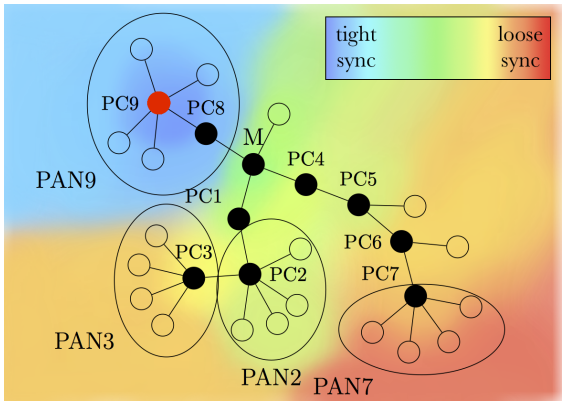


Figure 8. Example cluster tree structure of an IEEE 802.15.4 Network. The colours indicate the topological distance between each node and $PC9$.

A. Intra Cluster Synchronisation

The Intra Cluster Synchronisation is based on the CCS approach, see Section III-D. Therefore, each PAN slave $P_{s_j} \in Slaves$ has a virtual synchronized clock $VC_{s_j}(t)$. This clock uses the time stamps created by the node's hardware clock $C_{s_j}(t)$ and modifies it based on the current rate $r_{s_j,i}$:

$$VC_{s_j}(t) = r_{s_j,i}\left(C(t)_{s_j} - C(t_i)_{s_j}\right) + VC_{s_j}(t_i) \quad (4)$$

The task of the Intra Cluster Synchronisation is the estimation of the parameter $r_{s_j,i}$ for each slave at each synchronisation round i. The 802.15.4 standard allows a PAN Coordinator to attach additional information to the beacon frame. We use this to attach a 64bit time stamp $t_{c,i}$ to each beacon b_{i+1} transmitted by the coordinator. The attached time stamp represents the coordinator's time of successful transmission of the last beacon. This time stamp together with the local reception time of the last beacon $t_{s_{j,i}}$ is then evaluated by each slave P_{s_j} to compute a new rate $r_{s_j,i}$.

As described by DMTS, see Section III-C, hardware knowledge may be used to provide the needed local time stamps. The 802.15.4 standard provides the `PD-Data.confirm` primitive as a local event indicating completion of a transmission. The time of this event is used as the source of the time stamp $t_{c,i}$. On reception of the beacon each PAN slave P_{s_j} takes a local time stamp $t_{s_{j,i+1}}$. The networks tightness τ together with the internal computation time t_{comp} of the nodes limits the accuracy of the local time stamps. This computation time is mitigated by the `PD-DATA.indication` primitive of the 802.15.4 standard. Therefore, we consider t_{comp} to be very small in our approach and the time difference between creation of the local time stamps is bounded by τ.

After acquiring the time stamp for the actual synchronisation round the PAN slaves compute the offset $\delta_{j,i} = t_{c,i} - VC(t_{s_j,i})$ between their previous local time stamp $VC(t_{s_j,i})$ and the time stamp transmitted through the beacon $t_{c,i}$. This is used to compute a new rate $r_{s_j,i+1} = 1 + k_r\delta_{j,i}$ for the node's virtual clock to compensate the offset, with k_r being a proportional factor controlling the rate of adaption.

The Intra Cluster Synchronisation provides continuous clock synchronisation between the PAN Coordinator and its slaves. The overhead is minimal since no additional message is necessary and the beacons are only slightly enlarged. The robustness of the synchronisation mainly depends on the used algorithm to detect a crash and reselect a PAN Coordinator.

B. Inter Cluster Synchronisation

Diverging from the Intra Cluster Synchronisation, see Section IV-A, a PAN Coordinator never modifies its own clock. Instead every event received by a PAN Coordinator P_{c_r}, which is transmitted by another adjacent PAN Coordinator P_{c_s} is transformed in the time domain of the PAN Coordinators clock $C_r(t)$, as proposed by TSAN, see Section III-F. To achieve this, the PAN Coordinator P_{c_r} needs to calculate a virtual clock $VC_{r,s}(t)$ for each adjacent PAN Coordinator P_{c_s}.

The virtual clocks are handled similarly to the Intra Cluster Synchronisation, since all beacons of all adjacent PAN Coordinators P_{c_s} are received by PAN Coordinator P_{c_r}. On reception of a beacon containing a time stamp $t_{s,i}$, P_{c_r} acquires a local time stamp $t_{r,s,i+1}$. This enables the computation of the offset $\delta_{r,s,i} = t_{s,i} - t_{r,s,i}$ between P_{c_s} and P_{c_r}. Afterwards, P_{c_r} updates the rate $r_{r,s,i} = 1 + k_r\delta_{r,s,i}$ for the virtual clock $VC_{r,s}(t)$ towards P_{c_s}. Therefore, each PAN Coordinator has an internal list of virtual clocks following the clocks of each adjacent PAN Coordinator as visible in Figure 9.

On reception of an event e_n from P_{c_s} containing a time stamp $e_n.ts_s$, P_{c_r} is able to transform the time stamp of the event to its own clock $C_r(t)$. The transformation is done by adding the offset between the Virtual Clock of the sender $VC_s(t)$ and the clock of the receiver $C_r(t)$ to the event's time stamp:

$$e_n.ts_r = e_n.ts_s + C_r(t) - VC_s(t) \quad (5)$$

In case of multi-hop communication the event's time stamp is always transformed to the receiver's clock domain before forwarding it to next hop. Consequently, all nodes only need to estimate the offset using a virtual clock for their directly adjacent neighbors. An example scenario is shown in Figure 9. In this picture three PAN Coordinators are in direct vicinity and estimate their offset using virtual clocks. An event is transmitted from PC_0 to PC_2 via PC_1. During the forwarding of the event the time stamp of the event is adjusted by the estimated offsets to transform it to the local time domain of the current node.

C. Performance Estimation

The performance of the synchronisation depends on certain network and node parameters like the tightness of the network (including propagation speed and interrupt handling time) τ, the drift of the nodes ρ and the algorithm specific variable k_r. To analyse the behaviour of the algorithm we define two clocks: one for the transmitter of the beacon $C_s(t) = C_s(t_{s,i}) + (t - t_{s,i})(1 \pm \rho)$ and one for the receiver of the beacon $C_r(t) = C_r(t_{r,i}) + (t - t_{r,i})(1 \pm \rho)$. The offset between these two nodes is described as the difference between local clock of the sender and the Virtual Clock of the receiver as $o_{r,s}(t) = C_s(t) - VC_r(t)$. The receiver cannot observe this offset, but based on our algorithm it uses the visible offset $\delta_{r,s}(t) = o_{r,s}(t) \pm \tau$ based on the time stamp contained in the beacon. Consequently, the virtual clock's propagation formula may be rewritten to:

$$VC_r(t) = VC_r(t_{r,i}) + (C_r(t) - C_r(t_{r,i})(1 + k_r\delta_{r,s}(t_{r,i})) \quad (6)$$

