Technische Universität Braunschweig

Institute of Operating Systems and Computer Networks

A Sub-Microsecond Clock Synchronization Protocol for Wireless Industrial Monitoring and Control Networks

Georg von Zengen and Keno Garlichs and Yannic Schröder and Lars C Wolf

Authors post-print published on 2017-06-09
Originally published in *2017 IEEE International Conference on Industrial Technology (ICIT) Special Sessions*
Publisher version available at http://ieeexplore.ieee.org/document/7524540/?arnumber=7524539
DOI: 10.1109/ICIT.2017.7915545

Abstract:

Wireless networks are getting more and more important to industrial process monitoring in control. One of the key challenges in these domains is clock synchronization. Needed to be able to link data gathered on different devices in the network. More recently the idea of wireless closed loop controllers rose. With that even more precise clock synchronization is needed. In this paper we propose a protocol capable of sub-microsecond synchronization and clock drift compensation. The proposed protocol is inspired by the widely used PTP but was optimized for wireless networks. Our real world evaluation indicates, that clock drift is a serious challenge in networks utilizing modern microcontrollers. We also show that our approach is capable of mitigating that drift to reach a competitive degree of synchronicity.

A Sub-Microsecond Clock Synchronization Protocol for Wireless Industrial Monitoring and Control Networks

Georg von Zengen, Keno Garlichs, Yannic Schröder, Lars C. Wolf
Institute of Operating Systems and Computer Networks
Technische Universität Braunschweig
Email: [vonzengen, garlichs, schroeder, wolf]@ibr.cs.tu-bs.de

***Abstract*—Wireless networks are getting more and more important to industrial process monitoring in control. One of the key challenges in these domains is clock synchronization. Needed to be able to link data gathered on different devices in the network. More recently the idea of wireless closed loop controllers rose. With that even more precise clock synchronization is needed. In this paper we propose a protocol capable of sub-microsecond synchronization and clock drift compensation. The proposed protocol is inspired by the widely used PTP but was optimized for wireless networks. Our real world evaluation indicates, that clock drift is a serious challenge in networks utilizing modern microcontrollers. We also show that our approach is capable of mitigating that drift to reach a competitive degree of synchronicity.**

I. Introduction

Monitoring got more important in all industrial processes over the last years. The reasons to monitor the processes are manifold and reach from quality assurance to processes optimization. Timestamps are the link between different measurements and therefore they are essential to put measurements of different sensors in relation to each other. The synchronized time is also needed to determine at which time an event took place.

In controlling, a synchronized time source is even more important and the synchronization must be more precise. Without it, different machines are not able to cooperate and may destroy their work pieces. Clock synchronization needs to be even more precise when closed loop controllers are spread on different networked machines. To allow such systems, sub-microsecond clock synchronization is mandatory.

To tackle the challenge of synchronizing the clocks of different networked machines, different protocols were designed in the past. One of the most widely used ones is the Network Time Protocol (NTP) [1]. It is used all over the Internet and based on an hierarchical structure of time servers where each server receives the reference time from its parent and publishes it to its children. Due to this structure it scales well but the precision of the synchronization suffers on every hierarchical layer of the structure. Therefore, Precision Time Protocol (PTP) [2] was developed to provide a more precise synchronization for smaller non hierarchical networks. In wired networks PTP reaches a synchronization precision of up to $60\,ns$.

In recent years the idea of wireless monitoring in factories rose to prominence [3], [4], [5], [6]. Due to this trend, clock synchronization needs to be able to work in wireless network as well. The most recent ideas go even further, i.e. to wireless networked closed loop controllers.

That led to the development of wireless sub-microsecond synchronization protocols [7], [8], [9]. Most of these Protocols assume the hardware clock of the devices to be accurate in frequency. Thus, many of them do not handle clock drifts between different network nodes at all [7], [9] The assumption of accurate hardware clocks can not be held for modern microcontrollers like the STM32 series by ST Microelectronics [10]. These controllers often have an internal Phase Locked Loop (PLL) to multiply the frequency of an external oscillator and use this signal to clock their CPU. This is done to achieve higher frequency with cheap oscillators. Another reason is the ability to change the frequency during operation. The challenge introduced by cheap oscillators is the higher error they have in their frequency. This error leads to a rapid drifting of the clocks of the devices in one network.