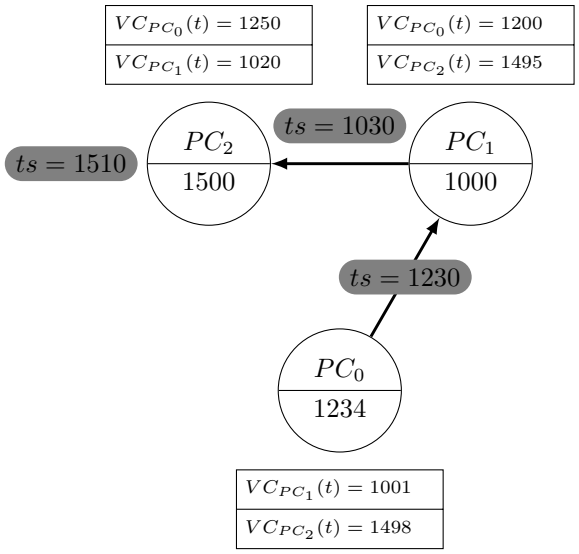


Figure 9. An example network composed of three PAN Coordinators with their respective virtual clocks estimating their offset towards each other. An event transmitted from PC_0 to PC_2 is shown in grey including its time stamp.

Entering $VC_r(t)$ in $o(t)$ enables replacing of $C_s(t) - VC_r(t_{r,i})$ with $o_{r,s}(t_{r,i}) + (t - t_{r,i})(1 \pm \rho)$. Together with the simplification of $C_r(t) - C_r(t_{r,i})$ to $(t - t_{r,i})(1 \pm \rho)$ one obtains:

$$o_{r,s}(t) = o_{r,s}(t_{r,i}) - (t - t_{r,i})(1 \pm \rho)k_r\delta_{r,s}(t_{r,i}) \pm (t - t_{r,i})2\rho \quad (7)$$

By substitution of $\delta_{r,s}(t_{r,i})$ and $(t - t_{r,i})$ to Δt as well as factoring out $o_{r,s}(t_{r,i})$ the following equation is achieved:

$$o_{r,s}(t) = o_{r,s}(t_{r,i})(1-\Delta t k_r) \pm (\Delta t(k_r \rho o(t_{r,i}) + \rho(2+\tau) + k_r\tau)) \quad (8)$$

To enable a normal flow of time, the rate of the Virtual Clock needs to be bigger than zero. Therefore, the value of k_r must not exceed $\frac{1}{max(\delta_{r,s}(t))}$. The propagation of the offset may now be separated into an offset compensating part: $o^-_{r,s}(t) = o_{r,s}(t_{r,i})(1 - \Delta t k_r)$ and an offset increasing part: $o^+_{r,s}(t) = \pm\Delta t(k_r\rho o_{r,s}(t_{r,i}) + \rho(2 + \tau) + k_r\tau)$. The offset decreasing part will converge towards zero, because it resembles the geometric sequence $a^{i+1} = a^i q, 0 < q < 1$. The offset inducing part can be separated into an offset dependent and offset independent part. The value of the offset dependent part $\Delta t k_r \rho o_{r,s}(t_{r,i})$ depends very much on the offset between sender and receiver at the last synchronisation and the time passed since this synchronisation. For tightly synchronized nodes the impact of this part will tend towards zero.

The guaranteed precision of the method for tightly synchronized nodes is the absolute value of the offset independent part of $o_{r,s}(t)$. The resulting precision of the Intra Cluster Synchronisation is $\pi_{intra}(t) = \Delta t(\rho(2 + \tau) + k_r\tau)$.

The offset independent part is $\Delta t(\rho(2 + \tau) + k_r\tau)$. In case no beacon loss occurs Δt will be approximately two times the beacon Δt_b interval, because every beacon contains the sender's time stamp of the last round and the maximum offset is always reached directly before the next beacon arrives. The resulting precision in case of OD beacon omissions will therefore be:

$$\pi_{intra} \leq (2 + \text{OD})\Delta t_b(\rho(2 + \tau) + k_r\tau) \quad (9)$$

The value k_r can be viewed as a trade-off factor choosing between fast synchronisation and robust synchronisation. This is caused by the presence of the factor in the offset compensating part, where it decreases the existing offset stronger if it is bigger. On the other hand, bigger k_r values will increase the results of the offset increasing part depending on the values of ρ and τ.

For the offset estimation of the adjacent neighbours the Inter Cluster synchronisation uses the same approach as the Intra Cluster Synchronisation uses. Therefore, the multi-hop synchronisation precision Π_{inter} for tightly synchronized nodes can be bounded using the hop count h:

$$\pi_{inter} \leq h(2 + \text{OD})\Delta t_b(\rho(2 + \tau) + k_r\tau) \quad (10)$$

For loosely synchronized nodes the offset of the last synchronisation will be a relevant issue. However, this is only a problem in mobile systems, since static systems will always compensate the offset as long as the time between synchronisations is not bigger than the maximum considered offset. A direct consequence is that the synchronisation precision in multi-hop scenarios can be greatly enhanced if routes of tightly synchronized nodes are chosen.

D. Mobility

Mobility influences the networks topology and therefore, the association of slaves to masters and the interconnectivity between masters. Consequently, if a slave loses connection to is respective master it is entering the orphaned state and may only communicate again after it re-associated. If the node re-associates with the same master the node's maximum offset $o_{r,s}(t)$ will depend on the time between loss of link and the reception of the next beacon Δt and the offset of master and slave at the last synchronisation as described by (8). A tightly synchronized slave will increase its offset based on the time of the last synchronisation and the drift of the nodes. A loosely synchronized node will additionally increase the offset by a value proportional to last offset $(o_{r,s}(t_{r,i}))$, time since last synchronisation (Δt), drift (ρ) and network tightness (τ). Depending on the value of k_r fast moving slaves will never reach a tightly synchronized state, because they are switching masters faster than the algorithm can compensate the initial offset. On the other hand, a high k_r will enforce the induced errors when in orphaned state and the slave reconnects to the same master.

In contrast to the independence of the movement of slaves, the movement of masters have an impact on the whole PAN created by this master. Therefore, the master election mechanism should select slowly moving nodes as masters, to increase the time of the algorithm to converge the offset towards zero.

The Inter Cluster Synchronisation handles mobility well, since the mobility of a node in the local neighbourhood of P_{c_1} does not change P_{c_1}'s virtual clocks of the adjacent

nodes. Therefore, the transformation of the events is independent of each other. Since there is no hierarchy between the PAN Coordinators, the movement of each node only effects the offset estimation towards its neighbours and the estimation of the neighbours towards this node. New nodes in a neighbourhood start with an infinite uncertainty in the offset estimation since they have not yet established an offset estimation towards their neighbours. While the nodes receive beacons from adjacent masters, they improve the offset estimation and decrease the uncertainty. Consequently, we explicitly specify the offset estimation uncertainty in a time interval $ti = [ts \pm \alpha], ts \in R^+, \alpha \in R^+$ replacing the time stamp $ts \in R^+$. Furthermore, each hop modifies the interval bounds by the currently estimated uncertainty of the synchronisation $\alpha_{r,s}$ as described by (11):

$$e_n.\alpha_r = e_n.\alpha_s + \alpha_{r,s} \tag{11}$$

E. Estimating the Uncertainty

The estimation of the current uncertainty of the synchronisation of the virtual clocks is difficult. Multiple factors influence the actual uncertainty in the synchronisation, like beacon losses and the current drift of the individual clocks. In our approach the synchronisation error $\epsilon_{r,s,i}$ of synchronisation round i between two adjacent PAN Coordinators P_{c_s} and P_{c_r} is characterized by their offset $\delta_{r,s,i+1}$ at beginning of synchronisation round $i+1$.

Following PCS, see Section III-E, we model the synchronisation error to be a zero-mean Gaussian distribution $N(0,\sigma)_{r,s}$. To estimate the standard deviation we use the synchronisation errors of the previous n synchronisation rounds as sample set $E_{r,s} = \{\epsilon_{i-n}, \epsilon_{i-n+1} \dots \epsilon_i\}$. We estimate the standard deviation $\sigma_{r,s}$ of our zero-mean Gaussian based on the sample set. Based on this we compute the confidence interval $\left[\bar{x} \pm z_{\left(\frac{1+\gamma}{2}\right)} \frac{\sigma}{\sqrt{n}}\right]$ of the synchronisation with typical probability γ. The value $z_{\left(\frac{1+\gamma}{2}\right)}$ represents the $\frac{1+\gamma}{2}$-quantile of the standardised normal distribution. The resulting size of the confidence interval $\alpha_{r,s} = z_{\left(\frac{1+\gamma}{2}\right)} \frac{\sigma}{\sqrt{n}}$ represents our current uncertainty estimation, which is added to current uncertainty of the event's time interval.

The complexity of this computation is only dependent on n, which represents a trade-off between estimation accuracy and memory and computation overhead. The quantile of the standardised normal distribution is a pre-defined constant, characterising the accuracy of the estimation.

F. Compatibility between Time Intervals and Time Stamps

As described in the introduction (Section I), WSN aim not only to distribute the acquired sensor data, but also need to process the data in form of events. To this end, Liebig et al. [23] described a way to combine multiple events even though their time stamps might not be exact. To achieve this they extended a time stamp to a time interval and derived an order relation $<$ for time intervals described in (12). Together with a known uncertainty of the event's time stamp, ordering might be possible even in loosely synchronized systems. However, this transition induced a partial order through the order relation. Consequently, there might be situations in which two events cannot be ordered. This is the case if the intervals of the events' time stamps overlap. The resulting partial order relation is similar to the one used by TSAN.

$$[ts_0 \pm \alpha_0] < [ts_1 \pm \alpha_1] \Leftrightarrow ts_0 + \alpha_0 < ts_1 - \alpha_1 \tag{12}$$

Compared to TSAN III-F, our aim is to provide all operations on time intervals that are available for time stamps. This enables applications to handle time intervals in the same manner as classical time stamps, but with the awareness of the induced uncertainty. Our time-interval algebra $([ts \pm \alpha], \{+, -, \times, \cdot, ()^{-1}, <\})$ is based on interval arithmetic [24] and the proposed partial order relation. The difference to the general interval arithmetic is the definition of the inverse operation $(()^{-1})$ and the ability to scale the time interval by a constant factor $(\cdot)$. Both operations are very useful to combine events. One example of such a composition is the deduction of events containing the speed (e_s) of an object based on events containing the position (e_p^i) of the object, as described by Steup et. al [25]. The computation necessary for such a composition is shown in (13).

$$e_s.speed = \left(e_p^1.pos - e_p^0.pos\right)\left(e_p^1.ts - e_p^0.ts\right)^{-1} \tag{13}$$

This type of computation is typical for cyber-physical systems containing physical processes. In general, these processes may be described by differential equations. As an approximation, we enable the computation of difference quotients if the basic events can be ordered. As defined by the partial order, two events are ordered whenever their time intervals are disjoint. Consequently, the time interval created by the subtraction of their time stamps may never contain zero. As a result we simplified the inverse operation compared to general interval arithmetic to (14).

$$[ts \pm \alpha]^{-1} = \left[ts^{-1} \pm \alpha^{-1}\right] \tag{14}$$

$$ts^{-1} = \frac{1/2}{ts + \alpha} + \frac{1/2}{ts - \alpha} \tag{15}$$

$$\alpha^{-1} = \frac{1/2}{ts - \alpha} - \frac{1/2}{ts + \alpha} \tag{16}$$

The resulting time interval algebra establishes a partially ordered vector space, which is easy to compute even for deeply embedded systems. The additional operations enable the computation of differential equations, which are necessary to describe most physical processes. The transformation back to classic time stamps is easily possible by omitting the uncertainty part of the interval.

V. Evaluation

The uncertainty aware hybrid clock synchronisation system was evaluated with a simulation in the Omnet++ Network Simulator [26] version 4.2 and a small scale WSN composed of six nodes.

A. Simulation Setup

For the simulated evaluation we used the INETMANET network model [27] as well as the MiXiM model [28]. The implementation is distributed over two layers of the ISO/OSI stack. One part is located at layer 5 of the ISO/OSI stack and handles the transformation of time stamps for the Inter Cluster synchronisation. The other is situated at layer 2 to gather high precision time stamps. Both layers are connected through a cross layer communication.

Our evaluation focuses on the Inter Cluster Synchronisation, since our simulation experiments shall investigate the scalability of the approach. The single-hop performance will be evaluated by the real world experiments. We evaluate two main aspects of the Inter Cluster Synchronisation. The first considers the influence of the beacon period on the precision of the synchronisation. This test will provide information on the trade-off between message overhead and synchronisation quality. The second test investigates the influence of the communication topology on the reachable multi-hop precision. It will evaluate the usability of the provided time stamps for smaller and longer routes. All tests used the internal 64 bit `simtime` of Omnet++ as reference for the synchronized clocks to evaluate the synchronisation error. The `simtime` was modified by a randomly initialized drift $\rho \leq 10^{-5}$, to provide a realistic clock for each node. The test considered 1000 randomly created routes between nodes in the network, which were created by an optimal routing algorithm.

Our simulation environment considers beacon losses, created by the collision of transmitted beacons of adjacent coordinators, and the resulting lack of information for the time synchronisation. However, we did not transmit data events in the simulation. This decouples our simulations from the used MAC Algorithm and its parameters. Consequently, the simulations consider an optimal MAC-Algorithm preventing all collision between beacons and events in the network.

Additionally, we did not simulate the interrupt handling time possibly decreasing the tightness of the network, since we estimated this time to be smaller then $1\mu s$ because of our optimization described in Section V-D.