In this paper we present a PTP-based protocol optimized for wireless networks. We clearly focus on the accuracy of the clock synchronization to enable wireless closed loop controllers in process and factory automation. To give the reader a background, we briefly describe PTP and other related work in Section II. In Section III we introduce the optimizations and adjustments we made to PTP. We also propose a method to compensate the drift of clocks introduced by cheap crystal oscillators. After the description of our approach to wireless sub-microsecond clock synchronization we discuss the evaluation and its results in Section IV. These results show the competitiveness of our approach even with relatively inaccurate and cheap crystal oscillators. Section V concludes this paper and gives an outlook to our future work.

II. Related Work

In this section we give a brief overview of sub-microsecond synchronization protocols for wireless networks. Further we describe PTP briefly to give the reader a background for the remainder of this paper.

A. Precision Time Protocol

PTP is a time synchronization protocol aiming at Local Area Networks (LANs) and providing microsecond accuracy, other than NTP which is used in the Internet and provides millisecond accuracy. It is standardized as IEEE 1588/IEC 61588 [2]. Although the protocol does not limit the medium to Ethernet, almost all implementations use it. But there are also working implementations using IEEE 802.11 for example [11]. When the protocol is started, the Best Master Clock (BMC) algorithm is started. Every node announces the capabilities of its clock in order to choose the best one among them as the master clock.

The messages exchanged to synchronize the clocks between the master and its slaves are depicted in Figure 1. The master occasionally multicasts a `SYNC` message to each slaves. The slave then records the reception timestamp t_1 as early and close to the hardware as possible. To do this, either a hardware timestamping unit inside the Network Interface Controller (NIC) can be used if available, or this can be done in software in the Medium Access Control (MAC) layer. The master has to prepare the message and write the actual timestamp into the `SYNC` message at that time. Due to processing time, medium access, and propagation delay, sending timestamp t_0 is not precise. To mitigate this issue, a `FOLLOW_UP` message is sent afterwards. It contains the exact sending time as it has been recorded by the hardware or, if not available, by the Interrupt Service Routine (ISR) processing the TX-complete interrupt. The choice between hardware timestamping (if available) and software timestamping has a large impact on the possible precision. According to [12], software only solutions reach a precision of 5 to $50\,\mu s$, while hardware supported implementations reach a typical precision of $\pm 60\,ns$. Having now acquired both timestamps t_0 and t_1, the slave can calculate its offset to the master. Yet, this has the error of the propagation delay τ_{prop} which has to be subtracted in order to get the correct value. To determine τ_{prop}, the slave sends a `DELAY_REQUEST` message to the master. This answers with the reception timestamp t_3 in the `DELAY_RESPONSE`. Now knowing all the relevant information and assuming a symmetric delay, the slave calculates its propagation delay to the master. Both Ω and τ_{prop} are noted and the slave can correct its clock to be in sync with the master. Where Ω is the offset of the slave's clock.

B. Glossy

Glossy is an efficient network flooding protocol proposed by Ferrari et al. [9]. Due to its special flooding mechanisms it also brings a synchronization protocol. The idea in Glossy is to forward certain packets as fast as possible without further processing. Thus, the time a node needs to forward a packet is constant and the error added in a multihop network can be predicted. Another advantage of constant packet processing times is the ability to transmit the packets by different nodes at the same time. This is only true if all nodes transmit the same data. Due to this ability there is no need for a MAC for this packet, this eliminates the delay introduced by clear channel assessments. Ferrari et al. do not propose a protocol for clock synchronization but an implicit synchronization based on packets transmitted for other reasons. In their evaluation Glossy was able to synchronize a network with 8 hops with a mean signed deviation of $0.4\,\mu s$.