B. Beacon Interval Analysis

The beacon interval analysis considered a rectangular grid of 50 PAN Coordinators. The area in which the nodes were distributed was $5000m$ times $5000m$. We used the 2.4GHz specification of the 802.15.4 standard at channel 11 with a maximum transmission power of $1mW$. The thermal noise was fixed at $110dBm$ and the receiver's sensitivity was set to $-85dBm$. Our simulation sweep started with a BO parameter of 8 up till the maximum allowed value of 14. The resulting beacon interval can be computed by $BI = \frac{16 \cdot 60 S \cdot 2^{\mathrm{BO}}}{SymbolRate}$. The $SymbolRate$ of the 2.4GHz band of $65.2 \cdot 10^3 \frac{S}{s}$ results in beacon intervals from $3.8s$ to $241.2s$.

Figure 10 shows a Box-Whisker plot of the simulation's results. The boxes represent the bounds, where 50% of all values are included. The lines represent the interval containing 75% of all values and remaining data points are included as points. As visible with linear increasing BO values the mean synchronisation error increases exponentially. This is to be expected because the beacon interval also increases exponentially. Additionally, one observes a large standard deviation independent of the hop count. This is caused by the unsynchronized beacons of the individual PAN Coordinators, which might collide and therefore increase the real beacon interval. Furthermore, the data base is better for smaller hop counts, since in the given scenario short routes are much more probable then longer routes.

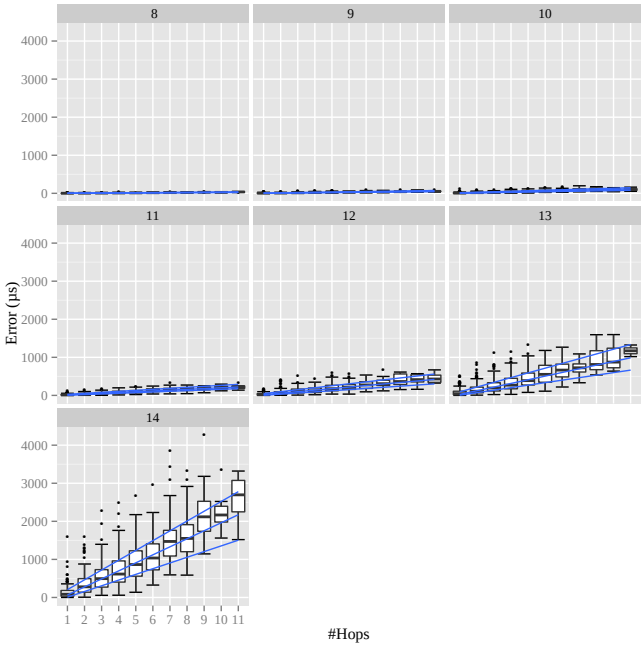


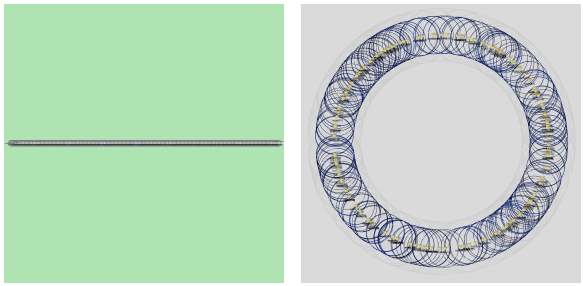
Figure 10. Box-Whisker plot of the precision of a 50 node grid network with varying BO values.

This test proved the expected direct correlation between the beacon interval and the synchronisation precision. Therefore, this value is to be considered critical for the performance of the system. Following our analyses of Section IV-C, the worst case performance should be better then $h2(0.0147 \cdot 2^{\mathrm{BO}} 2.1 \cdot 10^{-11} s$. In all our experiments we never crossed this analytical worst case bound. In the case of 1-hop long routes with a beacon order of 14 the estimated worst case precision is $9.6ms$, where our experiment showed a worst case result of $1.6ms$. For beacon order 13 and a hop count of three we achieved a worst case precision in our experiment of $1.2ms$ and analytically derived a worst case precision of $14.4ms$. We believe these large differences are created by the random drift, which in most cases will not create the largest possible offset.

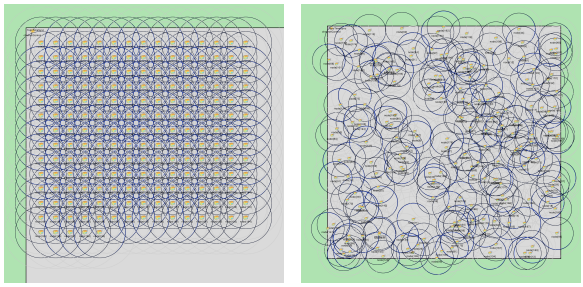
C. Topology Analysis

Our second evaluation considers the performance of the system in different topologies. This is interesting, because topologies might have an influence on the length of the routes, as well as the collision probability of the beacon frames. Therefore, we consider four basic topologies with 200 nodes each. We choose a linear (c.f. Figure 11a), a circular (c.f. Figure 11b), a grid (c.f. Figure 11c) and a randomly generated (c.f. Figure 11d) topology. For this scenario we use the same parameters as for the Beacon Interval Analysis, see Section V-B, but with a static BO parameter of 8.

As visible in Figure 12, the linear topology showed a maximum synchronisation error of $875\mu s$. This is mainly



(a) Linear topology. (b) Circular topology.



(c) Grid topology. (d) Random topology.

Figure 11. Different evaluated topologies with 200 nodes each.

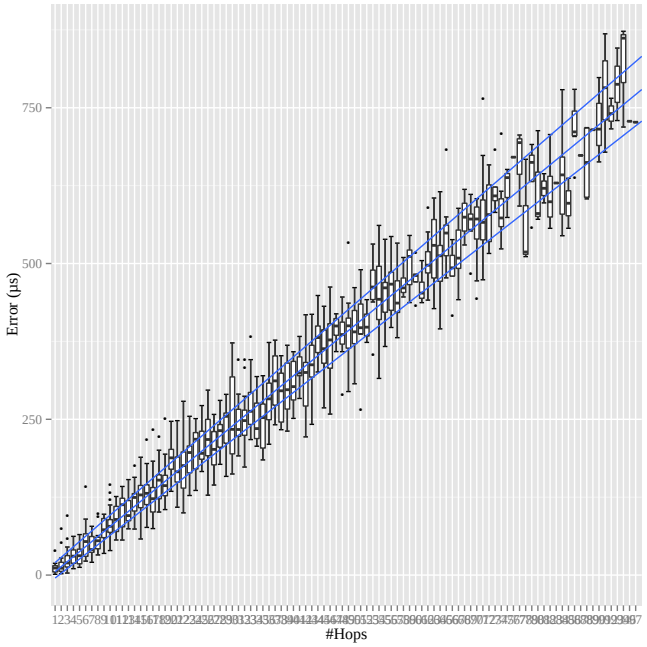


Figure 12. Box-Wisker plot of the synchronisation precision of a 200 Node linear topology.

caused by the extremely long routes created by the evaluation. The maximum hop count was 69 and for each pair of randomly selected nodes there is only one possible route. Consequently, a badly synchronized node will have a large influence on the resulting precision. Considering our worst case analyses we still performed far better than the worst case of $103.8ms$. This is caused by the very small possibility of randomly finding a long route containing 69 nodes with a maximum drift or a large amount of lost beacons.

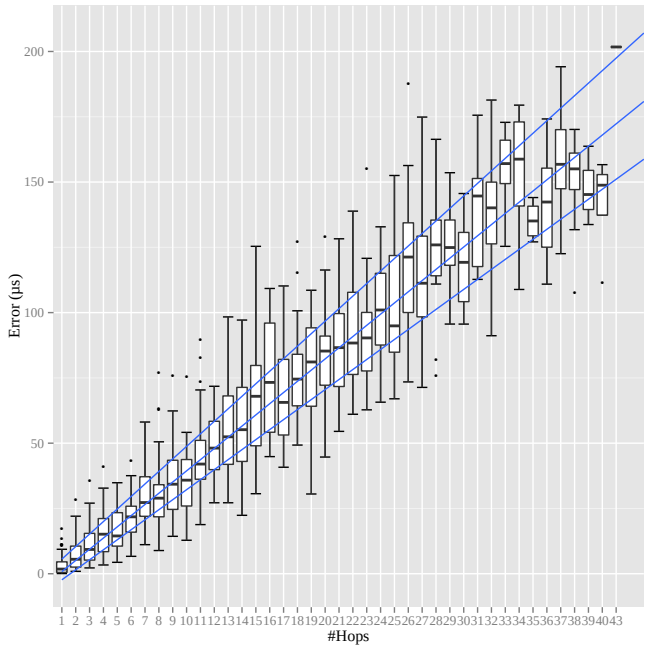


Figure 13. Box-Wisker plot of the synchronisation precision of a 200 Node circle topology.

The circular topology was expected to show similar results like the linear topology, but it performed a lot better, as visible in Figure 13. The maximum hop count was 40 with a typical synchronisation error of $150\mu s$. Compared with the linear topology, which provided a synchronisation error of approximately $500\mu s$, it outperformed the linear topology. This is caused by the larger amount of routes that were possible to connect two randomly selected nodes. A badly synchronized node does not have so much influence anymore, since it is not as likely a part of the random route. Additionally, collisions of beacons are quite unlikely, because only a small number of nodes are in the vicinity of each other. In this experiment one route of 14 hops had a comparably large synchronisation error of $175\mu s$ which was still smaller than the worst case approximation of $2107\mu s$.

The grid topologies (Figure 14) showed slightly worse performance compared to the circle topology. This is caused by the larger probability of beacon collisions caused by a higher density of nodes. At the same time the maximum hop count is only 25, which created a maximum synchronisation error of approximately $150\mu s$. The circle topology showed only a synchronisation error of approximately $100\mu s$ for routes of the same length.

In Figure 15, the performance of the random topology is visible. It shows a large deviation of individual results, which is caused by individual collisions of beacons. This problem is very dependent on the local setup of nodes around a PAN. Therefore, it is very hard to estimate and may only be observed on runtime by an uncertainty evaluation. Such hot spots are also quite likely to be contained in a randomly generated route, since hot spots contain more nodes compared to the

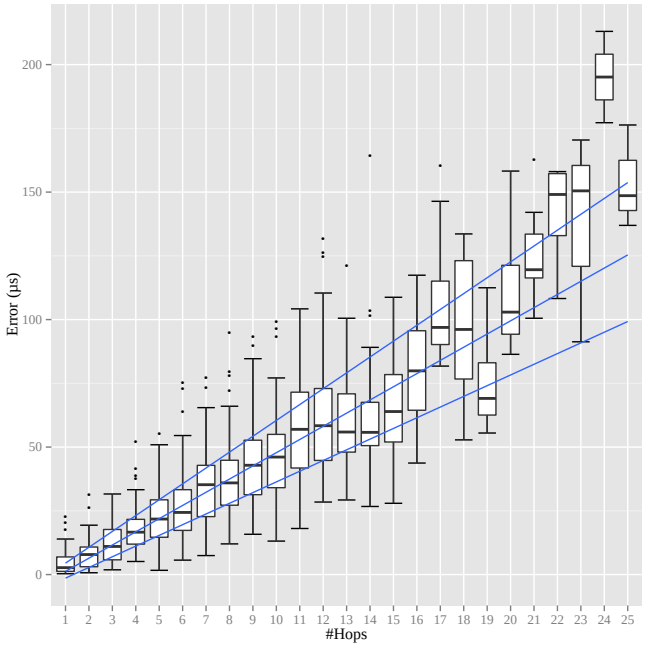


Figure 14. Box-Wisker plot of the synchronisation precision of a 200 Node grid topology.

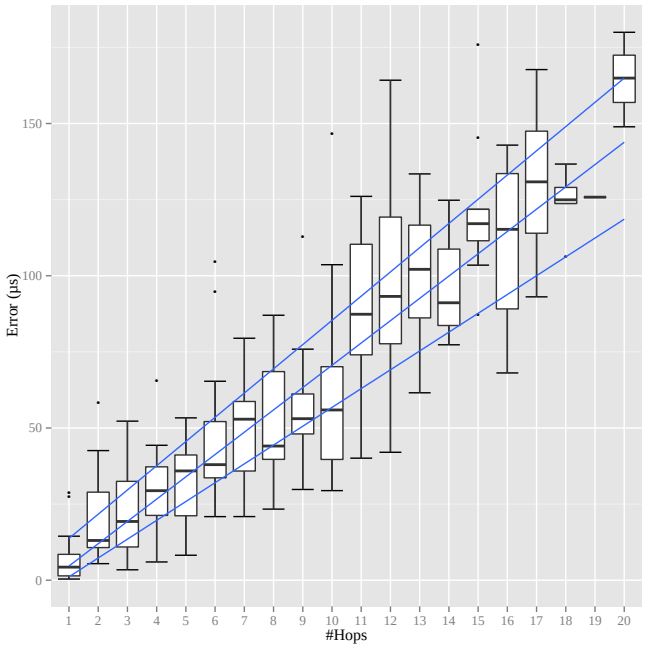


Figure 15. Box-Wisker plot of the synchronisation precision of a 200 Node random topology.

surrounding areas. These facts decrease the achievable mean precision. In this experiment a route of length 12 showed a large error of $161\mu s$ compared to the main quantile of the 12 hop long routes. The worst case analyses predicted a maximum error of $1806\mu s$.

TABLE II. Synchronisation error of the simulation of different topologies in μs.