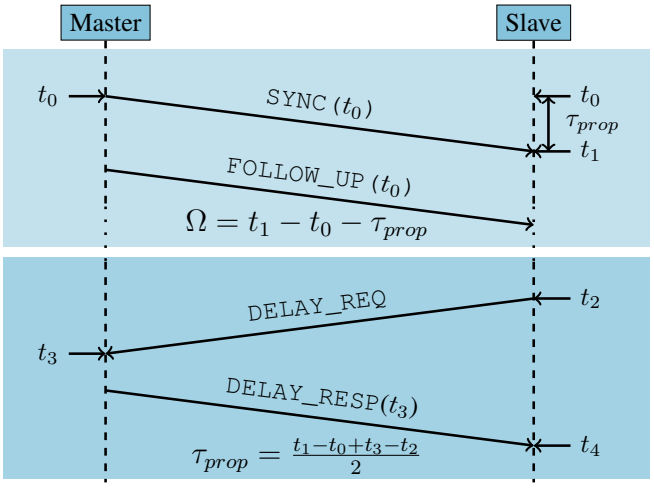

Figure 1. Messages exchanged in PTP

Although Glossy seems to be an elegant approach to clock synchronization in wireless networks it suffers from some conceptional disadvantages. Due to the elimination of clear channel assessments it is not able to coexist with other network in the same area and frequency. Another disadvantage is the need of precise crystal oscillators to meet $0.5\,\mu s$ slot with all nodes that should transmit the same packet.

C. TPSN

TPSN [13] was one a the first time synchronization protocols designed for wireless networks. It works in two phases: in a first the fixed hierarchical structures is build up across the whole network. In the second phase a two-message protocol is used to synchronize the nodes pairwise among the edges. Thus, the synchronization propagates through the network. In comparison to Glossy, TPSN uses unicast messages, not broadcast, to synchronize the pairs. The authors state that TPSN synchronizes a network with a precision of $20\,\mu s$, which does not meet the sub-microsecond barrier.

III. Clock Synchronization

In this section we describe our optimization to PTP and the design decisions that led to them in detail. As our main goal was to meet the requirements of wireless networked closed loop controllers, we focused on the synchronization accuracy of our protocol. To do so, we had to put other possible goals out of focus. In our case we decided to assume multihop capabilities as less important. The main reason was the transmission range of our transceiver (the DW1000 [14]) of up to $250\,m$. It is quite unlikely that a closed loop

controller is spread over more than $250\,m$ and still needs a sub-microsecond synchronization.

PTP was originally designed to work in wired LANs and supports software timestamping as well as hardware timestamping of packets in the NIC. Since our transceiver supports timestamping of received packages, we adopted PTP to our requirements and thereby achieved an improved performance.

The master occasionally multicasts a SYNC message to its slaves. The slave then records the reception timestamp t_1 utilizing the timestamping support of the DW1000 to achieve the maximum accuracy. The master has to prepare the message and write the actual timestamp into the SYNC message at that time. For traditional NICs the sending timestamp t_0 is not precise. To mitigate this issue, traditional PTP transmits a FOLLOW_UP message that contains the precise transmission timestamp t_0.

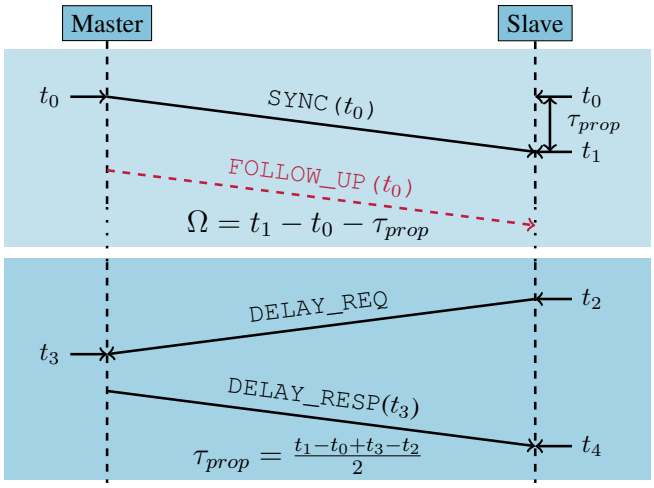

Figure 2. MoRT-Stack's clock synchronization protocol, the red FOLLOW_UP message is removed because it is not necessary due to precise timestamps.

Our adoption is optimized in the way that no FOLLOW_UP message is needed. By utilizing the DW1000 delayed transmission feature, we can already write the exact sending timestamp into the SYNC message. To do so it needs to be ensured that t_0 is far enough in the future to finish all necessary preprocessing of the SYNC message until t_0. This timestamp is precise due to the fact that the clock triggering the transmission is running inside the DW1000 and all computation needs be done beforehand [14]. In Figure 2 the removed FOLLOW_UP message is marked in red and dashed.

Having now acquired both timestamps t_0 and t_1, the slave can calculate it's offset Ω to the master.

$$\Omega = t_1 - t_0 - \tau_{prop} \tag{1}$$

Where τ_{prop} still needs to be determined in the way originally intended in PTP.

A. Master Selection

As in our setup all nodes have the same clock accuracy we needed a different way to select a clock master in our network. Therefore, the nodes start listening for SYNC messages for a certain time. If a node does not receive a SYNC message during that time, it starts transmitting SYNC messages and is therefore the master. If it received a SYNC message it is a slave and uses the information in that message. A conflict resolution algorithm was implemented to assure the correct behavior, even if multiple nodes are started at the same time and therefore all decided to become the new master. This can happen, for example, if a larger machine is powered on and with it all its various network nodes at the same time. The current solution is to just favor the node with the lowest MAC address.

For hardware platforms with an accurate watch crystal the operation frequency could be measured against that crystal to determine the accuracy of operation frequency. That way PTP's Best Master Clock algorithm could be implemented as a more elaborated solution.

B. Drift Compensation

As we show in Section IV our STM32 microcontroller suffers from a low cost crystal oscillator. Thus, the nodes' clocks will drift apart. To mitigate the drift, we designed a drift compensation that does not require any extra communication between the nodes. All synchronization clients measure the time span (ΔT_{Rx}) between the reception of two SYNC messages. Which is the time between two instances of t_1 from Figure 2. Equation (2) gives the this time span with n as a SYNC message interval counter.

$$\Delta T_{Rx} = t_1(n) - t_1(n-1) \tag{2}$$

To calculate the drift to the clock master, the client compares ΔT_{Rx} with the time span between the transmission timestamps (ΔT_{Tx}) of the SYNC messages.

$$\Delta T_{Tx} = t_0(n) - t_0(n-1) \tag{3}$$

Where t_0 is the transmission time stamp as in Figure 2 and n in the SYNC message interval counter. The difference (ΔT) between ΔT_{Rx} and ΔT_{Tx} gives the drift of the two clocks over the SYNC message interval. A ΔT closer to zero means less drift between the clocks.

$$\Delta T = \Delta T_{Rx} - \Delta T_{Tx} \tag{4}$$

IV. Evaluation

In this section we evaluate our clock synchronization protocol. All experiments were performed in the setup shown in Figure 3. For the evaluation $Node0$ was defined as the master to have reproducible results. $Node0$ toggles the states of one of its GPIOs with $1\,Hz$. The GPIO is wired to an interrupt input at $Node1$ and $Node2$. $Node0$ takes a timestamp every time it toggles the state of the GPIO and transmits it via Universal Asynchronous Receiver Transmitter (UART) to a logging PC. $Node1$ and $Node2$ take a timestamp every time

their interrupt is triggered by the GPIO. They also transmit theses timestamps via UART to the PC. All communication needed to synchronize the clocks of the nodes is wireless. We chose a wired GPIO trigger signal to minimize the delay and jitter a wireless triggering might have.

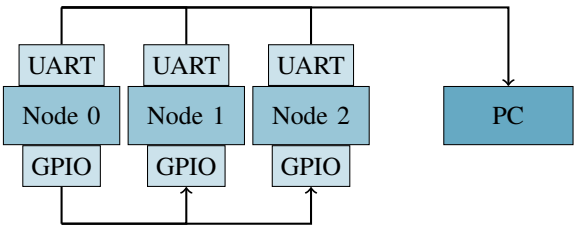

Figure 3. Measurement setup used to evaluate the clock synchronization

A. Ground Truth

The first experiment measures the drift of $Node1$ and $Node2$ relative to $Node0$ without any synchronization. We performed this experiment to have a measurement that shows how the clock behave without synchronization and to be able to evaluate the gain in precision our protocol brings. The results of these experiment are shown in Figure 4. There are two many observations from this experiment: the drifts of the nodes vary from each other but are constant over time. In our experiment we measured a relative clock error of $1.6\,\mu s/s$ for $Node1$ and $2.8\,\mu s/s$ for $Node2$. These results show that