#Hop Count	Topology	Mean	Standard Deviation
1	Random	7.069008	8.495
1	Grid	5.067219	5.454
1	Linear	12.181786	9.492
1	Circle	3.327730	3.993
6	Random	47.401797	23.945
6	Grid	26.574773	13.489
6	Linear	55.861373	31.209
6	Circle	21.579193	8.327
11	Random	89.191069	27.941
11	Grid	57.463772	19.879
11	Linear	91.408097	23.836
11	Circle	45.693382	15.802
16	Random	110.255113	29.990
16	Grid	80.882388	20.756
16	Linear	131.582874	31.220
16	Circle	73.628941	21.878

Table II shows an overview of the results of our evaluation of the different topologies. This table shows that the performance of the individual topologies towards the mean for each hop is $\pm 50\%$. However, the results for the standard deviation of the tests for equally long routes show a larger difference between the individual topologies. Additionally, the value of the standard deviation is for all single-hop routes approximately the same as the mean error. This clearly indicates a need for a runtime uncertainty estimation, as described in Section IV-E. All topologies supported our mathematical analyses of the worst case synchronisation error since no experiment exceeded the worst case.

As expected, in all simulations the linear topologies have the largest mean error, which is cause by the highest collision probability of the beacons. The random topology performed the second worst, which is caused by local hot spots in the topology with a lot of nodes increasing the probability of beacon losses. All topologies clearly showed a linear relation between the mean error and the hop count. This is to be expected, since the synchronisation error in the vicinity of each PAN is statistically the same within each topology. The summation of the uncertainties matches very well with the increasing error in the simulation. The beacon loss probability is network specific and does not only depend on the topology, but also on the density of nodes. This probability together with the mean length of routes in the network is the major influence towards the synchronisation precision.

From these experiments, we concluded that the basic assumptions were valid. Additionally, we observed that regular non-linear topologies provide better synchronisation results. As a next step, we want to evaluate the performance of the approach on real hardware.

D. Small Scale Wireless Sensor Network Setup

To evaluate the correctness of our assumptions in the simulation we choose a small wireless network of 6 nodes to compare the results with the simulation and related work. Our test network is composed of three PAN Coordinators, two Slaves and a Raspberry Pi Model A [29]. The nodes are

Cortex-M3 based devices from dresden elektronik [30] using a 2.4 GHz 802.14.5 transceiver of Atmel [31]. They run with an internal crystal oscillator and a PLL providing a clock speed of 32 MHz, which is divided by 32 to provide an internal clock with a granularity $g = 1\mu s$. The used $18.432MHz$ oscillator has a drift of $\rho = 3{\cdot}10^{-5}$. We implemented our approach using hardware timers of the Cortex-M3 microcontroller taking a time stamp on each interrupt in hardware. Using this approach the interrupt delay between reception of the packet and the generation of the time stamp should be decreased below $1\mu s$. The propagation time of the wireless signal in the typical range of the AT86RF232 is $t \leq 34ns$. The time between the reception of a packet and the generation of the interrupt by the transceiver is given as $t_{irq} = 9\mu s$. Because the interrupt latency of the transceiver is the same for sender and receiver and, according to data sheet, is constant, it may be omitted. The resulting tightness of our beacon network can be assumed to be $\tau \approx 1\mu s$.

The Raspberry Pi uses a Preempt-RT patched Linux kernel [32] with a real-time enabled listening program. The coordinators and the slaves use the Atmel MAC stack [33] to handle time stamp generation, association and beacon transmission.

Both slaves and the three coordinators are connected to the Raspberry Pi through a GPIO cable as visible in Figure 16. Additionally, each device is connected to an evaluation PC to log the time stamps. Each coordinator establishes its own PAN, but also receives the neighbouring PAN's beacons.

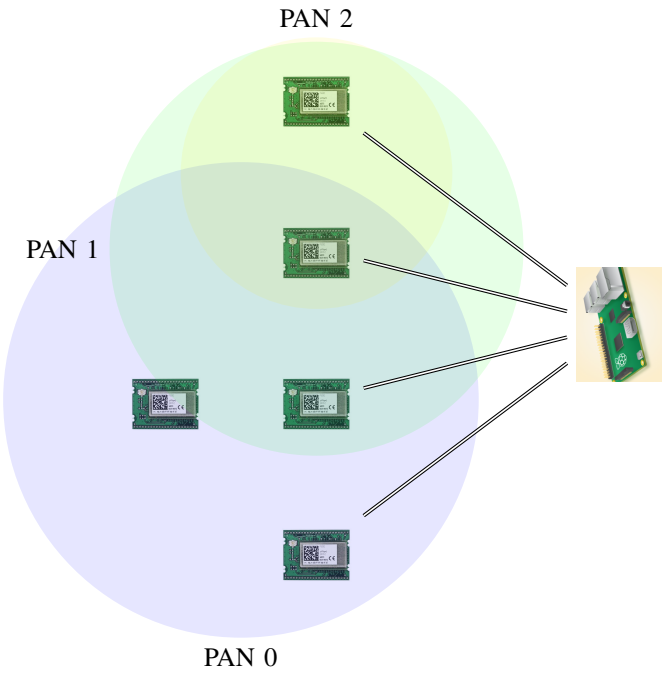


Figure 16. Small scale Wireless Sensor Network composed of dresden elektronik nodes [30] and a Raspberry Pi [29] used to evaluate the hybrid synchronisation.

Whenever a PAN Coordinator transmits its beacon it also logs its internal time to the evaluation PC and toggles its GPIO. On reception of a beacon each node transmits its virtual clock time stamp following the sender to the evaluation PC and toggle their GPIO-Pin. The Raspberry Pi monitors continuously the GPIO-Pins and takes a time stamp on each change. The resulting pair of Raspberry Pi and Cortex-M3 time stamps are correlated and analysed to provide an accurate offset estimation of the virtual clocks against the internal clocks of the nodes. Since beacons are transmitted unsynchronized on the different coordinators, we use linear interpolation to compute time stamps in between measured values.

The benefit of this setup is the minimal critical path, which is established by the GPIO connection. The toggling of the pin on the device is instantaneous. The available low-level access library of the Raspberry Pi provides extremely small latencies accessing the pins. Together with the used real-time program the measurement error should be smaller than $\approx 1\mu s$.

E. Single-Hop Synchronisation

This setup evaluated the single-hop synchronisation mechanism. It is used as a baseline to verify the correctness of the parameters of the simulation. Therefore, the results should be close to the one hop results of the simulation. All coordinators were running and transmitting beacons, but only the two slaves ran the virtual clock towards their PAN Coordinator. The experiment was run for 5 minutes with a beacon interval of $7.5s$ and $15s$ representing a BO value of 9 and 10, respectively.

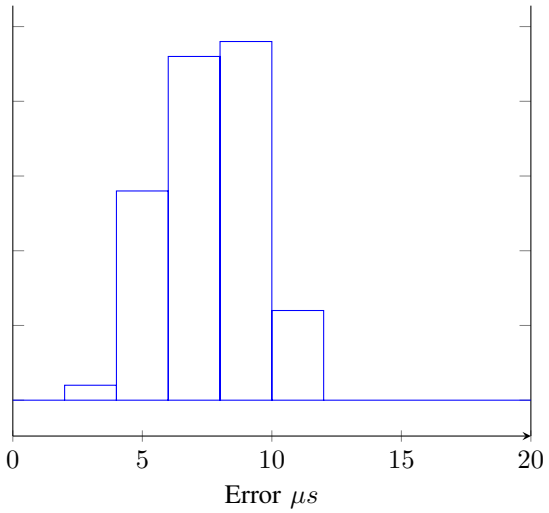


Figure 17. Histogram of the achieved precision of the internal synchronisation with a beacon interval of 7.5s.