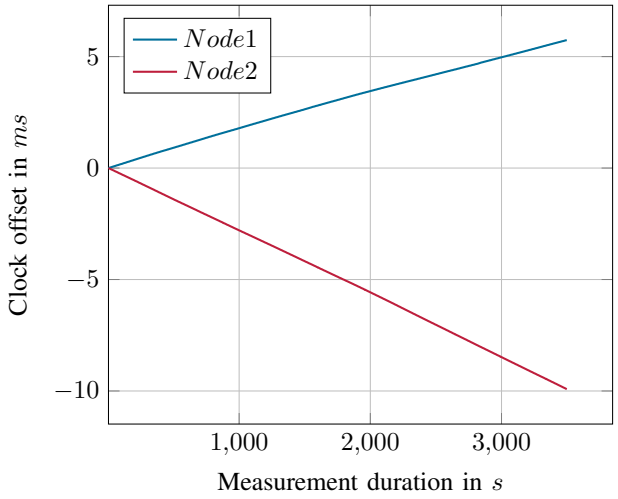

Figure 4. Clock dirft measurement without synchronization. The clock of $Node1$ dirfts by $1.6\,\mu s/s$,the one of $Node2$ by $2.8\,\mu s/s$.

the drift compensation presented in Section III-B is needed to synchronize the clock in a μs range without a SYNC message interval below one second.

B. Clock Synchronization without Drift Compensation

In this experiment we used a synchronization interval of one second. As Figure 5 shows the drift is still notable but it is around zero. The measurements form a saw-tooth like shape. This is due to the fact that every SYNC message brings the clocks in sync and afterwards they are again drifting apart. The overlapping of the saw-tooth signal is due to the low sampling rate of $1\,Hz$. We only show a five minute slice of the half hour experiment to make the figure more clear. For the same reason we left out $Node2$. The mean absolute offset of the synchronization is $0.54\,\mu s$ for $Node1$ and $0.33\,\mu s$ for $Node2$. As the mean absolute offset is below one microsecond our system is able to synchronize drifting clocks in a sub-microsecond manner. The Standard Deviation (SD) of the offset is $0.41\,\mu s$ for $Node1$ and $Node2$. Although these results might look competitive to other clock synchronization protocols [9], they still suffer from the drift of the clocks. That means, tasks near to the SYNC message are better synchronized than tasks more far away from that message.

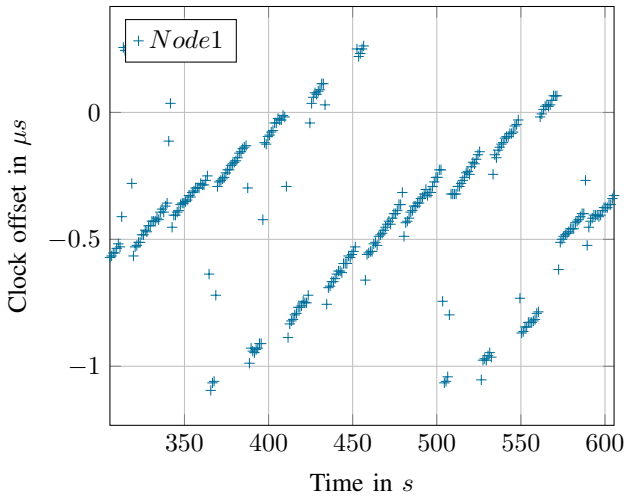

Figure 5. Exemplary five minutes slot of the 30 minute clock drift measurement with clock synchronization. The drifting clocks are still notable by saw-tooth like shape of the measurements

C. Clock Synchronization with Drift Compensation

To overcome the clock drift challenge we proposed a drift compensation in Section III-B In this experiment we evaluate the performance of this compensation. As Figure 6 shows, there is no more slope in the clock offset, therefore the clock drift is compensated by our proposed drift compensation. This result is supported by smaller mean absolute offsets: $0.19\,\mu s$ for $Node1$ and $0.22\,\mu s$ for $Node2$.

V. Conclusion

In this paper we proposed an optimized version of the PTP for wireless networks. We also present a drift compensation for our clock synchronization protocol. In contrast to most other approaches, ours does not rely on accurate clocking of the CPU. The evaluation shows that our approach is able to synchronize clocks up to a accuracy of $0.19\,\mu s$. That shows, our approach is competitive to other approaches even with inaccurate hardware clocks.