The mean precision of the internal synchronisation with a beacon interval of $7.5s$, as visible in Figure 17, was approximately $8\mu s$, which fits very good to the simulated results. The deviation was approximately $\pm 1\mu s$ with the maximum error being $13\mu s$. This was less than the deviation measured in the simulation. An explanation is the smaller network size creating less beacon loss. The distribution of the values is approximately normal distributed, fitting to our assumption used in the uncertainty estimation.

Figure 18 shows the results of the experiment with a beacon interval of $15s$. As visible the mean precision was also $8\mu s$ like in the previous experiment. The deviation of the precision is bigger being approximately $2\mu s$ with the maximum being

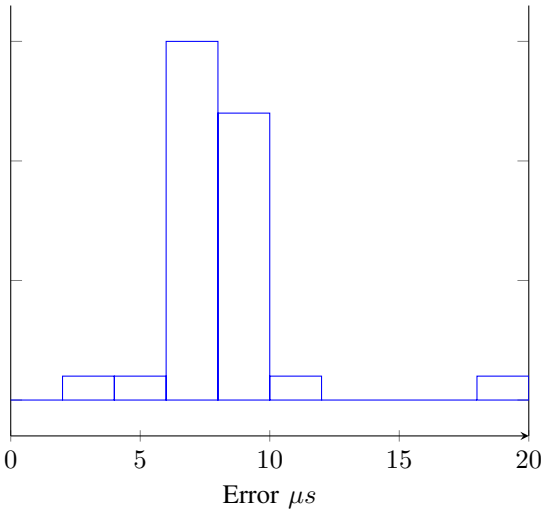


Figure 18. Plot of the achieved results of the single-hop synchronisation with a beacon interval of 15s.

$20\mu s$. This is again better then then simulation results, possibly because of less beacon loss and the crystal oscillator being better than described in the data sheet.

F. Multi-Hop Synchronisation

The evaluation of our Inter Cluster Synchronisation used only the three coordinators. All coordinators were broadcasting beacons in this setup and ran a virtual clock for each neighbouring node. We evaluate the virtual clock values of the different nodes and the time stamps of the internal clocks of the nodes at periodic intervals. The evaluation interval is the same as the beacon interval. No real event was routed through the network, since the used MAC stack provides no means of multi-hop communication. Therefore, we simply added the estimated offsets of the virtual clocks of the two pairs of nodes and compared them to the offsets of the internal clocks of the first and the last coordinator. This scenario used a fixed beacon interval of $7.5s$.

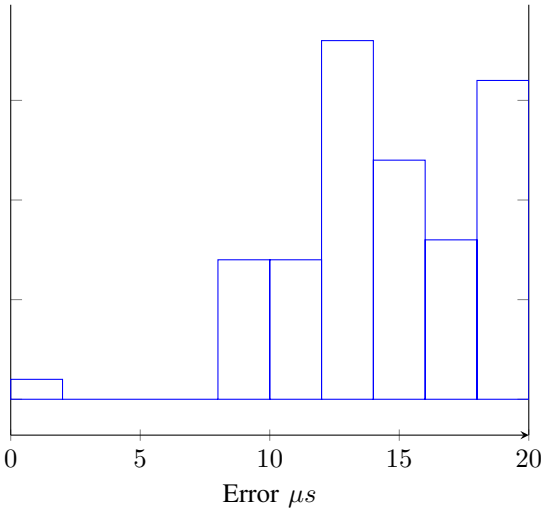


Figure 19. Plot of the achieved results of the inter cluster synchronisation with a beacon interval of 7.5s.

As visible the mean precision was $16\mu s$ with a large deviation of $\pm 5\mu s$ and a maximum offset of $20\mu s$. This result fits very well with the expected simulated and expected results. The error is approximately doubled compared to the single-hop scenario. Additionally, the deviation has increased by the same margin.

The results of our small scale wireless sensor network experiments fully support our simulated and expected performance of our approach. The baseline performance fitted very well with the exception of the larger beacon interval, which strangely showed nearly the same results.

G. Comparison with related protocols

Since the environments of the different described protocols differ, we compare our approach to protocols, with available multi-hop synchronisation data. DMTS provided a mean synchronisation error of $32\mu s$ for one hop and $46\mu s$ for two hop communication. Our approach performed better for single-hop ($8\mu s$) and two-hop synchronisation ($16\mu s$). In comparison to DMTS we do not need a possibly incorrect model of the latency induced by the interrupt handling of the nodes. Additionally, the granularity of the internal clocks of our nodes was far better ($g = 1\mu s$) than the ones used in the experiments evaluating DMTS ($g = 32\mu s$). Therefore, the results are not directly comparable. TSAN showed a mean synchronisation error of $200\mu s$ for one hop and $1113\mu s$ for six hop communication. Our approach performed better in both cases (on average $8\mu s$ and $25 - 37\mu s$ worst case $27\mu s$ and $110\mu s$). However, Römer et al. considered an unstructured abstract network, whereas we exploited the structure and the hardware of the network to increase the synchronisation precision without message overhead. Especially, the periodicity of the beacons enabled continuous synchronisation, which was not available to Römer's system. Mock et al. showed a single-hop synchronisation of approximately $150\mu s$, which was mainly caused by the driver abstraction and the interrupt handling of the used operating system, since they considered an experimentally derived tightness of the network $\tau = 46\mu s$. Our performance stems from the bare-metal implementation and the good local clock increasing the networks tightness to $\tau \approx 1\mu s$.

VI. Conclusion

This paper presents and evaluates a novel hybrid clock synchronisation approach that provides tight synchronisation for local clusters of nodes as well as looser synchronisation in multi-hop scenarios. The message overhead is minimal since existing periodic beacon messages of the 802.15.4 beacon-enabled mode are used to transmit the synchronisation data. To handle the different synchronisation precisions, uncertainty awareness is added to enable applications to decide in the case of ambiguity.

The evaluation is done using the well-established network simulator Omnet++ and a small scale wireless sensor network verifying the results and parameters of the simulation. The results of both experiments match very well with the theoretical concepts and the expected performance of the system. The system provides a baseline performance of $8\mu s$ with a deviation of $\pm 2\mu s$ in a single-hop environment in simulation and real test. In multi-hop scenarios a linearly decreasing precision can be observed depending on the hop count. The results show

the large difference between worst case approximation and real performance together with the large deviation between individual synchronisation results. This clearly indicates the need for uncertainty awareness in the delivered time stamps.

In future work, we want to evaluate the clock synchronisation in a real scenario with more realistic wireless sensor networks to evaluate the influence of unforeseen interference especially in non-regular topologies. Additionally, we want to investigate the effect of different MAC-Algorithms on the synchronisation quality.

ACKNOWLEDGEMENTS

We thank our students Anita Hrubos and David Bodnar for their effort in implementing, testing and evaluating the protocol on our microcontrollers. Additionally, we want to express our gratitude towards Andy Breuhan, who implemented and evaluated the approach in the OMNET++ network simulator.

REFERENCES

[1] C. Steup, S. Zug, J. Kaiser, and A. Breuhan, "Uncertainty Aware Hybrid Clock Synchronisation in Wireless Sensor Networks," in The 8th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM), IARIA. Rome, Italy: Curran Associates Inc., Aug. 2014, pp. 246–251.

[2] Y. Abe, "Safecast or the Production of Collective Intelligence on Radiation Risks after 3.11," in The Asia-Pacific Journal, vol. 12, Issue 7, No. 5, Feb. 2014, pp. 1–6.

[3] L. Yu, N. Wang, and X. Meng, "Real-time forest fire detection with wireless sensor networks," in Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing, vol. 2, Sep. 2005, pp. 1214–1217.

[4] F. Michahelles, P. Matter, A. Schmidt, and B. Schiele, "Applying wearable sensors to avalanche rescue," Computers & Graphics, vol. 27, no. 6, Dec. 2003, pp. 839–847.

[5] K. Römer and F. Mattern, "The design space of wireless sensor networks," IEEE Wireless Communications, vol. 11, no. 6, Dec. 2004, pp. 54–61.

[6] P. Veríssimo, Paulo and L. Rodrigues, Distributed Systems for System Architects, 1st ed. Kluwer Academic Publishers, 2001, ch. 2.

[7] P. Hetzel, "Time dissemination via the LF transmitter DCF77 using a pseudo-random phase-shift keying of the carrier," in Proceedings of the 2nd European Frequency and Time Forum (EFTF), 1988, pp. 351–364.

[8] P. Dana, "Global Positioning System (GPS) Time Dissemination for Real-Time Applications," Real-Time Systems, vol. 12, no. 1, 1997, pp. 9–40.

[9] H. Kopetz, Real-Time Systems: Design Principles for Distributed Embedded Applications, 2nd ed. Springer Science+Business Media, 2011, ch. 3.

[10] J. Burbank, D. Mills, and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification," Internet Engineering Task Force (IETF), Tech. Rep. RFC 5905, Jun. 2010.

[11] "IEEE Standard for a Precision Clock Synchronisation Protocol for Networked Measurement and Control Systems," IEEE, Tech. Rep. Standard 1588-2002, 2002.

[12] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronisation using reference broadcasts," in SIGOPS, vol. 36, no. SI. New York, NY, USA: ACM, Dec. 2002, pp. 147–163.

[13] P. Verissimo, L. Rodrigues, and A. Casimiro, "Cesiumspray: a precise and accurate global time service for large-scale systems," Real-Time Systems, vol. 12, no. 3, 1997, pp. 243–294.

[14] S. Ping, "Delay measurement time synchronisation for wireless sensor networks," Intel Research Berkeley Lab, Tech. Rep. IRB-TR-03-013, 2003.

[15] M. Mock, R. Frings, E. Nett, and S. Trikaliotis, "Continuous clock synchronisation in wireless real-time applications," in Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems, Oct. 2000, pp. 125–132.

[16] "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," IEEE, Tech. Rep. IEEE Standard 802.11, 1997.

[17] M. Gergeleit and H. Streich, "Implementing a distributed high-resolution real-time clock using the CAN-bus," in Proceedings of the 1st international CAN-Conference. Mainz, Germany: Can-in-Automation (CIA), 1994.

[18] S. PalChaudhuri, A. Saha, and D. B. Johnson, "Probabilistic clock synchronisation service in sensor networks," in IEEE Transactions on Networking, vol. 2, no. 2, 2003, pp. 177–189.

[19] K. Römer, "Time synchronisation in ad hoc networks," in Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing, ser. MobiHoc '01. New York, NY, USA: ACM, 2001, pp. 173–182.

[20] F. Cristian, "Probabilistic clock synchronisation," in Distributed Computing, vol. 3, no. 3. Springer, Sep. 1989, pp. 146–158.

[21] "IEEE Standard for Information Technology- Telecommunications and Information Exchange Between Systems- Local and Metropolitan Area Networks- Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)," Tech. Rep. IEEE Standard 802.15.4, 2006.

[22] Bluetooth SIG, "Specification of the Bluetooth System," Tech. Rep. 4.0, June 2010.

[23] C. Liebig, M. Cilia, and A. Buchmann, "Event Composition in Time-Dependent Distributed Systems," in Proceedings of the Fourth IECIS International Conference on Cooperative Information Systems, ser. COOPIS '99. Washington, DC, USA: IEEE Computer Society, Sep. 1999, pp. 70–78.

[24] R. E. Moore, R. B. Kaerfott, and M. J. Cloud, "The Interval Number System," in Introduction to Interval Analysis, 1st ed. Society for Industrial and Applied Mathematics, Jan. 2009, pp. 7–18.

[25] C. Steup, S. Zug, and J. Kaiser, "Achieving Cooperative Sensing in Automotive Scenarios through Complex Event Processing," in The 7th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM), IARIA. Porto, Portugal: Curran Associates Inc., Sep. 2013, pp. 26–30.

[26] G. Pongor, "OMNeT: objective modular network testbed," in Proceedings of the International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems. San Diego, CA, USA: Society for Computer Simulation International, Oct. 1993, p. 323–326.

[27] A. Ariza and A. Triviño, Simulation of Multihop Wireless Networks in OMNeT++. IGI Global, 2012, pp. 140–158.

[28] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. T. K. Haneveld, T. E. V. Parker, O. W. Visser, H. S. Lichte, and S. Valentin, "Simulating Wireless and Mobile Networks in OMNeT++ the MiXiM Vision," in Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops. Brussels, Belgium: Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, Mar. 2008, p. 71:1–71:8.

[29] Raspberry Pi Foundation, "Rapsberry Pi Model A+," at 10.03.15. [Online]. Available: http://www.raspberrypi.org/products/model-a-plus/

[30] dresden electronik, "deRFsam3-23M10," at 10.03.15. [Online]. Available: http://www.dresden-elektronik.de/funktechnik/products/radio-modules/oem-modules-derfsam3

[31] A. Corporation, "Low Power,2.4GHz Transceiver for ZigBee, IEEE 802.15.4, 6LoWPAN, RF4CE and ISM Applications," Tech. Rep. 8321, oct 2011.

[32] L. Fu and R. Schwebel, "RT PREEMPT HOWTO," at 10.03.15. [Online]. Available: https://rt.wiki.kernel.org/index.php/RT_PREEMPT_HOWTO

[33] Atmel Cooperation, "IEEE 802.15.4 MAC Software Package - User Guide," Tech. Rep. AVR2025, Jun. 2012. [Online]. Available: http://www.atmel.com/Images/doc8412.pdf