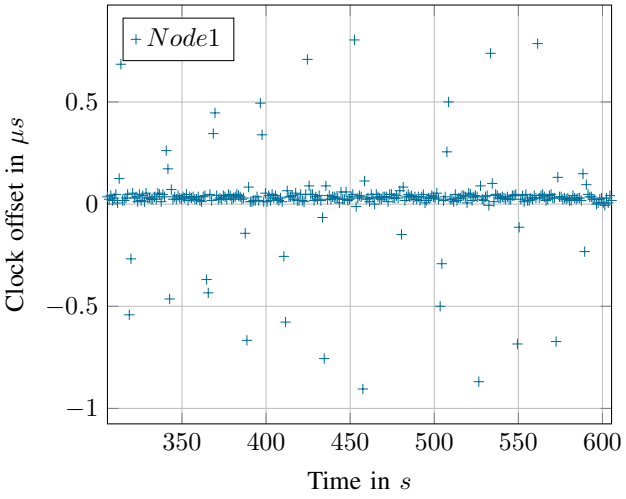

Figure 6. Exemplary five minutes slot of the 30 minutes clock drift measurement with clock synchronization. The drifting clocks are compensated and most measurements lie on a horizontal line near zero.

Although the results of our evaluation are already competitive to other protocols we plan to investigate deeper into the outliers we observed during our evaluation. With these outliers removed we plan to integrate this synchronization protocol in a much bigger network for wireless closed loop controllers.

REFERENCES

[1] D. L. Mills, "Network Time Protocol (NTP)," Internet Requests for Comments, RFC Editor, RFC 958, September 1985. [Online]. Available: http://www.rfc-editor.org/rfc/rfc958.txt

[2] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEC 61588:2009(E)*, pp. C1–274, February 2009.

[3] T. O'donovan, J. Brown, F. Büsching, A. Cardoso, J. Cecílio, J. D. Ó, P. Furtado, P. Gil, A. Jugel, W.-B. Pöttner, U. Roedig, J. S. Silva, R. Silva, C. Sreenan, V. Vassiliou, T. Voigt, L. Wolf, and Z. Zinonos, "The GINSENG System for Wireless Monitoring and Control: Design and Deployment Experiences," *ACM Transactions on Sensor Networks*, vol. 10, no. 1, pp. 4:1–4:40, Dec. 2013. [Online]. Available: http://doi.acm.org/10.1145/2529975

[4] J. Song, S. Han, A. Mok, D. Chen, M. Lucas, and M. Nixon, "WirelessHART: Applying Wireless Technology in Real-Time Industrial Process Control," in *IEEE Real-Time and Embedded Technology and Applications Symposium*, April 2008, pp. 377–386.

[5] R. Steigmann and J. Endresen, *Introduction to WISA*. ABB, July 2006.

[6] ISA100 Standards Committee, "ANSI/ISA-100.11a-2011 Wireless systems for industrial automation: Process control and related applications," 2011.

[7] R. Lim, B. Maag, and L. Thiele, "Time-of-flight aware time synchronization for wireless embedded systems," in *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks*, ser. EWSN '16. USA: Junction Publishing, 2016, pp. 149–158. [Online]. Available: http://dl.acm.org/citation.cfm?id=2893711.2893732

[8] M. Lévesque and D. Tipper, "A survey of clock synchronization over packet-switched networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2926–2947, 2016.

[9] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, April 2011, pp. 73–84.

[10] STMicroelectronics, "RM0090 Reference manual: STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced ARM® -based 32-bit MCUs," *http://www.st.com/web/en/resource/technical/document/reference_manual/DM00031020.pdf last accessed: 24.04.2016*, October 2015.

[11] J. Kannisto, T. Vanhatupa, M. Hännikäinen, and T. D. Hämäläinen, "Precision Time Protocol Prototype on Wireless Lan," in *Telecommunications and Networking-ICT 2004*. Springer, August 2004, pp. 1236–1245.

[12] A. Dreher and D. Mohl, "Präzise Uhrzeitsynchronisation - IEEE 1588 (White Paper)," Hirschmann Automation and Control GmbH, Neckartenzlingen, Germany, Tech. Rep., 2005.

[13] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, 2003, pp. 138–149.

[14] D. Ltd., "DW1000 User Manual - How to use, configure and program the DW1000 UWB transceiver," *http://www.decawave.com/support/download/file/nojs/629 last accessed: 10.11.2015*, 2015.